Lab 7: ALU and ALU Control

Jiasen Zhou and Jon Johnston

March 6, 2019

1 Executive Summary

The purpose of this lab is to design and simulate an ALU and an ALU Control. The ALU Control interprets the opcode and ALUOp signals, using them to generate a four-bit control singal that is sent to the ALU. The ALU itself does arithmetic and logic operations for the computer based on the control signal it receives. After running the test benches for the two modules, they generated the expected outputs, so the lab was successful.

2 Test Report

Verifying the operation of these modules required the use of two test benches, one for each of the modules.

1. ALU Control Test
2. ALU Test

Figure 1: Expected Results of the register test.

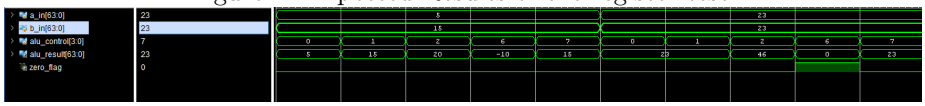
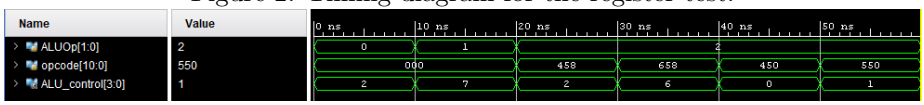


Figure 2: Timing diagram for the register test.



3 Code Appendix

Listing 1: Verilog code for testing a register.

```
`include "definitions.vh"
module ALU(
            input [`WORD-1:0] a_in,
            input [`WORD-1:0] b_in,
            input [3:0] alu_control,
            output reg [`WORD-1:0] alu_result,
            output reg zero_flag);

        always @(*)
         begin
             zero_flag <= (alu_result == 0);

             casex(alu_control)
                 `ALU_AND: begin
                     alu_result <= a_in & b_in;
                           end
                 `ALU_OR: begin
                     alu_result <= a_in | b_in;
                           end
                 `ALU_ADD: begin
                     alu_result <= a_in + b_in;
                           end
                 `ALU_SUB: begin
                     alu_result <= a_in - b_in;
                           end
                 `ALU_PASS: begin
                     alu_result <= b_in;
```

```
                    end
                endcase
            end

endmodule
```

Listing 2: Verilog code for implementing a register.

```
`include "definitions.vh"

module register(
    input wire clk,
    input wire reset,
    input  wire [`WORD-1:0] D,
    output reg [`WORD-1:0] Q=`WORD'b0
    );

    always @(posedge(clk),posedge(reset))begin
        if (reset==1'b1)
            Q<=`WORD'b0;
        else
            Q <= D;
    end

endmodule
```