ECEN 2350 Digital Logic:

Project 1 Report

Ramon Martinez

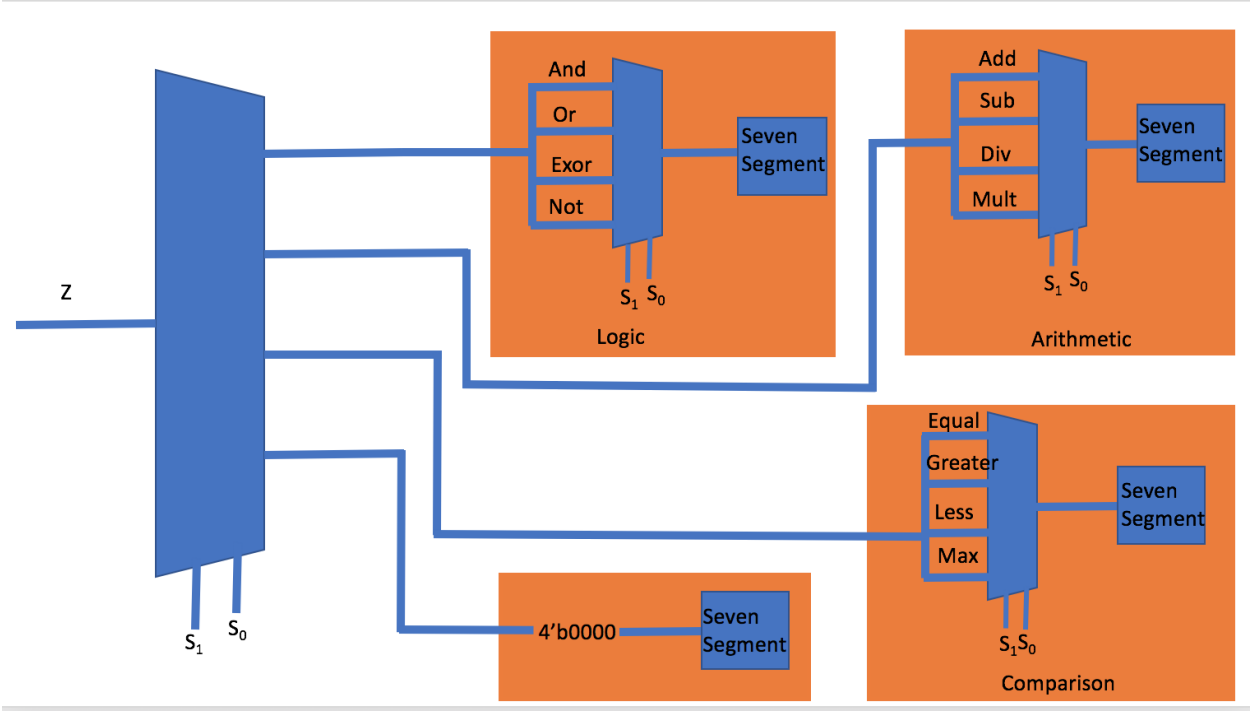
Sean DeVaney

Rafael Espinoza

Introduction:

For this project we've developed code that takes in number inputs from switches on the board and uses multiple modules to complete operations with them. To implement said operations, the two buttons are used to determine which module is used while the last two switches (SW8 & SW9) determine which operation is done. The rest if the switches (SW0 – SW7) are used to represent our numbered inputs in binary . If there is Overflow, LED0 lights up, and LED 8 & 7 signal which module is on currently.

Block Diagram:



Description of Code:

Top.v: Top first initializes all of the leds. It takes in the inputs from the switches and assigns them as a new value z. It then calls Multiplexer3 twice to decide whether SW8 & SW9 should be on, and then it checks if one of the buttons is pressed, to turn off the LEDs accordingly. It calls Multiplexer2 to obtain the results of the operations, which are passed into SevenSegment where they're turned into values that can be readable on a SSD. If there is overflow, LED0 will light up. The values are finally passed onto the according Hex to represent the correct numbers.

Multiplexer3.v: This module detects if one of the buttons was pressed, and assigns a value to turn on one of the LEDs if it is and turn off if it isn't.

Multiplexer2.v: Multiplexer takes the values to be passed and the selected values for the buttons and calls a module to which the selected button combination is assigned to.

Logical.v: This module takes in two values, x and y from the multiplexer and creates a new z value, comprised of x and y combined, and computes logical operations with four modules: And, Or, Xor and Not. And computes

and returns the and operation for x and y. Or computes and returns the or operation for x and y. Xor computes and returns the xor operation for x and y. Not takes the z value and computes and returns the not of z. The output is passed to Multiplexer.

Arithmetic.v: This module takes in two values, x and y from the multiplexer and creates a new z value, comprised of x and y combined, and computes arithmetical operations using three modules: AdderSubtractor, Multiple, and Divide. AdderSubtractor determines whether the operation is add or subtract, flipping the bits of y accordingly, and calls another module, Fourbit Adder, which calls Twobit adder and does ripplecarry add. Then, Fourbit adder checks for overflow or a carry bit and assigns 1 to Overflow if it finds one and 0 if it doesn't. Multiple takes in z and checks if z is greater than 127. If it is, then z is going to produce overflow and we assign 0 to z and 1 to Overflow. If it's less than 127, we do a left shift on z, return it and set Overflow to 0. Divide does a right shift on z and returns the value. The output is passed to Multiplexer.

Comparison.v: This module takes in two values, x and y from the mutiplexer and computes comparison operations with four modules: Equal,

Greater, Lessthan and Max. Equal checks if the two values are the same and returns 1 if the are and 0 if they aren't. Greater checks if x is greater than y and returns 1 if it is and 0 if it isn't. Lessthan checks if x is less than y and returns 1 if it is and 0 if it isn't. Max checks which value is bigger and returns the biggest value. The output is passed to Multiplexer.

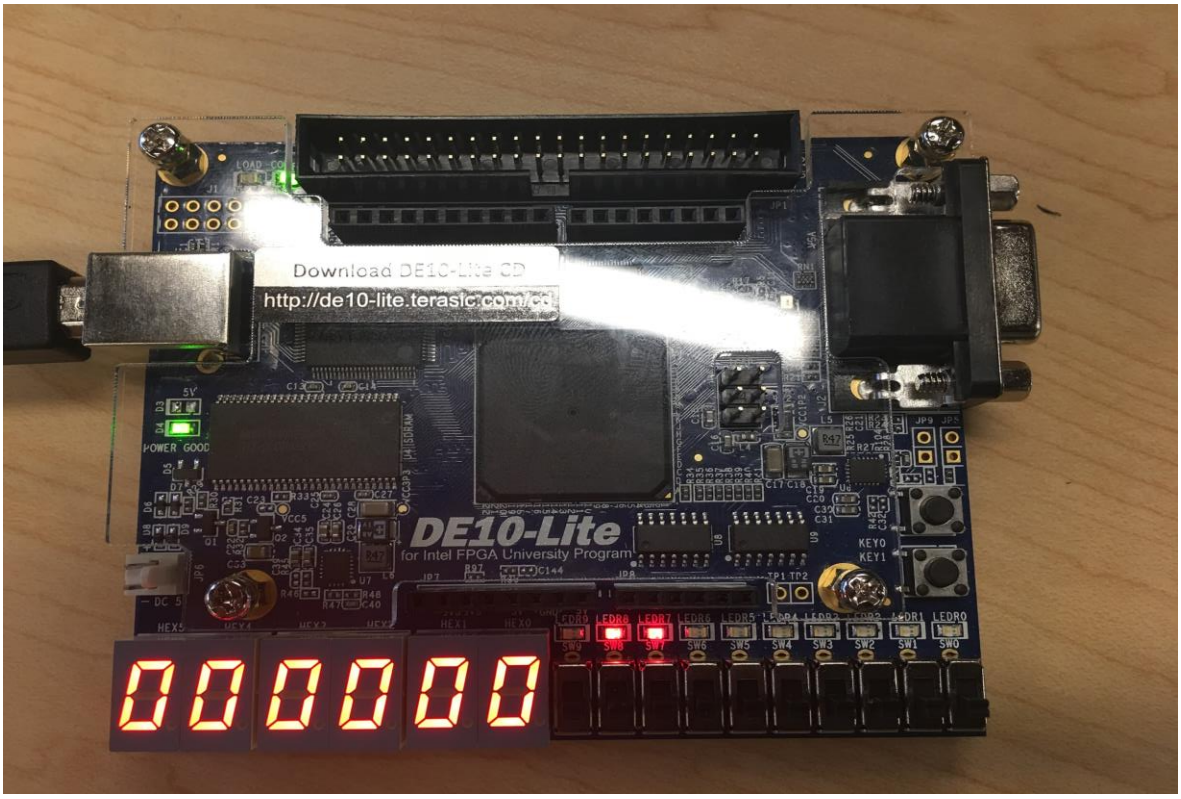
Multiplexer.v: This multiplexer takes in the value of one of the modules and returns the one that is currently selected through the combination of buttons.

SevenSegment.v: This module takes in a number input SS and determines the value of parts. Parts is assigned to be the a number that corresponds to SS in the seven segment display, and lights up the each part of the ssd accordingly.

Bitwise Encoding Table:

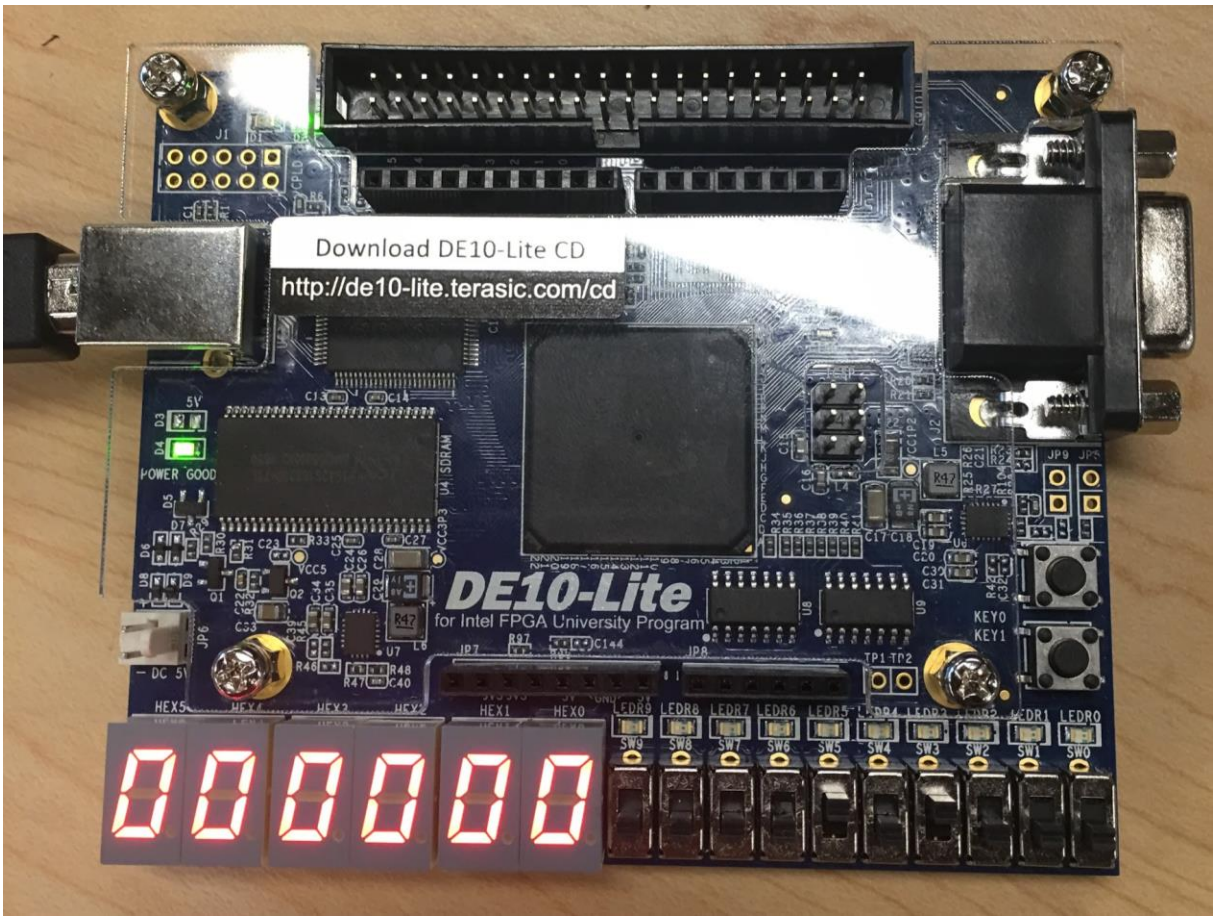
Button 1	Button 2	Switch 9	Switch 8	Mode	Operation
0	0	0	0	Logical	And
0	0	0	1	Logical	Or
0	0	1	0	Logical	Xor
0	0	1	1	Logical	Not
0	1	0	0	Arithmetic	Add
0	1	0	1	Arithmetic	Subtract
0	1	1	0	Arithmetic	Multiply
0	1	1	1	Arithmetic	Divide
1	0	0	0	Comparison	Equal
1	0	0	1	Comparison	Greater
1	0	1	0	Comparison	Less
1	0	1	1	Comparison	Max

Final Implementation and Working Board:



Board display when both selecting bits for Multiplexer 2 are 1.

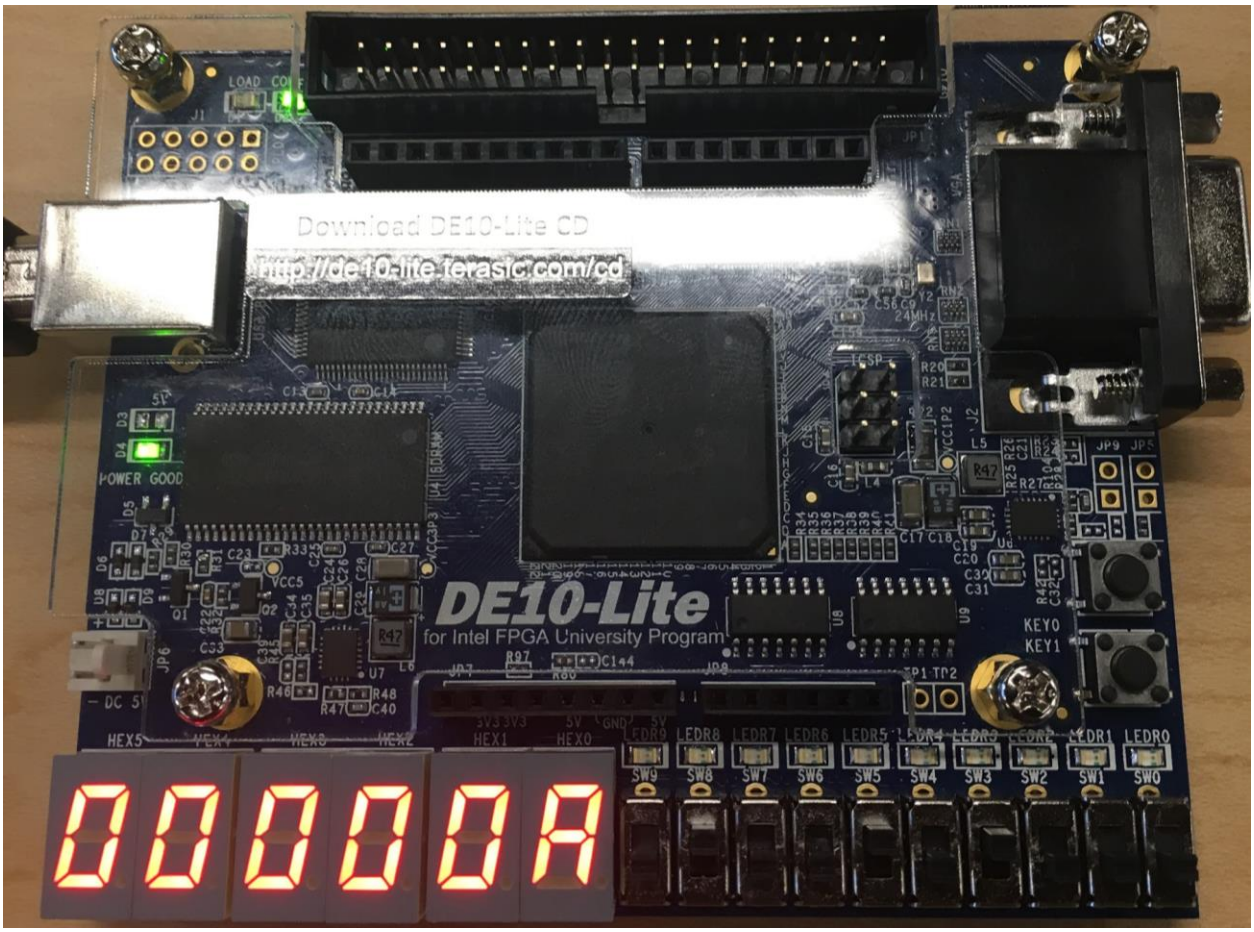
Logical Mode-And Operation:



Inputs:

- Button 1 = $(0)_2$
- Button 2 = $(0)_2$
- Switches 0-3 = $(1000)_2$ = $(8)_{10}$
- Switches 4-7 = $(0001)_2$ = $(1)_{10}$
- Switch 9 = $(0)_2$
- Switch 8 = $(0)_2$

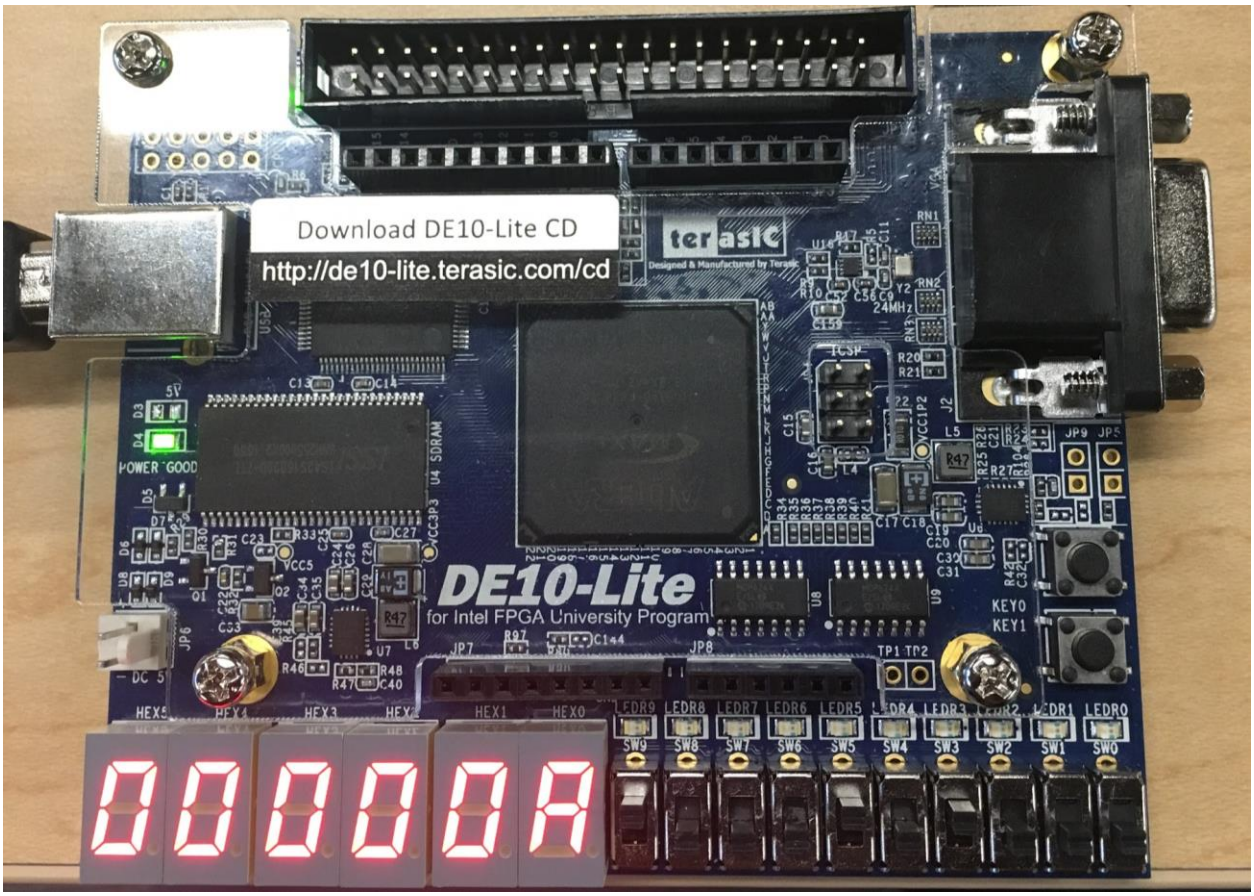
Logic Mode-Or Operation:



Inputs:

- Button 1 = $(0)_2$
- Button 2 = $(0)_2$
- Switches 0-3 = $(1000)_2 = (8)_{10}$
- Switches 4-7 = $(0001)_2 = (1)_{10}$
- Switch 9 = $(0)_2$
- Switch 8 = $(1)_2$

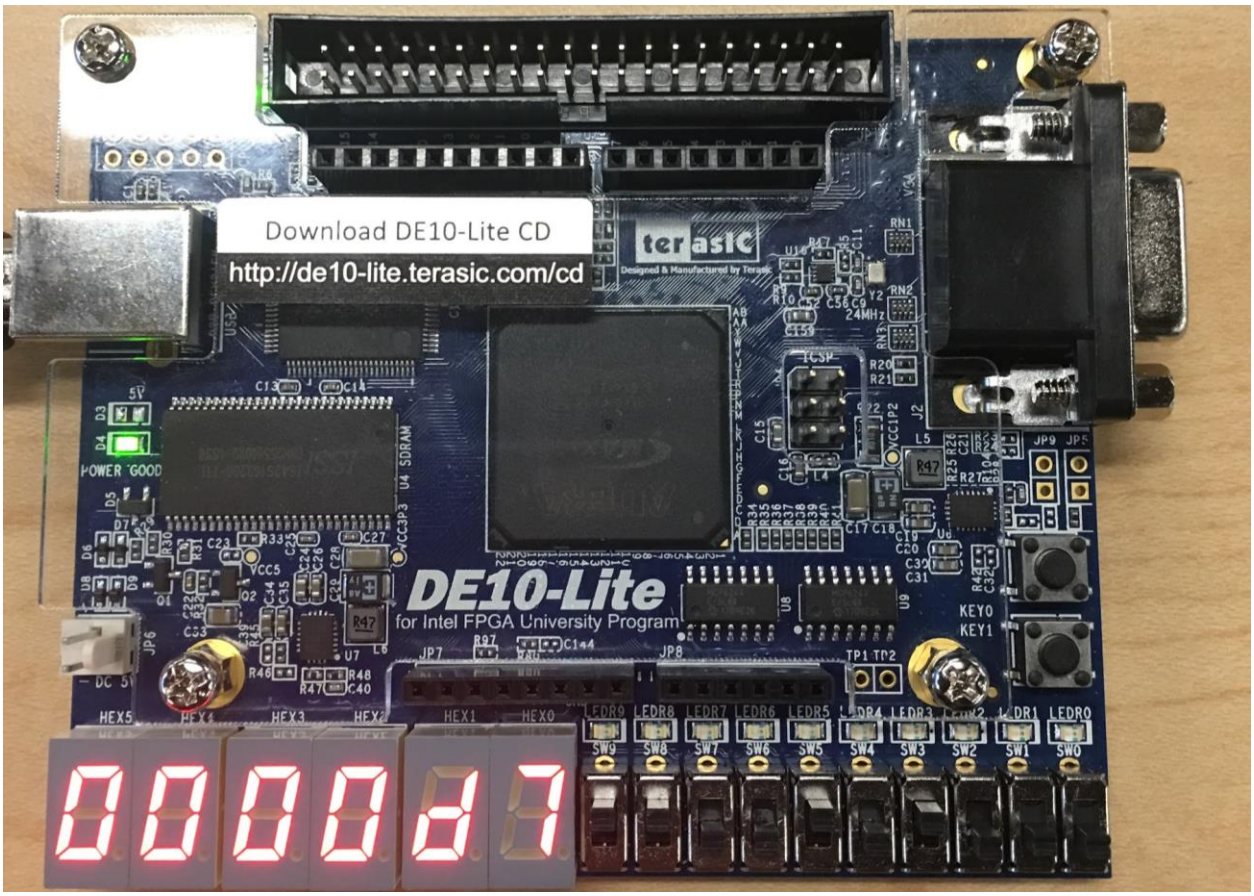
Logical Mode- Xor Operation:



Inputs:

- Button 1 = $(0)_2$
- Button 2 = $(0)_2$
- Switches 0-3 = $(1000)_2 = (8)_{10}$
- Switches 4-7 = $(0001)_2 = (1)_{10}$
- Switch 9 = $(1)_2$
- Switch 8 = $(0)_2$

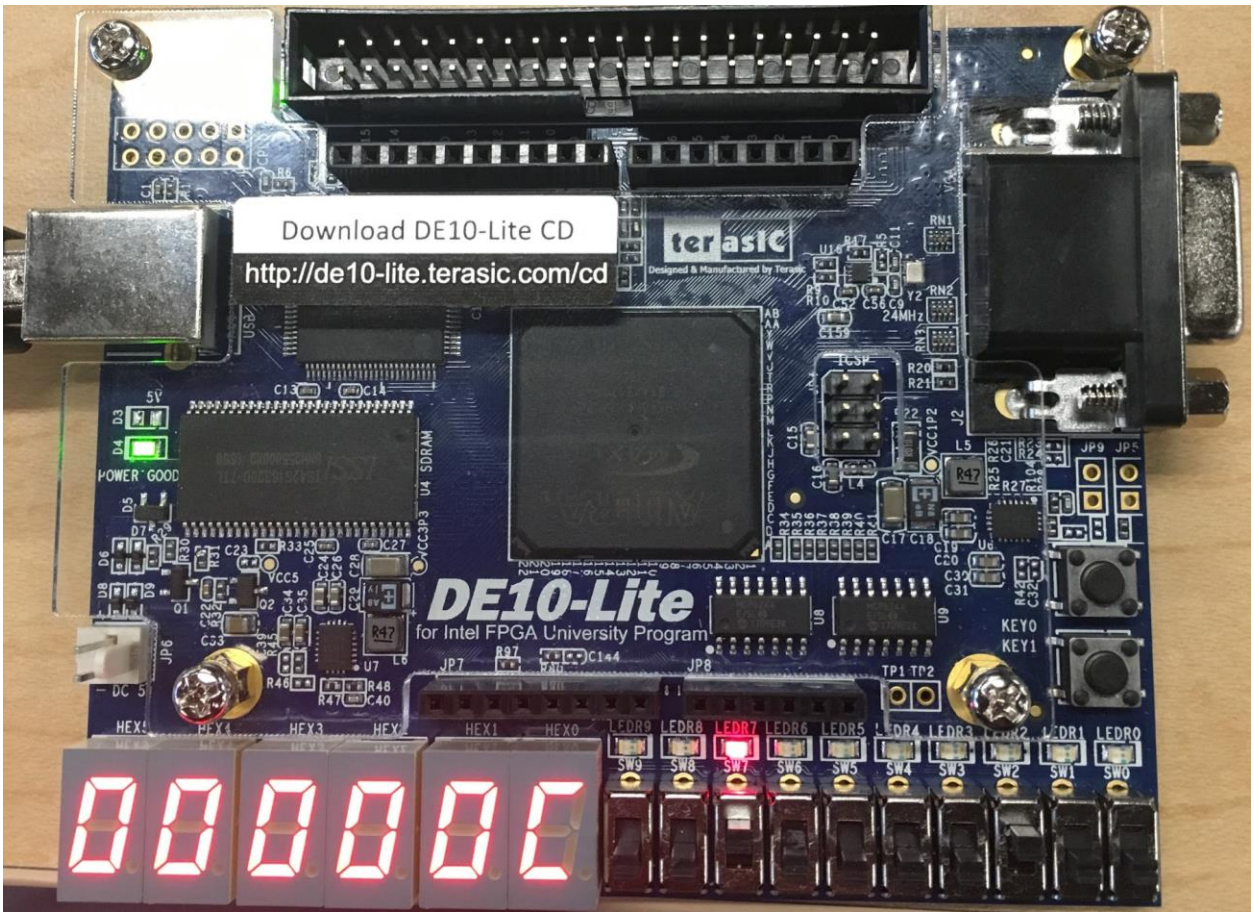
Logical Mode-Not Operation:



Inputs:

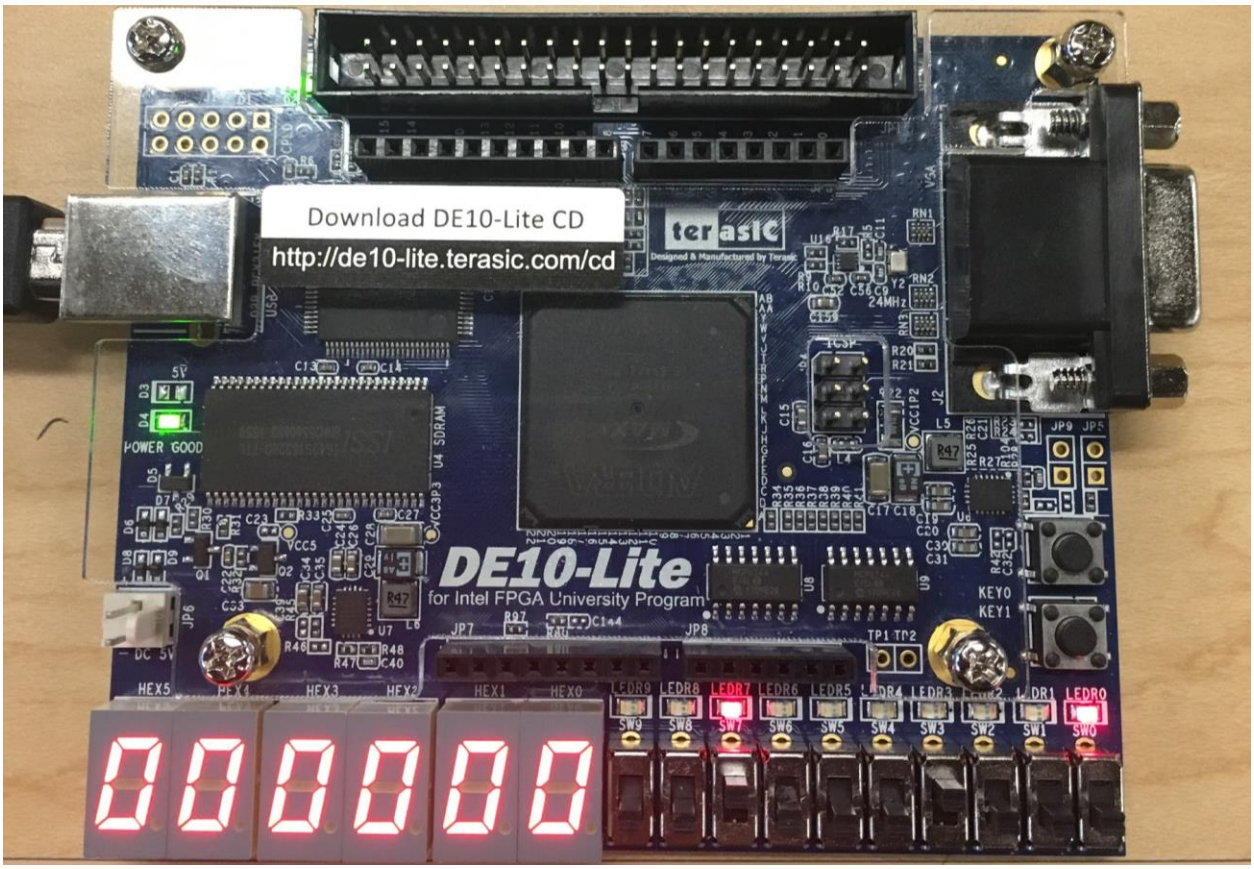
- Button 1 = $(0)_2$
- Button 2 = $(0)_2$
- Switches 7-0 = $(00101000)_2$ = $(40)_{10}$
- Switch 9 = $(1)_2$
- Switch 8 = $(1)_2$

Arithmetic Mode-Add Operation:



Inputs:

- Button 1 = $(0)_2$
- Button 2 = $(1)_2$
- Switches 0-3 = $(0100)_2$ = $(4)_{10}$
- Switches 4-7 = $(1000)_2$ = $(8)_{10}$
- Switch 9 = $(0)_2$
- Switch 8 = $(0)_2$

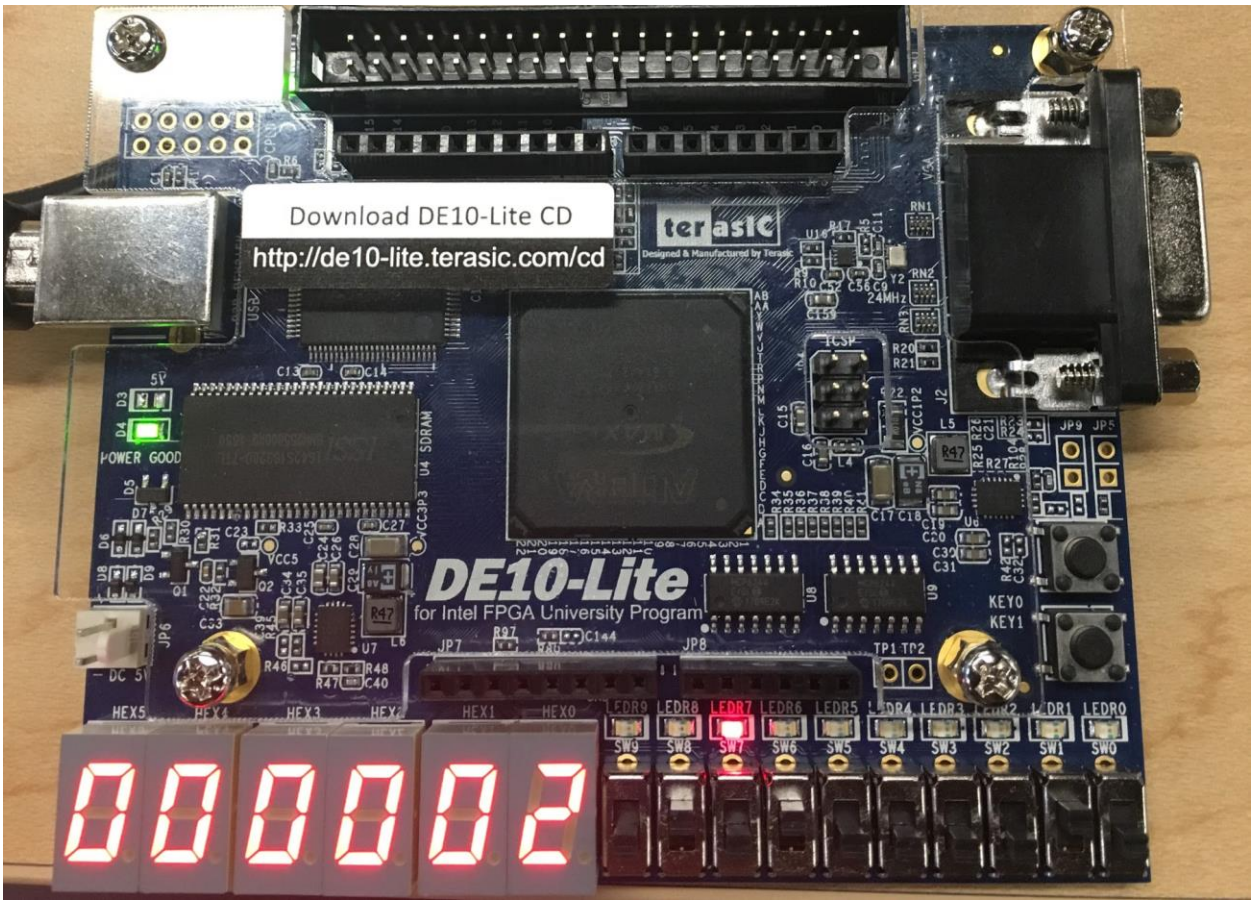


Inputs:

- Button 1 = $(0)_2$
- Button 2 = $(1)_2$
- Switches 0-3 = $(1000)_2 = (8)_{10}$
- Switches 4-7 = $(1000)_2 = (8)_{10}$
- Switch 9 = $(0)_2$
- Switch 8 = $(0)_2$

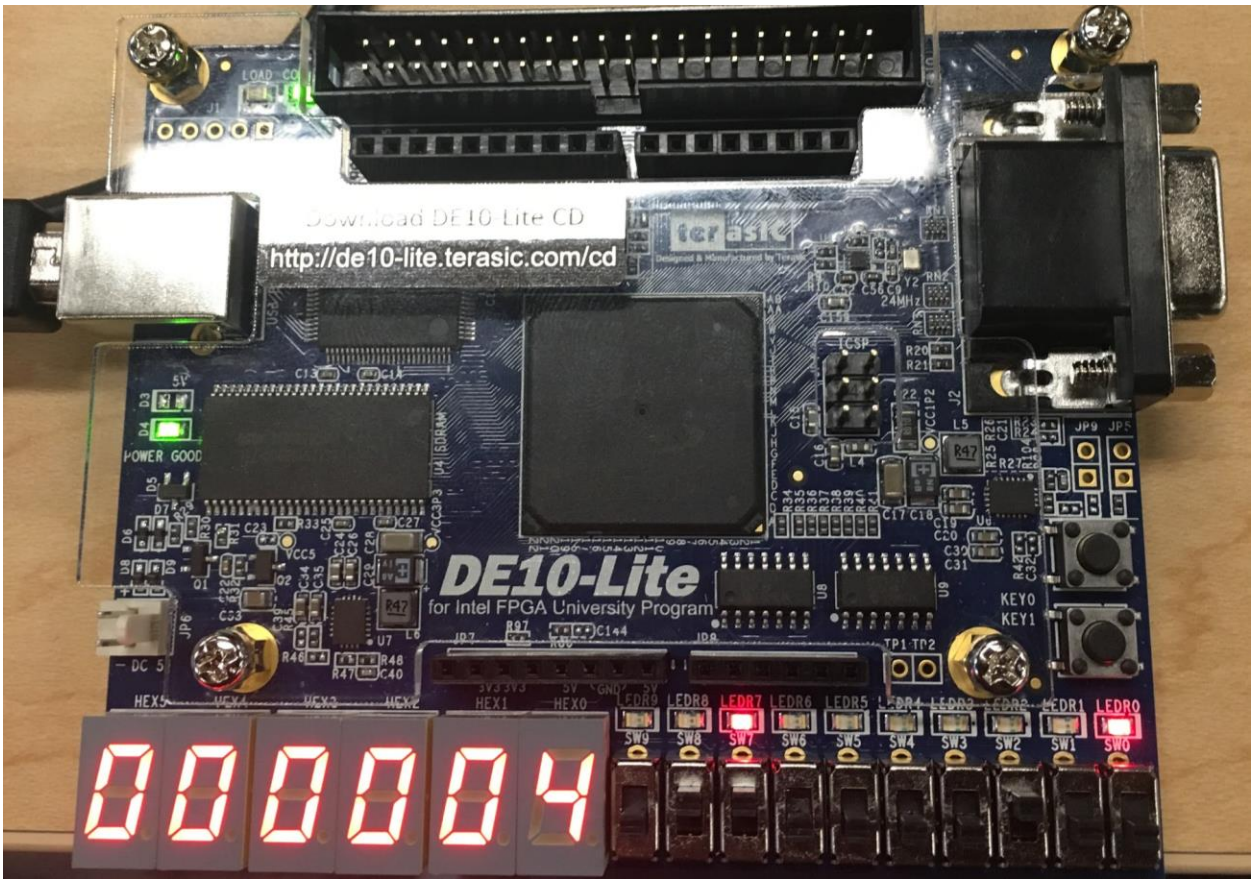
Overflow occurs so seven segment displays 0 and LED0 lights up to indicate overflow has occurred

Arithmetic Mode-Subtraction:



Inputs:

- Button 1 = $(0)_2$
- Button 2 = $(1)_2$
- Switches 0-3 = $(0010)_2$ = $(2)_{10}$
- Switches 4-7 = $(0100)_2$ = $(4)_{10}$
- Switch 9 = $(0)_2$
- Switch 8 = $(1)_2$

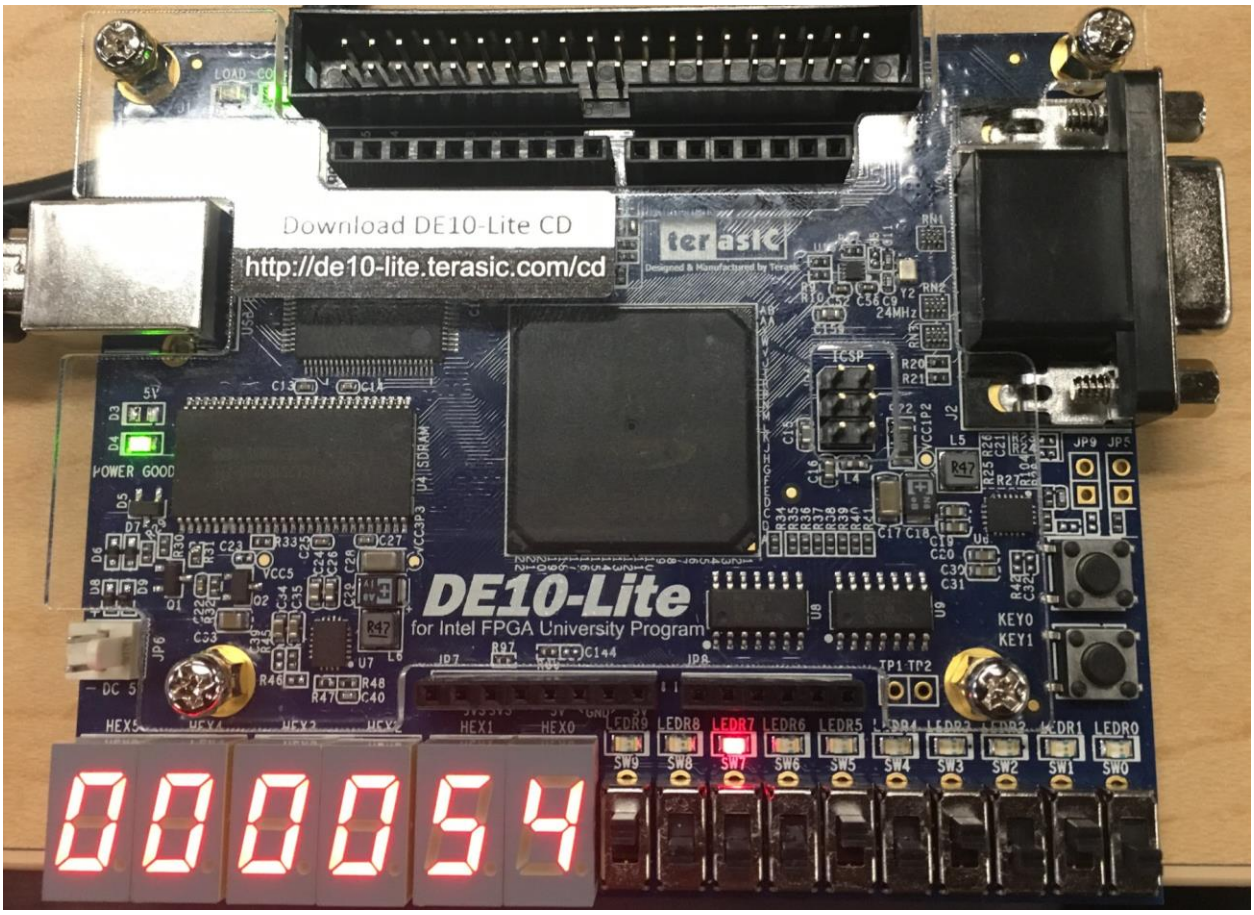


Inputs:

- Button 1 = $(0)_2$
- Button 2 = $(1)_2$
- Switches 0-3 = $(0100)_2$ = $(4)_{10}$
- Switches 4-7 = $(1000)_2$ = $(8)_{10}$
- Switch 9 = $(0)_2$
- Switch 8 = $(1)_2$

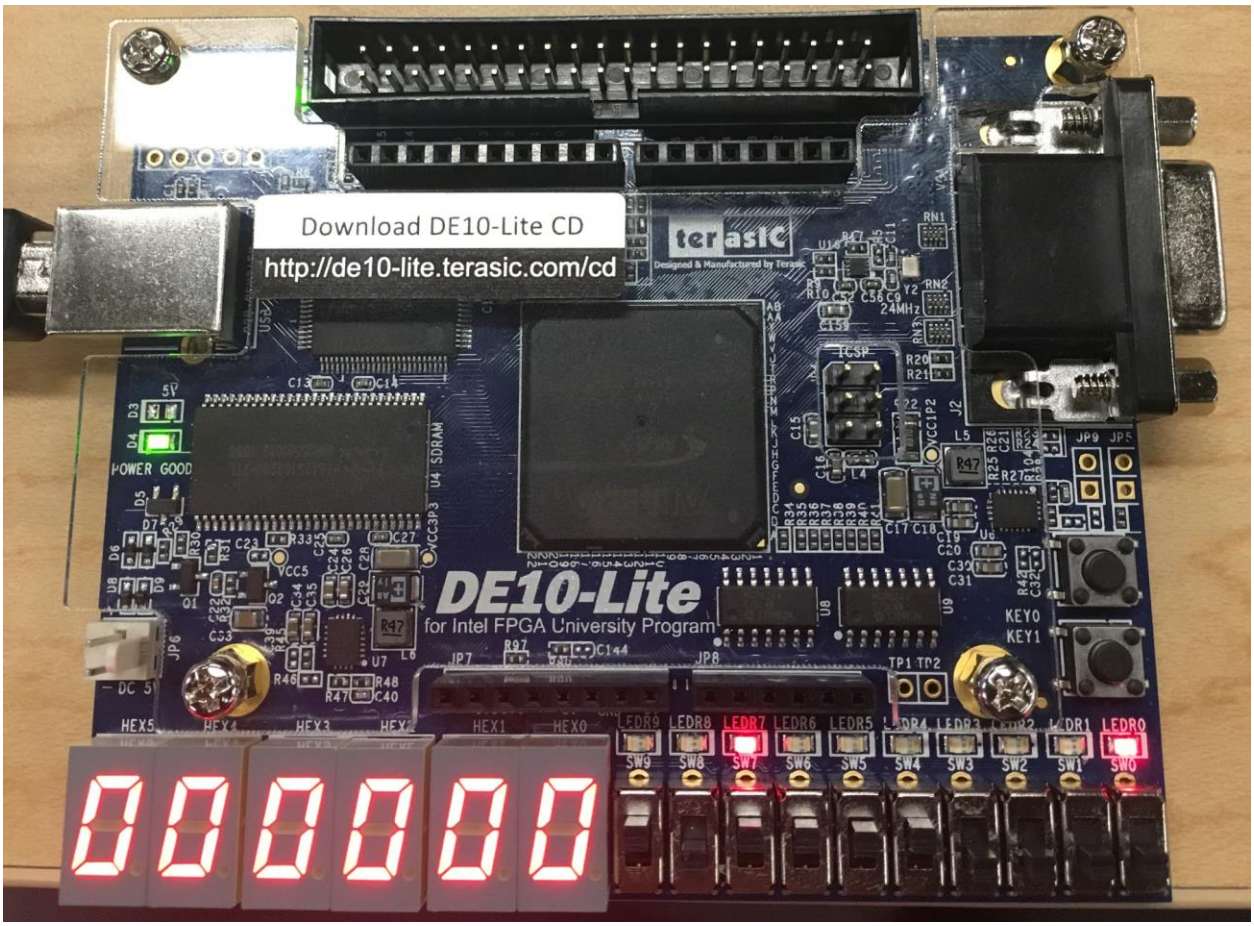
Overflow occurs so seven segment displays 0 and LED0 lights up to indicate overflow has occurred

Arithmetic Mode-Multiplication Operation:



Inputs:

- Button 1 = $(0)_2$
- Button 2 = $(1)_2$
- Switches 7-4 = $(00101010)_2 = (42)_{10}$
- Switch 9 = $(1)_2$
- Switch 8 = $(0)_2$

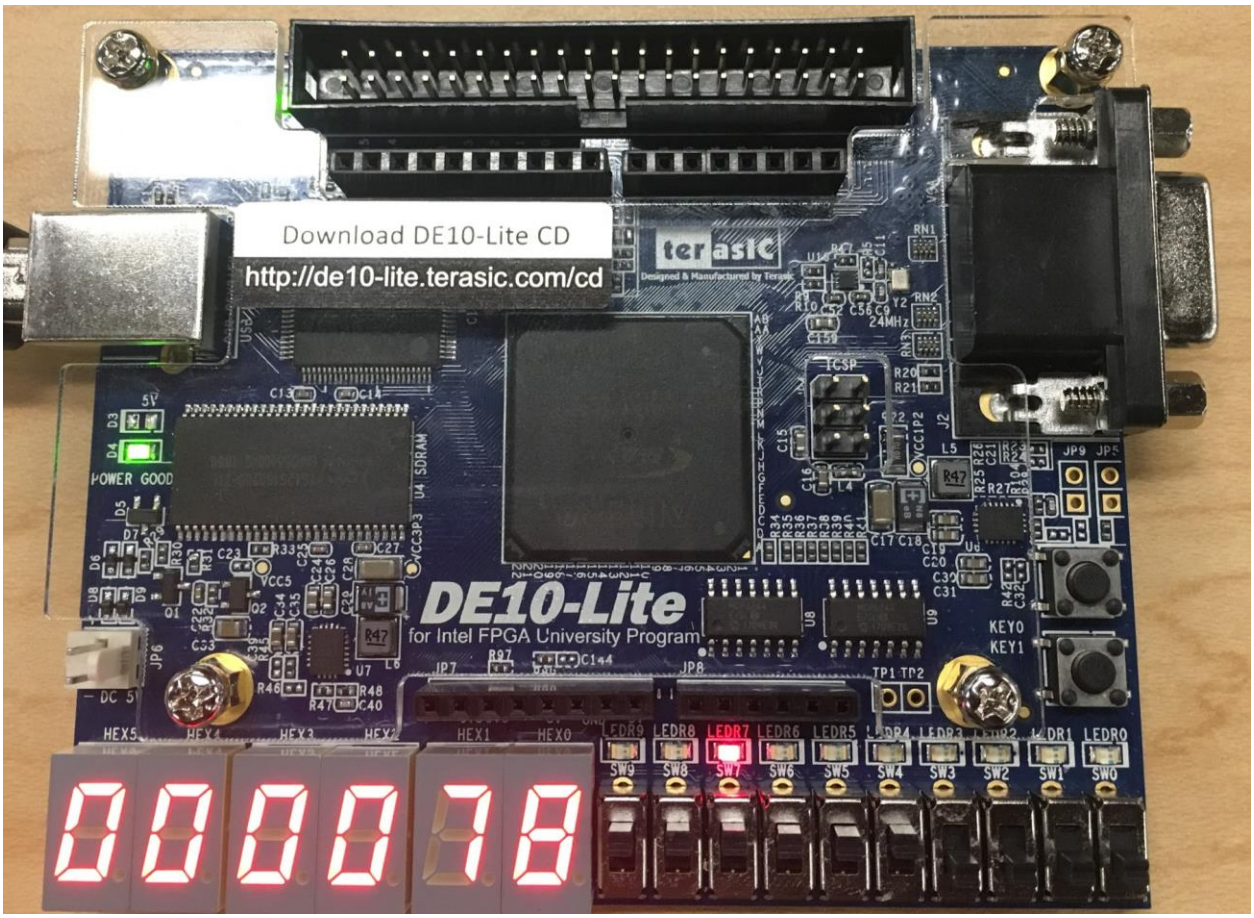


Inputs:

- Button 1 = $(0)_2$
- Button 2 = $(1)_2$
- Switches 7-0 = $(11110000)_2 = (240)_{10}$
- Switch 9 = $(1)_2$
- Switch 8 = $(0)_2$

Overflow occurs so seven segment displays 0 and LED0 lights up to indicate overflow has occurred

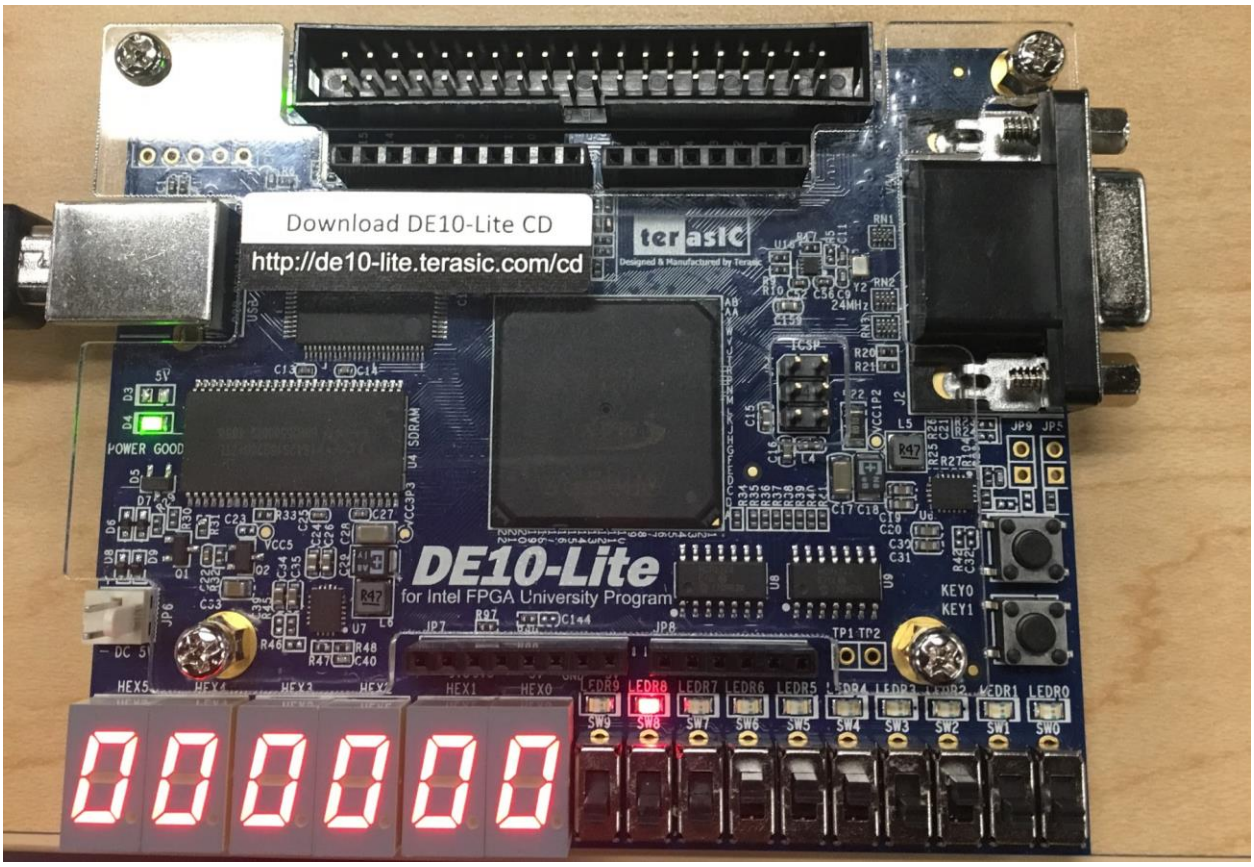
Arithmetic Mode-Division Operation:



Inputs:

- Button 1 = $(0)_2$
- Button 2 = $(1)_2$
- Switches 0-3 = $(11110000)_2 = (240)_{10}$
- Switch 9 = $(1)_2$
- Switch 8 = $(1)_2$

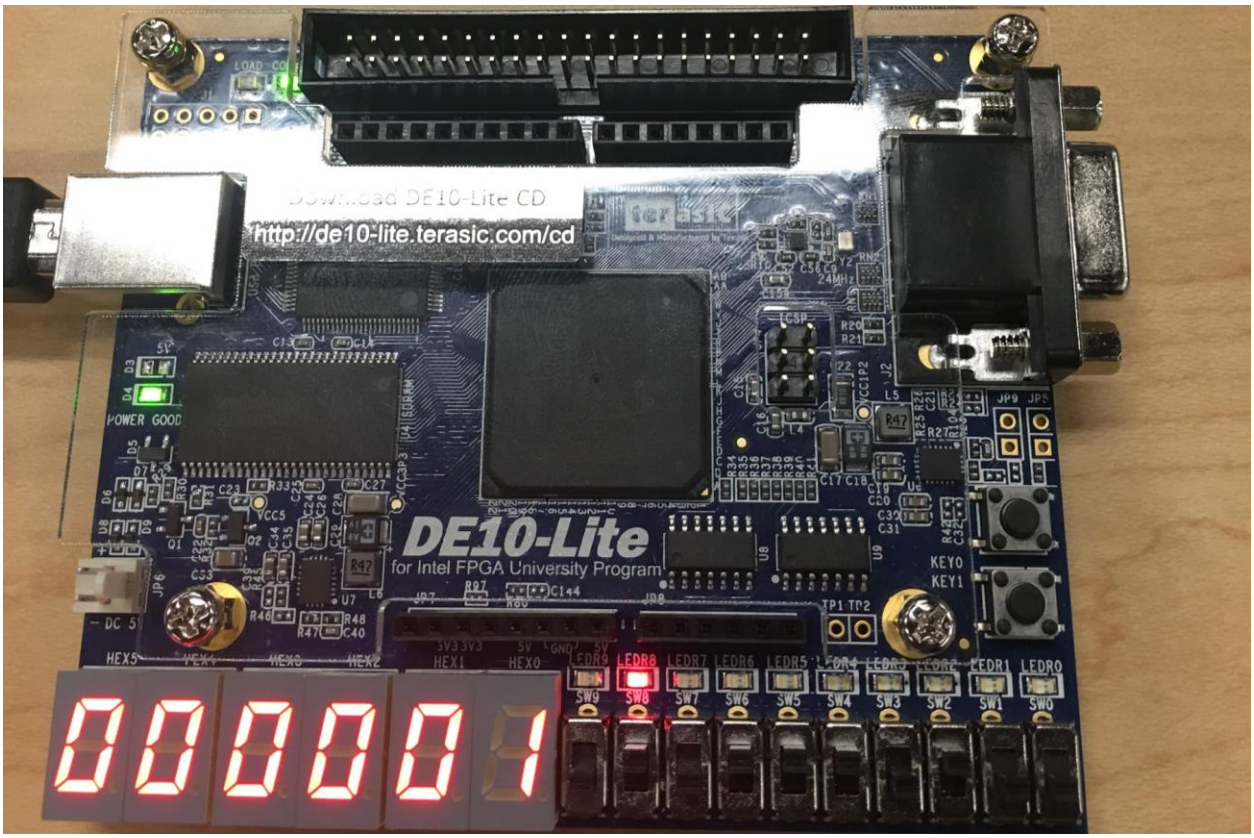
Comparison Mode-Equal Operation:



Inputs:

- Button 1 = $(1)_2$
- Button 2 = $(0)_2$
- Switches 0-3 = $(0100)_2$ = $(4)_{10}$
- Switches 4-7 = $(0111)_2$ = $(7)_{10}$
- Switch 9 = $(0)_2$
- Switch 8 = $(0)_2$

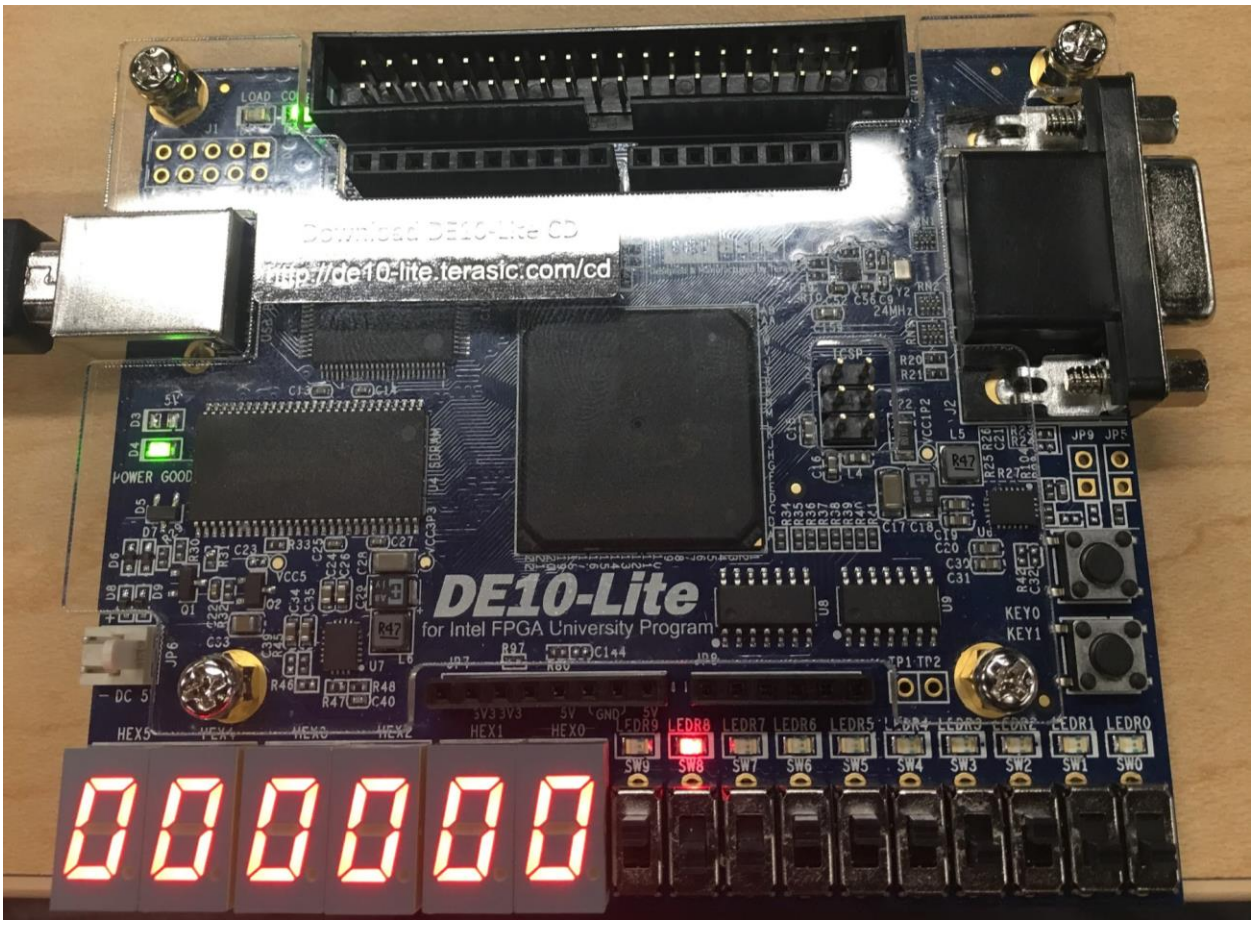
Comparison Operation-Greater Mode:



Inputs:

- Button 1 = $(1)_2$
- Button 2 = $(0)_2$
- Switches 0-3 = $(0100)_2$ = $(4)_{10}$
- Switches 4-7 = $(0111)_2$ = $(7)_{10}$
- Switch 9 = $(0)_2$
- Switch 8 = $(1)_2$

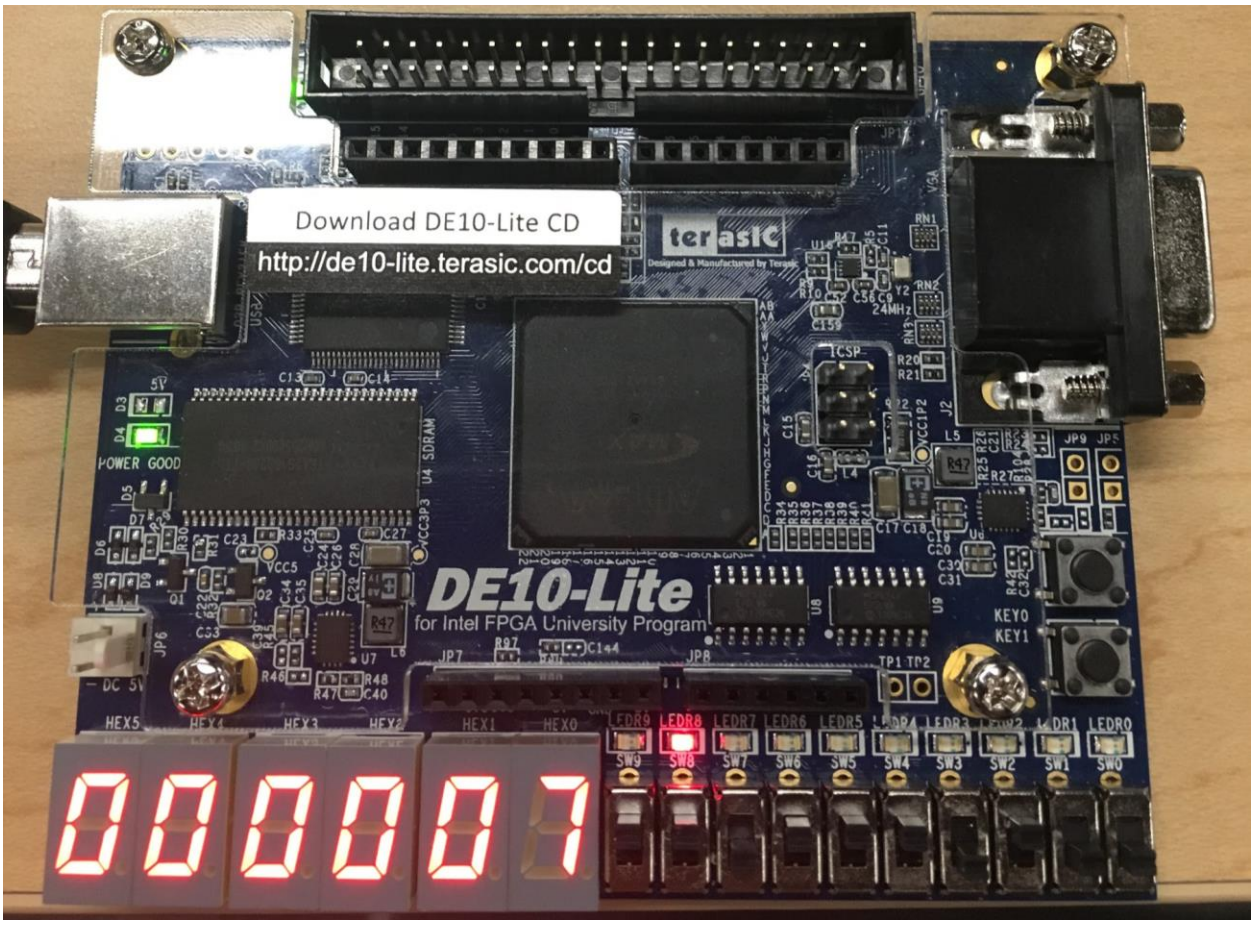
Comparison Mode-Less Operator:



Inputs:

- Button 1 = $(1)_2$
- Button 2 = $(0)_2$
- Switches 0-3 = $(0100)_2$ = $(4)_{10}$
- Switches 4-7 = $(0111)_2$ = $(7)_{10}$
- Switch 9 = $(1)_2$
- Switch 8 = $(0)_2$

Comparison Mode-Max Operator:



Inputs:

- Button 1 = $(1)_2$
- Button 2 = $(0)_2$
- Switches 0-3 = $(0100)_2$ = $(4)_{10}$
- Switches 4-7 = $(0111)_2$ = $(7)_{10}$
- Switch 9 = $(1)_2$
- Switch 8 = $(1)_2$

Conclusion:

In this project, we successfully programmed a mini-CPU capable of performing basic arithmetic as well as utilizing logical and comparison operations. We fed inputs to our program using combinations of buttons and switches on our board, and these produced various results that were output to a 7-segment display. Our code was also able to handle overflow and indicate such events by lighting an LED on the board.