```
1   ------------------------------------------------------------------------------
2   -- Design: VHDL Class Example 6-2 : design of sequential logic / FSM         --
3   --                                                                           --
4   -- Author : Peter Roessler                                                   --
5   -- Date   : 21 09 2014                                                       --
6   -- File   : railroad_xing.vhd                                                --
7   ------------------------------------------------------------------------------
8
9   library IEEE;
10  use IEEE.std_logic_1164.all;
11
12  entity railroad_xing is
13  port (train_approaches_i : in  std_logic;
14        train_leaves_i     : in  std_logic;
15        clk                : in  std_logic;
16        reset              : in  std_logic;
17        light_o            : out std_logic;
18        gate_close_o       : out std_logic;
19        gate_open_o        : out std_logic);
20  end railroad_xing;
21
22  architecture rtl of railroad_xing is
23
24    type t_state is (OPENED_S, OPEN1_S, OPEN2_S, OPEN3_S, OPEN4_S, OPEN5_S,
25                     CLOSED_ON_S, CLOSED_OFF_S,
26                     CLOSE1_S, CLOSE2_S, CLOSE3_S, CLOSE4_S, CLOSE5_S);
27
28    signal s_states : t_state;
29
30  begin
31    p_railroad_xing : process (clk, reset)
32    begin
33      if reset = '1' then
34        -- brings system to a safe state, under all pre-conditions ...
35        light_o <= '1';
36        gate_close_o <= '1';
37        gate_open_o <= '0';
38        s_states <= CLOSE1_S;
39      elsif clk'event and clk = '1' then
40        case s_states is
41          ------------------------------------------------------------------------
42          -- STATE "Gate is completely open"
43          ------------------------------------------------------------------------
44          when OPENED_S =>
45            light_o <= '0';
46            gate_close_o <= '0';
47            gate_open_o <= '0';
48            if train_approaches_i = '1' then
49              light_o <= '1';
50              gate_close_o <= '1';
51              s_states <= CLOSE1_S;
52            else
53              s_states <= OPENED_S;
54            end if;
55          ------------------------------------------------------------------------
56          -- STATE "Gate will be closed (1st second)"
57          ------------------------------------------------------------------------
```



```
58          when CLOSE1_S =>
59            light_o <= '0';
60            gate_close_o <= '1';
61            gate_open_o <= '0';
62            s_states <= CLOSE2_S;
63          ------------------------------------------------------------------------
64          -- STATE "Gate will be closed (2nd second)"
65          ------------------------------------------------------------------------
66          when CLOSE2_S =>
67            light_o <= '1';
68            gate_close_o <= '1';
69            gate_open_o <= '0';
70            s_states <= CLOSE3_S;
71          ------------------------------------------------------------------------
72          -- STATE "Gate will be closed (3rd second)"
73          ------------------------------------------------------------------------
74          when CLOSE3_S =>
75            light_o <= '0';
76            gate_close_o <= '1';
77            gate_open_o <= '0';
78            s_states <= CLOSE4_S;
79          ------------------------------------------------------------------------
80          -- STATE "Gate will be closed (4th second)"
81          ------------------------------------------------------------------------
82          when CLOSE4_S =>
83            light_o <= '1';
84            gate_close_o <= '0';
85            gate_open_o <= '0';
86            s_states <= CLOSED_ON_S;
87          ------------------------------------------------------------------------
88          -- STATE "Gate is completely closed, light is turned on"
89          ------------------------------------------------------------------------
90          when CLOSED_ON_S =>
91            light_o <= '0';
92            gate_close_o <= '0';
93            gate_open_o <= '0';
94            if train_leaves_i = '1' then
95              -- jump to OPEN2_S to ensure blinking frequency of exactly 0.5 Hz
96              gate_open_o <= '1';
97              s_states <= OPEN2_S;
98            else
99              s_states <= CLOSED_OFF_S;
100           end if;
101         ------------------------------------------------------------------------
102         -- STATE "Gate is completely closed, light is turned off"
103         ------------------------------------------------------------------------
104         when CLOSED_OFF_S =>
105           light_o <= '1';
106           gate_close_o <= '0';
107           gate_open_o <= '0';
108           if train_leaves_i = '1' then
109             gate_open_o <= '1';
110             s_states <= OPEN1_S;
111           else
112             s_states <= CLOSED_ON_S;
113           end if;
114         ------------------------------------------------------------------------
```

```
115           -- STATE "Gate will be opened (1st second)"
116           ------------------------------------------------------------------------
117           when OPEN1_S =>
118             light_o <= '0';
119             gate_close_o <= '0';
120             gate_open_o <= '1';
121             s_states <= OPEN2_S;
122           ------------------------------------------------------------------------
123           -- STATE "Gate will be opened (2nd or 1st second, respectively)"
124           ------------------------------------------------------------------------
125           when OPEN2_S =>
126             light_o <= '1';
127             gate_close_o <= '0';
128             gate_open_o <= '1';
129             s_states <= OPEN3_S;
130           ------------------------------------------------------------------------
131           -- STATE "Gate will be opened (3rd or 2nd second, respectively)"
132           ------------------------------------------------------------------------
133           when OPEN3_S =>
134             light_o <= '0';
135             gate_close_o <= '0';
136             gate_open_o <= '1';
137             s_states <= OPEN4_S;
138           ------------------------------------------------------------------------
139           -- STATE "Gate will be opened (4th or 3rd second, respectively)"
140           ------------------------------------------------------------------------
141           when OPEN4_S =>
142             light_o <= '1';
143             gate_close_o <= '0';
144             gate_open_o <= '1';
145             s_states <= OPEN5_S;
146           ------------------------------------------------------------------------
147           -- STATE "Gate will be opened (5tt or 4th second, respectively)"
148           ------------------------------------------------------------------------
149           when OPEN5_S =>
150             light_o <= '0';
151             gate_close_o <= '0';
152             gate_open_o <= '0';
153             s_states <= OPENED_S;
154           ------------------------------------------------------------------------
155           -- all other cases ...
156           ------------------------------------------------------------------------
157           when others =>
158             light_o <= '1';
159             gate_close_o <= '1';
160             gate_open_o <= '0';
161             s_states <= CLOSE1_S;
162         end case;
163       end if;
164     end process p_railroad_xing;
165   end rtl;
166
```

```
1  ------------------------------------------------------------------------------
2  -- Design: VHDL Class Example 6-2: design of sequential logic/FSM, testbench --
3  --                                                                          --
4  -- Author : Peter Roessler                                                  --
5  -- Date   : 21 09 2014                                                      --
6  -- File   : tb_railroad_xing.vhd                                            --
7  ------------------------------------------------------------------------------
8
9  library IEEE;
10 use IEEE.std_logic_1164.all;
11
12 entity tb_railroad_xing is end tb_railroad_xing;
13
14 architecture sim of tb_railroad_xing is
15
16   -- Declaration of the component under test
17   component railroad_xing
18     port (
19       train_approaches_i : in  std_logic;
20       train_leaves_i     : in  std_logic;
21       clk                : in  std_logic;
22       reset              : in  std_logic;
23       light_o            : out std_logic;
24       gate_close_o       : out std_logic;
25       gate_open_o        : out std_logic);
26   end component;
27
28   signal train_approaches_i : std_logic;
29   signal train_leaves_i : std_logic;
30   signal clk : std_logic;
31   signal reset : std_logic;
32   signal light_o : std_logic;
33   signal gate_close_o : std_logic;
34   signal gate_open_o : std_logic;
35
36 begin
37
38   -- Instantiate the design under test
39   i_railroad_xing : railroad_xing
40     port map (
41       train_approaches_i => train_approaches_i,
42       train_leaves_i     => train_leaves_i,
43       clk                => clk,
44       reset              => reset,
45       light_o            => light_o,
46       gate_close_o       => gate_close_o,
47       gate_open_o        => gate_open_o);
48
49   -- Generate clock
50   p_clk : process
51   begin
52     clk <= '0';
53     wait for 0.5 sec;
54     clk <= '1';
55     wait for 0.5 sec;
56   end process p_clk;
57
```



```
58   p_stim : process
59   begin
60     -- Initial states of all input ports and reset signal
61     train_approaches_i <= '0';
62     train_leaves_i <= '0';
63     reset <= '1';
64     wait for 2 sec;
65     -- Deassert reset
66     reset <= '0';
67     wait for 10 sec;
68     -- Start applying test patters
69     train_leaves_i <= '1';
70     wait for 1 sec;
71     train_leaves_i <= '0';
72     wait for 10 sec;
73     train_approaches_i <= '1';
74     wait for 1 sec;
75     train_approaches_i <= '0';
76     wait for 10 sec;
77     train_leaves_i <= '1';
78     wait for 1 sec;
79     train_leaves_i <= '0';
80     wait for 10 sec;
81     -- Stop simulation
82     assert false report "END OF SIMULATION" severity failure;
83   end process p_stim;
84
85 end sim;
86
```