# CECS 341 – LAB 1

# ALU Structural Model

# 03 February 2020

Rifa Safeer Shah

017138353

I certify that this submission is my original work

Lab Report: Lab Assignment 1 – ALU Structural Model

1. Goal: The goal of the ALU Structural Model Lab Assignment is to use Structural Verilog to model an Arithmetic Logic Unit.

2. Steps:
    a. Go to edaplayground.com and log in to your account.
    b. Rewrite the outline provided with this lab.
    c. Identify what is to be done to produce specified output.
    d. Use table 1.1 to fill in the appropriate missing code.
    e. Check for errors
    f. Run the code for test cases.

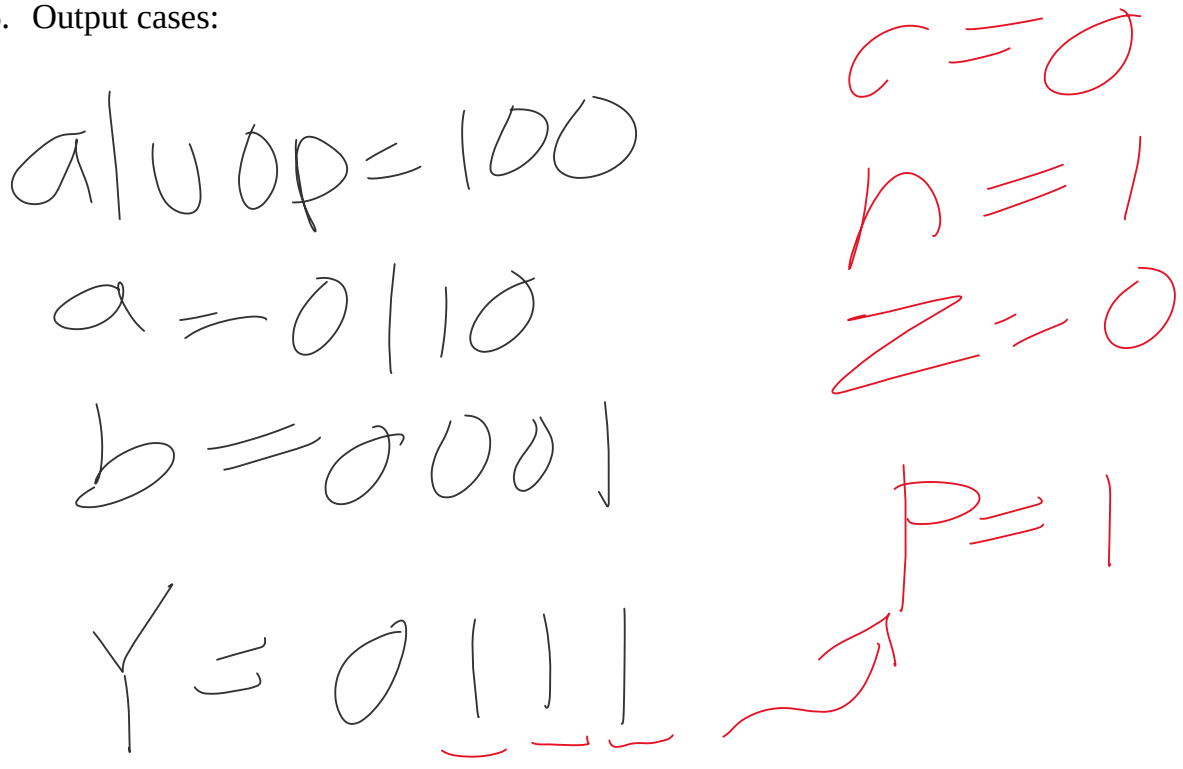
3. Results: In this lab assignment we tested for different ALU operations using Structural Verilog. The output displayed 72 test cases which was the same amount that was expected. Since the index starts from 0 the output includes test cases 0-71. The code iterates through the three loops in the testbench.sv The output shows the flags C, N, Z, P for each operation test case.

4. Conclusion: In conclusion of this lab, I learned the use of the two windows (testbench.sv and design.sv) on the eda playground. Some challenges I faced

during this lab included rewriting the skeleton code without any errors and learning to write the correct syntax. When I was trying to use the XOR function I was entering it in the wrong syntax on the eda playground which was giving me an error. Figuring out the logic behind the code and what the lab is doing was challenging but with the use of the notes it was clear to understand.

5. Notes: The C is the Carry Flag. If C equals to 1 then there is an unsigned overflow, else there is no unsigned overflow. N is the Sign Flag. It is the most significant bit (MSB) of ALU output. If N is equal to 1 then the Y value is negative, else the Y value is positive. Z is the Zero Flag. If the output Y is equal to 0 then Z will be 1, else Z will equal to 0. The P Flag is the Parity Flag. If the output Y has odd number of 1's then P is equal to 1, else P is equal to 0. The ALU operation code 000 refers to Add operation. 001 refers to Inc operation. 010 refers to And operation. 011 refers to Or operation. The 100 refers to Xor operation. 101 refers to Not operation. The 110 ALU operation code refers to Shl operation. 111 refers to Nop operation.

6. Output cases:



## 7. Lab Screenshots:

testbench.sv

SV/Verilog Testbench

```
`timescale 1ns/100ps
`define datasize  4
module alu_tester();
  reg [`datasize-1:0] a,b;
  reg [2:0] aluop;
  wire [`datasize-1:0] y;
  wire c,n,z,p;

  alu #(`datasize) dut(a,b,aluop,y,c,n,z,p);

  integer i,j,k,l, seed1, seed2;

  initial begin
    seed1 = 10*$random;
    seed2 = 10*$random;
    {a,b,aluop,i,j,k,l} = 0;
    for(i = 0; i < 8; i = i + 1)
      begin
        for(j = 0; j < 3; j = j + 1)
          begin
            for(k = 0; k < 3; k = k + 1)
              begin
                aluop = i;
                a = $random(seed1);
                b = $random(seed2);
                #1 $display("Test Case %d",l);
                #1 $display("aluop = %b\ta = %b\t b = %b\ty = %b",aluop,a,b,y);
                #1 $display("c = %b, n = %b, z = %b, p = %b",c,n,z,p);
                #1 $display("");
                l = l + 1;
                #1 $display("");
              end
          end
      end
  end
endmodule
```

design.sv

SV/Verilog Design

```
`timescale 1ns/100ps
module alu #(parameter width = 8)
  (
    input [width-1:0] a,b,
    input [2:0] aluop,
    output reg [width-1:0] y,
    output reg c,n,z,p
  );

  always @(a,b,aluop)
    begin
      {y,c,n,z,p} = 0;
      case(aluop)
        0: {c,y} = a + b;
        1: {c,y} = a + 1;
        2: y = a & b;
        3: y = a | b;
        4: y = a ^ b;
        5: y = ~ a;
        6: {c,y} = a << 1;
        7: y = 0;
        default: y = 32'bz;
      endcase
      n = y[width-1];
      z = !y;
      p = ^y;
    end
endmodule
```