LAB 8:

Arithmetic-Logic Unit Modeling

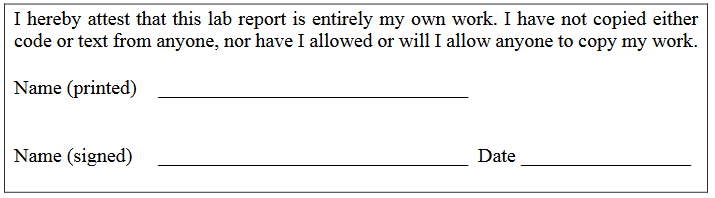
Professor Ronald Mehler

ECE 526L

Spring 2018

Garen Nikoyan

4/5/2018



**Objective:**

The objective of this lab is to create a module which allows different operations with the use of case statements. This can also be created using if statements, but doing so will make the module more congested and less efficient.

**Methodology:**

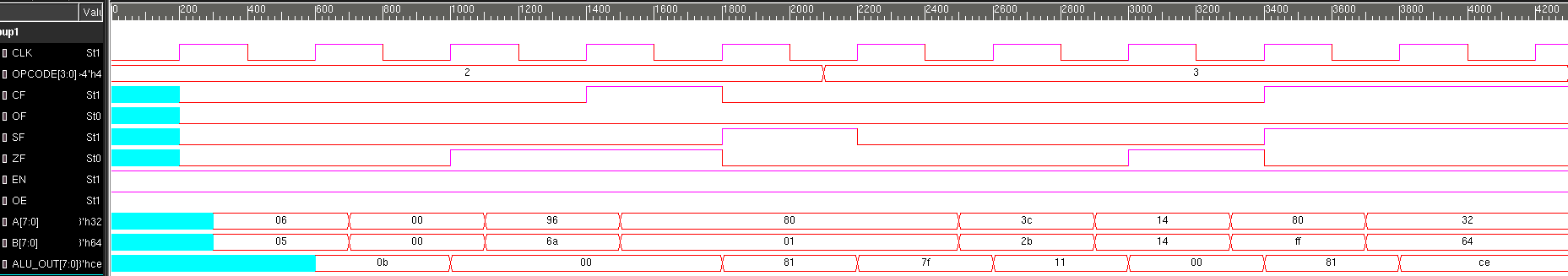
This lab was done by first creating the module, using case statements which relied on the OPCODE input. Based on that input, the module would then go to that case, and then perform the desired operation. A temporary variable was used in the module to ensure the output was passed correctly to the ? conditional operator, which was in a separate always block. This operator was in a separate block so that they would run in parallel and would run faster this way, and also because having them in the same block led to some errors in the output.

The test bench was able to check both signed and unsigned inputs, by commenting out a single line in the beginning, `ifdef SignedINTs. The test bench tested to see if each flag would work properly in each operation. The overflow and carry flags were not changed during Boolean operations, which is why they those flags carry over from the previous test vector.

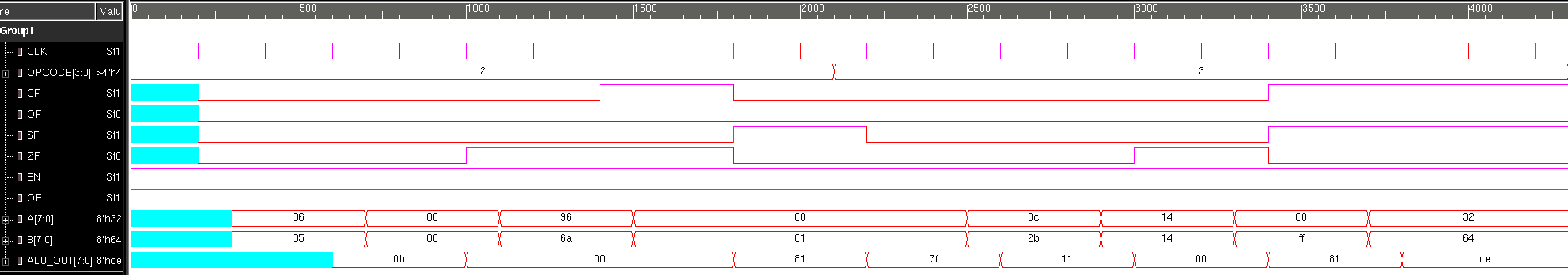
**Analysis:**

Looking at waveforms and the log, it can be seen that the modules worked correctly, as the outputs were as expected.

In Figure 1, the unsigned addition and subtraction operations’ waveforms can be seen, and signed in figure 2. The zero, carry, and signed flags work properly. Overflow is not really an issue for unsigned inputs. But it is important to note that for the signed input case, an error was made in the test bench and the numbers used did not create an overflow.

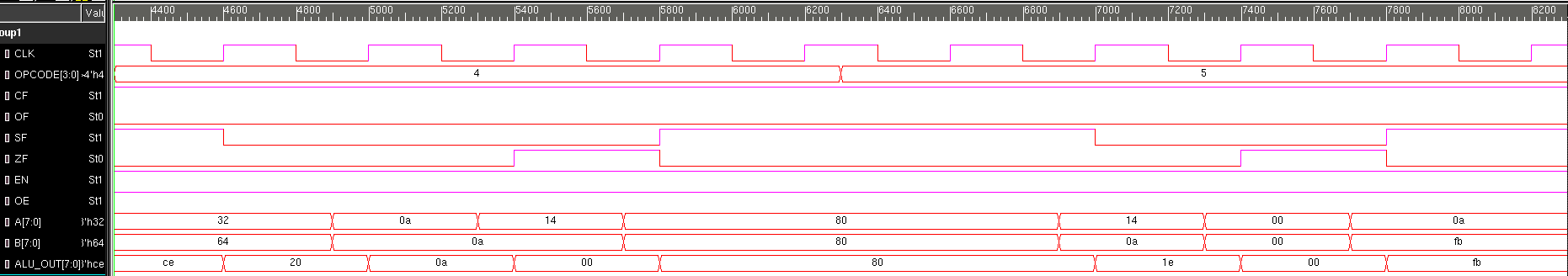


*Figure 1: Unsigned addition and subtraction*

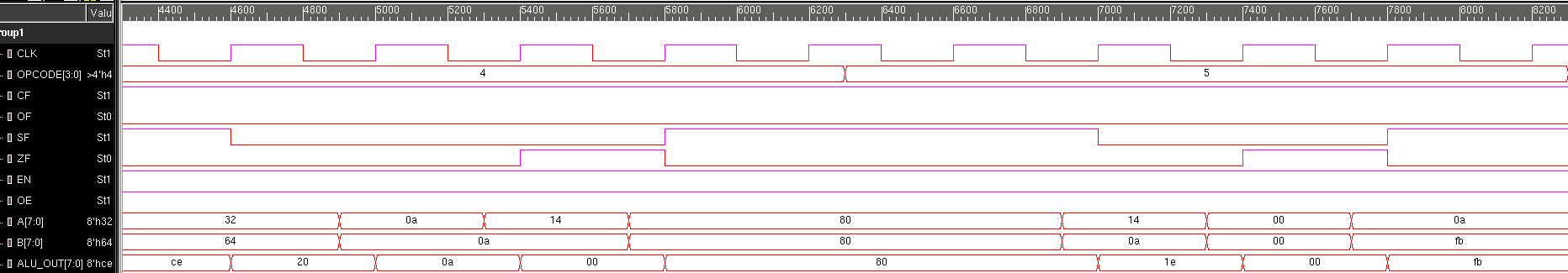
**

*Figure 2: Signed addition and subtraction*

In Figure 3 the unsigned AND and OR operations’ waveforms can be seen, and signed in Figure 4. For these operations, only the signed and zero flag will be set or not set, since Boolean operations cannot have carry or overflow. Those flags were left as is from previous test vectors. As can be seen, the zero and signed flags work as expected. In the unsigned case, the sign flag only indicates if the MSB is 1, but in the signed case, it indicates if the output is negative.

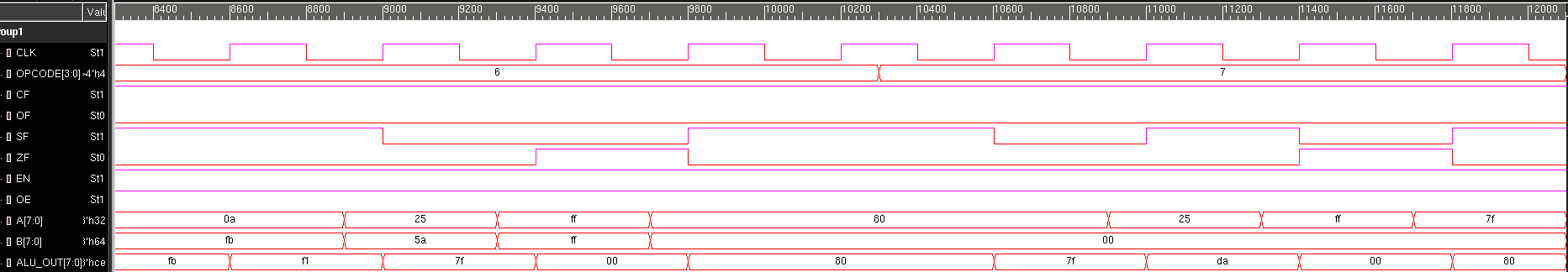


*Figure 3: Unsigned AND and OR*

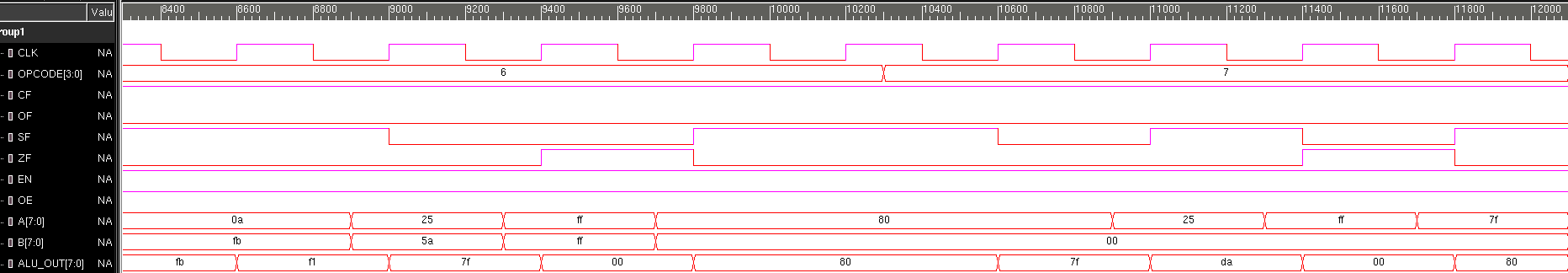
**

*Figure 4: Signed AND and OR*

In Figure 4, the unsigned XOR and NOT operations’ waveforms can be seen, and signed in Figure 5. For these operations, only the signed and zero flag will be set or not set, since Boolean operations cannot have carry or overflow. Those flags were left as is from previous test vectors. As can be seen, the zero and signed flags work as expected. In the unsigned case, the sign flag only indicates if the MSB is 1, but in the signed case, it indicates if the output is negative.



*Figure 5: Unsigned XOR and NOT*

**

*Figure 6: Signed XOR and NOT*

**Modules:**

**ALU.sv**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* ECE526L Experiment #8 Garen Nikoyan, Spring 2018

\*\*\* Arithmetic-Logic Unit Modeling

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Filename: ALU.sv Created by: Garen Nikoyan, 4/5/2018 \*\*\*

\*\*\* -Revision History

\*\*\* 4/5/2018: First draft

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* This module models an ALU

\*\*\* It can add, subtract, AND, OR, XOR and NOT. It has a carry flag,

\*\*\* overflow flag, signed flag, and zero flag.

\*\*\* EN needs to be high to enable the ALU, and OE needs to be high

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

`timescale 1ns/100ps

module ALU(CLK, EN, OE, OPCODE, A, B, ALU\_OUT, CF, OF, SF, ZF);

parameter WIDTH = 8;

output reg [ WIDTH-1 : 0] ALU\_OUT;

output reg CF, OF, SF, ZF;

input [3:0] OPCODE;

input [ WIDTH-1 : 0] A, B;

input reg CLK, EN, OE;

reg [ WIDTH-1 : 0] temp;

localparam ADDalu = 4'b0010,

SUBalu = 4'b0011,

ANDalu = 4'b0100,

ORalu = 4'b0101,

XORalu = 4'b0110,

NOTalu = 4'b0111;

always @(OE, temp, posedge CLK) ALU\_OUT = (OE) ? temp : 8'bz;

always @(posedge CLK) begin

if(EN) begin

case (OPCODE)

ADDalu: begin

temp = A + B;

// Carry Flag

if (A+B> 2\*\*WIDTH -1) CF=1;

else CF=0;

// Signed Flag

if (temp[WIDTH-1]==1) SF=1;

else SF=0;

// Zero Flag

if(temp==0) ZF=1;

else ZF=0;

// Overflow Flag

if(A>0 && B>0 && temp<0) OF=1;

else if (A<0 && B<0 && temp>0) OF=1;

else OF=0;

end

SUBalu: begin

temp = A - B;

// Carry Flag

if (A<B) CF=1;

else CF=0;

// Signed Flag

if (temp[WIDTH-1]==1) SF=1;

else SF=0;

// Zero Flag

if(temp==0) ZF=1;

else ZF=0;

// Overflow Flag

if(A>0 && B>0 && temp<0) OF=1;

else if (A<0 && B<0 && temp>0) OF=1;

else OF=0;

end

ANDalu: begin

temp = A&B;

// Signed Flag

if(temp[WIDTH-1]==1) SF=1;

else SF=0;

// Zero Flag

if(temp==0) ZF=1;

else ZF=0;

end

ORalu: begin

temp = A|B;

// Signed Flag

if(temp[WIDTH-1]==1) SF=1;

else SF=0;

// Zero Flag

if(temp==0) ZF=1;

else ZF=0;

end

XORalu: begin

temp = A^B;

// Signed Flag

if(temp[WIDTH-1]==1) SF=1;

else SF=0;

// Zero Flag

if(temp==0) ZF=1;

else ZF=0;

end

NOTalu: begin

temp = ~A;

// Signed Flag

if(temp[WIDTH-1]==1) SF=1;

else SF=0;

// Zero Flag

if(temp==0) ZF=1;

else ZF=0;

end

default: temp = 1'bx;

endcase

end

end

endmodule

**Testbench:**

**ALU\_TB.sv**

`timescale 1ns/100ps

//`define SignedINTs // if not commented out, signed inputs are used

module ALU\_TB();

parameter WIDTH = 8;

reg CF, OF, SF, ZF;

reg [3:0] OPCODE;

reg CLK, EN, OE;

`ifdef SignedINTs begin

reg signed [ WIDTH-1 : 0] ALU\_OUT;

reg signed [ WIDTH-1 : 0] A, B;

end

`else begin

reg [ WIDTH-1 : 0] ALU\_OUT;

reg [ WIDTH-1 : 0] A, B;

end

`endif

ALU UUT(CLK, EN, OE, OPCODE, A, B, ALU\_OUT, CF, OF, SF, ZF);

initial begin

CLK=1'b0;

forever #20 CLK=~CLK;

end

initial $monitor("OPCODE = %b A =%4d B =%4d EN =%b OE =%b \n\t\tALU\_OUT =%4d CF =%b OF =%b SF =%b ZF =%b", OPCODE, A, B, EN, OE, ALU\_OUT, CF, OF, SF, ZF);

initial begin

$vcdpluson;

$display("\n\t\t\t Testing ADD");

$monitoroff; OE=1; EN=1; OPCODE = 4'b0010; #10 $monitoron;

#20 $display("ADD"); A=8'd6; B=8'd5;

#40 $display("ADD, Zero flag"); A=8'd0; B=8'd0;

#40 $display("ADD, Overflow and Carry flags"); A=8'd150; B=8'd106;

#40 $display("ADD, Signed flag"); A=8'd128; B=8'd1;

#60 $display("\n\t\t\t Testing SUB");

$monitoroff; OE=1; EN=1; OPCODE = 4'b0011; $monitoron;

#40 $display("SUB"); A=8'd60; B=8'd43;

#40 $display("SUB, Zero flag"); A=8'd20; B=8'd20;

#40 $display("SUB, Overflow flag"); A=8'd128; B=-8'd1;

#40 $display("SUB, Signed and Carry flags"); A=8'd50; B=8'd100;

#60 $display("\n\t\t\t Testing AND");

$monitoroff; OE=1; EN=1; OPCODE = 4'b0100; $monitoron;

#60 $display("AND"); A=8'd10; B=8'd10;

#40 $display("AND, Zero flag"); A=8'd20; B=8'd10;

#40 $display("AND, Signed flags"); A=8'd128; B=8'd128;

#60 $display("\n\t\t\t Testing OR");

$monitoroff; OE=1; EN=1; OPCODE = 4'b0101; $monitoron;

#60 $display("OR"); A=8'd20; B=8'd10;

#40 $display("OR, Zero flag"); A=8'd0; B=8'd0;

#40 $display("OR, Signed flags"); A=8'd10; B=-8'd5;

#60 $display("\n\t\t\t Testing XOR");

$monitoroff; OE=1; EN=1; OPCODE = 4'b0110; $monitoron;

#60 $display("XOR"); A=8'd37; B=8'd90;

#40 $display("XOR, Zero flag"); A=8'd255; B=8'd255;

#40 $display("XOR, Signed flags"); A=8'd128; B=8'd0;

#60 $display("\n\t\t\t Testing NOT");

$monitoroff; OE=1; EN=1; OPCODE = 4'b0111; $monitoron;

#60 $display("NOT"); A=8'd37;

#40 $display("NOT, Zero flag"); A=8'd255;

#40 $display("NOT, Signed flags"); A=8'd127;

#40 $finish;

end

endmodule

// vcs -debug -full64 -sverilog ALU.sv ALU\_TB.sv

**Log:**

**Unsigned Log**

Chronologic VCS simulator copyright 1991-2017

Contains Synopsys proprietary information.

Compiler version M-2017.03-SP1\_Full64; Runtime version M-2017.03-SP1\_Full64; Apr 12 17:12 2018

VCD+ Writer M-2017.03-SP1\_Full64 Copyright (c) 1991-2017 by Synopsys Inc.

Testing ADD

OPCODE = 0010 A = x B = x EN =1 OE =1

ALU\_OUT = x CF =x OF =x SF =x ZF =x

OPCODE = 0010 A = x B = x EN =1 OE =1

ALU\_OUT = x CF =0 OF =0 SF =0 ZF =0

ADD

OPCODE = 0010 A = 6 B = 5 EN =1 OE =1

ALU\_OUT = x CF =0 OF =0 SF =0 ZF =0

OPCODE = 0010 A = 6 B = 5 EN =1 OE =1

ALU\_OUT = 11 CF =0 OF =0 SF =0 ZF =0

ADD, Zero flag

OPCODE = 0010 A = 0 B = 0 EN =1 OE =1

ALU\_OUT = 11 CF =0 OF =0 SF =0 ZF =0

OPCODE = 0010 A = 0 B = 0 EN =1 OE =1

ALU\_OUT = 0 CF =0 OF =0 SF =0 ZF =1

ADD, Overflow and Carry flags

OPCODE = 0010 A = 150 B = 106 EN =1 OE =1

ALU\_OUT = 0 CF =0 OF =0 SF =0 ZF =1

OPCODE = 0010 A = 150 B = 106 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

ADD, Signed flag

OPCODE = 0010 A = 128 B = 1 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

OPCODE = 0010 A = 128 B = 1 EN =1 OE =1

ALU\_OUT = 129 CF =0 OF =0 SF =1 ZF =0

Testing SUB

OPCODE = 0011 A = 128 B = 1 EN =1 OE =1

ALU\_OUT = 129 CF =0 OF =0 SF =1 ZF =0

OPCODE = 0011 A = 128 B = 1 EN =1 OE =1

ALU\_OUT = 127 CF =0 OF =0 SF =0 ZF =0

SUB

OPCODE = 0011 A = 60 B = 43 EN =1 OE =1

ALU\_OUT = 127 CF =0 OF =0 SF =0 ZF =0

OPCODE = 0011 A = 60 B = 43 EN =1 OE =1

ALU\_OUT = 17 CF =0 OF =0 SF =0 ZF =0

SUB, Zero flag

OPCODE = 0011 A = 20 B = 20 EN =1 OE =1

ALU\_OUT = 17 CF =0 OF =0 SF =0 ZF =0

OPCODE = 0011 A = 20 B = 20 EN =1 OE =1

ALU\_OUT = 0 CF =0 OF =0 SF =0 ZF =1

SUB, Overflow flag

OPCODE = 0011 A = 128 B = 255 EN =1 OE =1

ALU\_OUT = 0 CF =0 OF =0 SF =0 ZF =1

OPCODE = 0011 A = 128 B = 255 EN =1 OE =1

ALU\_OUT = 129 CF =1 OF =0 SF =1 ZF =0

SUB, Signed and Carry flags

OPCODE = 0011 A = 50 B = 100 EN =1 OE =1

ALU\_OUT = 129 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0011 A = 50 B = 100 EN =1 OE =1

ALU\_OUT = 206 CF =1 OF =0 SF =1 ZF =0

Testing AND

OPCODE = 0100 A = 50 B = 100 EN =1 OE =1

ALU\_OUT = 206 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0100 A = 50 B = 100 EN =1 OE =1

ALU\_OUT = 32 CF =1 OF =0 SF =0 ZF =0

AND

OPCODE = 0100 A = 10 B = 10 EN =1 OE =1

ALU\_OUT = 32 CF =1 OF =0 SF =0 ZF =0

OPCODE = 0100 A = 10 B = 10 EN =1 OE =1

ALU\_OUT = 10 CF =1 OF =0 SF =0 ZF =0

AND, Zero flag

OPCODE = 0100 A = 20 B = 10 EN =1 OE =1

ALU\_OUT = 10 CF =1 OF =0 SF =0 ZF =0

OPCODE = 0100 A = 20 B = 10 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

AND, Signed flags

OPCODE = 0100 A = 128 B = 128 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

OPCODE = 0100 A = 128 B = 128 EN =1 OE =1

ALU\_OUT = 128 CF =1 OF =0 SF =1 ZF =0

Testing OR

OPCODE = 0101 A = 128 B = 128 EN =1 OE =1

ALU\_OUT = 128 CF =1 OF =0 SF =1 ZF =0

OR

OPCODE = 0101 A = 20 B = 10 EN =1 OE =1

ALU\_OUT = 128 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0101 A = 20 B = 10 EN =1 OE =1

ALU\_OUT = 30 CF =1 OF =0 SF =0 ZF =0

OR, Zero flag

OPCODE = 0101 A = 0 B = 0 EN =1 OE =1

ALU\_OUT = 30 CF =1 OF =0 SF =0 ZF =0

OPCODE = 0101 A = 0 B = 0 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

OR, Signed flags

OPCODE = 0101 A = 10 B = 251 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

OPCODE = 0101 A = 10 B = 251 EN =1 OE =1

ALU\_OUT = 251 CF =1 OF =0 SF =1 ZF =0

Testing XOR

OPCODE = 0110 A = 10 B = 251 EN =1 OE =1

ALU\_OUT = 251 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0110 A = 10 B = 251 EN =1 OE =1

ALU\_OUT = 241 CF =1 OF =0 SF =1 ZF =0

XOR

OPCODE = 0110 A = 37 B = 90 EN =1 OE =1

ALU\_OUT = 241 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0110 A = 37 B = 90 EN =1 OE =1

ALU\_OUT = 127 CF =1 OF =0 SF =0 ZF =0

XOR, Zero flag

OPCODE = 0110 A = 255 B = 255 EN =1 OE =1

ALU\_OUT = 127 CF =1 OF =0 SF =0 ZF =0

OPCODE = 0110 A = 255 B = 255 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

XOR, Signed flags

OPCODE = 0110 A = 128 B = 0 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

OPCODE = 0110 A = 128 B = 0 EN =1 OE =1

ALU\_OUT = 128 CF =1 OF =0 SF =1 ZF =0

Testing NOT

OPCODE = 0111 A = 128 B = 0 EN =1 OE =1

ALU\_OUT = 128 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0111 A = 128 B = 0 EN =1 OE =1

ALU\_OUT = 127 CF =1 OF =0 SF =0 ZF =0

NOT

OPCODE = 0111 A = 37 B = 0 EN =1 OE =1

ALU\_OUT = 127 CF =1 OF =0 SF =0 ZF =0

OPCODE = 0111 A = 37 B = 0 EN =1 OE =1

ALU\_OUT = 218 CF =1 OF =0 SF =1 ZF =0

NOT, Zero flag

OPCODE = 0111 A = 255 B = 0 EN =1 OE =1

ALU\_OUT = 218 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0111 A = 255 B = 0 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

NOT, Signed flags

OPCODE = 0111 A = 127 B = 0 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

OPCODE = 0111 A = 127 B = 0 EN =1 OE =1

ALU\_OUT = 128 CF =1 OF =0 SF =1 ZF =0

$finish called from file "ALU\_TB.sv", line 72.

$finish at simulation time 12100

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

Time: 1210000 ps

CPU Time: 0.230 seconds; Data structure size: 0.0Mb

Thu Apr 12 17:12:56 2018

**Signed Log**

Chronologic VCS simulator copyright 1991-2017

Contains Synopsys proprietary information.

Compiler version M-2017.03-SP1\_Full64; Runtime version M-2017.03-SP1\_Full64; Apr 12 17:02 2018

VCD+ Writer M-2017.03-SP1\_Full64 Copyright (c) 1991-2017 by Synopsys Inc.

Testing ADD

OPCODE = 0010 A = x B = x EN =1 OE =1

ALU\_OUT = x CF =x OF =x SF =x ZF =x

OPCODE = 0010 A = x B = x EN =1 OE =1

ALU\_OUT = x CF =0 OF =0 SF =0 ZF =0

ADD

OPCODE = 0010 A = 6 B = 5 EN =1 OE =1

ALU\_OUT = x CF =0 OF =0 SF =0 ZF =0

OPCODE = 0010 A = 6 B = 5 EN =1 OE =1

ALU\_OUT = 11 CF =0 OF =0 SF =0 ZF =0

ADD, Zero flag

OPCODE = 0010 A = 0 B = 0 EN =1 OE =1

ALU\_OUT = 11 CF =0 OF =0 SF =0 ZF =0

OPCODE = 0010 A = 0 B = 0 EN =1 OE =1

ALU\_OUT = 0 CF =0 OF =0 SF =0 ZF =1

ADD, Overflow and Carry flags

OPCODE = 0010 A =-106 B = 106 EN =1 OE =1

ALU\_OUT = 0 CF =0 OF =0 SF =0 ZF =1

OPCODE = 0010 A =-106 B = 106 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

ADD, Signed flag

OPCODE = 0010 A =-128 B = 1 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

OPCODE = 0010 A =-128 B = 1 EN =1 OE =1

ALU\_OUT =-127 CF =0 OF =0 SF =1 ZF =0

Testing SUB

OPCODE = 0011 A =-128 B = 1 EN =1 OE =1

ALU\_OUT =-127 CF =0 OF =0 SF =1 ZF =0

OPCODE = 0011 A =-128 B = 1 EN =1 OE =1

ALU\_OUT = 127 CF =0 OF =0 SF =0 ZF =0

SUB

OPCODE = 0011 A = 60 B = 43 EN =1 OE =1

ALU\_OUT = 127 CF =0 OF =0 SF =0 ZF =0

OPCODE = 0011 A = 60 B = 43 EN =1 OE =1

ALU\_OUT = 17 CF =0 OF =0 SF =0 ZF =0

SUB, Zero flag

OPCODE = 0011 A = 20 B = 20 EN =1 OE =1

ALU\_OUT = 17 CF =0 OF =0 SF =0 ZF =0

OPCODE = 0011 A = 20 B = 20 EN =1 OE =1

ALU\_OUT = 0 CF =0 OF =0 SF =0 ZF =1

SUB, Overflow flag

OPCODE = 0011 A =-128 B = -1 EN =1 OE =1

ALU\_OUT = 0 CF =0 OF =0 SF =0 ZF =1

OPCODE = 0011 A =-128 B = -1 EN =1 OE =1

ALU\_OUT =-127 CF =1 OF =0 SF =1 ZF =0

SUB, Signed and Carry flags

OPCODE = 0011 A = 50 B = 100 EN =1 OE =1

ALU\_OUT =-127 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0011 A = 50 B = 100 EN =1 OE =1

ALU\_OUT = -50 CF =1 OF =0 SF =1 ZF =0

Testing AND

OPCODE = 0100 A = 50 B = 100 EN =1 OE =1

ALU\_OUT = -50 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0100 A = 50 B = 100 EN =1 OE =1

ALU\_OUT = 32 CF =1 OF =0 SF =0 ZF =0

AND

OPCODE = 0100 A = 10 B = 10 EN =1 OE =1

ALU\_OUT = 32 CF =1 OF =0 SF =0 ZF =0

OPCODE = 0100 A = 10 B = 10 EN =1 OE =1

ALU\_OUT = 10 CF =1 OF =0 SF =0 ZF =0

AND, Zero flag

OPCODE = 0100 A = 20 B = 10 EN =1 OE =1

ALU\_OUT = 10 CF =1 OF =0 SF =0 ZF =0

OPCODE = 0100 A = 20 B = 10 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

AND, Signed flags

OPCODE = 0100 A =-128 B =-128 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

OPCODE = 0100 A =-128 B =-128 EN =1 OE =1

ALU\_OUT =-128 CF =1 OF =0 SF =1 ZF =0

Testing OR

OPCODE = 0101 A =-128 B =-128 EN =1 OE =1

ALU\_OUT =-128 CF =1 OF =0 SF =1 ZF =0

OR

OPCODE = 0101 A = 20 B = 10 EN =1 OE =1

ALU\_OUT =-128 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0101 A = 20 B = 10 EN =1 OE =1

ALU\_OUT = 30 CF =1 OF =0 SF =0 ZF =0

OR, Zero flag

OPCODE = 0101 A = 0 B = 0 EN =1 OE =1

ALU\_OUT = 30 CF =1 OF =0 SF =0 ZF =0

OPCODE = 0101 A = 0 B = 0 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

OR, Signed flags

OPCODE = 0101 A = 10 B = -5 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

OPCODE = 0101 A = 10 B = -5 EN =1 OE =1

ALU\_OUT = -5 CF =1 OF =0 SF =1 ZF =0

Testing XOR

OPCODE = 0110 A = 10 B = -5 EN =1 OE =1

ALU\_OUT = -5 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0110 A = 10 B = -5 EN =1 OE =1

ALU\_OUT = -15 CF =1 OF =0 SF =1 ZF =0

XOR

OPCODE = 0110 A = 37 B = 90 EN =1 OE =1

ALU\_OUT = -15 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0110 A = 37 B = 90 EN =1 OE =1

ALU\_OUT = 127 CF =1 OF =0 SF =0 ZF =0

XOR, Zero flag

OPCODE = 0110 A = -1 B = -1 EN =1 OE =1

ALU\_OUT = 127 CF =1 OF =0 SF =0 ZF =0

OPCODE = 0110 A = -1 B = -1 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

XOR, Signed flags

OPCODE = 0110 A =-128 B = 0 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

OPCODE = 0110 A =-128 B = 0 EN =1 OE =1

ALU\_OUT =-128 CF =1 OF =0 SF =1 ZF =0

Testing NOT

OPCODE = 0111 A =-128 B = 0 EN =1 OE =1

ALU\_OUT =-128 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0111 A =-128 B = 0 EN =1 OE =1

ALU\_OUT = 127 CF =1 OF =0 SF =0 ZF =0

NOT

OPCODE = 0111 A = 37 B = 0 EN =1 OE =1

ALU\_OUT = 127 CF =1 OF =0 SF =0 ZF =0

OPCODE = 0111 A = 37 B = 0 EN =1 OE =1

ALU\_OUT = -38 CF =1 OF =0 SF =1 ZF =0

NOT, Zero flag

OPCODE = 0111 A = -1 B = 0 EN =1 OE =1

ALU\_OUT = -38 CF =1 OF =0 SF =1 ZF =0

OPCODE = 0111 A = -1 B = 0 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

NOT, Signed flags

OPCODE = 0111 A = 127 B = 0 EN =1 OE =1

ALU\_OUT = 0 CF =1 OF =0 SF =0 ZF =1

OPCODE = 0111 A = 127 B = 0 EN =1 OE =1

ALU\_OUT =-128 CF =1 OF =0 SF =1 ZF =0

$finish called from file "ALU\_TB.sv", line 72.

$finish at simulation time 12100

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

Time: 1210000 ps

CPU Time: 0.220 seconds; Data structure size: 0.0Mb

Thu Apr 12 17:02:47 2018

**Conclusion:**

Everything in this lab worked as expected, except for the signed input cases. This was because of user error. The if conditions for the flags were not properly set to deal with signed values. Also, the test vectors used in the test bench did not adequately test the signed values.

With regards to the lab question, yes, signed variables can be used for the data path of inputs A, B and the output, ALU\_OUT. One thing that will have to be considered is that when dealing with signed bits, you must make sure to make signed variables of the data type signed. You also should make sure that the way some of the flags are set up will be considered for when dealing with signed values, which in hindsight, is a mistake done in this lab.