

Problem 1: [12 points] Drill problem
Filename: hw5prob1.sv
AndrewID: xinyew

```
1 `default_nettype none
2
3 module DflipflopPR
4   (input logic D,
5    input logic clock, preset_L, reset_L,
6    output logic Q);
7
8   always_ff @(posedge clock, negedge reset_L, negedge preset_L)
9     if (~reset_L && ~preset_L)
10       Q <= 1'bx;
11     else if (~reset_L)
12       Q <= 1'b0;
13     else if (~preset_L)
14       Q <= 1'b1;
15     else
16       Q <= D;
17
18 endmodule : DflipflopPR
19
20 module DflipflopPR_test();
21
22   logic clock, reset_L, preset_L, D, Q;
23   DflipflopPR dut(.D(D),
24                   .Q(Q),
25                   .clock(clock),
26                   .reset_L(reset_L),
27                   .preset_L(preset_L));
28
29   // clock wave
30   initial begin
31     clock = 1'b0;
32     #10 clock = ~clock;
33     #20 clock = ~clock;
34     #20 clock = ~clock;
35     #10 clock = ~clock;
36     #10 clock = ~clock;
37     #10 clock = ~clock;
38     #10 clock = ~clock;
39   end
40
41   // reset_L wave
42   initial begin
43     reset_L = 1'b0;
44     #5 reset_L = ~reset_L;
45     #15 reset_L = ~reset_L;
46     #5 reset_L = ~reset_L;
47     #30 reset_L = ~reset_L;
48     #5 reset_L = ~reset_L;
49     #5 reset_L = ~reset_L;
50     #15 reset_L = ~reset_L;
51   end
52
53   // preset_L wave
54   initial begin
55     preset_L = 1'b1;
56     #35 preset_L = ~preset_L;
57     #5 preset_L = ~preset_L;
58     #45 preset_L = ~preset_L;
59   end
60
61   // D wave
62   initial begin
63     $monitor($time,, "D = %b, Q = %b, reset_L = %b, preset_L = %b",
64             D, Q, reset_L, preset_L);
65     // initial values
66     D = 1'b0;
67     // waveform
68     #5 D = ~D;
69     #10 D = ~D;
```

```
70      #5 D = ~D;
71      #5 D = ~D;
72      #10 D = ~D;
73      #10 D = ~D;
74      #10 D = ~D;
75      #20 D = ~D;
76      #10 D = ~D;
77      #10 D = ~D;
78      #5 $finish;
79  end
80 endmodule : DflipflopPR_test
```

Problem 2: [6 points] Drill problem
Filename: hw5prob2.sv
AndrewID: xinyew

```
1 `default_nettype none
2
3 module hw5prob2_test();
4     logic value, clock, hue, reset_n, water;
5
6     hw5prob2 dut(.hue(hue),
7                 .value(value),
8                 .clock(clock),
9                 .reset_n(reset_n),
10                .water(water));
11
12     initial begin
13         $monitor($time,, "state = %s, nextState = %s, in(%b%b), out(%b), reset_n(...
Line length of 82 (max is 80)
14         dut.state.name, dut.nextState.name, hue, value, water, reset_n);
15         // init -> Red
16         clock = 0;
17         reset_n = 1'b0;
18         reset_n <= 1'b1;
19
20         forever #10 clock = ~clock;
21     end
22
23     initial begin
24         {hue, value} <= 2'b00;
25         @(posedge clock); // #10 Red + 00 -> Pink
26         @(posedge clock); // #30 Pink + 00 -> Blue
27         {hue, value} <= 2'b01;
28         @(posedge clock); // #50 Blue + 01 -> Blue
29
30         // Blue Reset
31         #1 reset_n = 1'b0; // Blue + Reset -> Red
32         #1 reset_n = 1'b1; // release reset
33
34         @(posedge clock); // #70 Red + 01 -> Gold
35         @(posedge clock); // #90 Gold + 01 -> Green
36         {hue, value} <= 2'b11;
37         @(posedge clock); // #110 Green + 11 -> Green
38         {hue, value} <= 2'b00;
39         @(posedge clock); // #130 Green + 00 -> Red
40         {hue, value} <= 2'b10;
41         @(posedge clock); // #150 Red + 10 -> Blue
42
43         // Blue reset
44         #1 reset_n = 1'b0; // Blue + Reset -> Red
45         #1 reset_n = 1'b1; // release reset
46
47         {hue, value} <= 2'b11;
48         @(posedge clock); // #170 Red + 11 -> Pink
49         @(posedge clock); // # 190 Pink + 11 -> Green
50
51         // Green Reset
52         #1 reset_n = 1'b0; // Green + Reset -> Red
53         #1 reset_n = 1'b1; // release reset
54
55         // Red Reset
56         #1 reset_n = 1'b0; // Red + Reset -> Red
57         #1 reset_n = 1'b1; // release reset
58
59         @(posedge clock); // #210 Red + 11 -> Pink
60
61         // Pink Reset
62         #1 reset_n = 1'b0; // Pink + Reset -> Red
63         #1 reset_n = 1'b1; // release reset
64
65         {hue, value} <= 2'b01;
66         @(posedge clock); // #230 Red + 01 -> Gold
67
68         // Gold Reset
```

```
69     #1 reset_n = 1'b0; // Gold + Reset -> Red
70     #1 reset_n = 1'b1; // release reset
71
72     #1 $finish;
73 end
74
75 endmodule : hw5prob2_test
```

Problem 3: [6 points] Drill problem
Filename: hw5prob3.sv
AndrewID: xinyew

```
1 `default_nettype none
2 module hw5prob3_test();
3     logic gym, trainer, clock, reset, got_em_all;
4
5     hw5prob3 dut(.gym(gym),
6                 .trainer(trainer),
7                 .clock(clock),
8                 .reset(reset),
9                 .got_em_all(got_em_all));
10
11     initial begin
12         $monitor($time,, "state = %s, nextState = %s, out(%b), in(%b%b), reset(%b)",
13                 dut.state.name, dut.nextState.name, got_em_all, gym, trainer, re...
14         // init
15         clock = 1'b0;
16         reset = 1'b1;
17         reset <= 1'b0;
18
19         forever #10 clock = ~clock;
20     end
21
22     initial begin
23         {gym, trainer} <= 2'b10;
24         @(posedge clock); // #10 G 10 -> E 0
25         @(posedge clock); // #30 E 10 -> S 0
26         @(posedge clock); // #50 S 10 -> S 1
27         {gym, trainer} <= 2'b01;
28         @(posedge clock); // #70 S 01 -> G 1
29         @(posedge clock); // #90 G 01 -> E 0
30
31         // E + Reset/01 -> G 1
32         #1 reset = 1'b1;
33         #1 reset = 1'b0;
34
35         {gym, trainer} <= 2'b11;
36         @(posedge clock); // #110 G 11 -> S 0
37
38         // S + Reset/11 -> G 1
39         #1 reset = 1'b1;
40         #1 reset = 1'b0;
41
42         {gym, trainer} <= 2'b00;
43         @(posedge clock); // #130 G 00 -> S 1
44
45         // S + Reset/00 -> G 0
46         #1 reset = 1'b1;
47         #1 reset = 1'b0;
48
49         // G + Reset/00 -> G 0
50         #1 reset = 1'b1;
51         #1 reset = 1'b0;
52
53         #1 $finish;
54     end
55
56 endmodule : hw5prob3_test
```

Problem 6: [8 points]
Filename: hw5prob6.sv
AndrewID: xinyew

```
1 `default_nettype none
2
3 // button debouncing circuit
4 module hw5prob6
5     (input logic button,
6      input logic clock, reset,
7      output logic debounced);
8
9     // enum of all 14 states
10    enum logic [3:0] {
11        SAW0 = 4'd0, STABLE1_01 = 4'd1, STABLE1_02 = 4'd2,
12        STABLE1_03 = 4'd3, STABLE1_04 = 4'd4, STABLE1_05 = 4'd5,
13        STABLE1_06 = 4'd6, STABLE1_07 = 4'd7, STABLE1_08 = 4'd8,
14        STABLE1_09 = 4'd9, STABLE1_10 = 4'd10, STABLE1_11 = 4'd11,
15        STABLE1_12 = 4'd12, FINAL_STABLE1 = 4'd13
16    } state, nextState;
17
18    // async reset FF block
19    always_ff @(posedge clock, posedge reset)
20        if (reset)
21            state <= SAW0;
22        else
23            state <= nextState;
24
25    // next state logic
26    always_comb
27        unique case (state)
28            SAW0: nextState = (button) ? STABLE1_01 : SAW0;
29            STABLE1_01: nextState = STABLE1_02;
30            STABLE1_02: nextState = STABLE1_03;
31            STABLE1_03: nextState = STABLE1_04;
32            STABLE1_04: nextState = STABLE1_05;
33            STABLE1_05: nextState = STABLE1_06;
34            STABLE1_06: nextState = STABLE1_07;
35            STABLE1_07: nextState = STABLE1_08;
36            STABLE1_08: nextState = STABLE1_09;
37            STABLE1_09: nextState = STABLE1_10;
38            STABLE1_10: nextState = STABLE1_11;
39            STABLE1_11: nextState = STABLE1_12;
40            STABLE1_12: nextState = FINAL_STABLE1;
41            FINAL_STABLE1: nextState = (button) ? FINAL_STABLE1 : SAW0;
42        endcase
43
44    // output logic
45    always_comb
46        case (state)
47            SAW0: debounced = 0;
48            default: debounced = 1;
49        endcase
50
51 endmodule : hw5prob6
```