DSD Lab 4 Final Report

PAUL KIM

Fall 2019

Lab4_PK.sv:

```
module lab4_PK(input logic clk, input logic reset, input logic left, right, output logic lc, lb, la, ra, rb, rc);

   always@ (posedge clk)
   begin
      if (reset == 1)
      begin
         //On reset, the FSM should enter a state with all lights off.
         la = 0; lb = 0; lc = 0; ra = 0; rb = 0; rc = 0;
      end

      else
      begin
         // left and right input state only matters at light off state
         if ( (la == 0) & (lb == 0) & (lc == 0) & (ra == 0) & (rb == 0) & (rc == 0) )
         begin
            if ( (left == 1) & (right == 0) )
               //at light off state and only left is on, then left light sequence is triggered
               la = 1;

            else if ( (right == 1) & (left == 0) )
               //at light off state and only right is on, then right light sequence is triggered
               ra = 1;

            //if both left and right are either on or off synchronously, then neither left nor right is triggered
         end

         // if any of the light is on, then the pattern must be finished
         else
         //lc | lb | la | ra | rb | rc
         begin
         //left
            if ( (la == 1) & (lb == 0) & (lc  == 0) )
               //at 001000,  proceed to next state 011000
               lb = 1;
            else if ( (la == 1) & (lb == 1) & (lc  == 0) )
               //at 011000,  proceed to next state 111000
               lc = 1;
            else if ( (la == 1) & (lb == 1) & (lc  == 1) )
            begin
               //at 111000,  proceed light off state
               la = 0; lb = 0; lc = 0;
            end
```

```
43          //right
44             else if ( (ra == 1) & (rb == 0) & (rc == 0) )
45                //at 000100, proceed to next state 000110
46                rb = 1;
47             else if ( (ra == 1) & (rb == 1) & (rc == 0) )
48                //at 000110, proceed to next state 000111
49                rc = 1;
50             else if ( (ra == 1) & (rb == 1) & (rc == 1) )
51             begin
52                //at 000111, proceed to light off state
53                ra = 0; rb = 0; rc = 0;
54             end
55          end
56       end
57    end
58 endmodule
59
60 module lab4_PK_TB();
61    //testbench will be test with testvector.txt in the format of xx_yyyyyy
62    //each left or right one whole cycle requires 5 periods:
63    //ex)left: when left signal input is triggered,
64    //state 3(light off) -> state 0 -> state 1 -> state 2 -> state 3
65    logic clk, reset;
66    logic left, right; //input
67    logic lc, lb, la, ra, rb, rc; //output
68    logic lce, lbe, lae, rae, rbe, rce; //exepected output for testvector
69    logic [31:0] vectornum, errors; //bookkeeping variable
70    logic [7:0] testvectors[10000:0]; //array of testvectors
71
72    //instantiate device under test
73    lab4_PK dut(clk, reset, left, right, lc, lb, la, ra, rb, rc);
74
75    //generate clock
76    always //no sensitivity list, so it always executes
77    begin
78       clk = 1; #5; clk = 0; #5;
79    end
80
81    //at start of test, load vectors and pulse reset
82    initial
83    begin
84       $readmemb("testvector.txt", testvectors);
85       vectornum = 0; errors = 0;
86       reset = 1; #27; reset = 0;
87    end
88    //$readmemh reads testvector file written in hexadecimal
89
90    //apply test vector @ posedge of clk
91    always @(posedge clk)
92    begin
93       {left, right, lce, lbe, lae, rae, rbe, rce} = testvectors[vectornum];
94    end
95
96    //check results on falling edge of clk
97    always @(negedge clk)
98    // skip during reset
99    if (~reset)
100   begin
```

```
101         //left
102         if (la !== lae)
103         begin
104            $display( "Error: inputs = %b", {left, right});
105            $display( " outputs = %b (%b expected)", la, lae);
106            errors = errors + 1;
107         end
108         
109         else if (lb !== lbe)
110         begin
111            $display( "Error: inputs = %b", {left, right});
112            $display( " outputs = %b (%b expected)", lb, lbe);
113            errors = errors + 1;
114         end
115         
116         else if (lc !== lce)
117         begin
118            $display( "Error: inputs = %b", {left, right});
119            $display( " outputs = %b (%b expected)", lc, lce);
120            errors = errors + 1;
121         end
122         
123         //right
124         else if (ra !== rae)
125         begin
126            $display( "Error: inputs = %b", {left, right});
127            $display( " outputs = %b (%b expected)", ra, rae);
128            errors = errors + 1;
129         end
130         
131         else if (rb !== rbe)
132         begin
133            $display( "Error: inputs = %b", {left, right});
134            $display( " outputs = %b (%b expected)", rb, rbe);
135            errors = errors + 1;
136         end
137         
138         else if (rc !== rce)
139         begin
140            $display( "Error: inputs = %b", {left, right});
141            $display( " outputs = %b (%b expected)", rc, rce);
142            errors = errors + 1;
143         end
144         
145         //increment array index and read next testvector
146         vectornum = vectornum + 1;
147         if (testvectors[vectornum] === 8'bx)
148         begin
149            $display( "%d tests completed with %d errors", vectornum, errors );
150         end
151      end
152   endmodule
153
154
```

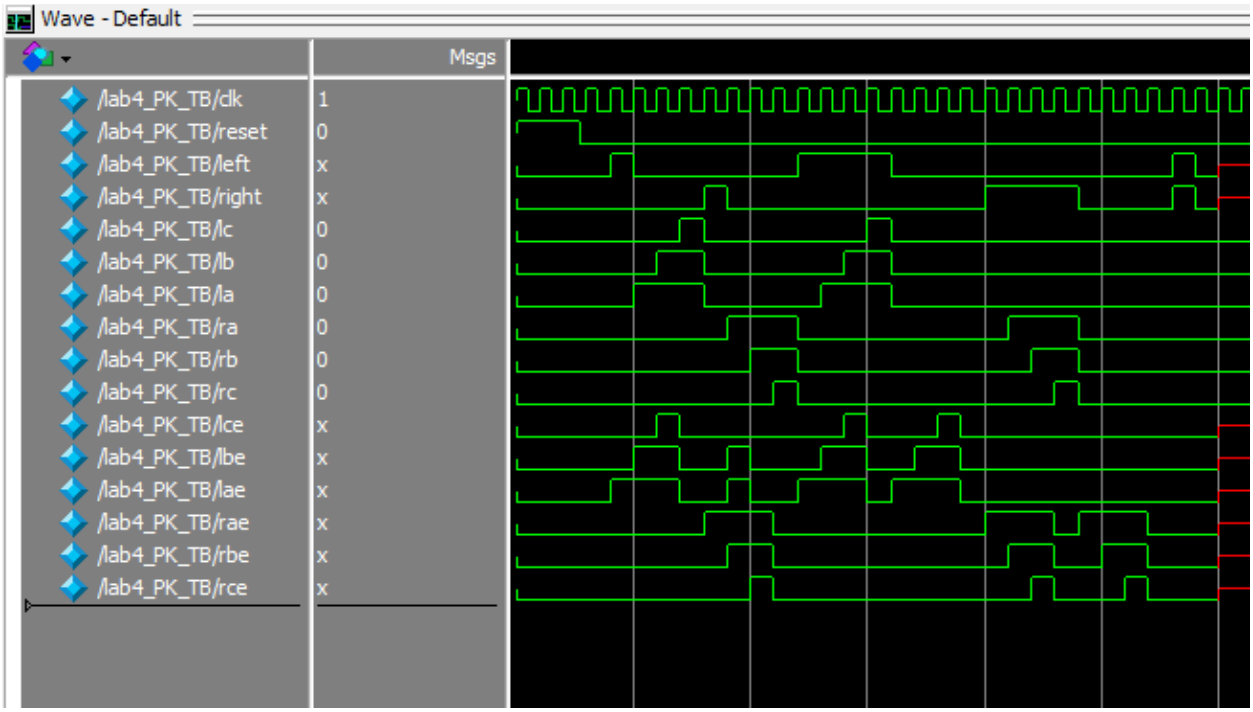
Testvector.txt:

testvector - Notepad

File Edit Format View Help

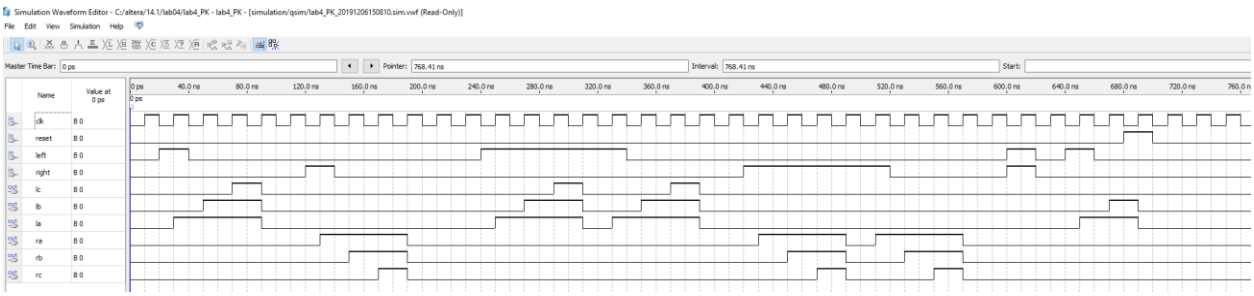
```
00_000000
10_001000
00_011000
00_111000
00_000000
01_000100
00_011110
00_000111
00_000000
10_001000
10_011000
10_111000
10_000000
00_001000
00_011000
00_111000
00_000000
01_000100
01_000110
01_000111
01_000000
00_000100
00_000110
00_000111
00_000000
11_000000
00_000000
```

RTL simulation waveform (testbench with testvectors):



For this lab, I had to code for a system that turns how 6 lights when either two inputs (left and right) is triggered in systemverilog hdl. Every left or right flashing sequence starts from 1 light turned on then 2 lights turned on then 3. Flashing sequence must finish flashing sequence even if the input is turned off. If both left and right turned on or off, the system ignores the input and flashing sequence won't happened.

I had a trouble writing the code because it wasn't easy to test your svhdl code. Only way to test without doing testbench was to use "the university program vwf" and draw out all the inputs by manually:



Which made me sure that my main code for the flashing sequence was correct. I've replicate these sequence results to testvector.txt except reset at the end since reset is test within the TB code.

Assignment Editor:

	Status	From	To	Assignment Name	Value	Enabled
1	✓		in right	Location	PIN_AB28	Yes
2	✓		in reset	Location	PIN_AC27	Yes
3	✓		out lc	Location	PIN_E18	Yes
4	✓		out lb	Location	PIN_F18	Yes
5	✓		out la	Location	PIN_F21	Yes
6	✓		out ra	Location	PIN_E19	Yes
7	✓		out rb	Location	PIN_F19	Yes
8	✓		out rc	Location	PIN_G19	Yes
9	✓		in left	Location	PIN_AC28	Yes
10	✓		in clk	Location	PIN_M23	Yes
11		<<new>>	<<new>>	<<new>>		

My inputs keys and output lights are following:

<u>Inputs:</u>

right = sw0 | left = sw1 | reset = sw2 | clk = key0

<u>Outputs:</u>

lc = LEDR5 | lb = LEDR4 | la = LEDR3 | ra = LEDR2 | rb = LEDR1 | rc = LEDR0

I was tested and was verified by my generous Mr. Tang that my hardware simulation was correct. He helped me understand this material and helped me get the correct result