Spring 2021

California State University, Northridge

Department of Electrical & Computer Engineering

Experiment 6B

Sum of Products

April 3, 2021

ECE 526L

Written By: Jose Luis Martinez

Introduction

In this lab we are tasked with creating a Sum of Products with verilog. The inputs of the circuit are to be scalable meaning that we should be able to just change a parameter and have it work for that input size. Our default size for the inputs should be 4 bits wide. The lowest level of the design should just be 3 modules, which are scalable multiplier, register, and adder. At the second level we will have two registers, two multipliers and one adder. For the second level we will have two instantiations of the first level and an adder. **Fig. 1** shows the Sum of Products circuit in diagram form.

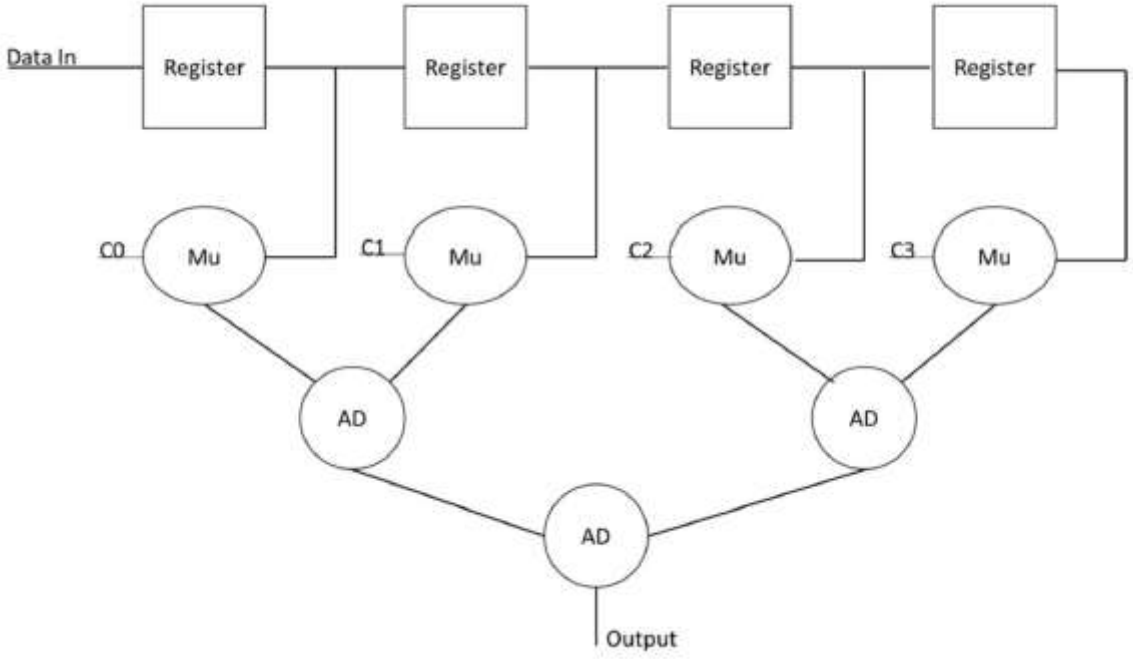

Fig. 1 Sum of Products Diagram

Methodology

I followed these steps to complete the lab:

1. The three lowest level modules were created first.
    a. Scalable register was made, shown in **Fig. 2.**
    b. Scalable multiplier was made, shown in **Fig. 3.**
    c. Scalable adder was made, shown in **Fig. 4.**
2. The second level module was created.
    a. Second level module was made, shown in **Fig. 5.**
        i. Was made using 2 scalable registers, 2 scalable multipliers, and 1 scalable adder.
        ii. The input D_in is connected to the input of the first register.
        iii. The C0 and C1 inputs are connected to one multiplier each.
        iv. The output is the result of the adder.
3. The top level module was created.
    a. Top level module was made, as shown in **Fig. 6.**
        i. Was made using 2 second level modules and one adder adding the outputs of each second level module.

4. The testbench file was created to test and ensure our top level module was working properly.
    a. A non exhaustive test bench was created as shown in **Fig. 7.**
        i. A small test vector of about a dozen elements was tested.
    b. An exhaustive test bench was created as shown in **Fig. 8.**
        i. All possible values of D_IN were tested and the answer was compared to a non-hierarchical output in order to ensure that my design is working properly.
        ii. An `ifdef compiler directive was used to force and error.

Results/Verilog Files

```
/*************************************************************************
***                                                                    ***
*** ECE 526 L Experiment #6B           Jose Luis Martinez, Spring, 2021  ***
***                                                                    ***
*** Experiment 6B - Sum of Products                                    ***
***                                                                    ***
*************************************************************************
*** Filename: sca_reg.v        Created by Jose Luis Martinez, April 1, 2021 ***
***                                                                    ***
*************************************************************************/

`timescale 1ns/100ps

module sca_reg(OUT, A, RST, CLK);
    parameter SIZE = 4;
    input CLK, RST;
    input [SIZE-1:0] A;
    output reg [SIZE-1:0] OUT;

    always@(posedge CLK or negedge RST) begin
        if(!RST) begin
            OUT <= 0;
        end else begin
            OUT <= A;
        end
    end
endmodule
```

Fig. 2 sca_reg.v

```
/*********************************************************************************
***                                                                            ***
*** ECE 526 L Experiment #6B            Jose Luis Martinez, Spring, 2021     ***
***                                                                            ***
*** Experiment 6B - Sum of Products                                           ***
***                                                                            ***
*********************************************************************************
*** Filename: sca_mult.v       Created by Jose Luis Martinez, April 1, 2021 ***
***                                                                            ***
*********************************************************************************/

`timescale 1ns/100ps

module sca_mult(OUT, A, B);
    parameter SIZE = 4;
    input [SIZE-1:0] A;
    input [SIZE-1:0] B;
    output [(SIZE*2)-1:0] OUT;

    assign OUT = A * B;
endmodule
```

Fig. 3 sca_mult.v

```
/*********************************************************************************
***                                                                            ***
*** ECE 526 L Experiment #6B            Jose Luis Martinez, Spring, 2021     ***
***                                                                            ***
*** Experiment 6B - Sum of Products                                           ***
***                                                                            ***
*********************************************************************************
*** Filename: sca_add.v        Created by Jose Luis Martinez, April 1, 2021 ***
***                                                                            ***
*********************************************************************************/

`timescale 1ns/100ps

module sca_add(OUT, A, B);
    parameter SIZE = 8;
    input [SIZE-1:0] A;
    input [SIZE-1:0] B;
    output [SIZE:0] OUT;

    assign OUT = A + B;
endmodule
```

Fig. 4 sca_add.v

```
/*****************************************************************************
***                                                                        ***
*** ECE 526 L Experiment #6B           Jose Luis Martinez, Spring, 2021    ***
***                                                                        ***
*** Experiment 6B - Sum of Products                                        ***
***                                                                        ***
*****************************************************************************
*** Filename: lvl2.v          Created by Jose Luis Martinez, April 1, 2021 ***
***                                                                        ***
*****************************************************************************/

`timescale 1ns/100ps

module lvl2(SUM_OUT, REG_OUT, A, B, C, CLK, RST);
    parameter SIZE = 4;

    input CLK, RST;
    input [SIZE-1:0] A;
    input [SIZE-1:0] B;
    input [SIZE-1:0] C;
    output [SIZE-1:0] REG_OUT;
    output [SIZE*2:0] SUM_OUT;

    wire [SIZE-1:0] r1r2;
    wire [(SIZE*2)-1:0] prod1;
    wire [(SIZE*2)-1:0] prod2;

    sca_reg #(.SIZE(SIZE)) sr1(.OUT(r1r2), .A(A), .RST(RST), .CLK(CLK));
    sca_reg #(.SIZE(SIZE)) sr2(.OUT(REG_OUT), .A(r1r2), .RST(RST), .CLK(CLK));

    sca_mult #(.SIZE(SIZE)) mult1(.OUT(prod1), .A(r1r2), .B(B));
    sca_mult #(.SIZE(SIZE)) mult2(.OUT(prod2), .A(REG_OUT), .B(C));

    sca_add #(.SIZE(SIZE*2)) add1(.OUT(SUM_OUT), .A(prod1), .B(prod2));
endmodule
```

Fig. 5 lvl2.v

```
/*********************************************************************************
***                                                                            ***
*** ECE 526 L Experiment #6B           Jose Luis Martinez, Spring, 2021    ***
***                                                                            ***
*** Experiment 6B - Sum of Products                                           ***
***                                                                            ***
*********************************************************************************
*** Filename: top_lvl.v        Created by Jose Luis Martinez, April 1, 2021 ***
***                                                                            ***
*********************************************************************************/

`timescale 1ns/100ps

module top_lvl(OUT, D_IN, C0, C1, C2, C3, RST, CLK);
    parameter SIZE = 4;

    input RST, CLK;
    input [SIZE-1:0] D_IN;
    input [SIZE-1:0] C0;
    input [SIZE-1:0] C1;
    input [SIZE-1:0] C2;
    input [SIZE-1:0] C3;
    output [(SIZE*2)+1:0] OUT;

    wire [SIZE*2:0] sum1;
    wire [SIZE*2:0] sum2;
    wire [SIZE-1:0] rout1;
    wire [SIZE-1:0] rout2;

    lvl2 #(.SIZE(SIZE)) lvl2_1(.SUM_OUT(sum1), .REG_OUT(rout1),
                               .A(D_IN), .B(C0), .C(C1), .CLK(CLK), .RST(RST));
    lvl2 #(.SIZE(SIZE)) lvl2_2(.SUM_OUT(sum2), .REG_OUT(rout2),
                               .A(rout1), .B(C2), .C(C3), .CLK(CLK), .RST(RST));

    sca_add #(.SIZE((SIZE*2)+1)) add1(.OUT(OUT), .A(sum1), .B(sum2));

endmodule
```

Fig. 6 top_lvl.v

```
/*********************************************************************************
***                                                                            ***
*** ECE 526 L Experiment #6B            Jose Luis Martinez, Spring, 2021     ***
***                                                                            ***
*** Experiment 6B - Sum of Products                                           ***
***                                                                            ***
*********************************************************************************
*** Filename: lab_6b_nonex_tb.vCreated by Jose Luis Martinez, April 1, 2021 ***
***                                                                            ***
*********************************************************************************/

`timescale 1ns/100ps

`define CLK_PER 20
`define SDISPLAY "%d, D_IN = %h, C0 = %h, C1 = %h, C2 = %h, C3 = %h, RST = %b, CLK = %b| C

module lab_6b_nonex_tb();
    parameter SIZE = 4;

    reg [SIZE-1:0] D_IN;
    reg [SIZE-1:0] C0;
    reg [SIZE-1:0] C1;
    reg [SIZE-1:0] C2;
    reg [SIZE-1:0] C3;
    reg RST, CLK;
    wire [(SIZE*2)+1:0] OUT;

    top_lvl #(.SIZE(SIZE)) tp1(.OUT(OUT), .D_IN(D_IN), .C0(C0), .C1(C1), .C2(C2), .C3(C3),

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

    initial begin
        $vcdpluson;
        D_IN <= 4'h4;
        RST <= 1'b1;
        C0 <= 4'h2;
        C1 <= 4'h3;
        C2 <= 4'h4;
        C3 <= 4'h9;
        $display(`SDISPLAY, $time, D_IN, C0, C1, C2, C3, RST, CLK, OUT);

        #(`CLK_PER)
        D_IN <= 4'h0;
```

```
        $display(`SDISPLAY, $time, D_IN, C0, C1, C2, C3, RST, CLK, OUT);

        #(`CLK_PER)
        D_IN <= 4'h1;
        $display(`SDISPLAY, $time, D_IN, C0, C1, C2, C3, RST, CLK, OUT);

        #(`CLK_PER)
        D_IN <= 4'hA;
        $display(`SDISPLAY, $time, D_IN, C0, C1, C2, C3, RST, CLK, OUT);

        #(`CLK_PER)
        D_IN <= 4'hB;
        $display(`SDISPLAY, $time, D_IN, C0, C1, C2, C3, RST, CLK, OUT);

        #(`CLK_PER)
```

```
64          D_IN <= 4'h2;
65          $display(`SDISPLAY, $time, D_IN, C0, C1, C2, C3, RST, CLK, OUT);
66
67          #(`CLK_PER)
68          D_IN <= 4'hF;
69          $display(`SDISPLAY, $time, D_IN, C0, C1, C2, C3, RST, CLK, OUT);
70
71          #(`CLK_PER)
72          D_IN <= 4'h8;
73          $display(`SDISPLAY, $time, D_IN, C0, C1, C2, C3, RST, CLK, OUT);
74
75          #(`CLK_PER)
76          D_IN <= 4'h0;
77          $display(`SDISPLAY, $time, D_IN, C0, C1, C2, C3, RST, CLK, OUT);
78
79          #(`CLK_PER)
80          D_IN <= 4'hD;
81          $display(`SDISPLAY, $time, D_IN, C0, C1, C2, C3, RST, CLK, OUT);
82
83          #(`CLK_PER)
84          $display(`SDISPLAY, $time, D_IN, C0, C1, C2, C3, RST, CLK, OUT);
85          $finish;
86      end
87
88  endmodule
```

Fig. 7 lab_6b_nonex_tb.v

```
/*********************************************************************
***                                                                ***
*** ECE 526 L Experiment #6B            Jose Luis Martinez, Spring, 2021    ***
***                                                                ***
*** Experiment 6B - Sum of Products                                ***
***                                                                ***
*********************************************************************
*** Filename: lab_6b_ex_tb.v   Created by Jose Luis Martinez, April 1, 2021 ***
***                                                                ***
*********************************************************************/

`timescale 1ns/100ps

`define CLK_PER 20
`define SDISPLAY "%d, D_IN = %d, C0 = %d, C1 = %d, C2 = %d, C3 = %d, RST = %b, CLK = %b| C

module lab_6b_ex_tb();
    parameter SIZE = 4;

    reg [SIZE-1:0] D_IN;
    reg [SIZE-1:0] C0;
    reg [SIZE-1:0] C1;
    reg [SIZE-1:0] C2;
    reg [SIZE-1:0] C3;
    reg [(SIZE*2)+1:0] E_OUT;
    reg [3:0] D_ARR [SIZE-1:0];
    reg RST, CLK;
    wire [(SIZE*2)+1:0] OUT;
    integer i;

    top_lvl #(.SIZE(SIZE)) tp1(.OUT(OUT), .D_IN(D_IN), .C0(C0), .C1(C1), .C2(C2), .C3(C3),

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

    initial $monitorb(`SDISPLAY, $time, D_IN, C0, C1, C2, C3, RST, CLK, OUT, E_OUT);

    initial begin
        $vcdpluson;
        RST <= 1'b1;
        D_IN <= 4'h0;
        D_ARR[0] <= 4'h0;
        C0 <= 4'h2;
        C1 <= 4'h3;
        C2 <= 4'h4;
        C3 <= 4'h9;

        #(`CLK_PER*6)
        $display("\t\t\tFirst set of test vectors.\n\n");
        $monitoroff;
        #(`CLK_PER*4)
        $monitoron;
        #(`CLK_PER*6)
        $display("\t\t\tLast set of test vectors.\n\n");
        $finish;
    end
```

```
61
62      always@(posedge CLK) begin
63          D_IN <= D_IN + 1'b1;
64
65          E_OUT <= C0*D_ARR[0] + C1*D_ARR[1] + C2*D_ARR[2] + C3*D_ARR[3];
66
67          #(1)
68          D_ARR[3] <= D_ARR[2];
69          D_ARR[2] <= D_ARR[1];
70          D_ARR[1] <= D_ARR[0];
71          D_ARR[0] <= D_IN;
72
73          if(E_OUT != OUT) begin
74              $display("\t\t\tError! Outputs do not match!\n");
75              $finish;
76          end
77      end
78
79      `ifdef ERROR
80      initial begin
81          #(`CLK_PER*9)
82          $monitoron;
83          #(`CLK_PER*6) force OUT = 4'b1x0x;
84      end
85      `endif
86
87  endmodule
```

Fig. 8 lab_6b_ex_tb.v

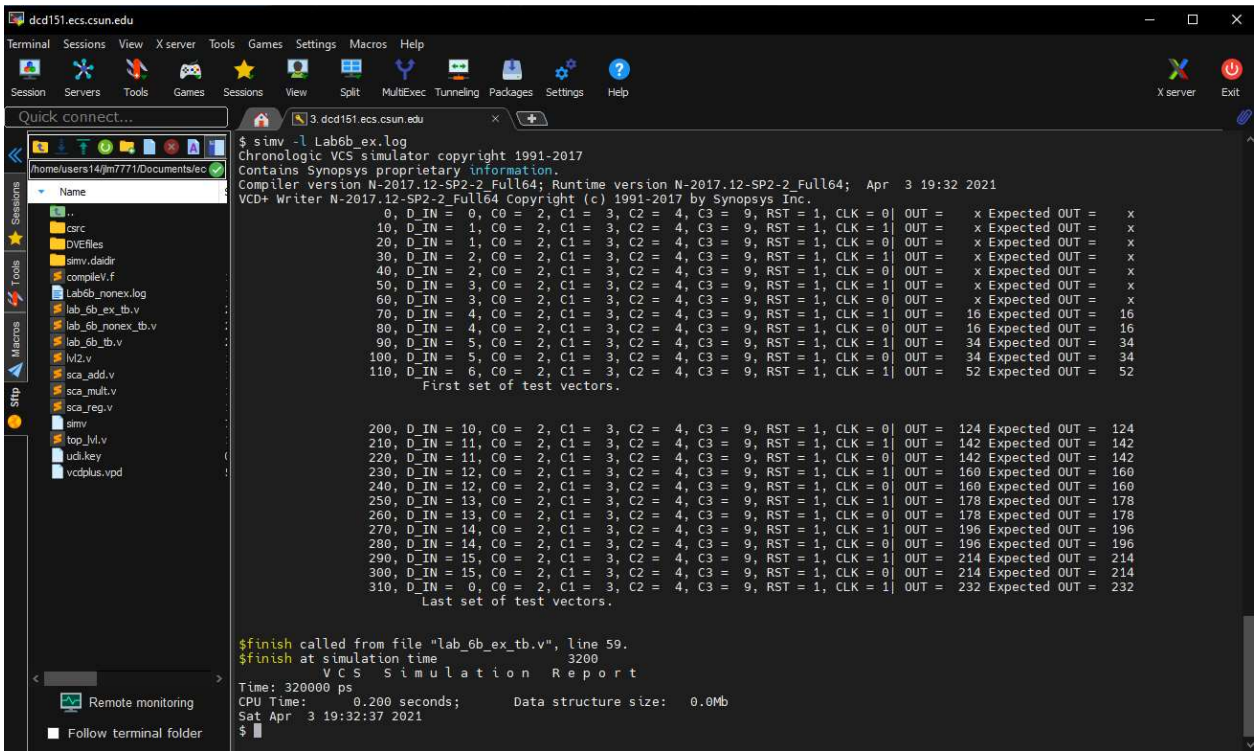

Fig. 9 exhaustive simv output

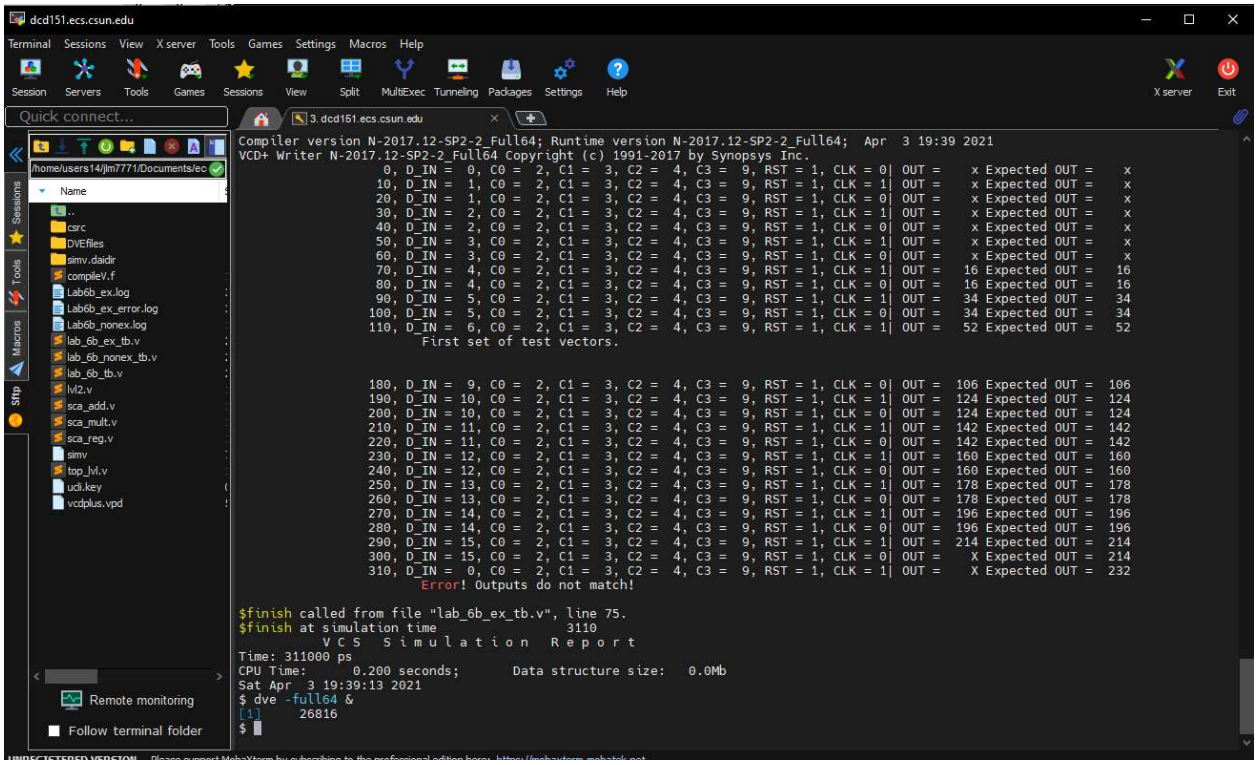

Fig. 10 exhaustive with error simv output

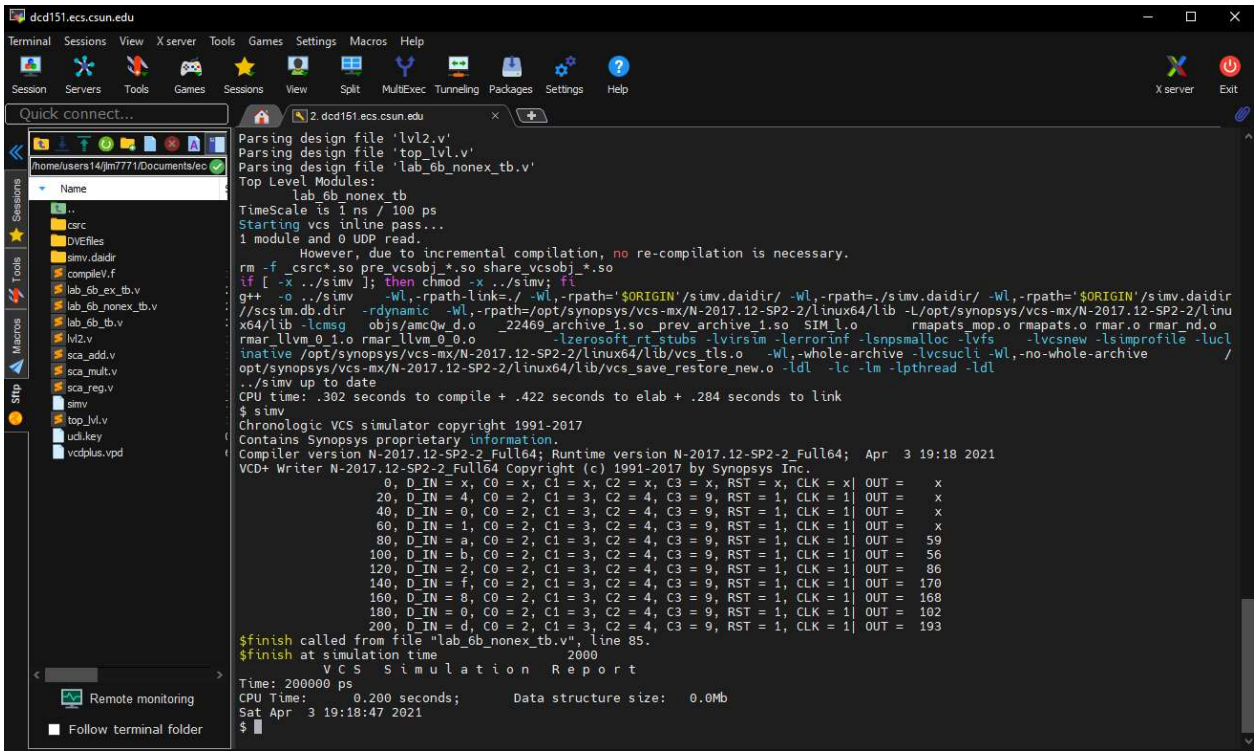

Fig. 11 non-exhaustive simv output

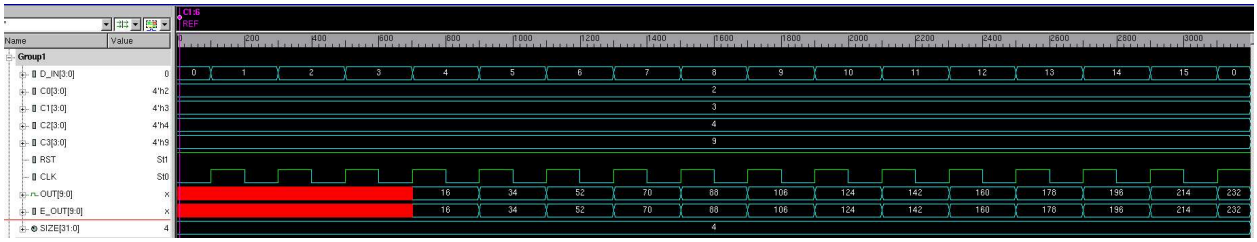

Fig. 12 exhaustive waveform

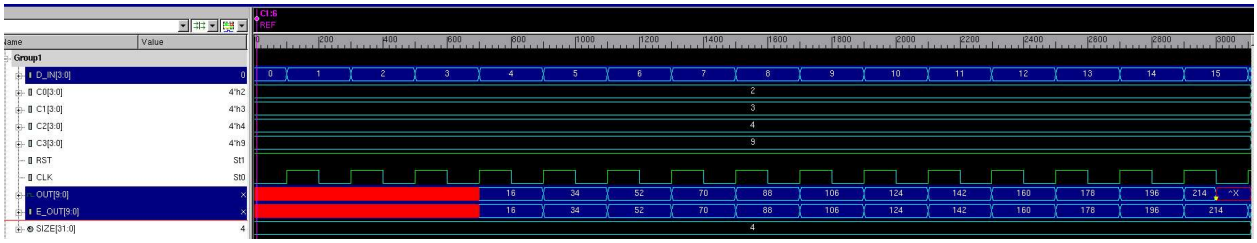

Fig. 13 exhaustive with error waveform

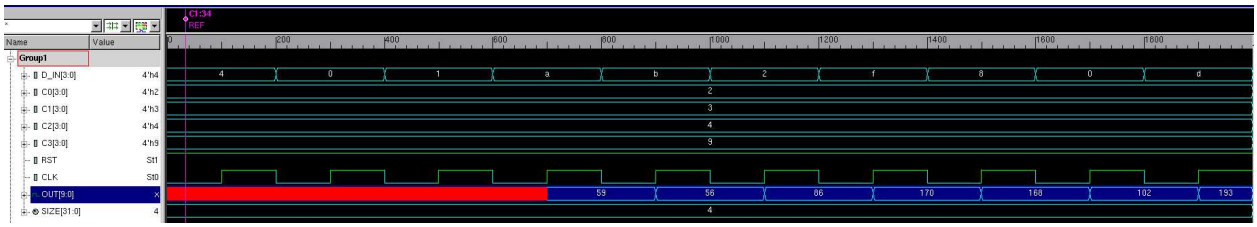

Fig. 14 non-exhaustive waveform

Analysis

For our non-exhaustive test we can see our results in **Fig. 11** and **Fig. 14**. Our OUTPUT is unknown for the first four clock cycles because our registers have not been initialized yet. So at the fourth cycle all registers will have a value and our OUT will not be X. This also applies to the other testbench file.

To verify that its working properly:

1. OUT = 4*C3+0*C2+1*C1+10*C0=4*9+0*4+1*3+10*2=59
2. OUT = B*C3+2*C2+F*C1+8*C0=B*9+2*4+F*3+8*2=168

These both match the waveform and simv outputs so we are able to verify that our design is working as designed.

For our exhaustive test, I tested all possible values of D_IN and we can see the results of this testbench in **Fig. 9, 10, 12, and 13**. In this testbench we also included a `ifdef compiler directive that will only compile that portion of the code if we specify the flag during compilation. For this testbench the `ifdef compiler directive forces our OUT to 4'b1x0x after 15 clock periods. In **Fig. 10 & 13** we compiled with defining ERROR so we were able to force our error. And as we can see in our simv output when our OUT is different from the EX_OUT our program prompts that there was an error and will terminate.

Conclusion

In conclusion, I created a hierarchical design for a Sum of Products digital filter. The design consisted of 3 modules on the lowest level being scalable register, multiplier, and adder. Two multipliers, two registers, and one adder from the lowest level were used to create the second level module. The top level module used two second level modules and one adder to complete the design. Two testbench files were created to test our design and as we can see from my results we are able to verify that the design is working as expected. For this lab I was able to complete all the requirements and have learned about the `ifdef compiler directive and I see myself using it more in the future.

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 L Martinez

Name (signed) Jose Martinez Date 4/03/2021