Lab #3

Combinational Network Design using Muxes and Adders

Lab Report

Bradley Manzo
916953788

TABLE OF CONTENTS

INTRODUCTION

OBJECTIVES

PART I

- Learn how to analyze a provided circuit and determine the correct output values of low level circuit elements.
- Learn how to translate low level logic circuits to symbols and then implement them in a top level circuit in the same project.
- Learn how a combination inputs going into a Multiplexer and a decoder can produce a decimal output

PART II

- Learn how a Full adder can be implemented in a Ripple Carry Circuit.

PART III

- Learn how both aspects of parts I and II can be combined.

PROCEDURE

DESIGN AND TEST PROCEDURE

PART I

- To begin this section of the Lab I analyzed the behavior of the provided circuit components including the comparator, circuit a, and circuit b. I created truth tables for all of them and, using kmaps, simplified them to their general equations as seen in *Figure 1*. For Z since there was only one input, I simplified by specifying which outputs were always high or always low, and then specified which outputs were directly tied to the input Z.
- I then created symbols for these three components. By using the general equations, I created lower level logic circuits to generate the proper outputs for their respective inputs. The symbols for Comparator and Circuit A can be seen in *Figure 2* and *Figure 3*

respectively. Circuit B does not require a symbol since it is already very simple.

- I then implemented the symbols in a larger top level circuit verifying my connections were identical to the provided circuit diagram. I used the recommended 74157 MUX and 7447 Decoder as seen in *Figure 4.*
- After debugging and compiling my circuit, I simulated it using Altera-Sim and uploaded it to my DE10-Lite Board to verify its behavior as seen in *Figure 5.*

PART II

- I began this section by creating a symbol for Full Adder as seen in *Figure 6,* using the provided schematic. I then implemented this into the provided Ripple Carry with Full Adder Schematic as seen in *Figure 7.*
- After debugging and compiling the circuit, I simulated 15 unique inputs and recorded the resulting waveform in *Figure 8.* I then downloaded the circuit to my DE10-Lite Board to verify its behavior.

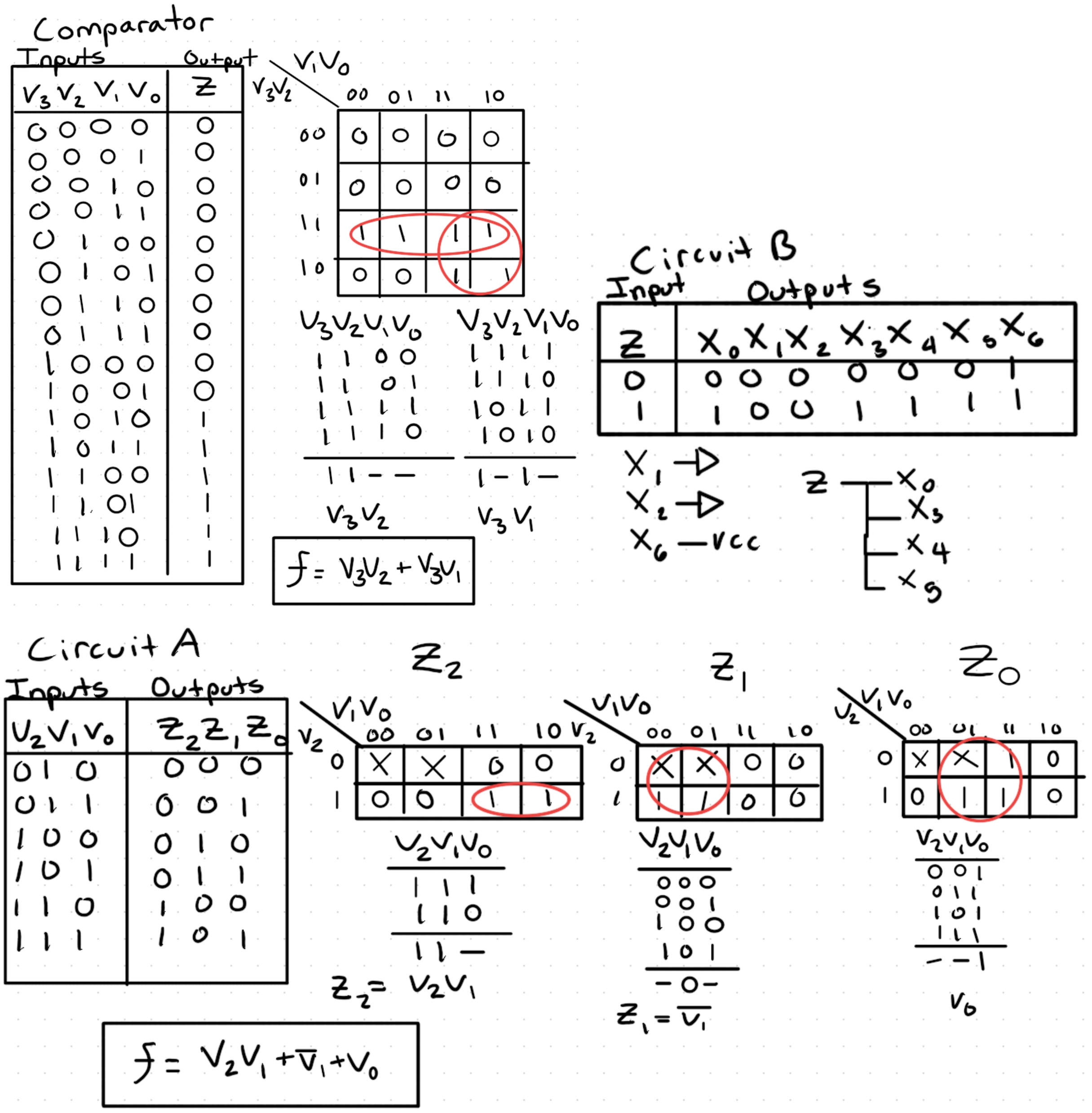
PART III

- I copied everything from Parts I and II into a new project folder and connected the LEDR[0..3] outputs of the Ripple Carry using Full adder circuit to the SW[0..3] inputs of the Binary to BCD converter circuit.
- I downloaded the circuit to my DE10-Lite board to verify its behavior was identical to Part II, while displaying a decimal number like in Part I.

ANALYSIS/RESULTS

DATA

PART I:



Comparator

Inputs				Output
V_3	V_2	V_1	V_0	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

$$\begin{array}{c|cccc} V_3V_2 \backslash V_1V_0 & 00 & 01 & 11 & 10 \\ \hline 00 & 0 & 0 & 0 & 0 \\ 01 & 0 & 0 & 0 & 0 \\ 11 & 1 & 1 & 1 & 1 \\ 10 & 0 & 0 & 1 & 1 \end{array}$$

$V_3V_2V_1V_0$	$V_3V_2V_1V_0$
1 1 0 0	1 1 1 1
1 1 0 1	1 1 1 0
1 1 1 1	1 0 1 1
1 1 1 0	1 0 1 0
1 1 – –	1 – 1 –
V_3V_2	V_3V_1

$$\boxed{f = V_3V_2 + V_3V_1}$$

Circuit B

Input	Outputs						
Z	X_0	X_1	X_2	X_3	X_4	X_5	X_6
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1

X_1 —▷
X_2 —▷
X_6 — Vcc

Z — X_0, X_3, X_4, X_5

Circuit A

Inputs			Outputs		
V_2	V_1	V_0	Z_2	Z_1	Z_0
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	0	1

Z_2

$$\begin{array}{c|cccc} V_2 \backslash V_1V_0 & 00 & 01 & 11 & 10 \\ \hline 0 & X & X & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{array}$$

$V_2V_1V_0$
1 1 1
1 1 0
1 1 –

$Z_2 = V_2V_1$

Z_1

$$\begin{array}{c|cccc} V_2 \backslash V_1V_0 & 00 & 01 & 11 & 10 \\ \hline 0 & X & X & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{array}$$

$V_2V_1V_0$
0 0 0
0 0 1
1 0 0
1 0 1
– 0 –

$Z_1 = \overline{V_1}$

Z_0

$$\begin{array}{c|cccc} V_2 \backslash V_1V_0 & 00 & 01 & 11 & 10 \\ \hline 0 & X & X & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{array}$$

$V_2V_1V_0$
0 0 1
0 1 1
1 0 1
1 1 1
– – 1

V_0

$$\boxed{f = V_2V_1 + \overline{V}_1 + V_0}$$

Figure 1: Truth Tables and General Equations

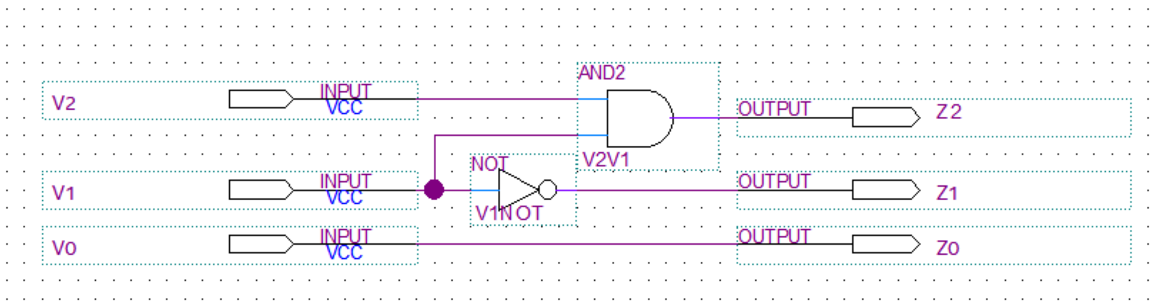


Figure 2: Circuit A

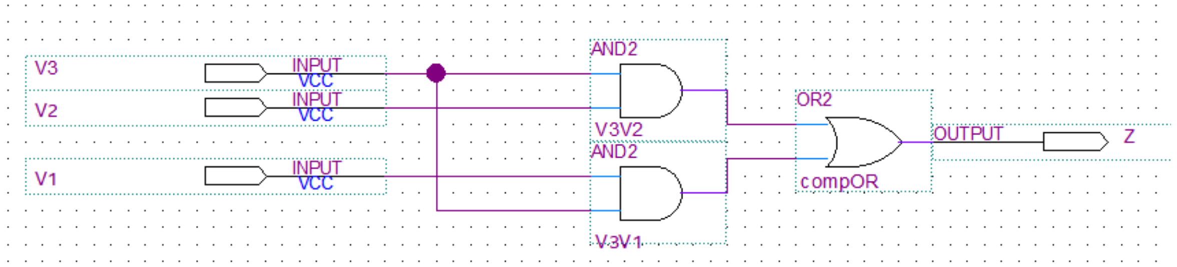


Figure 3: Comparator

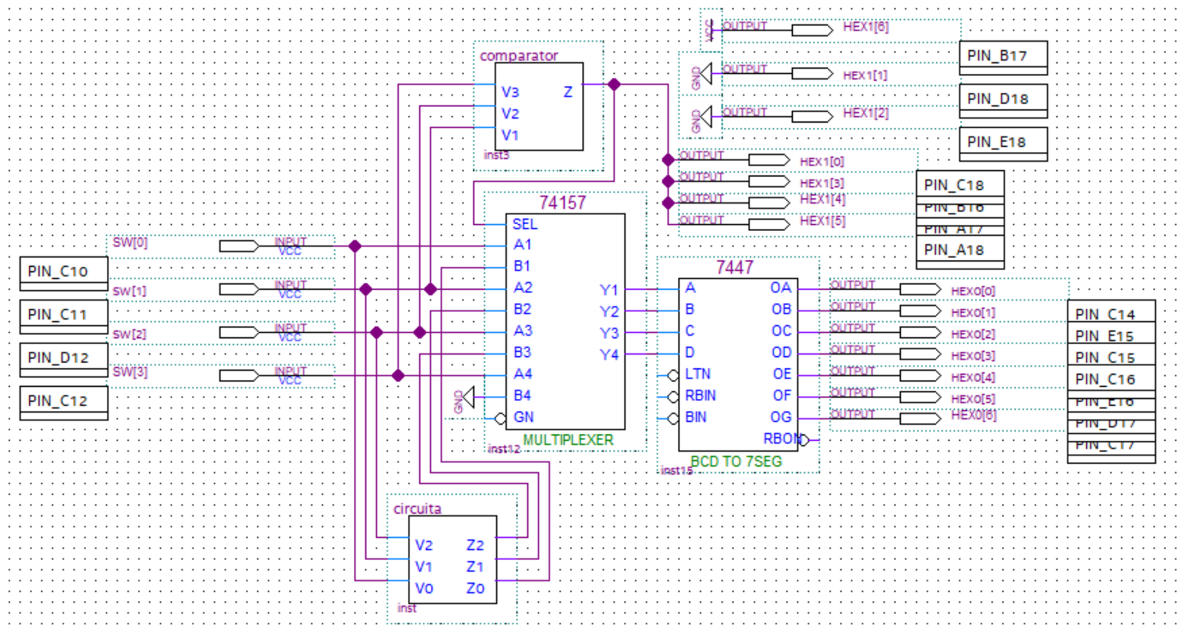


Figure 4: Binary to BCD Converter Circuit

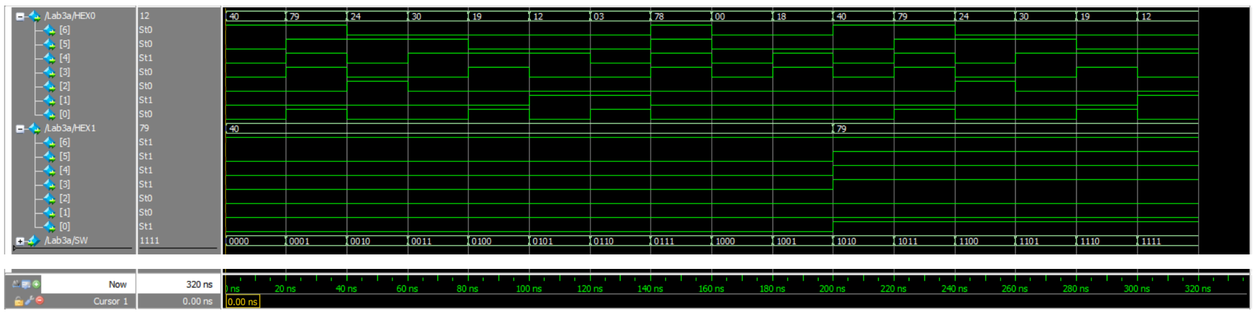


Figure 5: Binary to BCD Waveform

PART II:

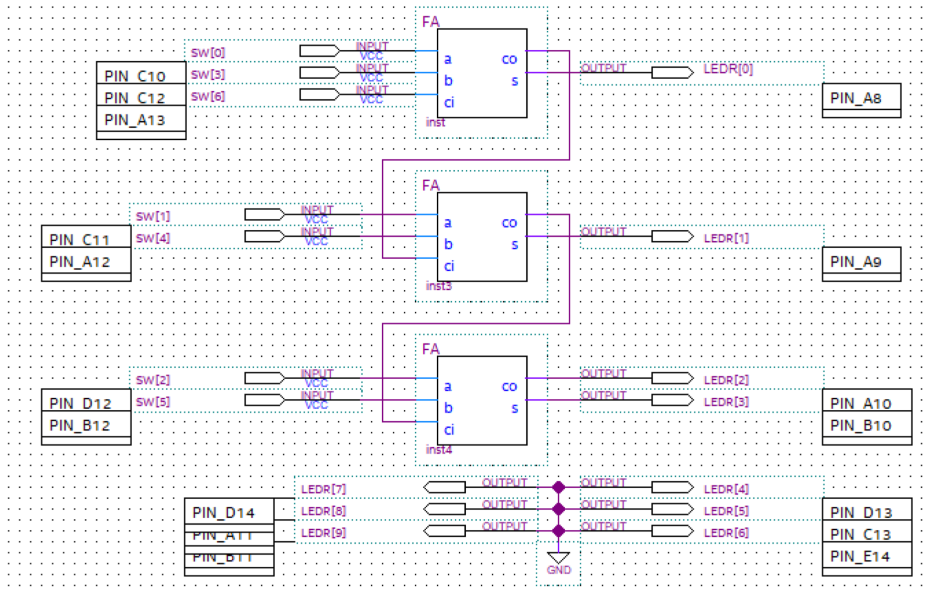


Figure 6: Ripple Carry using Full Adder Circuit

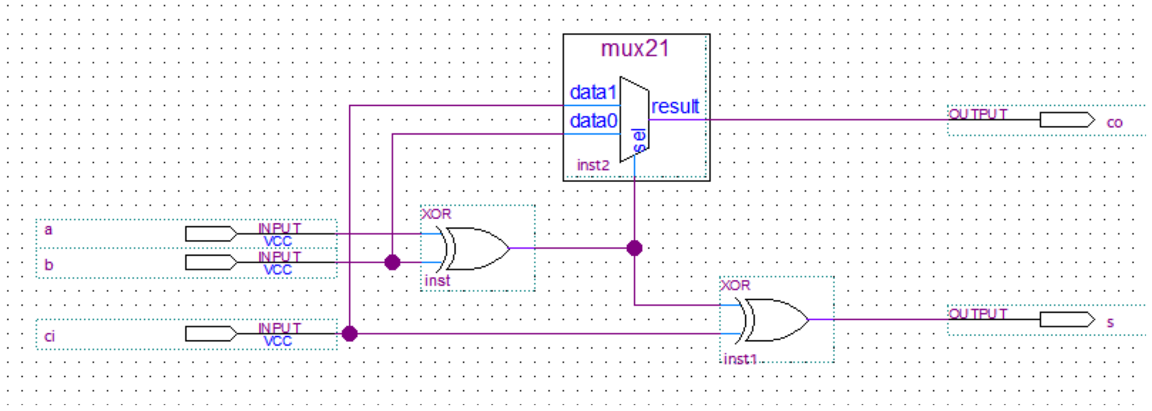


Figure 7: Full Adder Circuit

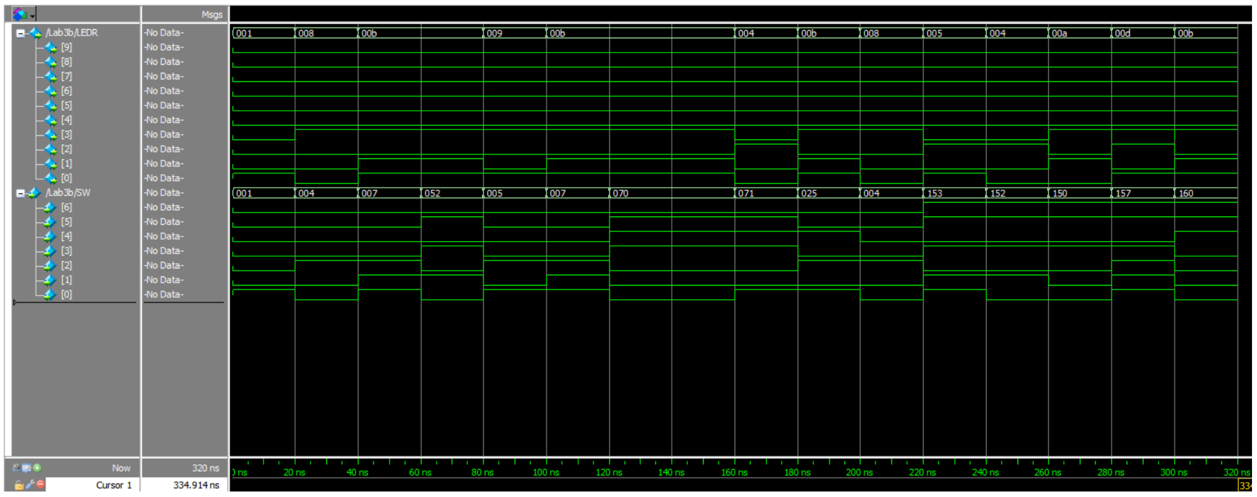


Figure 8: Ripple Carry using Full Adder Waveform

PART III:

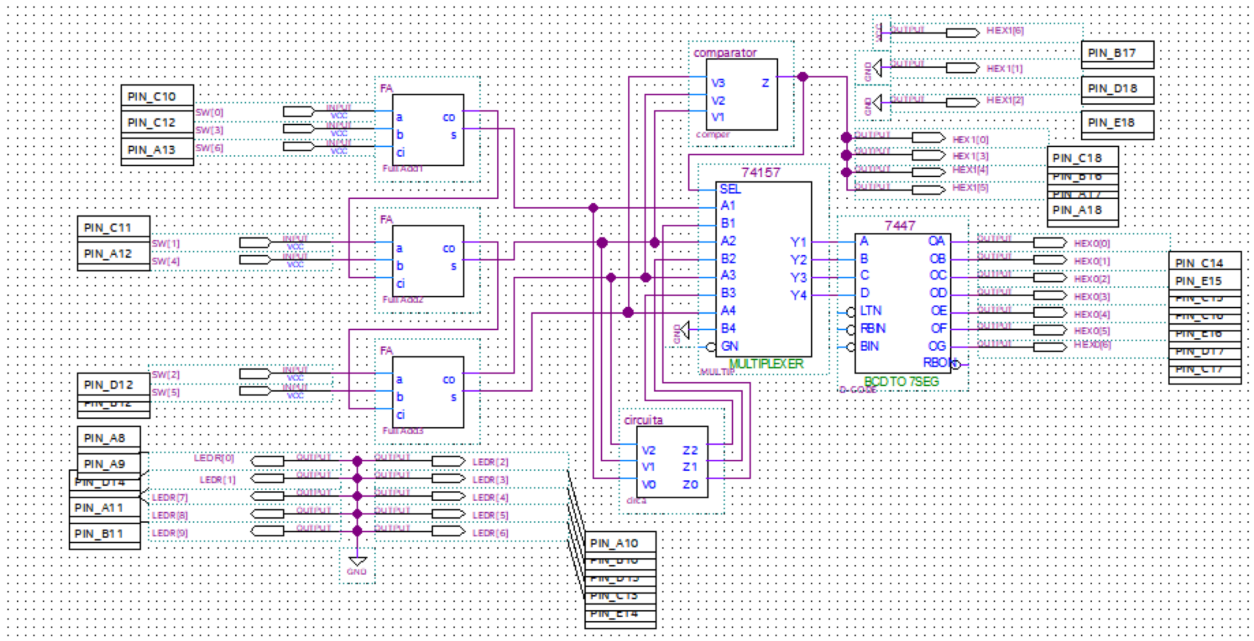


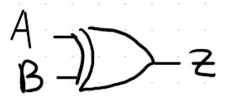
Figure 9: Ripple Carry Adder with Decimal Display circuit

Full Adder

	Inputs			Outputs	
	a	b	ci	co	s
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	0

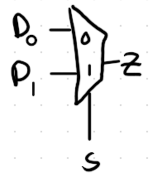
XOR Gate

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0



2to1 Mux

S	D_0	D_1	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



$\bar{S} \rightarrow D_0$

$S \rightarrow D_1$

CO

a \ b ci	00	01	11	10
0	0	0	1	0
1	0	1	1	1

a b ci
0 1 1
1 1 1
– 1 1
b ci

a b ci
1 0 1
1 1 1
1 – 1
a ci

a b ci
1 1 1
1 1 0
1 1 –
ab

$\boxed{f = bci + aci + ab}$

S

a \ b ci	00	01	11	10
0	0	1	0	1
1	1	0	0	0

$\boxed{f = \bar{a}\bar{b}ci + \bar{a}b\bar{ci} + a\bar{b}\bar{ci}}$

Figure 10: Full Adder Truth Table and Kmaps

ANALYZING RESULTS

PART I

- After completing part I of the lab, I originally had trouble getting the proper general equation for Z2 of circuit A, but after understanding that kmaps can be used to describe any number of outputs by replacing the empty values with don't/cares. Additionally my SW[] inputs were giving me trouble after wiring them in the configuration of the provided schematic however, after reversing the order my circuit worked as expected and was verified by the TA

PART II

- For part II of the lab I had little difficulty getting the provided schematic to work on the simulator and my DE-10 Lite Board. The full adder works as expected and was verified by the TA.

PART III

- For this part of the lab I had some difficulty wiring my circuits together properly. But after correctly importing my low level symbols into my project folder I had little difficulty getting my board to work identically to part II but with the decimal output of part I. The circuit works as expected and was verified by the TA.

ANSWERS TO HIGHLIGHTED QUESTIONS

- Is there an advantage to Figure 2a compared to the SoP equations?
  - If we use kmaps to find the SoP equations of the Full adder, we find that they are horribly inefficient, the S output for example has 3 prime implicants, and would require 3 AND gates and an OR gate. In total, the SoP would take at least 8 gates to implement, while the full adder only takes 2 XORs and a MUX.

- What is the 'octal' radix and why is it convenient to represent the SW[] input in octal inthe Part II simulation?
  - The octal radix appears to treat the SW[] inputs as an octal number rather than a binary, this way we can add larger numbers.
- Why isn't octal convenient for the LEDR[] output?
  - It wouldn't be convenient for the LEDR[] output since the answer is in binary and more easily translates to the LEDs.

CONCLUSION

- After completing part I of the lab, I can say for certain that I've gained a greater understanding of implementing low level logic circuits as symbols to reduce complexity and increase fluency. I've also expanded my understanding of how kmaps can be used to reduce multiple outputs to general equations, and how you can integrate the don't/cares into your kmap simplifications. I also feel I've developed a greater understanding of how multiplexers and decoders can work in series to produce a programmed decimal output from any number of digital inputs. I've also learned how Full adders can be implemented in order to build a ripple carry adder and add 2 binary inputs and output the sum. That being said, part III makes sense as a combination of these two principles, adding 2 binary inputs together to produce a decimal output.