Lab 2

Josh Young

February 6, 2017

1 Introduction

In Lab 2, the user is tasked with testing a MUX and an ADDER by creating a test bench and running a simulation with the MUX and ADDER by feeding it data and observing the output. Vivado was used to create the test bench as well as run the simulation for both the MUX and the ADDER.

2 Interface

The inputs for the ADDER are an indentation (i.e. 4) and the signal from the program counter. The output for the ADDER is used to get the next address by using the program counter. The MUX takes the output from the ADDER and feeds an input to the ADDER in order to get an address. The data that these two devices use are in binary.

3 Design

Specifically, the MUX and ADDER are used together in order to fetch addresses for instruction memory. There is a kind of feedback loop that connects the MUX and ADDER, where one will update the other.

4 Implementation

The code used for creating the MUX can be seen in Listing 1 on page 1. The code for creating the ADDER can be seen in Listing 2 on page 2.

Listing 1: Verilog code for implementing a MUX.

```
`include "definitions.vh"

module mux#(
    parameter SIZE=5)(
    input [SIZE-1:0] Ain,
    input [SIZE-1:0] Bin,
    input control,
    output [SIZE-1:0] mux_out
    );
    assign mux_out = control?Bin:Ain;
endmodule
```

Listing 2: Verilog code for implementing an ADDER.

```
`include "definitions.vh"

module adder(
    input [`WORD-1:0] Ain,
    input [`WORD-1:0] Bin,
    output [`WORD-1:0] add_out
    );
    assign add_out = Ain+Bin;
endmodule
```

5 Test Bench Design

The lab had the user create two test benches, one for the MUX, and the other for the ADDER. The goal for the test bench was to try and 'break' the code for the MUX or ADDER. For the ADDER, this was tried by feeding several different numbers to add, I used small and large numbers in decimal where carries would be required. Expected errors would be carry issues, and a potential for overflow. As for the MUX, the test bench was created so that the user could observe how the MUX would operate when feeding inputs, i.e. checking the output change while switching the control wire. The verilog code for the ADDER test bench can be viewed in Listing 4 on page 3, and the code for the MUX test may be viewed in Listing 4 on page 3.

Listing 3: Verilog code for testing an adder.

```
`timescale 1ns / 1ps
`include "definitions.vh"

module adder_test;

        // Inputs

        reg [`WORD-1:0] Ain;
        reg [`WORD-1:0] Bin;

        // Outputs
        wire [`WORD-1:0] add_out;
        // Instantiate the Unit Under Test (UUT)
        adder uut (
                .Ain(Ain),
                .Bin(Bin),
                .add_out(add_out)
        );

        initial begin

        Ain = 2;
        Bin = 10;
        #50;
        Ain = 5;
        Bin = 10;
        #50;
        Ain = 89;
        Bin = 120;
        #50;
        Ain = 24567;
        Bin = 4510;
        #50;
        end

endmodule
```

Listing 4: Verilog code for testing a mux.

```
`timescale 1ns / 1ps

module mux_test;

        reg [15:0] Ain;
        reg [15:0] Bin;
        reg control;
        wire [15:0] mux_out;

        mux#(16) uut (
                .Ain(Ain),
                .Bin(Bin),
                .control(control),
                .mux_out(mux_out)
        );

        initial begin

        Ain <= 0;
        Bin <= 12;
        control <= 1;
        #10;
        Ain <= 23;
        Bin <= 12;
        control <= 1;
        #10;
        Ain <= 0;
        Bin <= 1222;
        control <= 1;
        #10;
        Ain <= 0234;
        Bin <= 312;
        control <= 0;
        #10;
        Ain <= 1231;
        Bin <= 12;
        control <= 1;
        #10;
        Ain <= 0;
        Bin <= 102;
        control <= 0;
        #10;

        end

endmodule
```

6 Simulation

By viewing the timing diagrams for the MUX and the ADDER, one can see that both behave as expected. The timing diagram for the adder can be viewed in Figure 1. The timing diagram for the MUX may be viewed in Figure 2.

7 Conclusions

The goal of this lab was to create test benches for the MUX and ADDER. The purpose of testing these items is to ensure that the code that creates them wont throw errors. Overall, the MUX and ADDER behaved as expected

Figure 1: Timing diagram for the adder test.

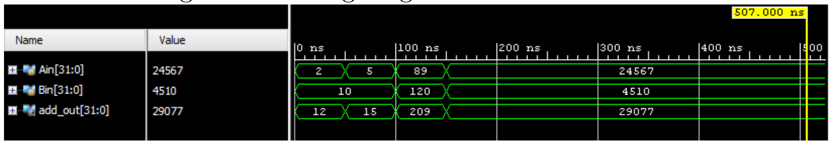
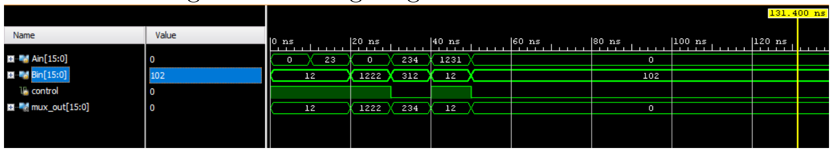


Figure 2: Timing diagram for the mux test.


131.400 ns
Name
Value
Ain[15:0]
0
Bin[15:0]
102
control
0
mux_out[15:0]
0
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
0
23
0
234
1231
0
12
1222
312
12
102
12
1222
234
12
0