CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

Department of Electrical and Computer Engineering

ECE 526L

LAB – 6

# RELOADABLE 8-BIT UP COUNTER

Professor : Orod Haghighara

Written By : Gunupudi. Tarun Kumar

**INTRODUCTION:**

In this lab we will create a reloadable 8-bitup counter. Also you will create a reset synchronizer(AASD type for the reset signal by using Synopsys VCS in Linux OS environment and different terminal commands

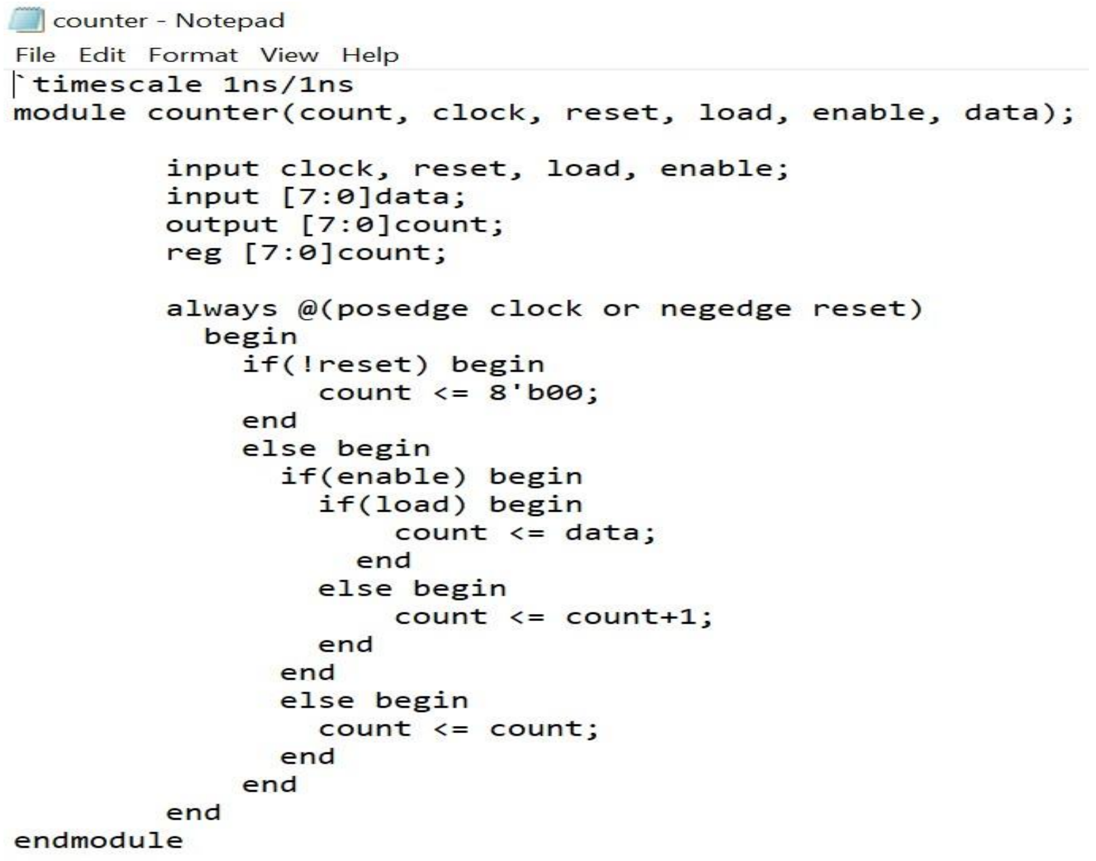
**METHODOLOGY:**

The first thing is being familiar with the Linux and Synopsys VCS that will navigate your system using terminal

Now to write a Verilog code for the module using a text editor. always use linux based text editors and the file should have ".v" extension.

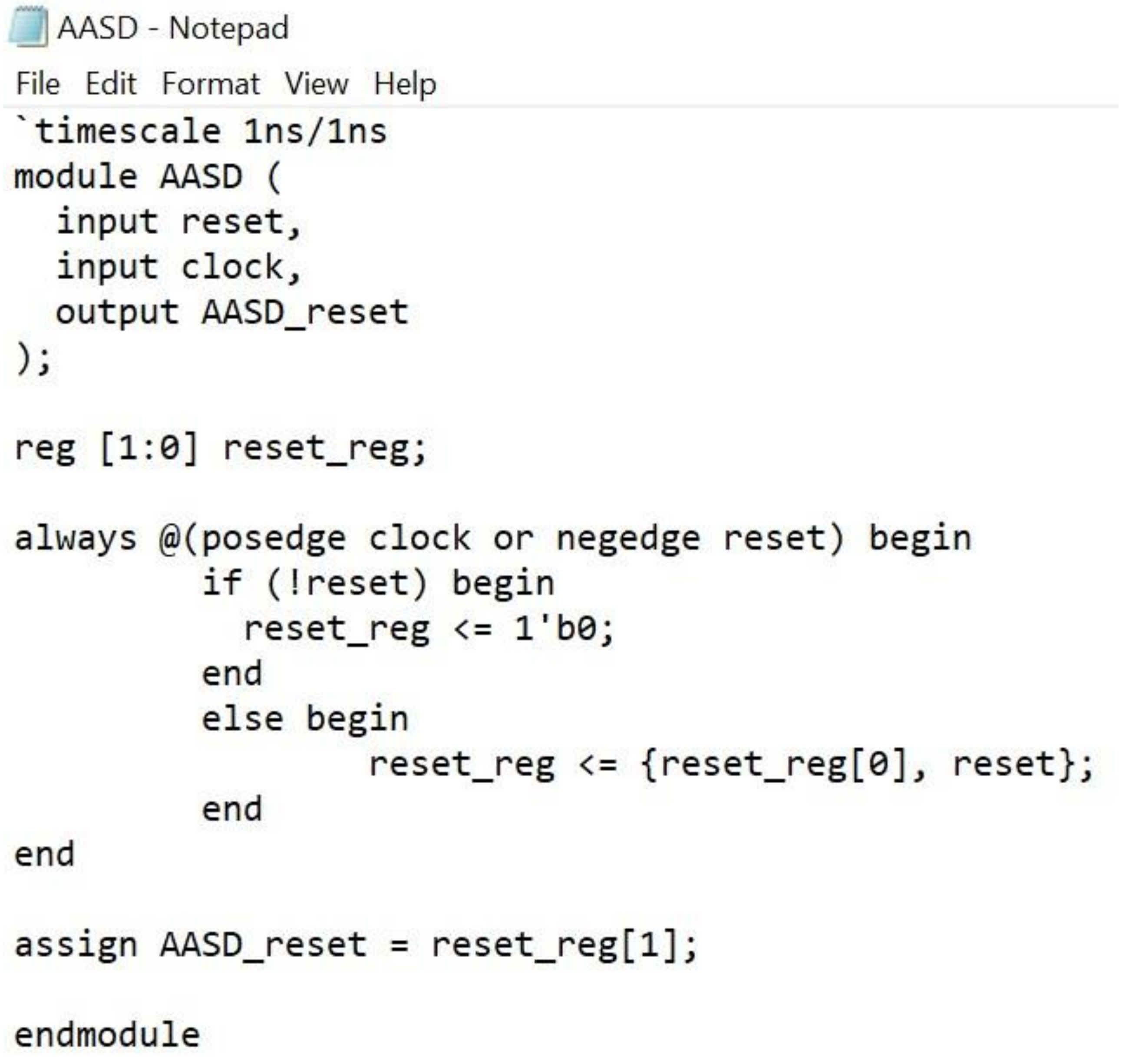
In this lab we will create a reloadable 8-bit up counter. for the counter we use the following ports single bit inputs: clock, reset, enable, load, eight bit data input and eight bit count output . And now save as "counter.v"

The module name and file name should be consistent



```
`timescale 1ns/1ns
module counter(count, clock, reset, load, enable, data);

        input clock, reset, load, enable;
        input [7:0]data;
        output [7:0]count;
        reg [7:0]count;

        always @(posedge clock or negedge reset)
          begin
            if(!reset) begin
                count <= 8'b00;
            end
            else begin
              if(enable) begin
                if(load) begin
                    count <= data;
                  end
                else begin
                    count <= count+1;
                end
              end
              else begin
                count <= count;
              end
            end
        end
endmodule
```

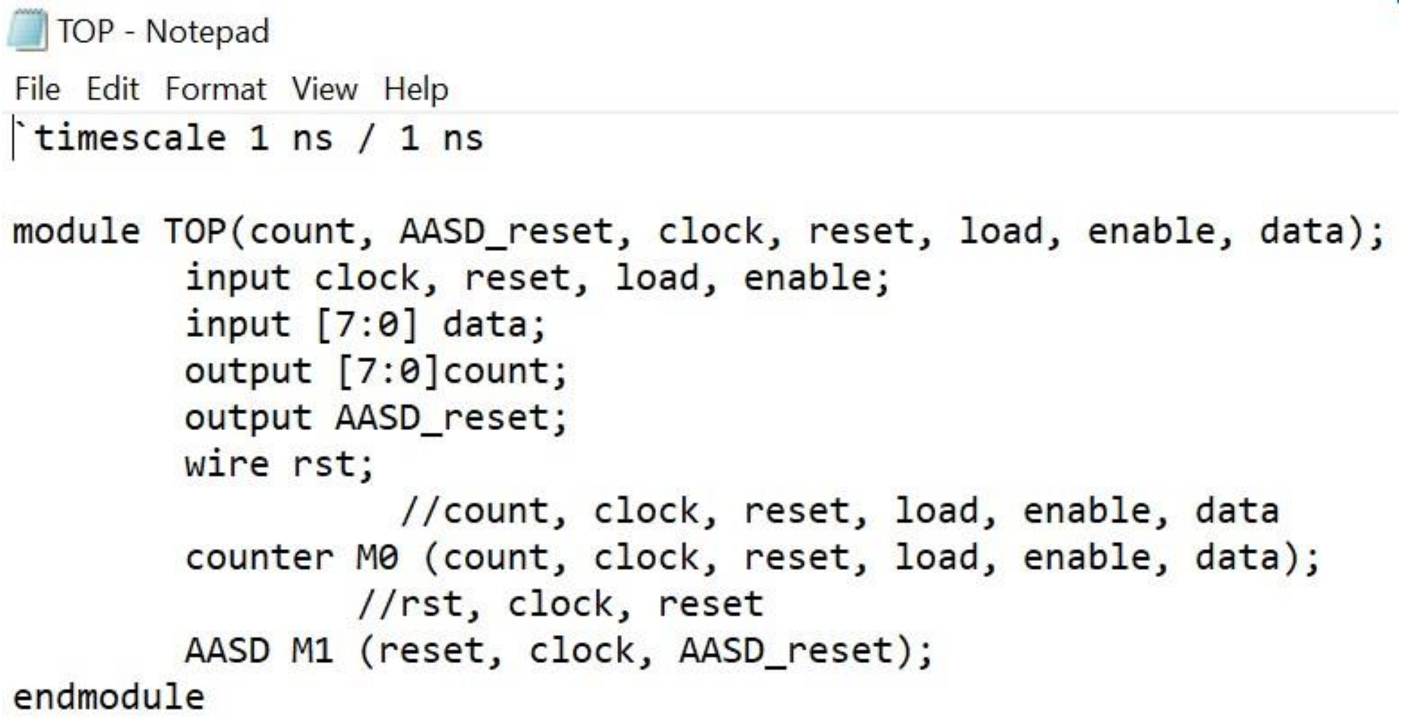
This is the fig of **“counter.v”,** store it in new file, also in folder **“lab6”** Now we should create a reset synchronizer (AASD) for the reset signal and save it as “AASD.v”



```
`timescale 1ns/1ns
module AASD (
  input reset,
  input clock,
  output AASD_reset
);

reg [1:0] reset_reg;

always @(posedge clock or negedge reset) begin
         if (!reset) begin
           reset_reg <= 1'b0;
         end
         else begin
                 reset_reg <= {reset_reg[0], reset};
         end
end

assign AASD_reset = reset_reg[1];

endmodule
```

**The above fig is “AASD.v”**

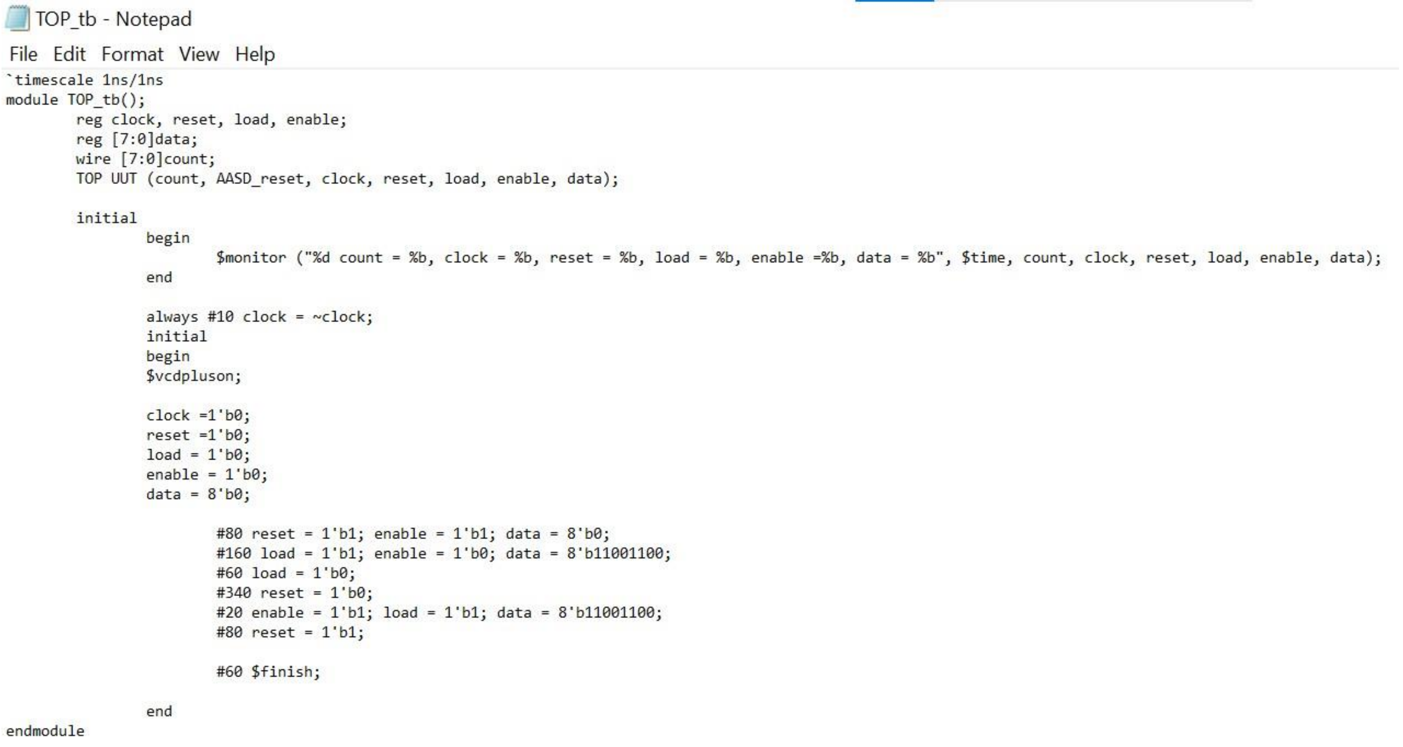
Now we should instantiate the two design modules in a top-level design unit. The AASD circuit should be at the same level as the counter, not embedded in it and save it as “ TOP.v ”



```
`timescale 1 ns / 1 ns

module TOP(count, AASD_reset, clock, reset, load, enable, data);
        input clock, reset, load, enable;
        input [7:0] data;
        output [7:0]count;
        output AASD_reset;
        wire rst;
                //count, clock, reset, load, enable, data
        counter M0 (count, clock, reset, load, enable, data);
              //rst, clock, reset
        AASD M1 (reset, clock, AASD_reset);
endmodule
```

**The above is fig of " TOP.v "**

Now we should create a testbench for the " TOP.v "design module that does the following things that is given lab assignment and save it as " TOP_tb.v "



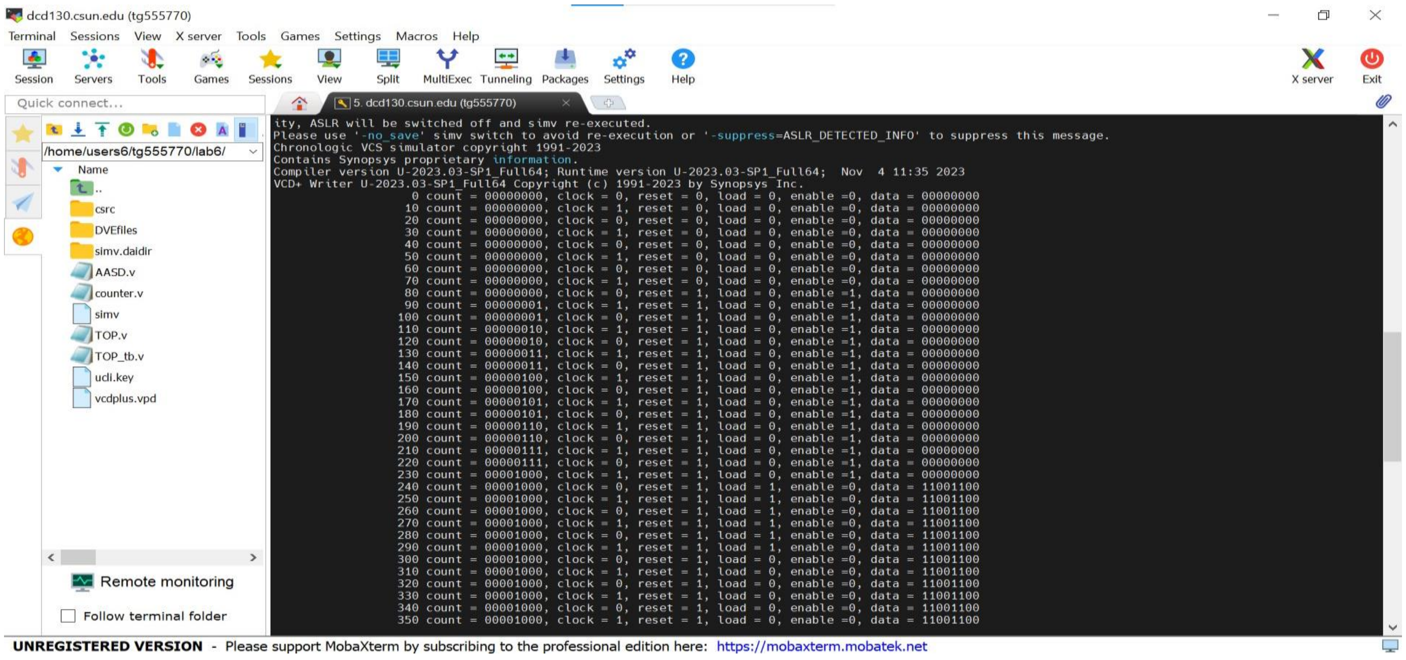
```
`timescale 1ns/1ns
module TOP_tb();
        reg clock, reset, load, enable;
        reg [7:0]data;
        wire [7:0]count;
        TOP UUT (count, AASD_reset, clock, reset, load, enable, data);

        initial
                begin
                        $monitor ("%d count = %b, clock = %b, reset = %b, load = %b, enable =%b, data = %b", $time, count, clock, reset, load, enable, data);
                end

                always #10 clock = ~clock;
                initial
                begin
                $vcdpluson;

                clock =1'b0;
                reset =1'b0;
                load = 1'b0;
                enable = 1'b0;
                data = 8'b0;

                        #80 reset = 1'b1; enable = 1'b1; data = 8'b0;
                        #160 load = 1'b1; enable = 1'b0; data = 8'b11001100;
                        #60 load = 1'b0;
                        #340 reset = 1'b0;
                        #20 enable = 1'b1; load = 1'b1; data = 8'b11001100;
                        #80 reset = 1'b1;

                        #60 $finish;

                end
endmodule
```

**This is the fig of “TOP_tb.v”**

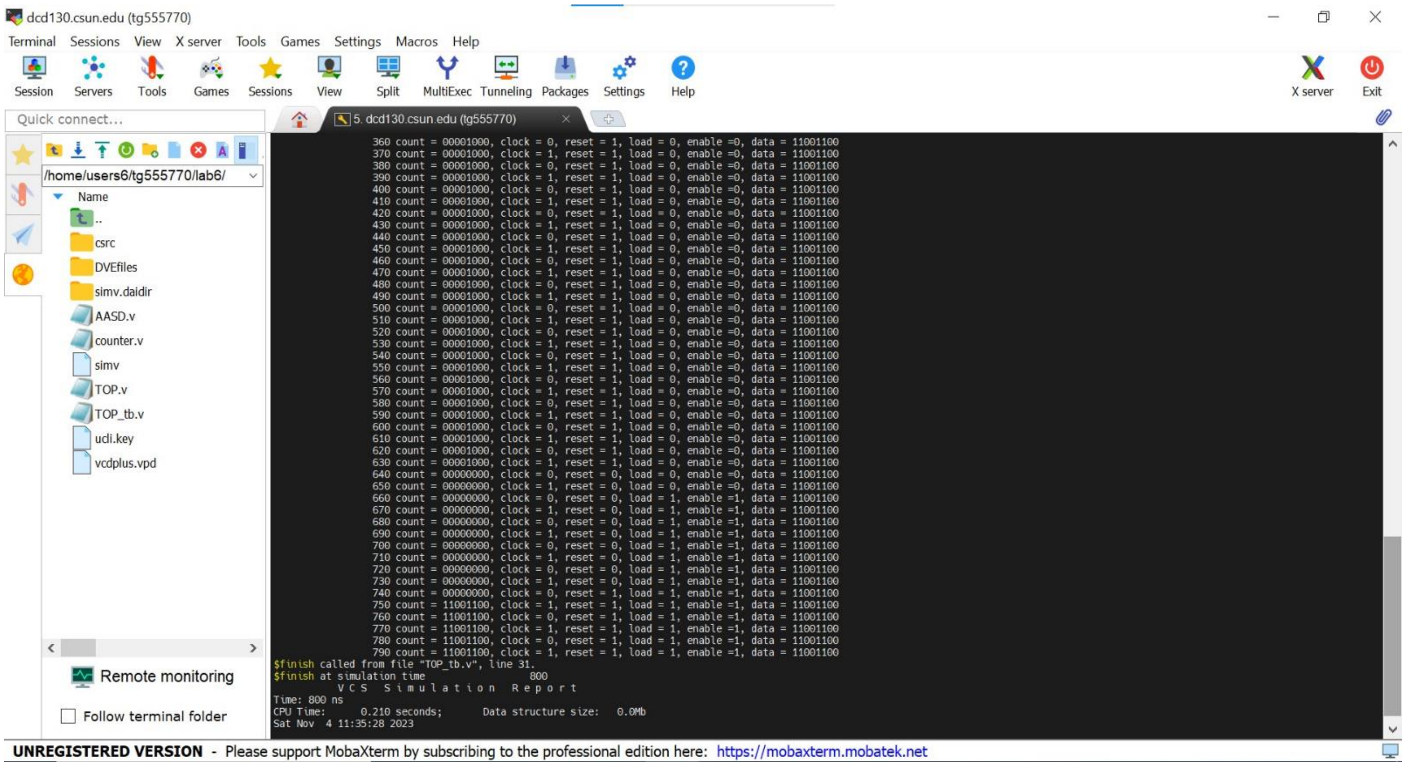
Then use the command as follows:

“ vcs -debug_access+all counter.v AASD.v TOP.v TOP_tb.v ” If we don’t see any error messages if everything is type correctly then output should look like below figure.

**SIMV:**



```
ity, ASLR will be switched off and simv re-executed.
Please use '-no_save' simv switch to avoid re-execution or '-suppress=ASLR_DETECTED_INFO' to suppress this message.
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP1_Full64; Runtime version U-2023.03-SP1_Full64;  Nov  4 11:35 2023
VCD+ Writer U-2023.03-SP1_Full64 Copyright (c) 1991-2023 by Synopsys Inc.
                   0 count = 00000000, clock = 0, reset = 0, load = 0, enable =0, data = 00000000
                  10 count = 00000000, clock = 1, reset = 0, load = 0, enable =0, data = 00000000
                  20 count = 00000000, clock = 0, reset = 0, load = 0, enable =0, data = 00000000
                  30 count = 00000000, clock = 1, reset = 0, load = 0, enable =0, data = 00000000
                  40 count = 00000000, clock = 0, reset = 0, load = 0, enable =0, data = 00000000
                  50 count = 00000000, clock = 1, reset = 0, load = 0, enable =0, data = 00000000
                  60 count = 00000000, clock = 0, reset = 0, load = 0, enable =0, data = 00000000
                  70 count = 00000000, clock = 1, reset = 0, load = 0, enable =0, data = 00000000
                  80 count = 00000000, clock = 0, reset = 1, load = 0, enable =1, data = 00000000
                  90 count = 00000001, clock = 1, reset = 1, load = 0, enable =1, data = 00000000
                 100 count = 00000001, clock = 0, reset = 1, load = 0, enable =1, data = 00000000
                 110 count = 00000010, clock = 1, reset = 1, load = 0, enable =1, data = 00000000
                 120 count = 00000010, clock = 0, reset = 1, load = 0, enable =1, data = 00000000
                 130 count = 00000011, clock = 1, reset = 1, load = 0, enable =1, data = 00000000
                 140 count = 00000011, clock = 0, reset = 1, load = 0, enable =1, data = 00000000
                 150 count = 00000100, clock = 1, reset = 1, load = 0, enable =1, data = 00000000
                 160 count = 00000100, clock = 0, reset = 1, load = 0, enable =1, data = 00000000
                 170 count = 00000101, clock = 1, reset = 1, load = 0, enable =1, data = 00000000
                 180 count = 00000101, clock = 0, reset = 1, load = 0, enable =1, data = 00000000
                 190 count = 00000110, clock = 1, reset = 1, load = 0, enable =1, data = 00000000
                 200 count = 00000110, clock = 0, reset = 1, load = 0, enable =1, data = 00000000
                 210 count = 00000111, clock = 1, reset = 1, load = 0, enable =1, data = 00000000
                 220 count = 00000111, clock = 0, reset = 1, load = 0, enable =1, data = 00000000
                 230 count = 00001000, clock = 1, reset = 1, load = 0, enable =1, data = 00000000
                 240 count = 00001000, clock = 0, reset = 1, load = 1, enable =0, data = 11001100
                 250 count = 00001000, clock = 1, reset = 1, load = 1, enable =0, data = 11001100
                 260 count = 00001000, clock = 0, reset = 1, load = 1, enable =0, data = 11001100
                 270 count = 00001000, clock = 1, reset = 1, load = 1, enable =0, data = 11001100
                 280 count = 00001000, clock = 0, reset = 1, load = 1, enable =0, data = 11001100
                 290 count = 00001000, clock = 1, reset = 1, load = 1, enable =0, data = 11001100
                 300 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 310 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 320 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 330 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 340 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 350 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
```

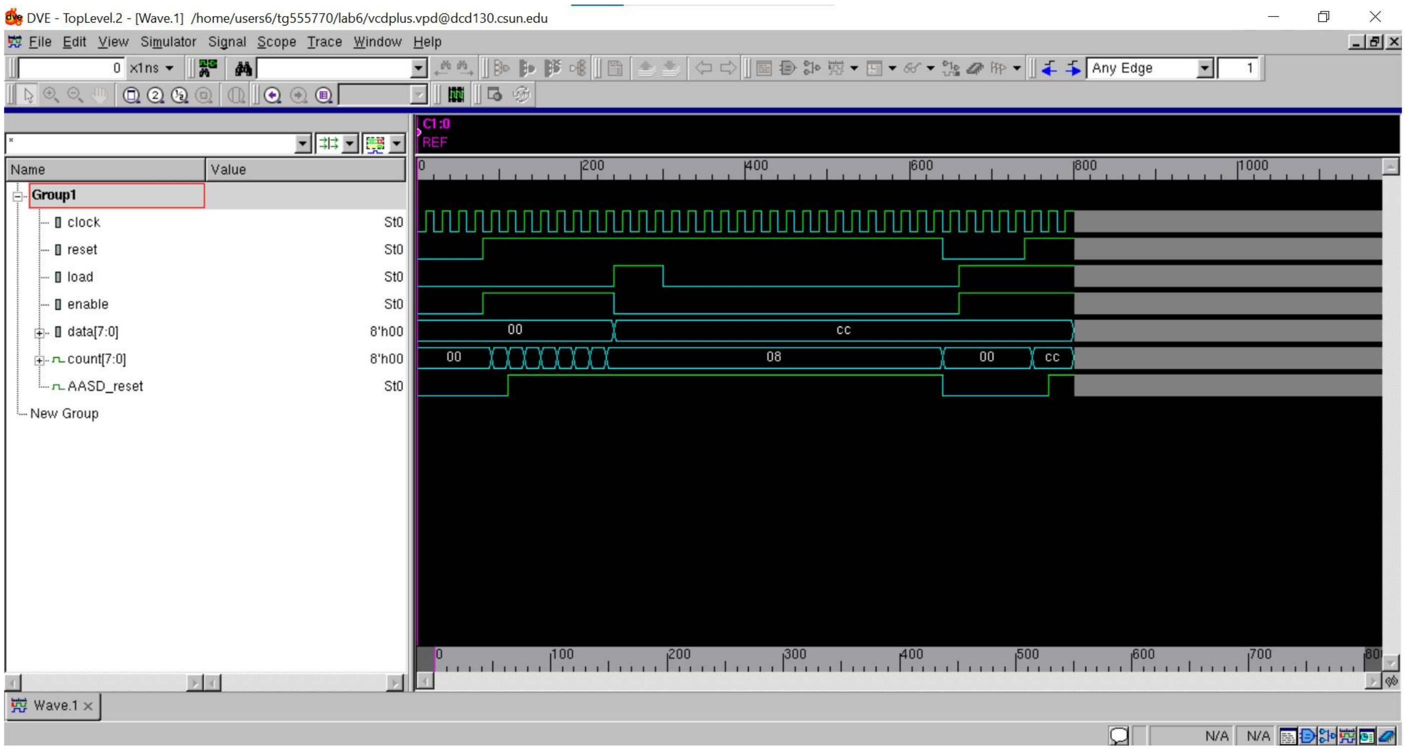


```
                 360 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 370 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 380 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 390 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 400 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 410 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 420 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 430 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 440 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 450 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 460 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 470 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 480 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 490 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 500 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 510 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 520 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 530 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 540 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 550 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 560 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 570 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 580 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 590 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 600 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 610 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 620 count = 00001000, clock = 0, reset = 1, load = 0, enable =0, data = 11001100
                 630 count = 00001000, clock = 1, reset = 1, load = 0, enable =0, data = 11001100
                 640 count = 00000000, clock = 0, reset = 0, load = 0, enable =0, data = 11001100
                 650 count = 00000000, clock = 1, reset = 0, load = 0, enable =0, data = 11001100
                 660 count = 00000000, clock = 0, reset = 0, load = 1, enable =1, data = 11001100
                 670 count = 00000000, clock = 1, reset = 0, load = 1, enable =1, data = 11001100
                 680 count = 00000000, clock = 0, reset = 0, load = 1, enable =1, data = 11001100
                 690 count = 00000000, clock = 1, reset = 0, load = 1, enable =1, data = 11001100
                 700 count = 00000000, clock = 0, reset = 0, load = 1, enable =1, data = 11001100
                 710 count = 00000000, clock = 1, reset = 0, load = 1, enable =1, data = 11001100
                 720 count = 00000000, clock = 0, reset = 0, load = 1, enable =1, data = 11001100
                 730 count = 00000000, clock = 1, reset = 0, load = 1, enable =1, data = 11001100
                 740 count = 00000000, clock = 0, reset = 1, load = 1, enable =1, data = 11001100
                 750 count = 11001100, clock = 1, reset = 1, load = 1, enable =1, data = 11001100
                 760 count = 11001100, clock = 0, reset = 1, load = 1, enable =1, data = 11001100
                 770 count = 11001100, clock = 1, reset = 1, load = 1, enable =1, data = 11001100
                 780 count = 11001100, clock = 0, reset = 1, load = 1, enable =1, data = 11001100
                 790 count = 11001100, clock = 1, reset = 1, load = 1, enable =1, data = 11001100
$finish called from file "TOP_tb.v", line 31.
$finish at simulation time                  800
           V C S   S i m u l a t i o n   R e p o r t
Time: 800 ns
CPU Time:      0.210 seconds;       Data structure size:   0.0Mb
Sat Nov  4 11:35:28 2023
```

To see the wave form first type “dve” 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 finally select all the signals with the mouse ,then right click and select to “ add to wave->new wave view”.

Then we will get the waveform as shown in below figure


DVE - TopLevel.2 - [Wave.1] /home/users6/tg555770/lab6/vcdplus.vpd@dcd130.csun.edu
File Edit View Simulator Signal Scope Trace Window Help
Any Edge
C1:0
REF
00
cc
00
08
00
cc
Wave.1
N/A N/A

| Name | Value |
|---|---|
| Group1 | |
| clock | St0 |
| reset | St0 |
| load | St0 |
| enable | St0 |
| data[7:0] | 8'h00 |
| count[7:0] | 8'h00 |
| AASD_reset | St0 |
| New Group | |

**This is the final wave form**

## Analysis of result :

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 will create a reloadable 8 -bit up counter and also reset synchronizer for the reset signal and a top level deign unit  by using some commands and some tools by the linux synopsis VCS.

## LAB REPORT QUESTIONS:

**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.

**2**.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.

I hereby attest that this lab report is entirely my own work.I have not copied either code or text from anyone ,nor have I allowed or I will anyone to copy my work.

Name(printed) Gunupudi Tarun Kumar

Name(signed) G.Tarun Date 11/04/2023