Bharat Kathi
ECE 154A
10/6/23

Lab 1 Report

Number of hours spent on lab

I spent about 12 hours on this lab.

Table 1

Test	F[2:0]	A	B	Result	Zero	Over	Car	Neg
ADD 0+0	0	00000000	00000000	00000000	1	0	0	0
ADD 0+(-1)	0	00000000	FFFFFFFF	FFFFFFFF	0	0	0	1
ADD 1+(-1)	0	00000001	FFFFFFFF	00000000	1	0	0	0
ADD FF + 1	0	000000FF	00000001	00000100	0	0	0	0
ADD 7FFFFFFF + 1	0	7FFFFFFF	00000001	80000000	0	1	0	1
SUB 0-0	1	00000000	00000000	00000000	1	0	0	0
SUB 0-(-1)	1	00000000	FFFFFFFF	00000001	0	0	0	0
SUB 1-1	1	00000001	00000001	00000000	1	0	1	0
SUB 100-1	1	00000100	00000001	000000FF	0	0	1	0
SUB 80000000 - 1	1	80000000	00000001	7FFFFFFF	0	1	1	0
SLT 0,0	5	00000000	00000000	00000000	1	0	0	0
SLT 0,1	5	00000000	00000001	00000001	0	0	0	0
SLT 0,-1	5	00000000	FFFFFFFF	00000000	1	0	0	0
SLT 1,0	5	00000001	00000000	00000000	1	0	0	0

Test	F[2:0]	A	B	Result	Zero	Over	Car	Neg
SLT -1,0	5	FFFFFFFF	00000000	00000001	0	0	0	0
AND FFFFFFFF, FFFFFFFF	2	FFFFFFFF	FFFFFFFF	FFFFFFFF	0	0	0	1
AND FFFFFFFF, 12345678	2	FFFFFFFF	12345678	12345678	0	0	0	0
AND 12345678, 87654321	2	12345678	87654321	02244220	0	0	0	0
AND 00000000, FFFFFFFF	2	00000000	FFFFFFFF	00000000	1	0	0	0
OR FFFFFFFF, FFFFFFFF	3	FFFFFFFF	FFFFFFFF	FFFFFFFF	0	0	0	1
OR 12345678, 87654321	3	12345678	87654321	97755779	0	0	0	1
OR 00000000, FFFFFFFF	3	00000000	FFFFFFFF	FFFFFFFF	0	0	0	1
OR 00000000, 00000000	3	00000000	00000000	00000000	1	0	0	0

alu.v

```
module alu(input [31:0] a, b,
    input [2:0] f,
    output [31:0] result,
    output zero,
    output overflow,
    output carry,
    output negative);

    reg [31:0] y;
    reg [31:0] z;
    wire [32:0] sum;

    assign sum = a + z;
    assign result = y;
    assign zero = ~|y;

    assign overflow = (a[31] ^ sum[31]) & (f[0] ^ a[31] ^ b[31]) & f[1];
    assign carry = sum[32] & !f[1] & !f[2];
    assign negative = y[31];

    always @(*) begin
        case (f[0])
            1'b0: z = b;
            1'b1: z = ~b + 32'b1;
        endcase
    end

    always @(*) begin
        case (f)
            3'b000: y = sum[31:0];
            3'b001: y = sum[31:0];
            3'b010: y = a & b;
            3'b011: y = a | b;
            3'b101: y = (sum[31] ^ overflow) ? 32'b1 : 32'b0;
            default: y = 32'b0;
        endcase
    end

endmodule
```

alu.tv

```
2   00000000    00000000    00000000    1
2   00000000    FFFFFFFF    FFFFFFFF    0
2   00000001    FFFFFFFF    00000000    1
2   000000FF    00000001    00000100    0
6   00000000    00000000    00000000    1
6   00000000    FFFFFFFF    00000001    0
6   00000001    00000001    00000000    1
6   00000100    00000001    000000FF    0
7   00000000    00000000    00000000    1
7   00000000    00000001    00000001    0
7   00000000    FFFFFFFF    00000000    1
```

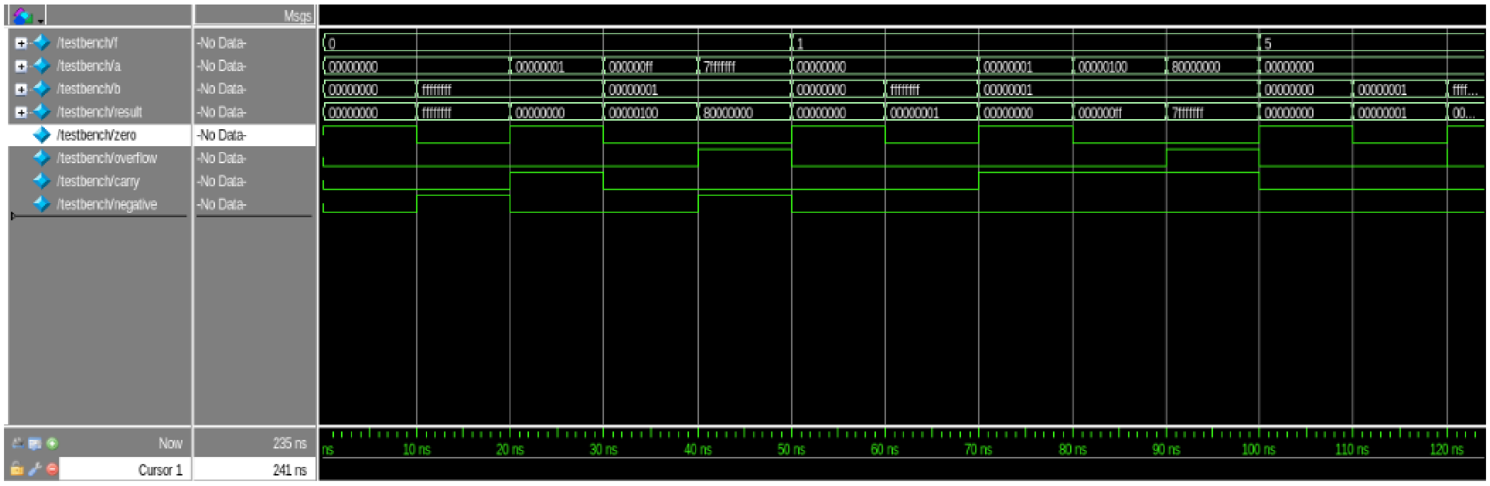
```
7   00000001    00000000    00000000    1
7   FFFFFFFF    00000000    00000001    0
0   FFFFFFFF    FFFFFFFF    FFFFFFFF    0
0   FFFFFFFF    12345678    12345678    0
0   12345678    87654321    02244220    0
0   00000000    FFFFFFFF    00000000    1
1   FFFFFFFF    FFFFFFFF    FFFFFFFF    0
1   12345678    87654321    97755779    0
1   00000000    FFFFFFFF    FFFFFFFF    0
1   00000000    00000000    00000000    1
```

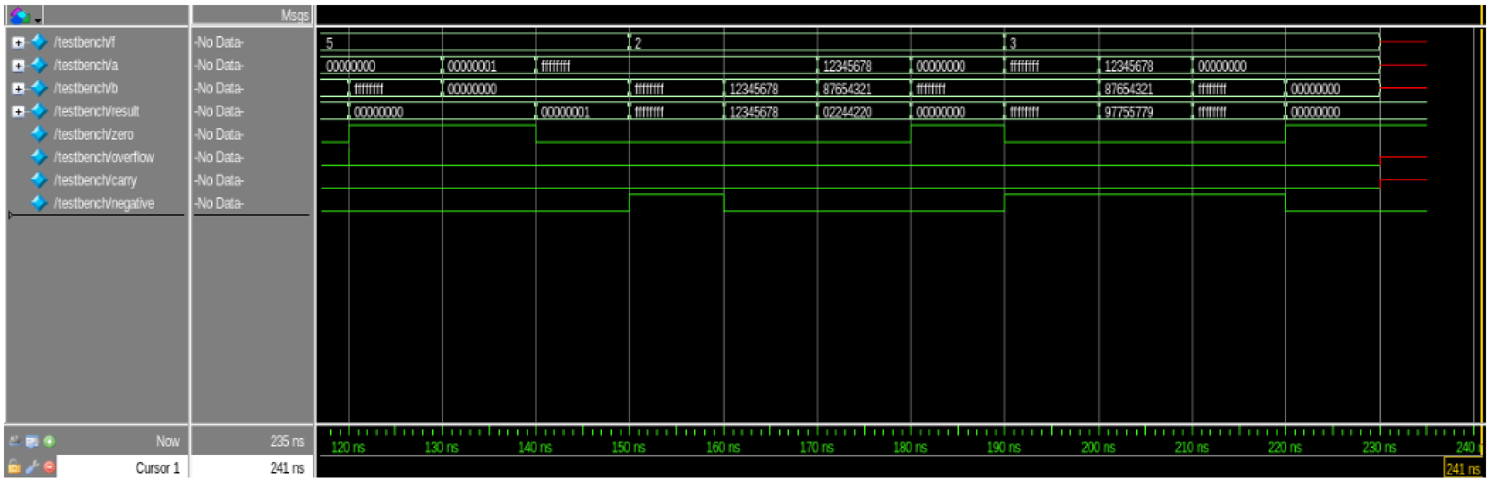
testbench.v

```
`timescale 1ns/1ns
`include "alu.v"

module testbench;

reg [31:0] data [0:183];
reg [31:0] input_a;
reg [31:0] input_b;
reg [2:0] input_f;

wire zero_output;
wire overflow_output;
wire carry_output;
wire negative_output;
wire [31:0] result_output;

integer i;
reg[31:0] check[0:4];
reg[31:0] DUToutput[0:4];

initial $readmemh("alu.tv", data);

alu DUT(
    .a(input_a),
    .b(input_b),
    .f(input_f),
    .zero(zero_output),
    .overflow(overflow_output),
    .carry(carry_output),
    .negative(negative_output),
    .result(result_output)
);

initial begin
```

```
    for (i = 0; i < 184; i = i + 8) begin
        input_a = data[i];
        input_b = data[i + 1];
        input_f = data[i + 2];
        check[0] = data[i + 3];
        check[1] = data[i + 4];
        check[2] = data[i + 5];
        check[3] = data[i + 6];
        check[4] = data[i + 7];
        #1;
        DUToutput[0] = result_output;
        DUToutput[1] = zero_output;
        DUToutput[2] = overflow_output;
        DUToutput[3] = carry_output;
        DUToutput[4] = negative_output;
        if (DUToutput != check) begin
            $display("Error at %d", i);
            $display("Expected: %b %b %b %b %b", check[0], check[1], check[2], check[3
            $display("Got:      %b %b %b %b %b", DUToutput[0], DUToutput[1], DUToutput
            $finish;
        end
    end
    $display("All tests passed!");
end

endmodule
```

Images of Waveforms


Msgs
/testbench/f
/testbench/a
/testbench/b
/testbench/result
/testbench/zero
/testbench/overflow
/testbench/carry
/testbench/negative
-No Data-
Now 235 ns
Cursor 1 241 ns
0 1 5
00000000 00000001 000000ff 7fffffff 00000000 00000001 00000100 80000000 00000000
00000000 ffffffff 00000001 00000000 ffffffff 00000001 00000000 00000001 ffff...
00000000 ffffffff 00000000 00000100 80000000 00000000 00000001 00000000 000000ff 7fffffff 00000000 00000001 00...
ns 10 ns 20 ns 30 ns 40 ns 50 ns 60 ns 70 ns 80 ns 90 ns 100 ns 110 ns 120 ns


Msgs
/testbench/f
/testbench/a
/testbench/b
/testbench/result
/testbench/zero
/testbench/overflow
/testbench/carry
/testbench/negative
-No Data-
Now 235 ns
Cursor 1 241 ns
5 2 3
00000000 00000001 ffffffff 12345678 00000000 ffffffff 12345678 00000000
ffffffff 00000000 ffffffff 12345678 87654321 ffffffff 87654321 ffffffff 00000000
00000000 00000001 ffffffff 12345678 02244220 00000000 ffffffff 97755779 ffffffff 00000000
120 ns 130 ns 140 ns 150 ns 160 ns 170 ns 180 ns 190 ns 200 ns 210 ns 220 ns 230 ns 240
241 ns

Lab Feedback

I think the part that took the longest was figuring out how to create the testbench file since I don't believe we ever used a test vector file before. Maybe providing some examples on this would be helpful for the future.