UTKU KALKANLI

SECTİON 02

PROJECT REPORT

PHYSİCAL MODULES:

4 x 4 KEYPAD: In my Project, ı used 4 x 4 keypad matrix to increment or decrement passenger number on each floor. I used the same logic on the keypad7\_SEG sample Project to take passenger inputs. Such as checking whether the key value is valid, determining which key is pressed and then processing the key value in terms of changing the passenger number. I used wires to connect the keypad to my Basys FPGA, I changed my constraint file in order to make it compatible with the wire connection.

8 X 8 DİSPLAY MATRİX: In my Project, i used 8 X 8 display to show passenger number on each floor and the location of the elevator in a particular time. I used the same logic on the RGB\_8X8Display sample Project such as declaring a 27 bit number, incrementing the number inside an always and using an if condition to make an elevator animation.

BASYS 3: I connected my BASYS3 FPGA to the BETİ Board. I connected the keypad to my FPGA by using wires that connects to the JA part of the FPGA. I used the 3 buttons on my FPGA. U18 to resetTimer, T18 to executeScenario, U17 to resetSystem. I used the 7 segment part of my FPGA to show my timer.

MODULES:

“EvacuationElevator.sv”

My Project is basically structured by a top module “EvacuationElevator.sv”. This module connects the already given modules keypad4X4, display8X8. And Seven\_Segment\_Display\_Controller to use timer on 7 segment display of Basys 3. Then i initialize image\_red, image\_blue and image\_green. I also use integer values to keep track of passenger numbers on the floor “f0pas”, “f1pas”, “f2pas”, “f3pas”. Also for keeping the track of elevator by “elevatorPass”(passenger number on elevator) and “elevatorFloor” to understand which floor is my elevator on.

At line 150, i define my states “A”, “F0”,”F1”,”F2”,”F3”,”resetState”. I used typedef enum logic to mention state names with string instead of bytes.

At line 153, i have an always @(posedge clk) statement, an always block which assigns the determined nextState to the current state of the execution. If systemReset button is activated, resetState becomes currentState.

At line 161, i have an another always block which creates the biggest part of my code.This always block starts with state changes. It has a case statement which contains all of my states. I also count a counter, i have if statements at the beginning of my states, when the counter is at a particular value it passes the if statement. By that way i divide the clock and create an animation. In the states i change the integer values that i defined earlier, (f0pas,f1pas,f2pas,elevatorFloor etc..).

At line 411, i have the part that takes user input from keypad. I check whether the input is valid, and then check the key value in a case statement. It increments of decrements the passenger number until execution of the scenario begins.

At line 446, elevator display part of the code begins. This part consist of if statements which contains all possible elevator display conditions. Such as “Floor 3, 3 passenger”, or “Floor 1, 4 passenger”, “Floor 2, 0 passenger”. When the integer values of elevatorFloor, and elevatorPass change on the states, this code statement turns on and off the related leds on the 8x8 display.

At line 555, passenger display part of the code begins. This part consist of case statement for the passengers numbers on each floor. I divided the 8x8 display as [7:6] to elevator, [5:0] to the passenger display. So i take a 6 bit number for each of the lines that shows passenger part, 2 6bit number for each floor . If passenger needs to be shown, that part of the 6bit number becomes 1, otherwise 0.

Module “Seven\_segment\_LED\_Display\_Controller”

This module takes clk signal and resetTimer signal as input and shows a timer on the 7seg. If the resetTimer button pressed, time becomes 0. I used the source www.fgpa4students.com to adapt it to my project.

Other modules are given modules by teaching assistants, i also re-designed inside of the SevSeg\_4Digit as follows, but i wasn’t able to implement it.

4'd0 : sseg\_LEDs = 7'b1000000; //to display 0

4'd1 : sseg\_LEDs = 7'b1111001; //to display 1

4'd2 : sseg\_LEDs = 7'b0100100; //to display 2

4'd3 : sseg\_LEDs = 7'b0110000; //to display 3

4'd4 : sseg\_LEDs = 7'b0011001; //to display 4

4'd5 : sseg\_LEDs = 7'b0010010; //to display 5

4'd6 : sseg\_LEDs = 7'b0000010; //to display 6

4'd7 : sseg\_LEDs = 7'b1111000; //to display 7

4'd8 : sseg\_LEDs = 7'b0000000; //to display 8

4'd9 : sseg\_LEDs = 7'b0010000; //to display 9

4'd10 : sseg\_LEDs = 7'b1111110; //to display -

4'd11 : sseg\_LEDs = 7'b1111101; //to display -

4'd12 : sseg\_LEDs = 7'b1111011; //to display -

4'd13 : sseg\_LEDs = 7'b1110111; //to display -

4'd14 : sseg\_LEDs = 7'b1101111; //to display -

4'd15 : sseg\_LEDs = 7'b1011111; //to display -

States:

A: Adding state, basically takes key value and increments or decrements the passenger number on each floor. When addition starts, it recursively stays in the A state until “startScenario” button pressed. If “startScenario” executed, nextState becomes F0.

F0: This state represents Floor 0. If any passenger exists on Floor 1, Floor 2, or Floor 3, nextState becomes F1. Otherwise, if elevatorPass > 0 , which means elevator comes from F1, the elevatorPass becomes 0. Which means emptying the passengers.

F1: This state represents Floor 1. If elevatorPass == 4, which means elevator came from upstairs full, nextState becomes F0. Otherwise, if f1pas >= 4, it decrements f1pas by 4, increments elevatorPass by 4, and send the elevator to F0. If f1pas < 4 and f2pas or f3pas > 4, the elevator goes second floor and nextState becomes F2. Otherwise, if f1pas is between 0 and 4 , it checks does Floor 2 and Floor 3 has passenger, if they have the elevator goes to F2, if not F1 gives its passengers to elevator and elevator goes down.

F2: This state represents Floor 2. If elevator is full, nextState becomes F1. If not and f2pas is bigger than 4, it fills the elevator with passengers and sends them to Floor 1, which is represented by F1. If f2pas is between 0 and 4, it checks does Floor 3 has passenger, if it does nextState becomes F3. If Floor 3 is empty, it fills the elevator with passengers if elevator has enough space, and nextState becomes F1. If f2pas is 0, it checks whether f3pas > 0, if yes nextState becomes F3, if not, nextState becomes F1. If elevatorPass == 4, which means elevator comes from F3, nextState becomes F1 without any change.

F3: This state represents Floor 3. If this state is reached, it means f3pas > 0 . If f3pas >= 4, it means elevator is empty because algorithm starts taking passengers 4 by 4 at first. F3pas decreases by 4, elevatorPass increase and nextState becomes F2. Which means elevator goes down. If f3pas is between 0 and 4, again it means elevatorPass is 0, because general algorithm takes passengers at every floor smaller than 4 by starting from highest floor with passenger exists, after taking all passengers on each floor 4 by 4. F3pas becomes 0, elevatorPass increase and nextState becomes F2.

ResetState: This state is executed is systemReset button is 1.

ResetState makes f0pas,f1pas,f2pas,f3pas,elevatorFloor,elevatorPass, all of them 0.

Final Note:

I used [www.fpga4students.com](http://www.fpga4students.com) as a source to implement my 7segment display timer. By using “Seven\_segment\_LED\_Display\_Controller” module.