Visvesvaraya National Institute of Technology (VNIT), Nagpur

Digital Hardware Design (ECL313)

Project Report

Submitted by :
Rajesh Nagula (BT18ECE059)
Shruti Murarka (BT18ECE099)
Dhruvi Shah (BT18ECE115)
Semester VI

Group No (9)

Submitted to :
Dr. Anamika Singh and Dr. Neha Nawandar
(Course Instructors)
Department of Electronics and Communication Engineering,
VNIT Nagpur

Contents

Title of the Project : Elevator Control System

1.1 Problem Statement: :

Design a 4-storey elevator control system. The system takes real-time inputs from multiple users at multiple floors simultaneously. Real-time requests can be made from outside (for calling the lift on the floor) or from inside (indicating your destination floor). The system initially takes as inputs the current floor/ state of the lift, the requests at hand and how the lift is currently moving (upwards/ downwards) or what was the last motion of the lift (upwards/ downwards).

1.2 Theory:

Introduction

The aim of this project is to come up with a Elevator control system whose aim is to control the motion of the lift and efficiently handle the requests of the users. The elevator has to mainly consider three things:

1. The current position of the elevator.
2. The positions at which the users are requesting it's service.
3. The current direction of motion of lift.

Algorithm

The basic algorithm of our model is as follows.

1. The elevator first checks if its current direction of motion is up or down and whether there is a request at the current floor.
2. If there is a request at the current floor, then the elevator opens its door.
3. Otherwise, if the present motion is upwards and there are demands at floors higher than the current floor, it would continue in the same direction, that is upwards. Similarly, if the elevator's last motion was downward and there are demands at floors below the present level, the elevator would begin to move in the same direction (downwards).

Explanation of the code in brief

The inputs to the controller are:

- clock

- Reset: Lift moves to ground/1st floor.
- initial_up_down: initial motion of lift.
- initial_floor: initial/current floor of lift
- initial_request: initial request array of lift or calls for lift.

The outputs to the controller are:

- op_led: Indicates door of lift (1 = open, 0 = close)
- output_floor: Real-time floor movement.
- output_request: Real-time request updates.

In the architecture we have defined signals for keeping track of the requests and the floor as it changes. The process sensitivity list includes the clk and the reset, which means that whenever either of them is changed the process block will get executed. We have used a signal named flag so that we can update the request vector every time someone begins the lift for the first time. In the process it checks the current state of the lift and accordingly it decides its next motion according to the above algorithm.

1.3 Code:

```
--Behavioral model for real-time 4 floor elevator.
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity elevator is
   Port( clk : in STD_LOGIC;
         reset : in STD_LOGIC;
-- to bring back lift to default state(ground floor, halt)
         op_led : out STD_LOGIC;
-- output led - status of lift (led is on when lift is open else off)
         initial_up_down : in STD_LOGIC;
-- initial motion of lift
         initial_floor : in STD_LOGIC_VECTOR (3 downto 0);
-- initial floor of lift
                initial_request : in std_logic_vector(3 downto 0);
-- initial request array of lift
```

```
        output_request : out std_logic_vector(3 downto 0);
-- output floor shows the current floor of lift
                output_floor : out std_logic_vector(3 downto 0));
-- output request shows the current request status of lift
-- Note: in simulation we can't --modify it in real time,
-- but in real scenario if button pressed modifies the array.
end elevator;

architecture arch of elevator is

-- signals defined
signal request: std_logic_vector(3 downto 0);
signal floor :  STD_LOGIC_VECTOR (3 downto 0);
signal up_down :  STD_LOGIC;
signal flag: std_logic := '0';

begin


control : process(clk,reset)
begin

if reset='1' then
-- if reset lift goes to ground floor.
        floor<= "0001";
        op_led<='0';
        report "reset";

elsif rising_edge(clk) and  reset = '0' then

        if flag='0' then
        -- initial assignment to signals using given inputs
                request <= initial_request;
                floor <= initial_floor;
                up_down <=initial_up_down;
                op_led <='0';
        end if;

    report "entered";
        op_led<='0';
```

```
    -- case 4th floor
    if floor="1000" then
            report "4th floor";
            if request(3)='1' then

            -- if 4th floor request open lift
                    report "4th floor open";
                    request(3)<='0'; -- request removed once lift opened
                    op_led<='1'; -- indicates opening of lift

            elsif (up_down='0' and
            (request(2)='1' or request(1) = '1' or request(0) = '1')) then
-- case: lift moving downwards and downwards floor request available
                    floor<="0100";
            else
-- since fourth floor lift needs to move downwards
                    up_down<='0';
    end if;


-- case 3rd floor
    elsif floor="0100" then
            report "3rd floor";
            if request(2)='1' then

            -- if 3rd floor request open lift
                    report "3rd floor open";
                    request(2)<='0'; -- request removed once lift opened
                    op_led<='1';  -- indicates opening of lift
        elsif (up_down='0' and(request(1)='1' or request(0)='1') ) then
    -- case: lift moving downwards and downwards floor request available
                    floor<="0010";
            elsif(up_down='1' and (request(3)='1') ) then
    -- case: lift moving upwards and upwards floor request available
                    floor<="1000";
            else
    -- case: lift moving upwards and downwards floor requests available
    -- or vice versa
```

```
-- toggle movement of lift
                up_down<= not(up_down);
        end if;


-- case 2nd floor
elsif floor="0010" then
        report "2nd floor";
        if request(1)='1' then

        -- if 2nd floor request open lift
                report "2nd floor open";
                request(1)<='0'; -- request removed once lift opened
                op_led<='1';  -- indicates opening of lift
        elsif(up_down='0' and (request(0)='1') ) then
-- case: lift moving downwards and downwards floor request available
                floor<="0001";
        elsif(up_down='1' and (request(2)='1' or request(3)='1') ) then
-- case: lift moving upwards and upwards floor request available
                floor<="0100";
        else
-- case: lift moving upwards and downwards floor requests available
-- or vice versa
-- toggle movement of lift
                up_down<= not(up_down);
        end if;



-- case 1st/ground floor
elsif floor="0001" then
        report "1st floor";
        if request(0)='1' then
                report "1st floor open";

-- if 1st floor request open lift
                request(0)<='0'; -- request removed once lift opened
                op_led<='1';  -- indicates opening of lift
        elsif(up_down='1' and
        (request(2)='1' or request(1)='1' or request(3)='1') ) then
```

```
        -- case: lift moving upwards and upwards floor request available
                        floor<="0010";
                else
        -- since first/ground floor lift needs to move upwards
                        up_down<='1';
                end if;


         else
                report "Lift stuck";
        end if;
flag<='1';
end if;

output_floor <= floor;
output_request <= request;

end process control;
end architecture;
```

1.4 Testbench:

```
--Testbench for real-time 4 floor elevator.

library ieee;
use ieee.std_logic_1164.all;

entity elevator_tb is
end elevator_tb;

architecture tb of elevator_tb is

component elevator
         port (  clk : in  STD_LOGIC;
                 reset : in  STD_LOGIC;
                 op_led : out STD_LOGIC;
                 initial_up_down : in STD_LOGIC;
                 initial_floor : in  STD_LOGIC_VECTOR (3 downto 0);
             initial_request : in std_logic_vector(3 downto 0);
                 output_request : out std_logic_vector(3 downto 0);
                 output_floor : out std_logic_vector(3 downto 0));
```

```
end component;

--inputs
signal clk : STD_LOGIC := '0';
signal reset : STD_LOGIC := '0';
signal initial_up_down : STD_LOGIC := '1';
signal initial_floor : STD_LOGIC_VECTOR (3 downto 0) := "0001";
signal initial_request : std_logic_vector (3 downto 0) := "0001";

--outputs
signal op_led :  STD_LOGIC ;
signal output_request : std_logic_vector(3 downto 0);
signal output_floor : std_logic_vector(3 downto 0);

constant clk_period : time := 100 ps;

begin
 uut : elevator
 port map(clk => clk,
            reset => reset,
                initial_floor =>  initial_floor,
                initial_request=> initial_request,
                initial_up_down => initial_up_down,
                op_led => op_led,
                output_floor => output_floor,
                output_request => output_request);
 process
 begin
         reset<='1';
         initial_request <= "1010";
         clk<='1';
         wait for clk_period;
         clk<='0';
         wait for clk_period;

         reset <= '0';
         initial_floor <= "0010";
         initial_request <= "0101";
         for i in 0 to 10 loop
                 clk<='1';
```

```
                wait for clk_period;
                clk<='0';
                wait for clk_period;
         end loop;

        reset <= '0';
        initial_floor <= "0100";
        initial_request <= "1101";
        for i in 0 to 10 loop
                clk<='1';
                wait for clk_period;
                clk<='0';
                wait for clk_period;
         end loop;

        reset <= '0';
        initial_floor <= "0010";
        initial_request <= "1101";
        for i in 0 to 10 loop
                clk<='1';
                wait for clk_period;
                clk<='0';
                wait for clk_period;
         end loop;

        reset <= '0';
        initial_floor <= "0001";
        initial_request <= "1110";
        for i in 0 to 10 loop
                clk<='1';
                wait for clk_period;
                clk<='0';
                wait for clk_period;
         end loop;

end process;

end tb;
```

1.5 RTL Schematic: Quartus II is used for RTL View of 4-storey elevator control system.

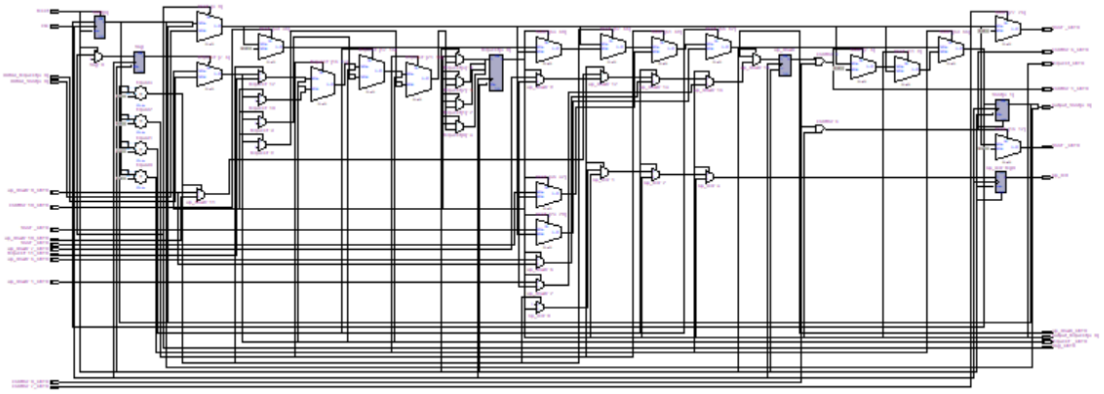


Figure 1: RTL View of the elevator control system

1.6 Simulation Waveform: ModelSim is used for Simulation of 4-storey elevator control system.

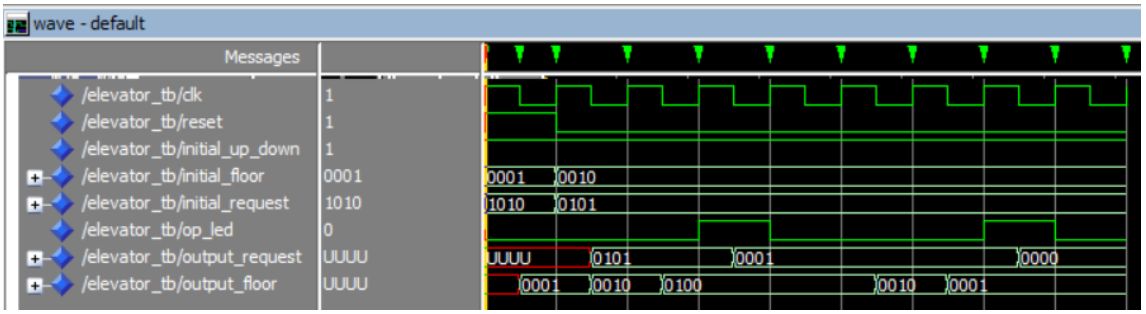
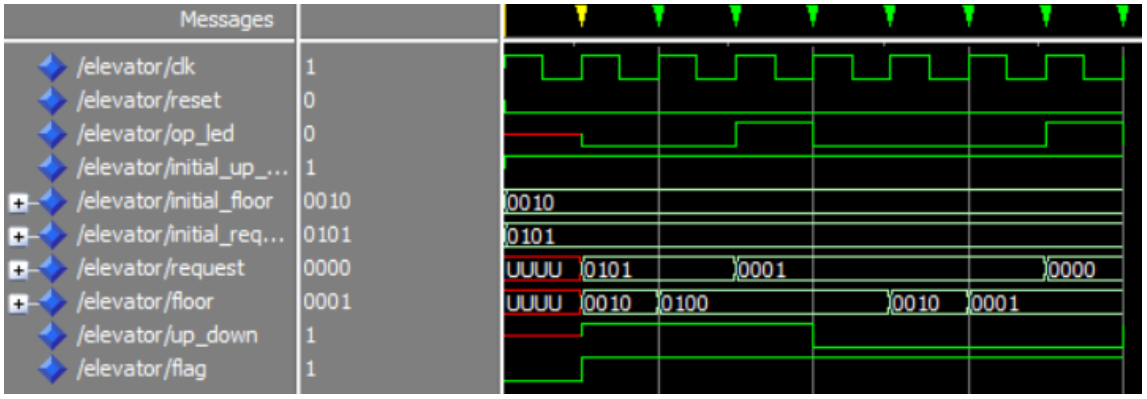
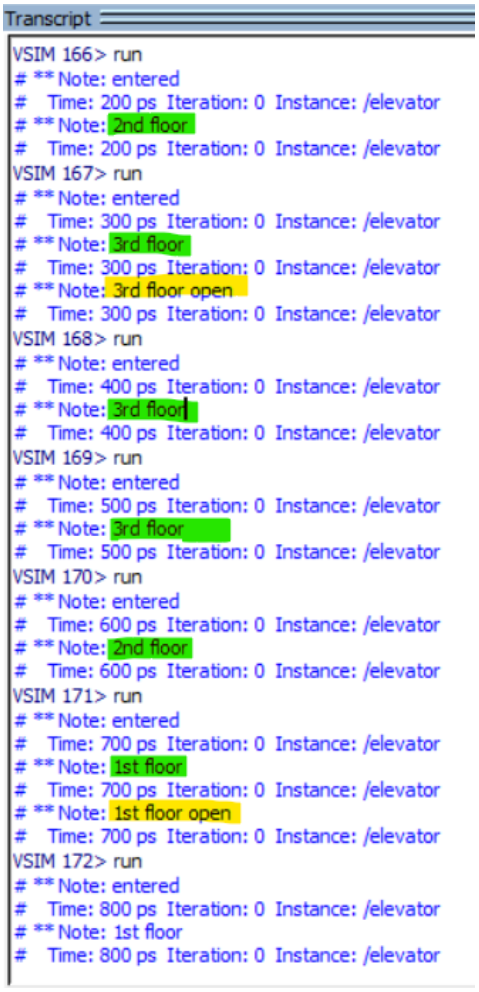


Figure 2: 4-storey elevator control system (using testbench)



(a) Simulation view of Real-time Elevator System



(b) Transcript of Real-time Elevator System.

Figure 3: 4-storey elevator control system (using force)

1.7 Conclusion: The working of the Elevator controller has been checked and verified using Modelsim and Quartus II softwares. The features of real-time elevator controller:

- In case the lift goes into a state other than the ones defined, than it reports : " Lift Stuck ".
- The buttons to request service from the lift are located on both sides, inside the lift for destination selection and on the floors to call the lift.
- A reset option is available. When the reset button is pressed, the lift returns to its default state, which is the first floor, and the default direction of the lift is 'up'.