Spring 2021

California State University, Northridge

Department of Electrical & Computer Engineering

Experiment 5

8-Bit Up Counter

February 25, 2021

ECE 526L

Written By: Jose Luis Martinez

Introduction

In this lab we will be designing an 8-bit up counter with an asynchronous assert synchronous deassert (AASD) reset. The picture in **Fig. 2** shows us the inputs and outputs of the up counter circuit. The picture in **Fig. 1** shows us how to design the AASD reset. We are also to use behavioral modeling to design all modules in this lab.

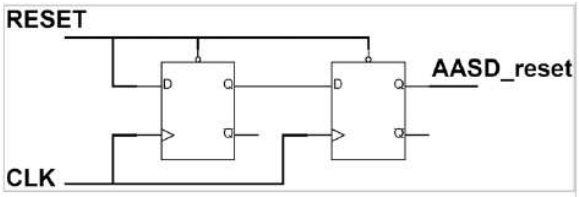

Fig. 1 AASD reset Model

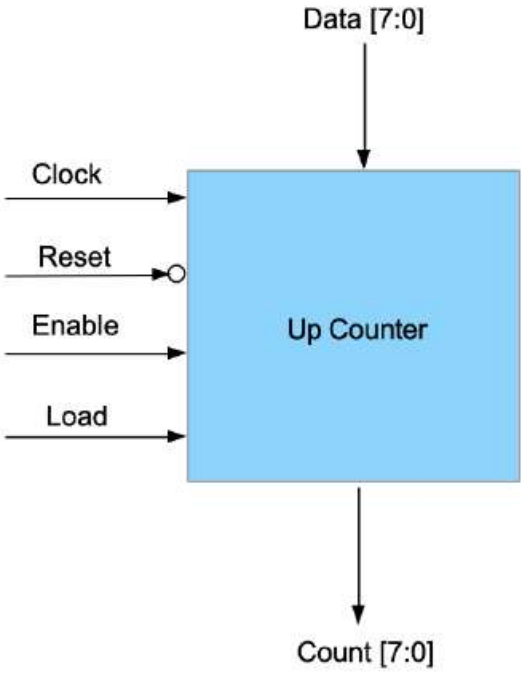

Fig. 2 Up Counter Model

Methodology

For laboratory 5 these were the steps I took to complete the experiment.

1. Made a definitions.v verilog file that contains the timescale, clock period, and strings. See **Fig. 3**.
2. Looking at **Fig. 1** and **Fig. 2** we need to create one module per each figure and then one more to combine them.
3. The model up counter in **Fig. 2** was created in the file counter.v in **Fig. 4**.
4. The model AASD reset in **Fig. 1** was created in the file AASD_reset.v in **Fig. 5**.
5. Then both modules were combined into one module called AASD_counter in **Fig. 6** where the output for the AASD_reset module is connected to the reset input of the counter module**.**
6. The top level module AASD_counter was then instantiated in the test bench file called Lab5_tb in **Fig. 7** to test out all cases required per the lab manual.

Results

```
/**********************************************************************
***                                                                 ***
*** ECE 526 L Experiment #5            Jose Luis Martinez, Spring, 2021    ***
***                                                                 ***
*** Experiment 5 - 8-Bit Counter                                    ***
***                                                                 ***
**********************************************************************
*** Filename: definitions.v Created by Jose Luis Martinez, Febuary 25, 2021 ***
***                                                                 ***
**********************************************************************/

`timescale 1ns/100ps

`define CLK_PER 20
`define DISPLAY_STRING "%d, clock = %b, reset = %b, enable = %b, load = %b, data = %d | count = %d"
```

Fig. 3 definitions.v

```
/**********************************************************************************
***                                                                             ***
*** ECE 526 L Experiment #5            Jose Luis Martinez, Spring, 2021    ***
***                                                                             ***
*** Experiment 5 - 8-Bit Counter                                               ***
***                                                                             ***
**********************************************************************************
*** Filename: counter.v     Created by Jose Luis Martinez, Febuary 25, 2021 ***
***                                                                             ***
**********************************************************************************/

`include "definitions.v"

module counter(clk, reset, enable, load, data, count);
    input clk, reset, enable, load;
    input [7:0]data;
    output reg[7:0]count;

    initial begin
        count <= 8'b0;
    end

    always@(posedge clk or negedge reset) begin

        if(!reset) begin count <= 8'b0; end
        else if(enable) begin
                if(load) begin count <= data; end
                else begin count <= count + 1; end
            end
    end

endmodule
```

Fig. 4 counter.v

```
/*****************************************************************************
***                                                                        ***
*** ECE 526 L Experiment #5            Jose Luis Martinez, Spring, 2021    ***
***                                                                        ***
*** Experiment 5 - 8-Bit Counter                                           ***
***                                                                        ***
*****************************************************************************
*** Filename: AASD_reset.v  Created by Jose Luis Martinez, Febuary 25, 2021 ***
***                                                                        ***
*****************************************************************************/

`include "definitions.v"

module AASD_reset(out, clk, reset);
    input clk, reset;
    output reg out;

    reg d;

    always@(posedge clk or negedge reset) begin
        if(!reset) begin
            out <= 1'b0;
            d <= 1'b0;
        end else begin
            d <= reset;
            out <= d;
        end
    end
endmodule
```

Fig. 5 AASD_reset.v

```
/*********************************************************************************
***                                                                            ***
*** ECE 526 L Experiment #5            Jose Luis Martinez, Spring, 2021    ***
***                                                                            ***
*** Experiment 5 - 8-Bit Counter                                              ***
***                                                                            ***
*********************************************************************************
*** Filename: AASD_counter.vCreated by Jose Luis Martinez, Febuary 25, 2021 ***
***                                                                            ***
*********************************************************************************/

`include "definitions.v"

module AASD_counter(clk, reset, enable, load, data, count);
    input clk, reset, enable, load;
    input [7:0]data;
    output [7:0]count;

    wire arst0;

        AASD_reset arst(.out(arst0), .clk(clk), .reset(reset));
        counter c1(.clk(clk), .reset(arst0), .enable(enable),
                   .load(load), .data(data), .count(count));

endmodule
```

Fig. 6 AASD_counter.v

```
/****************************************************************************
***                                                                      ***
*** ECE 526 L Experiment #5             Jose Luis Martinez, Spring, 2021  ***
***                                                                      ***
*** Experiment 5 - 8-Bit Counter                                         ***
***                                                                      ***
****************************************************************************
*** Filename: Lab5_tb.v     Created by Jose Luis Martinez, Febuary 25, 2021 ***
***                                                                      ***
****************************************************************************/

`include "definitions.v"

module Lab5_tb();
    reg clk, reset, enable, load;
    reg [7:0]data;
    wire [7:0]counter;
    AASD_counter c1(.clk(clk), .reset(reset), .enable(enable), .load(load), .data(data), .count(counter));

    initial begin
        clk <= 0;
        forever begin
            #(`CLK_PER/2) clk <= ~clk;
        end
    end

    initial begin
        $vcdpluson;
        reset = 1'b1;
        enable = 1'b1;
        load = 1'b0;
        data = 8'hfa;
        $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #5 reset = 1'b0;
        #(`CLK_PER-5) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        reset = 1'b1;
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        load = 1'b1;
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        load = 1'b0;
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        enable = 1'b0;
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #5 reset = 1'b0;
        #5 load = 1'b1;
        #(`CLK_PER-10) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        enable = 1'b1;
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        reset = 1'b1;
        load = 1'b0;
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
        #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
```

```
73          #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
74          #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
75          #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
76          #(`CLK_PER) $display(`DISPLAY_STRING, $time, clk, reset, enable, load, data, counter);
77
78          #(`CLK_PER) $finish;
79      end
80
81  endmodule
```

Fig. 7 Lab5_tb.v

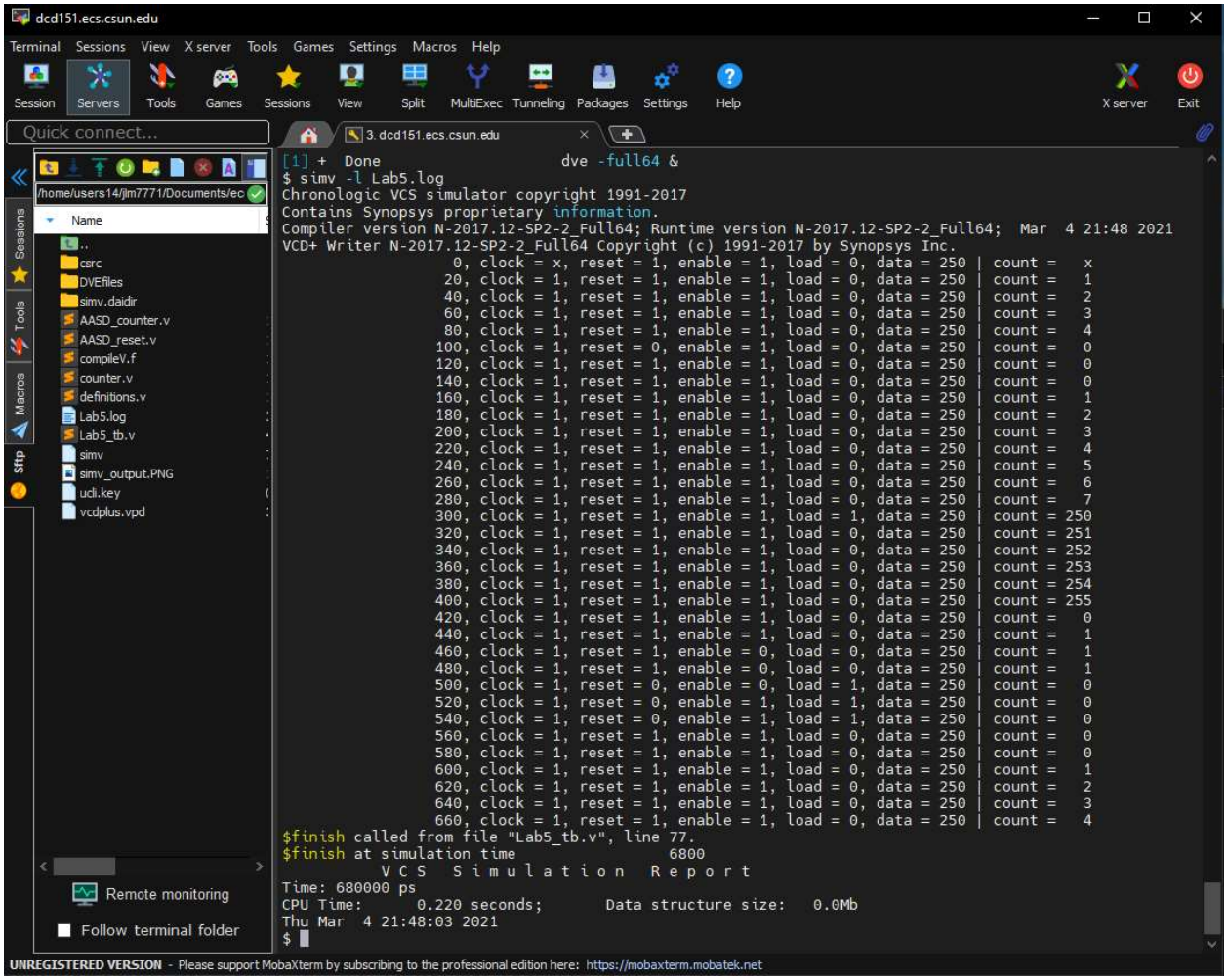

Fig. 8 simv output

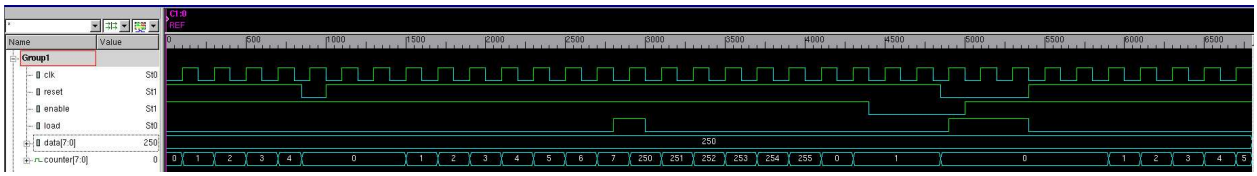

Fig. 9 8-Bit Up Counter waveform

Analysis

From our Lab5_tb module we generated a 50Mhz clock that is running at all times and then we changed the values of reset, enable, and load to test that the up counter is working correctly. We can see our results in **Fig. 8** and **Fig. 9** match for the logic that was intended in the

lab manual.

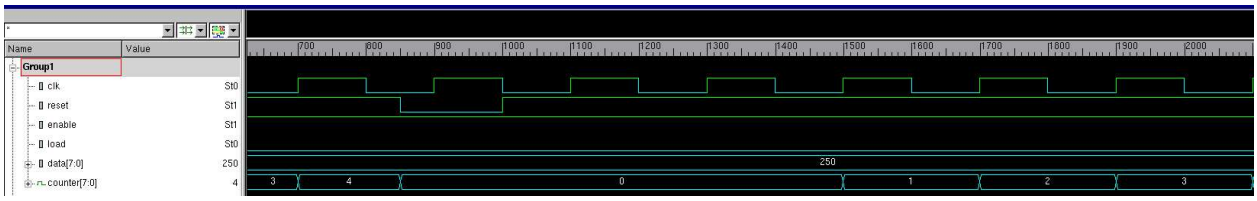

Fig. 10 Asynchronous Reset

As we can see from **Fig. 10** the reset signal turned LOW in the first quarter of the clock signal and it reset the counter to 0 immediately showing us that it is indeed an asynchronous reset. When the reset signal is brought up to logic HIGH the reset must wait 2 clock cycles before being applied thus making it synchronous deassert. The counter will start counting.

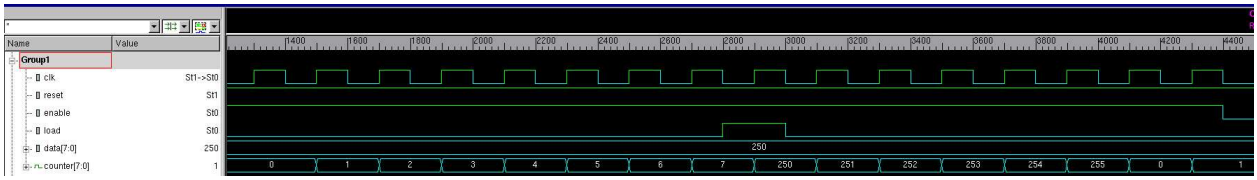

Fig. 11 Load and Counting after 255

From **Fig. 11** we can see when the load signal is logic level HIGH then the data will be loaded to the counter and begin counting from there. In this case we had 250 on the data signal so after we reached a count of 7 we made the load signal logic level HIGH for 20ns and 250 was loaded on the next posedge of the clock. The counter then counted from there until it reached 255 where it then went back to 0 and began counting from there.

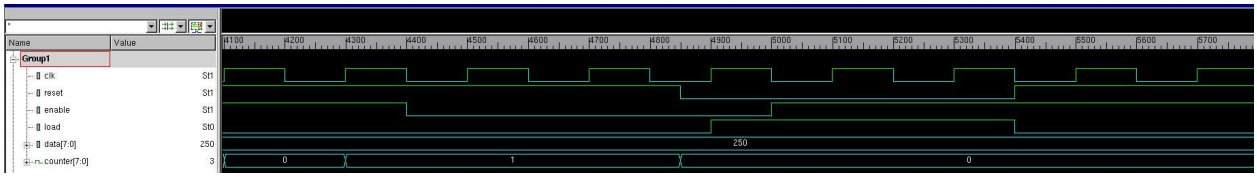

Fig. 12 Enable and Reset Priority

When the enable signal is held logic level LOW the counter should stop incrementing and hold its value. From **Fig. 12** we can see that when the enable signal is brought down to LOW the counter will hold its value of 1 and not increment. Even with the enable signal held LOW when the reset signal is brought down low it sets the counter to 0. We can also see in the waveform that when the load signal is brought HIGH the count is still 0 meaning that the reset overrides everything.

Conclusion

In conclusion, we learned how to make logic circuits using behavioral modeling, learned how to implement a 8-bit up counter, and learned about how to implement a AASD reset. From my lab results I was able to complete this lab in its entirety.

Questions

Extra Credit: If the reset was synchronous, how would the circuit behave differently? Answer this question in your lab report.

If the reset was synchronous, then whenever we would set the reset signal to logic level low the count would not immediately be set to 0. We would have to wait for the next posedge of the clock for the reset to take effect.

Academic Dishonesty

Submitting any report that is not entirely your own work is a form of academic dishonesty and will not be tolerated. Each and every lab report must include the following statement, signed and dated by the student. Lab reports without the statement will be summarily rejected.

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 will I allow anyone to copy my work.

Name (printed) Jose Luis Martinez

Name (signed) Jose Martinez Date 2/25/2021