Challenge:

1.Design a synchronous decimal counter using verilog.

inputs :start, stop , load, reset, clock,updownmode
outputs : count value,roll over

all inputs are asynchronous in nature.

conditions 1: on appilcation of reset the output should
be (00d) in up mode and (99d) in down mode

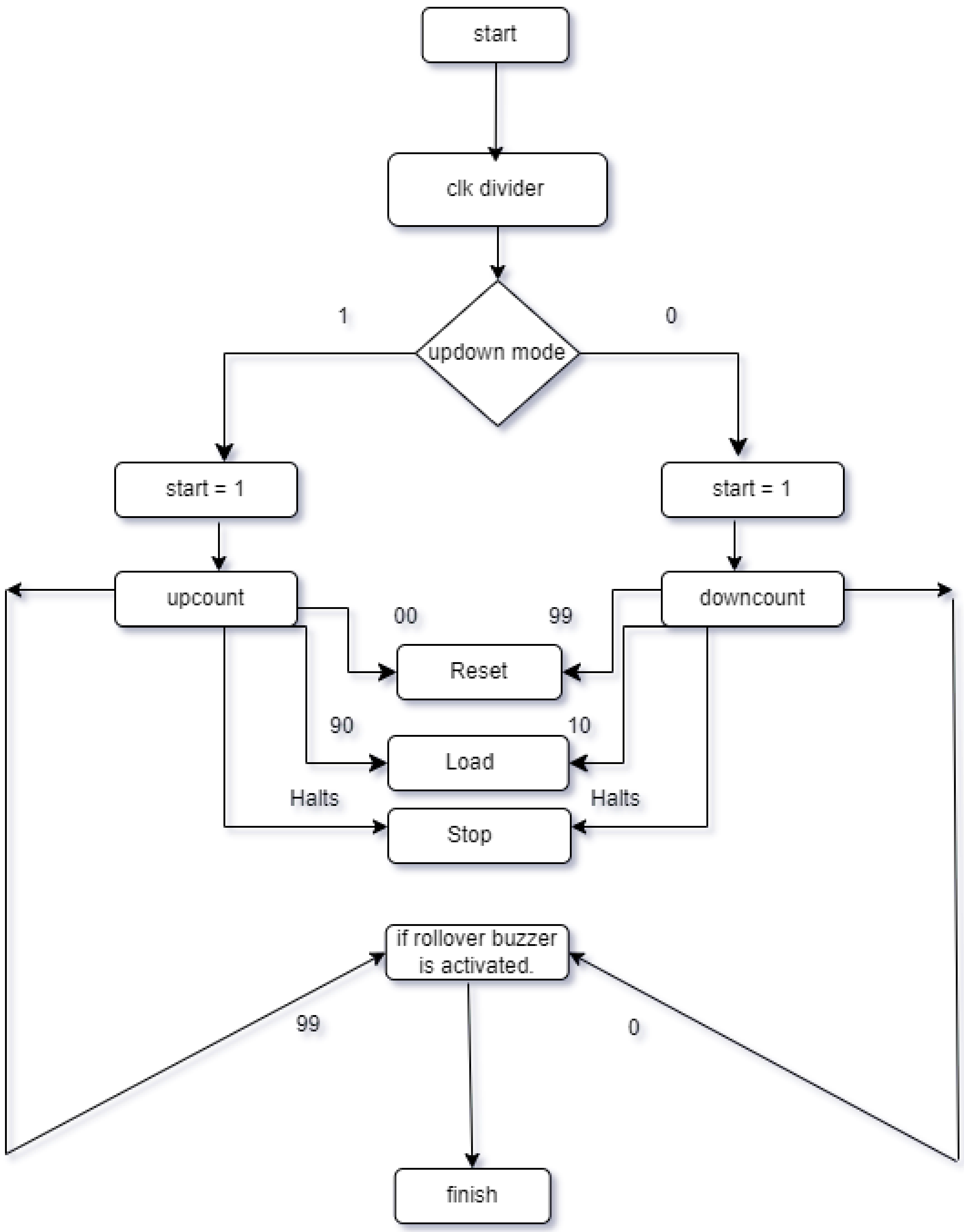
condition 2 : on application of load the count value must be (90d) in up mode 10 d in down mode
condition 3 : on application of start the output should be increment in up mode and decrement in down mode

condition 4 :on application of stop the output should be the counting operation resumes from where it left

condition 5: the counting must happen for every one second
hint (clock is 50 MHz available in FPGA f=1/t)

condition 6 : on every roll over the buzzer must get active to indicate the roll over
hint(keep the roll over signal active for one sec to hear its voice)
conditions are (99d) to (00d) in upmode and (00d) to (99d) in downmode
pls come up with precise verilog code

Flow Chart



start
clk divider
updown mode
1
0
start = 1
start = 1
upcount
downcount
00
99
Reset
90
10
Load
Halts
Halts
Stop
if rollover buzzer is activated.
99
0
finish

Algorithm

step 1 : Start the procedure by declaring the Module and required Ports for the design.

step 2 : Go ahead with the Clock Division statements which are necessary for reducing the 50Mhz inbuilt clock frequency.

step 3 : Take the input from the User as Up count or Down count, if the UpDown bit is '1' then go for Up count .i.e Increment the Counter
Check for the Start and Stop Condition, also Check for reset and Load.

If the UpDown bit is '0' then go for Down count .i.e Decrement the Counter.
Check for the Start and Stop Condition, also Check for reset and Load.

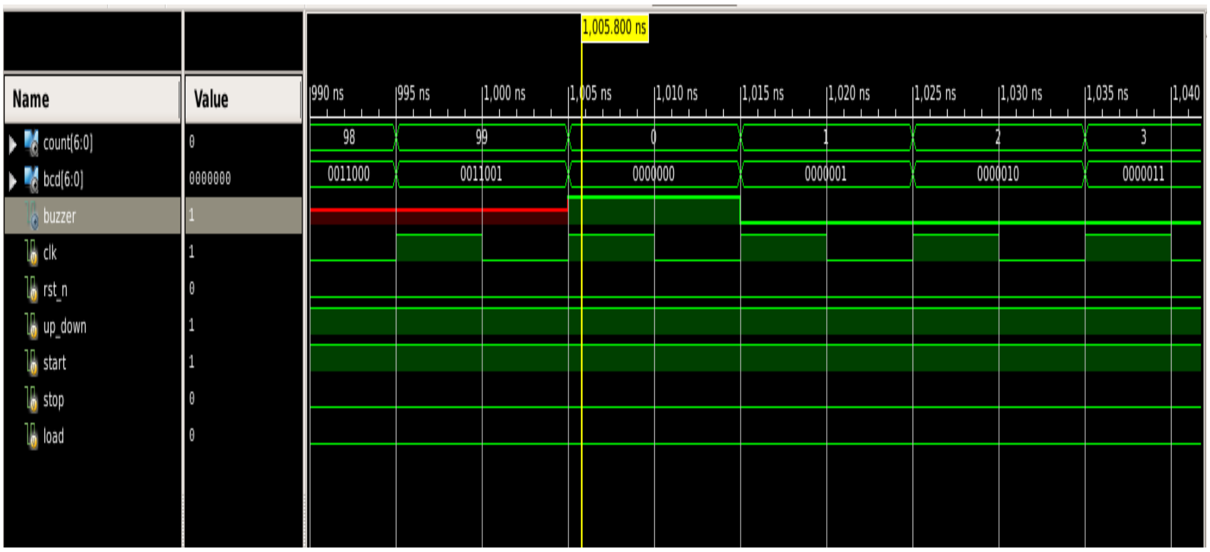
step 4 : Display the Result on the 7-Segment LED on the FPGA board and check for Output.

Testbench Output for the first code

Case1:

when updown=1 and start=1 and rest all buttons are low

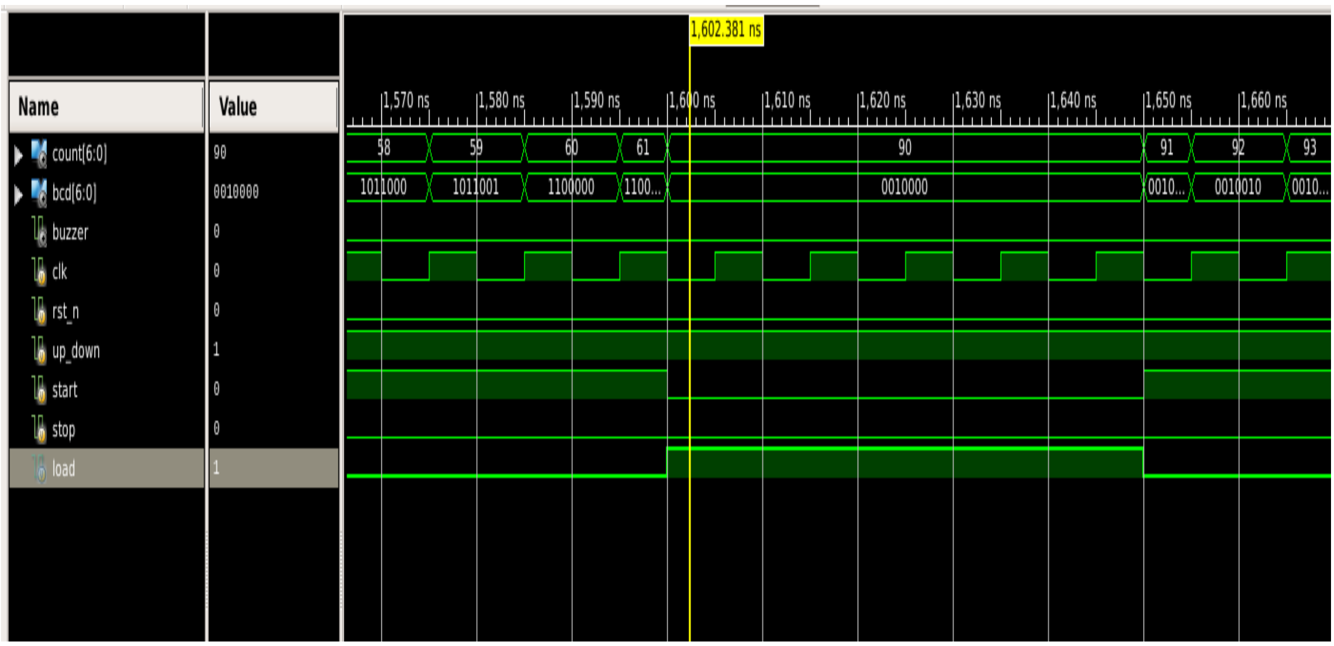
When count is 99 it starts counting back from 0 and the buzzer goes high



Case2:

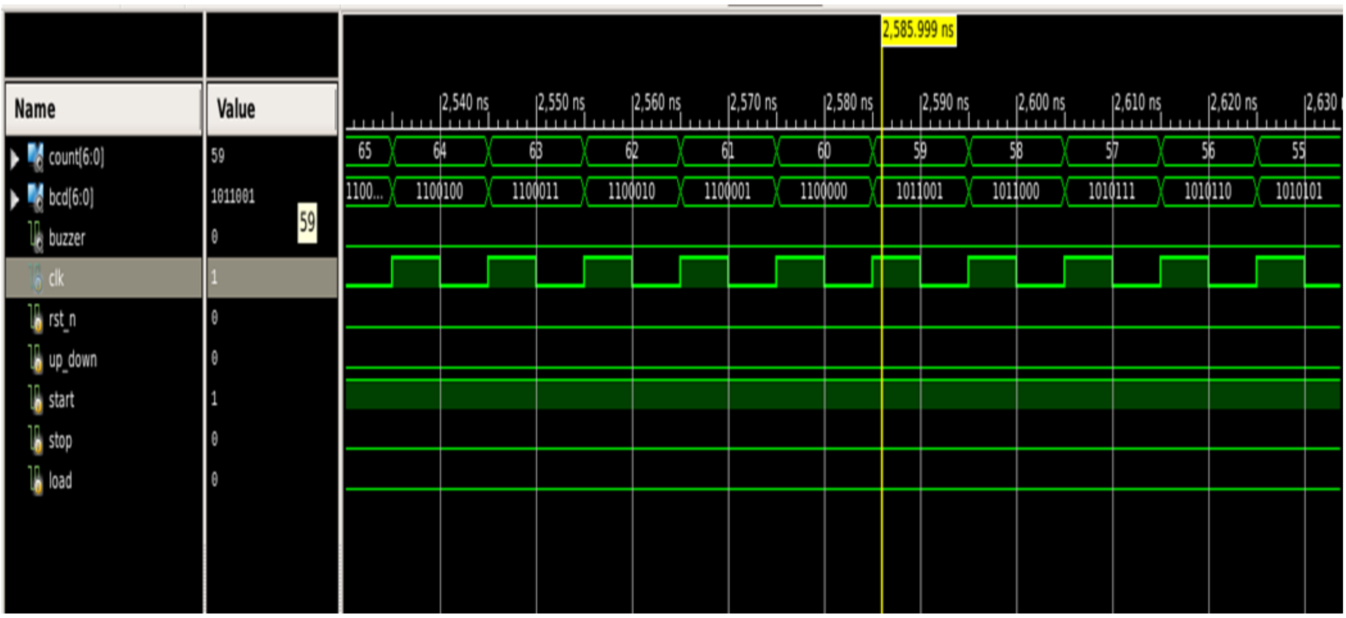
When load=1 and updown=1, count=90 irrespective of clk

And when load=0 and updown=1 , count starts incrementing from 90



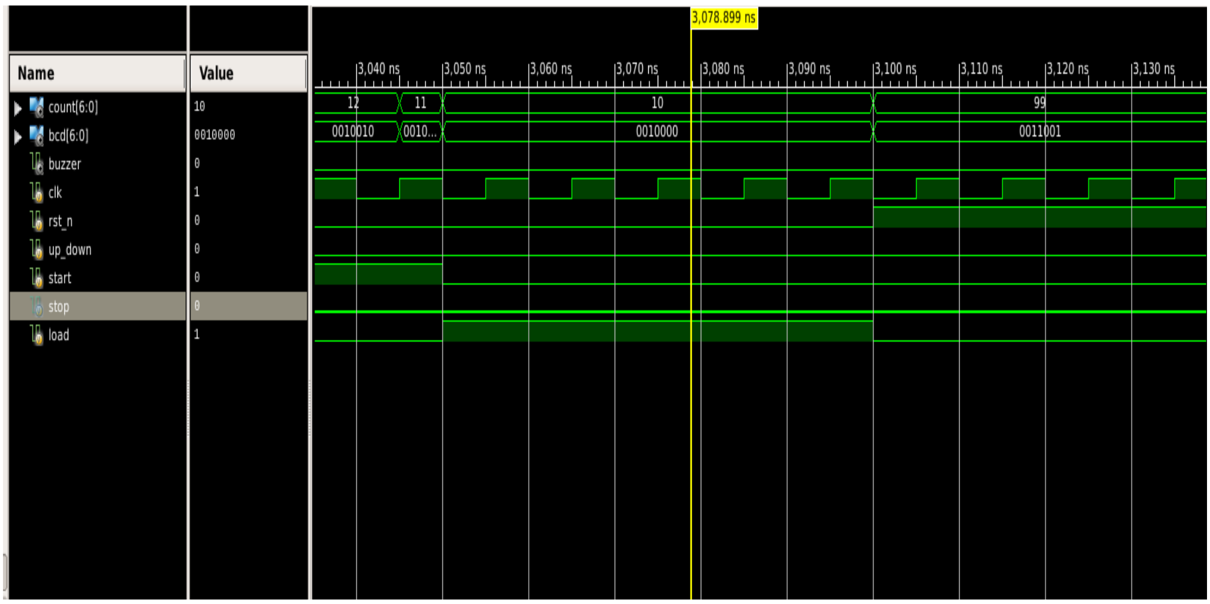
Case3:

when start=1 and updown=0 ,count starts decrementing



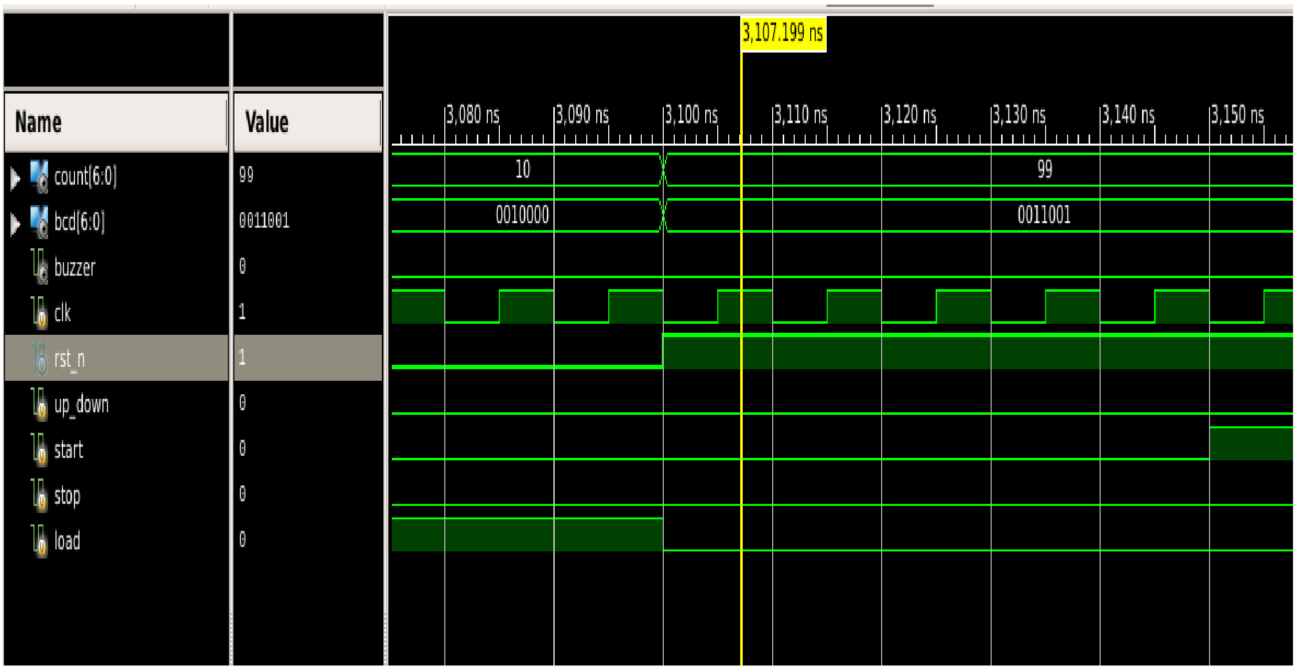
Case4:

when load=1 and updown=0 , count is 10



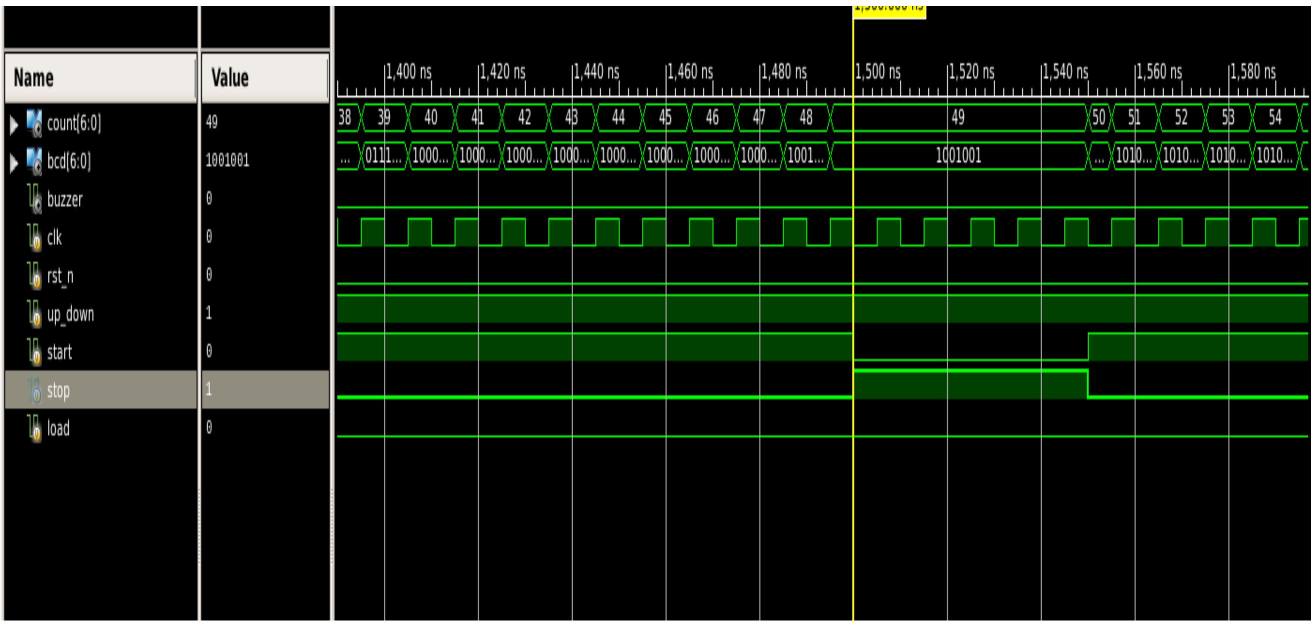
Case5:

when load=0, reset=1, updown=0 , count is 99

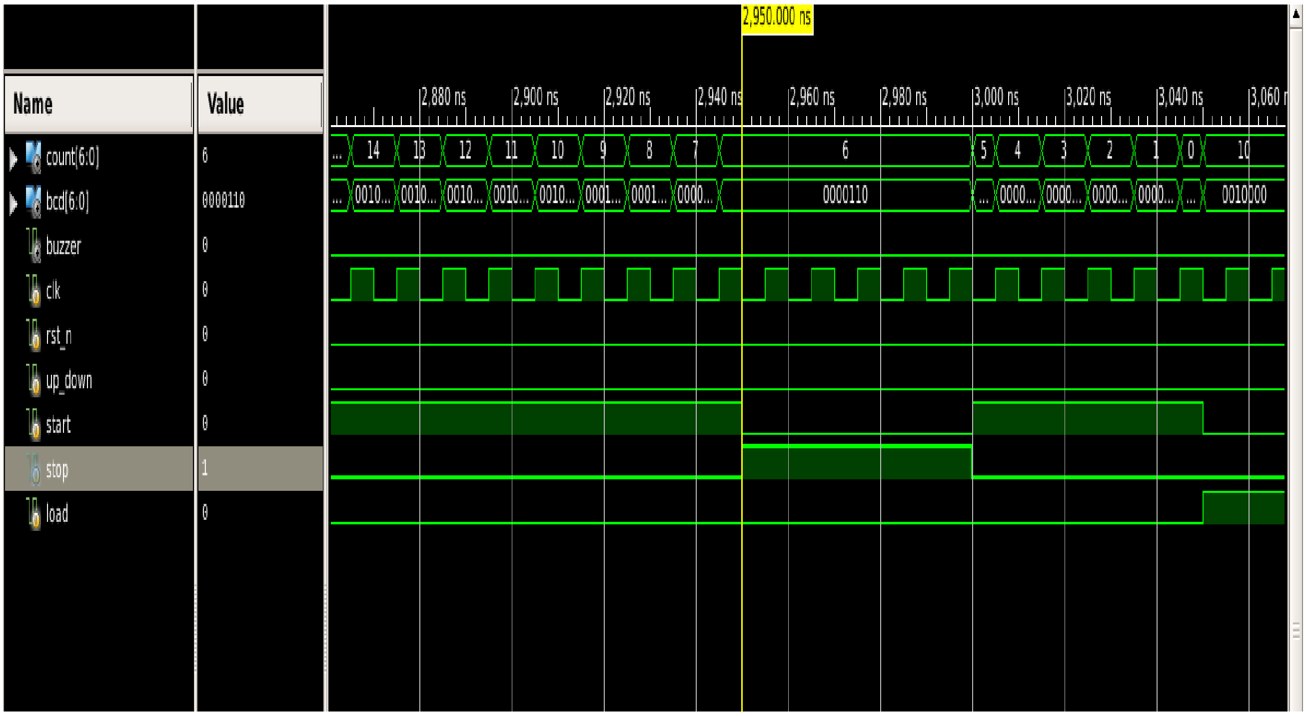


Case6:

when stop=1 and updown=1, count will stop at a particular value here it is 49



Case7: when stop=1 and updown=0 , count will stop at a particular value here it is 6



How we approached?

- In order to get the output according to the condition given, we first constructed the testbench for both the up and down counters using the if-else condition.We attempted to write code for a clockdivider, and while we were successful in that endeavor, the debouncing condition we attempted to implement on an FPGA did not function.
- We allocated more time for design and testing, therefore we had very little time left over for 7 segment and debouncing implementation.
- We also tried implementing debouncing concept using FSM and we were successful in synthesis but we didn't get the expected output on the testbench.
- Later, using the first code that we have written, we made few changes and applied debouncing and 7 segment display but we failed to do so.

What did we learn from this task?

Time management: Since we gave more time on testbench and design , we gave less time for important things like 7 segment and debouncing. So for the further tasks we would try to give more time for complex blocks.