Iowa State University

Final Project Report

Elevator

Henry Lehman and Miles Nichols

CPRE 281

Doctor Mohamed Selim

5/1/2024

We completed our own project idea, a functioning four floor elevator. This report details all of our strategies and attempts to build a working circuit in QuartusPrime and have it function on the TI ALTERA boards in the lab. In order to implement this working elevator, we used both a combination of Block Diagram Files, and Verilog logic files.

TOP LEVEL DIAGRAM

A diagram of a computer

Description automatically generated with medium confidence

This picture of a BDF shows top-level diagram. It maybe hard to see, but there are two main parts that allow the elevator to function.

First, the left side of the circuit is a register file that the user input is loaded onto and the elevator uses to understand its current position compared to where the user wants to go. This register file also is connected to 2 seven segment displays. These displays show the current value of the 2 2-bit registers nested inside the register file.

A computer screen shot of a computer

Description automatically generated

Inputs:

* + Data0 – the least significant bit of the floor the user wants to go to
  + Data1 – the most significant bit of the floor the user wants to go to
  + Clock – The clock the runs the clocks of the DFF’s inside the registers
  + Clear – Only Used for testing, not a necessary feature for the end user
  + WriteEnable – This is turned on when a user wants to enter their next floor
  + WriteAddress – Chooses register 1 or 0 when the user enters their next floor
  + ReadAddress – determines whether the the user needs to go up or down

Outputs:

* + R1A-G – Seven Segment display outputs for register 1
  + R0A-G – Seven Segment display outputs for register 0
  + Output1 – the most significant bit of the floor the user wants to go to
  + Ouput0 – the lest significant bit of the floor the user wants to go

Now if we want to go into the block symbol file of the register file to see the different registers

A diagram of a computer

Description automatically generated

In this block diagram that represents the register file of the elevator system. We can see 4 main parts. First is the 4 central single bit registers. They are arranged to act as 2 2-bit registers that can each store a floor 0-3. Then by itself on the left of the file we can see the 1 to 2 decoder that is used to control the address in the register that we are writing to in conjunction with an enable. Next we see 2 seven segment displays connected to the out put of each 2 bit register. These are constantly displaying the floor that each register is storing. Then at the bottom of the file there are 2 2 to 1 MUXes. These control the reading of the register file, based on the read input that is connected to their selector switch.

Now if we want to move to the other side of the elevator file.

A diagram of a computer

Description automatically generated with medium confidence

This is the actually functionality of the elevator file. On this side we are able to see 3 main parts. With some additional smaller features. First, on the very lefthand side of the file we can see a 2 2bit number comparator. A diagram of a computer

Description automatically generated

This comparator has 4 inputs and 3 outputs.

Inputs: Num1\_bit0 – the least significant bit of the current floor being displayed.

* + - Num1\_bit1- the most significant bit of the current floor being displayed.
    - Num2\_bit0 – the least significant bit of the floor being read from the register file
    - Num2\_bit1- the most significant bit of the floor being read from the register file

Outputs:

* eq – This output is 1 when the current floor being displayed is the same as the floor being read from the register file, all other times it is 0
* Gt – This output is 1 when the current floor being displayed is the greater than the floor being read from the registerfile, all other times is 0.
* Lt – this output is 1 when the current floor being displayed is less than the floor being read from the register file, all other times is 0.

Originally, we made this comparator using a large truth table(displayed below), however it was easier to implement using Verilog with the reduced expressions generated from the truth table, than to write the whole truth table in Verilog.

A piece of paper with writing on it

Description automatically generated A screenshot of a computer

Description automatically generated

Basically, the function of the comparator is to output the relationship of the next floor we are trying to travel to in relation to the floor we are currently on.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A diagram of a computer

Description automatically generated

The next part of this side of the elevator is the finite state machine. The strategy that we used to implement a FSM is to make a special counter.

This FSM has 4 states and 2 inputs and 2 outputs.

A piece of paper with writing on it

Description automatically generated

This FSM has 3 inputs and 2 outputs.

Inputs:

* W – this input is whether the counter counts up or counts down. (1 for up, 0 for down)
* E(enable)- this input controls whether the FSM can move state (1 for moving, 0 for lock at current state)
* There is also a clock input in the bdf but this is not part of the FSM

Outputs:

* Out1(Y1) – this is the most significant bit of the value of the floor we are currently on
* Out0(Y0) – this is the least significant bit of the value of the floor we are currently on

States:

* This FSM has 4 states, one to represent each floor of the building.
* This is a Moore type machine because the output is only dependent on the current state. Ie floor we are on.

For us, the most logical type of counter is a modulo 4 type counter, but it would need a some special features. First and most importantly it needed to be able to count both up and down, because our elevator needed to be able to go up and down. Additionally, we want the elevator to be able to stop at a floor, once we arrived at the floor the user originally entered. To do this we included an Enable which only lets the elevator move when it isn’t at the desired floor.

To actually implement this Counter, we made a state diagram, and from the there followed the steps to make a FSM, State diagram🡪State Assignment Table🡪State Variable Table🡪K-maps🡪Next State Expressions🡪BDF of the circuit.

A computer screen shot of a circuit board

Description automatically generated

A diagram of a computer

Description automatically generated

This is a seven segment display that displays the current location of the elevator, and is updated with each clock cycle.

Finally, there are some important things to note about the connection between the FSM and the comparator.

A diagram of a computer

Description automatically generated

The lt (less than output) pin of the comparator is connected to the input(w) of the counter because in the FSM, 1 means go up and 0 means go down, so we want to move up, when the current floor is less than the next floor, and we want to move down when the current floor is greater than the next floor. Additionally, we have the enable to connected to the eq(equal to) output pin **BAR** of the comparator because we only want to move when the current floor and next floor are not equal to one another. Additionally, there are three output pins in these highlighted sections. All three are output to LED’s on the board, and each one represents a different signal you would normally see in an elevator. The top output pin(E22) is the gt output lights up to indicate the elevator is moving down. The pin directly below this(E21) is the opposite and indicates that the elevator is moving up. The final pin that is connected to the eq output of the comparator, is on when the elevator is stops, and indicates an open door, and that a new user can enter the elevator and choose a new floor.

Seeing it work on the board.

A close up of a circuit board

Description automatically generatedA close up of a circuit board

Description automatically generated

Write Enable

Light on if Door open

Light on if moving up open

Clock

Light on if moving down open

Left Switch – MSB of input

Right Switch – LSB of input

Left Switch - RA

Right Switch– WA

7SD for current Floor

Left 7SD – Register 1

Right 7SD – Reister 0

Now we can see that the MSB and LSB of the input are set to 1.

Write Enable is set to 1.

Write Address is set to 1.

And after one clock cycle, register 1 is set to 1.

Now the door is closed.

The elevator is moving up.

A close up of a circuit board

Description automatically generated

Now we can see that the MSB and LSB of the input are set to 0.

Write Enable is set to 0.

The door is closed.

The elevator is moving up.

And after one more clock cycle the elevator has moved up one floor.

A close up of a circuit board

Description automatically generated

After one more clock cycle, the elevator is still moving up and is currently on the second floor.

A close up of a circuit board

Description automatically generated

After one more clock cycle, the elevator is now on the third floor.

The door light is open.

The elevator is no longer moving up.

A close up of a circuit board

Description automatically generated

Now if someone enters the elevator and say selects floor one.

The door closes and the down light is on.

And the elevator will continue the process indefinitely.