# CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

**Department of Electrical and Computer Engineering**

# ECE 526L

# 

# LAB – 6: Modelling of a reloadable 8-bit Up-counter

Written By: Raj Kumar

**Introduction:**

In this lab, we created a “**reloadable 8-bitup counter**”, also we created a reset “**synchronizer (AASD)**” type for the reset signal by using **Synopsys VCS** in **Linux OS** environment and different terminal commands.

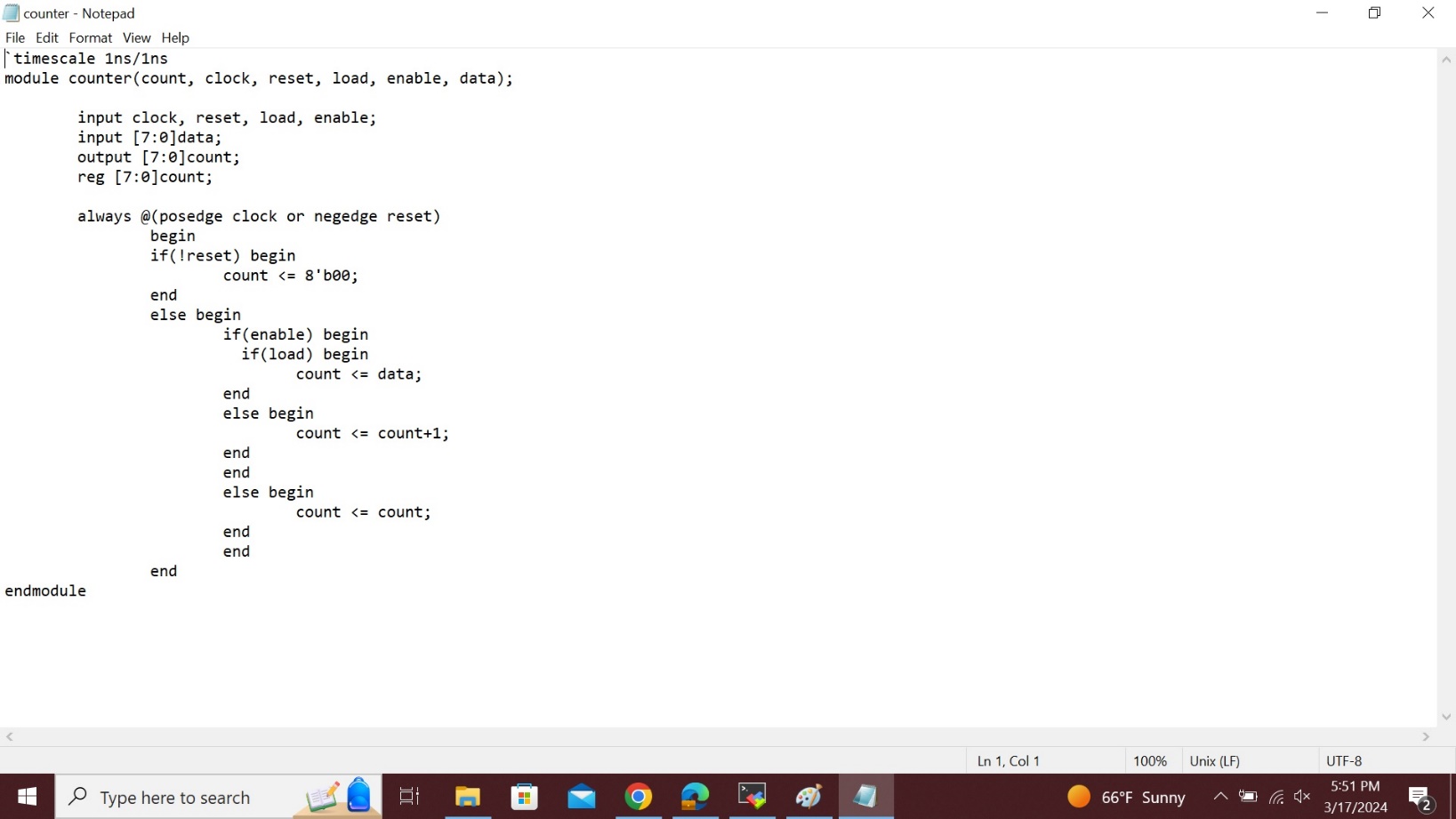
**Methodology:**The first thing is to be familiar with the Linux and Synopsys VCS that will navigate your system using terminal. Now to write, Verilog code for the module using a text editor. always use Linux based text editors and the file should have “**.v**” extension.

In this lab, we create a reloadable 8-bit up counter.

The counter we use the following ports single bit inputs are;

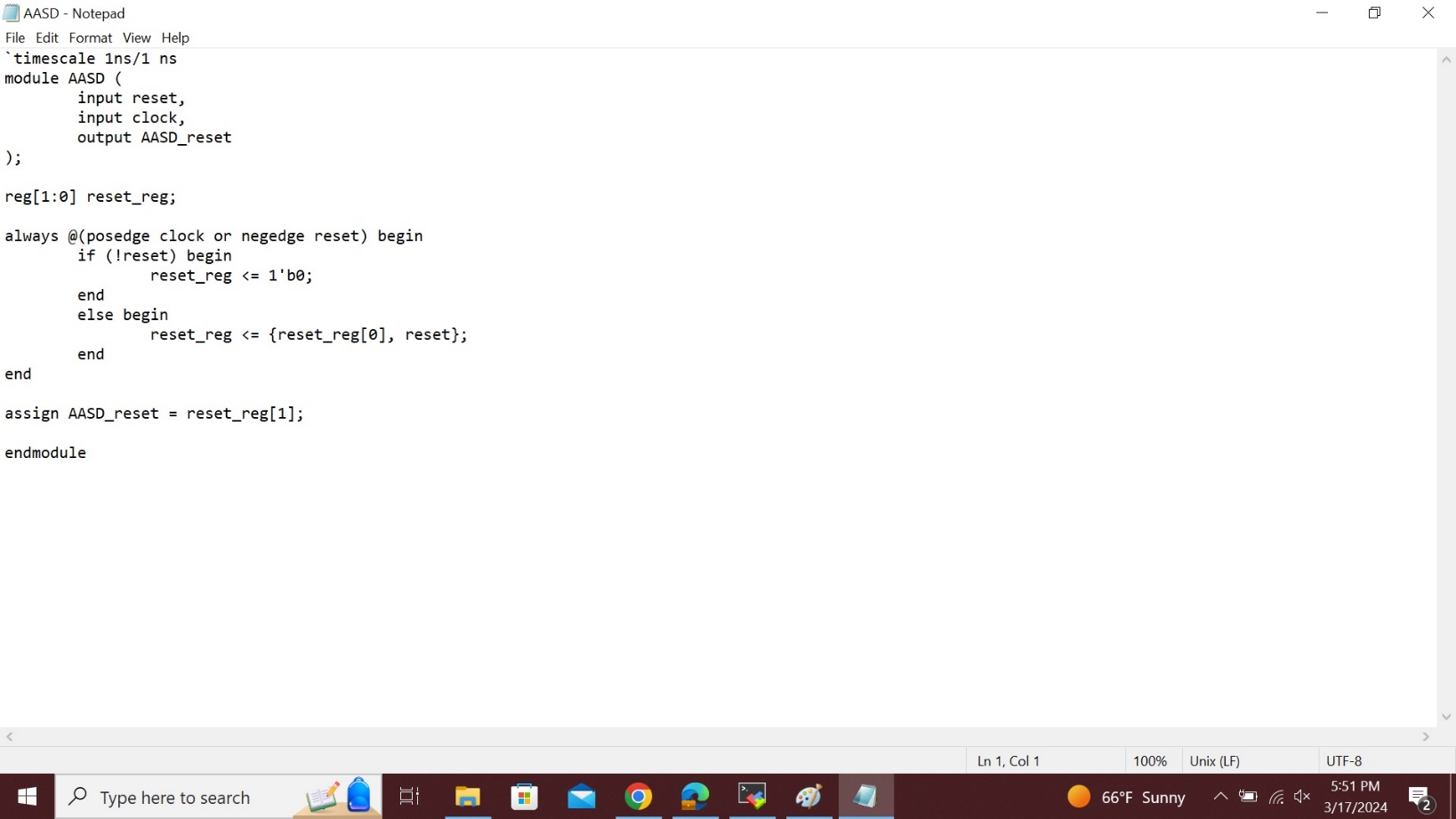
clock, reset, enable, load, 8-bit data input and 8-bit count output.

Save as “**counter.v**”



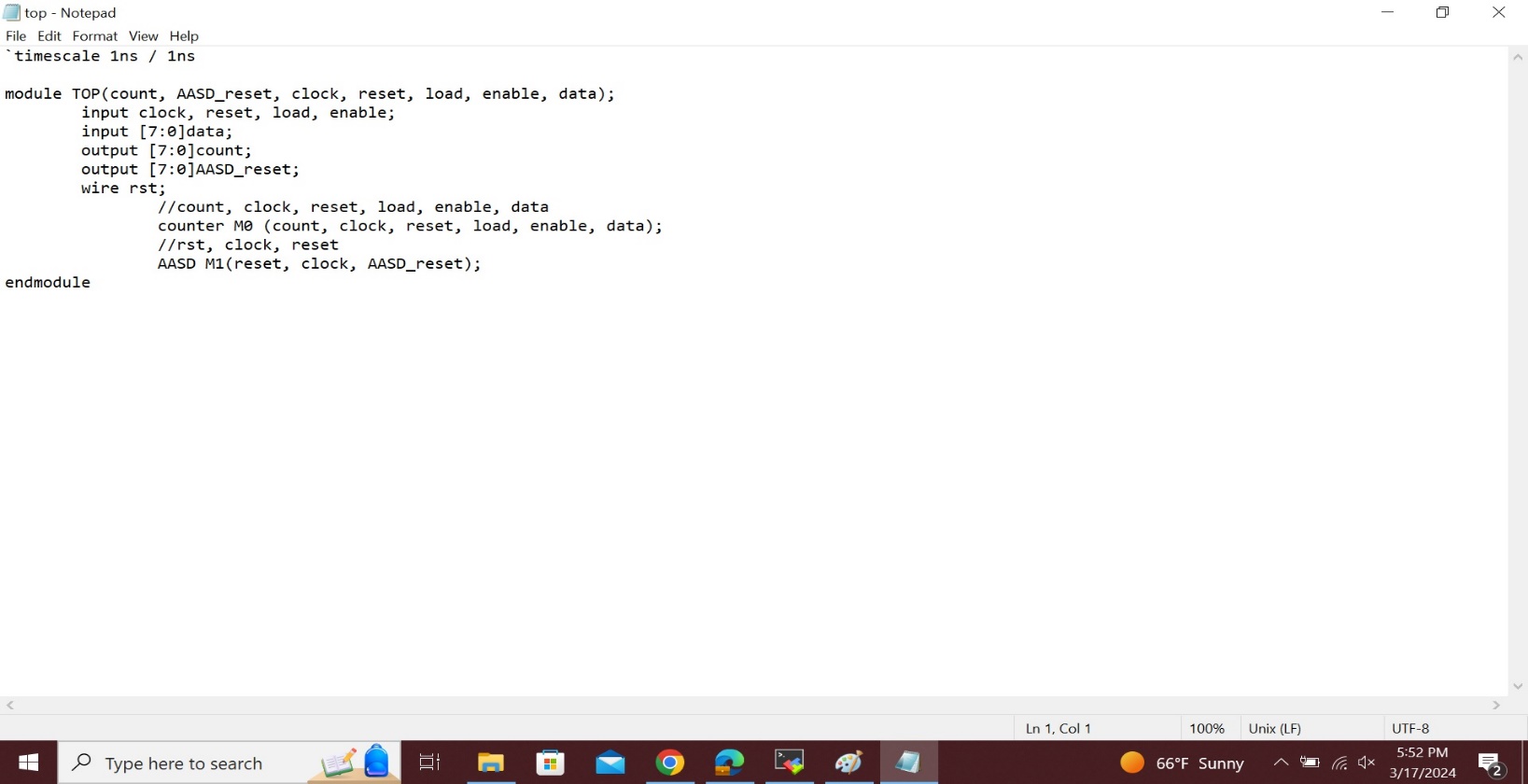
This is the fig of **“counter.v”,** store it in new file, also in folder **“Lab6”.**

Now, we should create a reset synchronizer (AASD) for the reset signal and save it as “**AASD.v**”.

The above fig is “**AASD.v**”.

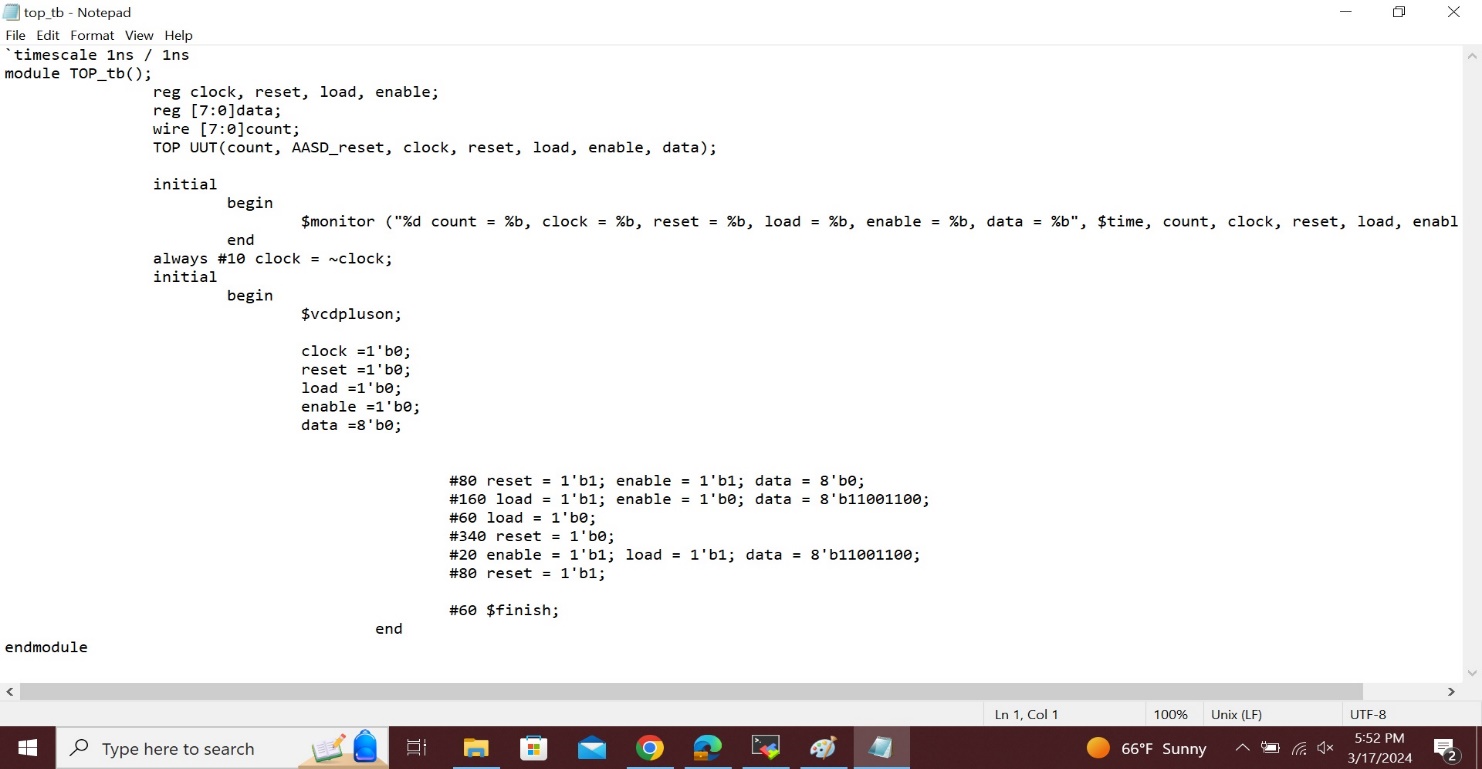
Now, we instantiate the two design modules in a top-level design unit.

The AASD circuit should be at the same level as the counter, not embedded in it and save it as “**top.v**”.



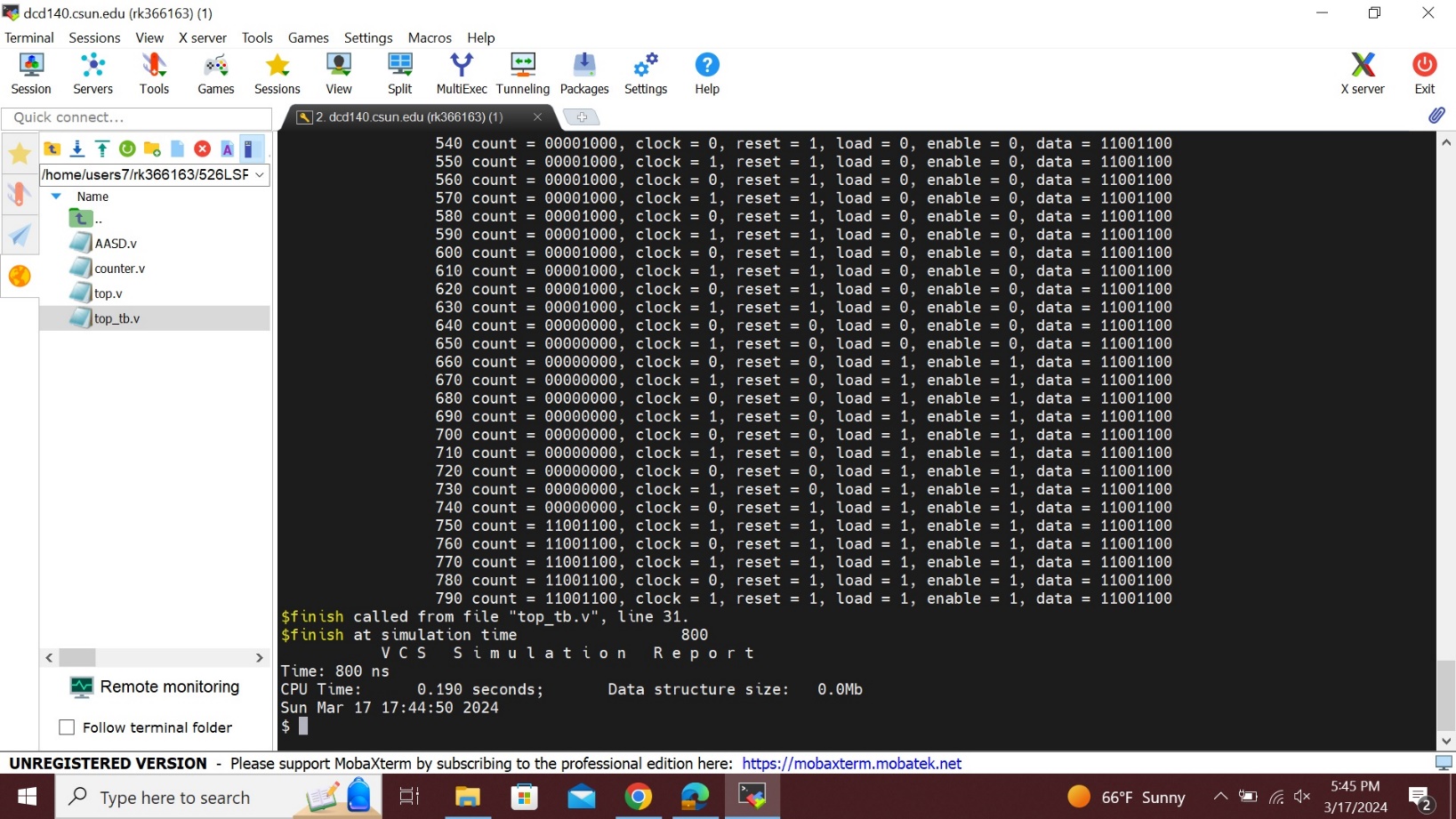
The above is figure of “**top.v**”.

Now we create a testbench for the “**top.v**” design module that does the following things that is given in the lab assignment and save it as “**top\_tb.v**”.

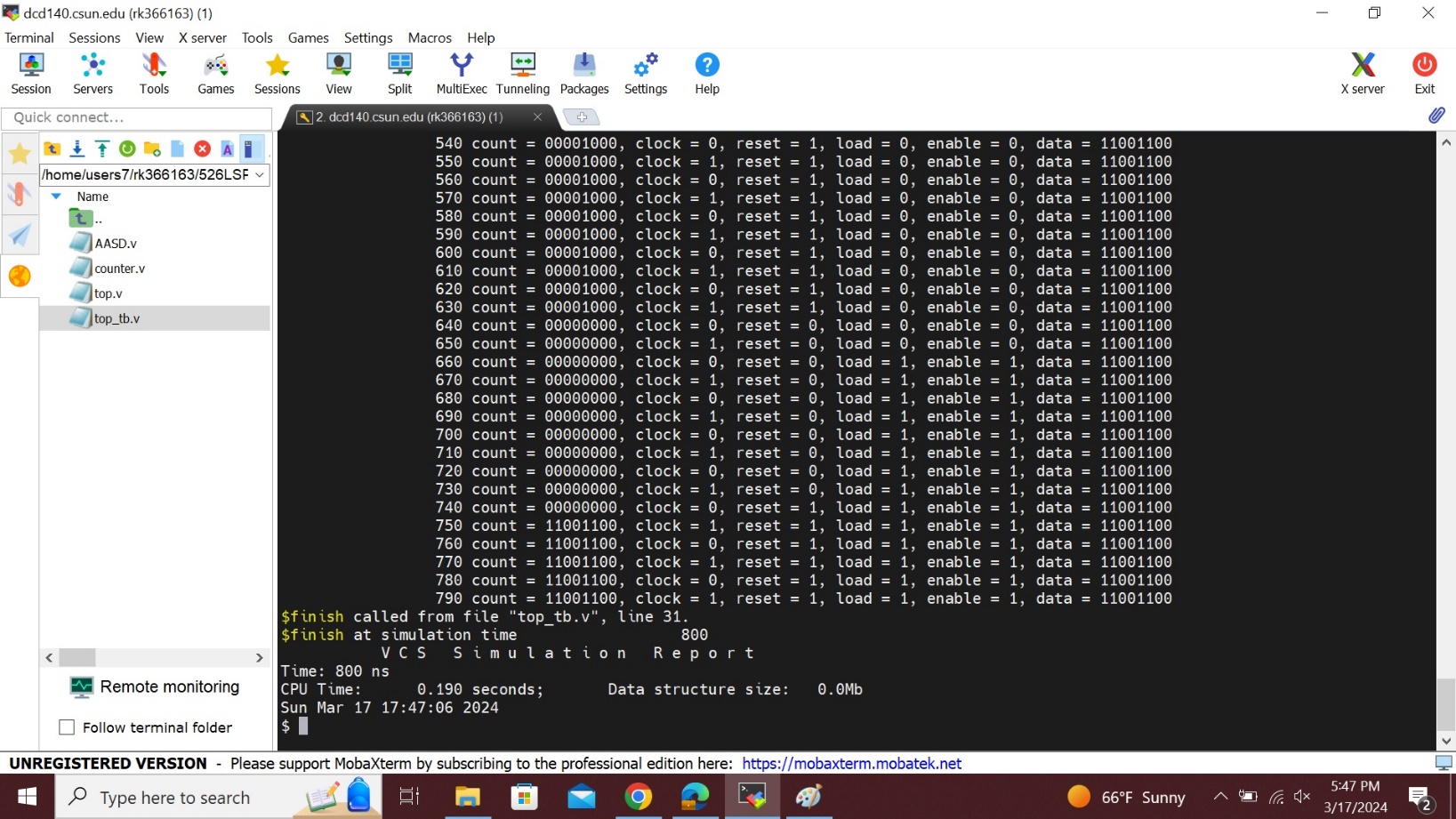
This is the fig of“**top\_tb.v**”.

Then use the command as follows;  
“**vcs -debug\_access+all counter.v AASD.v top.v top\_tb.v**”

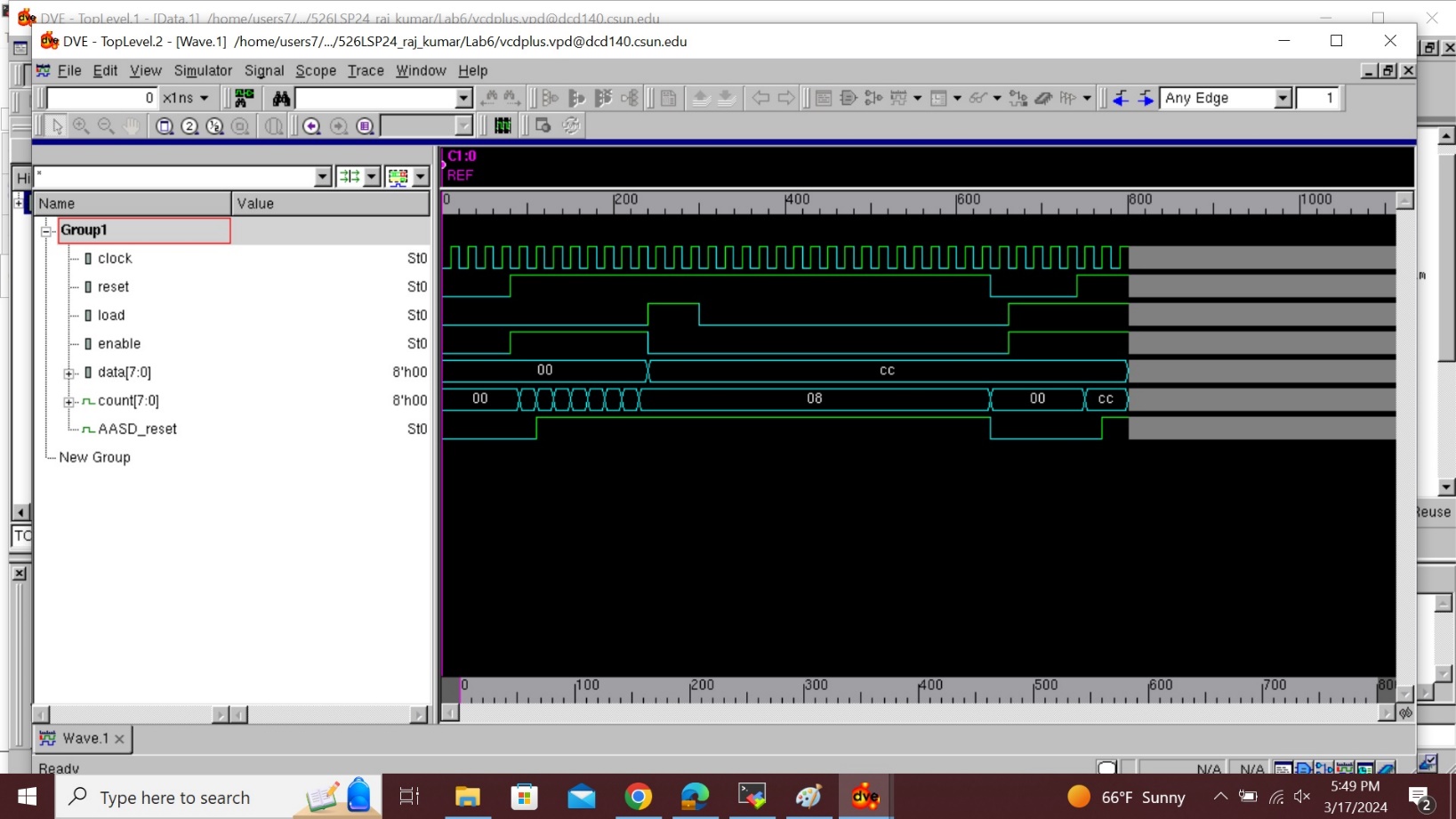
**simv:**



**Log File:**



To see the wave form first type “**dve** **-full64**” command line then we will get the blank simulation window then go to file and open database then choose **vcdplus.vpd** and open the file. Then, select all the signals with the mouse, then right click and select to   
“**add to wave => new wave view**”.



## Analysis: Usually, we look at the waveforms produced by your simulation program in order to evaluate the outcomes of the 8-bit counter simulation. The waveforms display the changes in signals over time as a result of the test scenarios that are specified in the test bench, these behaviors will be represented visually by the simulation waveforms. Using the test scenarios that have been provided, we should confirm that the counter performs as predicted. If the waveforms match what we expected, then the reset synchronizer and counter are operating as predicted. In the case that unexpected behaviors or problems arise, we must examine and debug our solution.

**Conclusion:**

In this lab, we create a reloadable 8-bit up counter and also reset synchronizer for the reset signal and a top-level design unit by using some commands and some tools by the Linux Synopsis VCS.

I hereby attest that this lab report is entirely my own work. I have not copied either code or text from anyone, neither have I allowed nor I will let anyone to copy my work.

Name(printed) Raj Kumar

Name(signed) Raj Kumar

Date 03/17/2024

## Lab Report Answers:

1. If the reset was synchronous, how would the circuit behave differently?

**Ans.** The Synchronous reset makes sure that the clock and other logic in the design are synchronized and under control during the reset process of the counter. By doing so, possible bugs or problems that can arise in asynchronous reset systems may be avoided. It might still cause a one-cycle delay in the reset process and present more timing concerns. our unique design specifications and scheduling limitations will determine which synchronous or asynchronous reset method is best for us.

1. How would you change the code to have a defined max counter 245 and then counter rolls over (i.e. returns to zero) and then starts counting back-up.

**Ans.** To create a counter that has a defined maximum count of 245, rolls over to zero, and then starts counting back up, you can modify the 8-bit counter module by adding logic to handle the maximum count limit. The counter's maximum value of 245 is now defined. It resets to zero when it hits this number (binary 11110101), making sure it stays below 245 completely. Then counter counts from 0 to 245 and goes back to 0 when it hits 245, at that point it rolls over to 0. This, module may provide us with a counter that behaves as we want it to, with a count limit of 245 before it rolls over to zero.