

CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

Department of Electrical and Computer Engineering

ECE 526L

LAB – 9: Sum of Products

Written By: Raj Kumar

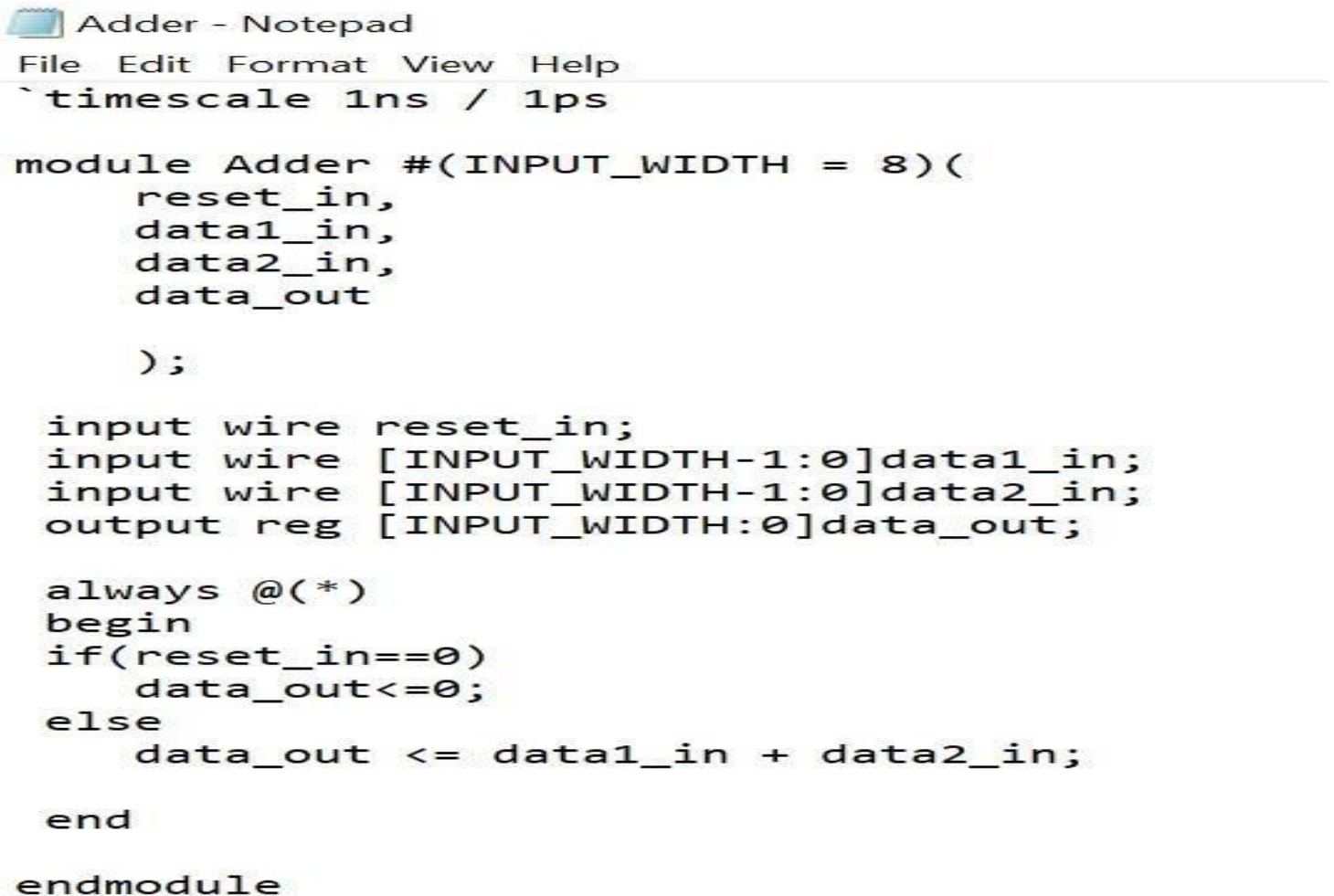
Introduction:

In this lab we will do sum of products using three levels of hierarchy, create a Verilog circuit description of this device. Data paths are to be scalable and 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 Using three levels of hierarchy, create a Verilog circuit description of this device. Data paths are to be scalable and at the lowest level we should write three design modules a register, a multiplier and an Adder. each is scalable and then save as “**Adder.v**”, “**Register.v**” and “**Multiplier.v**”.



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

module Adder #(INPUT_WIDTH = 8)(
    reset_in,
    data1_in,
    data2_in,
    data_out
);

    input wire reset_in;
    input wire [INPUT_WIDTH-1:0]data1_in;
    input wire [INPUT_WIDTH-1:0]data2_in;
    output reg [INPUT_WIDTH:0]data_out;

    always @(*)
    begin
        if(reset_in==0)
            data_out<=0;
        else
            data_out <= data1_in + data2_in;
        end
    endmodule
```

Multiplier - Notepad

File Edit Format View Help

`timescale 1ns / 1ps

```
module Multiplier #(MULTIPLICAND_WIDTH=8,MULTIPLIER_WIDTH=8)(
    reset_in,
    data_in1,
    data_in2,
    data_out

);

input wire reset_in;
input wire [MULTIPLICAND_WIDTH-1:0]data_in1;
input wire [MULTIPLIER_WIDTH-1:0]data_in2;
output reg [MULTIPLICAND_WIDTH+MULTIPLIER_WIDTH-1:0]data_out;

always @(*)
begin
    if(reset_in==0)
        data_out<=0;
    else
        data_out <= data_in1*data_in2;
    end
endmodule
```

Register - Notepad

File Edit Format View Help

`timescale 1ns / 1ps

```
module Register #(INPUT_WIDTH=8)(
    reset_in,
    clock_in,
    data_in,
    data_out

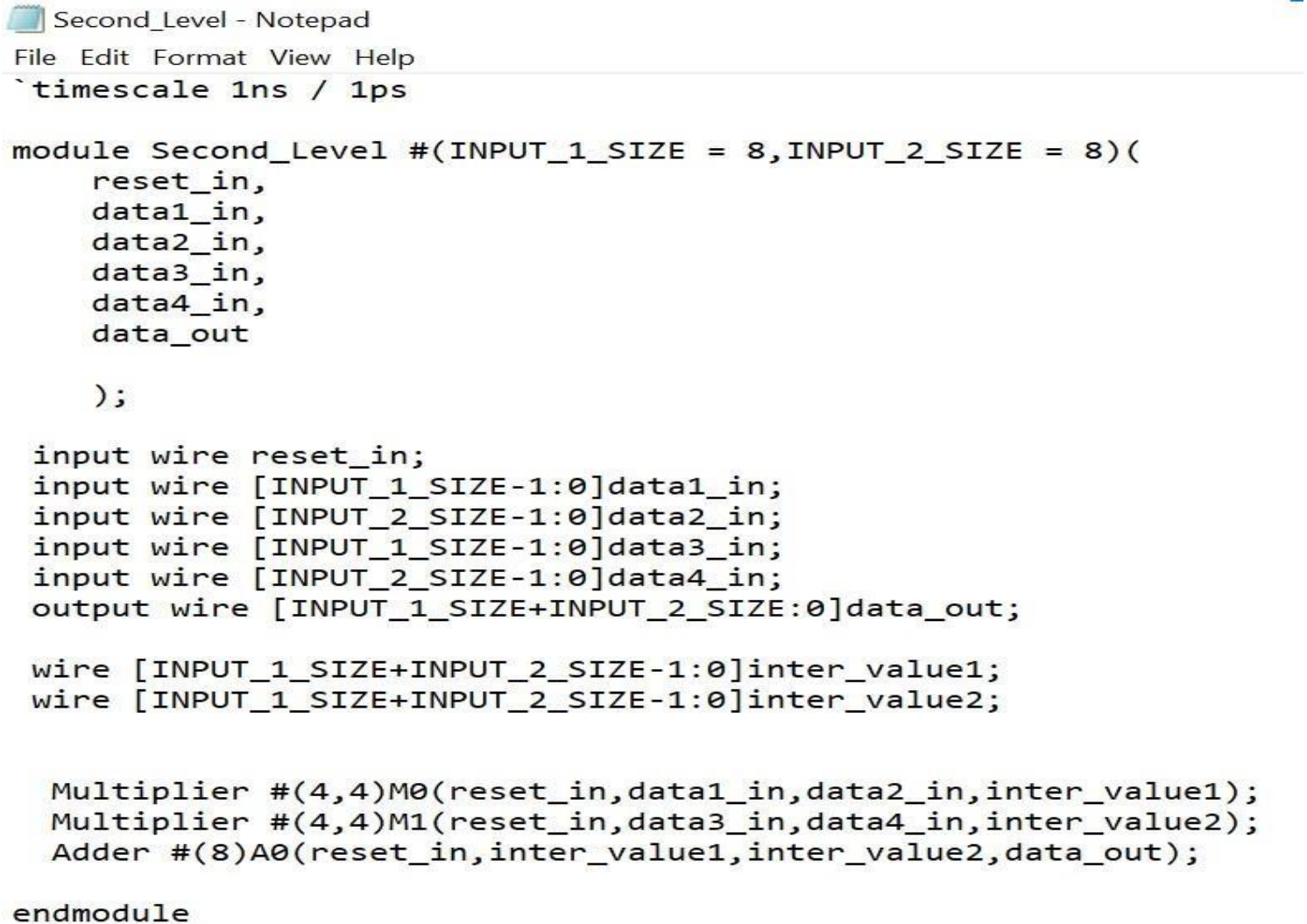
);

input wire reset_in;
input wire clock_in;
input wire [INPUT_WIDTH-1:0]data_in;
output reg [INPUT_WIDTH-1:0]data_out;

always @(posedge clock_in or negedge reset_in)
begin
    if(reset_in==0)
        data_out<=0;
    else
        data_out <= data_in;
    end
endmodule
```

This above are figures of three design modules of store it in new file, also in folder “**Lab9**”.

Now we should create a second level consists of one instance of an adder, two instances of multipliers and two instances of registers and save it as “**second_level.v**”.



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

module Second_Level #(INPUT_1_SIZE = 8,INPUT_2_SIZE = 8)(
    reset_in,
    data1_in,
    data2_in,
    data3_in,
    data4_in,
    data_out

);

input wire reset_in;
input wire [INPUT_1_SIZE-1:0]data1_in;
input wire [INPUT_2_SIZE-1:0]data2_in;
input wire [INPUT_1_SIZE-1:0]data3_in;
input wire [INPUT_2_SIZE-1:0]data4_in;
output wire [INPUT_1_SIZE+INPUT_2_SIZE:0]data_out;

wire [INPUT_1_SIZE+INPUT_2_SIZE-1:0]inter_value1;
wire [INPUT_1_SIZE+INPUT_2_SIZE-1:0]inter_value2;

Multiplier #(4,4)M0(reset_in,data1_in,data2_in,inter_value1);
Multiplier #(4,4)M1(reset_in,data3_in,data4_in,inter_value2);
Adder #(8)A0(reset_in,inter_value1,inter_value2,data_out);

endmodule
```

The above fig is “**second_level.v**”

Now we should create The top design level has two instances of the second level and one more adder the top level is to be instantiated in test benches and save it as “**top_level.v**”.

```

*Top - Notepad
File Edit Format View Help
`timescale 1ns / 1ps

module Top#(DATA_WIDTH=4,COEFFICIENT_WIDTH=4)(
reset_in,
clock_in,
C0_in,
C1_in,
C2_in,
C3_in,
Data_in,
Data_out

);

input wire reset_in;
input wire clock_in;
input wire [COEFFICIENT_WIDTH-1:0]C0_in;
input wire [COEFFICIENT_WIDTH-1:0]C1_in;
input wire [COEFFICIENT_WIDTH-1:0]C2_in;
input wire [COEFFICIENT_WIDTH-1:0]C3_in;
input wire [DATA_WIDTH-1:0]Data_in;
output wire [DATA_WIDTH+COEFFICIENT_WIDTH :0]Data_out;

wire [DATA_WIDTH-1:0]Data_out1;
wire [DATA_WIDTH-1:0]Data_out2;
wire [DATA_WIDTH-1:0]Data_out3;
wire [DATA_WIDTH-1:0]Data_out4;
wire [DATA_WIDTH+COEFFICIENT_WIDTH:0]interData_out1;
wire [DATA_WIDTH+COEFFICIENT_WIDTH:0]interData_out2;

Register #(DATA_WIDTH)R0(reset_in,clock_in, Data_in,Data_out1);
Register #(DATA_WIDTH)R1(reset_in,clock_in, Data_out1,Data_out2);
Register #(DATA_WIDTH)R2(reset_in,clock_in, Data_out2,Data_out3);
Register #(DATA_WIDTH)R3(reset_in,clock_in, Data_out3,Data_out4);
Second_Level #(DATA_WIDTH,COEFFICIENT_WIDTH)SL1(reset_in, Data_out1,C0_in,Data_out2,C1_in,interData_out1);
Second_Level #(DATA_WIDTH,COEFFICIENT_WIDTH)SL2(reset_in, Data_out3,C2_in,Data_out4,C3_in,interData_out2);
Adder #(DATA_WIDTH+COEFFICIENT_WIDTH+1)A1(reset_in,interData_out1,interData_out2,Data_out);

endmodule

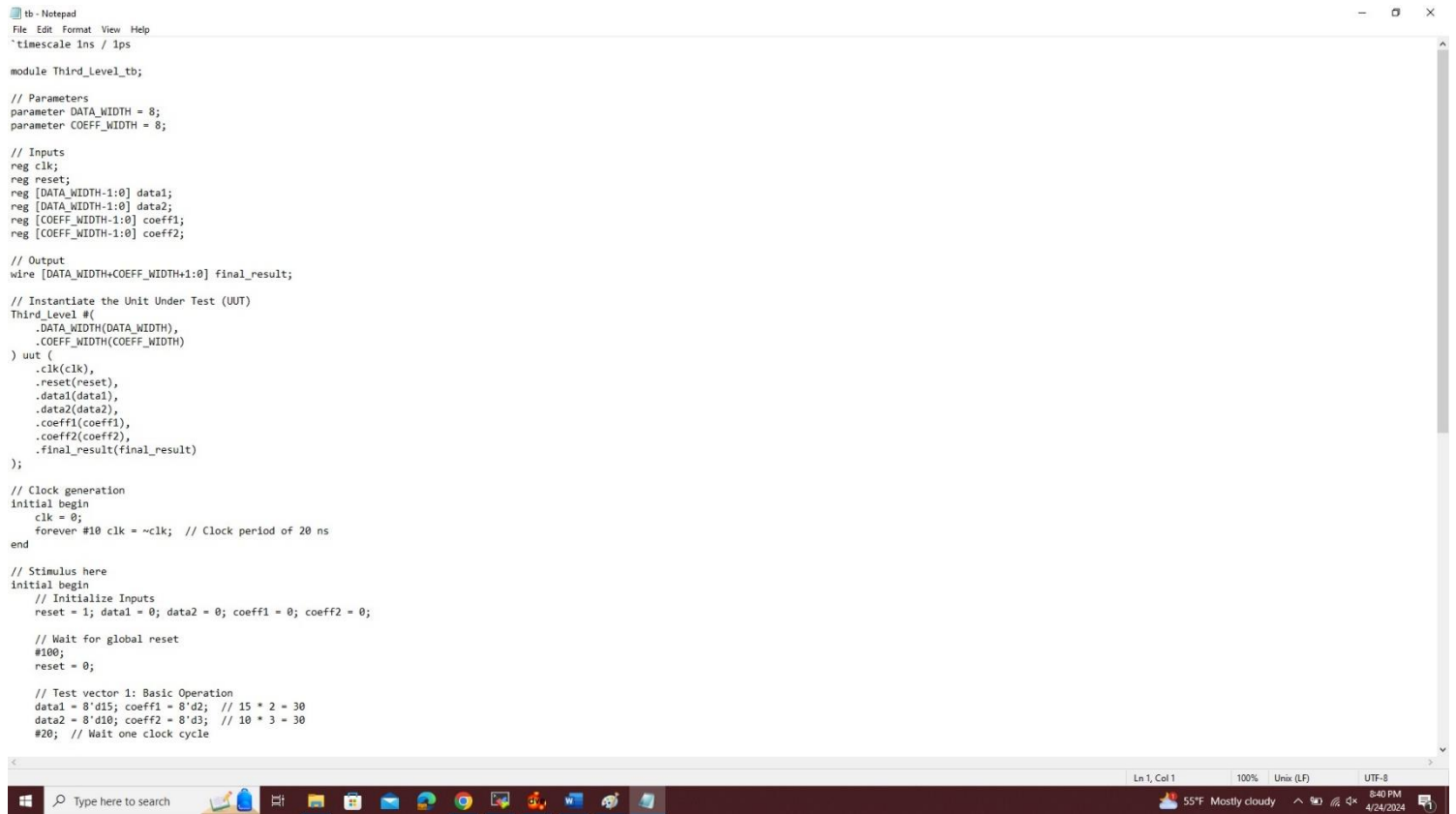
```

The above is fig of “**top.v**”.

Now we should write a non-exhaustive testbench to test the functionality of your device according to the conditions given in pdf an Write an exhaustive testbench to test your device. Compare your output to the output of a non- hierarchical sum of products.

To prove your exhaustive testbench will find an error, use a force assignment to override the behavioral code’s sum sometime after quarter of the test vectors had been applied. Use the `ifdef and `endif compiler directives to include or exclude the code that forces an error. I used the comment line to represent the error free code and without commenting the statement then it is a error force code.

This is the testbench code for both error free and error force code and save it as “**tb.v**”.



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

module Third_Level_tb;

// Parameters
parameter DATA_WIDTH = 8;
parameter COEFF_WIDTH = 8;

// Inputs
reg clk;
reg reset;
reg [DATA_WIDTH-1:0] data1;
reg [DATA_WIDTH-1:0] data2;
reg [COEFF_WIDTH-1:0] coeff1;
reg [COEFF_WIDTH-1:0] coeff2;

// Output
wire [DATA_WIDTH+COEFF_WIDTH+1:0] final_result;

// Instantiate the Unit Under Test (UUT)
Third_Level #(
    .DATA_WIDTH(DATA_WIDTH),
    .COEFF_WIDTH(COEFF_WIDTH)
) uut (
    .clk(clk),
    .reset(reset),
    .data1(data1),
    .data2(data2),
    .coeff1(coeff1),
    .coeff2(coeff2),
    .final_result(final_result)
);

// Clock generation
initial begin
    clk = 0;
    forever #10 clk = ~clk; // Clock period of 20 ns
end

// Stimulus here
initial begin
    // Initialize Inputs
    reset = 1; data1 = 0; data2 = 0; coeff1 = 0; coeff2 = 0;

    // Wait for global reset
    #100;
    reset = 0;

    // Test vector 1: Basic Operation
    data1 = 8'd15; coeff1 = 8'd2; // 15 * 2 = 30
    data2 = 8'd10; coeff2 = 8'd3; // 10 * 3 = 30
    #20; // Wait one clock cycle
end
```

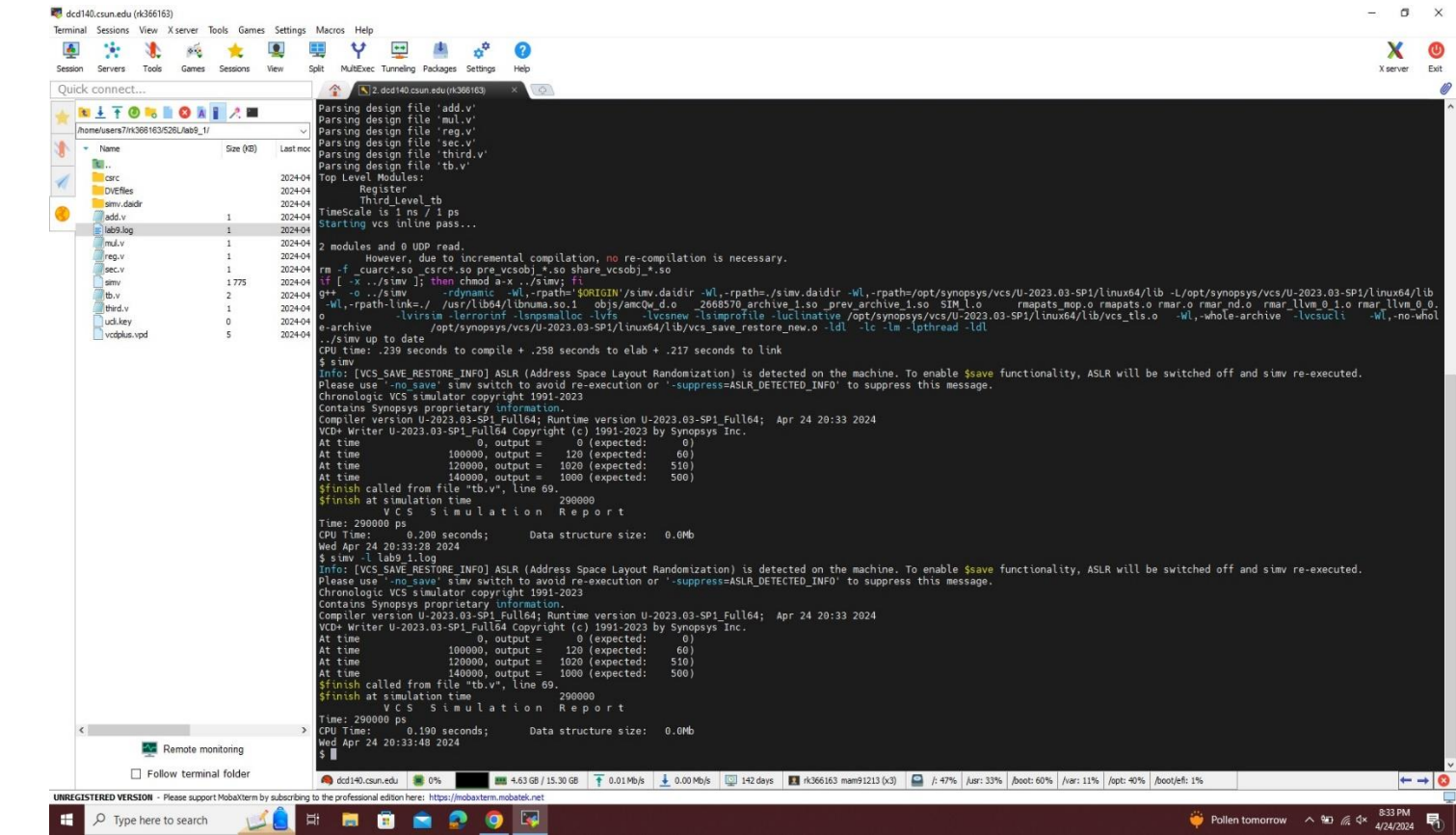
Then use the command as follows:

“**vcs -debug_access+all Adder.v Register.v Multiplier.v Second_Level.v Top.v Top_TB.v**”

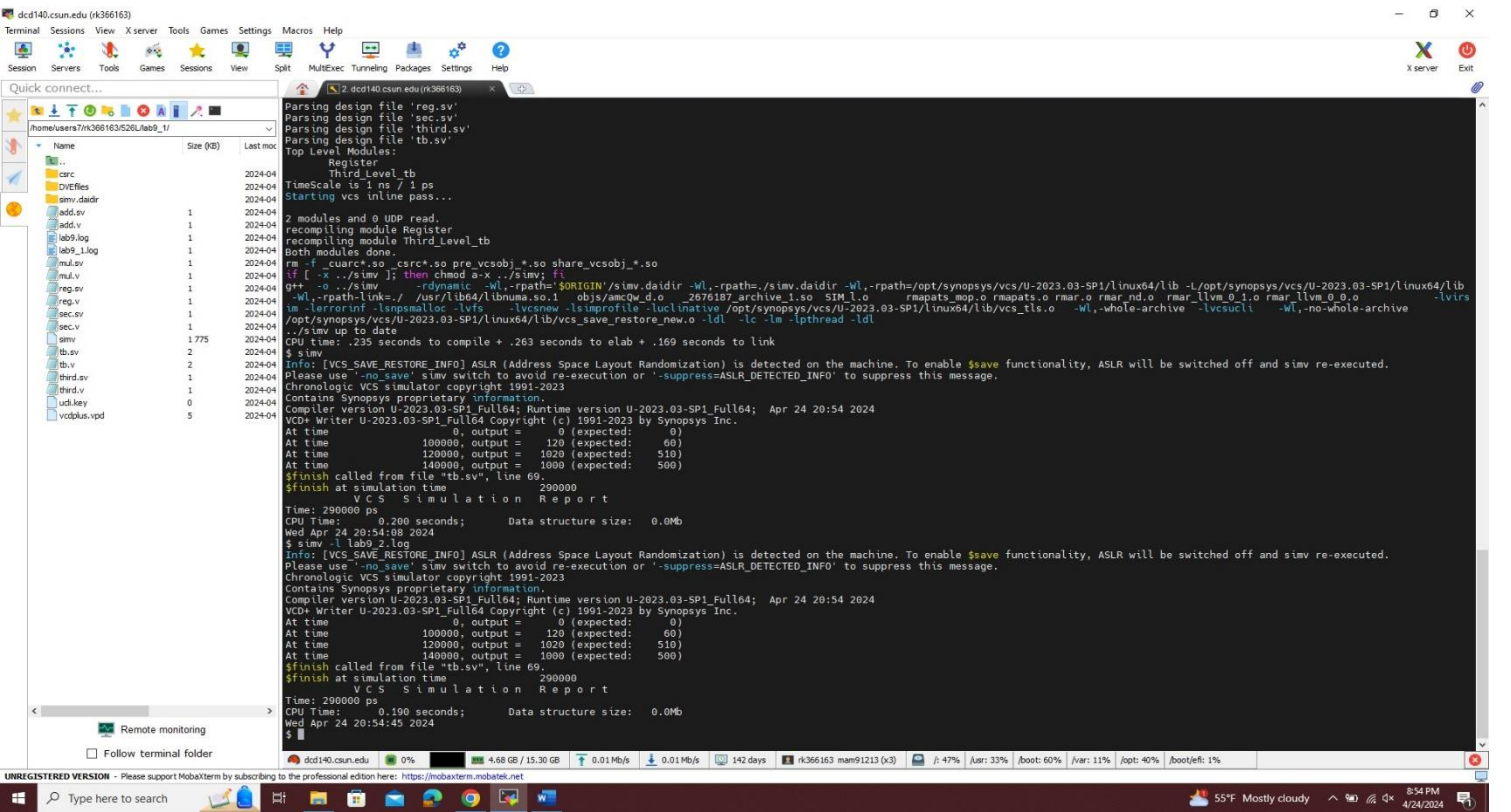
Run the **simv** two times one with comment and one with without comment.

If we don’t see any error messages if everything is typecorrectly then output should look like below figure.

simv:



simv2:

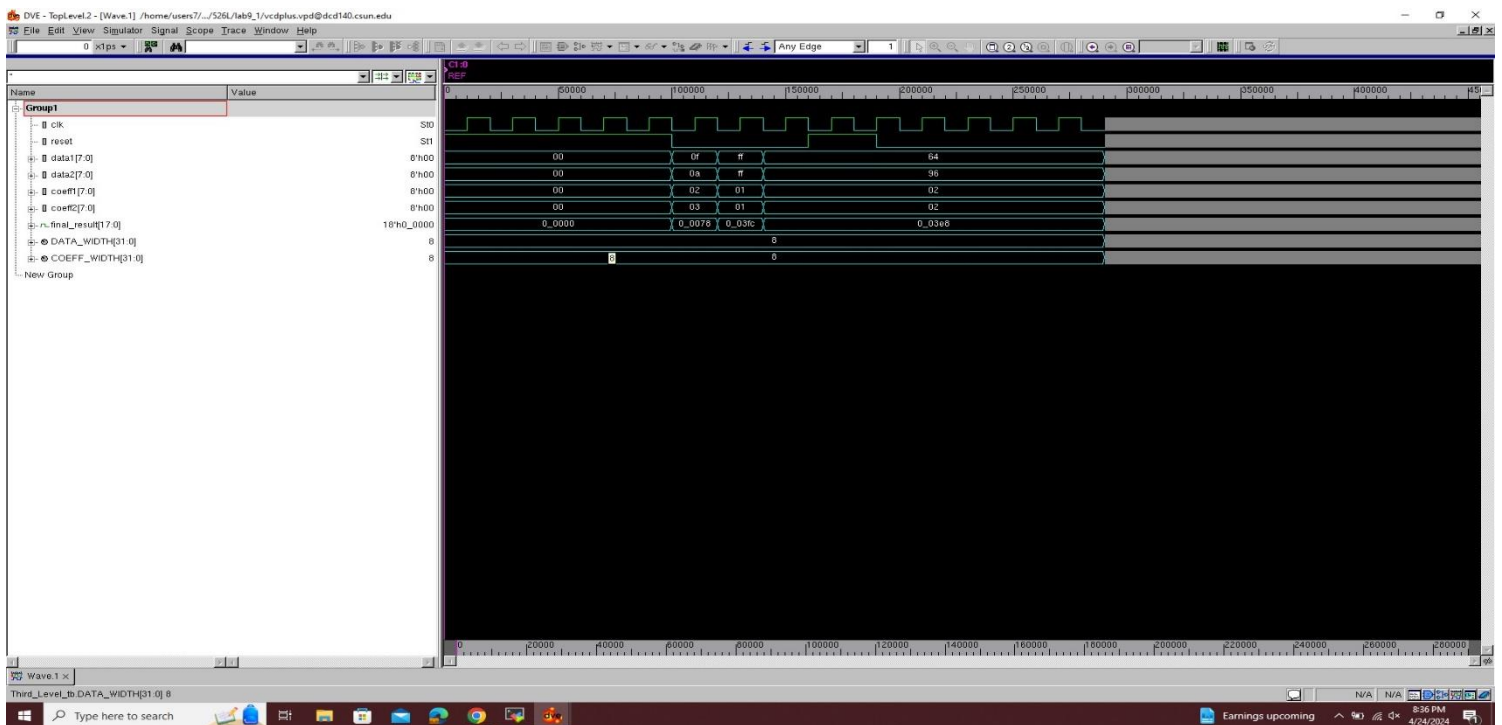


To see the wave form first type “dve” command line then we will get the blank simulation window then go to file and open database then choose **vcdplus.vpd** and open the file.

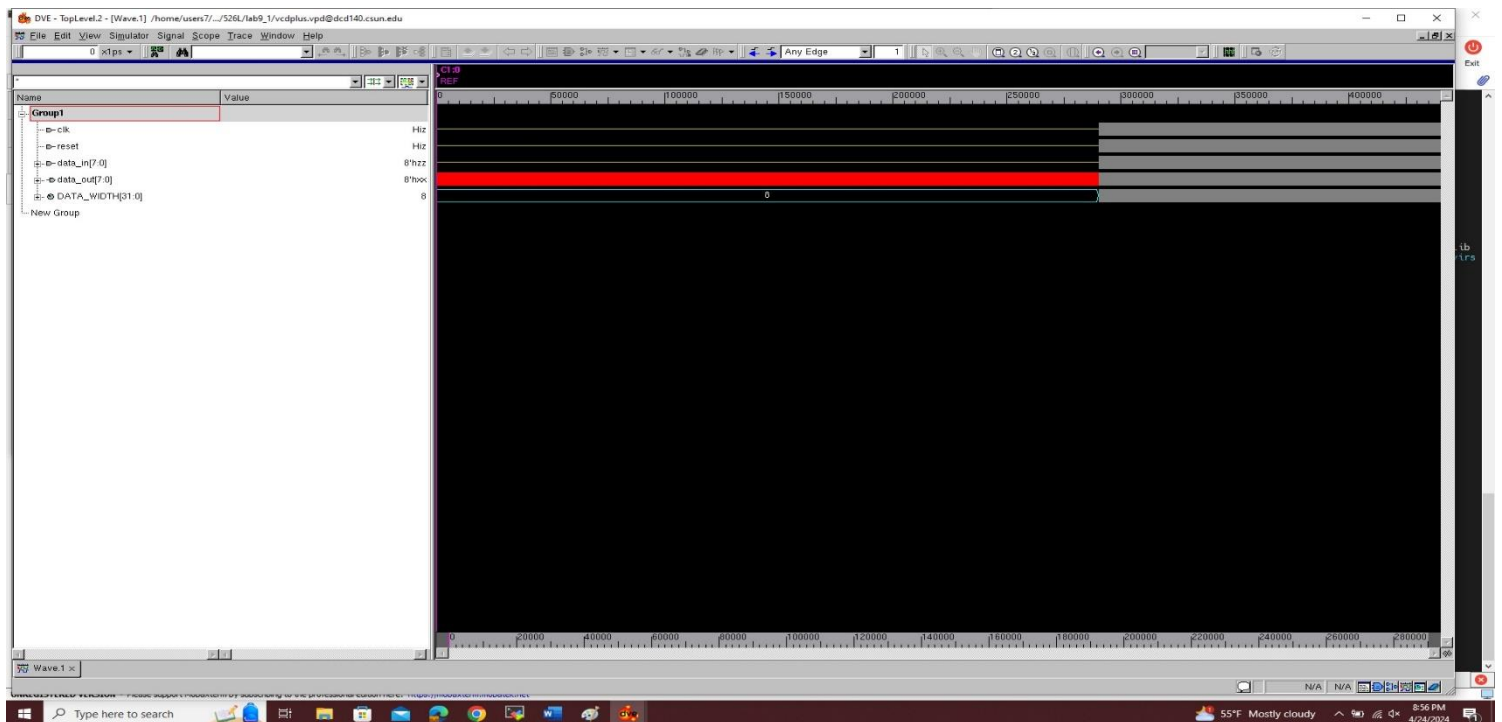
Then finally select all the signals with the mouse, then right click and select to “**add to wave->new wave view**”.

Then we will get the waveform as shown in below figure;

Wave form -1



Wave form -2



Analysis:

In this lab we have written codes for the sum of products with using the three levels of hierarchy by using the **adder, register, multiplexer, second level, top and testbench** based on the top and instructions provided in the pdf. we created two different outputs for this lab based on the error free code and error forced codes more over we used \$monitor on and off and use the ``ifdef` and ``endif` compiler directives to include or exclude the code that forces an error for us .so based on the comment line on the define error statement the data-out values changes.

Conclusion:

In concluding the sum of product lab implemented in SystemVerilog, we successfully designed, simulated, and verified a digital logic circuit that calculates the sum of products for given inputs. This exercise provided a practical application of Boolean algebra and reinforced the core concepts of digital design, including the use of logic gates and the implementation of combinational logic.

Throughout the lab, we leveraged the powerful features of SystemVerilog such as strong typing, concise syntax, and advanced modeling capabilities, which enhanced the development process and improved our ability to debug and validate the circuit. The simulation results confirmed the accuracy of our design, as the outputs matched the expected values for all input combinations.

The process of creating test benches was invaluable, as it allowed us to methodically test each component of the circuit and ensure that all possible scenarios were covered. This not only solidified our understanding of the design but also highlighted the importance of thorough testing in the field of digital electronics.

Moving forward, the skills and knowledge gained from this lab can be applied to more complex digital systems, preparing us for advanced projects that involve multi-layered logic circuits and integrated digital systems. This practical experience has laid a solid foundation for further exploration and innovation in the realm of digital design using SystemVerilog.

Lab Report Question:

1.Explain why you included or did not include a reset in your design?

Ans: I used the reset signal in the code for our design because in this example, the circuit is started in a predictable manner by using the reset signal to initialize the register when it is asserted. In the absence of the reset condition, the expression is applied to the input data. Based on the internal state, the output result can have more logic added as needed.

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 04/21/2024