LAB 4
ALU Structural Model

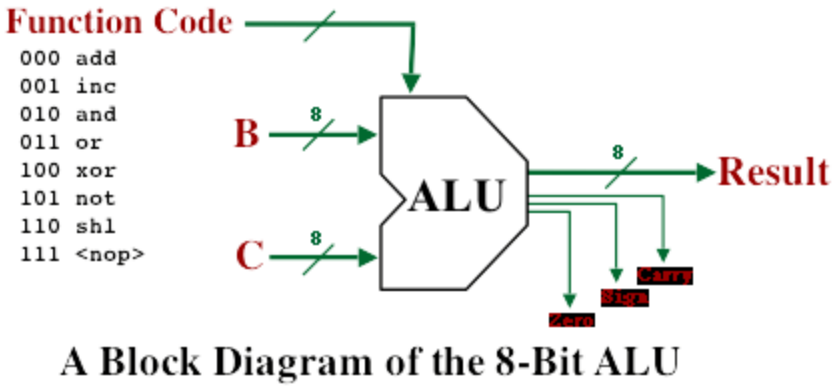

Figure 1.1 ALU Block diagram and Function Codes

OBJECTIVE: Model an Arithmetic Logic Unit using structural Verilog. The ALU can perform the following operations:

Table 1.1:

ALU Operation Code	Operation to Perform	Explanation	Flags affected
000	Add	Y = A+B	C,N,Z,P
001	Inc	Y = A+1	C,N,Z,P
010	And	Y = A&B	N,Z,P
011	Or	Y = A|B	N,Z,P
100	Xor	Y = AxorB	N,Z,P
101	Not	Y = ~A	N,Z,P
110	Shl	Y = A<<1	C,N,Z,P
111	Nop	Y = 0	N,Z,P

C is the Carry Flag: If(C==1) unsigned overflow; else no unsigned overflow

N is the Sign Flag (the MSB of ALU output): if(N==1) the Y value is negative; else Y is positive;

Z is the Zero Flag: if(Y==0) Z = 1; else Z = 0;

P is the Parity Flag: if(Y has an odd # of 1's) P = 1; else P = 0;

PROCEDURE: In this lab, an ALU with configurable data size will be created in Verilog. This will allow for the ALU module to be created once but the size of the data to be processed is determined when the ALU module is instantiated. Also this lab will serve as an introduction to the edaplayground.com development environment:

1. Go to https://www.edaplayground.com/

2. There are three methods to log in (Figure 1.2). Automatic login via Google or Facebook account is easily accomplished with a single button press (Figure 1.3). To use some other email account to log in, click the No Google or Facebook account? Link to go through the account sign up process.

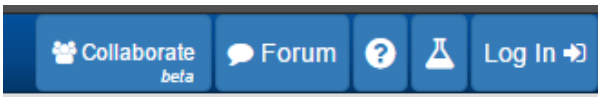

Figure 1.2 Log In menu

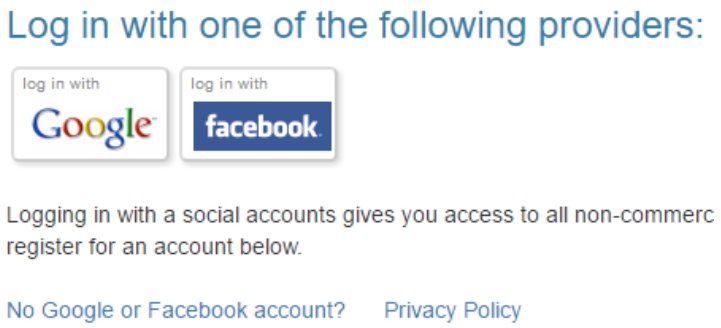

Figure 1.3 Logging in using Google or Facebook account

3. After logging in a blank project will be available.

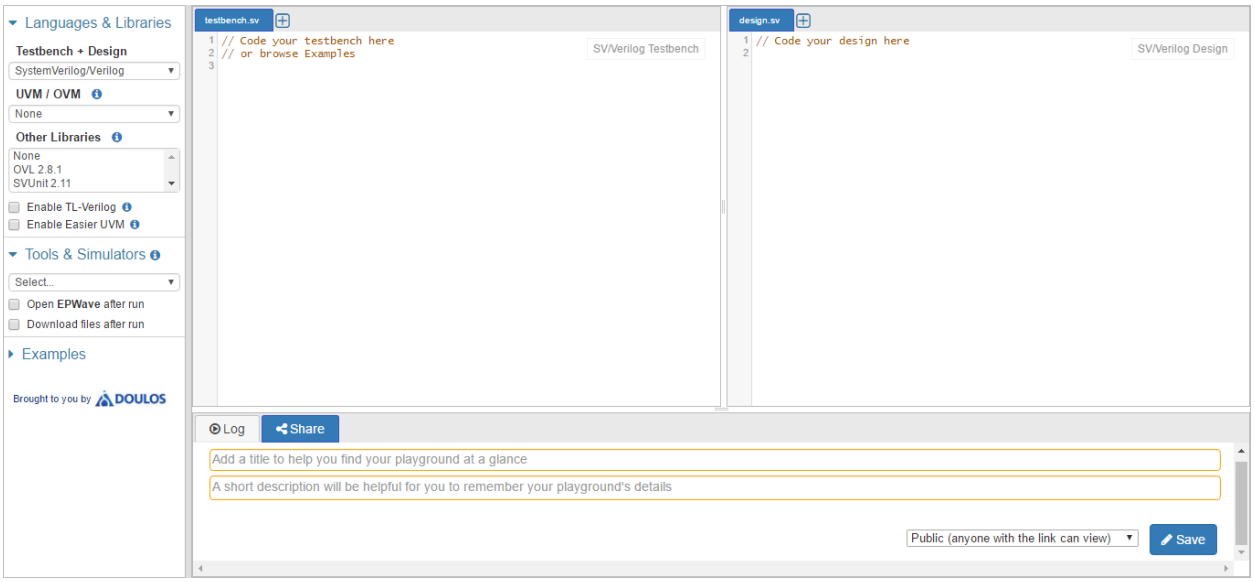

Figure 1.4 EDAPlayground: balk project - user interface

4. Your main task in this assignment is to complete the behavioral definition of the Arithmetic Logic Unit. The code is to be entered into the **design.sv** window. Use the Table 1 to fill in the missing portions of assignment statements (find Verilog operators that are necessary). The **parameter width** value can be overridden when the module is instantiated to change the data size. Skeleton code for the ALU is given in Figure 1.5:

design.sv

```
1  `timescale 1ns/100ps
2  module alu #(parameter width = 8)
3    (
4      input [width-1:0] a,b,
5      input [2:0] aluop,
6      output reg [width-1:0] y,
7      output reg c,n,z,p
8    );
9
10   always @(a,b,aluop)
11     begin
12       {y,c,n,z,p} = 0;
13       case(aluop)
14         0: {c,y} = a + b;
15         1: {c,y} =
16         2: y =
17         3: y =
18         4: y =
19         5: y =
20         6: {c,y} =
21         7: y = 0;
22         default: y = 32'bZ;
23       endcase
24       n = y[width-1];
25       z = !y;
26       p = ^y;
27     end
28 endmodule
```

Figure 1.5 Verilog code for ALU module with portions that need to be populated

5. In the testbench.sv window enter the code given in Figure 1.6.

testbench.sv

```
1  `timescale 1ns/100ps
2  `define datasize  4
3  module alu_tester();
4    reg [`datasize-1:0] a,b;
5    reg [2:0] aluop;
6    wire [`datasize-1:0] y;
7    wire c,n,z,p;
8
9    alu #(`datasize) dut(a,b,aluop,y,c,n,z,p);
10
11   integer i,j,k,l, seed1, seed2;
12
13   initial begin
14     seed1 = 10*$random;
15     seed2 = 32*$random;
16     {a,b,aluop,i,j,k,l} = 0;
17     for(i = 0; i < 8; i = i + 1)
18       begin
19         for(j = 0; j < 3; j = j + 1)
20           begin
21             for(k = 0; k < 3; k = k + 1)
22               begin
23                 aluop = i;
24                 a = $random(seed1);
25                 b = $random(seed2);
26                 #1 $display("Test Case %d",l);
27                 #1 $display("aluop = %b\ta = %b\t b = %b\ty = %b",aluop,a,b,y);
28                 #1 $display("c = %b, n = %b, z = %b, p = %b",c,n,z,p);
29                 #1 $display("");
30                 l = l + 1;
31                 #1 $display("");
32               end
33           end
34       end
35   end
36 endmodule
```

Figure 1.6 Verilog code for testbench

6. Once all the given code has been entered and completed, select Icarus Verilog as the simulator tool (Figure 1.7).

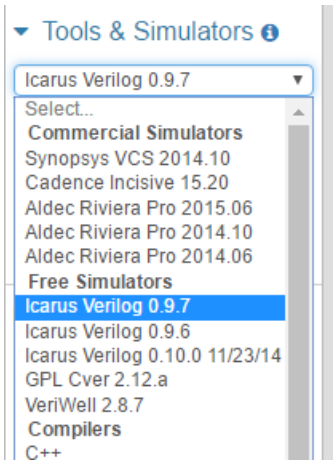

Figure 1.7 Drop down menu used to choose the simulation tool

7. Click the Run button (Figure 1.8) and the result of each test case will be written to the console (Figure 1.9) if there are no project errors. Each window pane can be resized as needed to see all relevant information!

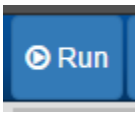

Figure 1.8 On the top of the window you will find “Run” button

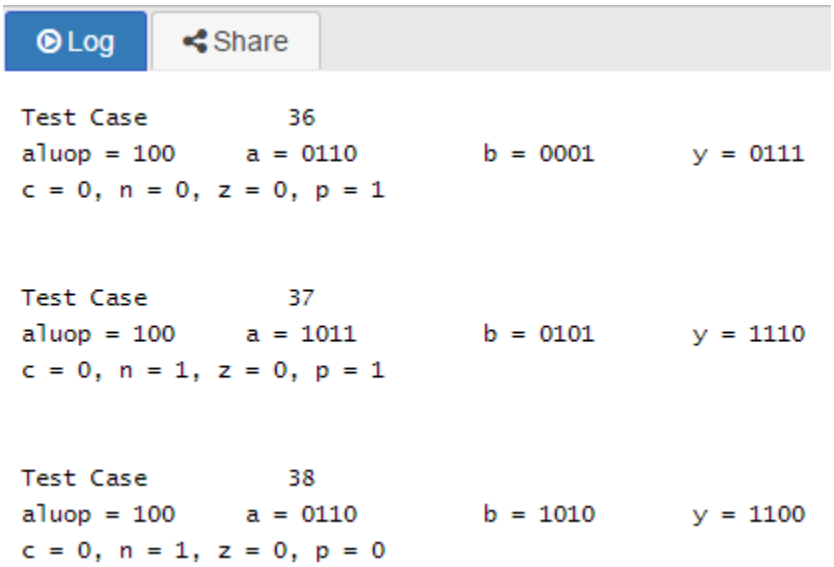

Figure 1.9 Console output showing some of the test cases.

8. Check the test results to verify the project is working correctly.

WHAT TO SUBMIT

Once you have verified proper functionality of your project, copy the contents of the **design.sv** to a text file named **alu.txt** and upload the BeachBoard Dropbox for Lab 4. Additionally, upload your Lab report (see the LabReportTemplete in the Documents folder on BeachBoard).

NOTES:

- Comment your code
- Keep these lab files as they will be needed for future labs!

Created by Josh Hayter, update by Gayathri Venna and Jelena Trajkovic