Class:	CPE300L - Digital System Architecture and Design Lab	Semester:	Fall 2025

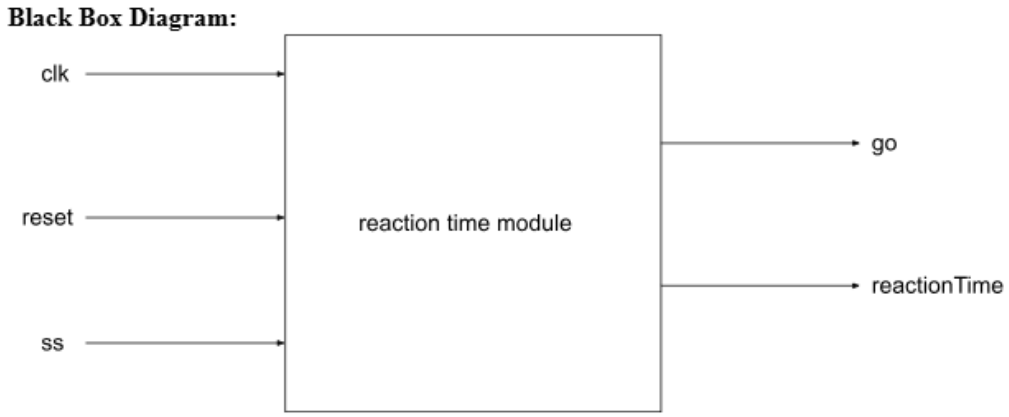
Points		Document author:	Darryll Mckoy
		Author's email:	Mckoyd1@unlv.nevada.edu
		Document topic:	Postlab 3

Instructor's comments:

1. Hours spent on this lab

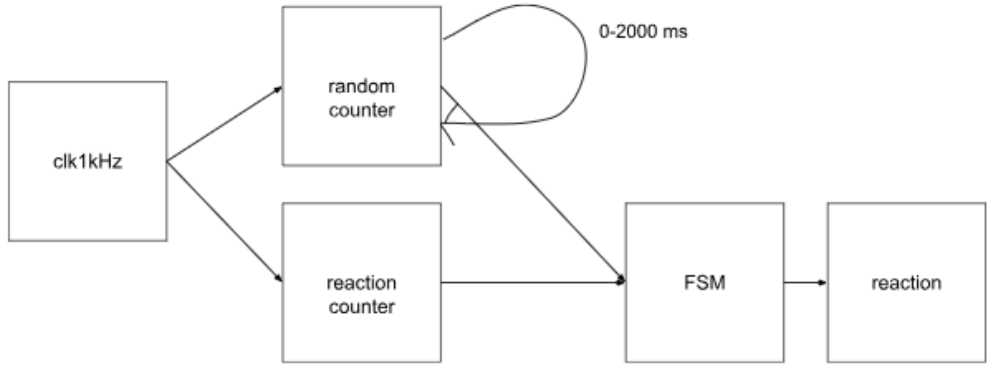
About 10 hours (could not get the DE2 usb blaster to work)

2. Reaction timer black box diagram



3. Reaction timer block diagram

Block Diagram:



---

4. System Verilog code and submodules

Reaction sv code (reaction.sv)

```
1   //Structural system verilog reaction timer Code (Lab 3)
2   module reaction(    input logic clk, reset,
3               input logic ss,
4               output logic go,
5               output logic [10:0] reactionTime
6   );
7
8       //internal signals
9       logic clk1k;
10      logic [31:0] msCnt;
11      logic [10:0] reactionCnt;
12      logic [10:0] rndmCnt;
13      logic capture;
14
15      //1 kHz clock submodule
16      clk1kHz u_div (
17      .clk(clk),
18      .reset(reset),
19      .clkout(clk1k)
20  );
21
22  //millisecond counter submodule
23  random_cnt u_reaction (
24      .random_clk(clk1k),
25      .reset(reset),
26      .cnt(reactionCnt)
27  );
```

```
28
29  //secondary counter for random time
30  random_cnt u_rand (
31      .random_clk(clk1k),
32      .reset(reset),
33      .cnt(rndmCnt)
34  );
35
36  //FSM submodule
37  reaction_FSM u_fsm (
38      .clk(clk1k),
39      .reset(reset),
40      .ss(ss),
41      .cnt(rndmCnt),
42      .go(go),
43      .capture(capture)
44  );
45
46  //reaction time register
47  always_ff @(posedge clk1k or posedge reset) begin
48      if (reset) begin
49          reactionTime <= 0;
50      end
51      else if (capture) reactionTime <= reactionCnt;
52  end
53
54  endmodule
```

1kHz clock sv code (clk1kHz.sv)

```
1
2   module clk1kHz(input  logic clk, reset, output logic clkout);
3
4     logic [15:0] count = 0;   // Counter to count clock cycles (e.g., 20 bits for 1M counts)
5     int value = 1000; // The number of clock cycles for 1 kHz
6     int clk_20kHz = 20000;
7
8   ////////////////////////////////////////////////////////////
9
10      // Frequency divider logic
11      always @(posedge clk or posedge reset) begin
12
13          if (reset) begin
14              count <= 16'd0;
15              clkout <= 1'b0;
16          end
17
18
19        else begin if (count == value) begin
20                  count <= 16'd0;
21                  clkout <= ~clkout;
22              end
23
24              else begin
25                  count <= count + 1;
26
27              end
28
29      end
30  end
31
32  endmodule
```

Random counter sv code (random_cnt.sv)

```
/* random counter
This loops from 0 - 2000 */

module random_cnt(input  logic random_clk, reset,
output logic [10:0] cnt);

  logic [10:0] rnd_count = 0;
  int value = 10; // The number of clock cycles for 2000ms

//////////////////////////////////////////////////////////
    always @(posedge random_clk or posedge reset) begin

        if (reset) begin
            rnd_count <= 11'b0;
            cnt <= 11'b0;
        end


        else begin if (rnd_count == value) begin
            cnt <= 0;
            rnd_count <= 11'b0;
        end

        else begin
            rnd_count <= rnd_count + 1;
            cnt <= rnd_count;
        end

    end

    end

endmodule

```

Reaction FSM code (reaction_FSM.sv)

```
//CPE 300L, Lab3 reaction_FSM

module reaction_FSM(input  logic clk, reset,
                    input  logic ss,
                   input  logic [10:0] cnt,
                    output logic go, capture);


  typedef enum logic [4:0] {S0, S1, S2, S3, S4} statetype;
  statetype state, nextstate;

  // state register
  always_ff @(posedge clk, posedge reset) begin
    if (reset) begin
      state <= S0;
      $display("reset pressed moving to state S0");
    end

    else begin

   state <= nextstate;

   end
end

  // next state logic
  always_comb begin

  capture <= 1'b0;

    case(state)

    ////////////S0 press ss to start system ////////////////////////////
      S0:     if (ss) begin
                //capture <= 1'b0;
                nextstate = S1;    //if ss pressed then move on to next state to start clock divider;
                $display("ss pressed moving to state S1");
      end

        else begin
        nextstate = S0;   //if ss not pressed then stay in current state.
        $display("ss not pressed staying in state S0");
        end

```

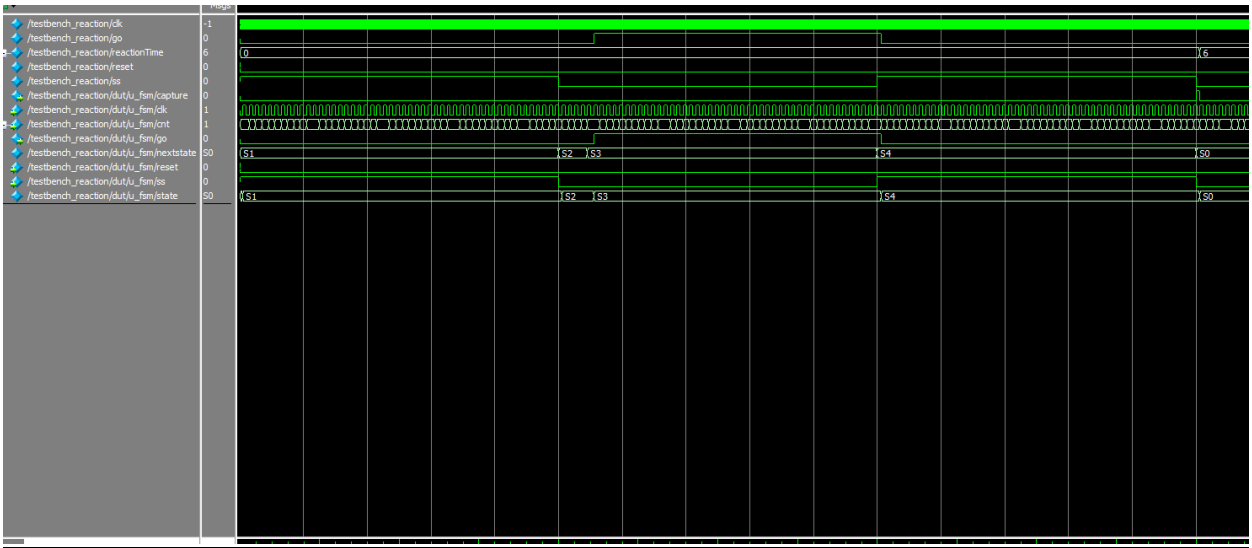
```
45      ////////// S1 start clock divider (50mhz to 1khz) in this state /////
46        S1:   if (!ss) begin
47                  nextstate = S2;
48                  $display("initializing random counter");
49        end
50
51          else begin
52            nextstate = S1;
53            $display("waiting for clk divider to complete");
54          end
55
56      ////////// S2 start random counter (loops from 0-2000) /////
57        S2:    if (cnt == 1'd0) begin
58                  nextstate = S3;
59                  $display("cnt equal to 0 asserting go and moving to state S3");
60        end
61
62          else begin
63            nextstate = S2;
64            $display("waiting for cnt to reach zero!");
65          end
66
67      ////////// S3 assert go (board LED gives signal) start reaction timer ///////
68
69        S3:  if (ss) begin
70          nextstate = S4;
71          $display("starting reaction timer and moving to state S4");
72        end
73
74      else begin
75        nextstate = S3;
76        $display("ss not pressed staying in state S3");
77      end
78
79      ////////// S4 capture reaction time, set go = 0, capture = 1 ////////////
80
81        S4:  if (!ss) begin
82          capture <= 1'b1;
83          //go <= ~go;
84          nextstate = S0;
85          $display("ss pressed asserting capture, outputting reaction time and moving to state S0");
86        end
```

```
89      else begin
90        nextstate = S4;
91        $display("ss not pressed staying in state S4");
92      end
93
94
95
96        default: begin
97      nextstate = S0;
98      //go = 1'b0;
99    end
100
101     endcase
102
103 end
104   // output logic
105   assign go = (state == S3);
106
107 endmodule
```

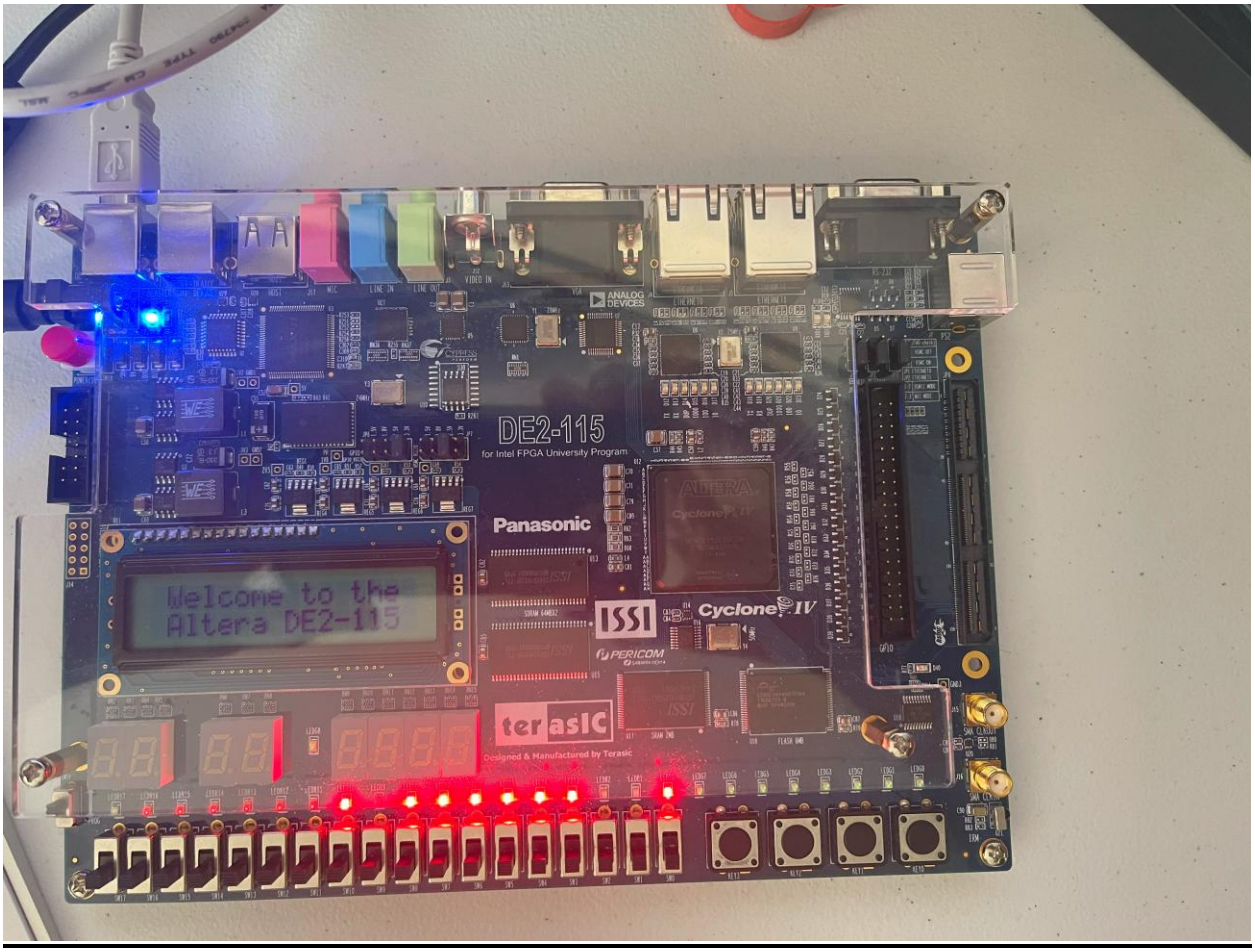
Reaction timer testbench (testbench_reaction.sv)

```
1   //reaction timer testbench (lab 3)
2
3   module testbench_reaction();
4     logic        clk, reset;
5     logic        ss, go;
6     logic [10:0] reactionTime;
7
8     // instantiate device under test
9     reaction dut(clk, reset, ss, go, reactionTime);
10
11    // generate clock
12    always
13      begin
14        clk = 1; #5; clk = 0; #5;
15
16      end
17
18    // at start of test, load vectors
19    // and pulse reset
20    initial
21      begin
22      reset = 1; #5; reset = 0;
23      ss = 1; #10;
24
25      end
26
27  endmodule
```

5. Reaction timer simulated waveform



6. Screenshot of binary reaction time on DE2-115 board



Video demonstration: https://youtu.be/7MMEMc39STY?si=XSUWIVGCfESqhqm5

7. Feedback

Would recommend a video demonstration or screenshots of expected outcomes on DE2 board within the assignment instruction PDF.