CSE100 Lab 4 writeup

Counting with flip flops

Student Name: Artyom Martirosyan

Lab Sec: 1B

Date: 5/8/2020

Description:

For this assignment the students were given the challenge of creating a 16bit counter which was capable of counting both upwards and downwards as well as being able to load in a 16-bit number into the register. The students were also required to synchronize several inputs using d-flip flops in order to synchronise with the clock inside of the 16-bit counter. The inputs were a btnL which would be the load option of the counter as well as a btnU and btnD which would have the clock count one bit. The students also needed to implement a btnC which would count upwards until it fffc-ffff where it would stop(cant count if first 14 bits were set to one). The outputs will be a 4 analogue digital display which will be alternating in order to show all 4 values(the device will be alternating too quickly for the human eye to notice the change). There will also be a UTC and DTC function which will be connected to two additional LEDS which will be lit if the counter has reached zero or if the counter has reached the highest number it can count to before starting over(FFFF). Inside of the 16 bit up down counter the student will be creating four 4-bit up down counters and putting them together in order to develop the 16 bit counter. The counters will also hold a UTC and DTC output which will tell the module if the counter has either reached the smallest value it can count to(zero) or the largest(F) and the student will need to combine these values from the 4 but counters in order to show the UTC and DTC when the 16 bit counter has reached its limits. Once the student has developed a counter which is giving the expected output the student will then need to create a ring counter as well as a selector. The ring counter will simply be 4 flip flops connected in a loop so that they would depict a 4 bit value which would only have one bit set to one at any time. This number would then be connected to both the analogues for the seven segment display as well as to a selector which would select which 4 bits would be sent to the seven segment hex converter(same one as used in the previous lab manual).

Methods:

T Flip-Flop:

If the student reads through the textbook they will hopefully uncover a schematic for a functioning up down counter and will notice that the counter relies on t flip-flops. Once releasing this they can also read the next section and find the schematic for the t flip-flop as shown below:

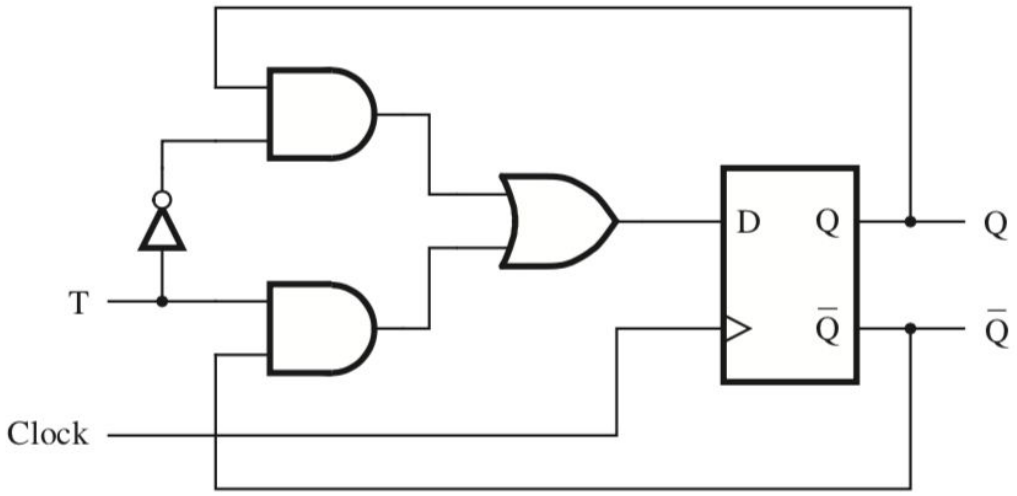


*Figure 1 is a schematic of a T flip-flop provided by the textbook

From this schematic the student should be able to create a T-flip flop

Edge detector:

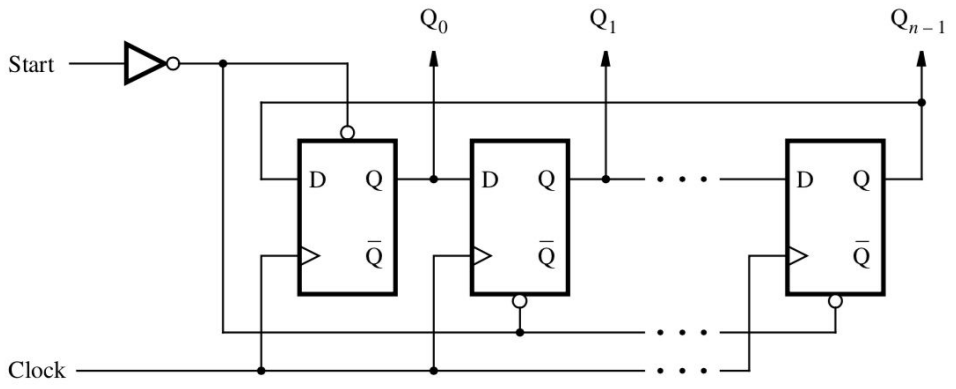
For the edge detector the student needs to create an edge detector such that a 1 is outputted when the input changes from zero to a one(a schematic will be shown below as well as logic)

Selector:

The selector can be looked at as a four to four multiplexor as it will be receiving a four bit value from the ring counter(where one of the four bits will be one and the rest zero). And it will simply select 4 bits according to the selector, for example if I was to receive 1000 meaning that the most significant bit of the selector is one i would take bits 15:12 from the 16 bit number given to me(the most significant four bits). Again please look in the section below for a schematic.

Ring counter:

For the ring counter the student will be reluctant to find a schematic for an n-bit ring counter in which if the student wishes to add more bits they will simply have to add an additional flip flop as seen below:



*The figure above(figure 2) is a schematic of an n-bit ring counter provided from the textbook.

Ring counter:

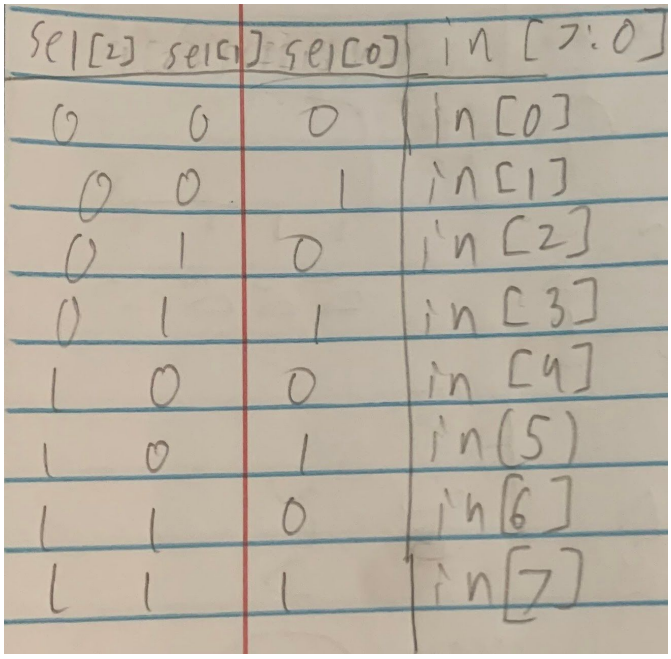
For the ring counter the student will be reluctant to find a schematic for an n-bit ring counter

8 to 1 Mux(m8_1e):

Note: the section below is a snippet from the lab 3 writeup since the multiplexor is implemented from the hex 7 segment display which is also borrowed from lab 3. This will be added to the manual with the assumption that the reader does not have access to previous lab manuals

For this module the student must start off by writing out the truth table and deriving a boolean expression for the multiplexer:

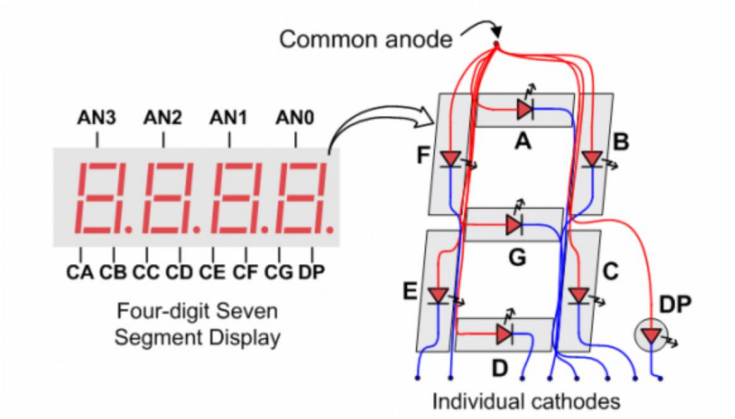
*The figure below(fig 3) is a truth table for m8_le taken from lab manual 3



From this the student is able to derive the boolean expression(note: actual schematic in results):
Output = e & ((~sel[2]&~sel[1]&~sel[0] & 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]) |

Seven segment display:

Again rather than repeating the same instruction from the previous lab manual here is the snipped from the lab 3 writeup(will be sourced below). Before we can design the seven segment display we must first understand the logic behind the display. For starters the display is an active high component meaning that the led/analogue will light up when it is set to zero and be turned off when it is set to one. Also each line in the individual analogues are depicted using 7 lines meaning that you will need to set the appropriate lines depending on the value you want displayed.



* figure 4 above is a schematic of the displays on the basys 3 circuit board taken from the reference manual

Since we are required to use multiplexors for the seven segment display we will start off by creating a truth table for one component of the display:

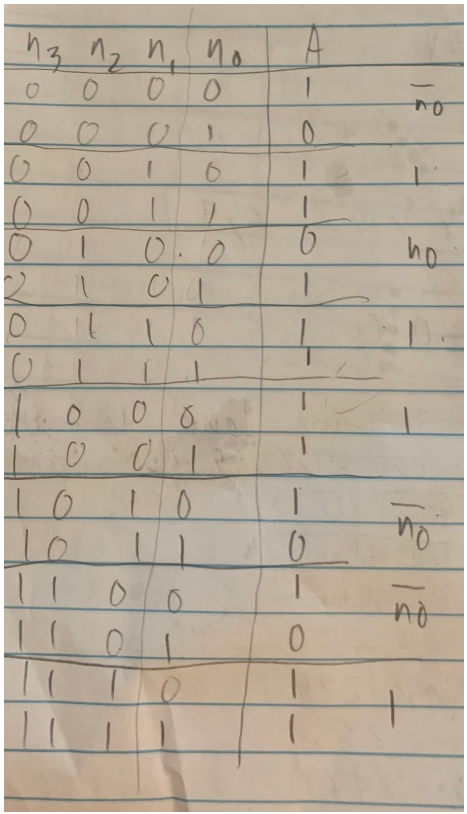


*Figure 5 to the left is a truth table of what A from the display above will represent according to the 4 bit vector it is provided.

As seen to the left we will be using n[3:1] as the switches for the mux. We will also use {1,~n[0], ~n[0], 1, 1, n[0], 1, ~n[0]} as the either bit input into the 8-1 multiplexor. The module will also take in an enable making sure that the wrong value isn't being displayed on one of the two analogues.

Four bit up/down counter(countUD4L):

The student will be creating a four bit up/down counter with a load capability meaning that the student must have a load function attached to each flip flop. Fortunately, the textbook has a schematic of a three-bit up down counter without load capabilities as seen below:

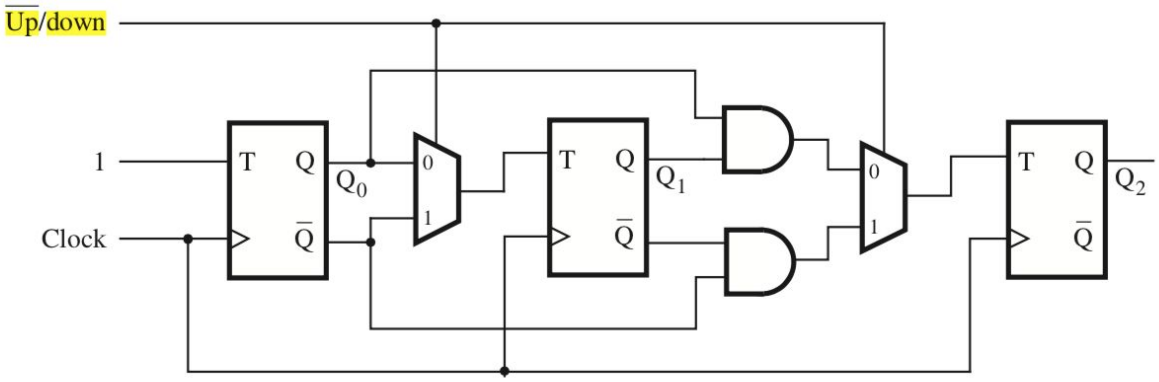
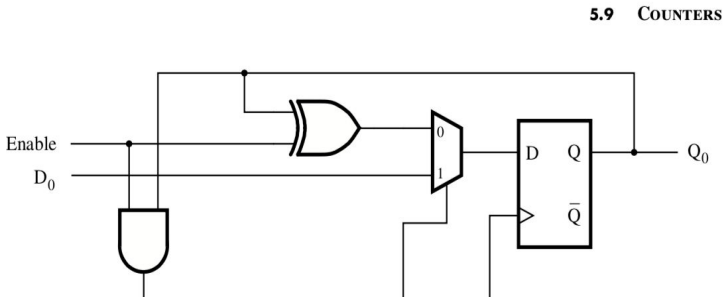


*Figure 6 above is an up/down 3-bit counter without load capabilities

With this data we can simply add an extra t-flip flop which would require another two to one multiplexor which would hold the logic and of all of the results from the flip flops(Q for upward and ~Q for downward counting). In order to add loading we can simply add an additional multiplexor which would decide if a load is being initialized(not in order to load a value you will need to do an xor with the result from the T flip-flop and the bit being loaded). Warning before you implement this keep in mind that you will be requiring a t-flip flop therefore you must create an additional module for the t flip flop and implement the schematic above. Below is an example of a parallel load on a counter(input logic should still be the same with an up/down counter as you simply need a multiplexor before the T inputs). For UTC and DTC simply and all of the ~Qs for DTC and all of the Qs for UTC.



The image to the left(figure 7) is a design for a load. As seen to the left you will need to xor the output value of the flip flip with the load bit before running it into a secondary multiplexor.

16 bit up/down counter(countUD26L):

For the 16 bit up/down counter we will simply be implementing the design provided to us from the image below:

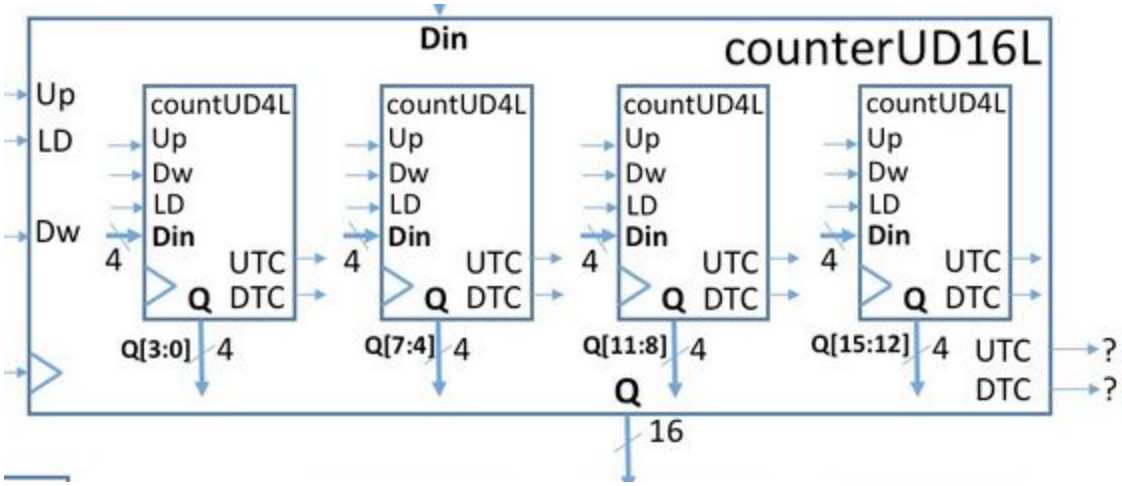
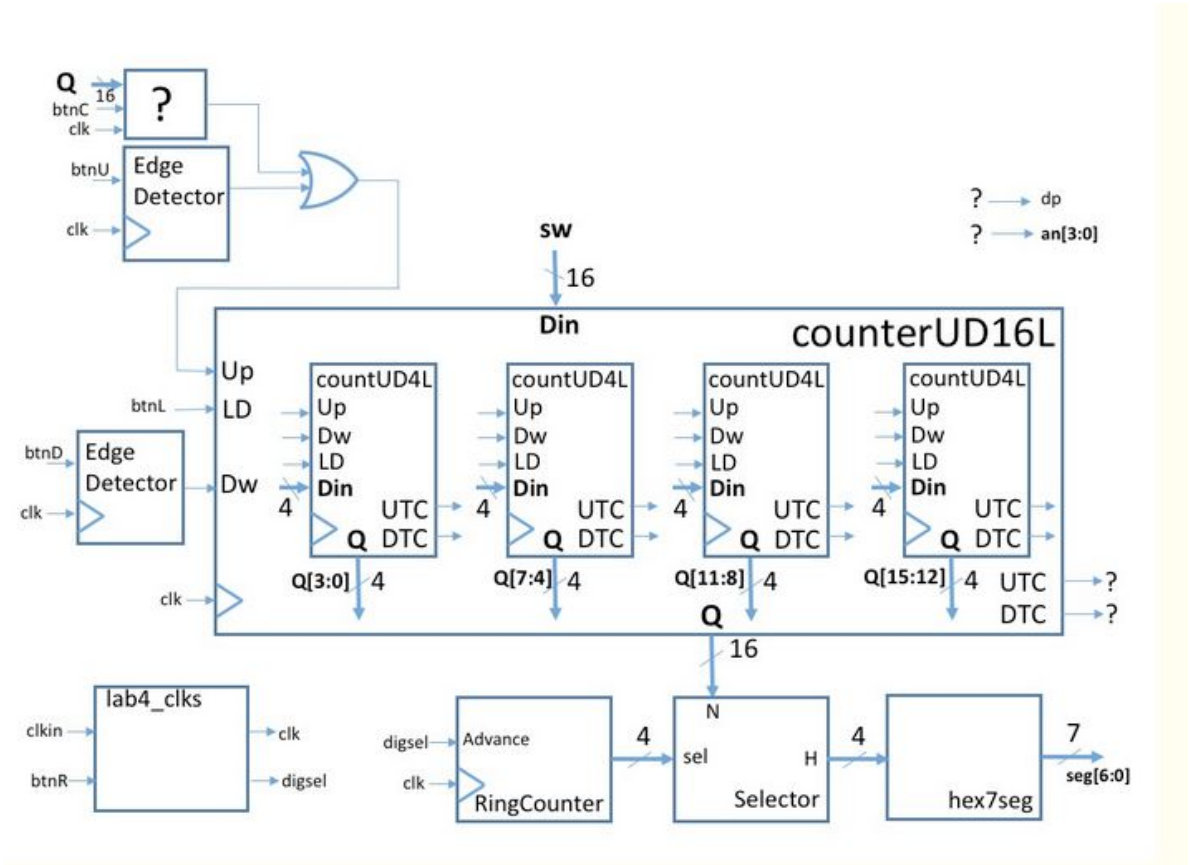


*Figure 8 above is a basic schematic of the 16 bit up/down counter taken from the lab four manual

As seen above we will simply be attaching several 4 bit counters to each other as well as loading in the appropriate four bits into each counter. Note: there are a few extra logical aspects not mentioned in the schematic above which will be mentioned in detail below. After the results from the four bit counters we can simply combine all 16 bits as the output as well as having the UTC be a logical and of the four UTCs from the four counters. The logic for the DTC is similar to the logic for UTC, but with the DTCs from the four counters instead.

Top module:

Before explaining the logic behind the top module here is the schematic:



*The figure above(figure 9) is the basic overview schematic of the top module for the lab

For the top module we will need to start off by synchronizing all of the different buttons and the clk provided from the lab4_clks module(provided by instructors). From there we will first need to implement the logic for btnc which will also need to be synchronised via a D flip-flip once we have synchronised it we will need to also check if the number is already between fffc and ffff as the counter is not allowed to reach these values if only btnC is selected(simply check if the 14 most significant bits are one). We will also need to attach led 0 and led 15(not shown in schematic above) to UTC and DTC so that we can confirm that the value is about to pass the counting edges. We will also need to connect the result from the ring counter to the analogues(invert the ring counter results since the analogue uses 1 as off and zero as on).

Results:

There weren't many issues implementing the top module: the main issues were notices inside of the four-bit counter as at first I was unaware that the load bit needed an xor (since what would the result be if the load bit was a one and the output of the flip

flop was also a one?). Once that issue was fixed the rest of the assembly worked as expected.

Design:

Top schematic:

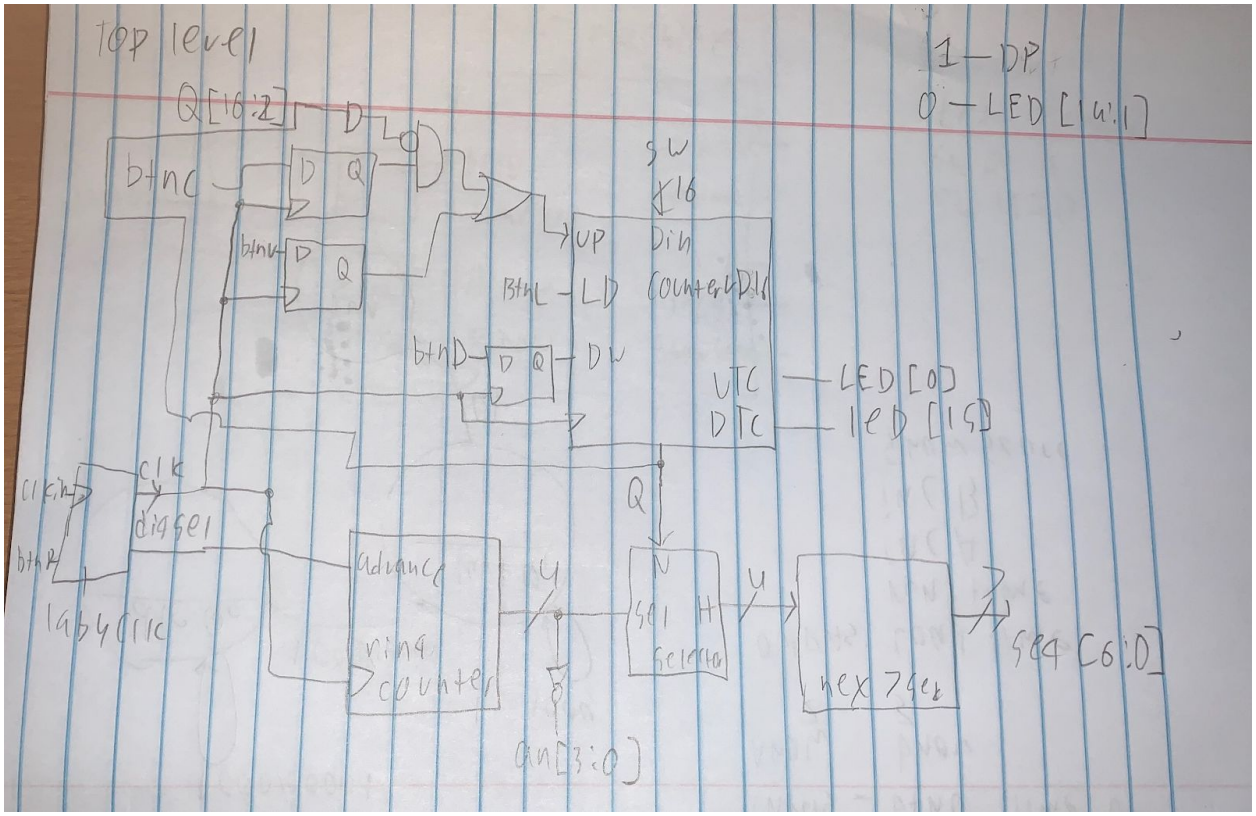


*Figure 10 above is a drawing of the top level schematic

For the top schematic the student needed to develop a case for btnC so that it didn't increment if Q was between fffc and ffff. If the student writes out these values in binary the student will notice that the numbers between fffc and ffff are simply all numbers where their first 14 bits are one whereas the rest vary. Therefore the student can simply and the first 14 bits in order to see if this condition was met. The rest of the schematic is very straightforward as btnU and btnD are going into edge detectors hench being synchronised so that they will only depict a one on the rising edge of D. Also keep in mind that dp and led[14:0] should all be set to off(zero for led 1 for dp) and that an[3:0] should be the result from the ring counter(remember to invert the result). The top module will have a clkin, btnL, btnD, sw[15:0], btnU, btnR, btnC as its inputs and dp,

an[3:0], led[15:0], and seg[6:0] as its outputs. Note:btnR is a reset for the clock provided from the manual.

T flip-flop:

Rather than hand drawing the schematic for the t flip-flop please refer to the detailed depiction of a t flip-flop in the methods section above. Keep in mind the reason that these are helpful is that these will make counting up and down easier as the logic can come out of one module making the counter module easier to read. The t flip flop will take in T(one single bit to be represented) and clk as its inputs and have Q and ~Q as its outputs.

Edge detector:

The edge detector will simply take in a value Q and clk and output Q and the inverted result from the d flip flop as seen below:

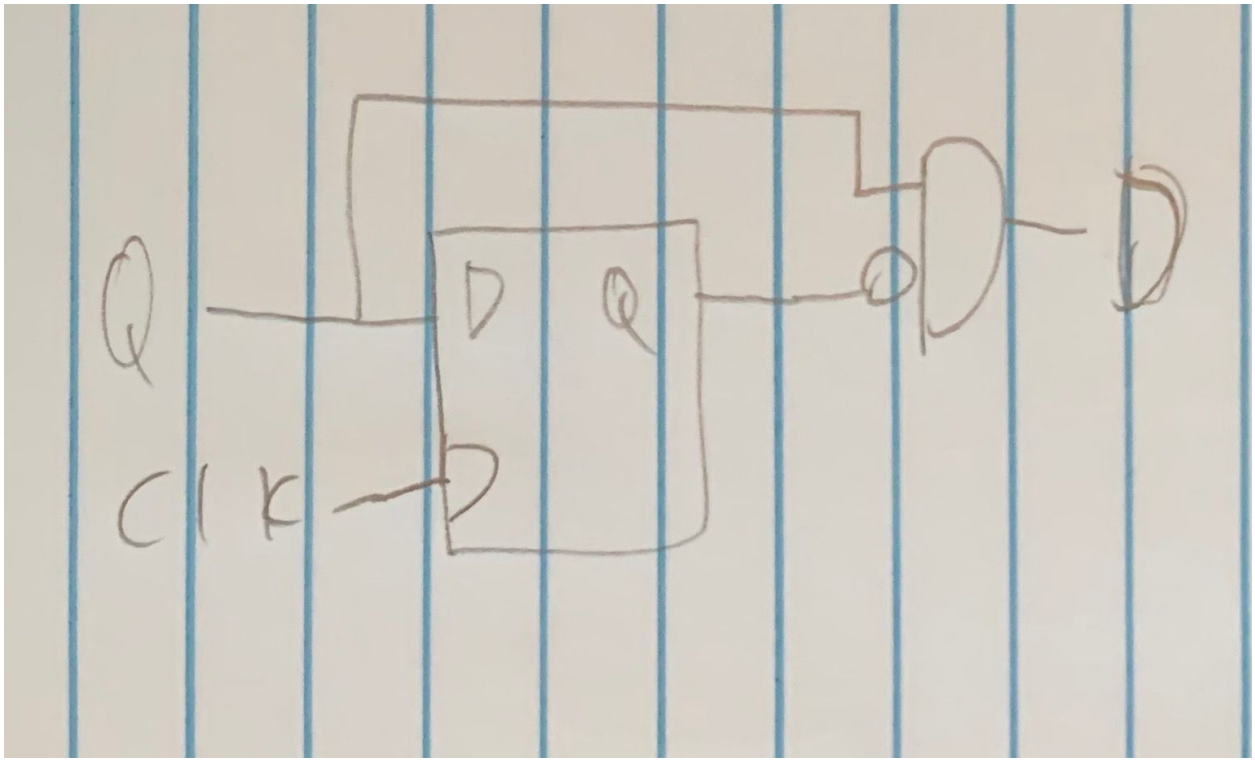
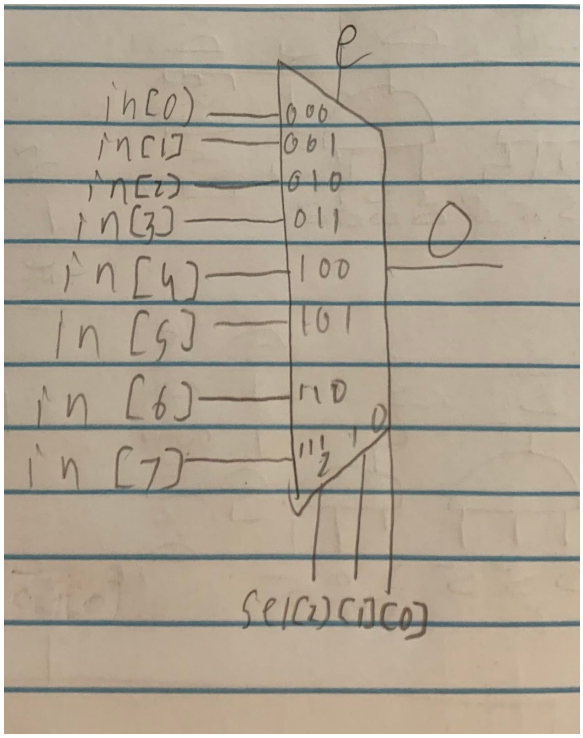


*Figure 11 is a schematic of the edge detector

M8_1e:

Note: for those who haven't viewed the previous manual the logic for the m8_1 multiplexor was reimplemented from the previous lab:

*Figure 12 to the left is the schematic of the 8 to 1 multiplexor.



For the actual logic for this look at the methods section as it is simply a set of and operations which will represent each possible input.

7segment display:

(note:this was taken from lab writeup 3)Due to the size of the seven segment display it is too difficult to develop a schematic, rather I will explain my logic. As seen in the methods section I created truth tables for each line of the display and I would use n[3:1] as the selecting switches meaning that I would use n[0] as the relation case since it aligned with most of the possible inputs. Once I had developed the inputs for each case(derived from the table mentioned above) I would then call 7 8-1 multiplexer(one for each line) deciding whether the inputted value would need the selected segment to be displayed. I personally believe that this module is more efficient then the logic which was implemented for the previous lab(lab3) since this module was based around

multiplexores making the modules less tedious and more developed(easier to understand/debug).

counterUD16L:

For the 16 bit counter you will be taking in [15:0]Din, Up, Dw, LD, clk, as inputs and UTC, DTC, as well as [15:0]Q as outputs. For the actual counting aspect follow the schematic below as each four bit counter depends on the result from the previous counter(if that counter has reached one of its limits). For assigning UTC and DTC simply and all of the UTCs and DTCs from the four 4bit counters as seen below:

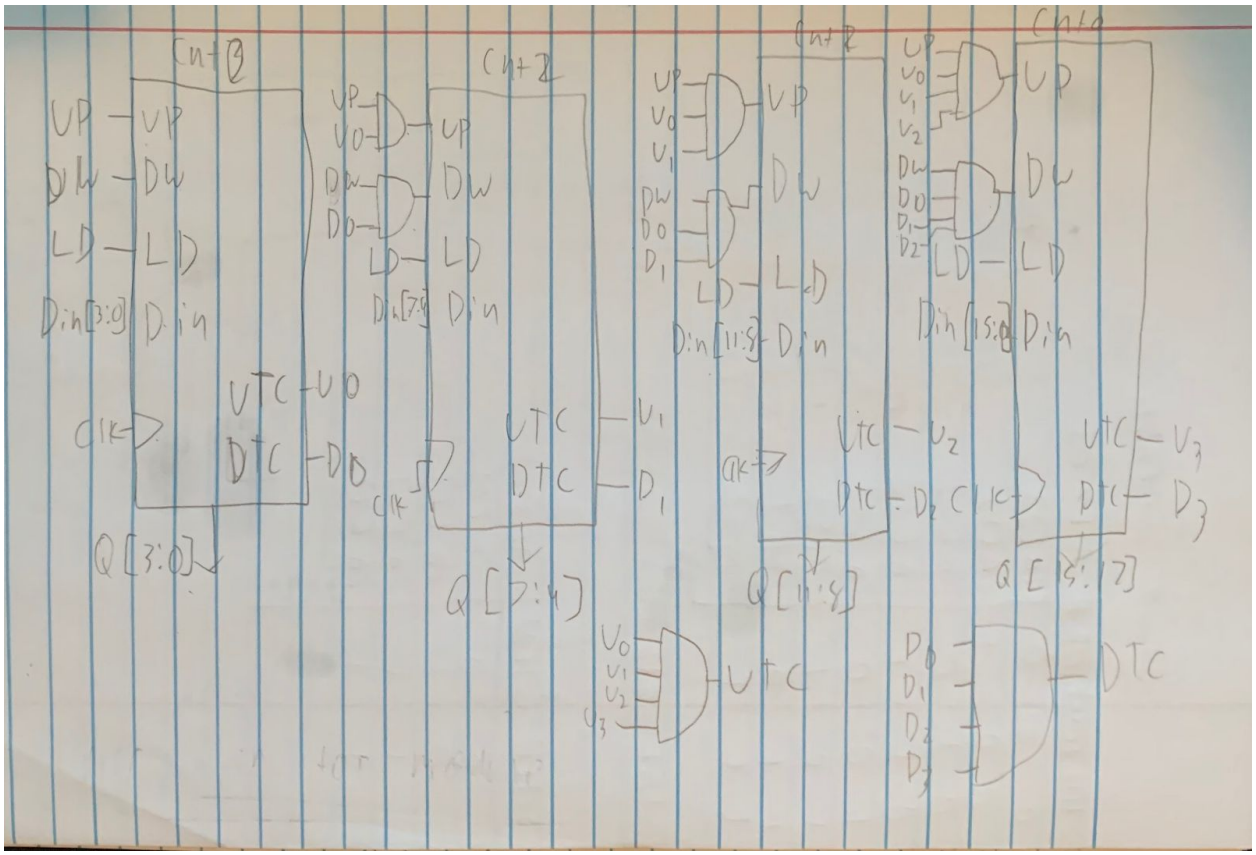
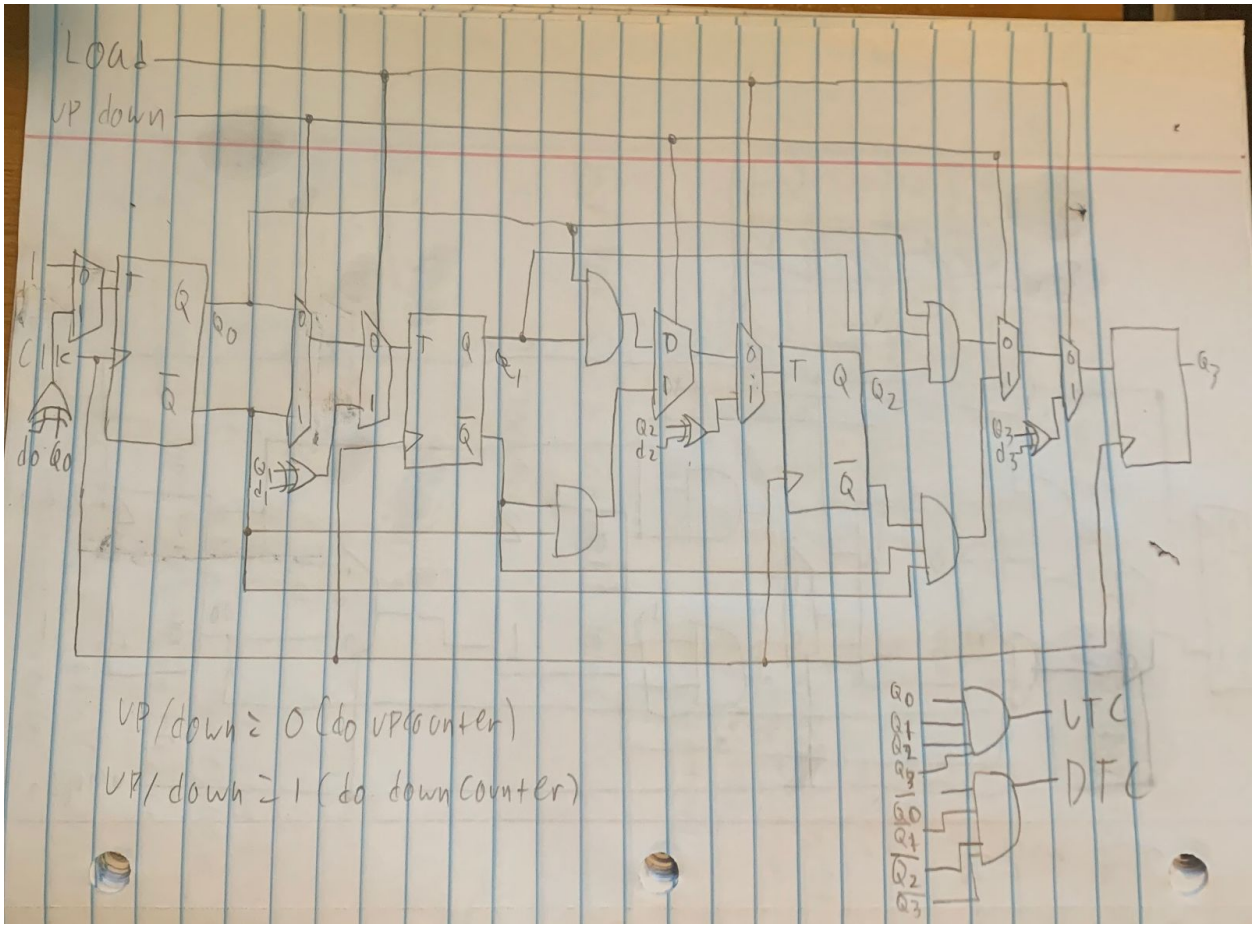


*figure 13 above is the schematic for the 16 bit up/down counter

counterUD5L:

For the 16 bit counter you will be taking in [3:0]Din, Up, Dw, LD, clk, as inputs and UTC, DTC, as well as Q[3:0] for the outputs. For this design we will be following the

schematic provided from the textbook above as well as adding an additional multiplexor in order to be able to implement the load feature.



*The figure above(figure 14) is the schematic for a four bit up/down counter

Ring counter:

For the ring counter the student should simply follow the schematic from the textbook by having four flip flops running into each other in a loop style. Where Q[3:0] are the different outputs from the flip flops.

selector:

For the selector the inputs are [15:0] N, [3:0] sel, and the outputs is [3:0] H simply had each set of 4 bit multiplied by 4{sel[x]} as seen below:

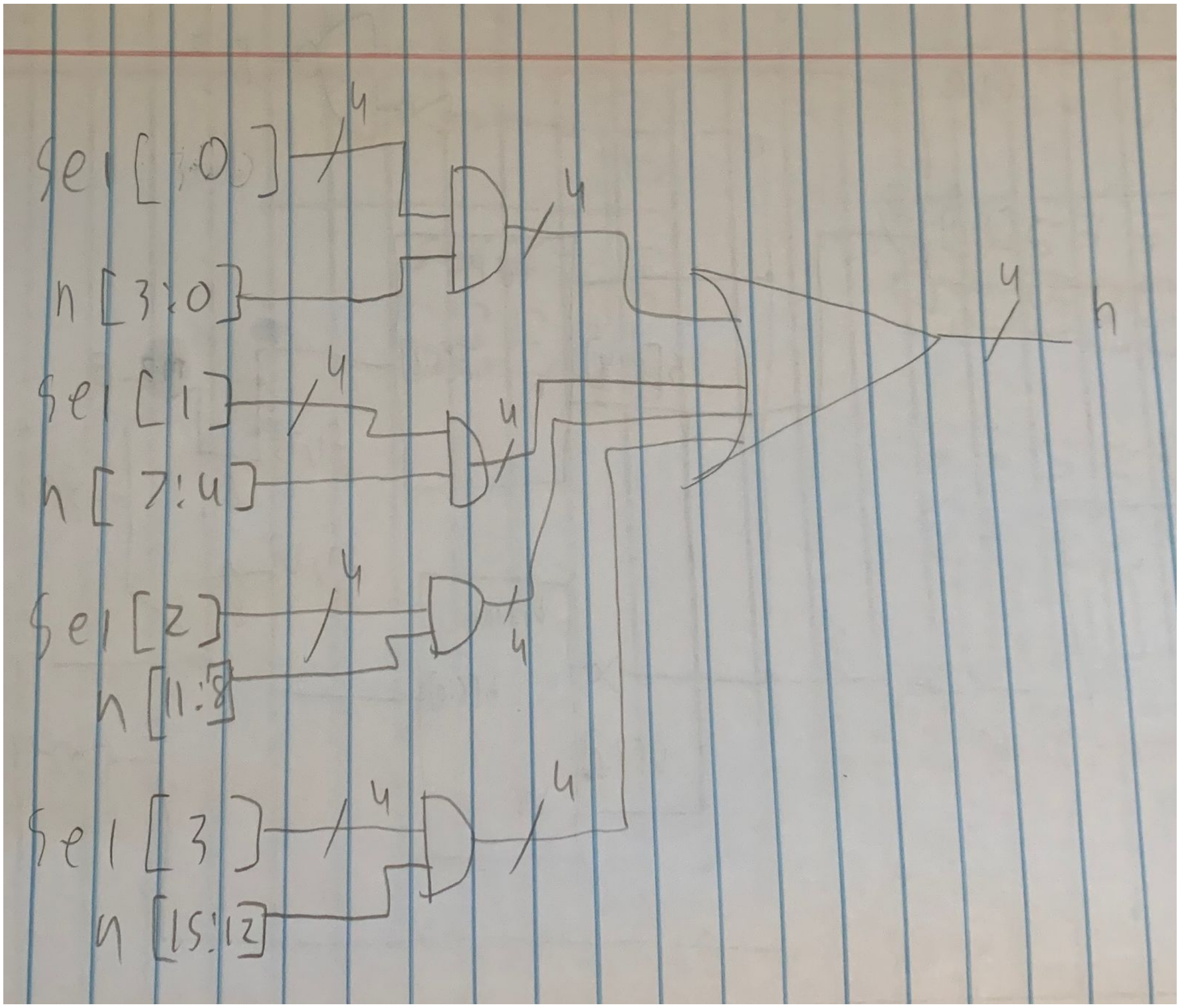


*figure 15 above is the schematic for the selector

For example the code for the first four bits would be:zero = {4{sel[0]}} & N[3:0];

Simulations:

Edge detector:

For simulating the edge detector I simply ran the program and had some scenarios where the input would be set to high and then low in order to make sure that the output was being set correctly.

countUD4l(four bit up/down counter sim) and countUD16L:

For simulating the four bit counter I started off with the edge case by setting all three buttons to one(LD,Up, DW) in order to make sure that LD had precedence over the other two conditions. I then made sure that the edge cases were met by incrementing upward past FFFF and counted down past the 0000 value in order to make sure that UTC and DTC were acting as expected. Both tests were nearly identical since the 16 bit counter is just four 4 bit counters. In this area I noticed that the counter would be able to load the value properly, but wouldn't be able to hold the value since I wasnt sending the result back to the flip flop and xoring it with LD which I realized later on after more experimenting.

Top module:

For simulating the top module I simply started off by loading a bit making sure that ld still had precedence, then I also tested btnc as well as btnU and btnD. One I did several tests in order to make sure the values were incrementing properly I also tested the edge cases in order to make sure that led 15 and led 0 were being lit up properly. Note: I also did the test provided by the instruction which would do some quick swaps in order to test the edge cases to make sure that UTC and DTC are being displayed accordingly. In this area I discovered that my leds were not plugged in properly and I quickly made the adjustments.

Questions:

There are no questions for this lab

Conclusion:

There were some challenges with properly implementing the four bit up/down counter, but once the counter was properly implemented everything else was put together smoothly. One thing which is very important on larger projects such as this one is to draw schematics prior to actually coding in order to make sure that the logic is correct so that there aren't any incorrect implementations later on in the logical aspect of this. This lab was the first time that students had actual experience with synchronization as before this there weren't really any cases where the inputs all needed to occur at the same time in order to prevent any errors. I personally had some issues with the adder, but fortunately I was able to find some schematics from the textbook which were extremely useful when actually implementing my logic. Also please note that the clock

module was provided to me meaning that I did not develop the modules myself. The results from the text requested in the writeup will be set as the **final image** of the appendix section, do keep in mind it is very difficult to make sure that all values are shown on the same screen.

Sources:

Cse 100 Lab 4 manual:

https://classes.soe.ucsc.edu/cse100/Spring20/lab/lab4/lab4.html

'Fundamentals of Digital Logic with Verilog design', Stephen Brown and Zvonko Vranesic

Basys 3 reference manual:

https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manual

Appendices:

```
## 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 clki

## 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: 04/23/2020 09:09:49 AM
// Design Name:
// Module Name: EdgeDetector
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module EdgeDetector(
    input D,
    input clk,
    output Q
    );
    wire qm;
    FDRE #(.INIT(1'b0)) Q4_FF (.C(clk), .CE(1'b1), .D(D), .Q(qm));
    assign Q = D & ~qm;

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/23/2020 12:03:33 PM
// Design Name:
// Module Name: countUD4L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module countUD4L(
    input Up,
    input Dw,
    input LD,
    input [3:0] Din,
    input clk,
    output UTC,
    output DTC,
    output [3:0] Q
    );
    wire updown, run;
    wire [3:0] qout, qnot, tin;
    assign run = (Up ^ Dw) | LD;
    assign updown = Dw & ~Up;                                       // assigns
updown so that if down = 1 and up = 0 count down else up
    assign tin[0] = (1'b1 & ~LD) | (LD & (Din[0]^qout[0]));

    assign tin[1] = (~LD & ((updown & qnot[0]) | (~updown & qout[0]))) | (LD &
(Din[1] ^ qout[1]));
    assign tin[2] = (~LD & ((updown & (&qnot[1:0])) | (~updown & (&qout[1:0])))) |
(LD & (Din[2] ^ qout[2]));
    assign tin[3] = (~LD & ((updown & (&qnot[2:0])) | (~updown & (&qout[2:0])))) |
(LD & (Din[3] ^ qout[3]));
    tflop zero (.clk(clk), .t(tin[0]), .enable(run), .q(qout[0]), .notq(qnot[0]));
```

```
    tflop one (.clk(clk), .t(tin[1]), .enable(run), .q(qout[1]), .notq(qnot[1]));
    tflop two (.clk(clk), .t(tin[2]), .enable(run), .q(qout[2]), .notq(qnot[2]));
    tflop three (.clk(clk), .t(tin[3]), .enable(run), .q(qout[3]), .notq(qnot[3]));
    assign Q = qout;
    assign UTC = &(qout[3:0]);
    assign DTC = &(qnot[3:0]);
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/23/2020 07:41:59 AM
// Design Name:
// Module Name: RingCounter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module RingCounter(
    input advance,
    input clk,
    output [3:0] out
    );
    wire one, two, three, four ,invert;
    FDRE #(.INIT(1'b1)) Q0_FF (.C(clk), .CE(advance), .D(four), .Q(one));
    FDRE #(.INIT(1'b0)) Q1_FF (.C(clk), .CE(advance), .D(one), .Q(two));
    FDRE #(.INIT(1'b0)) Q2_FF (.C(clk), .CE(advance), .D(two), .Q(three));
    FDRE #(.INIT(1'b0)) Q3_FF (.C(clk), .CE(advance), .D(three), .Q(four));
    assign out[0] = one;
    assign out[1] = two;
    assign out[2] = three;
    assign out[3] = four;



endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/23/2020 07:59:34 AM
// Design Name:
// Module Name: Selector
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Selector(
    input [3:0] sel,
    input [15:0] N,
    output [3:0] H
    );
    wire [3:0] zero, one, two, three;
    assign zero = {4{sel[0]}} & N[3:0];
    assign one = {4{sel[1]}} & N[7:4];
    assign two = {4{sel[2]}} & N[11:8];
    assign three = {4{sel[3]}} & N[15:12];
    assign H = zero | one | two | three;

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/23/2020 02:54:10 PM
// Design Name:
// Module Name: counterUD16L
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module counterUD16L(
    input [15:0] Din,
    input Up,
    input LD,
    input Dw,
    input clk,
    output [15:0] Q,
    output UTC,
    output DTC
    );
    wire [3:0] over, under;
    countUD4L first(.Up(Up), .Dw(Dw), .LD(LD), .Din(Din[3:0]), .clk(clk),
.UTC(over[0]), .DTC(under[0]), .Q(Q[3:0]));
    countUD4L second(.Up(Up & over[0]), .Dw(Dw & under[0]), .LD(LD), .Din(Din[7:4]),
.clk(clk), .UTC(over[1]), .DTC(under[1]), .Q(Q[7:4]));
    countUD4L third(.Up(Up & (&over[1:0])), .Dw(Dw & (&under[1:0])), .LD(LD),
.Din(Din[11:8]), .clk(clk), .UTC(over[2]), .DTC(under[2]), .Q(Q[11:8]));
    countUD4L fourth(.Up(Up & (&over[2:0])), .Dw(Dw & (&under[2:0])), .LD(LD),
.Din(Din[15:12]), .clk(clk), .UTC(over[3]), .DTC(under[3]), .Q(Q[15:12]));
    assign UTC = &over;
    assign DTC = &under;
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/23/2020 03:42:14 PM
// Design Name:
// Module Name: Top_module
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Top_module(
    input clkin,
    input btnR,
    input btnU,
    input btnD,
    input btnC,
    input btnL,
    input [15:0] sw,
    output [6:0] seg,
    output dp,
    output [3:0] an,
    output [15:0] led
    );
    wire clk, Up, to, Dw, UTC, DTC, digsel, nC, fast;
    wire [3:0] ringer, depict;
    wire [15:0] number;
    lab4_clks slowit (.clkin(clkin), .greset(btnR), .clk(clk), .digsel(digsel),
.fastclk());
    counterUD16L counter (.Din(sw), .Up(Up), .LD(btnL), .Dw(Dw), .clk(clk),
.Q(number), .UTC(UTC), .DTC(DTC));
    EdgeDetector foru (.clk(clk), .D(btnU), .Q(to));
    EdgeDetector fordw (.clk(clk), .D(btnD), .Q(Dw));
    RingCounter ring (.advance(digsel), .clk(clk), .out(ringer));
    Selector sel (.sel(ringer), .N(number), .H(depict));
```

```
    hex7seg display (.e(1'b1), .n(depict), .seg(seg[6:0]));
    FDRE #(.INIT(1'b0)) Q4_FA (.C(clk), .CE(1'b1), .D(btnC), .Q(fast));
    assign nC = fast & ~(&number[15:2]);
    assign Up = to | nC;
    assign an[3:0] = ~ringer;
    assign dp = 1;
    assign led[0] = UTC;
    assign led[15] = DTC;
    assign led[14:1] = 14'b00000000000000;



endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/28/2020 08:05:59 AM
// Design Name:
// Module Name: edgesim
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module edgesim(
);
reg D, clk;
wire Q;

 EdgeDetector edgey(.D(D), .clk(clk), .Q(Q));

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

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

    initial
    begin
        #1100;                      //1100
```

```
        D = 0;
        #100;
        D = 1;
        #100;
        D = 0;
        #100;
        D = 1;
        #100;
        D = 0;
        #100;
        D = 1;
        #100;
    end
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/23/2020 12:37:44 PM
// Design Name:
// Module Name: countsim
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module countsim(

    );
    reg Up, Dw, LD, clkin;
    reg [3:0] Din;
    wire UTC, DTC;
    wire [3:0] Q;
    countUD4L counter(.Up(Up), .Dw(Dw), .LD(LD), .Din(Din), .clk(clkin), .UTC(UTC),
.DTC(DTC), .Q(Q));
    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
        #1100;                      //1100
        LD = 1'b1;
        Up = 1'b0;
        Dw = 1'b0;
        Din = 4'hf;
        #300;                       //1200
        LD = 1'b0;
        Up = 1'b0;
        Dw = 1'b0;
        #300;                       //1300
        Up = 1'b1;
        #300;                        //1600
        Up = 1'b0;
        Dw = 1'b1;
        #100;
        Dw = 1'b0;
        #300;
    end
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 4/11/2020 05:26:52 PM
// Design Name:
// Module Name: show_7segDisplay
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


//wire [7:0] D7Seg3,D7Seg2,D7Seg1,D7Seg0;
//show_7segDisplay  showit (.seg(seg),.dp(dp),.an(an),
//  .D7Seg0(D7Seg0),.D7Seg1(D7Seg1),.D7Seg2(D7Seg2),.D7Seg3(D7Seg3));


module show_7segDisplay (
 input [6:0] seg,
 input dp,
 input [3:0] an,
 output reg [7:0] D7Seg0, D7Seg1, D7Seg2,D7Seg3);

 reg [7:0] val;

 wire AN0, AN1, AN2, AN3;
 assign AN0=an[0];
 assign AN1=an[1];
 assign AN2=an[2];
 assign AN3=an[3];

  always @(AN0 or val)
   begin
            if (AN0 == 0) D7Seg0 <= val;
            else if (AN0 == 1) D7Seg0 <= " ";
            else D7Seg0 <= 8'bX;   //  non-blocking assignment
```

```
 end

always @(AN1 or val)
 begin
          if (AN1 == 0) D7Seg1 <= val;
          else if (AN1 == 1) D7Seg1 <= " ";
          else D7Seg1 <= 8'bX;   //  non-blocking assignment
 end

always @(AN2 or val)
 begin
          if (AN2 == 0) D7Seg2 <= val;
          else if (AN2 == 1) D7Seg2 <= " ";
          else D7Seg2 <= 8'bX;   //  non-blocking assignment
 end

always @(AN3 or val)
 begin
          if (AN3 == 0) D7Seg3 <= val;
          else if (AN3 == 1) D7Seg3 <= " ";
          else D7Seg3 <= 8'bX;   //  non-blocking assignment
 end

  always @(seg)
  case (seg)
  7'b01111111:
       val = "-";
  7'b1111111:
       val = " ";
  7'b1000000:
       val = "0";
  7'b1111001:
       val = "1";
  7'b0100100:
       val = "2";
  7'b0110000:
       val = "3";
  7'b0011001:
       val = "4";
  7'b0010010:
       val = "5";
  7'b0000010:
       val = "6";
  7'b1111000:
       val = "7";
  7'b0000000:
```

```
        val = "8";
    7'b0011000:
        val = "9";
    7'b0001000:
        val = "A";
    7'b0000011:
        val = "B";
    7'b1000110:
        val = "C";
    7'b0100001:
        val = "D";
    7'b0000110:
        val = "E";
    7'b0001110:
        val = "F";
    default:
        val = 8'bX;
    endcase
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/16/2020 09:22:50 AM
// Design Name:
// Module Name: hex7seg
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module hex7seg(
    input e,
    input [3:0] n,
    output [6:0] seg
    );
    wire [7:0] a_in;
    wire [7:0] b_in;
    wire [7:0] c_in;
    wire [7:0] d_in;
    wire [7:0] e_in;
    wire [7:0] f_in;
    wire [7:0] g_in;
    assign a_in = {1'b0, n[0], n[0], 1'b0, 1'b0, ~n[0], 1'b0, n[0]};
    assign b_in = {1'b1, ~n[0], n[0], 1'b0, ~n[0], n[0], 1'b0, 1'b0};
    assign c_in = {1'b1, ~n[0], 1'b0, 1'b0, 1'b0, 1'b0, ~n[0], 1'b0};
    assign d_in = {n[0], 1'b0, ~n[0], n[0], n[0], ~n[0], 1'b0, n[0]};
    assign e_in = {1'b0, 1'b0, 1'b0, n[0], n[0], 1'b1, n[0], n[0]};
    assign f_in = {1'b0, n[0], 1'b0, 1'b0, n[0], 1'b0, 1'b1, n[0]};
    assign g_in = {1'b0, ~n[0], 1'b0, 1'b0, n[0], 1'b0, 1'b0, 1'b1};

    m8_1e ma(.in(a_in), .sel(n[3:1]), .e(e), .o(seg[0]));
    m8_1e mb(.in(b_in), .sel(n[3:1]), .e(e), .o(seg[1]));
    m8_1e mc(.in(c_in), .sel(n[3:1]), .e(e), .o(seg[2]));
    m8_1e md(.in(d_in), .sel(n[3:1]), .e(e), .o(seg[3]));
```

```
    m8_1e me(.in(e_in), .sel(n[3:1]), .e(e), .o(seg[4]));
    m8_1e mf(.in(f_in), .sel(n[3:1]), .e(e), .o(seg[5]));
    m8_1e mg(.in(g_in), .sel(n[3:1]), .e(e), .o(seg[6]));

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/14/2020 09:32:34 AM
// Design Name:
// Module Name: m8_1e
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module m8_1e(
    input [7:0] in,
    input [2:0] sel,
    input e,
    output o
    );
    wire zero, one, two, three, four, five, six, seven;
    assign zero = ~sel[2] & ~sel[1] & ~sel[0];
    assign one = ~sel[2] & ~sel[1] & sel[0];
    assign two = ~sel[2] & sel[1] & ~sel[0];
    assign three = ~sel[2] & sel[1] & sel[0];
    assign four = sel[2] & ~sel[1] & ~sel[0];
    assign five = sel[2] & ~sel[1] & sel[0];
    assign six = sel[2] & sel[1] & ~sel[0];
    assign seven = sel[2] & sel[1] & sel[0];
    assign o = e & (in[0] & zero | in[1] & one | in[2] & two | in[3] & three | in[4]
& four | in[5] & five | in[6] & six | in[7] & seven);
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/24/2020 01:50:14 PM
// Design Name:
// Module Name: sixteenbitsim
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module sixteenbitsim(

);

reg [15:0] Din;
reg Up, LD, Dw, clk;
wire [15:0] Q;
wire UTC, DTC;
counterUD16L count (.Din(Din), .Up(Up), .Dw(Dw), .LD(LD), .clk(clk), .Q(Q),
.UTC(UTC), .DTC(DTC));

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

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

```
    initial
    begin
        #1100;                      //1100
        Up = 1'b0;
        Dw = 1'b1;
        LD = 1'b1;
        Din = 16'hffa0;
        #300;                       //1400
        LD = 1'b0;
        Up = 1'b0;
        #300;                       //1700
        Up = 1;
        Dw = 1;
        #300;
        Dw = 1'b0;
        Up = 1'b1;
        #300;                       //2000
        LD = 1'b1;
        Din = 16'h0000;
        #300;                       //2300
        Din = 16'hffff;
        #300;                       //2600
        LD = 0;
        Up = 1;
        #300;
        Up = 0;
        Dw = 1;
        #700;
    end
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/23/2020 11:06:07 AM
// Design Name:
// Module Name: tflop
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module tflop(
    input t,
    input clk,
    input enable,
    output q,
    output notq
    );
    wire out, nout, din;
    assign din = (t & nout) | (~t & out);
    FDRE #(.INIT(1'b0)) Q5_FF (.C(clk), .CE(enable), .D(din), .Q(out));
    assign nout = ~out;
    assign q = out;
    assign notq = nout;

endmodule
```

```
`timescale 1ns/1ps
// Verilog code to test UTC and DTC of your  16 bit counter



module testTC();
  reg clkin, btnR, btnC, btnU, btnD, btnL;
  reg [15:0] sw;
  wire [15:0] led;
  wire [6:0] seg;
  wire [3:0] an;
  wire dp, UTC, DTC;
  wire [7:0] D7Seg3,D7Seg2,D7Seg1,D7Seg0;

  Top_module
   UUT (
      .clkin(clkin),
      .btnR(btnR),
      .btnU(btnU),
      .btnD(btnD),
      .btnL(btnL),
      .btnC(btnC),
      .sw(sw),
      .seg(seg),
      .dp(dp),
      .led(led),
      .an(an)
      );
  show_7segDisplay  showit (.seg(seg),.dp(dp),.an(an),
              .D7Seg0(D7Seg0),.D7Seg1(D7Seg1),.D7Seg2(D7Seg2),.D7Seg3(D7Seg3));

integer TX_ERROR = 0;
    assign UTC = led[0];
    assign DTC = led[15];

// Run this simulation for 3ms. If correct TX_ERROR should be 0 at the end.
// UTC should be high and then go low at 2,705us and go low at 2,706.3us.

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



    initial    // Clock process for clkin
    begin
      btnC = 1'bx;
```

```
  btnR = 1'b0;
  btnU = 1'bx;
  btnD = 1'bx;
  btnL = 1'bx;

   #OFFSET
    clkin = 1'b1;
   forever
     begin
        #(PERIOD-(PERIOD*DUTY_CYCLE)) clkin = ~clkin;
     end
  end

initial
begin
 #2000;
 btnC=1'b0;
 btnU=1'b0;
 btnD=1'b0;
 btnL=1'b0;
 btnR=1'b0;
 sw = 16'h9034;
 #200;
 btnR = 1'b1;
 #500 btnR = 1'b0;
 #300 btnC=1'b1;
 btnR = 1'b0;
 #2640000 btnC=1'b0;
 #200; btnU=1'b1;
 #600; btnU=1'b0;
 #200; btnU=1'b1;
 #700; btnU=1'b0;
 #200; btnU=1'b1;
 #800; btnU=1'b0;
 CHECK_UTC(1'b1); CHECK_DTC(1'b0);
 #400;
 CHECK_UTC(1'b1); CHECK_DTC(1'b0);
 #200;
 btnU=1'b1;
 #200;
 btnU=1'b0;
 #100;
 CHECK_UTC(1'b0); CHECK_DTC(1'b1);
 #200; btnU=1'b0;
 #200; btnU=1'b1;
 #100;
```

```
    CHECK_UTC(1'b0);  CHECK_DTC(1'b0);
    #300 btnD=1'b1;
    #200;
    CHECK_UTC(1'b0);  CHECK_DTC(1'b1);
    #200; btnD=1'b0;
    #200; btnD=1'b1;
    #100;
    CHECK_UTC(1'b1); CHECK_DTC(1'b0);
    #200; btnD=1'b0;
    #200; btnD=1'b1;
    #100;
    CHECK_UTC(1'b0); CHECK_DTC(1'b0);
    #200; btnU=1'b0;
    #200; btnU=1'b1;
    #100;
    CHECK_UTC(1'b1); CHECK_DTC(1'b0);
    #200; btnU=1'b0;
    #200; btnU=1'b1;
    #100;
    CHECK_UTC(1'b0); CHECK_DTC(1'b1);
    #100;

   end

   task CHECK_UTC;
       input good_TC;

       #0 begin
           if (good_TC !== UTC) begin
               TX_ERROR = TX_ERROR + 1;
           end
       end
   endtask

   task CHECK_DTC;
       input good_TC;

      #0 begin
          if (good_TC !== DTC) begin
              TX_ERROR = TX_ERROR + 1;
          end
      end
   endtask

endmodule
```

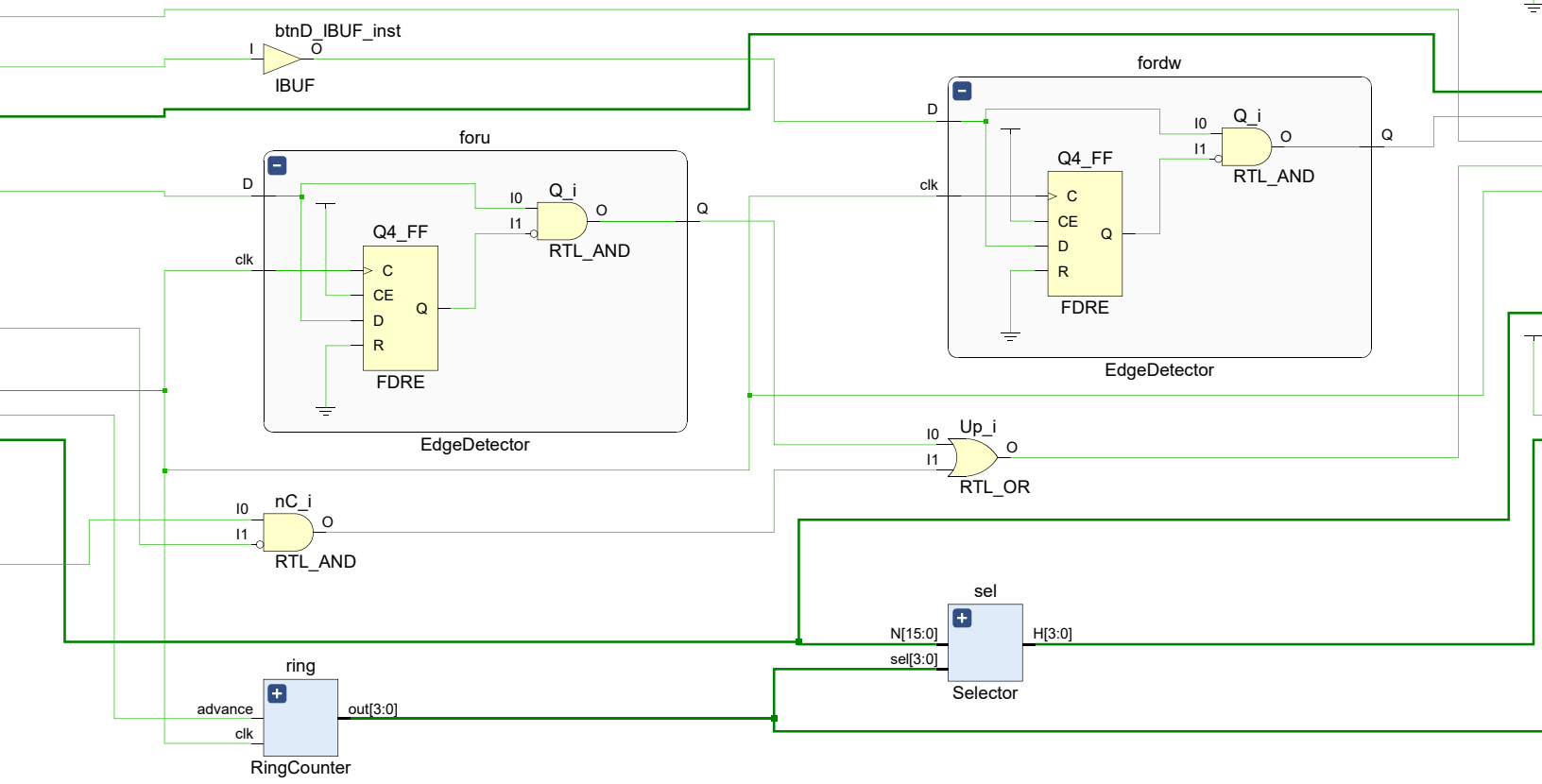
```
module show_7segDisplay (
 input [6:0] seg,
 input dp,
 input [3:0] an,
 output reg [7:0] D7Seg0, D7Seg1, D7Seg2,D7Seg3);

 reg [7:0] val;

 wire AN0, AN1, AN2, AN3;
 assign AN0=an[0];
 assign AN1=an[1];
 assign AN2=an[2];
 assign AN3=an[3];

  always @(AN0 or val)
   begin
            if (AN0 == 0) D7Seg0 <= val;
            else if (AN0 == 1) D7Seg0 <= " ";
            else D7Seg0 <= 8'bX;   //  non-blocking assignment
   end

  always @(AN1 or val)
   begin
            if (AN1 == 0) D7Seg1 <= val;
            else if (AN1 == 1) D7Seg1 <= " ";
            else D7Seg1 <= 8'bX;   //  non-blocking assignment
   end

  always @(AN2 or val)
   begin
            if (AN2 == 0) D7Seg2 <= val;
            else if (AN2 == 1) D7Seg2 <= " ";
            else D7Seg2 <= 8'bX;   //  non-blocking assignment
   end

  always @(AN3 or val)
   begin
            if (AN3 == 0) D7Seg3 <= val;
            else if (AN3 == 1) D7Seg3 <= " ";
            else D7Seg3 <= 8'bX;   //  non-blocking assignment
   end

    always @(seg)
    case (seg)
    7'b01111111:
         val = "-";
```

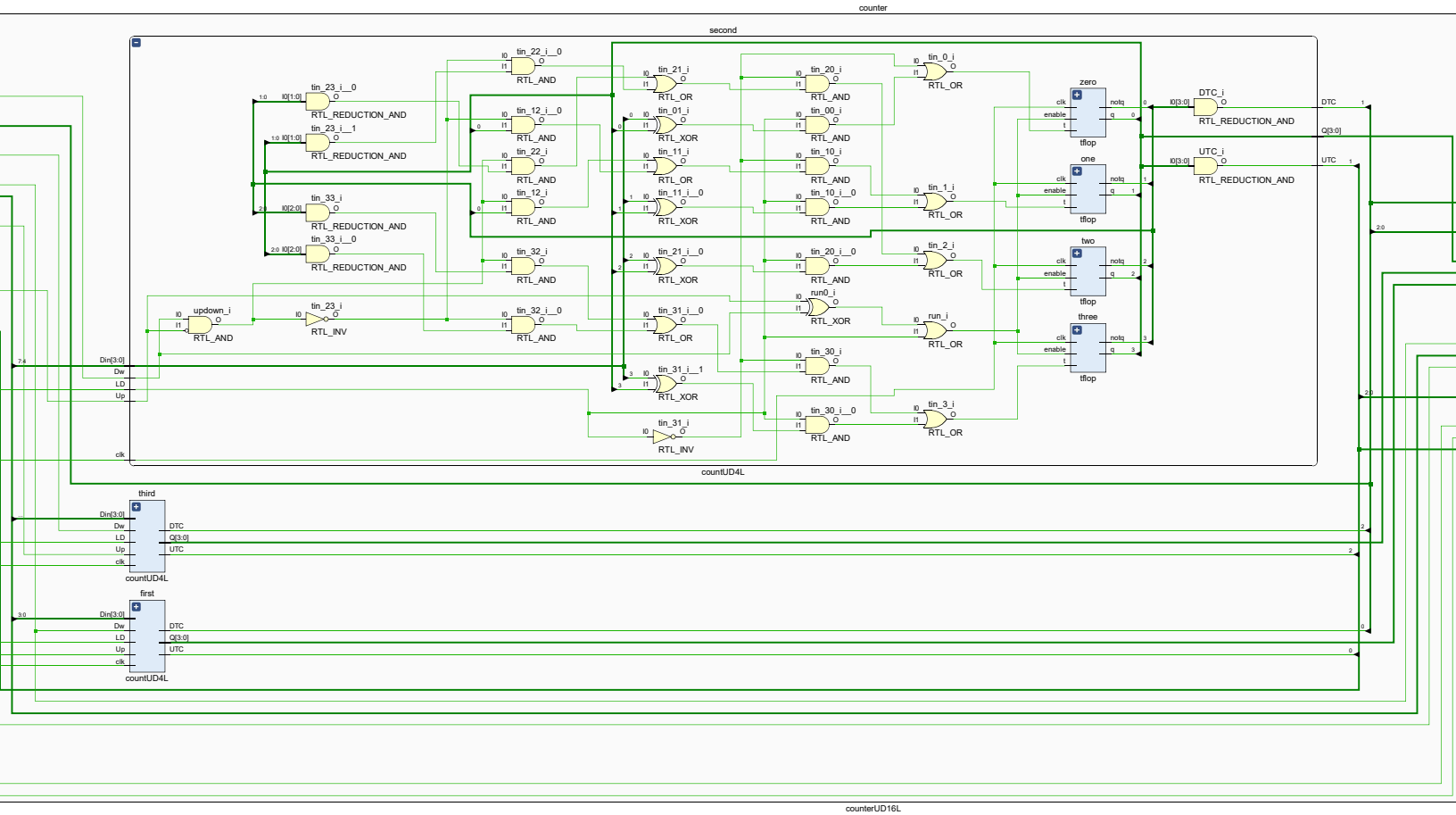
```
    7'b1111111:
         val = " ";
    7'b1000000:
         val = "0";
    7'b1111001:
         val = "1";
    7'b0100100:
         val = "2";
    7'b0110000:
         val = "3";
    7'b0011001:
         val = "4";
    7'b0010010:
         val = "5";
    7'b0000010:
         val = "6";
    7'b1111000:
         val = "7";
    7'b0000000:
         val = "8";
    7'b0011000:
         val = "9";
    7'b0001000:
         val = "A";
    7'b0000011:
         val = "B";
    7'b1000110:
         val = "C";
    7'b0100001:
         val = "D";
    7'b0000110:
         val = "E";
    7'b0001110:
         val = "F";
    default:
         val = 8'bX;
    endcase
endmodule
```

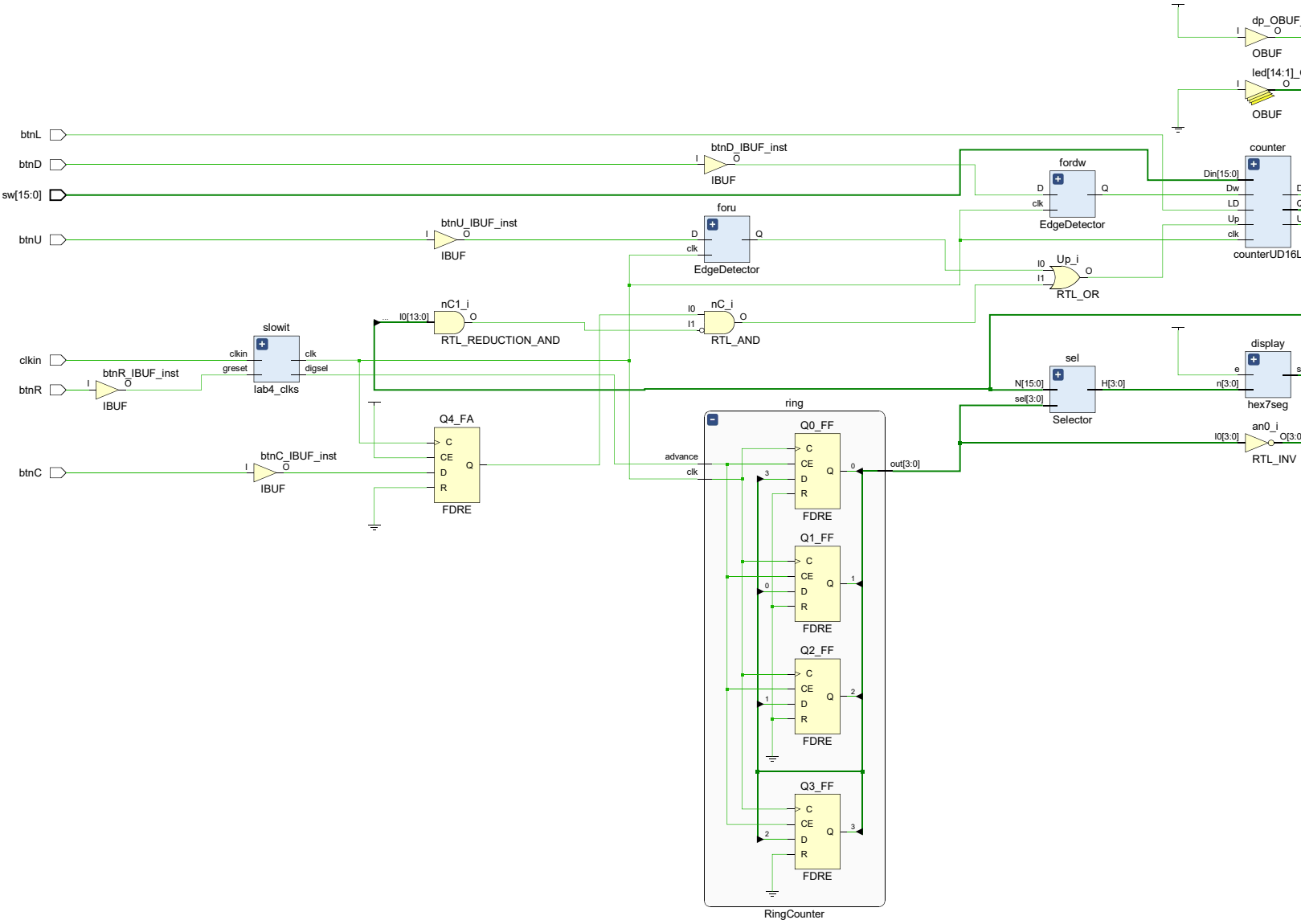
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/23/2020 04:29:32 PM
// Design Name:
// Module Name: top_simulation
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module top_simulation(

    );
    reg clkin, btnR, btnU, btnD, btnC, btnL;
    reg [15:0] sw;
    wire [6:0] seg;
    wire dp;
    wire [3:0] an;
    wire [15:0] led;
     parameter PERIOD = 10;
    parameter real DUTY_CYCLE = 0.5;
    parameter OFFSET = 2;
    wire [7:0] D7Seg3,D7Seg2,D7Seg1,D7Seg0; // Change the Radix of these signals to
ASCII
    show_7segDisplay  showit (.seg(seg),.dp(dp),.an(an),
     .D7Seg0(D7Seg0),.D7Seg1(D7Seg1),.D7Seg2(D7Seg2),.D7Seg3(D7Seg3));
     Top_module counter(.clkin(clkin), .btnR(btnR), .btnU(btnU), .btnD(btnD),
.btnC(btnC), .btnL(btnL), .sw(sw), .seg(seg), .dp(dp), .an(an), .led(led));
    initial    // Clock process for clkin
    begin
        #OFFSET
          clkin = 1'b1;
        forever
        begin
```

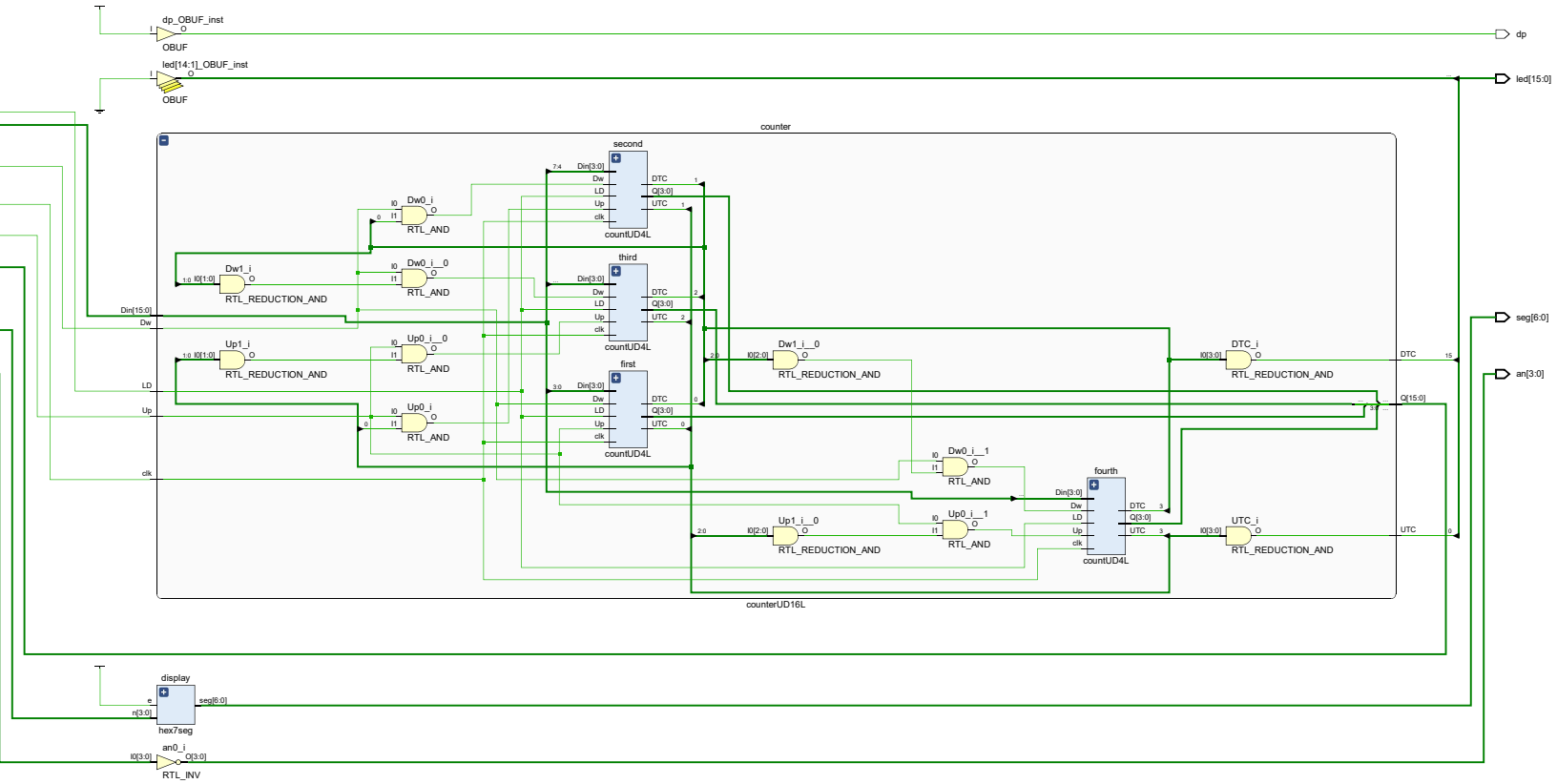
```
        #(PERIOD-(PERIOD*DUTY_CYCLE)) clkin = ~clkin;
    end
 end

initial
begin
      btnR = 1'b0;
      #1000;
      btnU = 1'b0;
      btnD = 1'b0;
      btnC = 1'b0;
      btnL = 1'b1;
      sw = 16'hfff0;
      #300;
      btnL = 0;
      #300;
      btnU = 1'b1;
      #100;
      btnU = 1'b0;
      btnD = 1'b1;
      #100;
      btnD = 1'b0;
      btnC = 1'b1;
      #500;
      btnU = 1'b1;
      #100;
      btnU = 1'b0;
      #100;
       btnU = 1'b1;
      #100;
      btnU = 1'b0;
      #100;
       btnU = 1'b1;
      #100;
      btnU = 1'b0;
      #100;
       btnU = 1'b1;
      #100;
      btnU = 1'b0;
      #100;
      btnU = 1'b1;
      #100;
      btnU = 1'b0;
      #100;

 end
```

```
endmodule
```

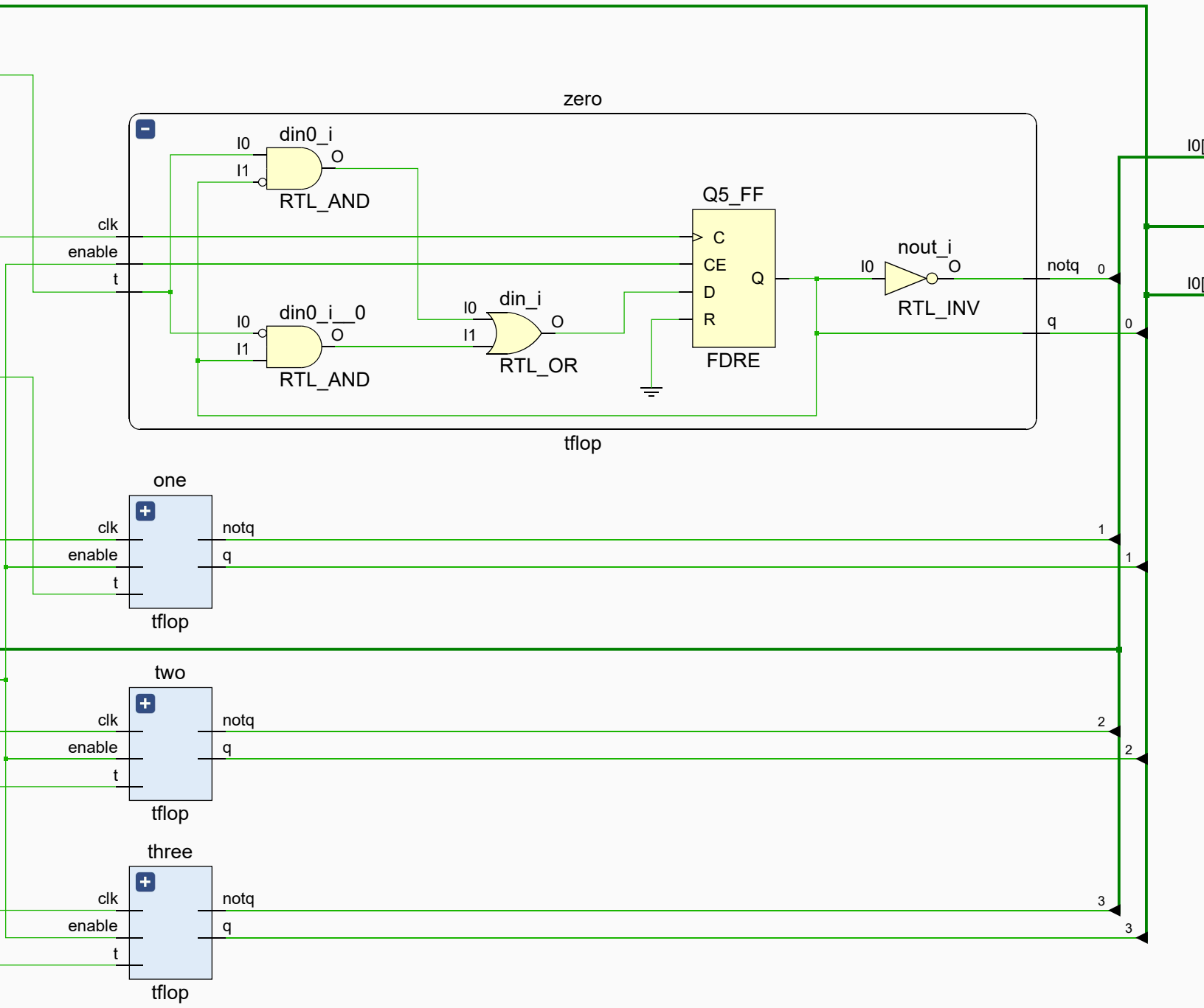

btnD_IBUF_inst
I
O
IBUF
foru
D
clk
Q4_FF
C
CE
D
R
Q
FDRE
I0
I1
Q_i
O
RTL_AND
Q
EdgeDetector
fordw
D
clk
Q4_FF
C
CE
D
R
Q
FDRE
I0
I1
Q_i
O
RTL_AND
Q
EdgeDetector
I0
I1
Up_i
O
RTL_OR
I0
I1
nC_i
O
RTL_AND
sel
N[15:0]
sel[3:0]
H[3:0]
Selector
ring
advance
clk
out[3:0]
RingCounter

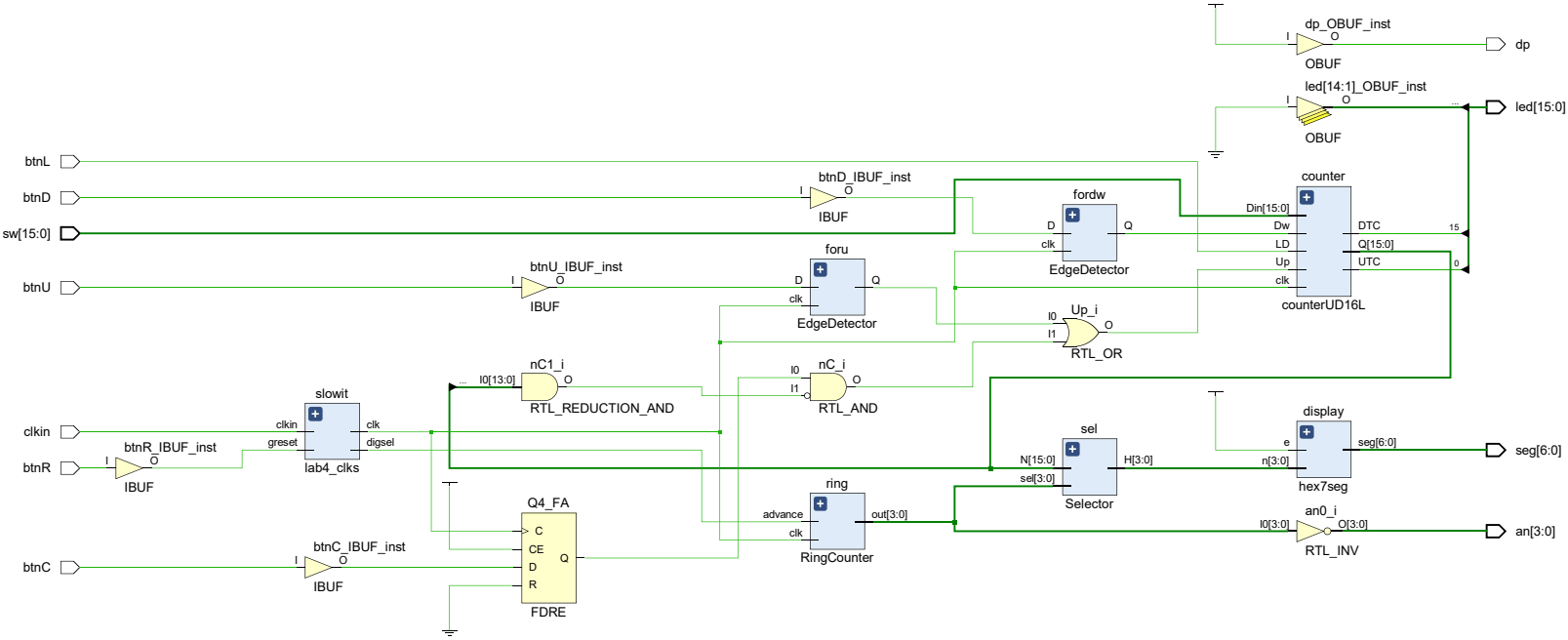

counter
second
tin_22_i__0
RTL_AND
tin_23_i__0
RTL_REDUCTION_AND
tin_12_i__0
RTL_AND
tin_23_i__1
RTL_REDUCTION_AND
tin_22_i
RTL_AND
tin_33_i
RTL_REDUCTION_AND
tin_12_i
RTL_AND
tin_33_i__0
RTL_REDUCTION_AND
tin_32_i
RTL_AND
updown_i
RTL_AND
tin_23_i
RTL_INV
tin_32_i__0
RTL_AND
tin_21_i
RTL_OR
tin_01_i
RTL_XOR
tin_11_i
RTL_OR
tin_11_i__0
RTL_XOR
tin_21_i__0
RTL_XOR
tin_31_i__0
RTL_OR
tin_31_i__1
RTL_XOR
tin_31_i
RTL_INV
tin_20_i
RTL_AND
tin_00_i
RTL_AND
tin_10_i
RTL_AND
tin_10_i__0
RTL_AND
tin_20_i__0
RTL_AND
run0_i
RTL_XOR
tin_30_i
RTL_AND
tin_30_i__0
RTL_AND
tin_0_i
RTL_OR
tin_1_i
RTL_OR
tin_2_i
RTL_OR
run_i
RTL_OR
tin_3_i
RTL_OR
zero
one
two
three
tflop
DTC_i
RTL_REDUCTION_AND
UTC_i
RTL_REDUCTION_AND
Din[3:0]
Dw
LD
Up
clk
DTC
Q[3:0]
UTC
countUD4L
third
first
countUD4L
countUD4L
counterUD16L

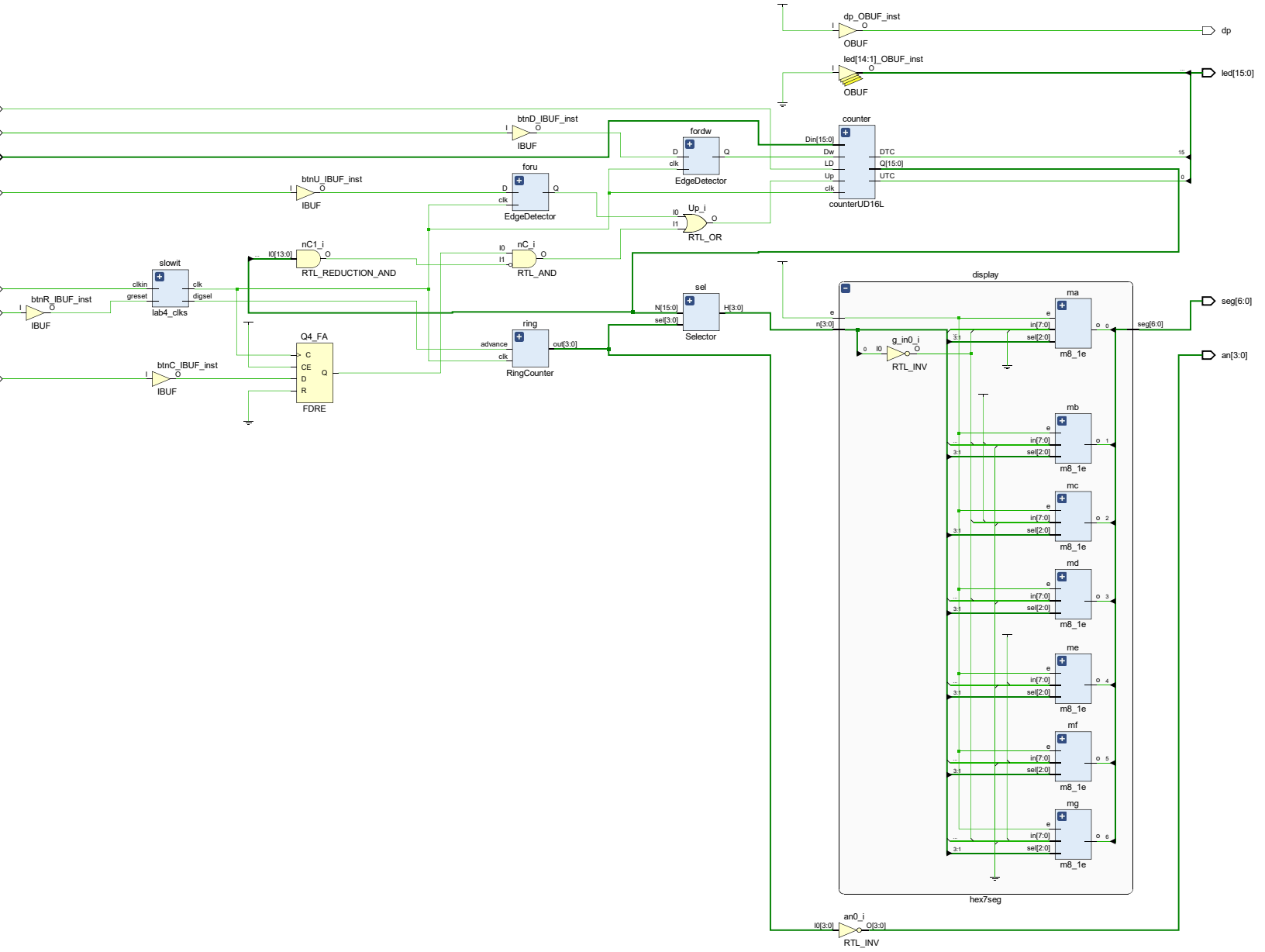

dp_OBUF_
OBUF
led[14:1]_
OBUF
btnL
btnD
sw[15:0]
btnU
clkin
btnR
btnC
IBUF
btnD_IBUF_inst
btnU_IBUF_inst
btnR_IBUF_inst
btnC_IBUF_inst
slowit
clkin
greset
clk
digsel
lab4_clks
nC1_i
I0[13:0]
RTL_REDUCTION_AND
nC_i
I0
I1
RTL_AND
foru
D
clk
Q
EdgeDetector
fordw
EdgeDetector
Up_i
RTL_OR
counter
Din[15:0]
Dw
LD
Up
clk
counterUD16L
Q4_FA
C
CE
D
R
FDRE
ring
advance
clk
out[3:0]
Q0_FF
Q1_FF
Q2_FF
Q3_FF
RingCounter
sel
N[15:0]
sel[3:0]
H[3:0]
Selector
display
e
n[3:0]
hex7seg
an0_i
I0[3:0]
O[3:0]
RTL_INV

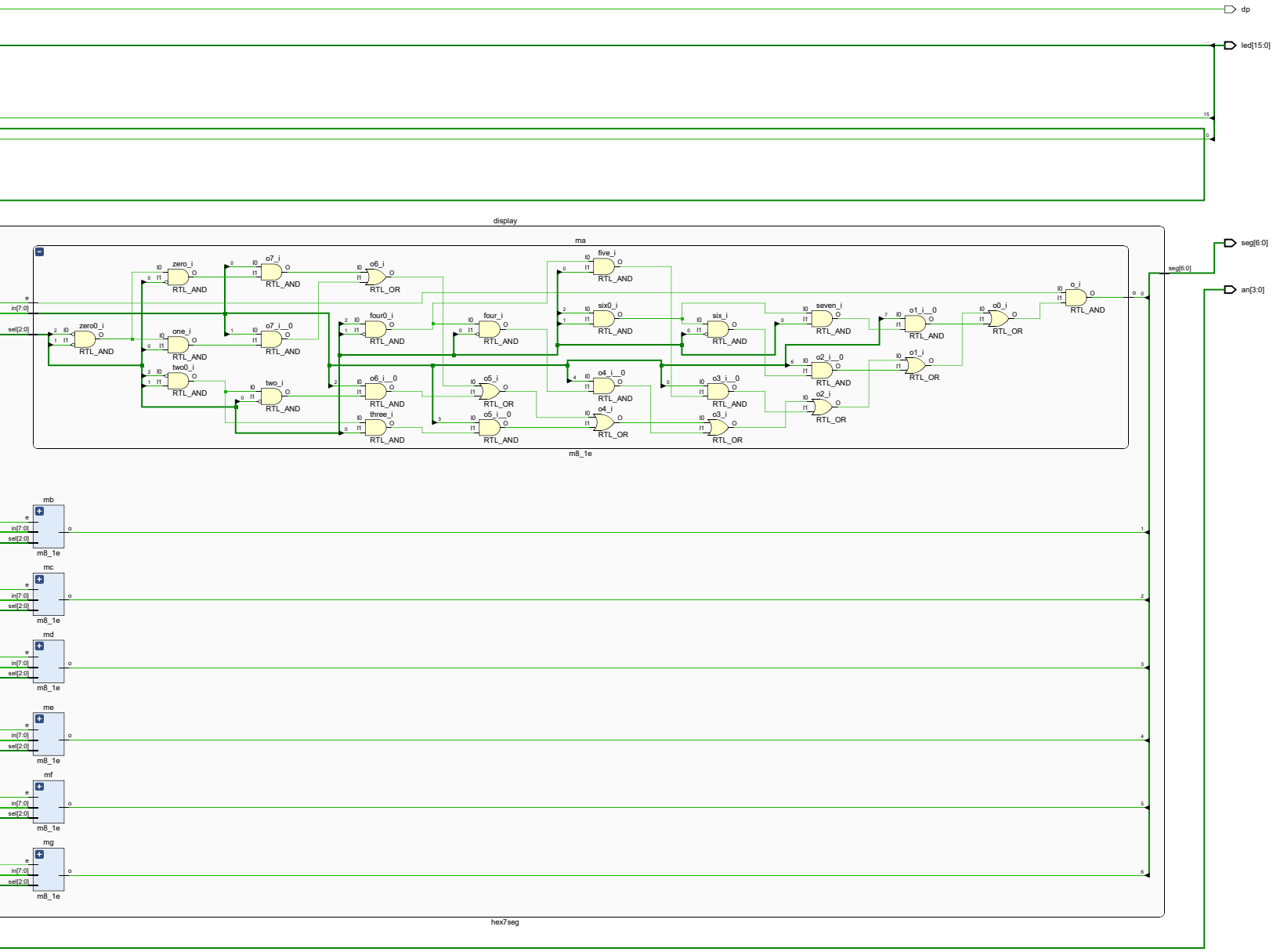

dp_OBUF_inst
OBUF
dp
led[14:1]_OBUF_inst
OBUF
led[15:0]
counter
second
countUD4L
third
countUD4L
first
countUD4L
fourth
countUD4L
Dw0_i
RTL_AND
Dw0_i__0
RTL_AND
Dw1_i
RTL_REDUCTION_AND
Up0_i_0
RTL_AND
Up0_i
RTL_AND
Up1_i
RTL_REDUCTION_AND
Dw1_i__0
RTL_REDUCTION_AND
DTC_i
RTL_REDUCTION_AND
Dw0_i__1
RTL_AND
Up0_i__1
RTL_AND
Up1_i__0
RTL_REDUCTION_AND
UTC_i
RTL_REDUCTION_AND
Din[15:0]
Dw
LD
Up
clk
DTC
Q[15:0]
UTC
counterUD16L
seg[6:0]
an[3:0]
display
hex7seg
an0_i
RTL_INV

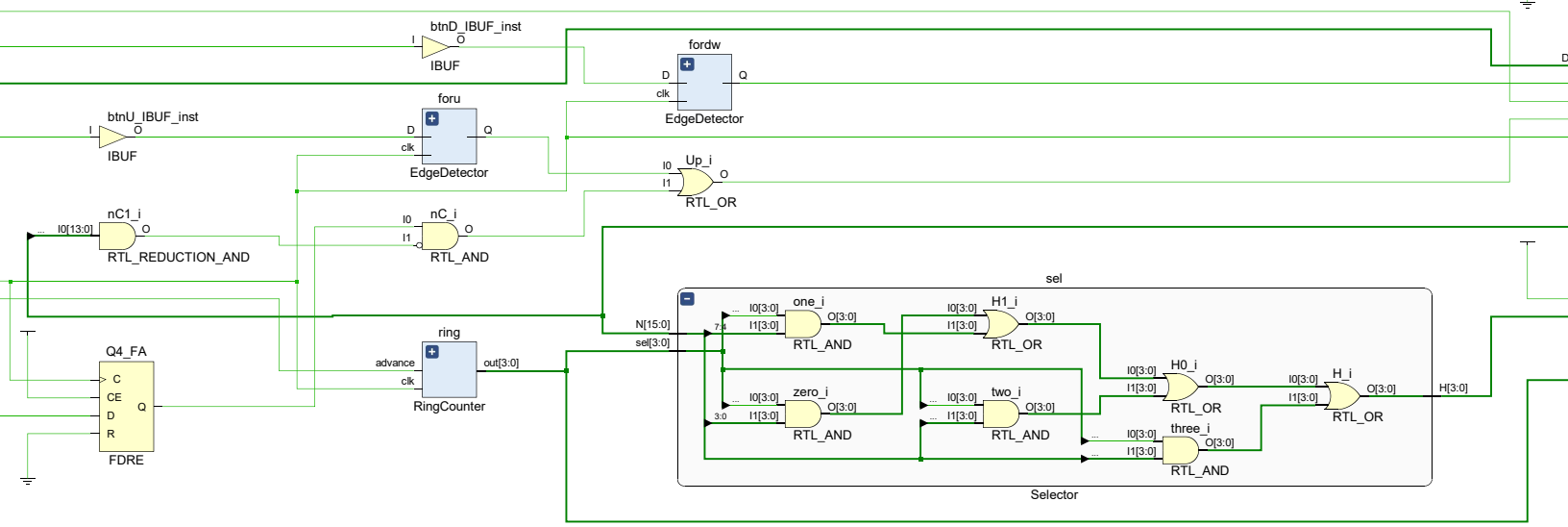
counter

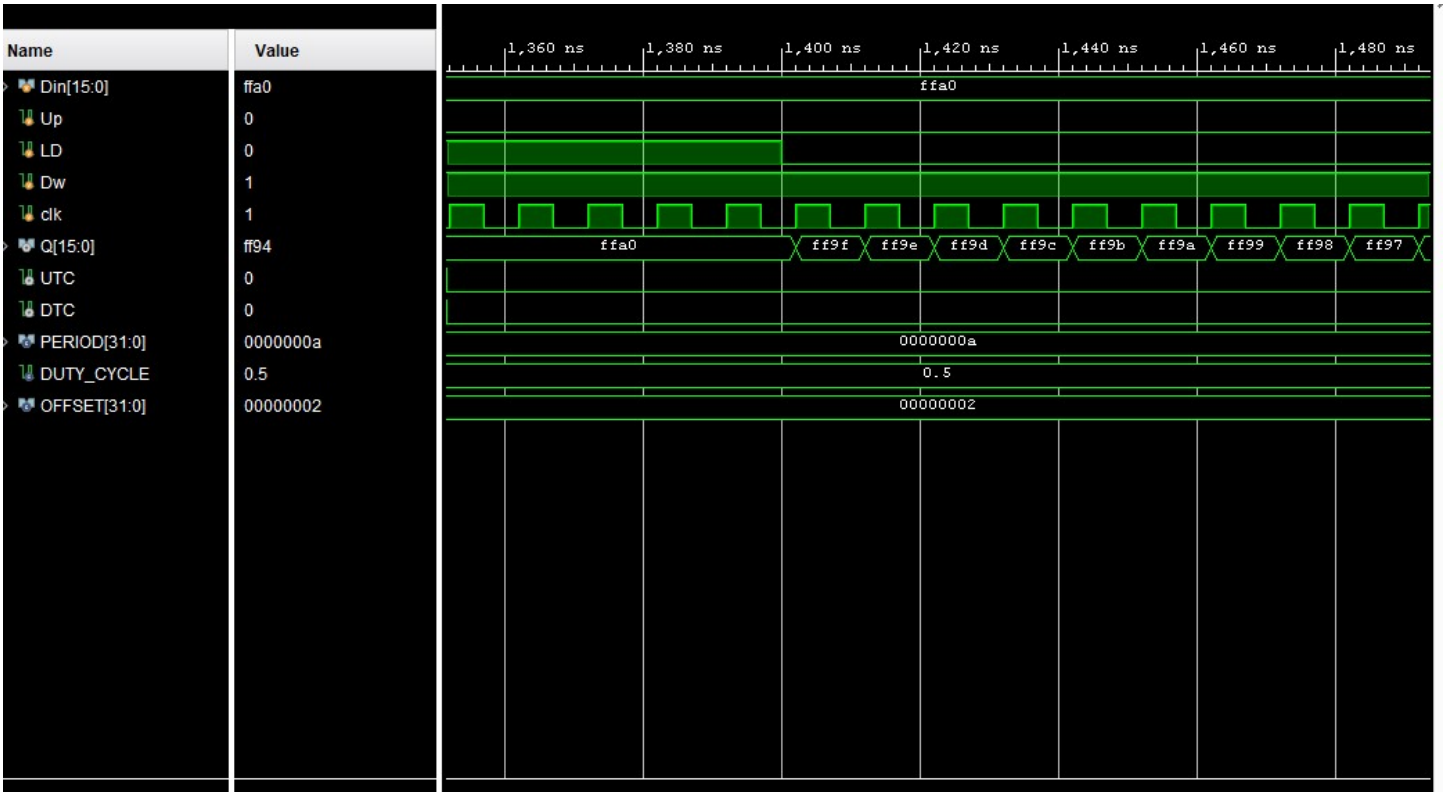

zero
din0_i
I0
I1
O
RTL_AND
Q5_FF
C
CE
D
R
Q
FDRE
nout_i
RTL_INV
notq
q
clk
enable
t
din0_i__0
din_i
RTL_OR
tflop
one
two
three


dp_OBUF_inst
OBUF
dp
led[14:1]_OBUF_inst
OBUF
led[15:0]
btnL
btnD
sw[15:0]
btnU
clkin
btnR
btnC
btnD_IBUF_inst
IBUF
btnU_IBUF_inst
IBUF
btnR_IBUF_inst
IBUF
btnC_IBUF_inst
IBUF
fordw
EdgeDetector
foru
EdgeDetector
counter
counterUD16L
Din[15:0]
Dw
LD
Up
clk
DTC
Q[15:0]
UTC
15
0
Up_i
RTL_OR
nC1_i
RTL_REDUCTION_AND
I0[13:0]
nC_i
RTL_AND
slowit
lab4_clks
clkin
greset
clk
digsel
Q4_FA
FDRE
C
CE
D
R
Q
ring
RingCounter
advance
clk
out[3:0]
sel
Selector
N[15:0]
sel[3:0]
H[3:0]
display
hex7seg
e
n[3:0]
seg[6:0]
an0_i
RTL_INV
I0[3:0]
O[3:0]
an[3:0]

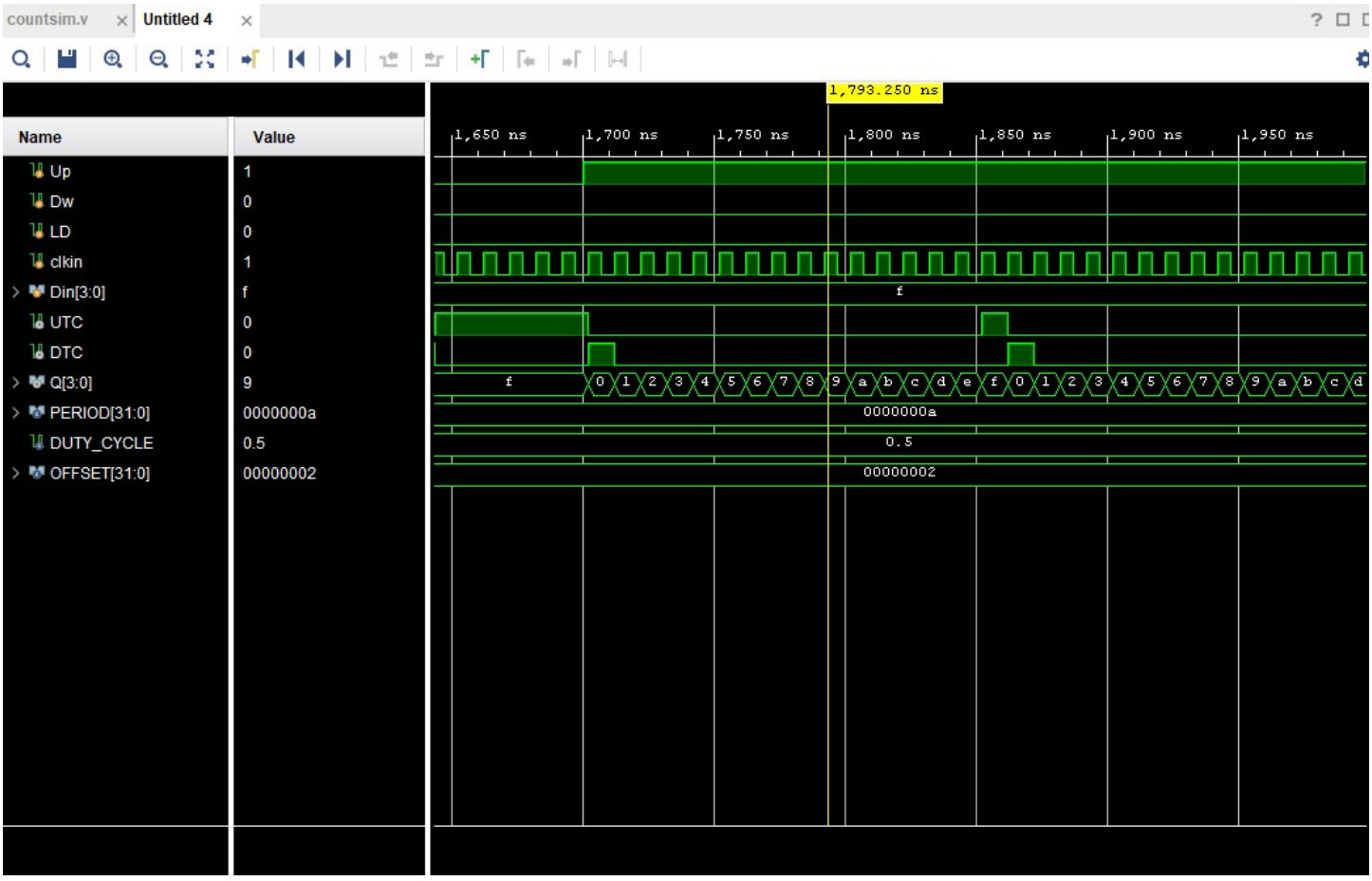

dp_OBUF_inst
OBUF
dp
led[14:1]_OBUF_inst
OBUF
led[15:0]
btnD_IBUF_inst
IBUF
fordw
EdgeDetector
counter
counterUD16L
btnU_IBUF_inst
IBUF
foru
EdgeDetector
Up_i
RTL_OR
nC1_i
RTL_REDUCTION_AND
nC_i
RTL_AND
slowit
lab4_clks
btnR_IBUF_inst
IBUF
sel
Selector
display
ma
m8_1e
seg[6:0]
an[3:0]
g_in0_i
RTL_INV
mb
mc
md
me
mf
mg
hex7seg
Q4_FA
FDRE
btnC_IBUF_inst
IBUF
ring
RingCounter
an0_i
RTL_INV


dp
led[15:0]
15
0
display
ma
e
in[7:0]
sel[2:0]
zero0_i
RTL_AND
zero_i
RTL_AND
one_i
RTL_AND
two0_i
RTL_AND
o7_i
RTL_AND
o7_i__0
RTL_AND
two_i
RTL_AND
o6_i
RTL_OR
four0_i
RTL_AND
o6_i__0
RTL_AND
three_i
RTL_AND
four_i
RTL_AND
o5_i
RTL_OR
o5_i__0
RTL_AND
five_i
RTL_AND
six0_i
RTL_AND
o4_i__0
RTL_AND
o4_i
RTL_OR
six_i
RTL_AND
o3_i__0
RTL_AND
o3_i
RTL_OR
seven_i
RTL_AND
o2_i__0
RTL_AND
o2_i
RTL_OR
o1_i__0
RTL_AND
o1_i
RTL_OR
o0_i
RTL_OR
o_i
RTL_AND
m8_1e
seg[6:0]
an[3:0]
mb
mc
md
me
mf
mg
m8_1e
hex7seg


btnD_IBUF_inst
I
O
IBUF
fordw
D
clk
Q
EdgeDetector
btnU_IBUF_inst
I
O
IBUF
foru
D
clk
Q
EdgeDetector
I0
I1
Up_i
O
RTL_OR
nC1_i
I0[13:0]
O
RTL_REDUCTION_AND
I0
I1
nC_i
O
RTL_AND
sel
N[15:0]
sel[3:0]
7:4
3:0
I0[3:0]
I1[3:0]
one_i
O[3:0]
RTL_AND
I0[3:0]
I1[3:0]
H1_i
O[3:0]
RTL_OR
I0[3:0]
I1[3:0]
zero_i
O[3:0]
RTL_AND
I0[3:0]
I1[3:0]
two_i
O[3:0]
RTL_AND
I0[3:0]
I1[3:0]
H0_i
O[3:0]
RTL_OR
I0[3:0]
I1[3:0]
three_i
O[3:0]
RTL_AND
I0[3:0]
I1[3:0]
H_i
O[3:0]
RTL_OR
H[3:0]
Selector
Q4_FA
C
CE
D
R
Q
FDRE
ring
advance
clk
out[3:0]
RingCounter


Name
Value
Din[15:0]
ffa0
Up
0
LD
0
Dw
1
clk
1
Q[15:0]
ff94
UTC
0
DTC
0
PERIOD[31:0]
0000000a
DUTY_CYCLE
0.5
OFFSET[31:0]
00000002
1,360 ns
1,380 ns
1,400 ns
1,420 ns
1,440 ns
1,460 ns
1,480 ns
ffa0
ffa0
ff9f
ff9e
ff9d
ff9c
ff9b
ff9a
ff99
ff98
ff97
0000000a
0.5
00000002

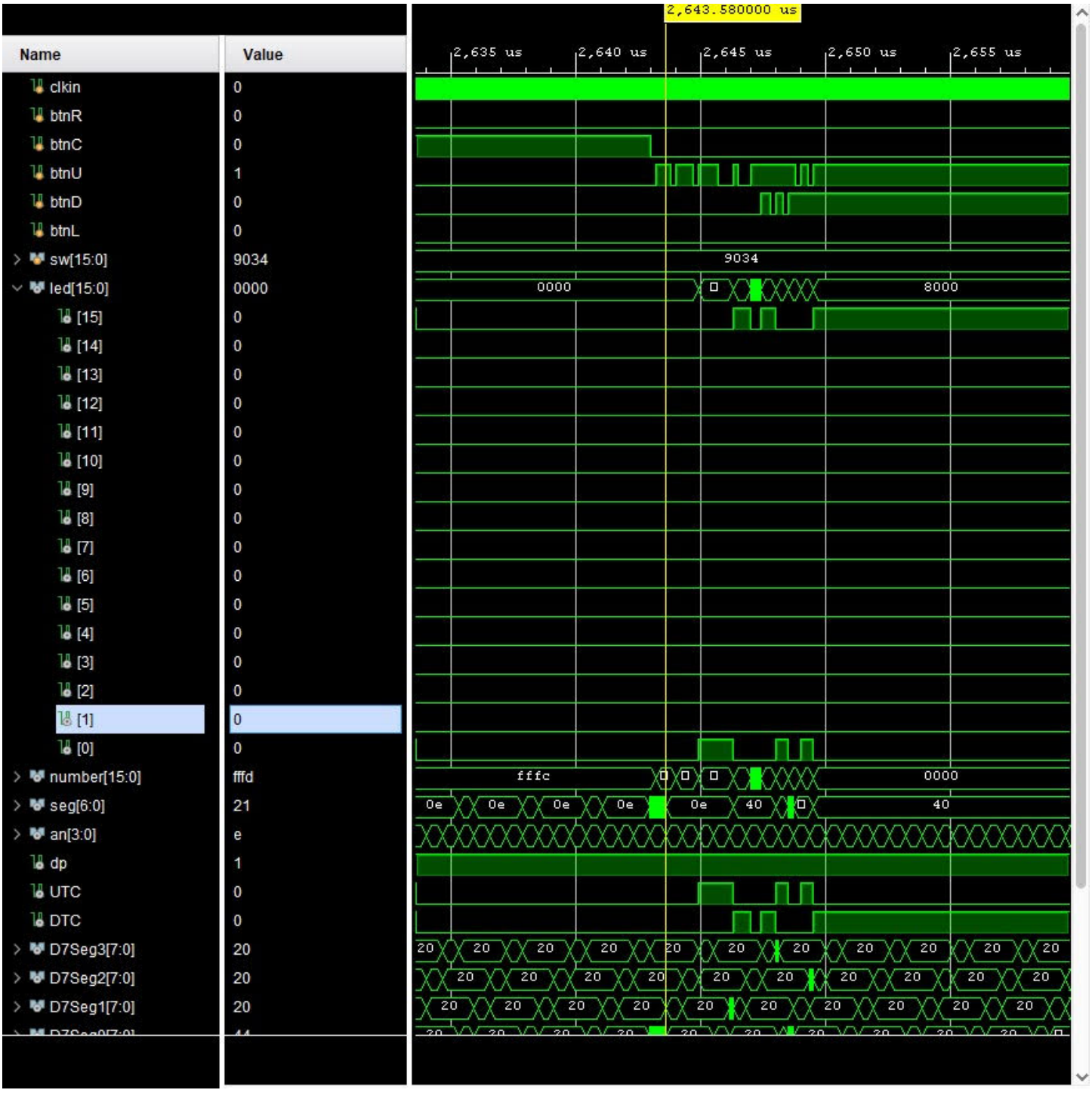

Name
Value
D
1
clk
0
Q
0
PERIOD[31:0]
0000000a
DUTY_CYCLE
0.5
OFFSET[31:0]
00000002
1,100 ns
1,200 ns
1,300 ns
1,400 ns
1,500 ns
1,600 ns
0000000a
0.5
00000002


countsim.v
Untitled 4
Name
Value
Up
1
Dw
0
LD
0
clkin
1
Din[3:0]
f
UTC
0
DTC
0
Q[3:0]
9
PERIOD[31:0]
0000000a
DUTY_CYCLE
0.5
OFFSET[31:0]
00000002
1,793.250 ns
1,650 ns
1,700 ns
1,750 ns
1,800 ns
1,850 ns
1,900 ns
1,950 ns
f
0000000a
0.5
00000002


Name
Value
Din[15:0]
ffff
Up
1
LD
1
Dw
0
clk
0
Q[15:0]
ffff
UTC
1
DTC
0
PERIOD[31:0]
0000000a
DUTY_CYCLE
0.5
OFFSET[31:0]
00000002
1,000 ns
1,500 ns
2,000 ns
2,500 ns
3,000 ns
3,500 ns
4,000
XXXX
ffa0
0000
ffff
0000
ffa0
ff82
0000
ffff
0000000a
0.5
00000002


Name
Value
clkin
1
btnR
0
btnU
1
btnD
0
btnC
0
btnL
0
sw[15:0]
fff0
seg[6:0]
40
number[15:0]
fff0
dp
1
an[3:0]
e
led[15:0]
0000
D7Seg3[7:0]
D7Seg2[7:0]
D7Seg1[7:0]
D7Seg0[7:0]
0
PERIOD[31:0]
0000000a
DUTY_CYCLE
0.5
OFFSET[31:0]
00000002
1,603.950 ns
1,000 ns
1,200 ns
1,400 ns
1,600 ns
1,800 ns
2,000 ns
2,200 ns
fff0
40
0e
0000
fff0
fff1
fff0
e
d
b
8000
0000
F
F
0
0000000a
0.5
00000002


Name
Value
clkin
1
btnR
0
btnU
1
btnD
0
btnC
0
btnL
0
sw[15:0]
fff0
seg[6:0]
40
number[15:0]
fff0
dp
1
an[3:0]
e
led[15:0]
0000
D7Seg3[7:0]
D7Seg2[7:0]
D7Seg1[7:0]
D7Seg0[7:0]
0
PERIOD[31:0]
0000000a
DUTY_CYCLE
0.5
OFFSET[31:0]
00000002
1.603950 us
0 us
1 us
2 us
3 us
4 us


2,643.580000 us
2,635 us
2,640 us
2,645 us
2,650 us
2,655 us
Name
Value
clkin 0
btnR 0
btnC 0
btnU 1
btnD 0
btnL 0
sw[15:0] 9034
led[15:0] 0000
[15] 0
[14] 0
[13] 0
[12] 0
[11] 0
[10] 0
[9] 0
[8] 0
[7] 0
[6] 0
[5] 0
[4] 0
[3] 0
[2] 0
[1] 0
[0] 0
number[15:0] fffd
seg[6:0] 21
an[3:0] e
dp 1
UTC 0
DTC 0
D7Seg3[7:0] 20
D7Seg2[7:0] 20
D7Seg1[7:0] 20
9034
0000
8000
fffc
0000
0e
40
20