California State University, Northridge

Department of Electrical & Computer Engineering



ECE 526L

Lab 8 Report

By

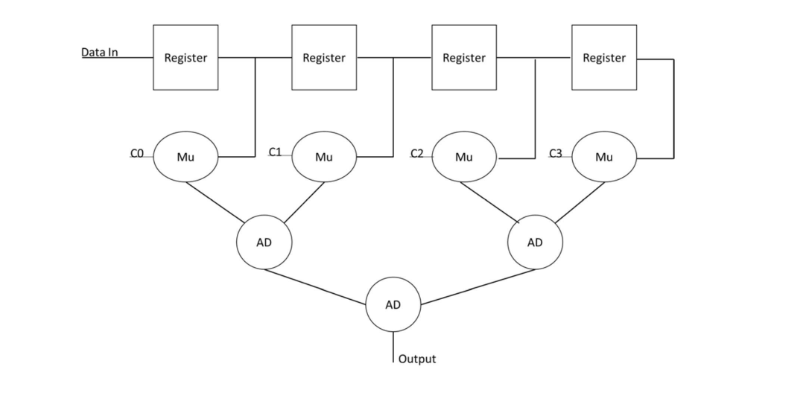
Avinash Damse

CSUN ID- 203131064

**1: Introduction**

The objective of this lab is to construct a Sum of Product circuit using three level hierarchy method (using scalable Register, Adder and Multiplier).

Here we have to use following diagram to build SOP.



**2: Procedure**

**a. Part 1: Creating REG Module**

In this lab I have created a REG module for scalable register. Inside the module I have assigned “data, clk and rst,” as input variables and “q and qbar” as output variable. Then I wrote the scalable register logic inside module. After completing the code I ended the module using and saved the file with name “REG.v”.

**b. Part 2: Creating ADDR Module**

In this lab I have created a scalable ADDR module. Inside the module I have assigned “ A1 and A2,” as input variables and “AD\_out” as output variable. Then I wrote the addition logic inside module. After completing the code I ended the module using and saved the file with name “ADDR.v”.

**c. Part 3: Creating MULT Module**

In this lab I have created a scalable MULT module. Inside the module I have assigned “ M and C,” as input variables and “out” as output variable. Then I wrote the multiplication logic inside module. After completing the code I ended the module using and saved the file with name “MULT.v”.

I have written the test bench for the REG module. We require test bench just to make sure that the module we have created is working properly. Here, in this testbench I have performed write to and read from operation and demonstrated an individual and block read.

**d. Part 4: Creating non\_hi\_sop Module**

In this lab I have created a non\_hi\_sop module. Inside the module I have assigned “ C0,C1,C2,C3, Data\_in and clk,” as input variables and “Output” as output variable. Then I instantiated three module of each REG, ADDR and MULT to perform nonhierarchical sum of product . After completing the code I ended the module using and saved the file with name “non\_hi\_sop.v”.

**e. Part 4: Creating non\_exhstv Testbench**

I have written the non-exhaustive test bench for the **non\_hi\_sop** module. We require test bench just to make sure that the module we have created is working properly. Here, in this testbench I have used small number of robust vectors to test strategy.

**f. Part 4: Creating exhstv Testbench**

I have written the exhaustive test bench for the **non\_hi\_sop** module. We require test bench just to make sure that the module we have created is working properly.

**c. Part 3: execution.**

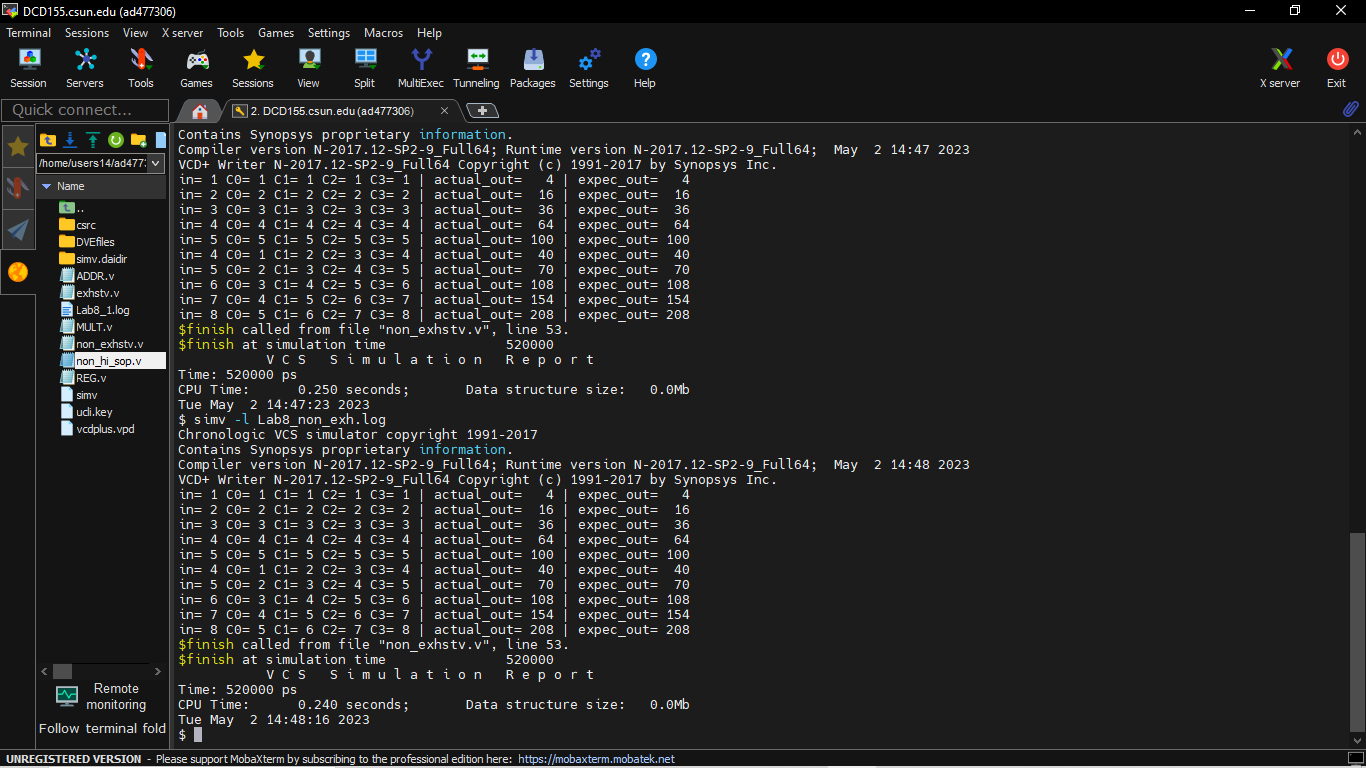
Using “vcs -debug -full64 REG.v ADDR.v MULT.v non\_hi\_sop.v non\_exhsv.b” command I executed first testbench file.

Again using “vcs -debug -full64 REG.v ADDR.v MULT.v non\_hi\_sop.v exhsv.b” command I executed second testbench file.

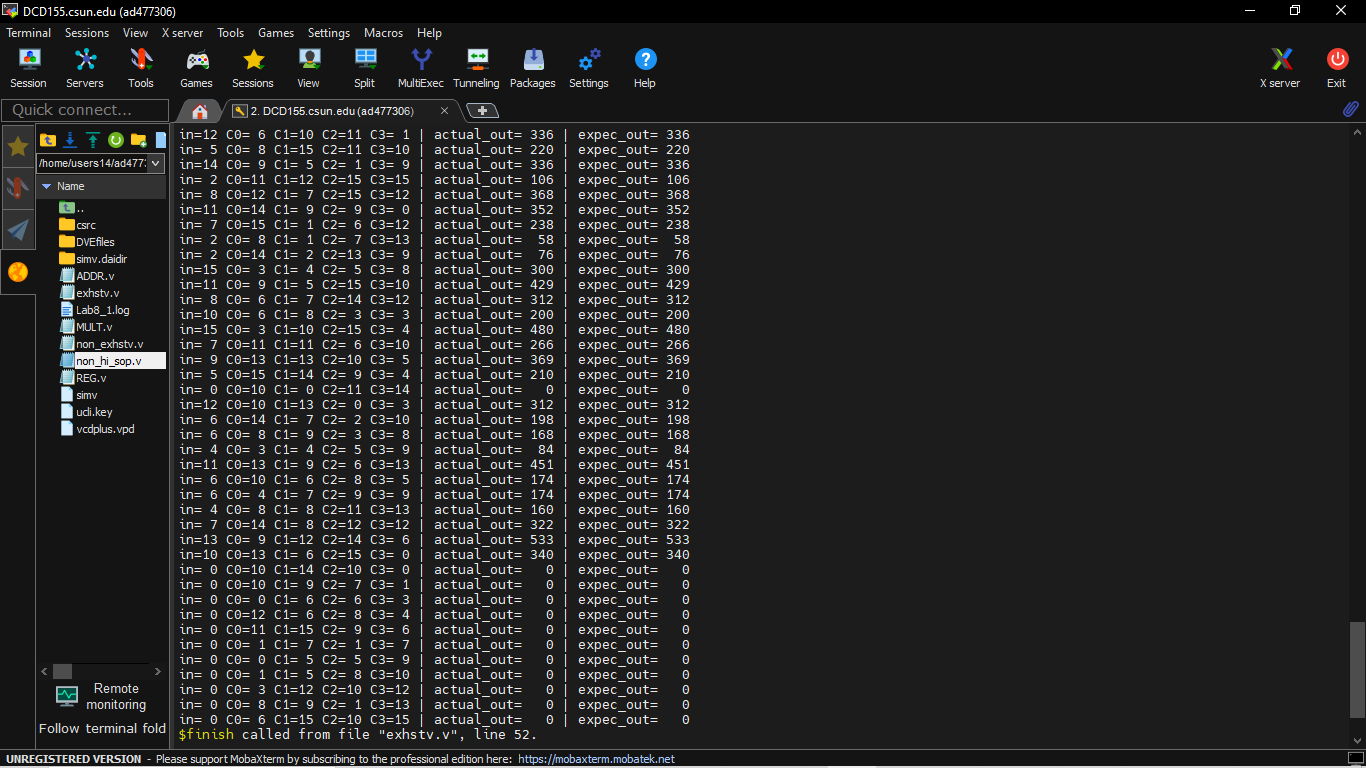
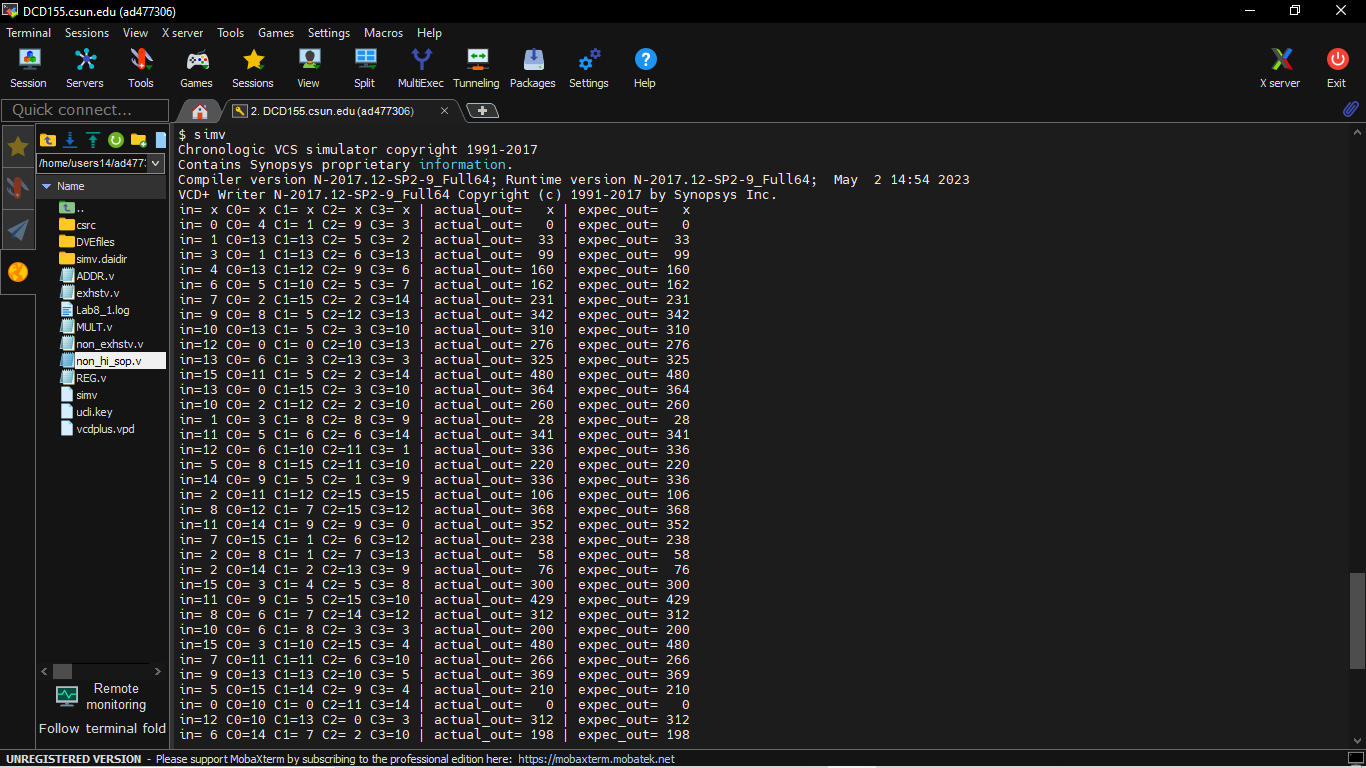
**d. Part 4: Simulation**

After an execution of all modules, I have run the command “simv” for simulation.

**Simulation for Non-Exhaustive Test Strategy :**



**Simulation for Exhaustive Test Strategy :**



**e. Part 5: Creating Log File**

After running the simulation I created the log file for first stategy using the “simv -l Lab8\_non\_exh.log” command

Command: /home/users14/ad477306/Verilog/Lab8/./simv -l Lab8\_non\_exh.log

Chronologic VCS simulator copyright 1991-2017

Contains Synopsys proprietary information.

Compiler version N-2017.12-SP2-9\_Full64; Runtime version N-2017.12-SP2-9\_Full64; May 2 14:48 2023

VCD+ Writer N-2017.12-SP2-9\_Full64 Copyright (c) 1991-2017 by Synopsys Inc.

in= 1 C0= 1 C1= 1 C2= 1 C3= 1 | actual\_out= 4 | expec\_out= 4

in= 2 C0= 2 C1= 2 C2= 2 C3= 2 | actual\_out= 16 | expec\_out= 16

in= 3 C0= 3 C1= 3 C2= 3 C3= 3 | actual\_out= 36 | expec\_out= 36

in= 4 C0= 4 C1= 4 C2= 4 C3= 4 | actual\_out= 64 | expec\_out= 64

in= 5 C0= 5 C1= 5 C2= 5 C3= 5 | actual\_out= 100 | expec\_out= 100

in= 4 C0= 1 C1= 2 C2= 3 C3= 4 | actual\_out= 40 | expec\_out= 40

in= 5 C0= 2 C1= 3 C2= 4 C3= 5 | actual\_out= 70 | expec\_out= 70

in= 6 C0= 3 C1= 4 C2= 5 C3= 6 | actual\_out= 108 | expec\_out= 108

in= 7 C0= 4 C1= 5 C2= 6 C3= 7 | actual\_out= 154 | expec\_out= 154

in= 8 C0= 5 C1= 6 C2= 7 C3= 8 | actual\_out= 208 | expec\_out= 208

$finish called from file "non\_exhstv.v", line 53.

$finish at simulation time 520000

V C S S i m u l a t i o n R e p o r t

Time: 520000 ps

CPU Time: 0.240 seconds; Data structure size: 0.0Mb

Tue May 2 14:48:16 2023

**Log for exhaustive strategy:**

Command: /home/users14/ad477306/Verilog/Lab8/./simv -l Lab8\_exh.log

Chronologic VCS simulator copyright 1991-2017

Contains Synopsys proprietary information.

Compiler version N-2017.12-SP2-9\_Full64; Runtime version N-2017.12-SP2-9\_Full64; May 2 15:04 2023

VCD+ Writer N-2017.12-SP2-9\_Full64 Copyright (c) 1991-2017 by Synopsys Inc.

in= x C0= x C1= x C2= x C3= x | actual\_out= x | expec\_out= x

in= 0 C0= 4 C1= 1 C2= 9 C3= 3 | actual\_out= 0 | expec\_out= 0

in= 1 C0=13 C1=13 C2= 5 C3= 2 | actual\_out= 33 | expec\_out= 33

in= 3 C0= 1 C1=13 C2= 6 C3=13 | actual\_out= 99 | expec\_out= 99

in= 4 C0=13 C1=12 C2= 9 C3= 6 | actual\_out= 160 | expec\_out= 160

in= 6 C0= 5 C1=10 C2= 5 C3= 7 | actual\_out= 162 | expec\_out= 162

in= 7 C0= 2 C1=15 C2= 2 C3=14 | actual\_out= 231 | expec\_out= 231

in= 9 C0= 8 C1= 5 C2=12 C3=13 | actual\_out= 342 | expec\_out= 342

in=10 C0=13 C1= 5 C2= 3 C3=10 | actual\_out= 310 | expec\_out= 310

in=12 C0= 0 C1= 0 C2=10 C3=13 | actual\_out= 276 | expec\_out= 276

in=13 C0= 6 C1= 3 C2=13 C3= 3 | actual\_out= 325 | expec\_out= 325

in=15 C0=11 C1= 5 C2= 2 C3=14 | actual\_out= 480 | expec\_out= 480

in=13 C0= 0 C1=15 C2= 3 C3=10 | actual\_out= 364 | expec\_out= 364

in=10 C0= 2 C1=12 C2= 2 C3=10 | actual\_out= 260 | expec\_out= 260

in= 1 C0= 3 C1= 8 C2= 8 C3= 9 | actual\_out= 28 | expec\_out= 28

in=11 C0= 5 C1= 6 C2= 6 C3=14 | actual\_out= 341 | expec\_out= 341

in=12 C0= 6 C1=10 C2=11 C3= 1 | actual\_out= 336 | expec\_out= 336

in= 5 C0= 8 C1=15 C2=11 C3=10 | actual\_out= 220 | expec\_out= 220

in=14 C0= 9 C1= 5 C2= 1 C3= 9 | actual\_out= 336 | expec\_out= 336

in= 2 C0=11 C1=12 C2=15 C3=15 | actual\_out= 106 | expec\_out= 106

in= 8 C0=12 C1= 7 C2=15 C3=12 | actual\_out= 368 | expec\_out= 368

in=11 C0=14 C1= 9 C2= 9 C3= 0 | actual\_out= 352 | expec\_out= 352

in= 7 C0=15 C1= 1 C2= 6 C3=12 | actual\_out= 238 | expec\_out= 238

in= 2 C0= 8 C1= 1 C2= 7 C3=13 | actual\_out= 58 | expec\_out= 58

in= 2 C0=14 C1= 2 C2=13 C3= 9 | actual\_out= 76 | expec\_out= 76

in=15 C0= 3 C1= 4 C2= 5 C3= 8 | actual\_out= 300 | expec\_out= 300

in=11 C0= 9 C1= 5 C2=15 C3=10 | actual\_out= 429 | expec\_out= 429

in= 8 C0= 6 C1= 7 C2=14 C3=12 | actual\_out= 312 | expec\_out= 312

in=10 C0= 6 C1= 8 C2= 3 C3= 3 | actual\_out= 200 | expec\_out= 200

in=15 C0= 3 C1=10 C2=15 C3= 4 | actual\_out= 480 | expec\_out= 480

in= 7 C0=11 C1=11 C2= 6 C3=10 | actual\_out= 266 | expec\_out= 266

in= 9 C0=13 C1=13 C2=10 C3= 5 | actual\_out= 369 | expec\_out= 369

in= 5 C0=15 C1=14 C2= 9 C3= 4 | actual\_out= 210 | expec\_out= 210

in= 0 C0=10 C1= 0 C2=11 C3=14 | actual\_out= 0 | expec\_out= 0

in=12 C0=10 C1=13 C2= 0 C3= 3 | actual\_out= 312 | expec\_out= 312

in= 6 C0=14 C1= 7 C2= 2 C3=10 | actual\_out= 198 | expec\_out= 198

in= 6 C0= 8 C1= 9 C2= 3 C3= 8 | actual\_out= 168 | expec\_out= 168

in= 4 C0= 3 C1= 4 C2= 5 C3= 9 | actual\_out= 84 | expec\_out= 84

in=11 C0=13 C1= 9 C2= 6 C3=13 | actual\_out= 451 | expec\_out= 451

in= 6 C0=10 C1= 6 C2= 8 C3= 5 | actual\_out= 174 | expec\_out= 174

in= 6 C0= 4 C1= 7 C2= 9 C3= 9 | actual\_out= 174 | expec\_out= 174

in= 4 C0= 8 C1= 8 C2=11 C3=13 | actual\_out= 160 | expec\_out= 160

in= 7 C0=14 C1= 8 C2=12 C3=12 | actual\_out= 322 | expec\_out= 322

in=13 C0= 9 C1=12 C2=14 C3= 6 | actual\_out= 533 | expec\_out= 533

in=10 C0=13 C1= 6 C2=15 C3= 0 | actual\_out= 340 | expec\_out= 340

in= 0 C0=10 C1=14 C2=10 C3= 0 | actual\_out= 0 | expec\_out= 0

in= 0 C0=10 C1= 9 C2= 7 C3= 1 | actual\_out= 0 | expec\_out= 0

in= 0 C0= 0 C1= 6 C2= 6 C3= 3 | actual\_out= 0 | expec\_out= 0

in= 0 C0=12 C1= 6 C2= 8 C3= 4 | actual\_out= 0 | expec\_out= 0

in= 0 C0=11 C1=15 C2= 9 C3= 6 | actual\_out= 0 | expec\_out= 0

in= 0 C0= 1 C1= 7 C2= 1 C3= 7 | actual\_out= 0 | expec\_out= 0

in= 0 C0= 0 C1= 5 C2= 5 C3= 9 | actual\_out= 0 | expec\_out= 0

in= 0 C0= 1 C1= 5 C2= 8 C3=10 | actual\_out= 0 | expec\_out= 0

in= 0 C0= 3 C1=12 C2=10 C3=12 | actual\_out= 0 | expec\_out= 0

in= 0 C0= 8 C1= 9 C2= 1 C3=13 | actual\_out= 0 | expec\_out= 0

in= 0 C0= 6 C1=15 C2=10 C3=15 | actual\_out= 0 | expec\_out= 0

$finish called from file "exhstv.v", line 52.

$finish at simulation time 2770000

V C S S i m u l a t i o n R e p o r t

Time: 2770000 ps

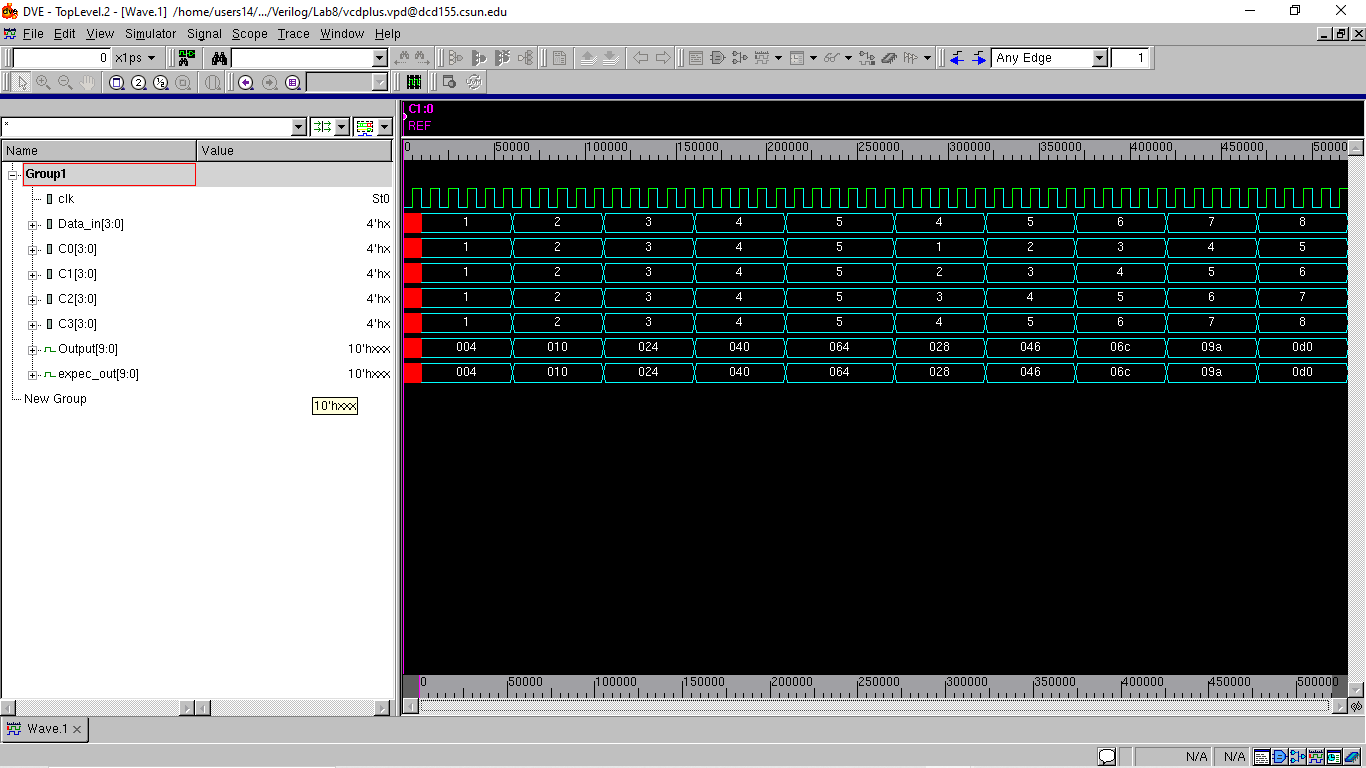
CPU Time: 0.240 seconds; Data structure size: 0.0Mb

Tue May 2 15:04:09 2023

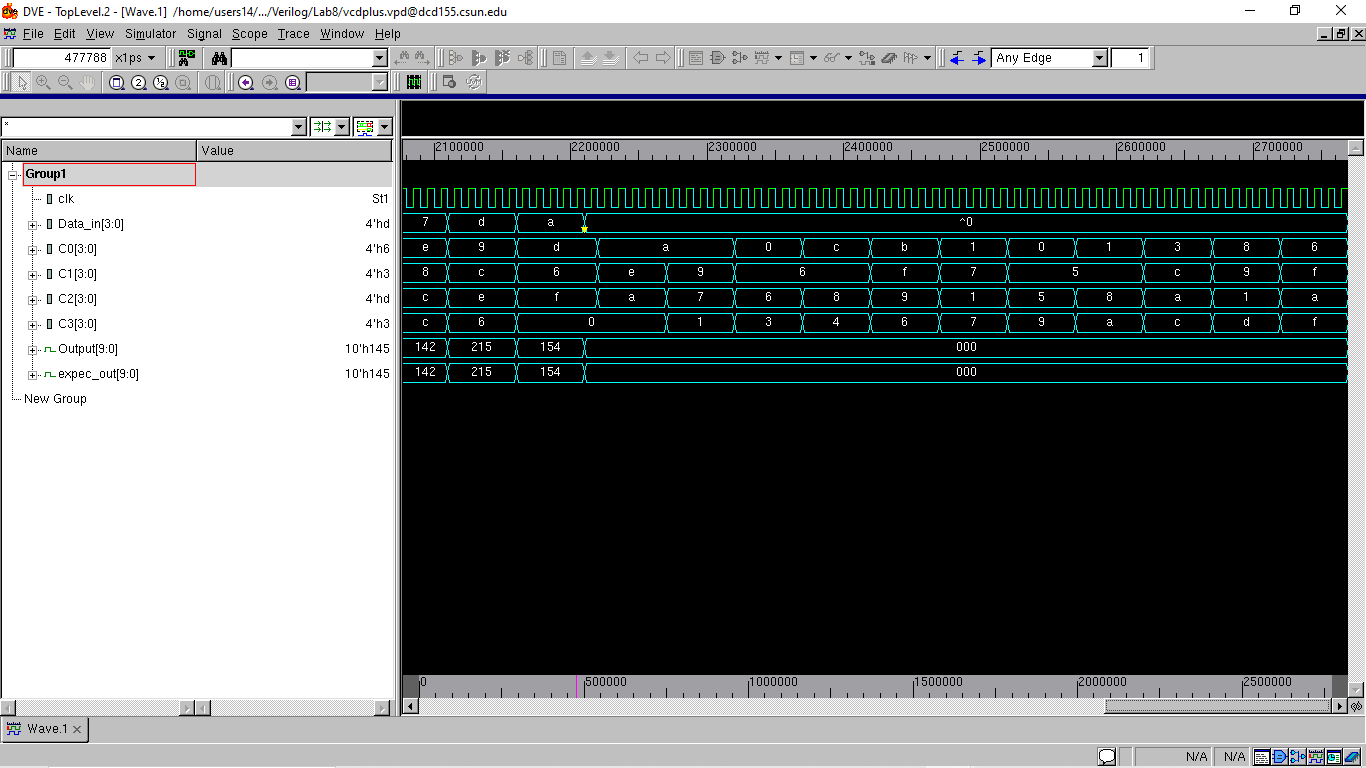
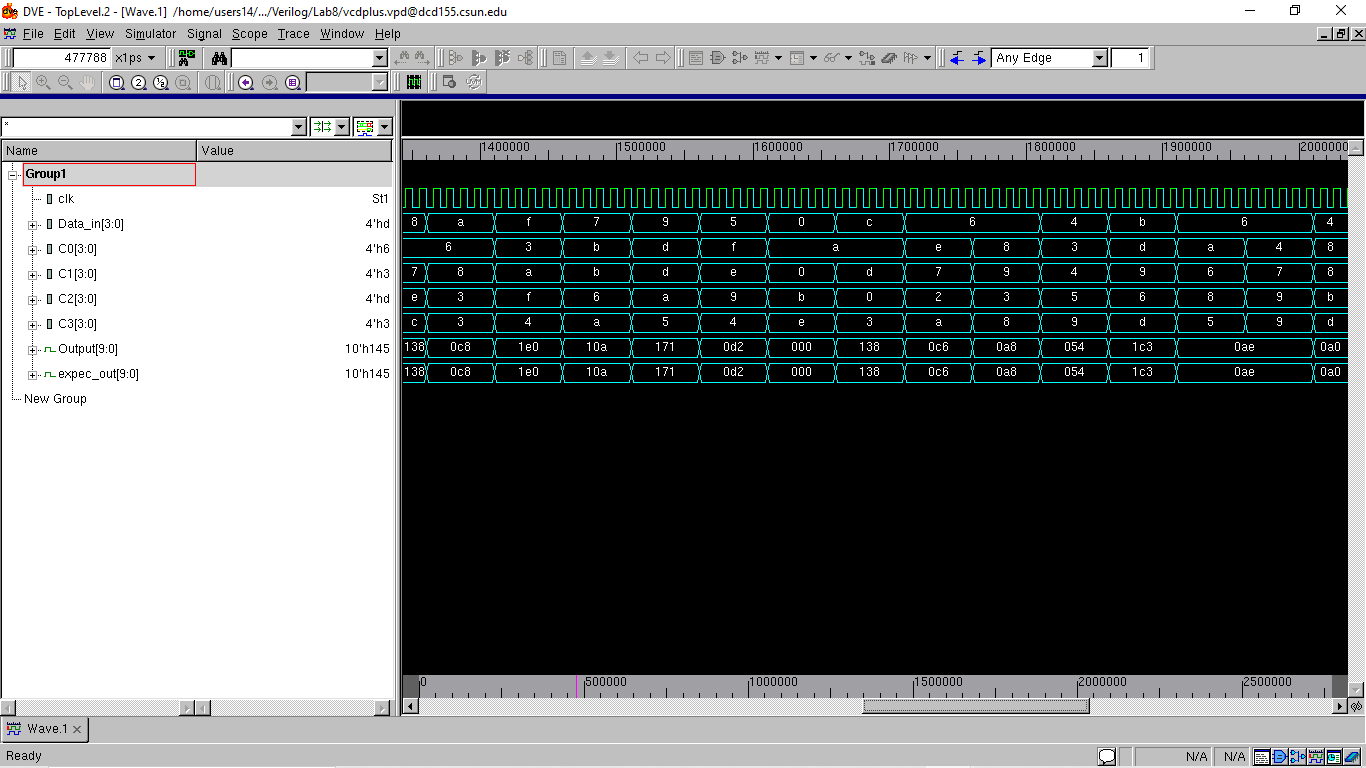
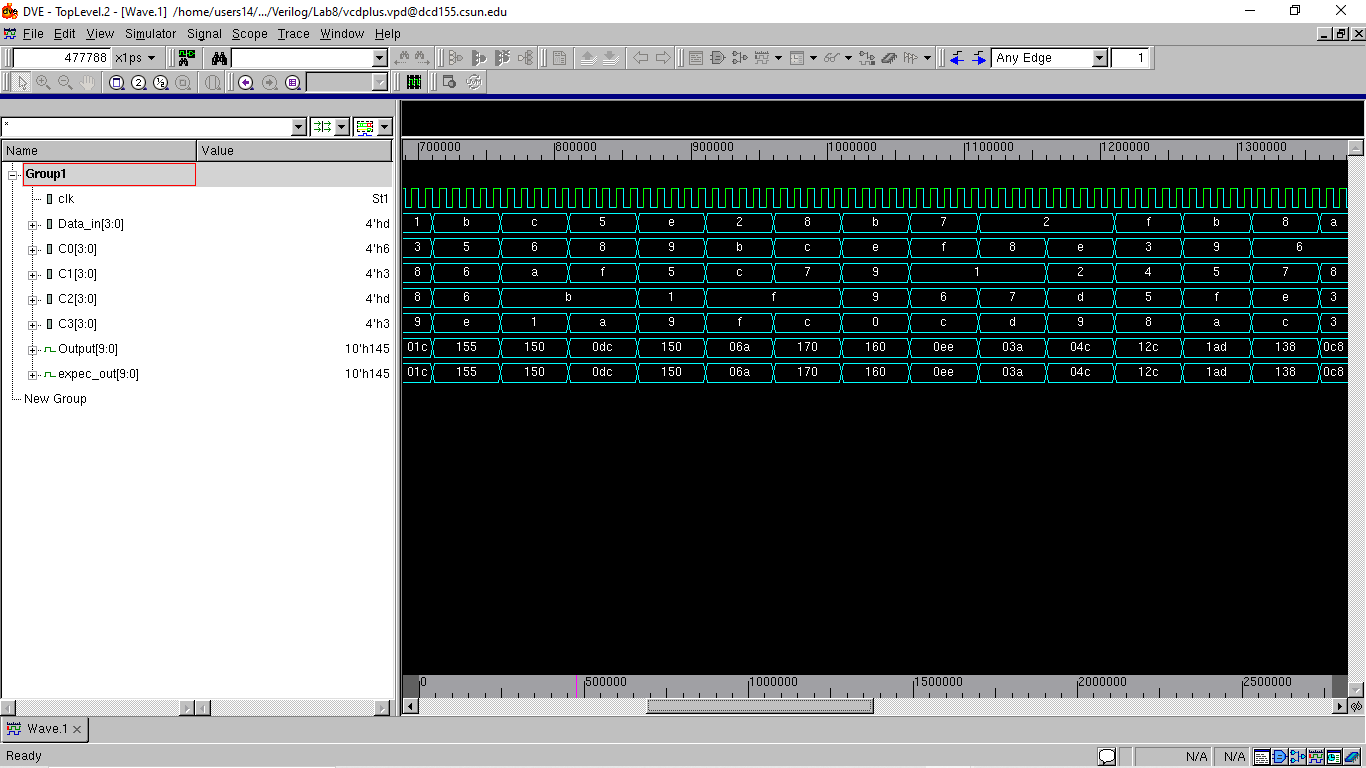
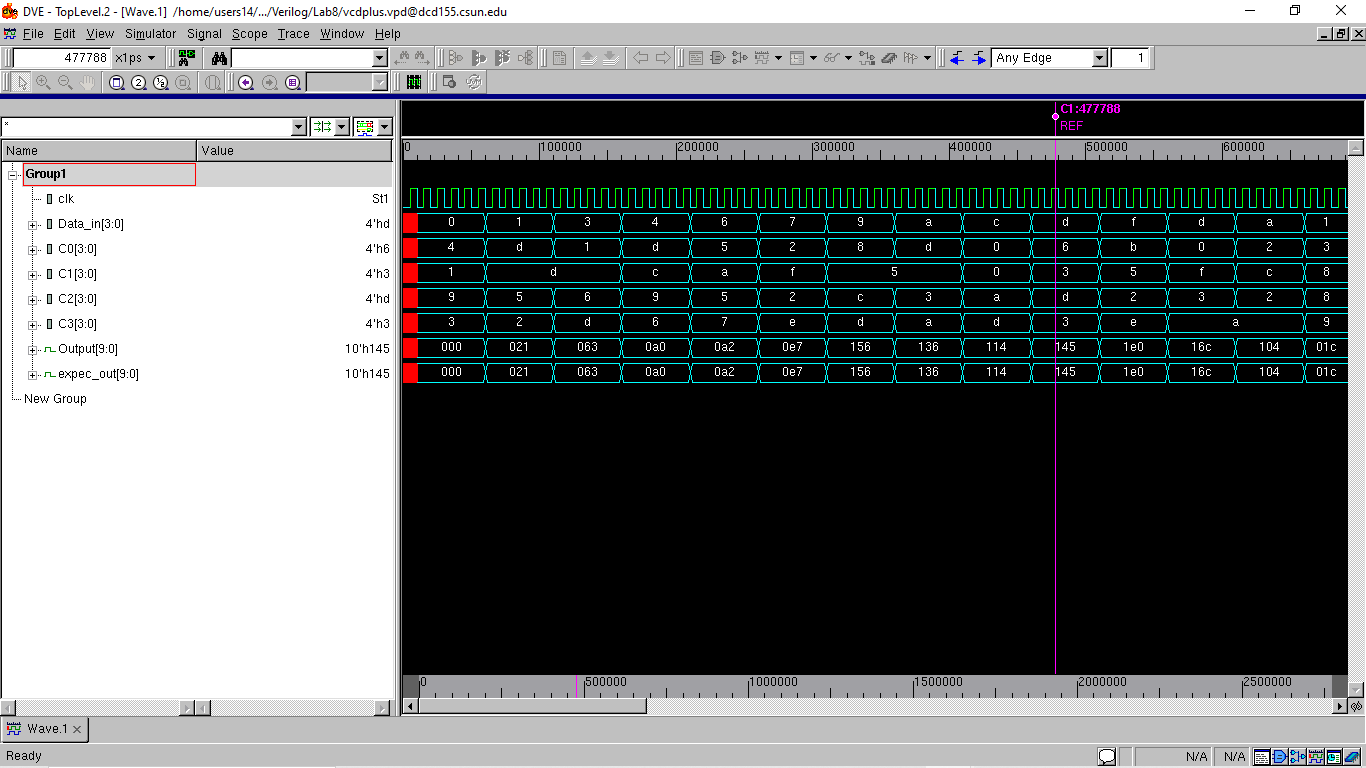
**f. Part 6: Seeing the waveform.**

After creating the log file I opened the DVE using “dve -full64 &” command to see the waveforms.

**Non\_Exhaustive:**



**Exhaustive:**



**Conclusion:**

The Sum of Product model was successfully implemented using three level hierarchical scalable Register, Adder and Multiplies. And successfully tested using Non-Exhaustive and Exhaustive test strategies.

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) Avinash Damse

Name(signed) Date : 2-May-2023

