John A. Gangemi
CDA 3201L Thursday Section
Lab Final Project - 3 Story Elevator Controller
07/19/2014

PURPOSE AND OBJECTIVES

The final project for the Logic Design Lab is to create a 3 story elevator controller based on a finite state machine, either Moore or Mealy. The guidelines for the final project specify the requirements of five inputs and three outputs.

Inputs to the finite state machine are given by Reset, Start, Up, Down, and setAlarm. Functionally the Reset input takes the elevator to its idle state, the Start input enables control of the elevator, the Up input allows the elevator to move up a single floor and likewise the Down input allows the elevator to move down a single floor, and finally the setAlarm input triggers a sequence of events that close the elevator doors and flash an alarm light.

Outputs to the finite state machine are given by Door, Alarm, and Floor. Functionally the Door output displays the current status of the elevator door, the Alarm output toggles the luminosity of a light, and the Floor output shows the current floor the elevator is on.

Operation of the elevator is described as starting in an idle state where the elevator is on the first floor, doors open and alarm light is disabled. Once the Start input is enabled the elevator's Up and Down inputs become operational. Selecting either the Up or Down inputs will close the elevator door, move the elevator up or down a single floor if possible, and when the elevator reaches the desired floor the doors must remain open until instructed otherwise. If the Reset input were to become active at any point in time during the execution of the elevator's program then the elevator will asynchronously switch to the idle state. Control of the elevator must function accordingly to its real-world counterpart and thus it cannot wrap around from the third floor to first floor and vice-versa.

COMPONENTS

- Integrated Circuits
    - 74LS04 Hex Inverter
    - 74LS08 Quad 2 Input AND
    - 74LS11 Triple 3 Input AND
    - 74LS32 Quad 2 Input OR
    - 74LS74 Dual D Flip-Flop with Preset and Clear
    - TLC555 LinCMOS Timer (astable operation @ 1Hz)

        - 1K Ohm Resistor
        - 4.7K Ohm resistor
        - 10K Ohm Resistor
        - 47 microFarad Electrolytic Capacitor
        - 0.01 microFarad Metal Film Capacitor
- 4 x 5mm Red LED
- 4 x 330 Ohm Resistors
- 5 Volt Regulated Power Supply
- Assortment of 22 AWG Wire
- 2250 Point Breadboard
- LogiSim version 2.7.1
- Icarus Verilog 0.9.7

DESIGN DESCRIPTION

The design and implementation of the 3 story elevator began with a logical reasoning of the requirements, specifically the movement of the elevator from floor-to-floor and the decision to use the Moore or Mealy finite state machine structures.

Ultimately the use of a Mealy model for the finite state machine was chosen based on the fact that the output for the FSM (finite state machine) is a function of the inputs (Reset and setAlarm specifically) and the current state.

Planning the sequence of movements for the elevator from floor-to-floor required a state diagram, state definition table and state transition table. In theory the elevator's operation is as follows:

- The elevator will close its doors on the current floor
- With the doors closed the elevator will move to the next floor
- Once on the desired floor the elevator will open its doors

The most challenging part of the design of the elevator's sequence of movements was deciding on how to best represent the states surrounding the second floor. The second floor is a special case in that there are four possible states, the first state is when the elevator is on the first floor and has begun to move up to second floor, the second state is when the elevator is on the second floor and begins to move up to the third floor, the third state is when the elevator is on the third floor and begins to move down to the second floor, and lastly the elevator is on the second floor and begins to move down to the first floor. It's apparent that when the elevator door is closed and the elevator is on the second floor it's necessary to have multiple "second floor, door closed" states to accommodate the state transitions of the state diagram. A judicious decision to use three bits to encode the current

floor of the elevator provides the appropriate amount of bit patterns. In this case the least significant bits Q2 and Q1 provide the 2-bit elevator floor output.

State Definitions:

State Definitions

Door Z	Q3	Floor Q2	Q1	State Identifier	State Description
0	0	0	0	S2	Door Closed, 2nd Floor to 2nd Floor
0	0	0	1	S1	Door Closed, 1st Floor to 2nd Floor
0	0	1	0	S4	Door Closed, 2nd Floor to 3rd Floor
0	0	1	1	S5	Door Closed, 3rd Floor to 3rd Floor
0	1	0	0	x	
0	1	0	1	S9	Door Closed, 1st Floor to 1st Floor
0	1	1	0	S8	Door Closed, 2nd Floor to 1st Floor
0	1	1	1	S7	Door Closed, 3rd Floor to 2nd Floor
1	0	0	0	x	
1	0	0	1	S0	Door Open, 1st Floor
1	0	1	0	S3	Door Open, 2nd Floor
1	0	1	1	S6	Door Open, 3rd Floor
1	1	0	0	x	
1	1	0	1	x	
1	1	1	0	x	
1	1	1	1	x	

Let it be noted that there is an unavoidable and incorrect bit representation of the least significant bits Q2 and Q1 for state S2 (second floor should be “10”). The output for bit Q2 can be corrected with combinational logic to select the appropriate bit value in the case of bits Q2 and Q1 being “00”. The circuit will display “00” as “10” after such correction is made.

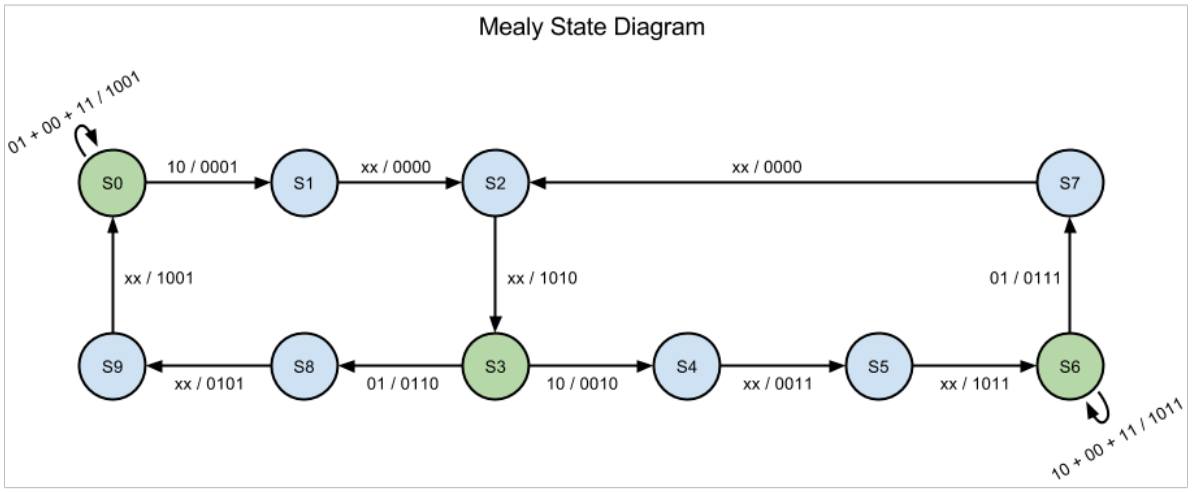
Q2	Q1	Adjusted Q2
0	0	1
0	1	0
1	0	1
1	1	1

	Q2'	Q2
Q1'	1	1
Q1	0	1

Q2 + Q1'

From the state definitions table the circuit will require four flip-flops, one for the state of the elevator door and three for the state of the floor the elevator currently resides on. For this circuit D Flip-Flops are being used to store the current state of the finite state machine.

State Diagram:



State Transitions:

State Transitions

Present State				Inputs		Next State			
Z	Q3	Q2	Q1	Up	Down	Z+	Q3+	Q2+	Q1+
1	0	0	1	1	0	0	0	0	1
0	0	0	1	x	x	0	0	0	0
0	0	0	0	x	x	1	0	1	0
1	0	1	0	1	0	0	0	1	0
0	0	1	0	x	x	0	0	1	1
0	0	1	1	x	x	1	0	1	1
1	0	1	1	0	1	0	1	1	1
0	1	1	1	x	x	0	0	0	0
0	0	0	0	x	x	1	0	1	0
1	0	1	0	0	1	0	1	1	0
0	1	1	0	x	x	0	1	0	1
0	1	0	1	x	x	1	0	0	1
1	0	0	1	0	1	1	0	0	1
1	0	1	1	1	0	1	0	1	1

Since there are now six variables a complete state table will consist of 64 rows and lead to tedious simplification techniques using Karnaugh maps or Quine McCluskey method by hand, thus it was more time conscience to use a computer aided logic simplification tool such as “Logic Friday”. This program requires the user complete a truth table of the logic and enter the relevant outputs, using the Espresso method to generate simplified Boolean expressions. The Boolean expressions that were generated from Logic Friday represent the combinational logic to each of the four D flip-flops in the circuit.

State Table:

State Table

Present State				Inputs		Next State				Flip-flops			
Z	Q3	Q2	Q1	Up	Down	Z+	Q3	Q2	Q1	Dz	Dq3	Dq2	Dq1
0	0	0	0	x	x	1	0	1	0	1	0	1	0
0	0	0	0	x	x	1	0	1	0	1	0	1	0
0	0	0	0	x	x	1	0	1	0	1	0	1	0
0	0	0	0	x	x	1	0	1	0	1	0	1	0
0	0	0	1	x	x	0	0	0	0	0	0	0	0
0	0	0	1	x	x	0	0	0	0	0	0	0	0
0	0	0	1	x	x	0	0	0	0	0	0	0	0
0	0	0	1	x	x	0	0	0	0	0	0	0	0
0	0	1	0	x	x	0	0	1	1	0	0	1	1
0	0	1	0	x	x	0	0	1	1	0	0	1	1
0	0	1	0	x	x	0	0	1	1	0	0	1	1
0	0	1	0	x	x	0	0	1	1	0	0	1	1
0	0	1	1	x	x	1	0	1	1	1	0	1	1
0	0	1	1	x	x	1	0	1	1	1	0	1	1
0	0	1	1	x	x	1	0	1	1	1	0	1	1
0	0	1	1	x	x	1	0	1	1	1	0	1	1
0	1	0	0	0	0	x	x	x	x	x	x	x	x
0	1	0	0	0	1	x	x	x	x	x	x	x	x
0	1	0	0	1	0	x	x	x	x	x	x	x	x
0	1	0	0	1	1	x	x	x	x	x	x	x	x
0	1	0	1	x	x	1	0	0	1	1	0	0	1
0	1	0	1	x	x	1	0	0	1	1	0	0	1
0	1	0	1	x	x	1	0	0	1	1	0	0	1
0	1	0	1	x	x	1	0	0	1	1	0	0	1
0	1	1	0	x	x	0	1	0	1	0	1	0	1
0	1	1	0	x	x	0	1	0	1	0	1	0	1
0	1	1	0	x	x	0	1	0	1	0	1	0	1
0	1	1	0	x	x	0	1	0	1	0	1	0	1
0	1	1	1	x	x	0	0	0	0	0	0	0	0
0	1	1	1	x	x	0	0	0	0	0	0	0	0
0	1	1	1	x	x	0	0	0	0	0	0	0	0
0	1	1	1	x	x	0	0	0	0	0	0	0	0

1	0	0	0	0	0	x	x	x	x	x	x	x	x		
1	0	0	0	0	1	x	x	x	x	x	x	x	x		
1	0	0	0	1	0	x	x	x	x	x	x	x	x		
1	0	0	0	1	1	x	x	x	x	x	x	x	x		
1	0	0	1	0	0	1	0	0	1	1	0	0	1		
1	0	0	1	0	1	1	0	0	1	1	0	0	1		
1	0	0	1	1	0	0	0	0	1	0	0	0	1		
1	0	0	1	1	1	1	0	0	1	1	0	0	1		
1	0	1	0	0	0	1	0	1	0	1	0	1	0		
1	0	1	0	0	1	0	1	1	0	0	1	1	0		
1	0	1	0	1	0	0	0	1	0	0	0	1	0		
1	0	1	0	1	1	1	0	1	0	1	0	1	0		
1	0	1	1	0	0	1	0	1	1	1	0	1	1		
1	0	1	1	0	1	0	1	1	1	0	1	1	1		
1	0	1	1	1	0	1	0	1	1	1	0	1	1		
1	0	1	1	1	1	1	0	1	1	1	0	1	1		
1	1	0	0	0	0	x	x	x	x	x	x	x	x		
1	1	0	0	0	1	x	x	x	x	x	x	x	x		
1	1	0	0	1	0	x	x	x	x	x	x	x	x		
1	1	0	0	1	1	x	x	x	x	x	x	x	x		
1	1	0	1	0	0	x	x	x	x	x	x	x	x		
1	1	0	1	0	1	x	x	x	x	x	x	x	x		
1	1	0	1	1	0	x	x	x	x	x	x	x	x		
1	1	0	1	1	1	x	x	x	x	x	x	x	x		
1	1	1	0	0	0	x	x	x	x	x	x	x	x		
1	1	1	0	0	1	x	x	x	x	x	x	x	x		
1	1	1	0	1	0	x	x	x	x	x	x	x	x		
1	1	1	0	1	1	x	x	x	x	x	x	x	x		
1	1	1	1	0	0	x	x	x	x	x	x	x	x		
1	1	1	1	0	1	x	x	x	x	x	x	x	x		
1	1	1	1	1	0	x	x	x	x	x	x	x	x		
1	1	1	1	1	1	x	x	x	x	x	x	x	x		

Next State Expressions

Dz = Q3 Q2' + Q2' Q1' + Z Up Dwn + Z Up' Dwn' + Z' Q3' Q2 Q1 + Z Q2' Dwn + Q3' Q2 Q1 Dwn'
Dq3 = Q3 Q1' + Z Q2 Up' Dwn
Dq2 = Q3' Q2 + Q2' Q1'
Dq1 = Q3 Q2' + Z Q1 + Q3' Q2 Q1 + Z' Q2 Q1'

Let R = Q3 Q2' & S = Q3' Q2 Q1 & T = Q2' Q1'

Dz = R + T + Z Up Dwn + Z Up' Dwn' + Z' S + Z Q2' Dwn + S Dwn'
Dq3 = Q3 Q1' + Z (Q2 Up' Dwn)
Dq2 = Q3' Q2 + T
Dq1 = R + Z Q1 + S + Z' Q2 Q1'

By the method of extraction the logic was further simplified to facilitate in the implementation of the circuit on the breadboard.

To design the Reset function it was fairly simple as it is an asynchronous operation and can be implemented using the asynchronous inputs of the D Flip-Flop, namely preset and clear of which both are active low thus the Reset input will now become an active low input as well to reduce the need for extra inverters to keep it active high. The asynchronous inputs of the D Flip-Flops are configured as follows:

- Dz has preset wired to Reset
- Dq3 has clear wired to Reset
- Dq2 has clear wired to Reset
- Dq1 has preset wired to Reset

The design of the alarm relies on both asynchronous and synchronous operation. In a similar manner to the Reset input the setAlarm input was made to be active low. When setAlarm is triggered the asynchronous clear input of the Dz (Door) flip-flop becomes enabled and thus the elevator door closes. Furthermore the active low setting for setAlarm disables the clock signal to the four flip-flops as this ensures that the elevator will remain on its current floor. The clock is then used to toggle the light emission of an LED representative of the alarm light.

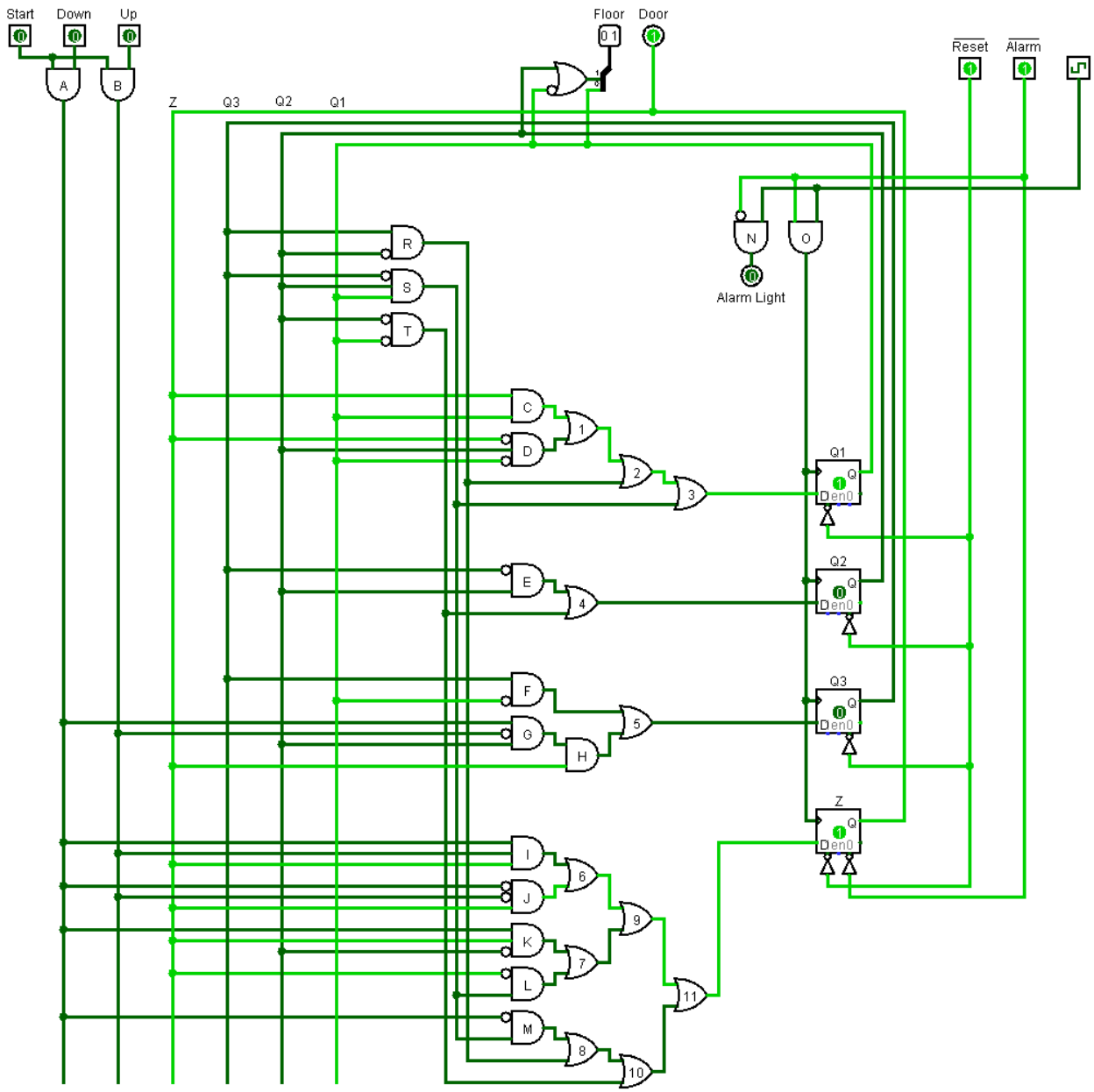
setAlarm	Clock	Flip-Flop Clock	Alarm Light
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

Flip-Flop Clock = setAlarm * Clock

Alarm Light = setAlarm' * Clock

Implementing the Start input is rather simplistic and does not justify the need for a truth table, essentially the Start input is “anded” separately with the Up and Down inputs (Start * Up and Start * Down). When Start is active high and either Up or Down are active high their respective signals are sent to the combinational logic network that drives the inputs to the D Flip-Flops.

Using LogiSim to create a logic circuit diagram results in the following:



OBSERVATIONS AND DATA ANALYSIS

In testing the logic simulation of the circuit using LogiSim it performed remarkably well. A series of test were created to represent possible scenarios that might obstruct the desired behavior of the elevator.

- Does the Reset input set the elevator to its idle state asynchronously?
- Does the Alarm input close the elevator door?
- Does the Alarm input toggle the output of the alarm light?
- Does the Alarm input stop the clock from reaching the flip-flops when asserted low?
- Does the Start input enable functionality of the Up and Down inputs?
- Will the elevator move down if on the first floor?
- Will the elevator move up if on the third floor?
- Will the elevator accept input when Up and Down are both active or both disabled?
- Is Up or Down input ignored when transitioning from floor-to-floor?
- Are any unused states introduced to the circuit?
- Is the decoding of the two least significant bits for the Floor output correctly displayed?

The only oddity found within the simulation of the circuit was when asserting the setAlarm high after previously being asserted low causes the elevator to move to the next floor and/or state on the next positive edge of the clock. For example if on the first floor and the setAlarm had been previously set low then the next clock cycle moves the elevator to the second floor and opens the door. Its counterpart, if on the third floor and the setAlarm had been previously set low then the next clock cycle moves the elevator to the second floor and opens the door.

While this behavior is not planned it is certainly not harmful to the operation of the elevator. The way to reason about it is to think if a person pressed the alarm button on a given floor then would it not be logical to move the elevator to the next floor in its sequence. Perhaps the circuit has gained artificial intelligence in LogiSim through extensive testing, whatever the case may be the elevator suffers from no negative side effects.

DISCUSSION AND CONCLUSION

In designing the 3 story elevator many iterations were created and subsequently disregarded. The first iteration of the project was a simple 2 story elevator with only Up and Down inputs, yes it did not meet the project guidelines instead it was a logical test in many ways. Did the state diagram translate to a working circuit? Can there be shortcuts made in this version that will benefit a larger design? Answering these questions led to bigger designs and eventually the final design that implemented all of the requirements.

Truth-be-told I was dreading this project as sequential logic has always been a more of a “chore” than an exercise and I never put real effort into learning the finer details as I did with combinational logic. However after state simplification techniques in lecture and from outside resources, mainly on the internet, I hold sequential logic in high regard.

In fact I would like to try and implement the project again using actual push buttons (debounced), a seven segment display and perhaps more than three floors. The only downside to a physical implementation on the breadboard is the time required to make the connections between the numerous TTL packages. Its for this reason that I am looking into using PAL/PLA/FPGA chips to do most of the two-level implementations but from what I gather programming these chips is not for the novice user. It might be best if I wait to take the FPGA hardware course.

Overall the Logic Design lab has been a real pleasure for someone who is an aspiring Computer Engineer.