Computer Engineering - ELC 363

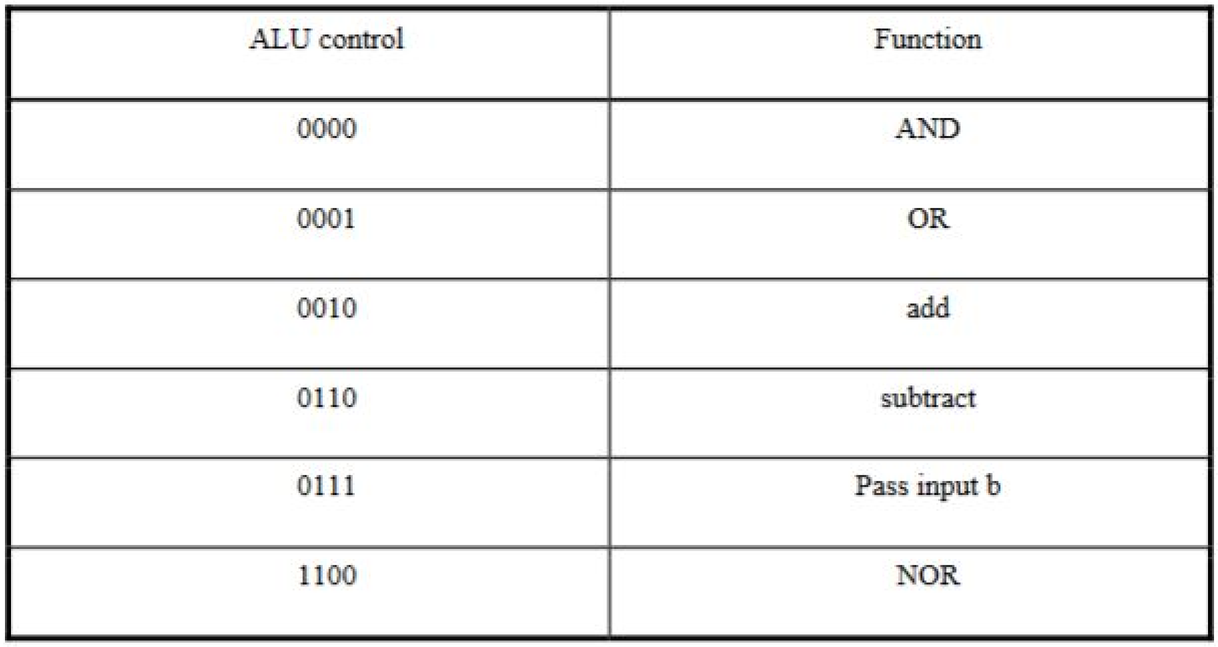
Lab 4 Report: ARM ALU

Nick Smith and Brian Dawson

Due: 11/07/2019

Introduction and Problem Description:

In this lab, students created a simple arithmetic logic unit (ALU) using Xilinx Vivado software. The ALU was a 64 bit ARM and the design approach for the lab was bottoms up. The ALU also did not support overflow detection and was based on the following function table.



ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	Pass input b
1100	NOR

Another additional requirement for the ALU was that it had to have an active high output if the result of the ALU was zero.

Results:

a. Verilog Design Code

```
`timescale 1ns / 1ps

module alu(
       input wire [63:0] a,b,
       input wire [3:0] control,
       output reg [63:0] c_out,
       output reg z
       );

       always @(control) begin
```

```
        case( control )

        4'b0000: // AND
               c_out = a & b;
        4'b0001: // OR
               c_out = a | b;
        4'b0010: // add
               c_out = a + b;
        4'b0110: //subtract
               c_out = a - b;
        4'b0111: // Pass input b
               c_out = b;
        4'b1100: // NOR
               c_out = ~(a | b);
        endcase

        if( c_out == 0)
        begin
        z = 1'b1;
        end
        else
        begin
        z = 1'b0;
        end
        end
endmodule
```

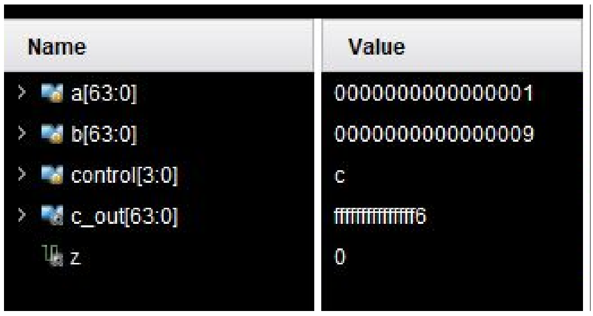
b. Verilog Test Code

```
`timescale 1ns / 1ps

module testbench;
//Input
        reg [63:0] a;
        reg [63:0] b;
        reg [3:0] control;
//Output
        wire[63:0] c_out;
        wire z;

        alu uut(.a(a), .b(b), .control(control), .c_out(c_out), .z(z));

        initial begin

   control = 4'b0000;
   a = 0;
   b = 0;
   #10
   control = 4'b0001;
   a = 1;
   b = 8;
   #10
   control = 4'b0010;
   a = 4;
   b = 8;
   #10
   control = 4'b0110;
   a = 5;
   b = 10;
   #10
   control = 4'b0111;
   a = 0;
   b = 6;
   #10
   control = 4'b1100;
   a = 50;
   b = 70;

   end
       initial #50 $finish;
endmodule
```

c. Waveforms for all functions

<u>Test Case 1:</u>

Operation	Input A (decimal)	Input B (decimal)
AND	6	3
OR	2	4
add	5	1
subtract	8	4
Pass input b	1	6
NOR	1	9



Name	Value
a[63:0]	0000000000000001
b[63:0]	0000000000000009
control[3:0]	c
c_out[63:0]	fffffffffffffff6
z	0

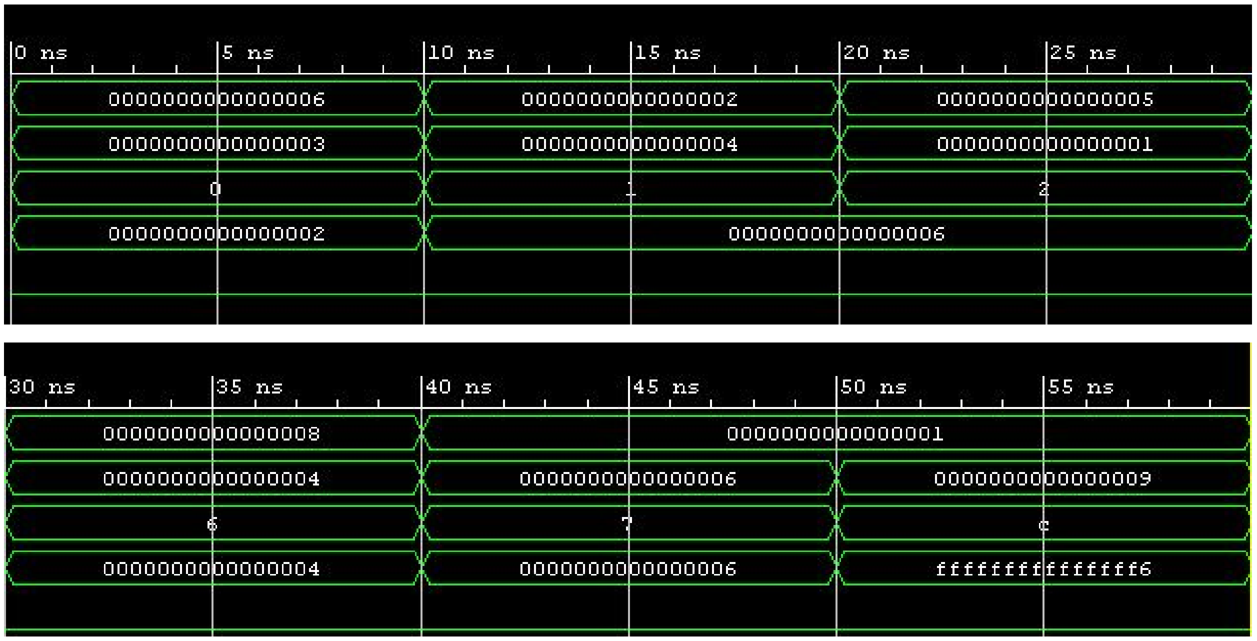
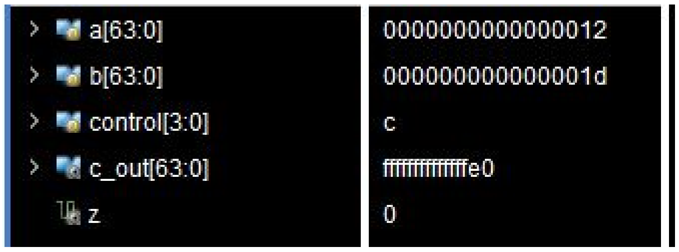


Figure 1: Trial 1 Waveform generation

<u>Test Case 2:</u>

Operation	Input A (decimal)	Input B (decimal)
AND	20	14
OR	16	19
add	30	13
subtract	32	41
Pass input b	11	26
NOR	18	29



```
a[63:0]         0000000000000012
b[63:0]         000000000000001d
control[3:0]    c
c_out[63:0]     ffffffffffffffe0
z               0
```

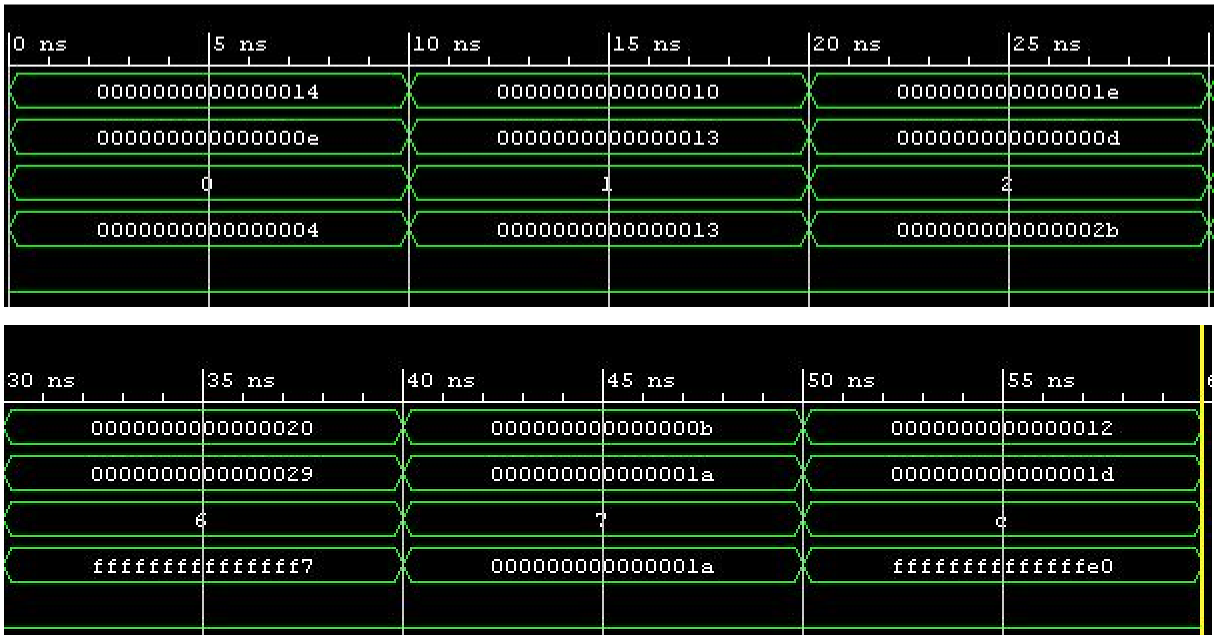
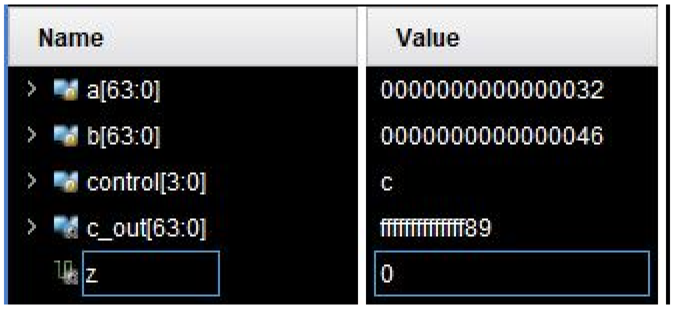


Figure 2: Trial 2 Waveform generation (more complex input values)

<u>Test Case 3:</u>

Operation	Input A (decimal)	Input B (decimal)
AND	0	0
OR	1	8
add	4	8
subtract	5	10
Pass input b	0	6
NOR	50	70



Name	Value
a[63:0]	0000000000000032
b[63:0]	0000000000000046
control[3:0]	c
c_out[63:0]	ffffffffffffff89
z	0

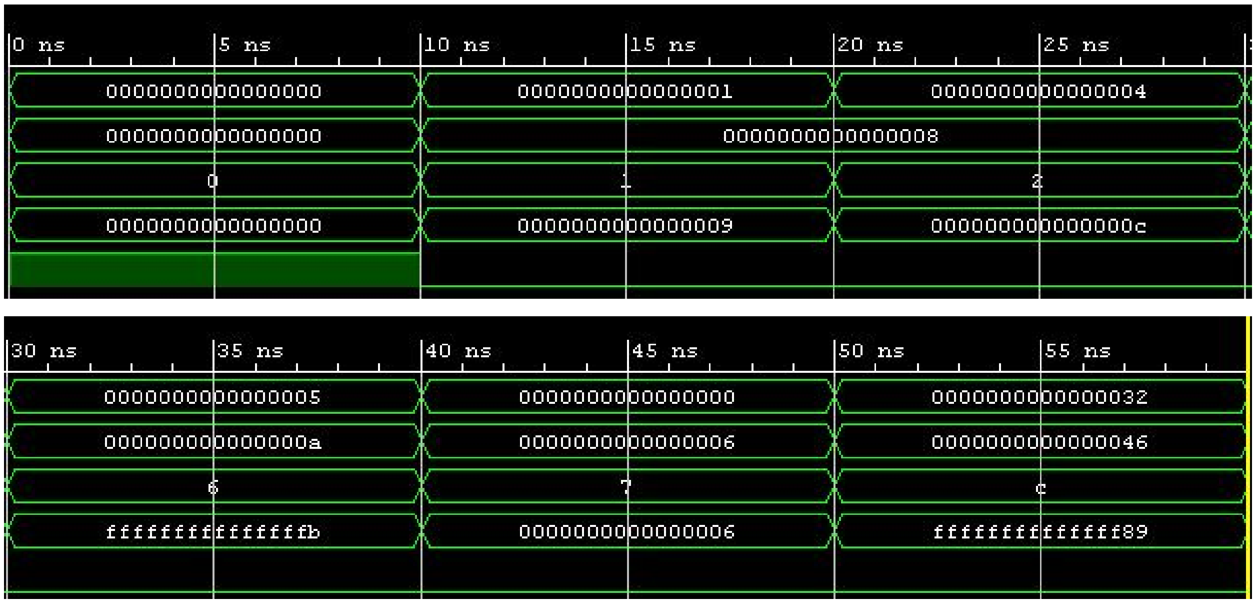


Figure 3: Trial 3 Waveform generation (active high when result is zero)

<u>Discussion:</u>

Using experience from the previous labs, students implemented the ALU design. The primary module for the ALU system consisted of test cases depending on the opcode that resulted in various different arithmetic and logical operations conducted on two input values. There were two main outputs, the normal output used for all of the operations and the zero output which was only ever activated if the normal output was equal to zero. The testbench then provides initial values for each test case and went through each instruction with different inputs and a delay to allow for viewing of individual instructions.

<u>Conclusion:</u>

The lab worked as expected. The ALU produced the correct outputs in all of the cases. This lab also helped develop skills in using the Xilinx Verilog software and also provided a vital piece of a later lab in this course, the full ARM processor.