**Written Project 1-Elevator**

**ECEN 424**

**Dr.Doss**

**By: Freddie Boadu**

**Table of Content**

Introduction……………………………………………………………………………………………………………………………………………3

Block diagram……………………………………………………………………………………………….…………………………………………3

Component description ………………………………………………………………………………………………………………………3-4

Explanation of block diagram changes………………………………………….………………………………………………………4-5

State diagram……………………………………………………………………………………………………………………………………….6

Pitfall and achievements…………………………………………………………………………………………………………………………6

Vhdl code and usf file…………………………………………………………………………………………………………………………7-23

Reference and conclusion………………………………………………………………………………………………………………………24

**Introduction**

This report is pertaining to the first project done in ECEN 424. This project was based around simulating an elevator. The elevator was to be able to go from floor to floor. As the elevator travels, it is to display each floor it passes by. If the elevator stops on a floor, the door is to open and start a timer that after 3 seconds is to close the door. The action of opening and closing is represented by the leds on the board. The elevator is also using a seven-segment display to display each floor as it passes by or lands on the floor. This design is to be able to reset the door timer every floor as it moves through the building.

**Block Diagram**

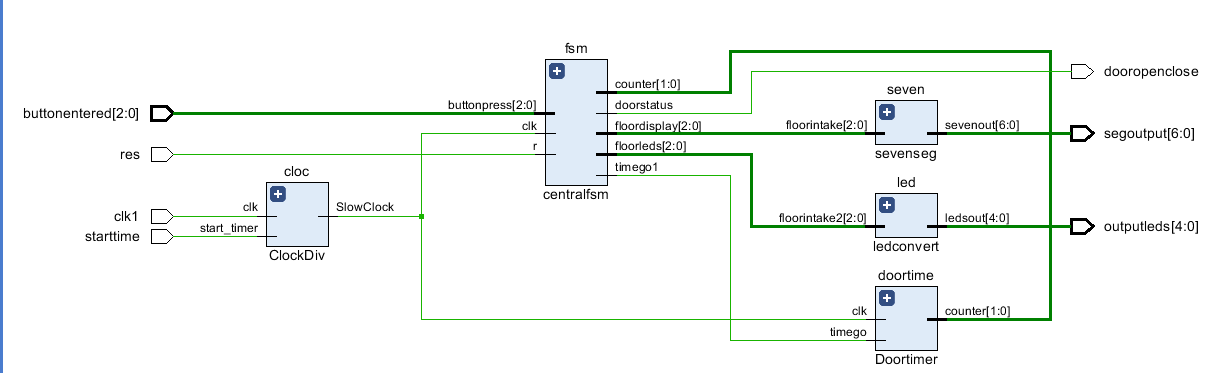
****

Figure 1.1 Block Diagram of the elevator design

**Component Description**

Clock divider: The clock divider has a clock signal, a user timer, led and a out signal to distribute a slow, medium or fast signal. The clock signal is tied to the board. The user timer will start the clock divider when called upon. This is connected to a switch for the design. The Led is used to time the clock and see the frequency at which the clock signal is creating an event.

Central Fsm: The finite state machine is the brains of the whole design. The finite state machine is taking in button presses that the user is doing externally on the board. This is coming from the switch inputs. Within the finite state machine, there is controller for the timer and moving from floor to floor. The finite state machine controls the timer of the door on each floor and controlling when the door is open or closed. The finite state machine also sends out signals to the LEDS to print out what floor the elevator is on, and also to the seven segments will display the current floor.

Seven coder: This is the seven segments of the device. When it gets a 3-bit code from the finite state machine to tell the coder what floor the elevator is current on, it projects onto the seven-segment display. The display also is showing as the elevator moves from floor to floor.

Door time: This component allows the door to stay open and then close after a certain amount of time. The time it will close is 3 seconds. After the timer gets a signal telling the timer to count or not, it will send the count to finite state machine to determine if it needs to stop or keep going.

Led coder: This component works like the seven-segment component of my model. This shows where the elevator is doing the whole process of going from floor to floor.

**Explanation of block diagram changes**

My block diagram changed drastically during the process of this project. I realized the importance of top down design and thinking about the project. When I initially made my diagram, I didn’t put into account the timer, or how I planned to keep the door open/closed for most of the project. As I started to design, I realized that needed to be added. I also got rid of my keypad. I realized that was an extra design that was not needed and as a result got ride of that for my whole design. Below will be the old and new block diagram together for comparison.

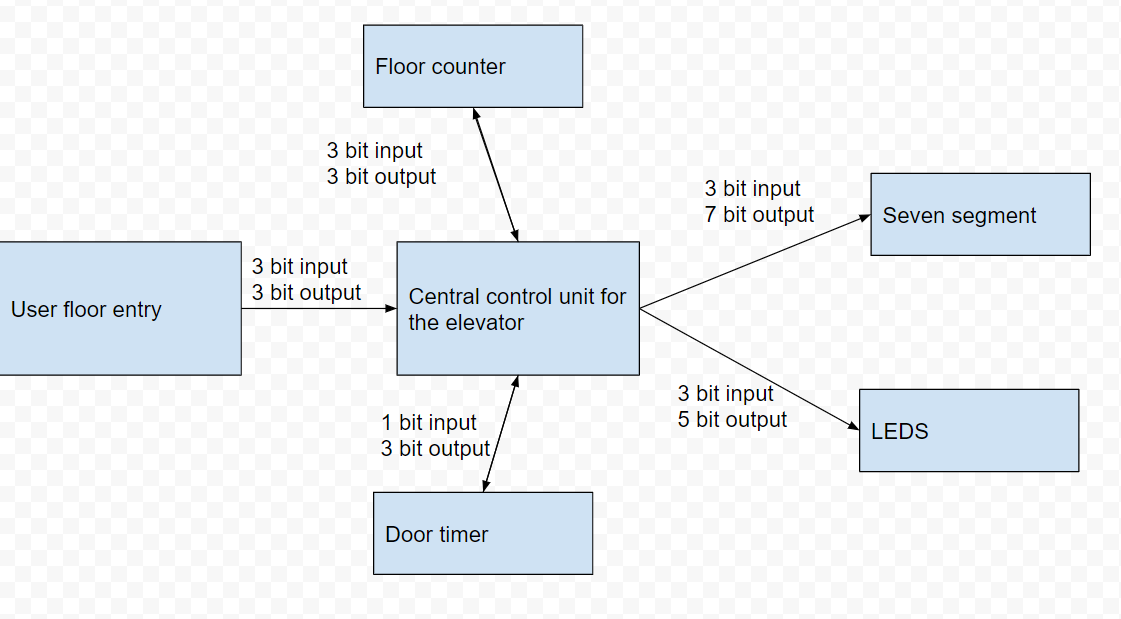


Figure 2- The first block diagram of the elevator design

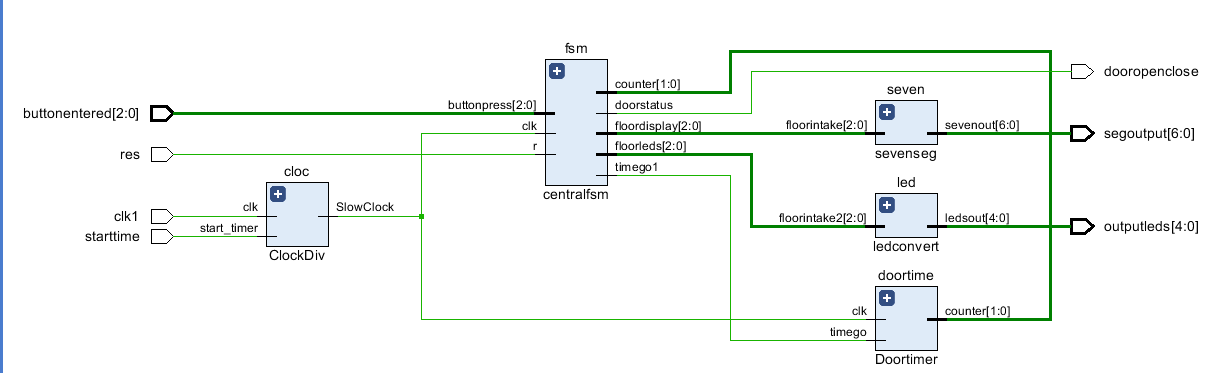
****

Figure 3- A repeat of figure 1

**State Diagram**

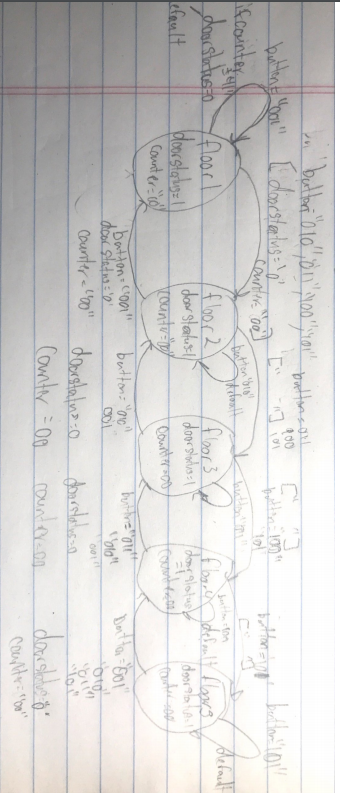


Figure 4- state diagram of the whole elevator system-handwritten

The state diagram consists of 5 states. Those 5 states represent each floor. The button represents a button being pressed. The movement of the state diagram is depicted from the arrows. When the same button is pressed the floor will remain the same. The state diagram also contains a door status for the door being open and closed, also a counter for determined whether the door remains open or closed as a result of the 3 second timer implemented in the design. If the button pressed is the same as the state, then the system is focused on the counter counting and closing the door when it reaches 3 seconds.

**Pitfalls and achievements**

During the design of this system, the idea of top down design was a huge achievement. The importance of learning and planning out details before fully implementing them is so important. It is an important component of systems engineering and exactly what is needed in the work world. Learning how to design the system according to the requirements before coding was extremely helpful. There was a failure with the timer which become the main pitfall of this project. The timer didn’t work to the degree wanted. It was extremely frustrated trying to work on the top level and the design and get the timer to feed back into finite state machine the way it should have. The design is correct but with more time and guidance I am sure the finite state machine could correctly read and implement the timer like desired.

**VHDL Code**

**Top level**

--top level

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity toplevel is

port( buttonentered : in std\_logic\_vector(2 downto 0);--button press on the elevator

clk1,res: in std\_logic;--clock and reset for the clock divider signals

starttime: in std\_logic;--start switch for the timer

outputleds: out std\_logic\_vector(4 downto 0);--leads that go out to tell the current floor

segoutput: out std\_logic\_vector(6 downto 0);--tells you the current floor you are on using the segment

dooropenclose:out std\_logic);-- get signals from doorstatus to determien if the door is closed or open

end;

architecture beh of toplevel is

signal clockfeeder: std\_logic;--clock tempo signal

signal ledsig: std\_logic\_vector(2 downto 0);

signal sevensig: std\_logic\_vector(2 downto 0);

signal timertime: std\_logic;

signal count: std\_logic\_vector(1 downto 0);

component ClockDiv port (

clk: in STD\_logic;

start\_timer: in STD\_logic;

FastClock , MediumClock , SlowClock , led0 : out STD\_LOGIC);

end component;

component centralfsm port (clk,r: in std\_logic;

buttonpress :in std\_logic\_vector(2 downto 0);--take in button values

doorstatus: out std\_logic;--0 if the door is closed and 1 if it is open

counter: buffer std\_logic\_vector(1 downto 0);--counter for the door to open and close

timego1: out std\_logic;--sends signal to start counting or stop

floordisplay: out std\_logic\_vector(2 downto 0);--output to the seven segment display

floorleds: out std\_logic\_vector(2 downto 0));--tells the current floor using the leds

end component;

component ledconvert

port (floorintake2: in std\_logic\_vector(2 downto 0);--gets 3 bit signal from the fsm to convert for the leds to tell the current floor

ledsout: out std\_logic\_vector(4 downto 0));

end component;

component sevenseg

port(floorintake: in std\_logic\_vector(2 downto 0);--gets 3 bit signal from the fsm to convert for the seven segment to tell the current floor

sevenout: out std\_logic\_vector(6 downto 0));

end component;

component Doortimer

port( clk,timego: in std\_logic;--clk from the divider and the signal for the timer to start for the door

counter: inout std\_logic\_vector(1 downto 0));--counter that is used to determine the counter getting to 3

end component;

begin

cloc : ClockDiv port map(clk=>clk1,start\_timer=>starttime,slowclock=>clockfeeder);

fsm: centralfsm port map(buttonpress=>buttonentered, clk=>clockfeeder,r=>res,floorleds=>ledsig,floordisplay=>sevensig,timego1=>timertime,counter=>count,doorstatus=>dooropenclose);

doortime: Doortimer port map(clk=> clockfeeder,timego=>timertime, counter=>count);

led:ledconvert port map(floorintake2=>ledsig,ledsout=>outputleds);

seven:sevenseg port map(floorintake=>sevensig,sevenout=>segoutput);

end;

**Door timer**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Doortimer is

port (clk,timego: in std\_logic;--clk from the divider and the signal for the timer to start for the door

counter: inout std\_logic\_vector(1 downto 0));--counter that is used to determine the counter getting to 3

end;

architecture beh of doortimer is

begin

process(clk,timego)

begin

if (clk' event and clk='1') then

if (timego='1') then

counter<=counter +1;--count to 3

elsif(timego='0') then

counter<="00";

end if;

end if;

end process;

end;

**Central fsm**

--central fsm for the elevator

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

use ieee.NUMERIC\_STD.all;

entity centralfsm is

port( clk,r: in std\_logic;

buttonpress :in std\_logic\_vector(2 downto 0);--take in button values

doorstatus: out std\_logic;--0 if the door is closed and 1 if it is open

counter: buffer std\_logic\_vector(1 downto 0);--counter for the door to open and close

timego1: out std\_logic;--sends signal to start counting or stop

floordisplay: out std\_logic\_vector(2 downto 0);--output to the seven segment display

floorleds: out std\_logic\_vector(2 downto 0));--tells the current floor using the leds

end;

architecture beh of centralfsm is

type currentfloor is (floor1, floor2, floor3,floor4, floor5);-- this will represent the floors

signal cfloor, nfloor:currentfloor;-- this will represent the current and next floor

signal floornumber: std\_logic\_vector(2 downto 0);

begin

process(clk,r)

begin

if (r='1') then

Cfloor<=floor1;

elsif(clk'event and clk='1') then

cfloor<=nfloor;

end if;

end process;

process(cfloor,nfloor,buttonpress)

begin

floorleds<=buttonpress;

floordisplay<= buttonpress;

doorstatus<='0';

timego1<='0';

case cfloor is

when floor1 =>--first floor

floorleds<="001";

floordisplay<="001";

if(buttonpress="001") then--current floor

nfloor<=cfloor;

elsif(buttonpress="010") then--move to floor 2

nfloor<=floor2;

elsif(buttonpress="011") then--move to floor 2

nfloor<=floor2;

elsif(buttonpress="100") then--move to floor 2

nfloor<=floor2;

elsif(buttonpress="101") then --move to floor 2

nfloor<=floor2;

else

nfloor<=Floor1;

end if;

--handle door commands

if (buttonpress="001" and counter /= "11") then--if the floor is 1 and the counter is not 3

timego1<='1';--send counter start signal

doorstatus<='1';--send counter stop signal

elsif(buttonpress="001" and counter ="11") then

timego1<='0';

doorstatus<='0';

end if;

when floor2=> --second floor

floorleds<="010";

floordisplay<="010";

if(buttonpress="001") then--move to floor 1

nfloor<=floor1;

elsif(buttonpress="010") then--current floor

nfloor<=cfloor;

elsif(buttonpress="011") then--move to floor 3

nfloor<=floor3;

elsif(buttonpress="100") then--move to floor 3

nfloor<=floor3;

elsif(buttonpress="101") then --move to floor 3

nfloor<=floor3;

else

nfloor<=floor2;

end if;

if (buttonpress="010" and counter /= "11") then

timego1<='1';

doorstatus<='1';

elsif(buttonpress="010" and counter ="11") then

timego1<='0';

doorstatus<='0';

end if;

when floor3=>

floorleds<="011";

floordisplay<="011";

if(buttonpress="001") then--move to floor 2

nfloor<=floor2;

elsif(buttonpress="010") then--move to floor 2

nfloor<=floor2;

elsif(buttonpress="011") then--current floor

nfloor<=cfloor;

elsif(buttonpress="100") then--move to floor 4

nfloor<=floor4;

elsif(buttonpress="101") then --move to floor 4

nfloor<=floor4;

else

nfloor<=floor3;

end if;

if (buttonpress="011" and counter /= "11") then

timego1<='1';

doorstatus<='1';

elsif(buttonpress="011" and counter ="11") then

timego1<='0';

doorstatus<='0';

end if;

when floor4=>

floorleds<="100";

floordisplay<="100";

if(buttonpress="001") then--move to floor 3

nfloor<=floor3;

elsif(buttonpress="010") then--move to floor 3

nfloor<=floor3;

elsif(buttonpress="011") then--move to floor 3

nfloor<=floor3;

elsif(buttonpress="100") then--current floor

nfloor<=cfloor;

elsif(buttonpress="101") then --move to floor 5

nfloor<=floor5;

else

nfloor<=Floor4;

end if;

if (buttonpress="100" and counter /= "11") then

timego1<='1';

doorstatus<='1';

elsif(buttonpress="100" and counter ="11") then

timego1<='0';

doorstatus<='0';

end if;

when floor5=>

floorleds<="101";

floordisplay<="101";

if(buttonpress="001") then--move to floor 4

nfloor<=floor4;

elsif(buttonpress="010") then--move to floor 4

nfloor<=floor4;

elsif(buttonpress="011") then--move to floor 4

nfloor<=floor4;

elsif(buttonpress="100") then--move to floor 4

nfloor<=floor4;

elsif(buttonpress="101") then --current floor

nfloor<=cfloor;

else

nfloor<=floor5;

end if;

if (buttonpress="101" and counter /= "11") then

timego1<='1';

doorstatus<='1';

elsif(buttonpress="101" and counter ="11") then

timego1<='0';

doorstatus<='0';

end if;

end case;

end process;

end;

**Ledconvert**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity ledconvert is

port( floorintake2: in std\_logic\_vector(2 downto 0);--gets 3 bit signal from the fsm to convert for the leds to tell the current floor

ledsout: out std\_logic\_vector(4 downto 0));

end;

architecture beh of ledconvert is

begin

process(floorintake2)

begin

Case floorintake2 is

when "001"=>ledsout<="00001";

when "010"=>ledsout<="00010";

when "011"=>ledsout<="00100";

when "100"=>ledsout<="01000";

when "101"=>ledsout<="10000";

when others=>ledsout<="00000";

end case;

end process;

end;

**Segconvert**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity sevenseg is

port( floorintake: in std\_logic\_vector(2 downto 0);--gets 3 bit signal from the fsm to convert for the seven segment to tell the current floor

sevenout: out std\_logic\_vector(6 downto 0));

end;

architecture beh of sevenseg is

begin

process(floorintake)

begin

Case floorintake is

when "001"=>sevenout<="1111001";-- change the display to 1

when "010"=>sevenout<="0100100";-- change the display to 2

when "011"=>sevenout<="0110000";-- change the display to 3

when "100"=>sevenout<="0011001";-- change the display to 4

when "101"=>sevenout<="0010010";-- change the display to 5

when others=>sevenout<="1000000";

end case;

end process;

end;

**Clock divider**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity ClockDiv is

port(clk: in STD\_logic;

start\_timer: in STD\_logic;

FastClock , MediumClock , SlowClock , led0 : out STD\_LOGIC );

end Clockdiv ;

architecture beh of Clockdiv is

signal slowClock\_sig : STD\_LOGIC;

begin

process

variable cnt : STD\_LOGIC\_VECTOR (26 downto 0):= "000000000000000000000000000";

begin

wait until (( clk 'EVENT) AND ( clk = '1'));

if ( start\_timer = '0') then

cnt := "000000000000000000000000000" ;

else

cnt := STD\_LOGIC\_VECTOR( unsigned ( cnt ) + 1);

end if ;

FastClock <= cnt (22);

MediumClock <= cnt (24);

SlowClock <= cnt (26);

slowClock\_sig <= cnt (26);

if ( slowClock\_sig = '1') then

led0 <= '1';

else

led0 <= '0';

end if ;

end process;

end beh;

**UCF**

# -------------------------------------------------------------------------- #

#

# Copyright (C) 2020  Intel Corporation. All rights reserved.

# Your use of Intel Corporation's design tools, logic functions

# and other software and tools, and any partner logic

# functions, and any output files from any of the foregoing

# (including device programming or simulation files), and any

# associated documentation or information are expressly subject

# to the terms and conditions of the Intel Program License

# Subscription Agreement, the Intel Quartus Prime License Agreement,

# the Intel FPGA IP License Agreement, or other applicable license

# agreement, including, without limitation, that your use is for

# the sole purpose of programming logic devices manufactured by

# Intel and sold by Intel or its authorized distributors.  Please

# refer to the applicable agreement for further details, at

# <https://fpgasoftware.intel.com/eula>.

#

# -------------------------------------------------------------------------- #

#

# Quartus Prime

# Version 20.1.0 Build 711 06/05/2020 SJ Lite Edition

# Date created = 15:05:50  October 30, 2020

#

# -------------------------------------------------------------------------- #

#

# Notes:

#

# 1) The default values for assignments are stored in the file:

# majordemo\_assignment\_defaults.qdf

#    If this file doesn't exist, see file:

# assignment\_defaults.qdf

#

# 2) Altera recommends that you do not modify this file. This

#    file is updated automatically by the Quartus Prime software

#    and any changes you make may be lost or overwritten.

#

# -------------------------------------------------------------------------- #

set\_global\_assignment -name FAMILY "MAX 10"

set\_global\_assignment -name DEVICE 10M50DAF484C7G

set\_global\_assignment -name TOP\_LEVEL\_ENTITY toplevel

set\_global\_assignment -name ORIGINAL\_QUARTUS\_VERSION 20.1.0

set\_global\_assignment -name PROJECT\_CREATION\_TIME\_DATE "15:05:50  OCTOBER 30, 2020"

set\_global\_assignment -name LAST\_QUARTUS\_VERSION "20.1.0 Lite Edition"

set\_global\_assignment -name PROJECT\_OUTPUT\_DIRECTORY output\_files

set\_global\_assignment -name MIN\_CORE\_JUNCTION\_TEMP 0

set\_global\_assignment -name MAX\_CORE\_JUNCTION\_TEMP 85

set\_global\_assignment -name DEVICE\_FILTER\_PIN\_COUNT 484

set\_global\_assignment -name DEVICE\_FILTER\_SPEED\_GRADE 7

set\_global\_assignment -name ERROR\_CHECK\_FREQUENCY\_DIVISOR 256

set\_global\_assignment -name POWER\_PRESET\_COOLING\_SOLUTION "23 MM HEAT SINK WITH 200 LFPM AIRFLOW"

set\_global\_assignment -name POWER\_BOARD\_THERMAL\_MODEL "NONE (CONSERVATIVE)"

set\_global\_assignment -name VHDL\_FILE ../project1/toplevelproject.vhd

set\_global\_assignment -name VHDL\_FILE ../project1/sevensegcoder.vhd

set\_global\_assignment -name VHDL\_FILE ../project1/ledcoder.vhd

set\_global\_assignment -name VHDL\_FILE ../project1/keypad.vhd

set\_global\_assignment -name VHDL\_FILE centralfsm.vhd

set\_global\_assignment -name VHDL\_FILE clockdiv.vhd

set\_global\_assignment -name PARTITION\_NETLIST\_TYPE SOURCE -section\_id Top

set\_global\_assignment -name PARTITION\_FITTER\_PRESERVATION\_LEVEL PLACEMENT\_AND\_ROUTING -section\_id Top

set\_global\_assignment -name PARTITION\_COLOR 16764057 -section\_id Top

set\_location\_assignment PIN\_A8 -to outputleds[0]

set\_location\_assignment PIN\_A9 -to outputleds[1]

set\_location\_assignment PIN\_A10 -to outputleds[2]

set\_location\_assignment PIN\_B10 -to outputleds[3]

set\_location\_assignment PIN\_D13 -to outputleds[4]

set\_location\_assignment PIN\_P11 -to clk1

set\_location\_assignment PIN\_C10 -to buttonentered[0]

set\_location\_assignment PIN\_C11 -to buttonentered[1]

set\_location\_assignment PIN\_D12 -to buttonentered[2]

set\_location\_assignment PIN\_F15 -to starttime

set\_location\_assignment PIN\_C12 -to res

set\_location\_assignment PIN\_C14 -to segoutput[0]

set\_location\_assignment PIN\_E15 -to segoutput[1]

set\_location\_assignment PIN\_C15 -to segoutput[2]

set\_location\_assignment PIN\_C16 -to segoutput[3]

set\_location\_assignment PIN\_E16 -to segoutput[4]

set\_location\_assignment PIN\_D17 -to segoutput[5]

set\_location\_assignment PIN\_C17 -to segoutput[6]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to buttonentered[1]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to buttonentered[2]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to buttonentered[0]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to buttonentered

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to clk1

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to outputleds[4]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to outputleds[3]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to outputleds[2]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to outputleds[1]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to outputleds[0]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to outputleds

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to res

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to segoutput[6]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to segoutput[5]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to segoutput[4]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to segoutput[3]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to segoutput[2]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to segoutput[1]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to segoutput[0]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to segoutput

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to starttime

set\_global\_assignment -name VHDL\_FILE doortimer.vhd

set\_location\_assignment PIN\_B11 -to Doortimerout[0]

set\_location\_assignment PIN\_A11 -to Doortimerout[1]

set\_location\_assignment PIN\_D14 -to Doortimerout[2]

set\_location\_assignment PIN\_E14 -to dooropenclose

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to dooropenclose

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to Doortimerout[2]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to Doortimerout[1]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to Doortimerout[0]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to Doortimerout

set\_location\_assignment PIN\_B14 -to doorstart

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to doorstart

set\_instance\_assignment -name PARTITION\_HIERARCHY root\_partition -to | -section\_id Top

**References**

1. The Deo Lab manual within blackboard
2. Class notes from ECEN 424 on fsm, timers, and top down design

**Conclusion**

The project involving the elevator was a huge success. The pitfalls and achieved helped implement growth as a student and a FPGA programmer. The elevator helps to teach the important of top down design, finite state machines, and timers in VHDL.