Project 3:
Elevator Controller & Simulator

Youngsoo Kang
Tommy Zarick

ECE3561: Advanced Digital Design
Professor Eylem Ekici
April 23 2018

Introduction:

This project involves implementing an elevator controller and an elevator simulator that will interface with each other and operate with specific inputs and outputs. This elevator controller should control the simulator synchronously for a 4 story building and should operate autonomously with only button inputs outside of each floor and button request inputs inside the elevator. The controller should control specific actions of the simulator such as moving the elevator up, down, opening the door, and closing the door.

Design:

The main task was to interface the controller ("Controller", B1) and the simulator ("Simulator", B6) in the correct way, with communication between them being smooth and responsive. The first step taken was to implement the simulator. This gives the implementation of the controller clearer interface points. The simulator is also more straight forward so getting it implemented gives a solid base to build upon. The simulator simply handles the inputs of EMVUP (move up), EMVDN (move down), ECLOSE (close door), and EOPEN (open door) and produces the correct EF floor output in addition to an ECOMP signal to let the controller know it is finished operating. Once the simulator was implemented, the controller could be developed in order to control the simulator. The controller's job is to receive 3 dynamically changing bus inputs and process them in order to produce the correct commands for the simulator block. In addition to this, the controller also outputs the FLOOR_IND bus directly as a status marker for the current floor. POC and SYSCLK are direct inputs to both the controller and the simulator. POC immediately resets the system to default 1st floor mode and SYSCLK is the CLK that the system is synchronized to.

In order to make the controller function smoothly and make the flow more logical, internal signals were used. Most notably, the DIR_CON (direction) signal was used in order to keep track of the elevator's direction (down, up, stationary). Also, the DOOR signal was used in order to keep track of whether the door is open or closed. This was necessary because the elevator should perform different actions depending on whether the door is open or closed and the simulator didn't feed this information to the controller by default. Also, delayCLK is used to delay the clock cycle for movement delay specification purposes. Finally, UP_QUEUE, DN_QUEUE, and GO_QUEUE are all internal bus signals that are used to store the up requests, down requests, and go requests. This is necessary because the requests are simply pulses and do not stay high after they are asserted. The queues store these requests and remember them in case of multiple requests that have to be executed where some requests will need to wait for others to finish first before executing. The requests in the queues must be cleared correctly in order for proper functionality. The respective request is cleared when the doors are opened. In addition to these, the inputs to the controller that are passed from the simulator are saved into the controller as internal signals in order to work with them and view them in the ISim simulator output for verification purposes. The internal signal DONE is also used in order to indicate that any one of the EMVUP, EMVDN, EOPEN, or ECLOSE signals is asserted. This feature was

implemented in order to make sure that these four actions are exclusive of each other per each ECOMP assertion. Therefore, only one of them is allowed to happen at a time.

The FLOOR_IND signal output is simply just a relay of the EF signal that is passed to the controller from the simulator.

One issue that was encountered is that signal value changes do not take effect within a single process block. Because of this, variables needed to be declared for the UP_QUEUE, DN_QUEUE, GO_QUEUE, and DIR_CON signals. These variables were named UP_VAR, DN_VAR, GO_VAR, and DIR_VAR.

The logic for the elevator controller is as follows:

First, POC is checked and if it is asserted, everything is cleared or set to default values. An assumption made here is that the default direction is stationary (“00”). After this, everything occurs synchronously, depending on SYSCLK. This system is active on the rising edge of the clock. SYSCLK is a 2Hz continuous signal. The controller then checks the ECOMP value in order to determine if the simulator is currently in action or not. The controller will continue processing in the event that the simulator is not currently functioning actively. Then, the controller checks if the direction needs to be changed. If there are only requests below the current floor, the direction changes to down. If there are only requests above the current floor, the direction is changed to up. Next the controller checks to see if the doors need to be closed. Next, the condition is handled for the case of if there is a request on the current floor. This means the elevator has arrived at the desired floor. Therefore, the doors are opened and that request is cleared from its respective queue. The Up and Down request queues are only checked if the elevator is going in the same direction as the queue whereas the Go request queue is checked regardless. Because this case means that people should be exiting the elevator, the elevator should be halted and direction should be set to stationary (“00”). In the case that there is no request on the current floor and the doors are closed, then the elevator continues moving in the previously set direction if the elevator is moving to an allowed floor. The controller’s internal outputs should always be cleared after it is asserted and processed. This allows for exclusivity of these signals (EMVUP, EMVDN, ECLOSE, EOPEN). When queueing the floor requests from the raw inputs UP_REQ, DN_REQ, and GO_REQ into their respective queues, some padding needs to take place. Because an up request is invalid on the 4th floor, the UP_QUEUE is only concerned with 3 least significant bits. However, in order for these queues to be processed smoothly and compared easily, they are all standardized to be 4 bit signals. Therefore, the UP_QUEUE is padded with one zero at the most significant bit. The DN_QUEUE is padded with one zero at the least significant bit because a down request at floor 1 is an invalid request. The GO_QUEUE is not padded because there can be a go request to any floor, no matter what.

Results:

The results were obtained simply by verifying the test bench simulations. Three different test benches were run in order to verify the functionality of the system. The first test was the test described in the project description which simply describes the input sequence as POC->UP_REQ(3)->UP_REQ(1)->DN_REQ(4)->GO_REQ(2)->FINISH each spaced by 1 second starting at second zero. The output can be viewed and verified in the ISim simulation output (Fig. 1, A1). The second test was a custom test with a routine as follows: POC[0sec]->DN_REQ(4)[1sec]->GO_REQ(1)[7sec]->FINISH. This elevator functionality output can be viewed and verified in the ISim simulation output (Fig. 2, A1). The third test routine included the input sequence POC[0sec]->UP_REQ(1)[1sec]->GO_REQ(3) and GO_REQ(4)[5sec]->DN_REQ(3)[20sec]->GO_REQ(1)[24sec]->FINISH. The output can be viewed and verified in the ISim simulation output (Fig. 3, A2). In order to verify these results, the timing specifications must hold true. These include the proper timing requirements for the door being open and the timing “requirements” for the elevator moving between floors.

Test Routine	**ISim Output**	**VHDL Routine**
1 (predefined by project description) *[POC->UP_REQ(3)->UP_REQ(1)->DN_REQ(4)->GO_REQ(2)->FINISH]*	Fig. 1, A1	“Test Bench 1”, B9
2 (custom) *[POC->DN_REQ(4)->GO_REQ(1)->FINISH]*	Fig. 2, A1	“Test Bench 2”, B12
3 (custom) *[POC->UP_REQ(1)->GO_REQ(3)->GO_REQ(4)->DN_REQ(3)->GO_REQ(1)->FINISH]*	Fig. 3, A2	“Test Bench 3”, B15

Comparisons:

At first, the output of the circuit was not the desired simulation output, as the elevator was not moving up and down correctly. This issue stemmed from the internal direction signal. The direction was not changing correctly or at correct times which in turn caused the request queues to not be cleared at the correct times which in turn caused the door to not open at the correct time. This error occured when the group was still using a simple binary representation for the DIR signal (up or down). This was fixed by changing the elevator logic and creating a DIR signal that encompasses 3 states (up, down, and stationary). This way, the functionality for servicing current floor requests is more robust. In addition, because the ECOMP signal lasts for one full clock cycle, the elevator seems to lag at first, but it is actually waiting the allotted time if the output is viewed as beginning right on the ECOMP edge. By the end of all troubleshooting, the output matched expectations.

Conclusions:

This project successfully implemented an elevator control system interfaced with an elevator simulator to simulate real outputs (because access to an actual elevator is limited). The group had to create the lower level entity as the simulator and tie it all together with the higher level entity as the controller. The group gained skills such as input/output logic design decision making, dealing with multiple entities and tying them into one, dealing with internal signals, troubleshooting and verifying a design via ISim simulation output, and further Xilinx computer aided design exposure using solely VHDL. The controller and simulator had to both be verified by test benches and the group also gained experience integrating this into the design/verification process. A few unexpected obstacles faced were navigating VHDL syntax and process boundaries, integrating the behavioral model and the simulation for all entities involved, knitting multiple pieces of VHDL architecture into one system all interfacing with each other, and troubleshooting invalid output, among others. By the end of the process, a fully functional elevator controller/simulator system was implemented.

Appendix A - Test Bench Outputs

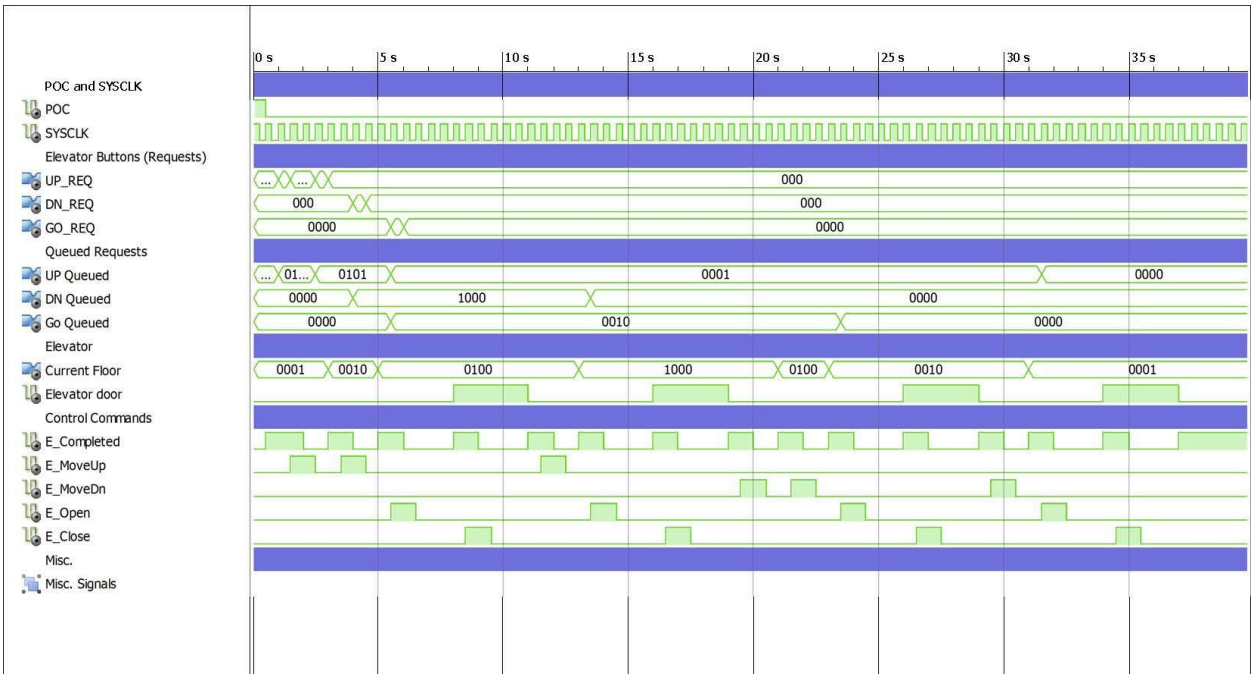

Figure 1: ISim Output for Test 1 (described in project description)

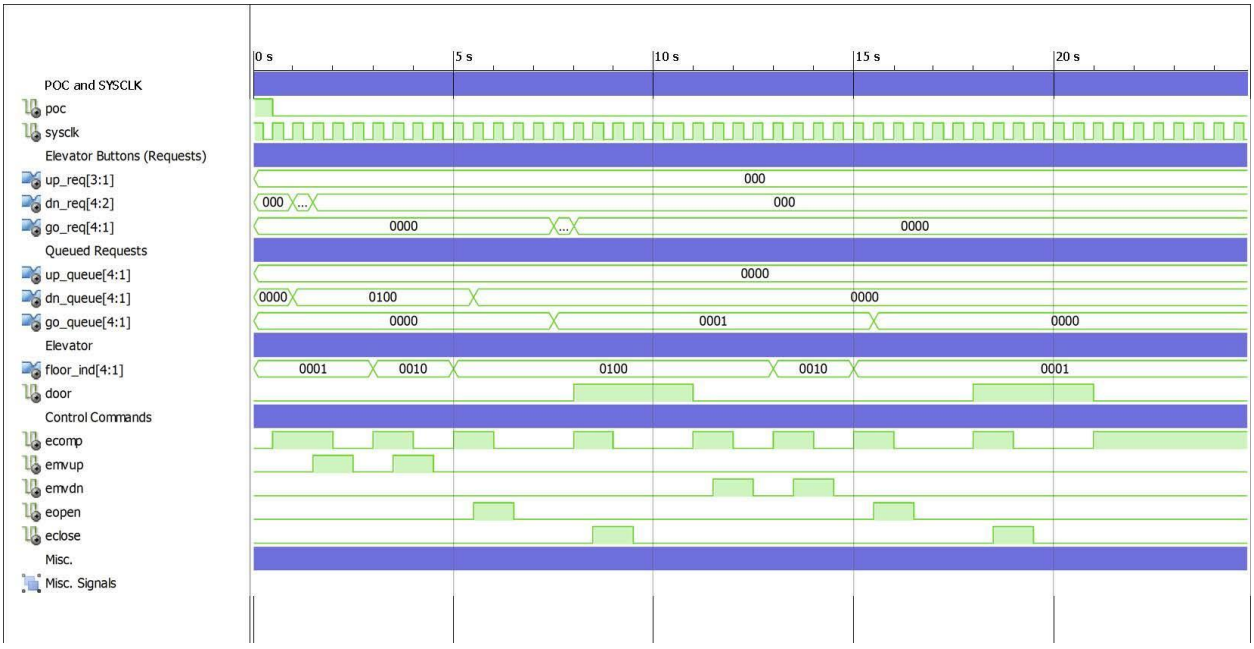

Figure 2: ISim Output for Test 2

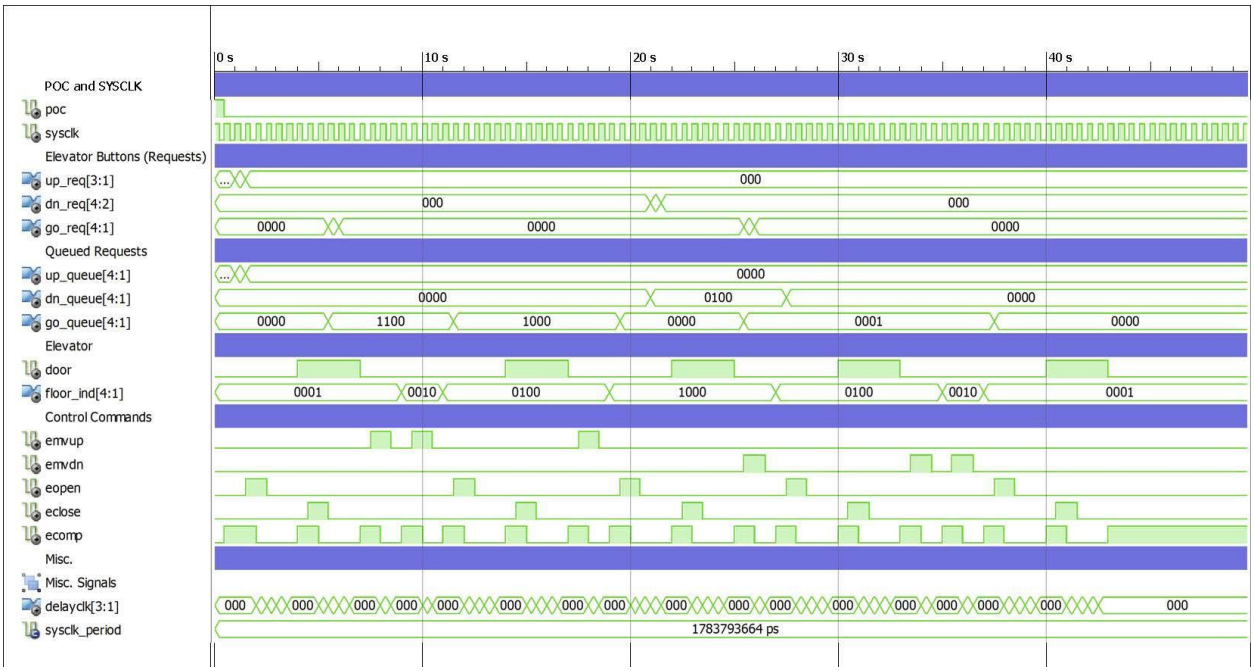

Figure 3: ISim Output for Test 3

Appendix B - VHDL code

Controller:


```
1  --------------------------------------------------------------------------------
2  -- ECE 3561 Project 3: Elevator
3  --
4  -- Project Name: Elevator Controller
5  --
6  -- Author: Youngsoo Kang
7  -- Author: Tommy Zarick
8  --
9  -- NOTE1: UP_REQ, DN_REQ, GO_REQ are asserted only if the button is pushed. When
10 --          button is released, the signal will be cleared.
11 -- NOTE2: Reqeust in queue will be cleared when elevator is opening the doors.
12 -- NOTE3: Controller's output to simulator (EMVUP, EMVDN, EOPEN, ECLOSE) will
13 --          be exclusive to each other in terms of assertion.
14 --------------------------------------------------------------------------------
15
16 library IEEE;
17 use IEEE.std_logic_1164.ALL;
18 use IEEE.NUMERIC_STD.ALL;
19
20 entity ElevatorController is
21    port (
22
23       -- External signals
24       POC : in std_logic;
25       SYSCLK : in std_logic;
26       UP_REQ : in std_logic_vector (3 downto 1);
27       DN_REQ : in std_logic_vector (4 downto 2);
28       GO_REQ : in std_logic_vector (4 downto 1);
29       FLOor_IND : out std_logic_vector (4 downto 1);
30
31       -- Internal signals that needs to be shown in ISim simulation
32       UP_QUEUE : buffer std_logic_vector (4 downto 1);
33       DN_QUEUE : buffer std_logic_vector (4 downto 1);
34       GO_QUEUE : buffer std_logic_vector (4 downto 1);
35       EMVUP : buffer std_logic;
36       EMVDN : buffer std_logic;
37       EOPEN : buffer std_logic;
38       ECLOSE : buffer std_logic;
39       ECOMP : buffer std_logic;
40       EF : buffer std_logic_vector (4 downto 1);
41
42       -- Status of elevator door: 0 = closed, 1 = opened
43       DOOR : buffer std_logic;
44
45       -- Delaying CLK cycle for specification of movement delay
46       delayCLK : buffer std_logic_vector (3 downto 1)
47    );
48 end ElevatorController;
49
50 architecture Behavioral of ElevatorController is
51
52    -- Importing Elevator Simulator
53    component ElevatorSimulator is
54       port (
55          POC : in  std_logic;
56          SYSCLK : in  std_logic;
57          EMVUP : in  std_logic;
```

```
58            EMVDN : in  std_logic;
59            EOPEN : in  std_logic;
60            ECLOSE : in  std_logic;
61            ECOMP : buffer std_logic;
62            EF : buffer  std_logic_vector (4 downto 1);
63            delayCLK : buffer std_logic_vector (3 downto 1);
64            DOOR : buffer std_logic
65         );
66      end component;
67
68      -- Convert one-hot assigned vector to the integer index of the asserted bit.
69      -- If vector is not one-hot assigned, result is undefined.
70      -- Note that compiler may give warning of this, but this function's limited
71      --    cases of returned integer will prevent any run-time error.
72      function OneHotVectorToIndex (
73         VEC_ARG : in std_logic_vector(4 downto 1))
74         return std_logic_vector is
75         variable result : std_logic_vector(3 downto 1);
76         begin
77            if VEC_ARG(4)='1' then
78               result := "100";
79            elsif VEC_ARG(3)='1' then
80               result := "011";
81            elsif VEC_ARG(2)='1' then
82               result := "010";
83            elsif VEC_ARG(1)='1' then
84               result := "001";
85            end if;
86         return std_logic_vector(result);
87      end;
88
89      -- Indicates one of EMVUP, EMVDN, EOPEN, ECLOSE is asserted.
90      -- This is to make any one of four actions exclusive per ECOMP asserted.
91      -- 0 for action not processed, 1 for action processed.
92      signal DONE : std_logic;
93
94      -- Internal signal to keep track of direction and movement.
95      -- 00 for halt, 01 for down, 10 for up.
96      signal DIR_CON : std_logic_vector(2 downto 1);
97
98   begin
99
100     U1 : ElevatorSimulator port map (
101        POC => POC,
102        SYSCLK => SYSCLK,
103        EMVUP => EMVUP,
104        EMVDN => EMVDN,
105        EOPEN => EOPEN,
106        ECLOSE => ECLOSE,
107        ECOMP => ECOMP,
108        EF => EF,
109        delayCLK => delayCLK,
110        DOOR => DOOR
111     );
112
113     -- Relaying current floor information to output directly
114     FLOor_IND <= EF;
```

```
115
116     process(SYSCLK, POC)
117
118        -- Since signal values does not take effect of changes within
119        --    a single process run but at the end of process,
120        --    internal variable values for certain signal values are needed for
121        --    to use it in conditional statements that needs updated values.
122        variable UP_VAR : std_logic_vector(4 downto 1);
123        variable DN_VAR : std_logic_vector(4 downto 1);
124        variable GO_VAR : std_logic_vector(4 downto 1);
125        variable DIR_VAR : std_logic_vector(2 downto 1);
126
127     begin
128
129        -- Asynchronous power-on-clear
130        if (POC='1') then
131           EMVUP<='0';
132           EMVDN<='0';
133           EOPEN<='0';
134           ECLOSE<='0';
135           DIR_CON<="00";
136           UP_QUEUE <= "0000";
137           DN_QUEUE <= "0000";
138           GO_QUEUE <= "0000";
139           DONE<='0';
140
141        -- Process elevator controller synchronously on active-high edge
142        elsif (SYSCLK'event and SYSCLK='1') then
143
144           -- Assign signal values that are used for conditional
145           --    statements to internal variables.
146           UP_VAR := UP_QUEUE;
147           DN_VAR := DN_QUEUE;
148           GO_VAR := GO_QUEUE;
149           DIR_VAR := DIR_CON;
150
151           -- When simulator is not in action...
152           if (ECOMP='1') then
153
154              -- Check if direction needs to be changed:
155              -- Change to down direction if there are requests only below current floor.
156              if (  (UP_VAR/="0000" and (UP_VAR < EF)) or
157                    (DN_VAR/="0000" and (DN_VAR <= EF)) or
158                    (GO_VAR/="0000" and (GO_VAR < EF)) ) and DIR_VAR/="01" then
159                 DIR_VAR:="01";
160
161              -- Change to up direction if there are requests only above current floor.
162              elsif (  (UP_VAR/="0000" and (UP_VAR >= EF)) or
163                       (DN_VAR/="0000" and (DN_VAR > EF)) or
164                       (GO_VAR/="0000" and (GO_VAR > EF)) ) and DIR_VAR/="10" then
165                 DIR_VAR:="10";
166
167              end if;
168
169              -- if doors are open...
170              if DOOR='1' and DONE='0' then
171
```

```
172                  -- Close the doors no matter what.
173                  DIR_VAR:="00";
174                  ECLOSE<='1';
175                  DONE<='1';
176
177               -- if doors are closed...
178               else
179
180                  -- Determine if there is any request on current floor.
181                  -- if there is, clear requests of current floor and open the doors.
182                  -- Note that Go reqeust is checked no matter what, and
183                  --    Up & Down requests are checked only if it's the same direction
     with
184                  --    the direction where the elevator was moving.
185                  if (  (GO_VAR and EF)/="0000" or
186                        (DIR_VAR="01" and (DN_VAR and EF)/="0000") or
187                        (DIR_VAR="10" and (UP_VAR and EF)/="0000" ) ) then
188
189                     -- Clear go request of current floor no matter what
190                     GO_VAR(to_integer(unsigned(OneHotVectorToIndex(EF)))) := '0';
191
192                     -- Clear down request of current floor only when elevator was going
     down.
193                     if (DIR_VAR="01") then
194                        DN_VAR(to_integer(unsigned(OneHotVectorToIndex(EF)))) := '0';
195                     end if;
196
197                     -- Clear up request of current floor only when elevator was going up.
198                     if (DIR_VAR="10") then
199                        UP_VAR(to_integer(unsigned(OneHotVectorToIndex(EF)))) := '0';
200                     end if;
201
202                     -- Since current floor is being served, halt the elevator.
203                     DIR_VAR:="00";
204
205                     -- If no other action has been taken within a ECOMP cycle...
206                     if DONE='0' then
207                        EOPEN<='1';
208                        DONE<='1';
209                     end if;
210
211                  -- if there is no any request on current floor...
212                  else
213
214                     -- if the doors are closed and there are no request on the current
     floor,
215                     --    then continue moving in the previous set direction only if
216                     --    the elevator is moving to allowed floor.
217                     if (DIR_VAR="01" and EF/="0001" and DONE='0') then
218                        EMVDN<='1';
219                        DONE<='1';
220                     elsif (DIR_VAR="10" and EF/="1000" and DONE='0') then
221                        EMVUP<='1';
222                        DONE<='1';
223                     end if;
224
225                  end if;
```

```
226               end if;
227
228            -- When simulator is in action (ECOMP='0')...
229            else
230
231               -- ECOMP cycle has been finished.
232               DONE <= '0';
233
234               -- Controller's internal outputs will always be negated right after
235               --    it's asserted and processed. This is to make following signals
236               --    be exclusive to each other.
237               EMVUP<='0';
238               EMVDN<='0';
239               EOPEN<='0';
240               ECLOSE<='0';
241
242            end if;
243
244            -- Queueing floor requests. All three vectors are padded accordingly
245            --    to match its floor indications.
246            UP_QUEUE <= UP_VAR or ('0' & UP_REQ);
247            DN_QUEUE <= DN_VAR or (DN_REQ & '0');
248            GO_QUEUE <= GO_VAR or GO_REQ;
249            DIR_CON <= DIR_VAR;
250
251         end if;
252
253      end process;
254
255   end Behavioral;
256
257
```

Simulator:


```
1   --------------------------------------------------------------------------------
2   -- ECE 3561 Project 3: Elevator
3   --
4   -- Project Name: Elevator Simulator
5   --
6   -- Author: Youngsoo Kang
7   -- Author: Tommy Zarick
8   --
9   -- NOTE1: since there is no way to measure real time, clock will be used to give
10  --          delay to this circuit by counting numbers of clock cycles,
11  --          instead of measuring real seconds.
12  -- NOTE2: changes to elevator state will take effect after the delay.
13  --------------------------------------------------------------------------------
14
15  library IEEE;
16  use IEEE.STD_LOGIC_1164.ALL;
17  use IEEE.NUMERIC_STD.ALL;
18
19  entity ElevatorSimulator is
20     port (
21
22        -- External signals
23        POC : in  STD_LOGIC;
24        SYSCLK : in  STD_LOGIC;
25        EMVUP : in  STD_LOGIC;
26        EMVDN : in  STD_LOGIC;
27        EOPEN : in  STD_LOGIC;
28        ECLOSE : in  STD_LOGIC;
29        ECOMP : buffer  STD_LOGIC;
30        EF : buffer STD_LOGIC_VECTOR (4 downto 1);
31
32        -- To count clock cycles for delaying the circuit.
33        -- Non-zero value will be assigned when delaying is requested.
34        delayCLK : buffer STD_LOGIC_VECTOR (3 downto 1);
35
36        -- Internal signal to keep track of door status
37        -- 0 for closed, 1 for opened.
38        DOOR : buffer std_logic
39     );
40
41  end ElevatorSimulator;
42
43  architecture Behavioral of ElevatorSimulator is
44
45
46     -- Internal signal to keep track of direction and movement.
47     -- 00 for halt, 01 for down, 10 for up.
48     signal DIR_SIM : std_logic_vector (2 downto 1);
49
50     -- Internal signals to keep track of action received.
51     signal EOPEN_INT : std_logic;
52     signal ECLOSE_INT : std_logic;
53     signal EMVUP_INT : std_logic;
54     signal EMVDN_INT : std_logic;
55
56  begin
57
```

```
58      process(SYSCLK, POC)
59      begin
60
61         -- Asynchronous power-on-clear
62         if (POC='1') then
63            EF<="0001";
64            ECOMP<='0';
65            DOOR<='0';
66            delayCLK<="000";
67            EOPEN_INT<='0';
68            ECLOSE_INT<='0';
69            EMVUP_INT<='0';
70            EMVDN_INT<='0';
71            DIR_SIM<="00";
72
73         -- Process elevator simulator synchronously, with active-high edge
74         elsif (SYSCLK'event and SYSCLK='1') then
75
76            -- If the circuit is in action...
77            if (ECOMP='0') then
78
79               -- If the circuit is being delayed, decrement count and
80               --    skip this active-edge clock cycle.
81               if (unsigned(delayCLK) > 0) then
82                  delayCLK <= STD_LOGIC_VECTOR(unsigned(delayCLK) - 1);
83
84               -- If the circuit is in action and delay is over...
85               else
86
87                  -- If elevator was not at top floor and moving up...
88                  if (DIR_SIM="10" and EF/="1000" and EMVUP_INT<='1') then
89                     EMVUP_INT<='0';
90                     EF <= EF(3 downto 1) & '0';
91
92                  -- If elevator was not at bottom floor and moving down...
93                  elsif (DIR_SIM="01" and EF/="0001" and EMVDN_INT<='1') then
94                     EMVDN_INT<='0';
95                     EF <= '0' & EF(4 downto 2);
96
97                  -- If the door is closed and requested to be opened...
98                  elsif (DIR_SIM="00" and DOOR='0' and EOPEN_INT='1') then
99                     EOPEN_INT <= '0';
100                    DOOR<='1';
101
102                 -- If the door is opened and requested to be closed...
103                 elsif (DIR_SIM="00" and DOOR='1' and ECLOSE_INT='1') then
104                    ECLOSE_INT <= '0';
105                    DOOR<='0';
106
107                 end if;
108
109                 -- Simulator finished its action.
110                 ECOMP<='1';
111
112              end if;
113
114           -- If the circuit is not in action (ECOMP='1')...
```

```
115            else
116
117               -- Moving up takes 2 seconds.
118               -- This action is skipped if the elevator is
119               --    trying to move beyond allowed floor.
120               if  (EMVUP='1' and EF/="1000") then
121                  DIR_SIM<="10";
122                  EMVUP_INT<='1';
123                  ECOMP<='0';
124                  delayCLK <= "001";
125
126               -- Moving down takes 2 seconds.
127               -- This action is skipped if the elevator is
128               --    trying to move beyond allowed floor.
129               elsif (EMVDN='1' and EF/="0001") then
130                  DIR_SIM<="01";
131                  EMVDN_INT<='1';
132                  ECOMP<='0';
133                  delayCLK <= "001";
134
135               -- Opening doors takes 3 seconds.
136               -- This action is skipped if the doors are already opened.
137               elsif (EOPEN='1' and DOOR='0') then
138                  DIR_SIM<="00";
139                  EOPEN_INT <= '1';
140                  ECOMP<='0';
141                  delayCLK <= "011";
142
143               -- Closing doors takes 3 seconds.
144               -- This action is skipped if the doors are already closed.
145               elsif (ECLOSE='1' and DOOR='1') then
146                  DIR_SIM<="00";
147                  ECLOSE_INT <= '1';
148                  ECOMP<='0';
149                  delayCLK <= "011";
150
151               end if;
152
153            -- If no valid action is requested, skip current clock cycle.
154
155            end if;
156         end if;
157
158      end process;
159
160   end Behavioral;
161
162
```

Test Routines:

Test Bench 1


```
1   --------------------------------------------------------------------------------
2   -- *** REQUIRED TEST: Shown in the project description ***
3   --
4   -- ECE 3561 Project 3: Elevator
5   --
6   -- Test Bench Name: ElevatorTestOFFICIAL
7   --
8   -- Author: Youngsoo Kang
9   -- Author: Tommy Zarick
10  --
11  --------------------------------------------------------------------------------
12
13  LIBRARY ieee;
14  USE ieee.std_logic_1164.ALL;
15
16  ENTITY ElevatorTestOFFICIAL IS
17  END ElevatorTestOFFICIAL;
18
19  ARCHITECTURE behavior OF ElevatorTestOFFICIAL IS
20
21     -- Component Declaration for the Unit Under Test (UUT)
22     COMPONENT ElevatorController
23     PORT(
24        UP_REQ : IN  std_logic_vector(3 downto 1);
25        DN_REQ : IN  std_logic_vector(4 downto 2);
26        GO_REQ : IN  std_logic_vector(4 downto 1);
27        POC : IN  std_logic;
28        SYSCLK : IN  std_logic;
29        FLOOR_IND : OUT  std_logic_vector(4 downto 1);
30        UP_QUEUE : buffer std_logic_vector (4 downto 1);
31        DN_QUEUE : buffer std_logic_vector (4 downto 1);
32        GO_QUEUE : buffer std_logic_vector (4 downto 1);
33        EMVUP : buffer STD_LOGIC;
34        EMVDN : buffer STD_LOGIC;
35        EOPEN : buffer STD_LOGIC;
36        ECLOSE : buffer STD_LOGIC;
37        ECOMP : buffer STD_LOGIC;
38        EF : buffer STD_LOGIC_VECTOR (4 downto 1);
39        DOOR : buffer std_logic;
40        delayCLK : buffer std_logic_vector (3 downto 1)
41     );
42     END COMPONENT;
43
44
45     -- Signals to be simulated.
46     signal POC : std_logic := '0';
47     signal SYSCLK : std_logic := '0';
48     signal UP_REQ : std_logic_vector(3 downto 1) := (others => '0');
49     signal DN_REQ : std_logic_vector(4 downto 2) := (others => '0');
50     signal GO_REQ : std_logic_vector(4 downto 1) := (others => '0');
51     signal UP_QUEUE : std_logic_vector (4 downto 1);
52     signal DN_QUEUE : std_logic_vector (4 downto 1);
53     signal GO_QUEUE : std_logic_vector (4 downto 1);
54     signal FLOOR_IND : std_logic_vector(4 downto 1);
55     signal EF : STD_LOGIC_VECTOR (4 downto 1);
56     signal EMVUP : STD_LOGIC;
57     signal EMVDN : STD_LOGIC;
```

```
58      signal EOPEN : STD_LOGIC;
59      signal ECLOSE : STD_LOGIC;
60      signal ECOMP : STD_LOGIC;
61      signal DOOR : STD_LOGIC;
62      signal delayCLK : std_logic_vector (3 downto 1);
63
64      -- Clock period definitions
65      constant SYSCLK_period : time := 500 ms;
66
67   BEGIN
68
69      -- Instantiate the Unit Under Test (UUT)
70      uut: ElevatorController PORT MAP (
71         POC => POC,
72         SYSCLK => SYSCLK,
73         UP_REQ => UP_REQ,
74         DN_REQ => DN_REQ,
75         GO_REQ => GO_REQ,
76         FLOOR_IND => FLOOR_IND,
77         GO_QUEUE => GO_QUEUE,
78         DN_QUEUE => DN_QUEUE,
79         UP_QUEUE => UP_QUEUE,
80         EMVUP => EMVUP,
81         EMVDN => EMVDN,
82         EOPEN => EOPEN,
83         ECLOSE => ECLOSE,
84         ECOMP => ECOMP,
85         EF => EF,
86         DOOR => DOOR,
87         delayCLK => delayCLK
88      );
89
90      -- Clock process definitions
91      SYSCLK_process :process
92      begin
93         SYSCLK <= '1';
94         wait for SYSCLK_period/2;
95         SYSCLK <= '0';
96         wait for SYSCLK_period/2;
97      end process;
98
99      -- POC_Assertion at the beginning
100     pwr_on_clr: process
101     begin
102        POC <= '1';
103        wait for SYSCLK_period;
104        POC <= '0';
105        wait;
106     end process;
107
108     -- Test Routine
109     test_routine: process
110     begin
111        wait for 1 sec;
112        UP_REQ <= "100";
113        wait for SYSCLK_PERIOD;
114        UP_REQ <= "000";
```

```
115
116         wait for 1 sec;
117         UP_REQ <= "001";
118         wait for SYSCLK_PERIOD;
119         UP_REQ <= "000";
120
121         wait for 1 sec;
122         DN_REQ <= "100";
123         wait for SYSCLK_PERIOD;
124         DN_REQ <= "000";
125
126         wait for 1 sec;
127         GO_REQ <= "0010";
128         wait for SYSCLK_PERIOD;
129         GO_REQ <= "0000";
130
131         wait;
132      end process;
133
134   END;
135
```

Test Bench 2


```
1   --------------------------------------------------------------------------------
2   -- ECE 3561 Project 3: Elevator
3   --
4   -- Test Bench Name: ElevatorTestRoutine1
5   --
6   -- Author: Youngsoo Kang
7   -- Author: Tommy Zarick
8   --
9   -- Test Routine:
10  -- RESET > Dn@4 > Pick@4 > Go@1 > Drop@1 > FINISH
11  --
12  -- Legends: Dn@#:    DN_REQ with #bit asserted
13  --          Up@#:    UP_REQ with #bit asserted
14  --          Go@#: GO_REQ with #bit asserted
15  --          Pick@#:  EOPEN & ECLOSE executed
16  --          Drop@#:  EOPEN & ECLOSE executed
17  --          RESET:   Initial State
18  --          FINISH:  End of Test Bench
19  --------------------------------------------------------------------------------
20
21  LIBRARY ieee;
22  USE ieee.std_logic_1164.ALL;
23
24  ENTITY ElevatorTestRoutine1 IS
25  END ElevatorTestRoutine1;
26
27  ARCHITECTURE behavior OF ElevatorTestRoutine1 IS
28
29     -- Component Declaration for the Unit Under Test (UUT)
30     COMPONENT ElevatorController
31     PORT(
32        UP_REQ : IN  std_logic_vector(3 downto 1);
33        DN_REQ : IN  std_logic_vector(4 downto 2);
34        GO_REQ : IN  std_logic_vector(4 downto 1);
35        POC : IN  std_logic;
36        SYSCLK : IN  std_logic;
37        FLOOR_IND : OUT  std_logic_vector(4 downto 1);
38        UP_QUEUE : buffer std_logic_vector (4 downto 1);
39        DN_QUEUE : buffer std_logic_vector (4 downto 1);
40        GO_QUEUE : buffer std_logic_vector (4 downto 1);
41        EMVUP : buffer STD_LOGIC;
42        EMVDN : buffer STD_LOGIC;
43        EOPEN : buffer STD_LOGIC;
44        ECLOSE : buffer STD_LOGIC;
45        ECOMP : buffer STD_LOGIC;
46        EF : buffer STD_LOGIC_VECTOR (4 downto 1);
47        DOOR : buffer std_logic;
48        delayCLK : buffer std_logic_vector (3 downto 1)
49     );
50     END COMPONENT;
51
52
53     -- Signals to be simulated.
54     signal POC : std_logic := '0';
55     signal SYSCLK : std_logic := '0';
56     signal UP_REQ : std_logic_vector(3 downto 1) := (others => '0');
57     signal DN_REQ : std_logic_vector(4 downto 2) := (others => '0');
```

```
58      signal GO_REQ : std_logic_vector(4 downto 1) := (others => '0');
59      signal UP_QUEUE : std_logic_vector (4 downto 1);
60      signal DN_QUEUE : std_logic_vector (4 downto 1);
61      signal GO_QUEUE : std_logic_vector (4 downto 1);
62      signal FLOOR_IND : std_logic_vector(4 downto 1);
63      signal EF : STD_LOGIC_VECTOR (4 downto 1);
64      signal EMVUP : STD_LOGIC;
65      signal EMVDN : STD_LOGIC;
66      signal EOPEN : STD_LOGIC;
67      signal ECLOSE : STD_LOGIC;
68      signal ECOMP : STD_LOGIC;
69      signal DOOR : STD_LOGIC;
70      signal delayCLK : std_logic_vector (3 downto 1);
71
72      -- Clock period definitions
73      constant SYSCLK_period : time := 500 ms;
74
75  BEGIN
76
77      -- Instantiate the Unit Under Test (UUT)
78      uut: ElevatorController PORT MAP (
79         POC => POC,
80         SYSCLK => SYSCLK,
81         UP_REQ => UP_REQ,
82         DN_REQ => DN_REQ,
83         GO_REQ => GO_REQ,
84         FLOOR_IND => FLOOR_IND,
85         GO_QUEUE => GO_QUEUE,
86         DN_QUEUE => DN_QUEUE,
87         UP_QUEUE => UP_QUEUE,
88         EMVUP => EMVUP,
89         EMVDN => EMVDN,
90         EOPEN => EOPEN,
91         ECLOSE => ECLOSE,
92         ECOMP => ECOMP,
93         EF => EF,
94         DOOR => DOOR,
95         delayCLK => delayCLK
96      );
97
98      -- Clock process definitions
99      SYSCLK_process :process
100     begin
101        SYSCLK <= '1';
102        wait for SYSCLK_period/2;
103        SYSCLK <= '0';
104        wait for SYSCLK_period/2;
105     end process;
106
107     -- POC_Assertion at the beginning
108     pwr_on_clr: process
109     begin
110        POC <= '1';
111        wait for SYSCLK_period;
112        POC <= '0';
113        wait;
114     end process;
```

```
115
116      -- Test Routine
117      test_routine: process
118      begin
119
120         wait for 1 sec;
121         DN_REQ <= "010";
122         wait for SYSCLK_period;
123         DN_REQ <= "000";
124
125         wait for 6 sec;
126         GO_REQ <= "0001";
127         wait for SYSCLK_period;
128         GO_REQ <= "0000";
129         wait;
130      end process;
131
132   END;
133
```

Test Bench 3


```
1   --------------------------------------------------------------------------------
2   -- ECE 3561 Project 3: Elevator
3   --
4   -- Test Bench Name: ElevatorTestRoutine1
5   --
6   -- Author: Youngsoo Kang
7   -- Author: Tommy Zarick
8   --
9   -- Test Routine:
10  -- RESET > Up@1 > Pick@1 > Go@3&4 >
11  --    Dn@3 > Drop@3 > Drop@4 > Pick@3 >
12  --    Go@1 > Drop@1 > FINISH
13  --
14  -- Legends: Dn@#:    DN_REQ with #bit asserted
15  --          Up@#:    UP_REQ with #bit asserted
16  --          Go@#: GO_REQ with #bit asserted
17  --          Pick@#:  EOPEN & ECLOSE executed
18  --          Drop@#:  EOPEN & ECLOSE executed
19  --          RESET:   Initial State
20  --          FINISH:  End of Test Bench
21  --------------------------------------------------------------------------------
22
23  LIBRARY ieee;
24  USE ieee.std_logic_1164.ALL;
25
26  ENTITY ElevatorTestRoutine2 IS
27  END ElevatorTestRoutine2;
28
29  ARCHITECTURE behavior OF ElevatorTestRoutine2 IS
30
31     -- Component Declaration for the Unit Under Test (UUT)
32     COMPONENT ElevatorController
33     PORT(
34        UP_REQ : IN  std_logic_vector(3 downto 1);
35        DN_REQ : IN  std_logic_vector(4 downto 2);
36        GO_REQ : IN  std_logic_vector(4 downto 1);
37        POC : IN  std_logic;
38        SYSCLK : IN  std_logic;
39        FLOOR_IND : OUT  std_logic_vector(4 downto 1);
40        UP_QUEUE : buffer std_logic_vector (4 downto 1);
41        DN_QUEUE : buffer std_logic_vector (4 downto 1);
42        GO_QUEUE : buffer std_logic_vector (4 downto 1);
43        EMVUP : buffer STD_LOGIC;
44        EMVDN : buffer STD_LOGIC;
45        EOPEN : buffer STD_LOGIC;
46        ECLOSE : buffer STD_LOGIC;
47        ECOMP : buffer STD_LOGIC;
48        EF : buffer STD_LOGIC_VECTOR (4 downto 1);
49        DOOR : buffer std_logic;
50        delayCLK : buffer std_logic_vector (3 downto 1)
51     );
52     END COMPONENT;
53
54
55     -- Signals to be simulated.
56     signal POC : std_logic := '0';
57     signal SYSCLK : std_logic := '0';
```

```
58      signal UP_REQ : std_logic_vector(3 downto 1) := (others => '0');
59      signal DN_REQ : std_logic_vector(4 downto 2) := (others => '0');
60      signal GO_REQ : std_logic_vector(4 downto 1) := (others => '0');
61      signal UP_QUEUE : std_logic_vector (4 downto 1);
62      signal DN_QUEUE : std_logic_vector (4 downto 1);
63      signal GO_QUEUE : std_logic_vector (4 downto 1);
64      signal FLOOR_IND : std_logic_vector(4 downto 1);
65      signal EF : STD_LOGIC_VECTOR (4 downto 1);
66      signal EMVUP : STD_LOGIC;
67      signal EMVDN : STD_LOGIC;
68      signal EOPEN : STD_LOGIC;
69      signal ECLOSE : STD_LOGIC;
70      signal ECOMP : STD_LOGIC;
71      signal DOOR : STD_LOGIC;
72      signal delayCLK : std_logic_vector (3 downto 1);
73
74      -- Clock period definitions
75      constant SYSCLK_period : time := 500 ms;
76
77  BEGIN
78
79      -- Instantiate the Unit Under Test (UUT)
80      uut: ElevatorController PORT MAP (
81         POC => POC,
82         SYSCLK => SYSCLK,
83         UP_REQ => UP_REQ,
84         DN_REQ => DN_REQ,
85         GO_REQ => GO_REQ,
86         FLOOR_IND => FLOOR_IND,
87         GO_QUEUE => GO_QUEUE,
88         DN_QUEUE => DN_QUEUE,
89         UP_QUEUE => UP_QUEUE,
90         EMVUP => EMVUP,
91         EMVDN => EMVDN,
92         EOPEN => EOPEN,
93         ECLOSE => ECLOSE,
94         ECOMP => ECOMP,
95         EF => EF,
96         DOOR => DOOR,
97         delayCLK => delayCLK
98      );
99
100     -- Clock process definitions
101     SYSCLK_process :process
102     begin
103        SYSCLK <= '1';
104        wait for SYSCLK_period/2;
105        SYSCLK <= '0';
106        wait for SYSCLK_period/2;
107     end process;
108
109     -- POC_Assertion at the beginning
110     pwr_on_clr: process
111     begin
112        POC <= '1';
113        wait for SYSCLK_period;
114        POC <= '0';
```

```
115         wait;
116      end process;
117
118      -- Test Routine
119      test_routine: process
120      begin
121         wait for 1 sec;
122         UP_REQ <= "001";
123         wait for SYSCLK_PERIOD;
124         UP_REQ <= "000";
125
126         wait for 4 sec;
127         GO_REQ <= "1100";
128         wait for SYSCLK_PERIOD;
129         GO_REQ <= "0000";
130
131         wait for 1 sec;
132         DN_REQ <= "010";
133         wait for SYSCLK_PERIOD;
134         DN_REQ <= "000";
135
136         wait for 17 sec;
137         GO_REQ <= "0001";
138         wait for SYSCLK_PERIOD;
139         GO_REQ <= "0000";
140
141         wait;
142      end process;
143
144   END;
145
```