Lab 10: Memory

Jiasen Zhou, Jon Johnston

April 1, 2019

1 Executive Summary

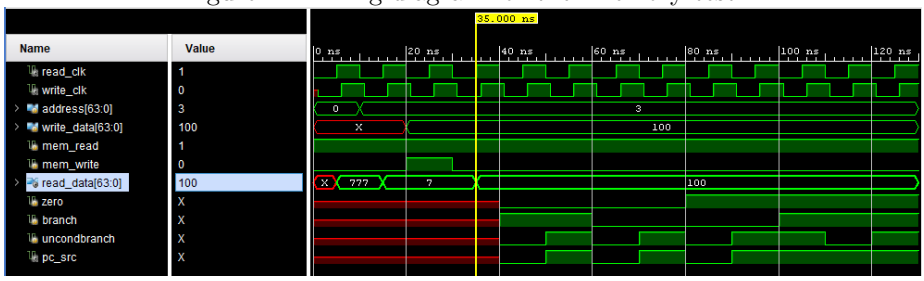
The purpose of this lab is to design and simulate the Memory stage of the datapath with the iMemory module. The primary functionality of this stage is to manage the data memory used in the load and store commands. However, the branch resolution logic gates are also a part of this stage, and they handle what values are passed to the pc_src for conditional and unconditional branching. After running the testbench simulation, it is clear that the lab was successful.

2 Test Report

To verify operation of these module, this lab requires one test bench.

1. iMemory Test Bench

Figure 1: Timing diagram for the iMemory test.



3 Code Appendix

Listing 1: Verilog code for iMemory module.

```
`include "definitions.vh"
module iMemory(
                input read_clk, write_clk,
                input uncondbranch,
                input branch,
                input mem_read,
                input mem_write,
                input zero,
                input [`WORD-1:0] address,
                input [`WORD-1:0] write_data,
                output pc_src,
                output [`WORD-1:0] read_data);

                wire and_zero;

                assign and_zero = (branch & zero);
                assign pc_src = (uncondbranch | and_zero);

          data_mem DM(.read_clk(read_clk),
                      .write_clk(write_clk),
                      .mem_write(mem_write),
                      .mem_read(mem_read),
                      .write_data(write_data),
                      .address(address),
                      .read_data(read_data));

endmodule
```

Listing 2: Verilog code for data_mem component.

```
`include "definitions.vh"

module data_mem(
    input read_clk, write_clk, mem_write, mem_read,
    input [`WORD-1:0] write_data, address,
    output reg [`WORD-1:0] read_data);

    reg [`WORD-1:0] ramf [31:0];

    always @(posedge read_clk)
    begin
        if(mem_read)
            read_data <= ramf[address];
        else
            read_data <= 64'dZ;
    end

    always @(posedge write_clk)
    begin
        if(mem_write)
           ramf[address] <= write_data;
    end

    initial
        $readmemb(`DMEMFILE, ramf);
endmodule
```

Listing 3: Verilog code for testing the iMemory module .

```
`include "definitions.vh"
module iMemory_test;

    wire read_clk;
    wire write_clk;
    reg [`WORD - 1:0] address;
    reg [`WORD - 1:0] write_data;
    reg mem_read;
    reg mem_write;
    reg zero;
    reg branch;
    reg uncondbranch;
    wire [`WORD - 1:0] read_data;
    wire pc_src;

    oscillator clk_gen(read_clk);
```

```
delay delay_one(
        .a(read_clk),
        .a_delayed(write_clk)
        );

iMemory IM(
        .read_clk(read_clk),
        .write_clk(write_clk),
        .address(address),
        .write_data(write_data),
        .mem_read(mem_read),
        .mem_write(mem_write),
        .zero(zero),
        .branch(branch),
        .uncondbranch(uncondbranch),
        .read_data(read_data),
        .pc_src(pc_src));

    initial
        begin
        //memread memwrite


        //read data from address
        mem_write = 0;
        mem_read = 1;
        address <= 'WORD'd0; #10;
        address <= 'WORD'd3; #10;

        //write data
        mem_write = 1;
        address <= 'WORD'd3;
        write_data <= 'WORD'd100; #10;

        //read again to check
        mem_write = 0;
        mem_read = 1;
        address <= 'WORD'd3; #10;

        //branching test
        branch = 1;
        zero = 0;
        uncondbranch = 0; #10;
        uncondbranch = 1; #10;
```

```
            branch = 0;
            zero = 0;
            uncondbranch = 0; #10;
            uncondbranch = 1; #10;

            branch = 0;
            zero = 1;
            uncondbranch = 0; #10;
            uncondbranch = 1; #10;


            branch = 1;
            zero = 1; #10;
            uncondbranch = 0; #10;
            uncondbranch = 1; #10;



            $finish;
        end
endmodule
```