Hardware Software Co-Design

Shobhit Kumar <2021mt12072@wilp.bits-pilani.ac.in>

BITS ID – 2021MT12072

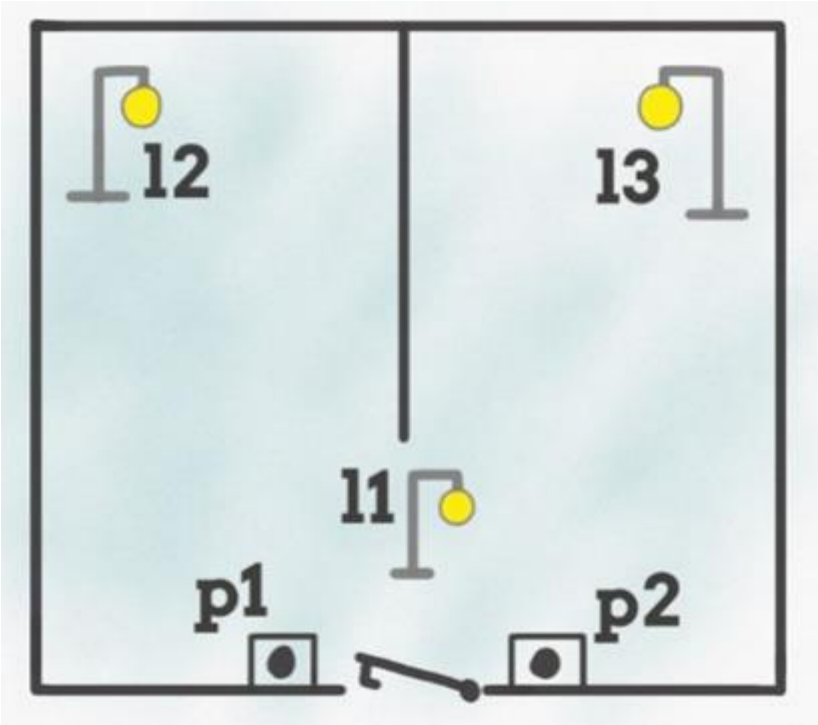
HSCD Assignment 1

1. Problem Statement

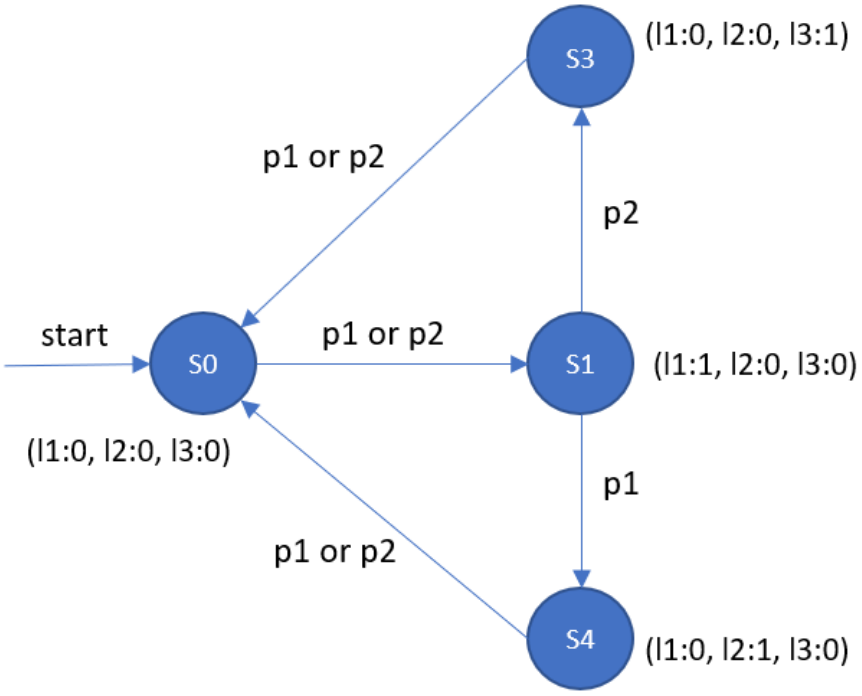
Q. Find a State Machine as the following and as expressed in a floorplan diagram. At the default state, the lights are all turned off. When you enter the house, you can press one of the 2 push buttons you have, p1 or p2. When you press any of those buttons, the l1 light turns on. Imagine this is the entrance light, and you can take your jacket off. Once you are done, you decide which room you want to enter (kitchen or bedroom, for example). If you press the button p1, l1 turns off and l2 turns on. Instead, if you press the button p2, l1 turns off and l3 turns on. Pressing another time any of the 2 buttons, p1 or p2, the light that is currently on will turn off, and we'll get back at the initial state of the system.

Model the system using SystemC and write test bench to run the simulation. You need to submit a single pdf file containing the problem statement, FSM diagram, SystemC code, test bench, and assumptions made if any. The code should be commented on appropriately.

You can preferably use Modelsim 32-bit version provided to you by the Institute.



2. FSM Diagram



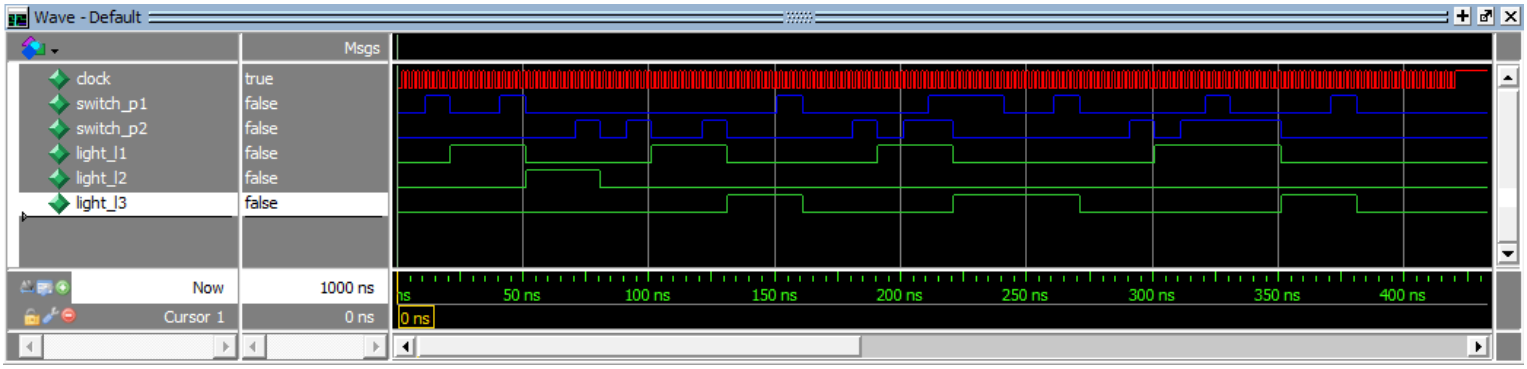
3. Assumption

- The design assumes that each of the buttons p1 and p2 are push buttons and state change happens when we press and release the switch
- If both the switches are pressed at the same time or one after the other without releasing the first pressed switch, the switch that is pressed first would be treated for state change after release irrespective of in which order the switches were released. The last two test cases in the testbench demonstrate this.

4. Simulation output with waveform

ModelSim> vsim -gui BITS.sc_main
vsim -gui BITS.sc_main
Start time: 12:34:47 on Mar 04,2023
** Note: (vsim-3813) Design is being optimized due to module recompilation...
Loading C:/Users/shobh/BITS/wilp-mtech/sem2/HSCD/sysc-lights/BITS_sc\win32_gcc-4.2.1\systemc.so
Loading C:/Users/shobh/BITS/wilp-mtech/sem2/HSCD/sysc-lights/BITS.sc_main
Initializing the house
@0 s Starting Simulation
VSIM 138> add wave sim:/sc_main/*
VSIM 138> run
@11 ns Press Switch: P1
@21 ns Release Switch: P1
@21 ns L1: 1, L2: 0, L3: 0
#
@41 ns Press Switch: P1
@51 ns Release Switch: P1
@51 ns L1: 0, L2: 1, L3: 0
#
@71 ns Press Switch: P2
@81 ns Release Switch: P2
@81 ns L1: 0, L2: 0, L3: 0
#
@91 ns Press Switch: P2
@101 ns Release Switch: P2
@101 ns L1: 1, L2: 0, L3: 0
#
@121 ns Press Switch: P2
@131 ns Release Switch: P2
@131 ns L1: 0, L2: 0, L3: 1
#
@151 ns Press Switch: P1
@161 ns Release Switch: P1
@161 ns L1: 0, L2: 0, L3: 0
#
@181 ns Press Switch: P2
@191 ns Release Switch: P2
@191 ns L1: 1, L2: 0, L3: 0
#
@201 ns Press Switch: P2
@211 ns Press Switch: P1
@221 ns Release Switch: P2
@221 ns L1: 0, L2: 0, L3: 1
#

```
# @241 ns Release Switch: P1
# @261 ns Press Switch: P1
# @271 ns Release Switch: P1
# @271 ns L1: 0, L2: 0, L3: 0
#
# @291 ns Press Switch: P2
# @301 ns Release Switch: P2
# @301 ns L1: 1, L2: 0, L3: 0
#
# @311 ns Press Switch: P2
# @321 ns Press Switch: P1
# @331 ns Release Switch: P1
# @351 ns Release Switch: P2
# @351 ns L1: 0, L2: 0, L3: 1
#
# @371 ns Press Switch: P1
# @381 ns Release Switch: P1
# @381 ns L1: 0, L2: 0, L3: 0
#
```



5. Design Code – design.cpp

```
#include "systemc.h"

using namespace std;

// System State definitions
typedef enum {
    STATE_0,
    STATE_1,
    STATE_2,
    STATE_3,
    STATE_MAX,
} house_state;

// Possible signals - switches
typedef enum {
    SIGNAL_P1,
    SIGNAL_P2,
    SIGNAL_MAX
} house_events;

// Initialize static light states based on the system states
typedef struct {
    int l1;
    int l2;
    int l3;
} light_state;

static light_state lights[STATE_MAX] = {
    { 0, 0, 0 },
    { 1, 0, 0 },
    { 0, 1, 0 },
    { 0, 0, 1 },
};

// Define the state transition for every signal
static int machine[STATE_MAX][SIGNAL_MAX] = {
    { STATE_1, STATE_1 },
    { STATE_2, STATE_3 },
    { STATE_0, STATE_0 },
    { STATE_0, STATE_0 }
};

SC_MODULE(my_house) {
```

```
49.
50.     // define the module input, output and clock signals
51.     sc_in_clk clock;
52.     sc_in<bool> switch_p1;
53.     sc_in<bool> switch_p2;
54.
55.     sc_out<bool> light_l1;
56.     sc_out<bool> light_l2;
57.     sc_out<bool> light_l3;
58.
59.     int curr_signal, curr_state;
60.     int change_state, pressed;
61.
62.     void control_lights() {
63.         // when any of the signal is asserted, read state of each signal to
64.         // know which one is asserted
65.         int sw1 = switch_p1.read();
66.         int sw2 = switch_p2.read();
67.
68.         if (sw1 == 1 && pressed == 0) {
69.             // only if sw1 is asserted and it was not just help
70.             // high but pressed first time this will also ensure
71.             // if this was pressed first in case of simultaneous
72.             // switch presses, sw1 will be tracked for state change
73.             curr_signal = SIGNAL_P1;
74.             pressed = 1;
75.         } else if (sw2 == 1 && pressed == 0) {
76.             // only if sw2 is asserted and it was not just help high
77.             // but pressed first time this will also ensure if this
78.             // was pressed first in case of simultaneous switch presses,
79.             // sw2 will be tracked for state change
80.             curr_signal = SIGNAL_P2;
81.             pressed = 1;
82.         } else if (sw1 == 0 && curr_signal == SIGNAL_P1) {
83.             // track release of sw1 as state change trigger if it was
84.             // already asserted high
85.             change_state = 1;
86.         } else if (sw2 == 0 && curr_signal == SIGNAL_P2) {
87.             // track release of sw2 as state change trigger if it was
88.             // already asserted high
89.             change_state = 1;
90.         } else {
91.             return;
92.         }
93.
```

```
94.         if (!change_state) {
95.             // No state change recorded yet
96.             return;
97.         }
98.
99.         // Based on current state and final signal determined
100.            // for state change go to next state as per state machine
101.            curr_state = machine[curr_state][curr_signal];
102.
103.            // based on the current state, get what should be the
104.            // light status and assert the output signal for the
105.            // lights accordingly
106.            light_state lt_state = lights[curr_state];
107.            light_l1.write(lt_state.l1);
108.            light_l2.write(lt_state.l2);
109.            light_l3.write(lt_state.l3);
110.
111.            cout << "@" << sc_time_stamp() << " L1: " << lt_state.l1 << ", L2:
    " << lt_state.l2 << ", L3: " << lt_state.l3 << "\n" << endl;
112.
113.            // reset the tracking variables
114.            curr_signal = SIGNAL_MAX;
115.            change_state = 0;
116.            pressed = 0;
117.        }
118.
119.        SC_CTOR(my_house) {
120.            cout << "Initializing the house" << endl;
121.            curr_state = STATE_0;
122.            curr_signal = SIGNAL_MAX;
123.            pressed = 0;
124.            change_state = 0;
125.
126.            // main method to handle signals
127.            SC_METHOD(control_lights);
128.
129.            // set sensitivity to both the input signals
130.            sensitive << switch_p1;
131.            sensitive << switch_p2;
132.        }
133.    };
134.
```

6. Testbench – testbench.cpp

```
#include "systemc.h"
#include "design.cpp"

using namespace std;

#define RUN_SIM(duration) \
        for (i = 0; i < duration; i++) { \
                clock = 0; \
                sc_start(1, SC_NS); \
                clock = 1; \
                sc_start(1, SC_NS); \
        } \

int sc_main (int argc, char* argv[])
{
        sc_signal<bool> clock;
        sc_signal<bool> switch_p1;
        sc_signal<bool> switch_p2;

        sc_signal<bool> light_l1;
        sc_signal<bool> light_l2;
        sc_signal<bool> light_l3;

        int i = 0;
        // Initialize and connect to the DUT
        my_house house("house");
        house.clock(clock);
        house.switch_p1(switch_p1);
        house.switch_p2(switch_p2);
        house.light_l1(light_l1);
        house.light_l2(light_l2);
        house.light_l3(light_l3);

        cout << "@" << sc_time_stamp() <<" Starting Simulation" << endl;
        sc_start(1, SC_NS);
        switch_p1 = 0;
        switch_p2 = 0;

        RUN_SIM(5);
        cout << "@" << sc_time_stamp() <<" Press Switch: P1" << endl;
        switch_p1 = 1;    // Press P1
        RUN_SIM(5);
        cout << "@" << sc_time_stamp() <<" Release Switch: P1" << endl;
```

```
50.         switch_p1 = 0;    // Release P1
51.
52.         RUN_SIM(10);
53.         cout << "@" << sc_time_stamp() <<" Press Switch: P1" << endl;
54.         switch_p1 = 1;    // Press P1
55.         RUN_SIM(5);
56.         cout << "@" << sc_time_stamp() <<" Release Switch: P1" << endl;
57.         switch_p1 = 0;    // Release P1
58.
59.         RUN_SIM(10);
60.         cout << "@" << sc_time_stamp() <<" Press Switch: P2" << endl;
61.         switch_p2 = 1;    // Press P2
62.         RUN_SIM(5);
63.         cout << "@" << sc_time_stamp() <<" Release Switch: P2" << endl;
64.         switch_p2 = 0;    // Release P2
65.
66.         RUN_SIM(5);
67.         cout << "@" << sc_time_stamp() <<" Press Switch: P2" << endl;
68.         switch_p2 = 1;    // Press P2
69.         RUN_SIM(5);
70.         cout << "@" << sc_time_stamp() <<" Release Switch: P2" << endl;
71.         switch_p2 = 0;    // Release P2
72.
73.         RUN_SIM(10);
74.         cout << "@" << sc_time_stamp() <<" Press Switch: P2" << endl;
75.         switch_p2 = 1;    // Press P2
76.         RUN_SIM(5);
77.         cout << "@" << sc_time_stamp() <<" Release Switch: P2" << endl;
78.         switch_p2 = 0;    // Release P2
79.
80.         RUN_SIM(10);
81.         cout << "@" << sc_time_stamp() <<" Press Switch: P1" << endl;
82.         switch_p1 = 1;    // Press P1
83.         RUN_SIM(5);
84.         cout << "@" << sc_time_stamp() <<" Release Switch: P1" << endl;
85.         switch_p1 = 0;    // Release P1
86.
87.         RUN_SIM(10);
88.         cout << "@" << sc_time_stamp() <<" Press Switch: P2" << endl;
89.         switch_p2 = 1;    // Press P2
90.         RUN_SIM(5);
91.         cout << "@" << sc_time_stamp() <<" Release Switch: P2" << endl;
92.         switch_p2 = 0;    // Release P2
93.
94.         RUN_SIM(5);
```

```
95.         cout << "@" << sc_time_stamp() <<" Press Switch: P2" << endl;
96.         switch_p2 = 1;    // Press P2
97.         RUN_SIM(5);
98.         cout << "@" << sc_time_stamp() <<" Press Switch: P1" << endl;
99.         switch_p1 = 1;    // Press P1
100.           RUN_SIM(5);
101.           cout << "@" << sc_time_stamp() <<" Release Switch: P2" << endl;
102.           switch_p2 = 0;    // Release P2
103.           RUN_SIM(10);
104.           cout << "@" << sc_time_stamp() <<" Release Switch: P1" << endl;
105.           switch_p1 = 0;    // Release P1
106.
107.           RUN_SIM(10);
108.           cout << "@" << sc_time_stamp() <<" Press Switch: P1" << endl;
109.           switch_p1 = 1;    // Press P1
110.           RUN_SIM(5);
111.           cout << "@" << sc_time_stamp() <<" Release Switch: P1" << endl;
112.           switch_p1 = 0;    // Release P1
113.
114.           RUN_SIM(10);
115.           cout << "@" << sc_time_stamp() <<" Press Switch: P2" << endl;
116.           switch_p2 = 1;    // Press P2
117.           RUN_SIM(5);
118.           cout << "@" << sc_time_stamp() <<" Release Switch: P2" << endl;
119.           switch_p2 = 0;    // Release P2
120.
121.           RUN_SIM(5);
122.           cout << "@" << sc_time_stamp() <<" Press Switch: P2" << endl;
123.           switch_p2 = 1;    // Press P2
124.           RUN_SIM(5);
125.           cout << "@" << sc_time_stamp() <<" Press Switch: P1" << endl;
126.           switch_p1 = 1;    // Press P1
127.           RUN_SIM(5);
128.           cout << "@" << sc_time_stamp() <<" Release Switch: P1" << endl;
129.           switch_p1 = 0;    // Release P1
130.           RUN_SIM(10);
131.           cout << "@" << sc_time_stamp() <<" Release Switch: P2" << endl;
132.           switch_p2 = 0;    // Release P2
133.
134.           RUN_SIM(10);
135.           cout << "@" << sc_time_stamp() <<" Press Switch: P1" << endl;
136.           switch_p1 = 1;    // Press P1
137.           RUN_SIM(5);
138.           cout << "@" << sc_time_stamp() <<" Release Switch: P1" << endl;
139.           switch_p1 = 0;    // Release P1
```

```
140.          RUN_SIM(20);
141.
142.          sc_start(-1);
143.          return 0;// Terminate simulation
144.   }
145.
```