

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-bit up 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.

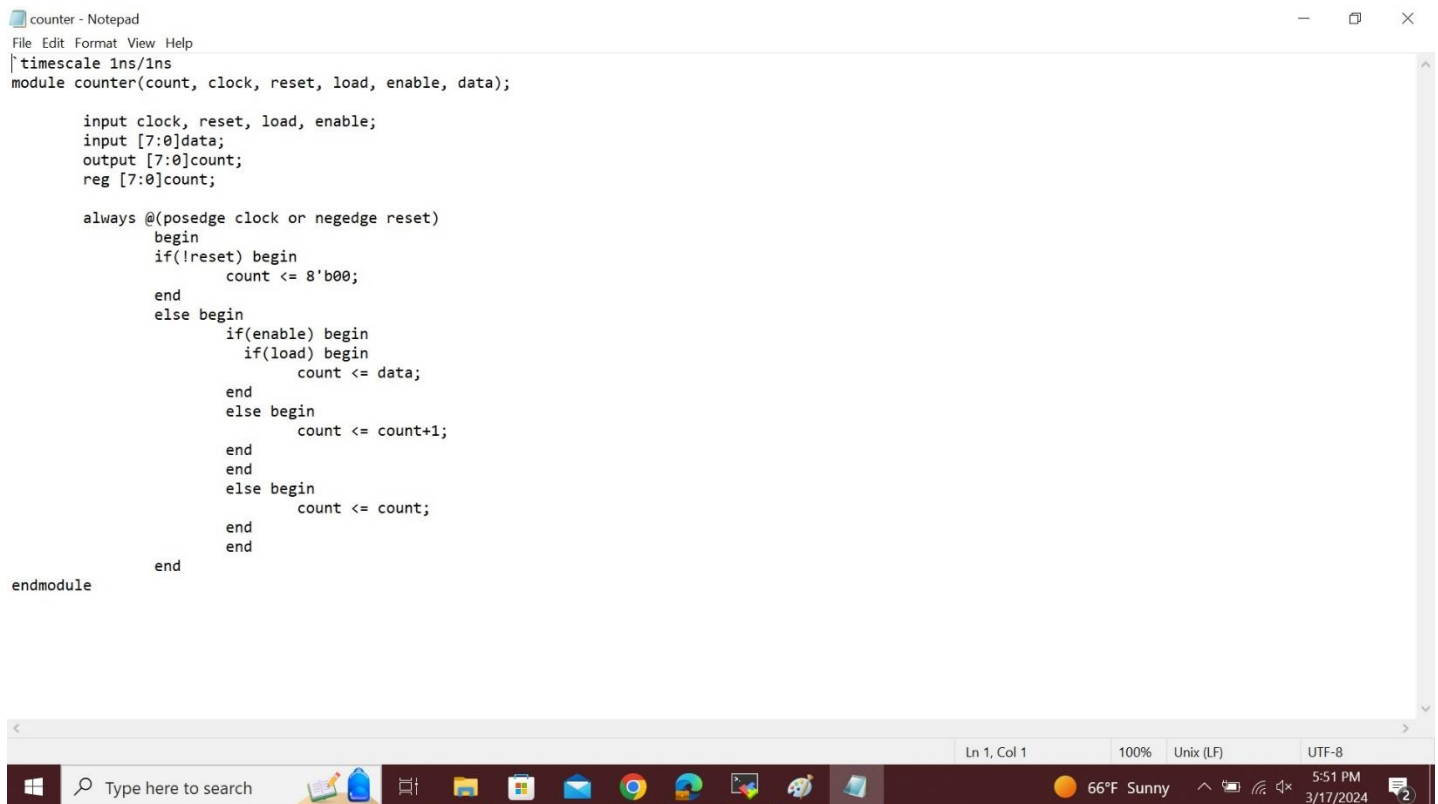
Methodology:

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**”



```
counter - Notepad
File Edit Format View Help
|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;

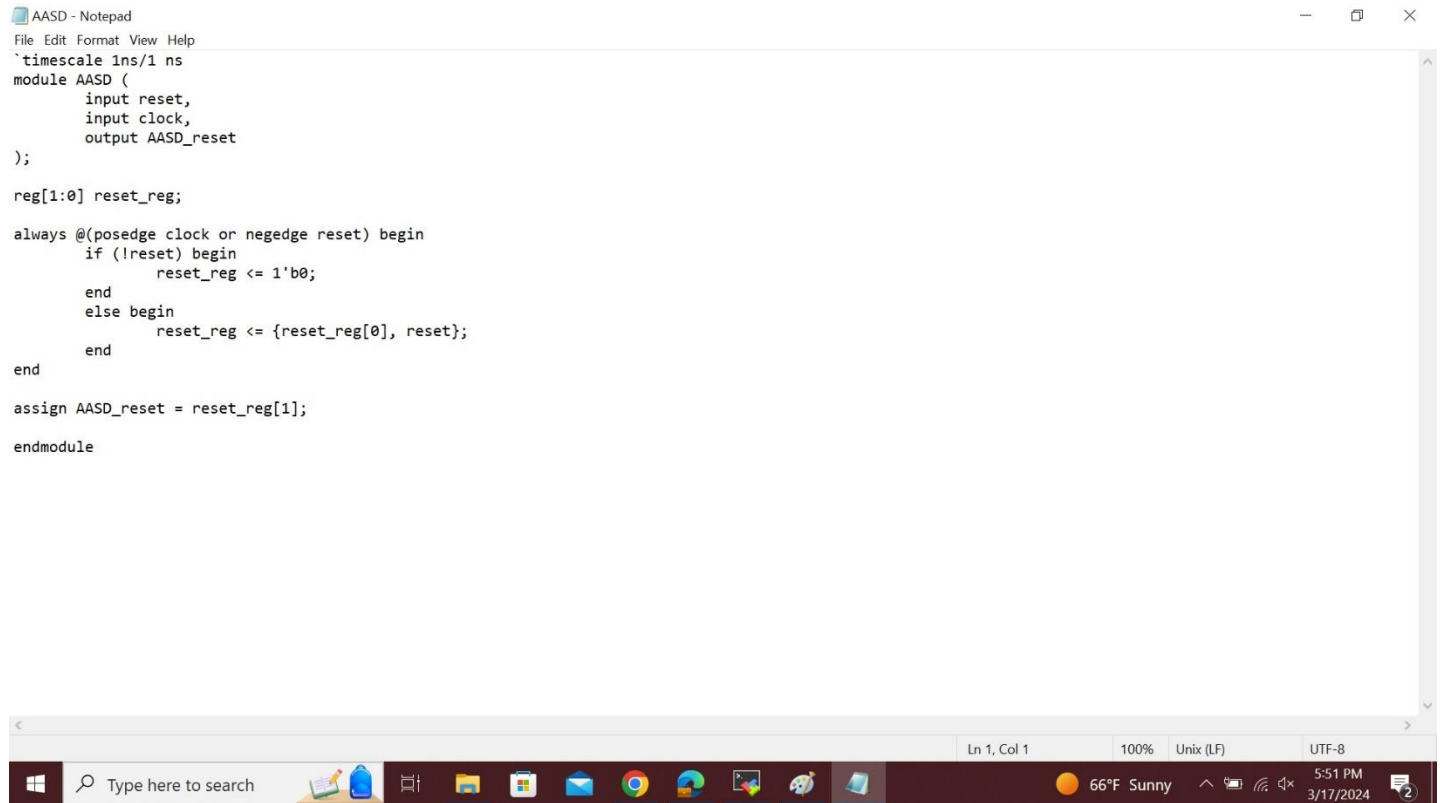
    always @(posedge clock or negedge reset)
    begin
        if(!reset) begin
            count <= 8'b00;
        end
        else begin
            if(enable) begin
                if(load) begin
                    count <= data;
                end
                else begin
                    count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

Ln 1, Col 1 100% Unix (LF) UTF-8

66°F Sunny 5:51 PM 3/17/2024

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**”.



```
AASD - Notepad
File Edit Format View Help
`timescale 1ns/1 ns
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

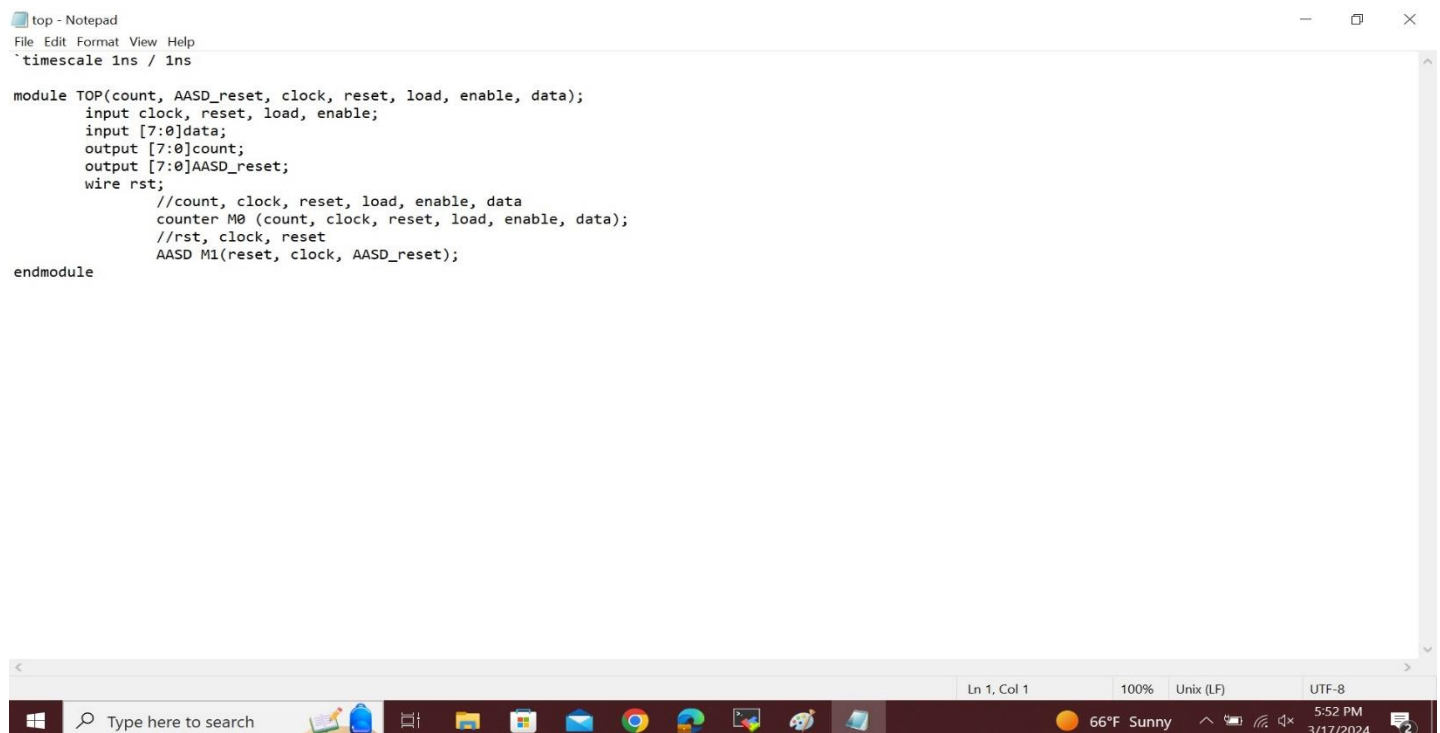
assign AASD_reset = reset_reg[1];

endmodule
```

The above fig is “**AASD.v**”.

Now, we 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**”.



```
top - Notepad
File Edit Format View Help
`timescale 1ns / 1ns

module TOP(count, AASD_reset, clock, reset, load, enable, data);
    input clock, reset, load, enable;
    input [7:0]data;
    output [7:0]count;
    output [7:0]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 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**”.

```
top_tb - Notepad
File Edit Format View Help
`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);
    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;
        #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**”

simv:

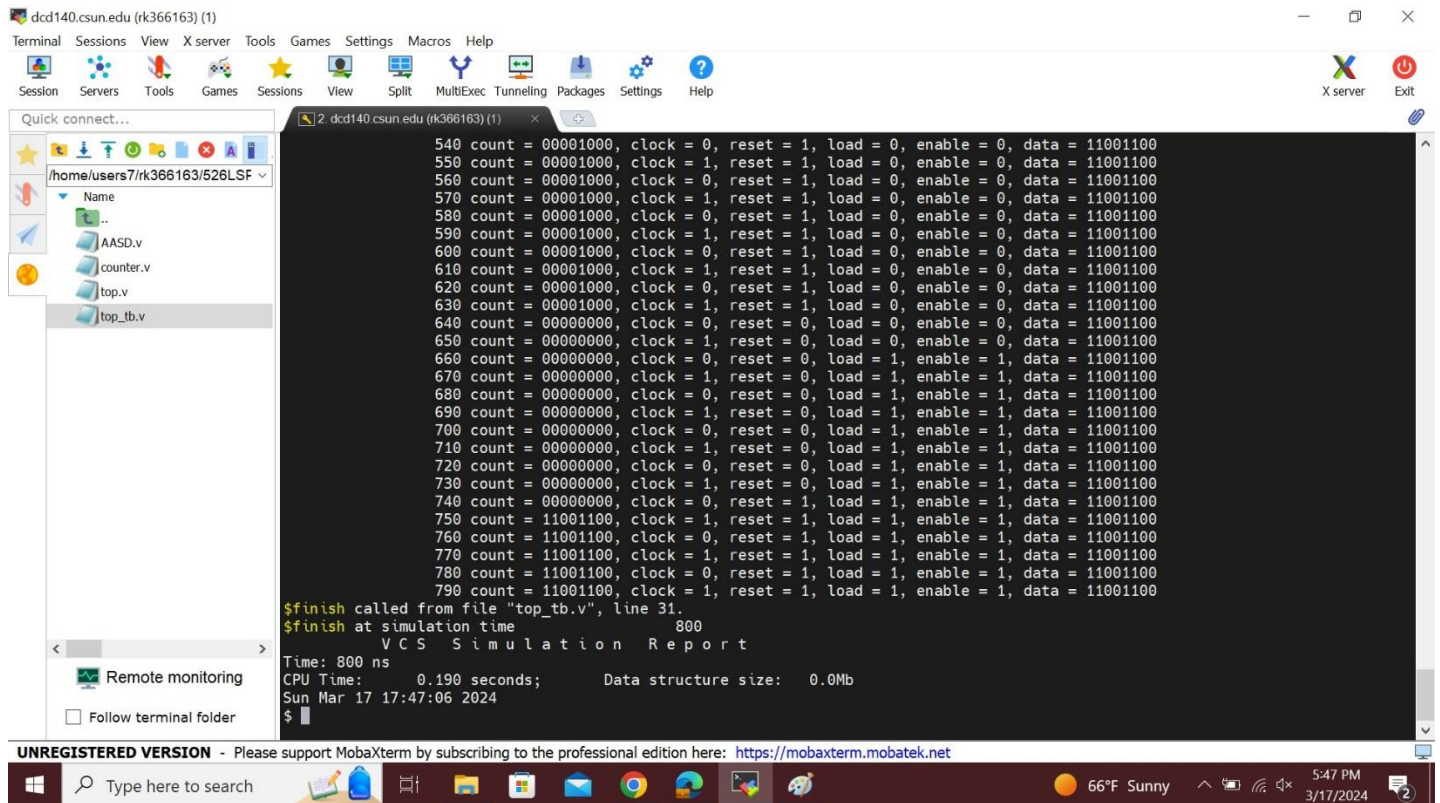
```
dcd140.csun.edu (rk366163) (1)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help

Quick connect...
/home/users7/rk366163/526LSF
Name
...
AASD.v
counter.v
top.v
top_tb.v

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
VCS Simulation Report
Time: 800 ns
CPU Time: 0.190 seconds; Data structure size: 0.0Mb
Sun Mar 17 17:44:50 2024
$
```

Log File:



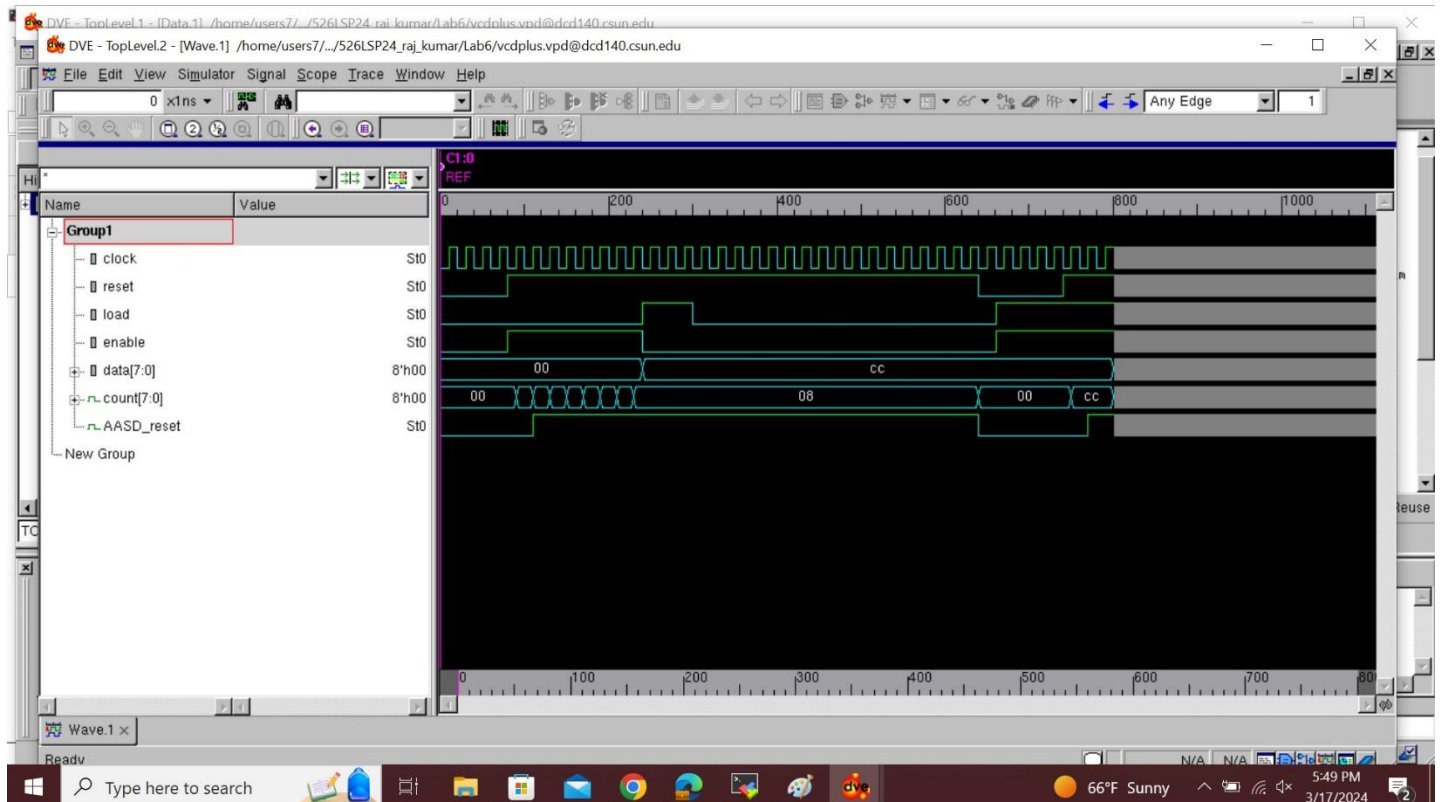
The screenshot shows a MobaXterm terminal window with a file explorer on the left. The terminal displays a list of simulation data for a counter circuit, followed by a VCS Simulation Report.

```
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
VCS Simulation Report
Time: 800 ns
CPU Time: 0.190 seconds; Data structure size: 0.0Mb
Sun Mar 17 17:47:06 2024
$
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

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.

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.

