Fall 2019

California State University, Northridge

Department of Electrical and Computer Engineering

ECE 526L

Experiment # 8
Arithmetic-Logic Unit Modeling

November 7, 2019
SUBMITTED: December 10, 2019

Authors: Ridge Tejuco
Professor: Ronald Mehler Ph.D

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) _____

Name (signed) _____

Date _____

Analysis of ALU

The following test vectors were tested in the ALU

A: 2

B: 17

Figure 1: Test bench for A = 2, B = 17

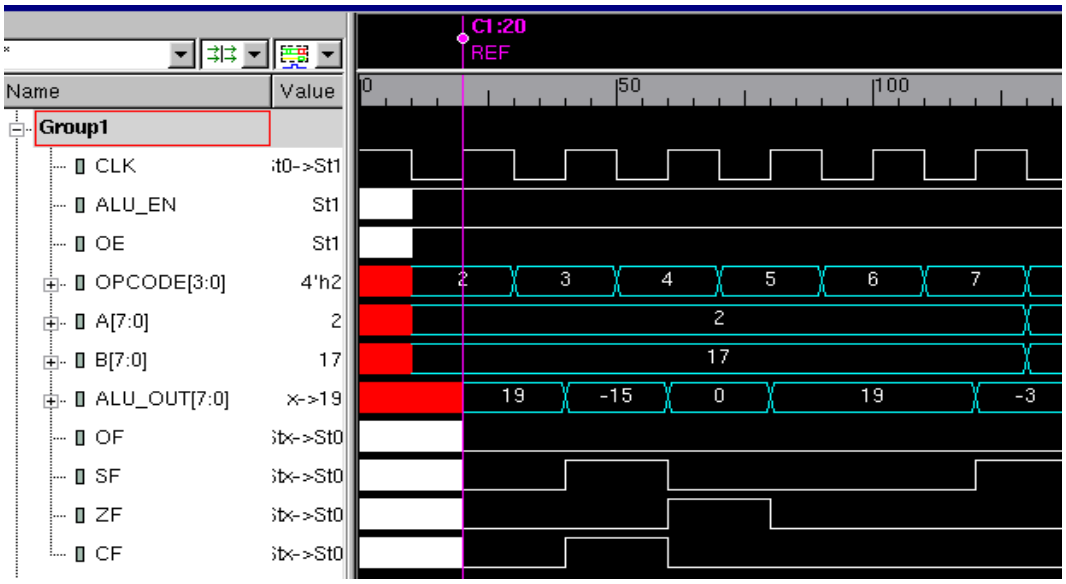


Figure 1 shows the following results starting at 20 ps.

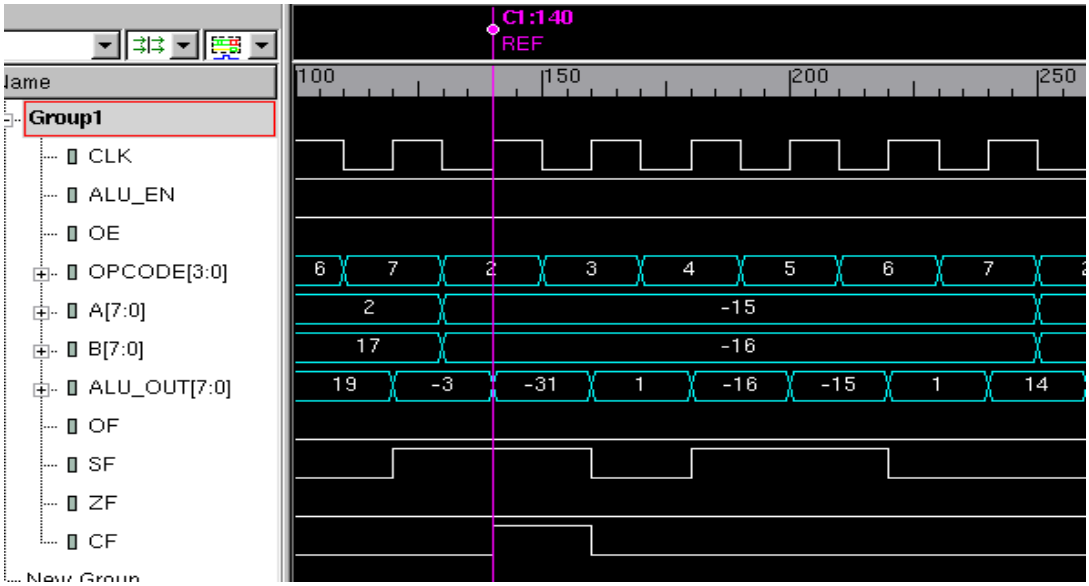
Result		Verification	Flags set
A + B = 19	------	2 + 17 = 19	none
A - B = -15	------	2 - 17 = -15	
negative,borrow			
A & B = 0	------	00000010	
		AND 00010001	
		00000000	zero
A | B = 19	------	00000010	
		OR 00010001	
		00010011 = $(19)_{decimal}$	none
A ^ B = 19	------	00000010	none
		XOR 00010001	
		00010011 = $(19)_{decimal}$	
~A = -3	------	~ 00000010	
		11111101 = $(-3)_{decimal}$	negative

A - B sets the negative flag because it produces a negative answer and it sets the carry flag because A is less than B, meaning this operation needs a borrow.A & B produces a zero answer and therefore sets the zero flag. Next, the following test vector was tested starting at 140 ps.

A: -15

B: -16

Figure 2. A = -15, B = -16

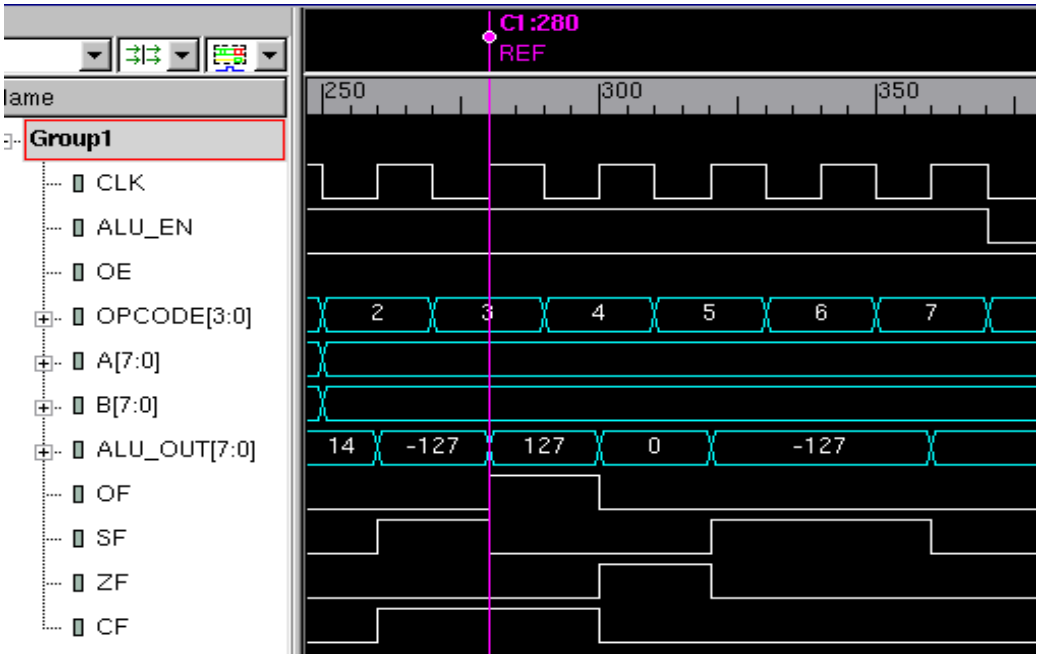


Result		Verification	Flags set
A + B = 19	------	-15 + -16 = -31	negative,carry
		11110001	
		+ 11110000	
		1 11100001 = $(-31)_{decimal}$	
A - B = -15	------	-15 - (-16) = 1	none
A & B = 0	------	11110001	
		AND 11110000	
		11110000 = $(-16)_{decimal}$	negative
A | B = 19	------	11110001	
		OR 11110000	
		11110001 = $(-15)_{decimal}$	negative
A ^ B = 19	------	11110001	
		XOR 11110000	
		00000001 = $(1)_{decimal}$	none

~A = 14 ------ ~ 11110001
00001110 = $(14)_{decimal}$ none

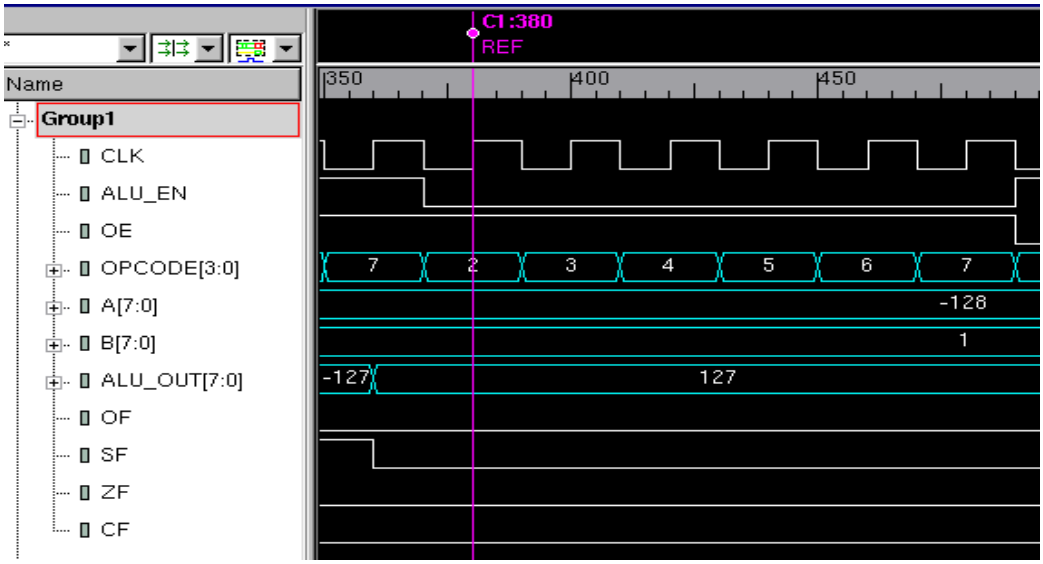
This following vector was used to test overflow.
A: -128
B: 1

Figure 3. Overflow test



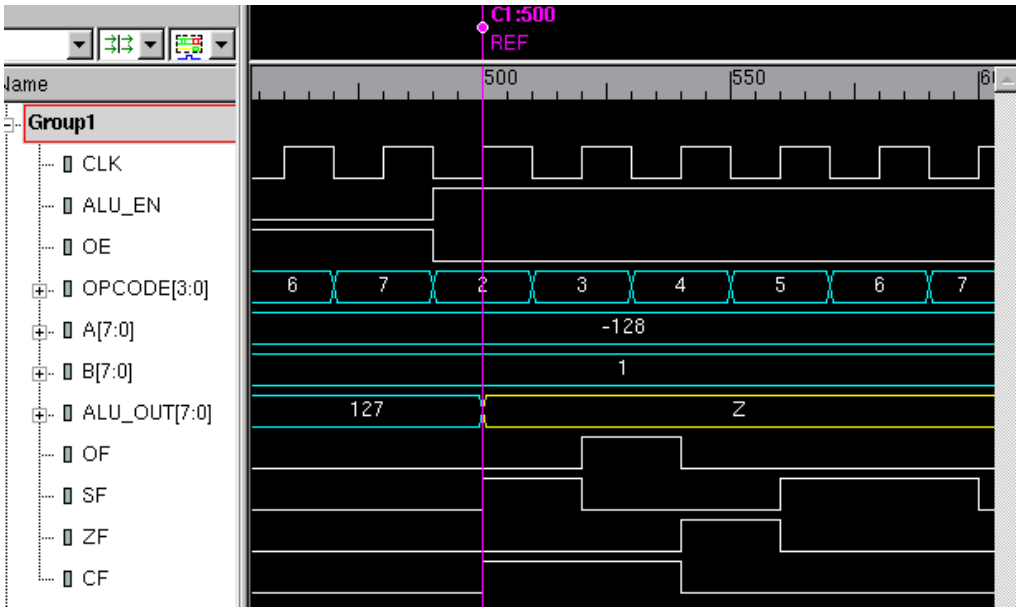
At 280 ps the overflow flag is set. The current operation is A - B. This overflow occurs because the result of (-128 - 1) is -129 which does not fit our range of possible values -128 to 127. Instead the operation loops back over and produces the wrong result of 127.

Figure 4 ALU_EN disabled



At 380 ps, ALU_EN and regardless of the operation change the ALU_OUT and the flags remain the same result.

Figure 5. OE disabled

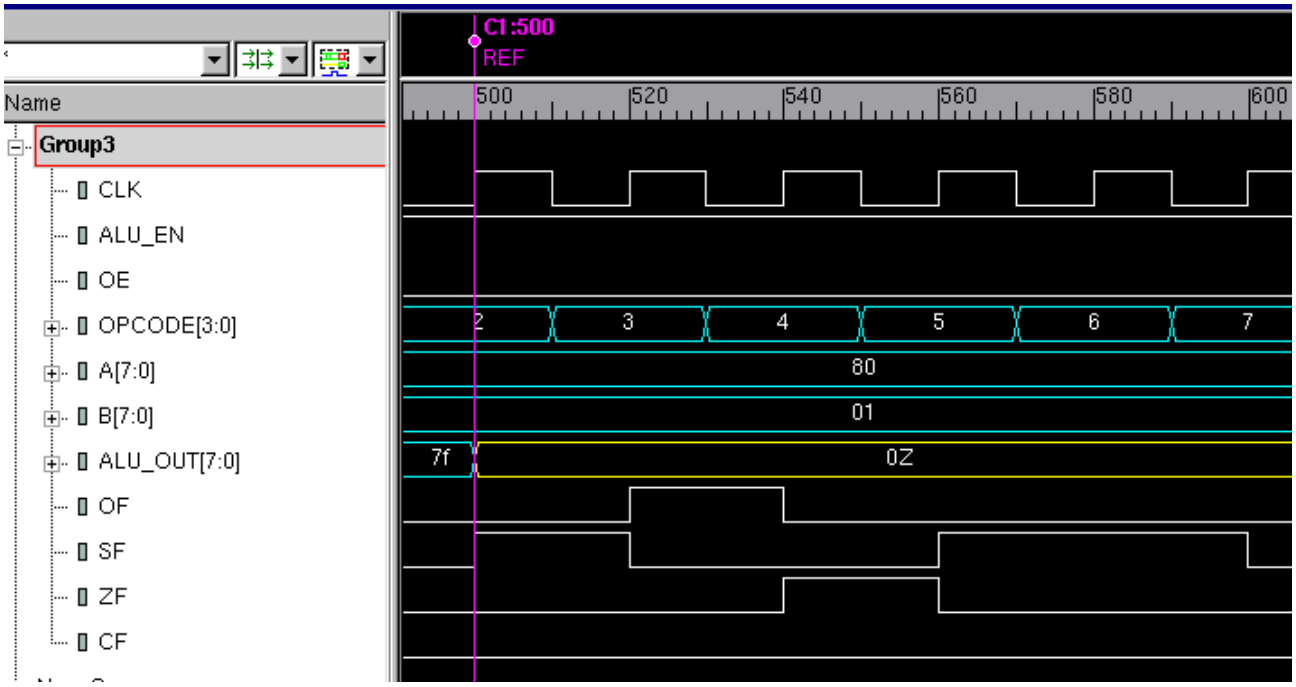


At 500 ps, OE is clocked low. The flags continue to change based on the current operation, but the output remains high impedance regardless of the operation.

The signed declaration in verilog does not change the arithmetic of the ALU_OUT, but it does affect the interpretation of some of our flags.
A problem could occur when adding -128 and 1 signed values under the following code.

```
{CF, ALU_OUT} <= A + B;
```

Figure 6. Carry out of unsigned A + B @ 260 ps



Looking closer at Figure 5 and Figure 6, CF is set for signed A + B, but not for unsigned A + B.

When signed, A and B are extended by the sign value of the Most significant bit.
When unsigned A and B are extended by zeroes.

```
Carry out                      no carry out
  1 10000000                     0 10000000
+ 0 00000001                   + 0 00000001
  1 10000001                     0 10000001
```

This problem can be solved my manually extending the addition when assigning the carry bit by the following code.

```
{CF, ALU_OUT} <= {1'b0,A} +{1'b0,B};
```

```
//ALU.sv
/*==================================
Ridge Tejuco
CSUN ECE 526 LAB 08
==================================*/
`timescale 1 ns / 100ps
module ALU (CLK, ALU_EN, OE, OPCODE, A, B, ALU_OUT,OF,SF,ZF,CF);
    parameter OP_WIDTH = 4, BUS_WIDTH = 8;
    localparam AB_SUM = 4'b0010,
               AB_SUB = 4'b0011,
               AB_AND = 4'b0100,
               AB_OR = 4'b0101,
               AB_XOR = 4'b0110,
               A_NOT = 4'b0111;

    input CLK, ALU_EN, OE;
    input [OP_WIDTH-1:0] OPCODE;
    input [BUS_WIDTH-1:0] A,B;
    output reg [BUS_WIDTH-1:0] ALU_OUT;
    output reg OF,SF,ZF,CF;
    reg [BUS_WIDTH-1:0] RESULT;

    always_ff@(posedge CLK) begin
     if(ALU_EN) begin
           ZF <= ~|(RESULT);
           SF <= RESULT[BUS_WIDTH-1];
           case(OPCODE) inside
                4'b0010: begin
                     {CF,ALU_OUT} <= {1'b0,A} + {1'b0,B};
                     if(A[BUS_WIDTH-1] == B[BUS_WIDTH-1] &&
RESULT[BUS_WIDTH-1] != A[BUS_WIDTH-1])
                          OF <= 1'b1;
                     else
                          OF <= 1'b0;
                end
                4'b0011: begin
                     CF <= A < B;
                     ALU_OUT <= RESULT;
```

```
                    if(A[BUS_WIDTH-1] != B[BUS_WIDTH-1] &&
RESULT[BUS_WIDTH-1] != A[BUS_WIDTH-1])
                         OF <= 1'b1;
                    else
                         OF <= 1'b0;
               end
               4'b01??: begin
                    CF <= 1'b0;
                    OF <= 1'b0;
                    ALU_OUT <= RESULT;
               end
               default: begin
                    CF <= CF;
                    OF <= OF;
                    ALU_OUT <= ALU_OUT;
               end
          endcase
      end else
          ALU_OUT <= ALU_OUT;
     if(~OE)
     ALU_OUT <= {BUS_WIDTH-1,(1'bz)};
    end

    always_comb begin
      case(OPCODE)
          4'b0010: RESULT <= A+B;
          4'b0011: RESULT <= A+(~B)+1;
          4'b0111: RESULT <= ~A;
          4'b0100: RESULT <= A&B;
          4'b0101: RESULT <= A|B;
          4'b0110: RESULT <= A^B;
          default: RESULT <= RESULT;
      endcase
    end
endmodule
```

```
//TB_ALU.sv
/*==================================
Ridge Tejuco
CSUN ECE 526 LAB 08
==================================*/
`timescale 1 ns / 100ps
module TB_ALU ();
    parameter DELAY = 2,OP_WIDTH = 4, BUS_WIDTH = 8;
    localparam AB_SUM = 4'b0010,
               AB_SUB = 4'b0011,
               AB_AND = 4'b0100,
               AB_OR = 4'b0101,
               AB_XOR = 4'b0110,
               A_NOT = 4'b0111;

    reg CLK, ALU_EN, OE;
    reg [OP_WIDTH-1:0] OPCODE;
    reg [BUS_WIDTH-1:0] A,B;
    reg [BUS_WIDTH-1:0] ALU_OUT;
    reg OF,SF,ZF,CF;
    reg [BUS_WIDTH-1:0] RESULT;

    //alu (CLK, ALU_EN, OE, OPCODE, A, B, ALU_OUT,OF,SF,ZF,CF);
    ALU DUT(CLK,ALU_EN,OE,OPCODE,A,B,ALU_OUT,OF,SF,ZF,CF);
    initial begin
      CLK = 1'b1;
      forever #(DELAY/2) CLK = ~CLK;
    end

    initial begin
      $vcdpluson;
     // TESTING 2 and 17
           #1 ALU_EN = 1'b1; OE = 1'b1;
           A = 8'h02;B = 8'h11;
           OPCODE = 4'b0010;
           repeat(5) #DELAY OPCODE = OPCODE + 1;

     //TESTING -16 and -15
           #DELAY A = 8'hF1;B = 8'hF0;
           OPCODE = 4'b0010;
```

```
            repeat(5) #DELAY OPCODE = OPCODE + 1;

      //TESTING -128 and 1
            #DELAY A = 8'h80;B = 8'h01;
            OPCODE = 4'b0010;
            repeat(5) #DELAY OPCODE = OPCODE + 1;

      //TESTING DISABLED ALU_EN
            #DELAY OPCODE = 4'b0010; ALU_EN = 1'b0;
         repeat(5) #DELAY OPCODE = OPCODE + 1;

      //TESTING DISABLED OE
            #DELAY ALU_EN = 1'b1; OE = 1'b0;OPCODE = 4'b0010;
      repeat(5) #DELAY OPCODE = OPCODE + 1;
      #DELAY;
       $finish;
     end
endmodule
```