PHYS 234 Lab 8: Memory

Ethan Yang

This lab attempts to design and test the read and write function of the RAM. The first module is a classic memory circuit that reads and writes a 5 bit number. The second module implements a memory with one write port and two read ports, meaning you can read two different values simultaneously through two different registers.

For the first Verilog code snippet, “we” represents the write enable, and only writes values into the memory whenever it’s high. A memory array with length of the amount of addresses was created to store our values. The clock block implements the write function, where the input value is written into the memory, into the address index, at the clock edge. The assignment line implements the read function by calling a specific index of the memory array.

```
`timescale 1ns / 1ps

module eelab8 #(parameter n = 3, m = 5)(
      input clk,
      input we,
      input [n-1:0] adr,
      input [m-1:0] din,
      output [m-1:0] dout
      );

reg [m-1:0] mem [2**n-1:0];

always @ (posedge clk)
      if (we) mem[adr] <= din;

assign dout = mem[adr];

endmodule
```

We allocated 3 bits for the address and 5 bits for the input value, and the memory successfully saves and retrieves the 5 bit values based on the address input. For example, if we input “adr = 010” and “din = 111”, the memory location 010 would store the value 5. There are a total of 8 values that we could write into the circuit because there are only 2^3 = 8 memory locations. Increasing the number of bits in the address can increase the number of memory locations the circuit can have.

The second code snippet works fundamentally the same as the first, with additional inputs to implement the extra read port. Here, “we3” is the write enable, “ra” 1 and 2 are the read addresses, and “wa3” and “wd3” are the write address and write data respectively. “Rd” 1 and 2 are read data.

Similarly, the write function was implemented in the clock edge block, where our write data is stored into the write address index of the memory. The read function is implemented in the assignments, where our data are assigned to our register ports if the read addresses are non-zero.

```
`timescale 1ns / 1ps

module eelab8(
      input clk,
      input we3,
      input [1:0] ra1, ra2, wa3,
      input [1:0] wd3,
      output [1:0] rd1, rd2
      );

reg [31:0] rf[31:0];

always @ (posedge clk)
      if (we3) rf[wa3] <= wd3;

assign rd1 = (ra1 != 0) ? rf[ra1] : 0;
assign rd2 = (ra2 != 0) ? rf[ra2] : 0;

endmodule
```

Simply put, there is an array of data with 3 addresses because reading the index 0 always returns a 0, and the registers can read 2 addresses at a time. For example, if I write the value 01 to address 01, 10 to address 10, and so on, we can choose 2 values to display through the read registers at the same time. If I set the read address on the first port to 01 and the second port to 11, I will see the LED displays the values 01 and 11. The LED always shows 00 when the read address is 00.