TEST BENCH CODE:

```verilog
module tb_phase1;

    reg clk = 0, rst = 1, code_in;

    wire status_done, fail;

    phase1 uut(.clk(clk), .rst(rst), .code_in(code_in), .status_done(status_done), .fail(fail));


    always #5 clk = ~clk;


    initial begin
      $dumpfile("wave.vcd");
      $dumpvars(0,tb_phase1);
        $display("Starting Phase 1 Test...");
        #10 rst = 0;
        code_in = 1; #10;
        code_in = 0; #10;
        code_in = 1; #10;
        code_in = 1; #10;
        #10 $display("Done: %b, Fail: %b", status_done, fail);
        $finish;
    end
endmodule

module tb_phase2;

    reg clk = 0, rst = 1;

    reg [3:0] switch_in;

    wire status_done, fail;


    phase2 uut(.clk(clk), .rst(rst), .switch_in(switch_in), .status_done(status_done), .fail(fail));


    always #5 clk = ~clk;
```

```verilog
    initial begin
      $dumpfile("wave.vcd");
     $dumpvars(0,tb_phase2);
       $display("Starting Phase 2 Test...");
       #10 rst = 0;
       switch_in = 4'b1101; #10;
       $display("Done: %b, Fail: %b", status_done, fail);
       #10 switch_in = 4'b1010; #10;
       $display("Recheck - Done: %b, Fail: %b", status_done, fail);
       $finish;
    end
endmodule
module tb_phase3;
    reg clk = 0, rst = 1;
    reg [2:0] dir_in;
    wire status_done, fail;


    phase3 uut(.clk(clk), .rst(rst), .dir_in(dir_in), .status_done(status_done), .fail(fail));


    always #5 clk = ~clk;


    initial begin
      $dumpfile("wave.vcd");
     $dumpvars(0,tb_phase3);
       $display("Starting Phase 3 Test...");
       #10 rst = 0;
       dir_in = 3'b000; #10; // UP
       dir_in = 3'b011; #10; // RIGHT
       dir_in = 3'b001; #10; // DOWN
       dir_in = 3'b010; #10; // LEFT
       dir_in = 3'b000; #10; // UP
```

```verilog
            $display("Done: %b, Fail: %b", status_done, fail);
            $finish;
        end
endmodule


module tb_phase4;
    reg clk = 0, rst = 1;
    reg [7:0] plate_in;
    wire status_done, fail;


    phase4 uut(.clk(clk), .rst(rst), .plate_in(plate_in), .status_done(status_done), .fail(fail));


    always #5 clk = ~clk;


    initial begin
        $dumpfile("wave.vcd");
        $dumpvars(0,tb_phase4);
            $display("Starting Phase 4 Test...");
            #10 rst = 0;
            plate_in = 8'b10101010; #10;
            plate_in = 8'b11001100; #10;
            plate_in = 8'b11110000; #10;
            $display("Done: %b, Fail: %b", status_done, fail);
            $finish;
        end
endmodule
module tb_phase5;
    reg clk = 0, rst = 1;
    wire [1:0] time_lock_out;
    wire status_done, fail;
```

```verilog
    phase5 uut(.clk(clk), .rst(rst), .time_lock_out(time_lock_out), .status_done(status_done),
.fail(fail));


    always #5 clk = ~clk;


    initial begin
      $dumpfile("wave.vcd");
      $dumpvars(0,tb_phase5);
      $display("Starting Phase 5 Test...");
      #10 rst = 0;
      #100;
      $display("Output: %b, Done: %b, Fail: %b", time_lock_out, status_done, fail);
      $finish;
    end
endmodule
```

DESIGN CODE:


```verilog
module phase1(input clk, input rst, input code_in, output reg status_done, output reg fail);
    reg [3:0] code;
    reg [2:0] count;
    parameter CORRECT_CODE = 4'b1011;


    always @(posedge clk or posedge rst) begin
      if (rst) begin
        count <= 0; code <= 0; status_done <= 0; fail <= 0;
      end else begin
        if (status_done || fail) begin
          count <= 0; code <= 0; status_done <= 0; fail <= 0;
        end else begin
```

```verilog
            code <= {code[2:0], code_in};

            count <= count + 1;

            if (count == 3) begin

                if ({code[2:0], code_in} == CORRECT_CODE)

                    status_done <= 1;

                else

                    fail <= 1;

            end

        end

    end

end

endmodule

module phase2(input clk, input rst, input [3:0] switch_in, output reg status_done, output reg fail);

    parameter CORRECT = 4'b1101;


    always @(posedge clk or posedge rst) begin

        if (rst) begin

            status_done <= 0;

            fail <= 0;

        end else begin

            if (switch_in == CORRECT)

                status_done <= 1;

            else

                fail <= 1;

        end

    end

endmodule

module phase3(input clk, input rst, input [2:0] dir_in, output reg status_done, output reg fail);

    reg [2:0] expected_seq [4:0];

    reg [2:0] index;

    reg [2:0] count;
```

```verilog
    initial begin

      expected_seq[0] = 3'b000; // UP

      expected_seq[1] = 3'b011; // RIGHT

      expected_seq[2] = 3'b001; // DOWN

      expected_seq[3] = 3'b010; // LEFT

      expected_seq[4] = 3'b000; // UP

    end


    always @(posedge clk or posedge rst) begin
      if (rst) begin

        index <= 0;

        status_done <= 0;

        fail <= 0;

      end else begin

        if (dir_in == expected_seq[index]) begin

          index <= index + 1;

          if (index == 4)

            status_done <= 1;

        end else begin

          fail <= 1;

          index <= 0;

        end

      end

    end

  end
endmodule
module phase4(input clk, input rst, input [7:0] plate_in, output reg status_done, output reg fail);

  reg [1:0] count;

  reg [7:0] expected_seq [2:0];


  initial begin
```

```verilog
      expected_seq[0] = 8'b10101010;

      expected_seq[1] = 8'b11001100;

      expected_seq[2] = 8'b11110000;

   end


   always @(posedge clk or posedge rst) begin

      if (rst) begin

         count <= 0; status_done <= 0; fail <= 0;

      end else begin

         if (plate_in == expected_seq[count]) begin

            count <= count + 1;

            if (count == 2)

               status_done <= 1;

         end else begin

            fail <= 1;

            count <= 0;

         end

      end

   end

endmodule

module phase5(input clk, input rst, output reg [1:0] time_lock_out, output reg status_done, output
reg fail);

   reg [1:0] step;

   reg [3:0] timer;


   always @(posedge clk or posedge rst) begin

      if (rst) begin

         step <= 0; time_lock_out <= 0; status_done <= 0; fail <= 0; timer <= 0;

      end else begin

         timer <= timer + 1;

         case (step)
```

```verilog
          0: if (timer == 4) begin time_lock_out <= 2'b01; step <= 1; timer <= 0; end

          1: if (timer == 4) begin time_lock_out <= 2'b10; step <= 2; timer <= 0; end

          2: if (timer == 4) begin time_lock_out <= 2'b11; step <= 3; timer <= 0; end

          3: status_done <= 1;

          default: fail <= 1;

        endcase

      end

    end

endmodule
```

OUTPUT:

https://edaplayground.com/x/BdBC