Lab 8 - Final Lab Project 3-Story Elevator Controller

CDA 3201L-003
Jason Keene and Jacob Manfre
Submitted April 26th, 2015

Purpose and Objectives

We set out to implement a controller circuit for a three story elevator. The circuit has a number of inputs (up, down, reset, start, and setAlarm), shows the current floor on a 7-seg display, has an LED that flashes when the alarm is on, and an LED to show when the door is open or closed.

Component List

- Breadboard
- Wiring
- 5v power supply
- TTL 0.5 Htz Function Generator
- 2 x Hex Inverter (74LS04)
- 4 x Quad 2-input AND Gate (74LS08)
- 2 x Triple 3-input AND Gate (74LS11)
- 1 x Dual 4-input AND Gate (74LS21)
- 5 x Quad 2-input OR Gate (74LS32)
- 2 x Dual D Flip-Flops (74LS74)
- 1 x BCD to 7-seg Decoder (74LS247)
- 7-Seg Display
- 2x LEDs
- 3x Resistors (470 Ohms 5%)

Design

The first step we took was to eliminate as many inputs from our state machine as possible. The start and setAlarm inputs could simply be ANDed with the clock to enable/disable the flip flops. The reset input just asynchronously sets the state of the flip flops so it could be ignored.

That left us with up and down inputs. We created a state diagram based on these inputs (Figure 1).

Once we had the state diagram we created a state table (Figure 2). We were able to eliminate state I as it was the same as state C. We then assigned binary codes to each of our states (Figure 3).

We then created K-Maps to obtain the state and output equations (Figure 4).

Once we had the state and output equations we implemented the circuit in logisim (Figure 5 and 6). Once the circuit was tested we implemented it in hardware.

Discussion and Conclusion

During this lab we applied what we learned about combinational and sequential logic to build a circuit that does something somewhat interesting. Building upon the techniques learned in this lab we can imagine constructing an ALU, interfacing with μ-controllers and networks, creating hardware user interfaces, and displaying information with hardware displays. The economy of complexity in building circuits from combinational and sequential logic vs building solutions in software was apparent throughout this lab. Implementing solutions in hardware is massively less complex but requires a greater investment in time and energy.

Figures

Figure 1 - State Diagram

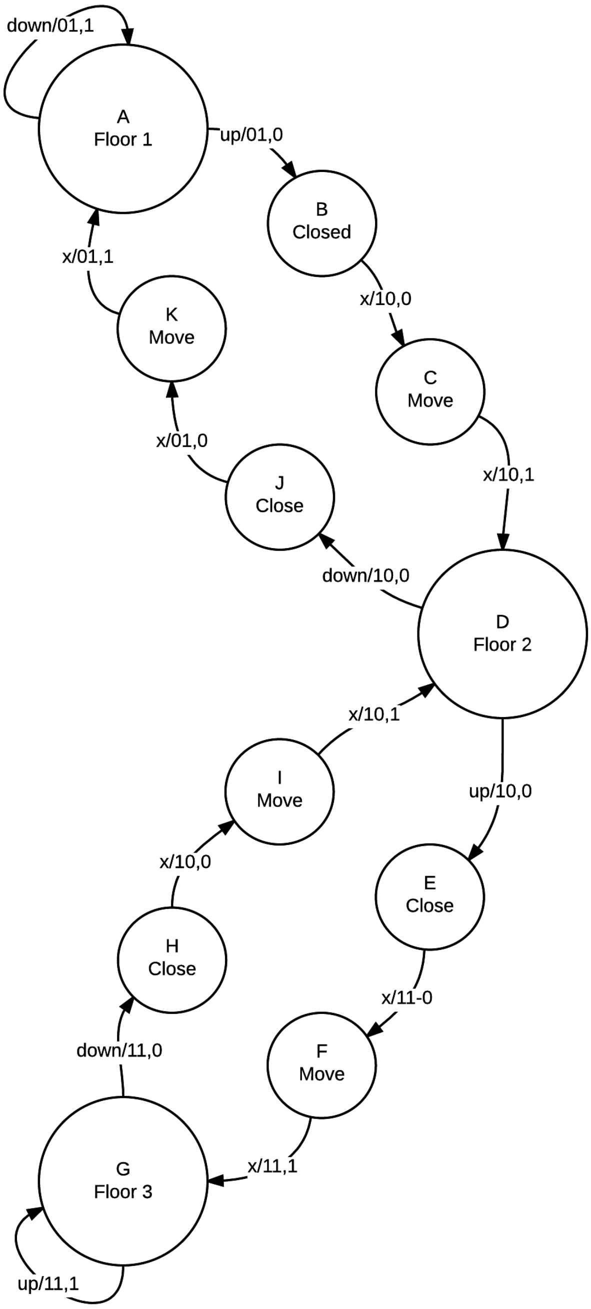


Figure 2 - State Table

up	down	PS	NS	floor	door	D3	D2	D1	D0
0	0	B	C	10	0	0	1	0	0
0	1	B	C	10	0	0	1	0	0
1	0	B	C	10	0	0	1	0	0
1	1	B	X	X	X	X	X	X	X
0	0	A	A	01	1	0	0	1	1
0	1	A	A	01	1	0	0	1	1
1	0	A	B	01	0	0	0	1	0
1	1	A	X	X	X	X	X	X	X
0	0	C	D	10	1	0	1	0	1
0	1	C	D	10	1	0	1	0	1
1	0	C	D	10	1	0	1	0	1
1	1	C	X	X	X	X	X	X	X
0	0	D	D	10	1	0	1	0	1
0	1	D	J	10	0	1	1	0	1
1	0	D	E	10	0	1	1	0	0
1	1	D	X	X	X	X	X	X	X
0	0	F	G	11	1	0	1	1	1
0	1	F	G	11	1	0	1	1	1
1	0	F	G	11	1	0	1	1	1
1	1	F	X	X	X	X	X	X	X
0	0	G	G	11	1	0	1	1	1
0	1	G	H	11	0	1	1	1	0
1	0	G	G	11	1	0	1	1	1
1	1	G	X	X	X	X	X	X	X

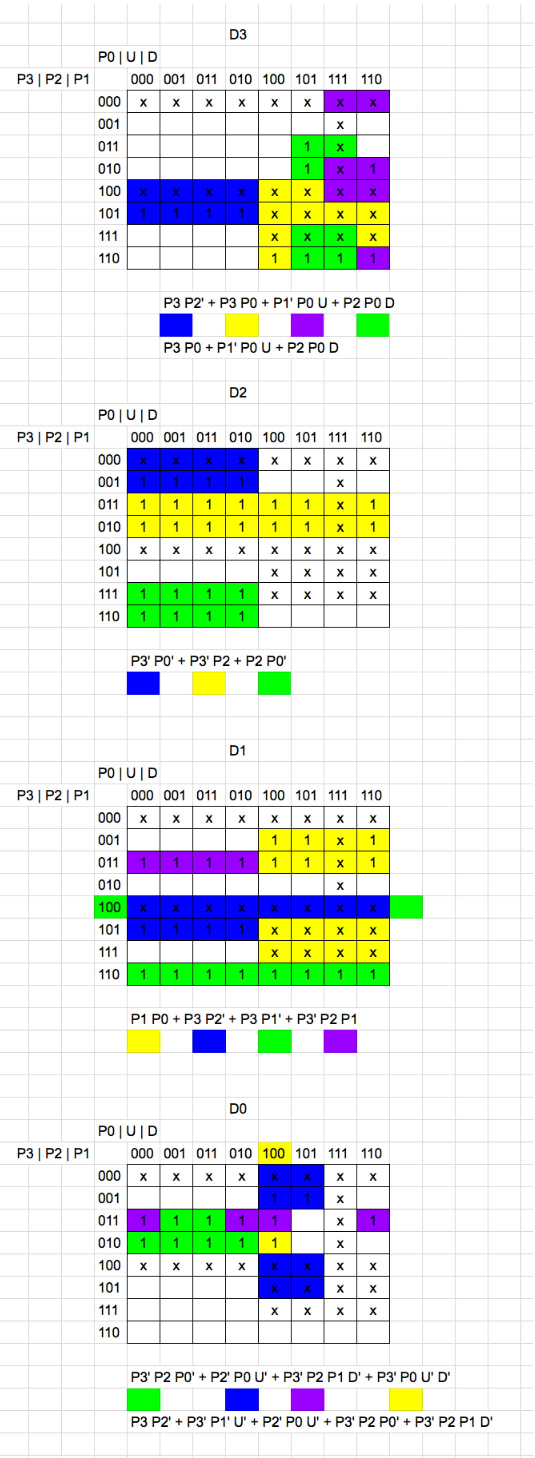
up	down	PS	NS	floor	door	D3	D2	D1	D0
0	0	K	A	01	1	0	0	1	1
0	1	K	A	01	1	0	0	1	1
1	0	K	A	01	1	0	0	1	1
1	1	K	X	X	X	X	X	X	X
0	0	E	F	11	0	0	1	1	0
0	1	E	F	11	0	0	1	1	0
1	0	E	F	11	0	0	1	1	0
1	1	E	X	X	X	X	X	X	X
0	0	J	K	01	0	1	0	1	0
0	1	J	K	01	0	1	0	1	0
1	0	J	K	01	0	1	0	1	0
1	1	J	X	X	X	X	X	X	X
0	0	H	C	10	0	0	1	0	0
0	1	H	C	10	0	0	1	0	0
1	0	H	C	10	0	0	1	0	0
1	1	H	X	X	X	X	X	X	X

Figure 3 - State Assignments

state	code
A	0011
B	0010
C	0100
D	0101
E	1100
F	0110

state	code
G	0111
H	1110
J	1101
K	1010

Figure 4 - K-Maps



D3

P0 | U | D

P3 | P2 | P1

	000	001	011	010	100	101	111	110
000	x	x	x	x	x	x	x	x
001							x	
011						1	x	
010						1	x	1
100	x	x	x	x	x	x	x	x
101	1	1	1	1	x	x	x	x
111					x	x	x	x
110					1	1	1	1

P3 P2' + P3 P0 + P1' P0 U + P2 P0 D

🟦 🟨 🟪 🟩

P3 P0 + P1' P0 U + P2 P0 D

D2

P0 | U | D

P3 | P2 | P1

	000	001	011	010	100	101	111	110
000	x	x	x	x	x	x	x	x
001	1	1	1	1			x	
011	1	1	1	1	1	1	x	1
010	1	1	1	1	1	1	x	1
100	x	x	x	x	x	x	x	x
101					x	x	x	x
111	1	1	1	1	x	x	x	x
110	1	1	1	1				

P3' P0' + P3' P2 + P2 P0'

🟦 🟨 🟩

D1

P0 | U | D

P3 | P2 | P1

	000	001	011	010	100	101	111	110
000	x	x	x	x	x	x	x	x
001					1	1	x	1
011	1	1	1	1	1	1	x	1
010							x	
100	x	x	x	x	x	x	x	x
101	1	1	1	1	x	x	x	x
111					x	x	x	x
110	1	1	1	1	1	1	1	1

P1 P0 + P3 P2' + P3 P1' + P3' P2 P1

🟨 🟦 🟩 🟪

D0

P0 | U | D

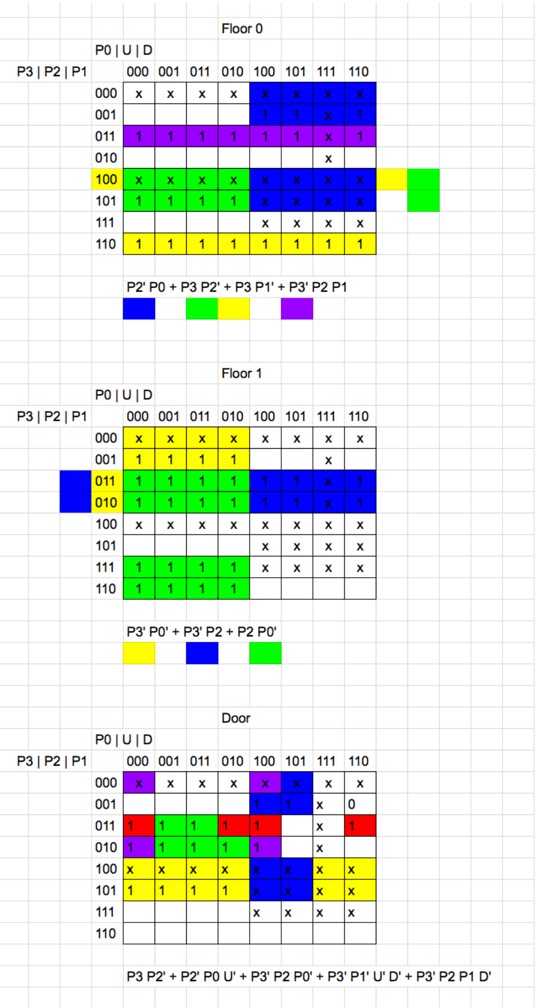
P3 | P2 | P1

	000	001	011	010	100	101	111	110
000	x	x	x	x	x	x	x	x
001					1	1	x	
011	1	1	1	1	1		x	1
010	1	1	1	1	1		x	
100	x	x	x	x	x	x	x	x
101					x	x	x	x
111					x	x	x	x
110								

P3' P2 P0' + P2' P0 U' + P3' P2 P1 D' + P3' P0 U' D'

🟩 🟦 🟪 🟨

P3 P2' + P3' P1' U' + P2' P0 U' + P3' P2 P0' + P3' P2 P1 D'



Floor 0

P0 | U | D

P3 | P2 | P1

	000	001	011	010	100	101	111	110
000	x	x	x	x	x	x	x	x
001					1	1	x	1
011	1	1	1	1	1	1	x	1
010							x	
100	x	x	x	x	x	x	x	x
101	1	1	1	1	x	x	x	x
111					x	x	x	x
110	1	1	1	1	1	1	1	1

P2' P0 + P3 P2' + P3 P1' + P3' P2 P1

🟦 🟩🟨 🟪

Floor 1

P0 | U | D

P3 | P2 | P1

	000	001	011	010	100	101	111	110
000	x	x	x	x	x	x	x	x
001	1	1	1	1			x	
011	1	1	1	1	1	1	x	1
010	1	1	1	1	1	1	x	1
100	x	x	x	x	x	x	x	x
101					x	x	x	x
111	1	1	1	1	x	x	x	x
110	1	1	1	1				

P3' P0' + P3' P2 + P2 P0'

🟨 🟦 🟩

Door

P0 | U | D

P3 | P2 | P1

	000	001	011	010	100	101	111	110
000	x	x	x	x	x	x	x	x
001					1	1	x	0
011	1	1	1	1	1		x	1
010	1	1	1	1	1		x	
100	x	x	x	x	x	x	x	x
101	1	1	1	1	x	x	x	x
111					x	x	x	x
110								

P3 P2' + P2' P0 U' + P3' P2 P0' + P3' P1' U' D' + P3' P2 P1 D'

Figure 5 - Logisim Sequential Circuit

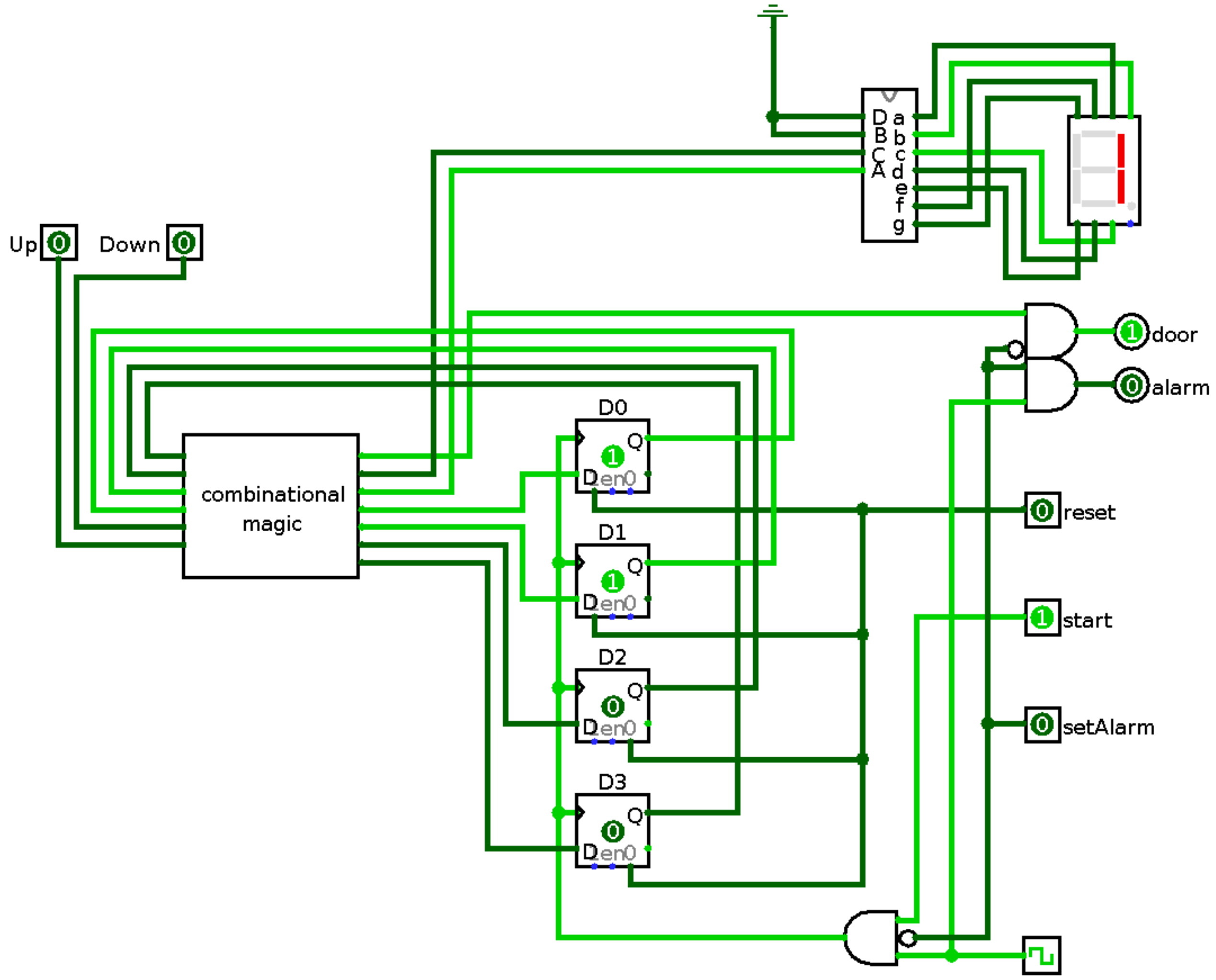
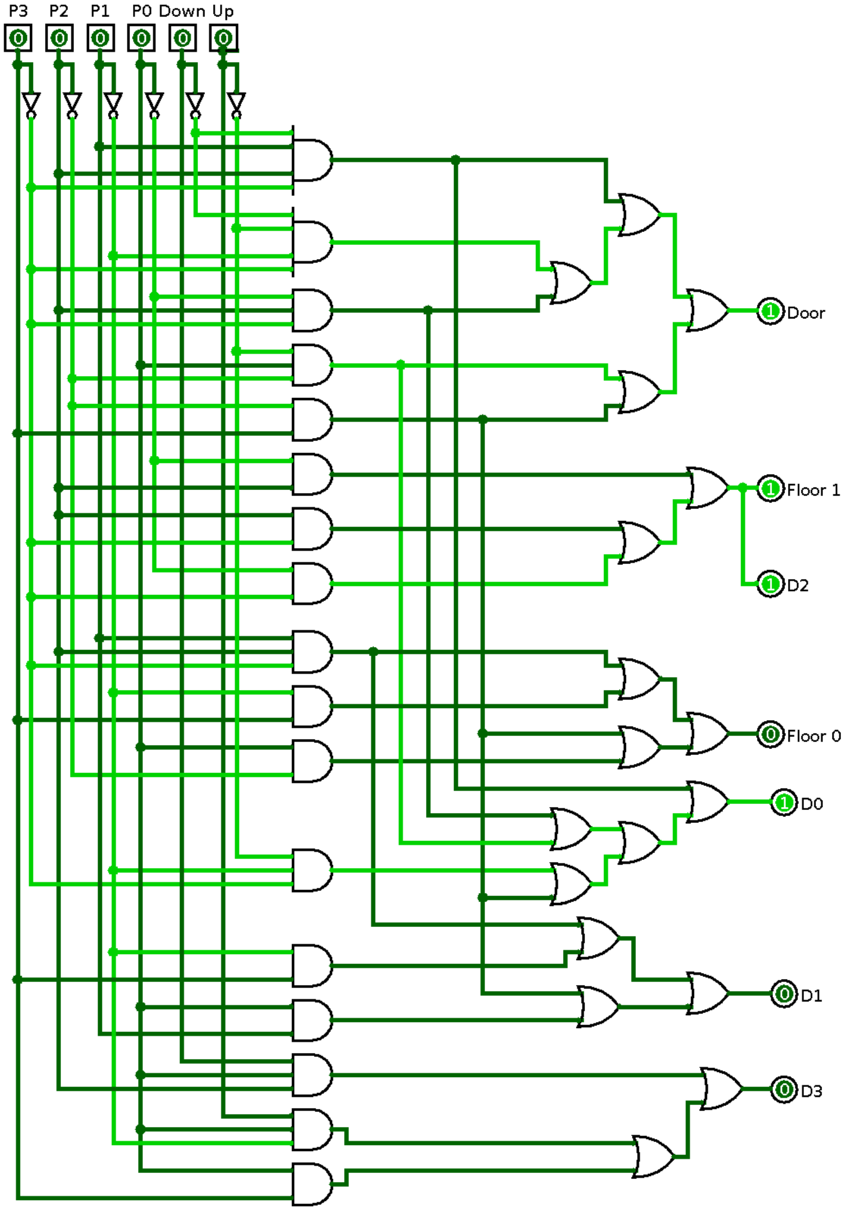


Figure 6 - Logisim Combinational Circuit


P3
P2
P1
P0
Down
Up
0
0
0
0
0
0
1
Door
1
Floor 1
1
D2
0
Floor 0
1
D0
0
D1
0
D3