Lab 6

Josh Young

March 15, 2017

1 Introduction

In Lab 6, the user is tasked with finishing and testing a regfile for the MIPS machine by creating a test bench and running a simulation. The results of the simulation were observed in order to be sure that the regfile is reading and writing properly. Vivado was used to create the test bench as well as run the simulation for the regfile.

2 Interface

The Register file has several inputs, these consist of 2 registers, a write register, write control, data to write, a clock, and a reset. There are only a couple of outputs for the register file though, which is just reading data from the registers.

3 Design

The register file is an invaluable part of the decoding stage for a MIPS machine. The register file is used as a channel between the decode and write back stages. It is necessary though to use a buffer with the register file, however that isn't addressed in lab 6.

4 Implementation

The code used for creating the regfile can be seen in Listing 1 on page 1.

Listing 1: Verilog code for implementing a regfile.

```
`include "definitions.vh"

module regfile#(
    parameter SIZE=32, bits=`CLOG2(SIZE))(
    input clk,
    input reset,
    input write,
    input [(bits - 1):0] address_A,
    input [(bits - 1):0] address_B,
    input [(bits - 1):0] address_dest,
    input [`WORD - 1:0] write_data,
    output reg [`WORD - 1:0] A=`WORD'b0,
    output reg [`WORD - 1:0] B=`WORD'b0
    );

    reg[`WORD - 1:0] rf [SIZE-1:0];
    //integer i;//uncomment for reset code

    // handle input
    always @(negedge(clk))begin
        //reset is commented out to allow data
        //loading at start, didn't delete FYI
        //if (reset)
        //      for(i=0;i<SIZE;i=i+1)
        //          rf[i]<=0;
        //else

        //your code

        if (write) begin
        rf[address_dest] <= write_data;
        end


    end

    //handle output
    always @(posedge(clk))begin
        //your code 2 lines
        B <= rf[address_B];
        A <= rf[address_A];
    end

    initial
        $readmemh(`RMEMFILE, rf);

endmodule
```

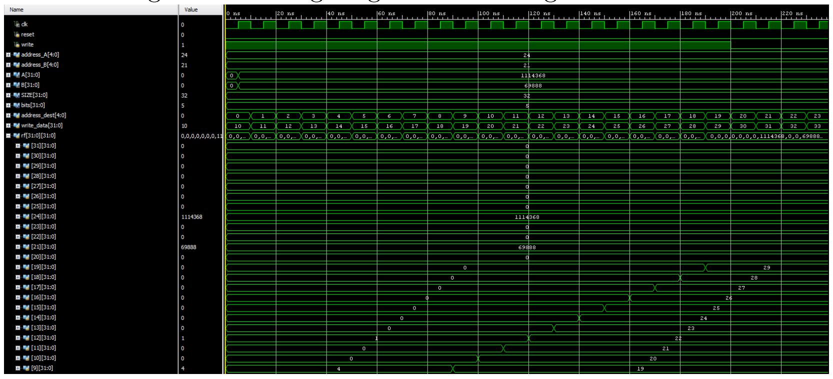
5 Test Bench Design

The lab had the user create a test bench for the regfile. The goal for the test bench was to try and 'break' the code for the regfile. This test was done by setting the addresses for register A and B, and then changing the writing address as well as the data. Then, the outputs were monitored to be sure that the regfile behaved correctly. The verilog code for the regfile test bench can be viewed in Listing 2 on page 3.

Listing 2: Verilog code for testing the regfile.

```
`include "definitions.vh"

module regfile#(
    parameter SIZE=32, bits=`CLOG2(SIZE))(
    input clk,
    input reset,
    input write,
    input [(bits - 1):0] address_A,
    input [(bits - 1):0] address_B,
    input [(bits - 1):0] address_dest,
    input [`WORD - 1:0] write_data,
    output reg [`WORD - 1:0] A=`WORD'b0,
    output reg [`WORD - 1:0] B=`WORD'b0
    );

    reg[`WORD - 1:0] rf [SIZE-1:0];
    //integer i;//uncomment for reset code

    // handle input
    always @(negedge(clk))begin
        //reset is commented out to allow data
        //loading at start, didn't delete FYI
        //if (reset)
        //      for(i=0;i<SIZE;i=i+1)
        //          rf[i]<=0;
        //else

        //your code

        if (write) begin
        rf[address_dest] <= write_data;
        end


    end
```

Figure 1: Timing diagram for the register file test.



Name	Value
clk	0
reset	0
write	1
address_A[4:0]	24
address_B[4:0]	21
A[31:0]	0
B[31:0]	0
SIZE[31:0]	32
bits[31:0]	5
address_dest[4:0]	0
write_data[31:0]	10
rf[31:0][31:0]	0,0,0,0,0,0,0,11
[31][31:0]	0
[30][31:0]	0
[29][31:0]	0
[28][31:0]	0
[27][31:0]	0
[26][31:0]	0
[25][31:0]	0
[24][31:0]	1114368
[23][31:0]	0
[22][31:0]	0
[21][31:0]	69888
[20][31:0]	0
[19][31:0]	0
[18][31:0]	0
[17][31:0]	0
[16][31:0]	0
[15][31:0]	0
[14][31:0]	0
[13][31:0]	0
[12][31:0]	1
[11][31:0]	0
[10][31:0]	0
[9][31:0]	4

```
    //handle output
    always @(posedge(clk)) begin
        //your code 2 lines
        B <= rf[address_B];
        A <= rf[address_A];
    end

    initial
        $readmemh(`RMEMFILE, rf);

endmodule
```

6 Simulation

By viewing the timing diagrams for the register file test, one can see that the register file behaved as expected. The timing diagram for the register file can be viewed in Figure 1.

7 Conclusions

The goal for lab 6 was to finish the register file and create a test bench for it. The simulation shows that the register file acted as it should have. The register file should be able to read from registers and write to registers based off of a given control command.