CDA 3201L - Thursday (3:30 - 4:45PM) Section 005

Lab #08 - Lab Final

12/02/2016 - Steven Romeiro, Jennyfer Munoz

Purpose & Objectives:

The objective of this lab was to design and implement the operation of an elevator in a 2-story building.

Components Used:

Name	Type	Quantity
74LS08	AND IC (2-inputs)	1
74LS11	AND IC (3-inputs)	1
74LS04	Inverter IC	1
74LS32	OR IC	2
74LS86	XOR IC	1
74LS74	DFF	2
74LS163	BCD Counter	
74LS247	BCD to 7-Segment Decoder	1
LED	7-Segment LED	1
470 Ω Resistor	Resistor	2
LED	Red LED	2
Power Supply	5v	1
Wire Kit	Assorted	1

Description:

This elevator should operate by moving from floor 0 to floor 1 after a small sequences of steps. The operations should perform as follows:

1. The elevator begins in an idle state with the doors closed awaiting user input.
2. The user toggles the elevator from its idle state by pressing a button that begins the sequence of operations.
3. Once toggled, the elevator first opens the doors for 4 seconds to allow the user in.
4. After 4 seconds have passed the door closes.
5. Once the doors have closed, the elevator begins moving to the next floor. Moving from one floor to another should take 4 seconds.
6. Upon arrival to the new floor the elevator opens the door for 4 seconds to allow the user to exit and returns to an idle state.

The implementation of this design will require the use of a Binary Counter 74LS163 chip. This chip will count from 0 - 15 at a frequency of 4 Hz. The Period for 4 Hz is 0.25s. The counter will count 16 steps, 16 x 0.25 = 4s. The 74LS163 chip will produce an output of 1 every 4s at a frequency of 4Hz

This should allow us to satisfy the timing condition of the the door and moving operation. Each state after the sequence has begun should take 4s to complete until the elevator returns to idle again. Here are the possible states for a full operating cycle of this elevator:

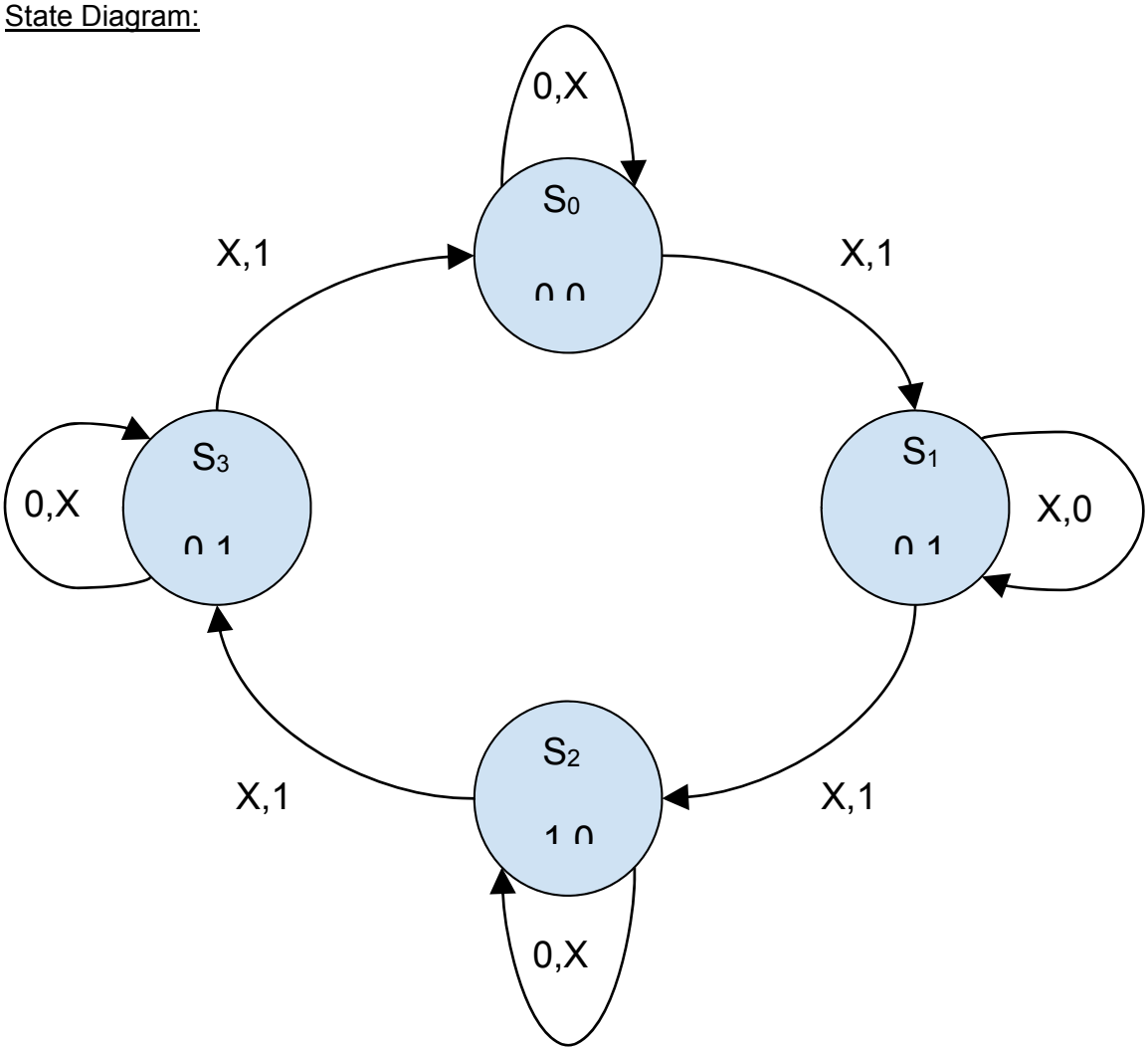
1. Idle state, elevator not moving and door is closed
2. Enable input receive, door opens for 4s before closing again
3. Door is closed and elevator starts moving for 4s
4. Elevator stops moving and door opens at the new destination

The states above will be displayed as S_0, S_1, S_2 and S_3 respectively from here on out. Binary state assignments need to be made for these states. We have a total of 4 states, this means we can accomplish this design with 2 Flip Flops. This is due to the fact that $2^n = m$, where ‘n’ is the number of Flip Flops required for an ‘m’ amount of states. We have chosen D-Flip Flops for the implementation of this design. The new states need to be assigned binary values. These assignments have been shown below:

Assignments:

	A	B
S_0 =	0	0
S_1 =	0	1
S_2 =	1	0
S_3 =	0	1

State Diagram:



Inputs: E,T

E = Enable bit

T = Tc from BInary Counter signaling 4s have passed

Outputs: M,D

M = Moving/Not moving

D = Door open/closed

State Table:

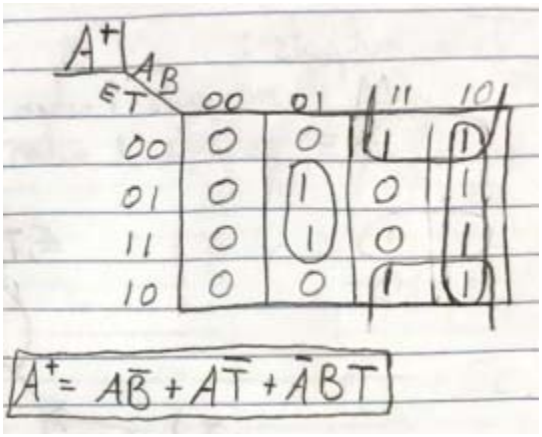
Present State	Next State				Output
	E,T = 00	E,T = 01	E,T = 11	E,T = 10	M,D
S_0	S_0	S_0	S_1	S_1	0 0
S_1	S_1	S_2	S_2	S_1	0 1
S_2	S_2	S_3	S_3	S_2	1 0
S_3	S_3	S_0	S_0	S_3	0 1

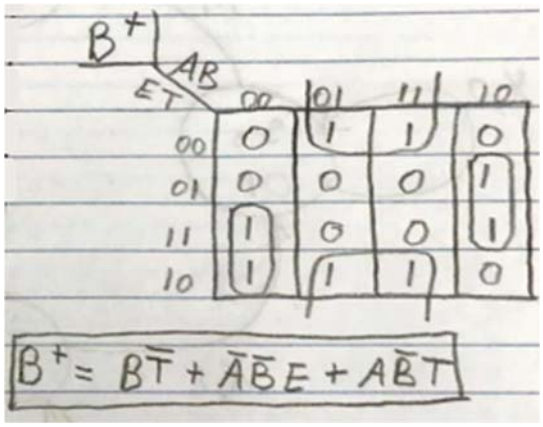
Truth Table:

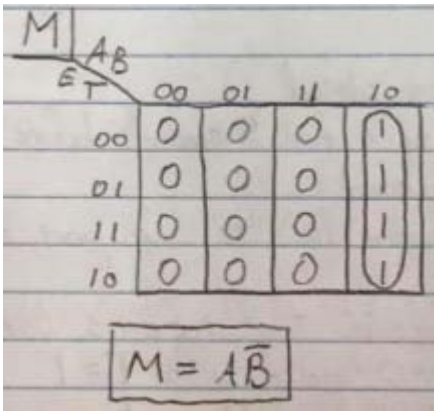
A	B	E	T	A^+	B^+	M	D
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	1	0	0
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	1	0	0	1
0	1	1	0	0	1	0	1
0	1	1	1	1	0	0	1
1	0	0	0	1	0	1	0
1	0	0	1	1	1	1	0
1	0	1	0	1	0	1	0
1	0	1	1	1	1	1	0
1	1	0	0	1	1	0	1

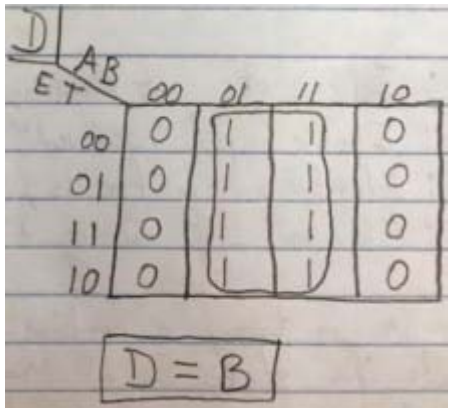
1	1	0	1	0	0	0	1
1	1	1	0	1	1	0	1
1	1	1	1	0	0	0	1

KMAPS:









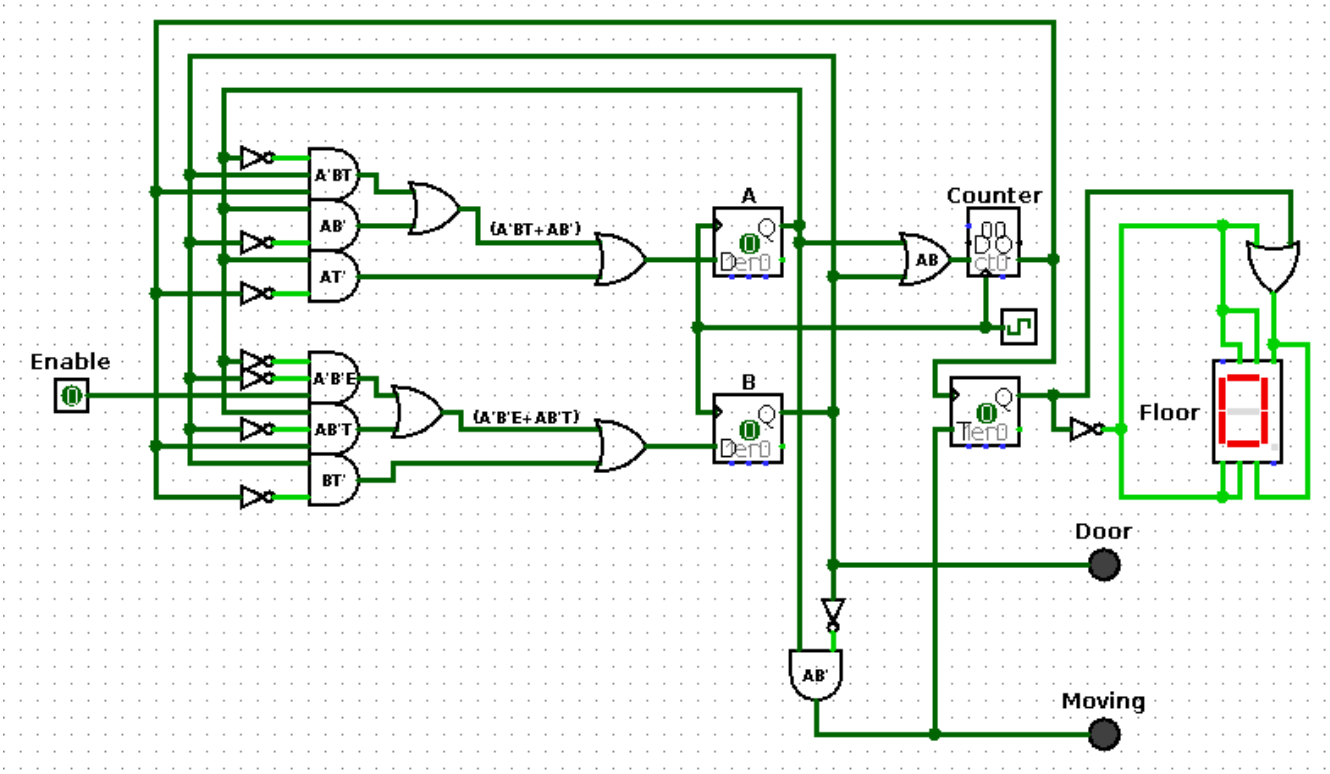
T-FF Truth Table:

A	B	T	TFF
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0

1	0	1	1
1	1	0	0
1	1	1	0

- **TFF = AB'T**

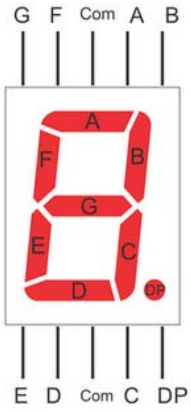
Diagram



Observations:

The T-Flip Flop used in this implementation is toggling the floor that the elevator is currently on. The toggle happens when the elevator is moving and 4s have passed. The flip flop is receiving a rising edge from the TC output of the Binary Counter to allow the toggle to happen. The moving bit can be set, but it will only toggle the next floor once the 4s have passed and TC sends a signal to the clock of the flip flop.

Another observation is that the counter chip should only be enabled if either A or B is high. Flip flop A and B represent the current state of the machine. The only time the counter should start working is when the elevator has begun its sequence. This only happens after the Enable input has been given and state S_1 has been initiated. The counter should continue working until the idle state is reached again where both A and B are low.



For the 7-Segment Display, a constant high is required on the B and C pin. Floor zero and one both require those pins to be on indefinitely. This is accomplished by OR'ing Q with Q' from our T-Flip Flop, the remaining pins can receive a Q' input to allow for the proper floors to be represented.

Discussion & Conclusion:

Design phase of this build proved a greater difficulty than expected. Understanding that the floor doesn't need to be accounted for in the State Diagram and consequently the State Table was certainly the trickiest part. Other designs with more states could account for tracking the floor as one of the outputs but this would increase the the amount of components in the design. To keep the design simpler, we decided to use 4 states that kept track of two outputs, one for the elevator door and one for elevator moving. Once the realization that a combination of the state outputs and present inputs could be used to determine the toggling of the floor number the design phase became very straight forward. State diagrams were developed, then state tables, truth tables and eventually KMAPs which allowed us to arrive at the simulation phase of our design.

Once the simulation phase proved successful it was just a matter of implementing the working design on a breadboard. Implementation provided its own set of difficulties as the chip work differently when implementing them. Several additional requirements such as setting CLR and LOAD pins to constant high or accounting for the fact that certain input pins are automatically inverted on the chip. These small details easily allow for mistakes that are difficult to trace back. Another implementation phase issue we ran into is that T-Flip Flops are not readily available in our kit. This forced us to convert a D-Flip Flop into a TFF by using an XOR chip with the input and the Q output of the flip flop. Another approach is to use a JK-Flip Flop where the same J input is sent to the K pin. Once these design and implementation difficulties were overcome, the project became easy to design.