Lab 3
Simon Carballo
1618309
10/30/2020
Section D

Write Up

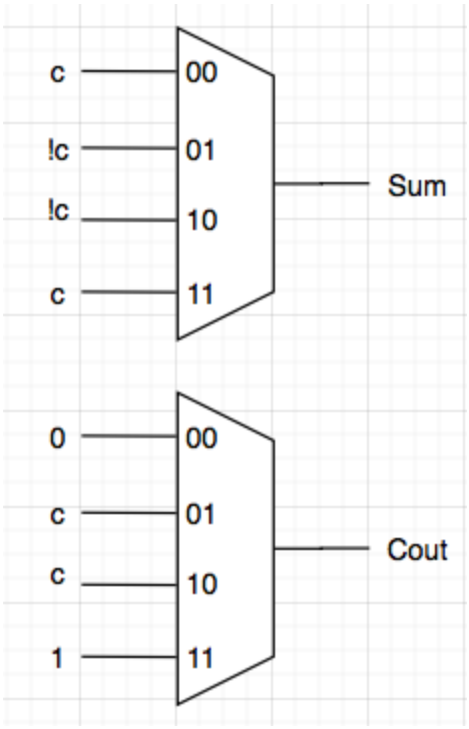
- **Description:** In this lab we were asked to design and implement a combinational circuit which adds 0-3 to an 8-bit binary number. This displays the sum on the two rightmost 7-segment LED displays on the Basys 3 Board. This will be operated using the 8 switches (sw[0] - sw[7]) and two extra buttons (btnU, btnD) that represent a 2-bit addition. The challenge of this lab is to build the adders and converters using MUX's.

- **Design:** For this design we have the Top Level that calls to 5 subsections which includes: Incrementer, Upper/Lower Segment Converter, Digital Selector, and a 2to1 MUX. As we go over each subsection we will look more into depth of how each section is designed.
    - <u>Incrementer:</u>
      First let's look at the incrementer that acts as the 8-bit adder in this design:
        - Truth Table of a Full Adder:

a	b	c	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

        - As we see in the truth table we add the sum of a, b, and cin, and when there is a value over 1, there would be an overflow representing Cout.
        - Now that we are refreshed on how an Adder works. This time we will create a 4-1MUX with the truth table by designating a, and b as the selector bits(in this case). This can be seen by the color coding of each selector state that will be used in the MUX. This creates the following boolean equation with for the two outputs:
            - Sum = c!a!b + !c!ab + !ca!b + cab

    - Cout = c!ab + ca!b + ab
- Visually these MUX would look like this for the two outputs:



- By putting the two MUXs together we create a full adder!

- Now we have to make it into the 8-bit incrementer by calling the Adder 8 times and wiring them together just like how we would wire adders together. In this design we also need to be aware that there are two extra buttons (btnU, btnD) that act as a 2-bit adder which would be added into the rest of the 8-bit address. In order to do this we set the first two adders as the second input while the rest would be set to 0. This would create the incrementer design that would be implemented into the TopLevel Design.

- <u>Upper/Lower Segment Converter:</u>
  Now we will take a look at the seven segment converter for this design
    - Like the adder we will by using MUXs to design our seven segment converter. In this design we will be using a 8-1 MUX to find the outputs for each segment.
    - First let's take a look at the truth table:

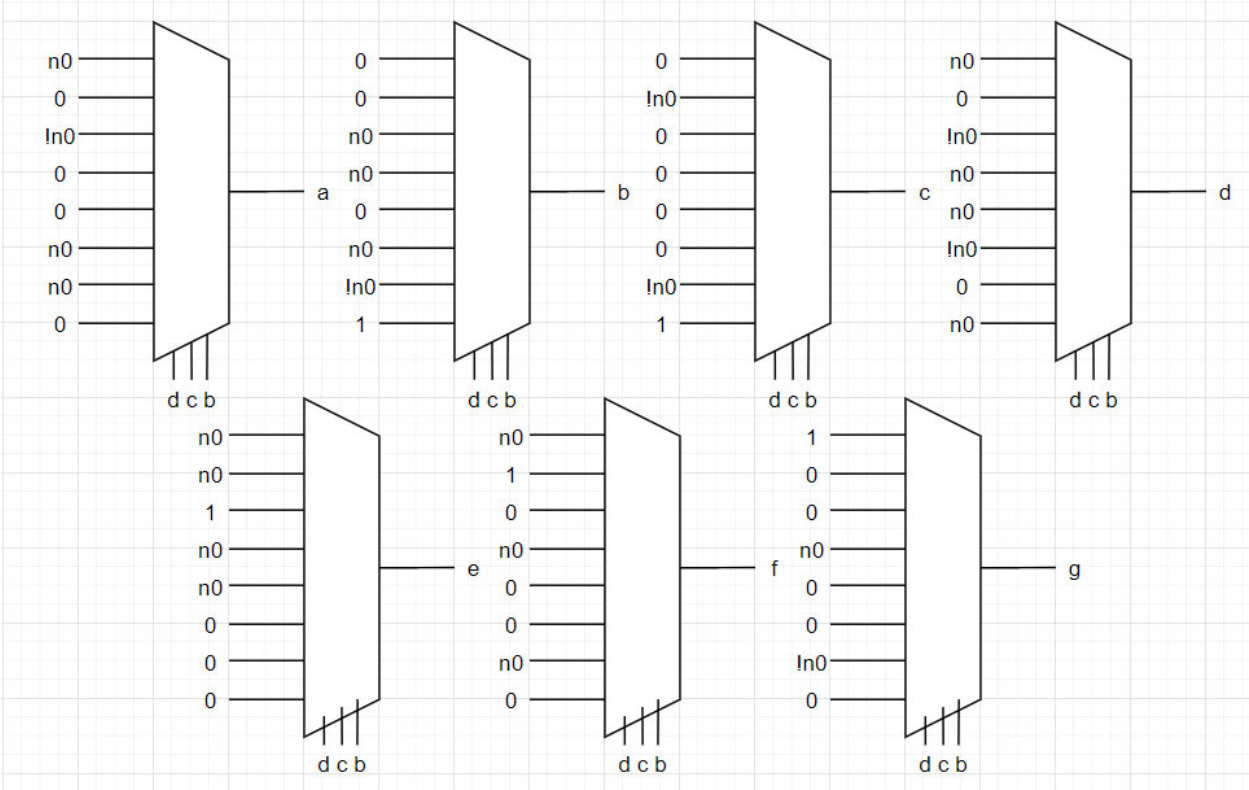
n3	n2	n1	n0	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1

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

- This is a 4-variable truth table for the 16hex values available on a seven segment display. a - g represents each segment of the Cathode Segment Display. There is a DP segment but that will be unused for this design. In order to figure out active high and lows of each segment we go through each hex value and determine whether the segment will be on or off. For example, when we want the value “1” we would need to have the segments ‘b’ and ‘c’ on therefore making them the lows(0s) for a cathode diode.
- Now that we refresh on how a converter works, we will be using 7 8-1 MUXs to create our converter. In order to do this we will use n1, n2, and n3 as the selector bits shown by the color coding in the truth table. By finding the highs and lows of n0 at each selector state we can design the converter we get these boolean expressions for each 8to1 MUX.
    - Selector : n1, n2, n3
    - a = n0(!n1!n2!n3) + !n1(!n1n2!n3) + n0(n1!n2n3) + n0(!n1n2n3)
    - b = n0(!n1n2!n3) + !n0(n1n2!n3) + n0(n1!n2n3) + !n0(!n1n2n3) + (n1n2n3)
    - c = !n0(n1!n2!n3) + !n0(!n1n2n3) + (n1n2n3)

- d = n0(!n1!n2!n3) + !n0(!n1n2!n3) + n0(n1n2!n3) + n0(!n1!n2n3) + !n0(n1!n2n3) + n0(n1n2n3)
- e = n0(!n1!n2!n3) + n0(n1!n2!n3) + (n1n2n3) + n0(n1n2!n3) + n0(n1!n2n3)
- f = n0(!n1!n2!n3) + (n1n2n3) + n0(n1n2!n3) + n0(!n1n2n3)
- g = (!n1!n2!n3) + n0(n1n2!n3) + !n0(!n1n2n3)

- By following the boolean expression we obtain a 8to1 MUX that looks like this:



- Just like a regular converter, when called by the top level each input will go through the 7 MUXs for their respective segments. Therefore in order to have two seven segment displays to work, we would need two groups of 8to1 MUXs for the lower and upper values of the 8-bit adder.

- <u>Digsel</u>:
  - This module/subsection is provided to us via the lab manual. It's primary function is to output a low frequency of highs and lows for the two seven segment displays to work.

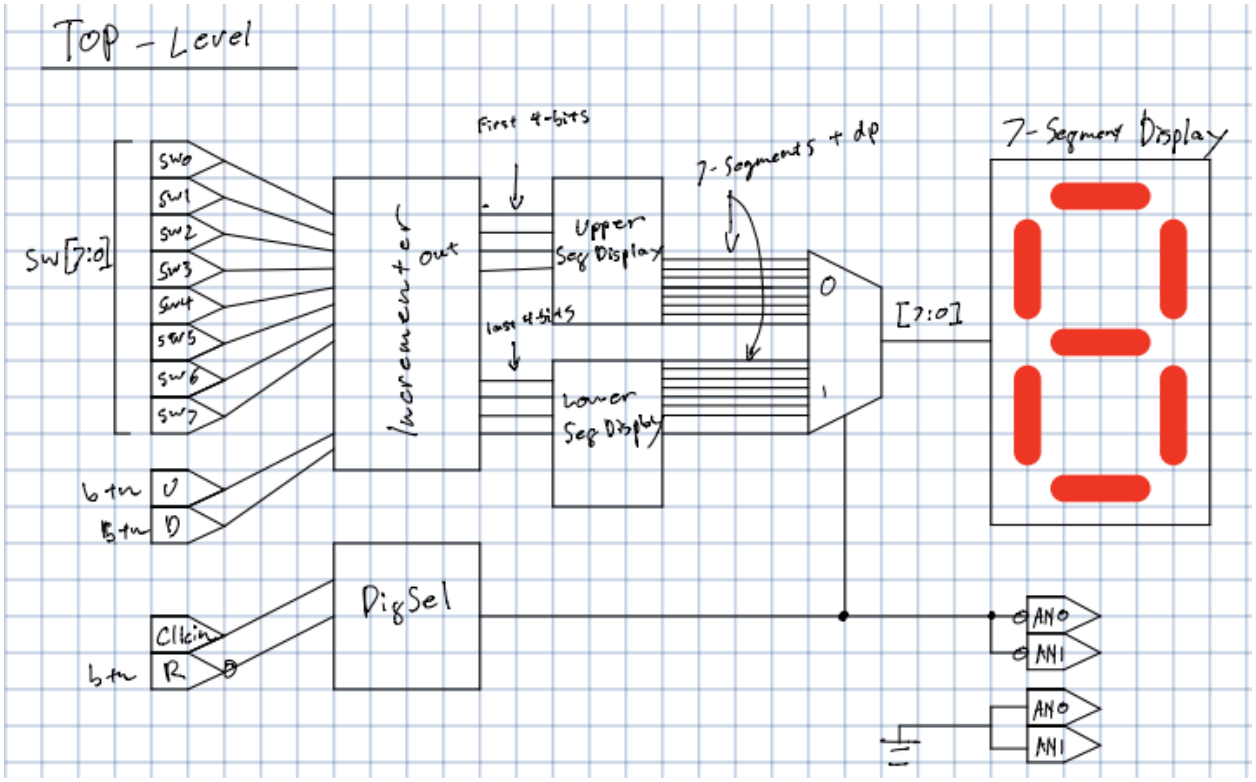
- <u>2to1 MUX:</u>

- Finally the final piece is the MUX that takes inputs from the two converters we designed above to be selected by the Digsel for display on the seven segment converter. We need to make sure that 8-bits gets through the MUX. To summarize, we need this MUX to be able to display two seven segments at the same time in human eyes that would work in unison with the 8-bit adder or incrementer.

- Top Level:
-

- With this truth table we can acquire a boolean expression for each segment to make the 16Hex values (n0,n1, n2, n3 == a, b, c, d):
  - a = a!b!c!d + !abc!d + ab!cd + a!bcd
  - b = a!bc!d + !abc!d + ab!cd + !a!bcd + !abcd + abcd
  - c = !ab!c!d + !a!bcd + !abcd + abcd
  - d = a!b!c!d + !a!bc!d + abc!d + a!b!cd + !ab!cd + abcd
  - e = a!b!c!d + ab!c!d + !a!bc!d + a!bc!d + abc!d + a!b!cd
  - f = a!b!c!d + !ab!c!d + ab!c!d + abc!d + a!bcd
  - g = !a!b!c!d + a!b!c!d + abc!d + !a!bcd
- We can input this boolean expression into the converter by taking in the 4 outputs and wiring each output to its respective segment. We will do this in the Top_Level Diagram as shown below.

- Top-Level:



- As you can see in the diagram above these are all the subsections put together to create the design. The 8 inputs are directed into the increment which outputs a summation of 8bits. The first 4-bits are connected to the lower segment converter and the last 4-bits are connected to the upper segment converter. We connect these are the two inputs for the 2x1 MUX with the selector being Digsel operated with a reset and internal clock. This MUX will then frequently output the two inputs of the MUX to two separate seven segment displays as shown with the wire to AN0 and AN1.

- **Testing & Simulation:**
  In order to test my design I used two different resources like the previous lab. The first resource is by running a simulation directly from the Vivado application. Using the provided testbench file, I entered the test cases spaced between each 100ns. These tests consisted of displaying the 16HEX values on both displays, mismatching them, and finally overloading them with the buttons that add 2-bit values. Another source was by programming the bitstream to the Basys3 board. This method is a more efficient method of testing since I can visually see if I wired the wrong switches or swapped the buttons.

- **Results:**
  - How fast the signal **dig_sel** is oscillating (in simulation).

        - I cannot find the exact number that the dig_sel is oscillating at, but I know for a fact that the oscillations can only be detected in simulation when observing microseconds.
    - Whether you observed any flickering in the 7-segment display
        - I did not see any flickering on the 7-segment display, this is because the digsel is oscillating so fast that a human eye can not detect the on and off of the seven segment display.

- **Conclusion**
    - From this lab I learned how to implement MUXs to the previous modules we have created (Adder, Seven Segment Converter). By using this previous knowledge I was able to create a 8-bit adder and separate the outputs into two different displays. I also learned how to use busses to keep wiring tidy in a bigger design. The most interesting part of this design is how the digsel oscillation works to provide power for two displays using a MUX. It is definitely very fascinating to see how fast an oscillation is, and undetectable to the human eye.

```
## This file is a general .xdc for the Basys3
rev B board
## To use it in a project:
## - uncomment the lines corresponding to used
pins
## - rename the used ports (in each line,
after get_ports) according to the top level
signal names in the project

## Clock signal
set_property PACKAGE_PIN W5 [get_ports clkin]

    set_property IOSTANDARD LVCMOS33 [get_ports
clkin]
    create_clock -add -name sys_clk_pin -period
10.00 -waveform {0 5} [get_ports clkin]

## Switches
set_property PACKAGE_PIN V17 [get_ports
{sw[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{sw[0]}]
set_property PACKAGE_PIN V16 [get_ports
{sw[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{sw[1]}]
set_property PACKAGE_PIN W16 [get_ports
```

```
{sw[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{sw[2]}]
set_property PACKAGE_PIN W17 [get_ports
{sw[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{sw[3]}]
set_property PACKAGE_PIN W15 [get_ports
{sw[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{sw[4]}]
set_property PACKAGE_PIN V15 [get_ports
{sw[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{sw[5]}]
set_property PACKAGE_PIN W14 [get_ports
{sw[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{sw[6]}]
set_property PACKAGE_PIN W13 [get_ports
{sw[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{sw[7]}]
#set_property PACKAGE_PIN V2 [get_ports
{sw[8]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {sw[8]}]
```

```
#set_property PACKAGE_PIN T3 [get_ports
{sw[9]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {sw[9]}]
#set_property PACKAGE_PIN T2 [get_ports
{sw[10]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {sw[10]}]
#set_property PACKAGE_PIN R3 [get_ports
{sw[11]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {sw[11]}]
#set_property PACKAGE_PIN W2 [get_ports
{sw[12]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports
{sw[13]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports
{sw[14]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {sw[14]}]
#set_property PACKAGE_PIN R2 [get_ports
{sw[15]}]
    #set_property IOSTANDARD LVCMOS33
```

```
[get_ports {sw[15]}]


## LEDs
#set_property PACKAGE_PIN U16 [get_ports
{led[0]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[0]}]
#set_property PACKAGE_PIN E19 [get_ports
{led[1]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[1]}]
#set_property PACKAGE_PIN U19 [get_ports
{led[2]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[2]}]
#set_property PACKAGE_PIN V19 [get_ports
{led[3]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[3]}]
#set_property PACKAGE_PIN W18 [get_ports
{led[4]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[4]}]
#set_property PACKAGE_PIN U15 [get_ports
{led[5]}]
    #set_property IOSTANDARD LVCMOS33
```

```
[get_ports {led[5]}]
#set_property PACKAGE_PIN U14 [get_ports
{led[6]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[6]}]
#set_property PACKAGE_PIN V14 [get_ports
{led[7]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[7]}]
#set_property PACKAGE_PIN V13 [get_ports
{led[8]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[8]}]
#set_property PACKAGE_PIN V3 [get_ports
{led[9]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[9]}]
#set_property PACKAGE_PIN W3 [get_ports
{led[10]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[10]}]
#set_property PACKAGE_PIN U3 [get_ports
{led[11]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[11]}]
#set_property PACKAGE_PIN P3 [get_ports
{led[12]}]
```

```
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[12]}]
#set_property PACKAGE_PIN N3 [get_ports
{led[13]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[13]}]
#set_property PACKAGE_PIN P1 [get_ports
{led[14]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[14]}]
#set_property PACKAGE_PIN L1 [get_ports
{led[15]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {led[15]}]


##7 segment display
set_property PACKAGE_PIN W7 [get_ports
{seg[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{seg[0]}]
set_property PACKAGE_PIN W6 [get_ports
{seg[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{seg[1]}]
set_property PACKAGE_PIN U8 [get_ports
{seg[2]}]
```

```
    set_property IOSTANDARD LVCMOS33 [get_ports
{seg[2]}]
set_property PACKAGE_PIN V8 [get_ports
{seg[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{seg[3]}]
set_property PACKAGE_PIN U5 [get_ports
{seg[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{seg[4]}]
set_property PACKAGE_PIN V5 [get_ports
{seg[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{seg[5]}]
set_property PACKAGE_PIN U7 [get_ports
{seg[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{seg[6]}]

set_property PACKAGE_PIN V7 [get_ports dp]

    set_property IOSTANDARD LVCMOS33 [get_ports
dp]

set_property PACKAGE_PIN U2 [get_ports
{an[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
```

```
{an[0]}]
set_property PACKAGE_PIN U4 [get_ports
{an[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{an[1]}]
set_property PACKAGE_PIN V4 [get_ports
{an[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{an[2]}]
set_property PACKAGE_PIN W4 [get_ports
{an[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports
{an[3]}]


##Buttons
#set_property PACKAGE_PIN U18 [get_ports btnC]

    #set_property IOSTANDARD LVCMOS33
[get_ports btnC]
set_property PACKAGE_PIN T18 [get_ports btnU]

    set_property IOSTANDARD LVCMOS33 [get_ports
btnU]
#set_property PACKAGE_PIN W19 [get_ports btnL]

    #set_property IOSTANDARD LVCMOS33
```

```
[get_ports btnL]
set_property PACKAGE_PIN T17 [get_ports btnR]

    set_property IOSTANDARD LVCMOS33 [get_ports
btnR]
set_property PACKAGE_PIN U17 [get_ports btnD]

    set_property IOSTANDARD LVCMOS33 [get_ports
btnD]



##Pmod Header JA
##Sch name = JA1
#set_property PACKAGE_PIN J1 [get_ports
{JA[0]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JA[0]}]
##Sch name = JA2
#set_property PACKAGE_PIN L2 [get_ports
{JA[1]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JA[1]}]
##Sch name = JA3
#set_property PACKAGE_PIN J2 [get_ports
{JA[2]}]
    #set_property IOSTANDARD LVCMOS33
```

```
[get_ports {JA[2]}]
##Sch name = JA4
#set_property PACKAGE_PIN G2 [get_ports
{JA[3]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JA[3]}]
##Sch name = JA7
#set_property PACKAGE_PIN H1 [get_ports
{JA[4]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JA[4]}]
##Sch name = JA8
#set_property PACKAGE_PIN K2 [get_ports
{JA[5]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JA[5]}]
##Sch name = JA9
#set_property PACKAGE_PIN H2 [get_ports
{JA[6]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JA[6]}]
##Sch name = JA10
#set_property PACKAGE_PIN G3 [get_ports
{JA[7]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JA[7]}]
```

```
##Pmod Header JB
##Sch name = JB1
#set_property PACKAGE_PIN A14 [get_ports
{JB[0]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JB[0]}]
##Sch name = JB2
#set_property PACKAGE_PIN A16 [get_ports
{JB[1]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JB[1]}]
##Sch name = JB3
#set_property PACKAGE_PIN B15 [get_ports
{JB[2]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JB[2]}]
##Sch name = JB4
#set_property PACKAGE_PIN B16 [get_ports
{JB[3]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JB[3]}]
##Sch name = JB7
#set_property PACKAGE_PIN A15 [get_ports
{JB[4]}]
    #set_property IOSTANDARD LVCMOS33
```

```
[get_ports {JB[4]}]
##Sch name = JB8
#set_property PACKAGE_PIN A17 [get_ports
{JB[5]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JB[5]}]
##Sch name = JB9
#set_property PACKAGE_PIN C15 [get_ports
{JB[6]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JB[6]}]
##Sch name = JB10
#set_property PACKAGE_PIN C16 [get_ports
{JB[7]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JB[7]}]



##Pmod Header JC
##Sch name = JC1
#set_property PACKAGE_PIN K17 [get_ports
{JC[0]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JC[0]}]
##Sch name = JC2
#set_property PACKAGE_PIN M18 [get_ports
```

```
{JC[1]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JC[1]}]
##Sch name = JC3
#set_property PACKAGE_PIN N17 [get_ports
{JC[2]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JC[2]}]
##Sch name = JC4
#set_property PACKAGE_PIN P18 [get_ports
{JC[3]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JC[3]}]
##Sch name = JC7
#set_property PACKAGE_PIN L17 [get_ports
{JC[4]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JC[4]}]
##Sch name = JC8
#set_property PACKAGE_PIN M19 [get_ports
{JC[5]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JC[5]}]
##Sch name = JC9
#set_property PACKAGE_PIN P17 [get_ports
{JC[6]}]
    #set_property IOSTANDARD LVCMOS33
```

```
[get_ports {JC[6]}]
##Sch name = JC10
#set_property PACKAGE_PIN R18 [get_ports
{JC[7]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JC[7]}]


##Pmod Header JXADC
##Sch name = XA1_P
#set_property PACKAGE_PIN J3 [get_ports
{JXADC[0]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JXADC[0]}]
##Sch name = XA2_P
#set_property PACKAGE_PIN L3 [get_ports
{JXADC[1]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JXADC[1]}]
##Sch name = XA3_P
#set_property PACKAGE_PIN M2 [get_ports
{JXADC[2]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JXADC[2]}]
##Sch name = XA4_P
#set_property PACKAGE_PIN N2 [get_ports
{JXADC[3]}]
```

```
    #set_property IOSTANDARD LVCMOS33
[get_ports {JXADC[3]}]
##Sch name = XA1_N
#set_property PACKAGE_PIN K3 [get_ports
{JXADC[4]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JXADC[4]}]
##Sch name = XA2_N
#set_property PACKAGE_PIN M3 [get_ports
{JXADC[5]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JXADC[5]}]
##Sch name = XA3_N
#set_property PACKAGE_PIN M1 [get_ports
{JXADC[6]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JXADC[6]}]
##Sch name = XA4_N
#set_property PACKAGE_PIN N1 [get_ports
{JXADC[7]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {JXADC[7]}]



##VGA Connector
#set_property PACKAGE_PIN G19 [get_ports
```

```
{vgaRed[0]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {vgaRed[0]}]
#set_property PACKAGE_PIN H19 [get_ports
{vgaRed[1]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {vgaRed[1]}]
#set_property PACKAGE_PIN J19 [get_ports
{vgaRed[2]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {vgaRed[2]}]
#set_property PACKAGE_PIN N19 [get_ports
{vgaRed[3]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {vgaRed[3]}]
#set_property PACKAGE_PIN N18 [get_ports
{vgaBlue[0]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {vgaBlue[0]}]
#set_property PACKAGE_PIN L18 [get_ports
{vgaBlue[1]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {vgaBlue[1]}]
#set_property PACKAGE_PIN K18 [get_ports
{vgaBlue[2]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {vgaBlue[2]}]
```

```
#set_property PACKAGE_PIN J18 [get_ports
{vgaBlue[3]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {vgaBlue[3]}]
#set_property PACKAGE_PIN J17 [get_ports
{vgaGreen[0]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {vgaGreen[0]}]
#set_property PACKAGE_PIN H17 [get_ports
{vgaGreen[1]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {vgaGreen[1]}]
#set_property PACKAGE_PIN G17 [get_ports
{vgaGreen[2]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {vgaGreen[2]}]
#set_property PACKAGE_PIN D17 [get_ports
{vgaGreen[3]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {vgaGreen[3]}]
#set_property PACKAGE_PIN P19 [get_ports
Hsync]
    #set_property IOSTANDARD LVCMOS33
[get_ports Hsync]
#set_property PACKAGE_PIN R19 [get_ports
Vsync]
    #set_property IOSTANDARD LVCMOS33
```

```
[get_ports Vsync]


##USB-RS232 Interface
#set_property PACKAGE_PIN B18 [get_ports RsRx]

    #set_property IOSTANDARD LVCMOS33
[get_ports RsRx]
#set_property PACKAGE_PIN A18 [get_ports RsTx]

    #set_property IOSTANDARD LVCMOS33
[get_ports RsTx]


##USB HID (PS/2)
#set_property PACKAGE_PIN C17 [get_ports
PS2Clk]
    #set_property IOSTANDARD LVCMOS33
[get_ports PS2Clk]
    #set_property PULLUP true [get_ports
PS2Clk]
#set_property PACKAGE_PIN B17 [get_ports
PS2Data]
    #set_property IOSTANDARD LVCMOS33
[get_ports PS2Data]
    #set_property PULLUP true [get_ports
PS2Data]
```

```
##Quad SPI Flash
##Note that CCLK_0 cannot be placed in 7
series devices. You can access it using the
##STARTUPE2 primitive.
#set_property PACKAGE_PIN D18 [get_ports
{QspiDB[0]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {QspiDB[0]}]
#set_property PACKAGE_PIN D19 [get_ports
{QspiDB[1]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {QspiDB[1]}]
#set_property PACKAGE_PIN G18 [get_ports
{QspiDB[2]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {QspiDB[2]}]
#set_property PACKAGE_PIN F18 [get_ports
{QspiDB[3]}]
    #set_property IOSTANDARD LVCMOS33
[get_ports {QspiDB[3]}]
#set_property PACKAGE_PIN K19 [get_ports
QspiCSn]
    #set_property IOSTANDARD LVCMOS33
[get_ports QspiCSn]
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/22/2020 01:19:30 AM
// Design Name:
// Module Name: MUX4_1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////


module MUX4_1(
    input [3:0] in,
    input [1:0] sel,
```

```
    output out
    );

    assign out = (in[0] & ~sel[1] &
~sel[0])|(in[1] & ~sel[1] & sel[0])|(in[2] &
sel[1] & ~sel[0])|(in[3] & sel[1] & sel[0]);


endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/22/2020 01:27:56 AM
// Design Name:
// Module Name: MUX8_1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module MUX8_1(
    input [7:0] in,
    input [3:0] sel,
```

```
    output out
    );

    assign out = (in[0] & ~sel[2] & ~sel[1] &
~sel[0])|(in[1] & ~sel[2] & ~sel[1] &
sel[0])|(in[2] & ~sel[2] & sel[1] &
~sel[0])|(in[3] & ~sel[2] & sel[1] &
sel[0])|(in[4] & sel[2] & ~sel[1] &
~sel[0])|(in[5] & sel[2] & ~sel[1] &
sel[0])|(in[6] & sel[2] & sel[1] &
~sel[0])|(in[7] & sel[2] & sel[1] & sel[0]);

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/22/2020 01:52:49 AM
// Design Name:
// Module Name: MUX2_1x8
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////


module MUX2_1x8(
    input [7:0] in0,
    input [7:0] in1,
```

```
    input sel,
    output [7:0] out
    );

    assign out = (in0 & {8{~sel}})|(in1[7:0] &
{8{sel}});

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/22/2020 03:43:47 PM
// Design Name:
// Module Name: Full_Adder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////


module Full_Adder(
    output sum, cout,
    input a, b, cin
```

```
    );

    MUX4_1 MUX_cout(.in({1'b1, cin, cin,
1'b0}), .sel({a, b}), .out(cout));
    MUX4_1 MUX_sum(.in({cin, ~cin, ~cin,
cin}), .sel({a, b}), .out(sum));

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/22/2020 01:36:14 AM
// Design Name:
// Module Name: Incrementer
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Incrementer(
    input Cin,
    input [7:0] a,
```

```
    input [1:0] b,
    output [7:0] s
    );

    wire [7:0] c;

    Full_Adder Adder1(.a(a[0]), .b(b[0]),
.cin(Cin), .sum(s[0]), .cout(c[0]));
    Full_Adder Adder2(.a(a[1]), .b(b[1]),
.cin(c[0]), .sum(s[1]), .cout(c[1]));
    Full_Adder Adder3(.a(a[2]), .b(1'b0),
.cin(c[1]), .sum(s[2]), .cout(c[2]));
    Full_Adder Adder4(.a(a[3]), .b(1'b0),
.cin(c[2]), .sum(s[3]), .cout(c[3]));
    Full_Adder Adder5(.a(a[4]), .b(1'b0),
.cin(c[3]), .sum(s[4]), .cout(c[4]));
    Full_Adder Adder6(.a(a[5]), .b(1'b0),
.cin(c[4]), .sum(s[5]), .cout(c[5]));
    Full_Adder Adder7(.a(a[6]), .b(1'b0),
.cin(c[5]), .sum(s[6]), .cout(c[6]));
    Full_Adder Adder8(.a(a[7]), .b(1'b0),
.cin(c[6]), .sum(s[7]), .cout());


endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/22/2020 02:40:39 PM
// Design Name:
// Module Name: Segment_Display
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////


module Segment_Display(
    input [3:0] n,
    output [6:0] sego
```

```
    );

    MUX8_1 SegA(.in({1'b0, n[0], n[0], 1'b0,
1'b0, ~n[0], 1'b0, n[0]}), .sel(n[3:1]),
.out(sego[0]));
    MUX8_1 SegB(.in({1'b1, ~n[0], n[0], 1'b0,
~n[0], n[0], 1'b0, 1'b0}), .sel(n[3:1]),
.out(sego[1]));
    MUX8_1 SegC(.in({1'b1, ~n[0], 1'b0, 1'b0,
1'b0, 1'b0, ~n[0], 1'b0}), .sel(n[3:1]),
.out(sego[2]));
    MUX8_1 SegD(.in({n[0], 1'b0, ~n[0], n[0],
n[0], ~n[0], 1'b0, n[0]}), .sel(n[3:1]),
.out(sego[3]));
    MUX8_1 SegE(.in({1'b0, 1'b0, 1'b0, n[0],
n[0], 1'b1, n[0], n[0]}), .sel(n[3:1]),
.out(sego[4]));
    MUX8_1 SegF(.in({1'b0, n[0], 1'b0, 1'b0,
n[0], 1'b0, 1'b1, n[0]}), .sel(n[3:1]),
.out(sego[5]));
    MUX8_1 SegG(.in({1'b0, ~n[0], 1'b0, 1'b0,
n[0], 1'b0, 1'b0, 1'b1}), .sel(n[3:1]),
.out(sego[6]));

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 01/20/2017 12:17:24 PM
// Design Name:
// Module Name: lab3_digsel
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

`timescale 1ps/1ps

module clk_wiz_0
```

```
 (// Clock in ports
  // Clock out ports
  output        clk_out1,
  // Status and control signals
  input         reset,
  output        locked,
  input         clk_in1
 );
  // Input buffering
  //------------------------------------
wire clk_in1_clk_wiz_0;
wire clk_in2_clk_wiz_0;
  IBUF clkin1_ibufg
   (.O (clk_in1_clk_wiz_0),
    .I (clk_in1));


  // Clocking PRIMITIVE
  //------------------------------------

  // Instantiation of the MMCM PRIMITIVE
  //    * Unused inputs are tied off
  //    * Unused outputs are labeled unused

  wire        clk_out1_clk_wiz_0;
  wire        clk_out2_clk_wiz_0;
  wire        clk_out3_clk_wiz_0;
```

```
wire        clk_out4_clk_wiz_0;
wire        clk_out5_clk_wiz_0;
wire        clk_out6_clk_wiz_0;
wire        clk_out7_clk_wiz_0;

wire [15:0] do_unused;
wire        drdy_unused;
wire        psdone_unused;
wire        locked_int;
wire        clkfbout_clk_wiz_0;
wire        clkfbout_buf_clk_wiz_0;
wire        clkfboutb_unused;
  wire clkout0b_unused;
 wire clkout1_unused;
 wire clkout1b_unused;
 wire clkout2_unused;
 wire clkout2b_unused;
 wire clkout3_unused;
 wire clkout3b_unused;
 wire clkout4_unused;
wire        clkout5_unused;
wire        clkout6_unused;
wire        clkfbstopped_unused;
wire        clkinstopped_unused;
wire        reset_high;

MMCME2_ADV
```

```
#(.BANDWIDTH              ("OPTIMIZED"),
  .CLKOUT4_CASCADE        ("FALSE"),
  .COMPENSATION           ("ZHOLD"),
  .STARTUP_WAIT           ("FALSE"),
  .DIVCLK_DIVIDE          (1),
  .CLKFBOUT_MULT_F        (9.125),
  .CLKFBOUT_PHASE         (0.000),
  .CLKFBOUT_USE_FINE_PS ("FALSE"),
  .CLKOUT0_DIVIDE_F       (36.500),
  .CLKOUT0_PHASE          (0.000),
  .CLKOUT0_DUTY_CYCLE     (0.500),
  .CLKOUT0_USE_FINE_PS    ("FALSE"),
  .CLKIN1_PERIOD          (10.0))
mmcm_adv_inst
   // Output clocks
 (
  .CLKFBOUT             (clkfbout_clk_wiz_0),
  .CLKFBOUTB            (clkfboutb_unused),
  .CLKOUT0              (clk_out1_clk_wiz_0),
  .CLKOUT0B             (clkout0b_unused),
  .CLKOUT1              (clkout1_unused),
  .CLKOUT1B             (clkout1b_unused),
  .CLKOUT2              (clkout2_unused),
  .CLKOUT2B             (clkout2b_unused),
  .CLKOUT3              (clkout3_unused),
  .CLKOUT3B             (clkout3b_unused),
  .CLKOUT4              (clkout4_unused),
```

```
    .CLKOUT5                (clkout5_unused),
    .CLKOUT6                (clkout6_unused),
     // Input clock control
    .CLKFBIN
(clkfbout_buf_clk_wiz_0),
    .CLKIN1                 (clk_in1_clk_wiz_0),
    .CLKIN2                 (1'b0),
     // Tied to always select the primary
input clock
    .CLKINSEL               (1'b1),
    // Ports for dynamic reconfiguration
    .DADDR                  (7'h0),
    .DCLK                   (1'b0),
    .DEN                    (1'b0),
    .DI                     (16'h0),
    .DO                     (do_unused),
    .DRDY                   (drdy_unused),
    .DWE                    (1'b0),
    // Ports for dynamic phase shift
    .PSCLK                  (1'b0),
    .PSEN                   (1'b0),
    .PSINCDEC               (1'b0),
    .PSDONE                 (psdone_unused),
    // Other control and status signals
    .LOCKED                 (locked_int),
    .CLKINSTOPPED
(clkinstopped_unused),
```

```
    .CLKFBSTOPPED
(clkfbstopped_unused),
    .PWRDWN                 (1'b0),
    .RST                    (reset_high));
  assign reset_high = reset;

  assign locked = locked_int;
// Clock Monitor clock assigning
//--------------------------------------
 // Output buffering
  //-----------------------------------

  BUFG clkf_buf
   (.O (clkfbout_buf_clk_wiz_0),
    .I (clkfbout_clk_wiz_0));



  BUFG clkout1_buf
   (.O   (clk_out1),
    .I   (clk_out1_clk_wiz_0));



endmodule
```

```
module clkcntrl4(clkin,
                 //clkb2,
                 seldig);

   input clkin;
  // output clkb2;
   output seldig;

   //wire XLXN_38;
   //wire XLXN_39;
   wire XLXN_44;
   wire XLXN_47;
   wire XLXN_70;
   wire XLXN_71;
   wire XLXN_72;
   wire XLXN_73;
   wire XLXN_74;
   wire XLXN_75;
   wire XLXN_76;
   wire clkb2_DUMMY;

   GND  XLXI_24 (.G(XLXN_44));

   (* HU_SET = "XLXI_37_73" *)
   CB4CE_MXILINX_clkcntrl4  XLXI_37
(.C(clkb2_DUMMY),
```

```
.CE(XLXN_73),

.CLR(XLXN_76),

.CEO(XLXN_72),
                                        .Q0(),
                                        .Q1(),

.Q2(XLXN_74),
                                        .Q3(),
                                        .TC());
   (* HU_SET = "XLXI_38_74" *)
   CB4CE_MXILINX_clkcntrl4  XLXI_38
(.C(clkb2_DUMMY),

.CE(XLXN_72),

.CLR(XLXN_76),

.CEO(XLXN_70),
                                        .Q0(),
                                        .Q1(),
                                        .Q2(),
                                        .Q3(),
                                        .TC());
   (* HU_SET = "XLXI_39_75" *)
   CB4CE_MXILINX_clkcntrl4  XLXI_39
```

```
(.C(clkb2_DUMMY),

.CE(XLXN_70),

.CLR(XLXN_76),

.CEO(XLXN_71),
                                        .Q0(),
                                        .Q1(),
                                        .Q2(),
                                        .Q3(),
                                        .TC());
   (* HU_SET = "XLXI_40_76" *)
   CB4CE_MXILINX_clkcntrl4  XLXI_40
(.C(clkb2_DUMMY),

.CE(XLXN_71),

.CLR(XLXN_76),
                                        .CEO(),

.Q0(XLXN_75),
                                        .Q1(),
                                        .Q2(),
                                        .Q3(),
                                        .TC());
   VCC  XLXI_41 (.P(XLXN_73));
```

```
   GND  XLXI_43 (.G(XLXN_76));
   BUF  XLXI_328 (.I(clkin),
                  .O(clkb2_DUMMY));
`ifdef XILINX_SIMULATOR
   BUF  XLXI_336 (.I(XLXN_74),
`else   BUF  XLXI_336 (.I(XLXN_75),
`endif
                  .O(seldig));
endmodule


module lab3_digsel(
    input clkin,
    input greset,  //btnR
    output digsel);

       clk_wiz_0 my_clk_inst (.clk_out1(clk),
.reset(greset), .locked(), .clk_in1(clkin));


       clkcntrl4 slowit (.clkin(clk),
.seldig(digsel));

       STARTUPE2 #(.PROG_USR("FALSE"), //
Activate program event security feature.
Requires encrypted bitstreams.
                     .SIM_CCLK_FREQ(0.0)   //
```

```
Set the Configuration Clock Frequency(ns) for
simulation.
                                        )
              STARTUPE2_inst (.CFGCLK(),  //
1-bit output: Configuration main clock output
                             .CFGMCLK(), //
1-bit output: Configuration internal
oscillator clock output
                             .EOS(),     //
1-bit output: Active high output signal
indicating the End Of Startup.
                             .PREQ(),// 1-bit
output: PROGRAM request to fabric output
                             .CLK(),  //
1-bit input: User start-up clock input
                             .GSR(greset),
// 1-bit input: Global Set/Reset input (GSR
cannot be used for the port name)
                             .GTS(),  //
1-bit input: Global 3-state input (GTS cannot
be used for the port name)
                             .KEYCLEARB(), //
1-bit input: Clear AES Decrypter Key input
from Battery-Backed RAM (BBRAM)
                             .PACK(), //
1-bit input: PROGRAM acknowledge input
                             .USRCCLKO(), //
```

```
1-bit input: User CCLK input
                                .USRCCLKTS(), //
1-bit input: User CCLK 3-state enable input
                                .USRDONEO(), //
1-bit input: User DONE pin output control
                                .USRDONETS() //
1-bit input: User DONE 3-state enable output

                               );  // End of
STARTUPE2_inst instantiation

endmodule



module FTCE_MXILINX_clkcntrl4(C,
                              CE,
                              CLR,
                              T,
                              Q);

   parameter INIT = 1'b0;

    input C;
    input CE;
    input CLR;
    input T;
```

```
   output Q;

   wire TQ;
   wire Q_DUMMY;

   assign Q = Q_DUMMY;
   XOR2  I_36_32 (.I0(T),
                  .I1(Q_DUMMY),
                  .O(TQ));
   ///(* RLOC = "X0Y0" *)
   FDCE  I_36_35 (.C(C),
                  .CE(CE),
                  .CLR(CLR),
                  .D(TQ),
                  .Q(Q_DUMMY));
endmodule
`timescale 1ns / 1ps

module CB4CE_MXILINX_clkcntrl4(C,
                               CE,
                               CLR,
                               CEO,
                               Q0,
                               Q1,
                               Q2,
                               Q3,
                               TC);
```

```
 input C;
 input CE;
 input CLR;
output CEO;
output Q0;
output Q1;
output Q2;
output Q3;
output TC;

wire T2;
wire T3;
wire XLXN_1;
wire Q0_DUMMY;
wire Q1_DUMMY;
wire Q2_DUMMY;
wire Q3_DUMMY;
wire TC_DUMMY;

assign Q0 = Q0_DUMMY;
assign Q1 = Q1_DUMMY;
assign Q2 = Q2_DUMMY;
assign Q3 = Q3_DUMMY;
assign TC = TC_DUMMY;
(* HU_SET = "I_Q0_69" *)
FTCE_MXILINX_clkcntrl4 #( .INIT(1'b0) )
```

```
I_Q0 (.C(C),
                                        .CE(CE),
                                        .CLR(CLR),
                                        .T(XLXN_1),
                                        .Q(Q0_DUMMY));
   (* HU_SET = "I_Q1_70" *)
   FTCE_MXILINX_clkcntrl4 #( .INIT(1'b0) )
I_Q1 (.C(C),
                                        .CE(CE),
                                        .CLR(CLR),
                                        .T(Q0_DUMMY),
                                        .Q(Q1_DUMMY));
   (* HU_SET = "I_Q2_71" *)
   FTCE_MXILINX_clkcntrl4 #( .INIT(1'b0) )
I_Q2 (.C(C),
                                        .CE(CE),
                                        .CLR(CLR),
                                        .T(T2),
                                        .Q(Q2_DUMMY));
   (* HU_SET = "I_Q3_72" *)
   FTCE_MXILINX_clkcntrl4 #( .INIT(1'b0) )
I_Q3 (.C(C),
                                        .CE(CE),
                                        .CLR(CLR),
                                        .T(T3),
                                        .Q(Q3_DUMMY));
   AND4  I_36_31 (.I0(Q3_DUMMY),
```

```
                    .I1(Q2_DUMMY),
                    .I2(Q1_DUMMY),
                    .I3(Q0_DUMMY),
                    .O(TC_DUMMY));
   AND3  I_36_32 (.I0(Q2_DUMMY),
                    .I1(Q1_DUMMY),
                    .I2(Q0_DUMMY),
                    .O(T3));
   AND2  I_36_33 (.I0(Q1_DUMMY),
                    .I1(Q0_DUMMY),
                    .O(T2));
   VCC   I_36_58 (.P(XLXN_1));
   AND2  I_36_67 (.I0(CE),
                    .I1(TC_DUMMY),
                    .O(CEO));
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/22/2020 02:46:53 PM
// Design Name:
// Module Name: Top_Level
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////


module Top_Level(
    input [7:0] sw,
    input btnU,
```

```
    input btnD,
    input btnR,
    input clkin,
    output [6:0] seg,
    output dp,
    output [3:0] an
    );

    wire [7:0] t;
    wire [6:0] a;
    wire [6:0] b;
    wire dig_sel;

    assign an[0] = dig_sel;
    assign an[1] = ~dig_sel;
    assign an[2] = 1'b1;
    assign an[3] = 1'b1;
    assign dp = 1'b1;

    Incrementer Adder(.a(sw[7:0]),
.b({btnU,btnD}),
.s({t0,t1,t2,t3,t4,t5,t6,t7}));
    Segment_Display Lower(.n({t0,t1,t2,t3}),
.sego(a[6:0]));
    Segment_Display Upper(.n({t4,t5,t6,t7}),
.sego(b[6:0]));
    lab3_digsel DigSel(.clkin(clkin),
```

```
.greset(btnR), .digsel(dig_sel));
    MUX2_1x8 Final(.in0({1'b0,b[6:0]}),
.in1({1'b0,a[6:0]}), .sel(dig_sel),
.out(seg[6:0]));

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/22/2020 04:39:39 PM
// Design Name:
// Module Name: Top_Level_Simulation
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Top_Level_Simulation();
    reg sw0, sw1, sw2, sw3, sw4, sw5, sw6,
sw7, btnU, btnD, btnR, clkin;
```

```
    wire seg0, seg1, seg2, seg3, seg4, seg5,
seg6, dp, an0, an1, an2, an3;

    Top_Level UUT (
    .sw({sw0, sw1, sw2, sw3, sw4, sw5, sw6,
sw7}),
    .btnU(btnU),.btnD(btnD), .btnR(btnR),
.clkin(clkin),
    .seg({seg0, seg1, seg2, seg3, seg4, seg5,
seg6}),
    .an({an0, an1, an2, an3}),
    .dp(dp)
    );

    parameter PERIOD = 10;
    parameter real DUTY_CYCLE = 0.5;
    parameter OFFSET = 2;

    initial    // Clock process for clkin
    begin
        #OFFSET
            clkin = 1'b1;
        forever
        begin
            #(PERIOD-(PERIOD*DUTY_CYCLE))
clkin = ~clkin;
        end
```

```
    end

   initial
   begin
     // add your (input) stimuli here
     // to set signal foo to value 0 use
     // foo = 1'b0;
     // to set signal foo to value 1 use
     // foo = 1'b1;
     //always advance time my multiples of
100ns
     // to advance time by 100ns use the
following line
    #100;
    //Lower
    sw0=1'b0;
    sw1=1'b0;
    sw2=1'b0;
    sw3=1'b0;
    //Upper
    sw4=1'b0;
    sw5=1'b0;
    sw6=1'b0;
    sw7=1'b0;
    // Button
    btnU=1'b0;
    btnD=1'b0;
```

```
// Display 0, 0
// --------------------- Time: 100ns
#100;
//Lower
sw0=1'b1;
sw1=1'b0;
sw2=1'b0;
sw3=1'b0;
//Upper
sw4=1'b1;
sw5=1'b0;
sw6=1'b0;
sw7=1'b0;
//Button
btnU=1'b0;
btnD=1'b0;
// Display 1, 1
// --------------------- Time: 200ns
#100;
//Lower
sw0=1'b0;
sw1=1'b1;
sw2=1'b0;
sw3=1'b0;
//Upper
sw4=1'b0;
sw5=1'b1;
```

```
sw6=1'b0;
sw7=1'b0;
//Button
btnU=1'b0;
btnD=1'b0;
// Display 2, 2
// --------------------- Time: 300ns
#100;
//Lower
sw0=1'b1;
sw1=1'b1;
sw2=1'b0;
sw3=1'b0;
//Upper
sw4=1'b1;
sw5=1'b1;
sw6=1'b0;
sw7=1'b0;
//Button
btnU=1'b0;
btnD=1'b0;
// Display 3, 3
// --------------------- Time: 400ns
#100;
//Lower
sw0=1'b0;
sw1=1'b0;
```

```
sw2=1'b1;
sw3=1'b0;
//Upper
sw4=1'b0;
sw5=1'b0;
sw6=1'b1;
sw7=1'b0;
//Button
btnU=1'b0;
btnD=1'b0;
// Display 4, 4
// ---------------------- Time: 500ns
#100;
//Lower
sw0=1'b1;
sw1=1'b0;
sw2=1'b1;
sw3=1'b0;
//Upper
sw4=1'b1;
sw5=1'b0;
sw6=1'b1;
sw7=1'b0;
//Button
btnU=1'b0;
btnD=1'b0;
// Display 5, 5
```

```
// -------------------- Time: 600ns
#100;
//Lower
sw0=1'b0;
sw1=1'b1;
sw2=1'b1;
sw3=1'b0;
//Upper
sw4=1'b0;
sw5=1'b1;
sw6=1'b1;
sw7=1'b0;
//Button
btnU=1'b0;
btnD=1'b0;
// Display 6, 6
// -------------------- Time: 700ns
#100;
//Lower
sw0=1'b1;
sw1=1'b1;
sw2=1'b1;
sw3=1'b0;
//Upper
sw4=1'b1;
sw5=1'b1;
sw6=1'b1;
```

```
sw7=1'b0;
//Button
btnU=1'b0;
btnD=1'b0;
// Display 7, 7
// -------------------- Time: 800ns
#100;
//Lower
sw0=1'b0;
sw1=1'b0;
sw2=1'b0;
sw3=1'b1;
//Upper
sw4=1'b0;
sw5=1'b0;
sw6=1'b0;
sw7=1'b1;
//Button
btnU=1'b0;
btnD=1'b0;
// Display 8, 8
// -------------------- Time: 900ns
#100;
//Lower
sw0=1'b1;
sw1=1'b0;
sw2=1'b0;
```

```
sw3=1'b1;
//Upper
sw4=1'b1;
sw5=1'b0;
sw6=1'b0;
sw7=1'b1;
//Button
btnU=1'b0;
btnD=1'b0;
// Display 9, 9
// --------------------- Time: 1000ns
#100;
//Lower
sw0=1'b0;
sw1=1'b1;
sw2=1'b0;
sw3=1'b1;
//Upper
sw4=1'b0;
sw5=1'b1;
sw6=1'b0;
sw7=1'b1;
//Button
btnU=1'b0;
btnD=1'b0;
// Display A, A
// --------------------- Time: 1100ns
```

```
#100;
//Lower
sw0=1'b1;
sw1=1'b1;
sw2=1'b0;
sw3=1'b1;
//Upper
sw4=1'b1;
sw5=1'b1;
sw6=1'b0;
sw7=1'b1;
//Button
btnU=1'b0;
btnD=1'b0;
// Display b, b
// --------------------- Time: 1200ns
#100;
//Lower
sw0=1'b0;
sw1=1'b0;
sw2=1'b1;
sw3=1'b1;
//Upper
sw4=1'b0;
sw5=1'b0;
sw6=1'b1;
sw7=1'b1;
```

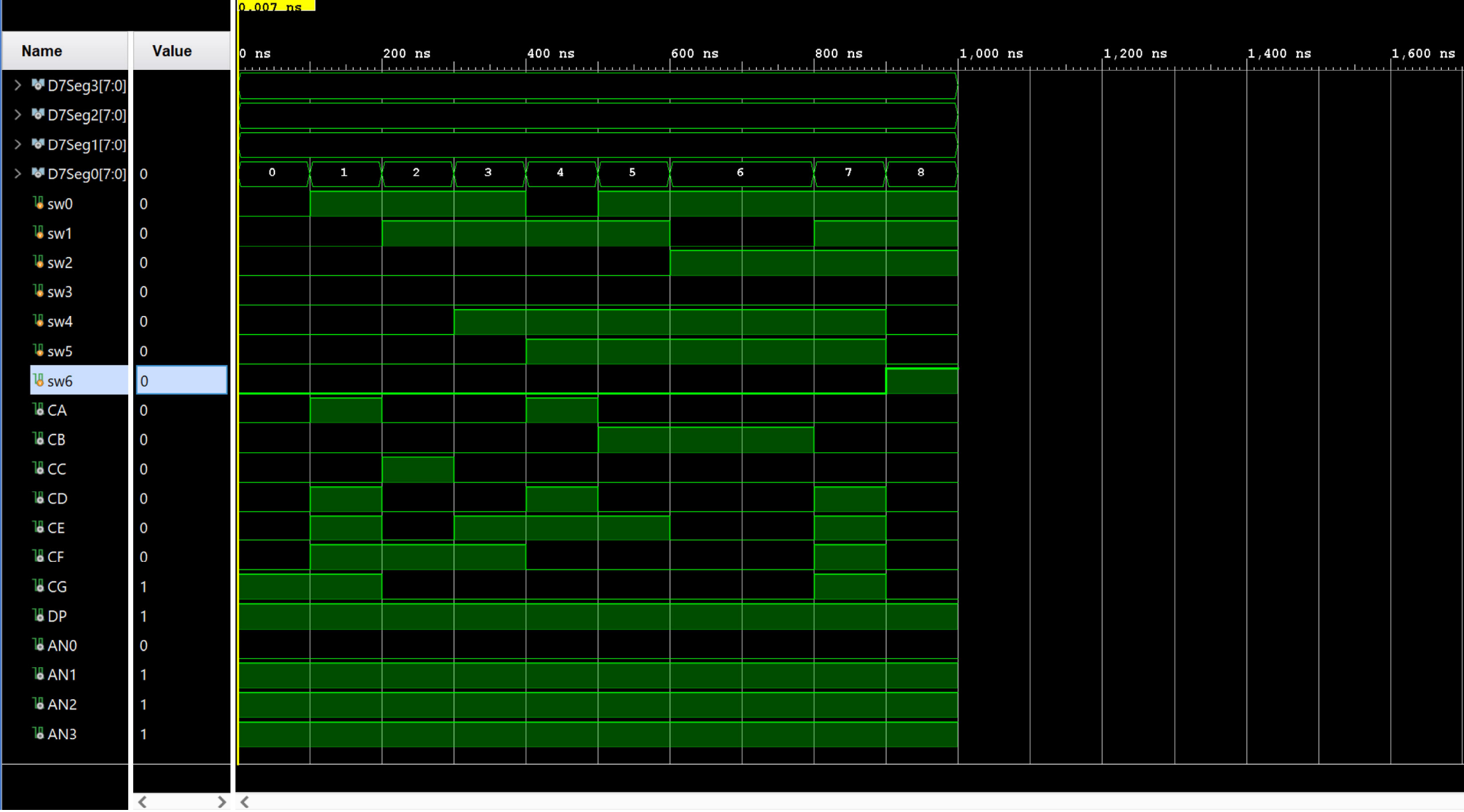
```
//Button
btnU=1'b0;
btnD=1'b0;
// Display C, C
// --------------------- Time: 1300ns
#100;
//Lower
sw0=1'b1;
sw1=1'b0;
sw2=1'b1;
sw3=1'b1;
//Upper
sw4=1'b1;
sw5=1'b0;
sw6=1'b1;
sw7=1'b1;
//Button
btnU=1'b0;
btnD=1'b0;
// Display d, d
// --------------------- Time: 1400ns
#100;
//Lower
sw0=1'b0;
sw1=1'b1;
sw2=1'b1;
sw3=1'b1;
```

```
//Upper
sw4=1'b0;
sw5=1'b1;
sw6=1'b1;
sw7=1'b1;
//Button
btnU=1'b0;
btnD=1'b0;
// Display E, E
// --------------------- Time: 1500ns
#100;
//Lower
sw0=1'b1;
sw1=1'b1;
sw2=1'b1;
sw3=1'b1;
//Upper
sw4=1'b1;
sw5=1'b1;
sw6=1'b1;
sw7=1'b1;
//Button
btnU=1'b0;
btnD=1'b0;
// Display F, F
// --------------------- Time: 1600ns
#100;
```

```
//Lower
sw0=1'b1;
sw1=1'b0;
sw2=1'b0;
sw3=1'b0;
//Upper
sw4=1'b0;
sw5=1'b1;
sw6=1'b0;
sw7=1'b0;
//Button
btnU=1'b0;
btnD=1'b0;
// Display 1, 2
// ---------------------- Time: 1700ns
#100;
//Lower
sw0=1'b1;
sw1=1'b1;
sw2=1'b0;
sw3=1'b0;
//Upper
sw4=1'b0;
sw5=1'b1;
sw6=1'b1;
sw7=1'b0;
//Button
```

```
btnU=1'b0;
btnD=1'b0;
// Display 3, 6
// --------------------- Time: 1800ns
#100;
//Lower
sw0=1'b1;
sw1=1'b0;
sw2=1'b0;
sw3=1'b0;
//Upper
sw4=1'b1;
sw5=1'b1;
sw6=1'b0;
sw7=1'b1;
//Button
btnU=1'b0;
btnD=1'b0;
// Display 1, b
// --------------------- Time: 1900ns
#100;
//Lower
sw0=1'b0;
sw1=1'b0;
sw2=1'b1;
sw3=1'b1;
//Upper
```

```
    sw4=1'b0;
    sw5=1'b0;
    sw6=1'b1;
    sw7=1'b0;
    //Button
    btnU=1'b0;
    btnD=1'b0;
    // Display c, 4
    // --------------------- Time: 2000ns
    #100;
    //Lower
    sw0=1'b1;
    sw1=1'b1;
    sw2=1'b1;
    sw3=1'b1;
    //Upper
    sw4=1'b1;
    sw5=1'b1;
    sw6=1'b1;
    sw7=1'b1;
    //Button
    btnU=1'b0;
    btnD=1'b1;
    // Display A, A+2 OVER TEST
    // --------------------- Time: 2100ns
   end
endmodule
```


0.007 ns
Name
Value
0 ns
200 ns
400 ns
600 ns
800 ns
1,000 ns
1,200 ns
1,400 ns
1,600 ns
D7Seg3[7:0]
D7Seg2[7:0]
D7Seg1[7:0]
D7Seg0[7:0] 0
sw0 0
sw1 0
sw2 0
sw3 0
sw4 0
sw5 0
sw6 0
CA 0
CB 0
CC 0
CD 0
CE 0
CF 0
CG 1
DP 1
AN0 0
AN1 1
AN2 1
AN3 1
0
1
2
3
4
5
6
7
8