Tyler Swenson

EE457

Lab 9 Report

4/10/2025

EE457 Memory Blocks

# Introduction

In this lab, we will be implementing RAM modules for use in memory blocks as an exercise for general issues around implementing memory. We will be implementing RAM modules that utilize a five-bit address port, a four-bit data port, and a write control port. These, along with the read data from the address will be displayed on the HEX displays to be able to ensure valid usage.

# Theory of Operation

## Requirements

1. The design will reset when the reset signal is asserted low.
2. Switches will be used to input data for address, write enable, and data.
   1. Part a will utilize:
      1. Switches 8-4 will specify the write and read address.
      2. Switches 3-0 will provide data for write.
      3. Switch 9 will use a rising edge detector as the write signal.
   2. Part b will utilize:
      1. Switches 8-4 will specify the address.
      2. Switches 3-0 will provide data for write.
      3. Switch 9 will use a rising edge detector as the write signal.
      4. Instead of switches providing the read address, a counter will be used to count through every address.
3. HEX displays will show values from each part:
   1. Part a will use:
      1. HEX 5-4 will show the address value.
      2. HEX 2 will show the data being input into memory.
      3. HEX 0 will show the data read out from memory.
   2. Part b will use:
      1. HEX 5-4 will show the write address.
      2. HEX 3-2 will show the read address.
      3. HEX 1 will show the write data.
      4. HEX 0 will show the read data.
4. Part b will use a memory initialization file (MIF).
5. All inputs and resets will be synchronized.

## Description of the Design

A diagram of a computer program

AI-generated content may be incorrect.

Figure 1: RAM Block Diagram

### Part a

This part will utilize a 1-port RAM that will take in switches 8-4 for both the read and the write address. Switches 3-0 will then be used for the data for the write. The write enabled will then only occur when on a rising edge from switch 9. The output from the RAM will be constantly read. The hex displays will then be used to display the current address, the data read, and the write data from the switches.

### Part b

This part will utilize a 2-port RAM that will take two separate read and write addresses, respectively. Switches 8-4 will provide the write address, while a counter will be used for the read address to cycle through the addresses. Again, a rising edge from switch 9 will be used for the write enable. Similar to part a, the hex displays will be used to display the write address, read address, the data in the write port, and the data being read. In addition, this part will use a memory initialization file (MIF) to initialize the memory blocks to predefined values.

# Verification

## Test Plan

### Part a

To test the design, a testbench will test writing data to some addresses to view the output signal resulting after the write switch is turned from on to off. To ensure that only data is written when the switch is changed from on to off, data will be written, and then the data that is being provided will be changed, while the switch stays in the on position, and then will be changed from on to off, to ensure this doesn’t trigger a write.

Another address will be written to, then proceeding with this, the previous address that was written to will be checked to ensure the previous data at the address is still valid.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 2: Code snippet for writing 0111 to address 1

A grey background with white text

AI-generated content may be incorrect.

Figure 3: Code snippet for changing data while wren is asserted to 1

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 4: Code snippet for writing 0100 to address 16

A screen shot of a computer code

AI-generated content may be incorrect.

Figure 5: Checking the previous addresses still contain their set values

### Part b

A computer screen shot of a program

AI-generated content may be incorrect.

Figure 6: Checking the mif file addresses and testing writes

## Test Bench

### Part a

A screen shot of a computer

AI-generated content may be incorrect.

Figure 7: Writing 1110 to address 1 and writing. Changing data to 1100 to see if the data holds.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 8: Writing 0100 to address 16

A screen shot of a computer

AI-generated content may be incorrect.

Figure 9: Ensuring the data is still in the addresses after the switch

### Part b



Figure 10: Reading through the first 16 addresses to ensure mif data.

A close up of numbers

AI-generated content may be incorrect.

Figure 11: Checking the written addresses for the change on read

# Conclusion

In this lab, I learned how to utilize both a 1-port ram and 2-port ram. The process of using the wizard was fairly simple to get through if you know what you need to use in memory. I wanted to take the extra challenge and infer RAM as pointed out in the instructions and wanted to make it generic so that it could be used from any different address or data bit width. This was a little more complicated, and I ran into issues where synthesis would optimize the design out so it would not infer the RAM like desired. This makes sense to where the design wouldn’t want to allocate RAM if it doesn’t need to. Though, after that it was satisfying to have the RAM work as expected similar to the wizard.