Name: ______________________________ Score: ____________ / 100

Laboratory #6 Memory (1 weeks)

EE 310 Fundamentals of Computer Engineering
School of Informatics, Computing, and Cyber Systems
Northern Arizona University
Spring 2017

Objective

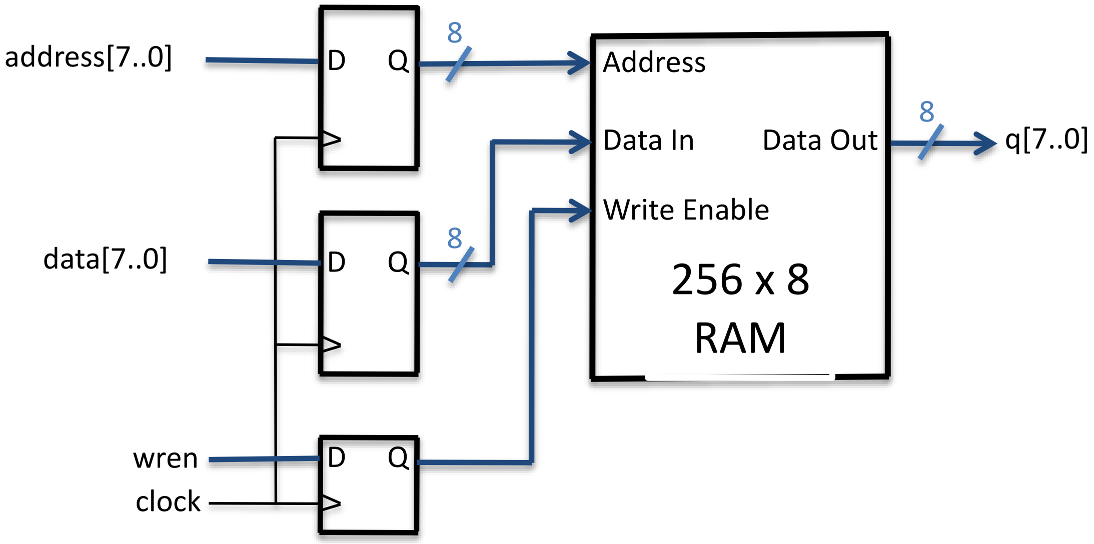
At the completion of this lab, the student will be able to implement a memory module on the Cyclone V GX FPGA, write initial data into the memory, write data to the memory using the input signals, and read data from the memory.

Important Concepts

1. The Cyclone V family has two ways of implementing memory: using the flip-flop in each Logic Element (not very efficient), or M10K blocks. The M10K blocks each have 10 x 2^{10} = 10240 bits and can be configured in various ways. A common term used to specify the size of a memory is its aspect ratio, which gives the depth in words and the width in bits (depth x width). Some aspect ratios supported by the M10K block are 8K x 1, 4K x 2, 2K x 4, 1k x 8 and 512 x 16. We will utilize the **256 x 8** mode in this exercise, which will allow plenty of room for testing our processor. Many other modes of operation can be supported with M10K blocks. You can learn about them from reading the Altera Cyclone V FPGA documentation.

2. A block diagram of our random access memory (RAM) is shown below. It contains 256 bytes that are accessed using an eight-bit **address** port, an eight-bit **data** port for input data, and a **wren** control input for write enable. The data output, called **q**, is not registered. Since this RAM has one address port, it is called a one-port memory. (Note that a two-port memory would provide two address ports for two simultaneous operations.)

   Let's look at the RAM's behavior. If the **Write Enable** signal to the RAM is high, the **Data In** is written to the memory location specified by **Address**. If **Write Enable** is low, the **Data Out** provides the information that is read from the memory location.



3. Quartus includes a Qsys IP catalog tool to create VHDL for common subsystems. These are called IP Core Modules.

Activity #1 Implement an IP Core Memory

Click on **Tools > IP Catalog**. A panel will open on the *right side* of the Quartus window. Go to **Library > Basic Functions > On-Chip Memory > RAM 1-PORT** and double click. A dialog will open, **Save IP Variation**.

1. Give the name **ramlpm**, choose **VHDL**, and click **Next**.
2. Another Dialog box will now open **MegaWizard Plug-in Manager** page 1. Choose an aspect ratio of **256 x 8**, choose **M10K** in the list, choose ***single clock*** and click on **Next**.
3. MegaWizard page 2: *Uncheck* the **'q' output port**. We will not be using a registered q output bus for this lab. Click on **Next**.
4. MegaWizard page 3: Choose the output to be **New Data**. Click on **Next**.
5. MegaWizard page 4: Leave the initial content of the memory blank, click on **Next**.
6. MegaWizard page 5: Click on **Next**.
7. MegaWizard page 6: Make sure the files with the following extensions are checked: *.cmp*, *.bsf*, *.vhd*. Click on **Finish**.
8. Press **Yes** for the **Quartus Prime IP Files** dialog box.

The entity and architecture for the memory is in the file **ramlpm.vhd**. Open up the VHDL file (as TEXT) and take a look at what was just created for you. What are the input and output signals and how many bits are they?

Inputs:	Outputs:
address (8 bits) clock (1 bit) data (8 bits) wren (1 bit)	q (8 bits)

What is the value of the generic parameter **operation_mode**? SINGLE_PORT

What is the value of **outdata_reg_a**? UNREGISTERED

What is the aspect ratio of this memory? 256 x 8

Create a simulation test bench and a force file to test the first **16** locations of this memory design. Display all of the input and output signals of the memory. First, count through the first 16 addresses (0x00 to 0x0F) and display the default memory values. Next, write the new values to each the first 16 memory addresses as follows. The upper nibble (first hex digit) should be the same as the address. The lower nibble (second hex digit) should be the 1's complement of that. For example, location 0x00 should contain 0x0F, 0x01 should contain 0x1E, and so on. Then read every memory address to verify the data was written correctly. *Use the block that was generated by the software!*

Submission checklist:

- VHDL code for the ram
- Force file
- Questions in this lab instruction
- Functional simulation results

Activity #2 Memory Initialization

Quartus II provides a mechanism for initializing memory. This is useful for getting input data into RAM or initializing a RAM or ROM with a program.

Two steps are required: (1) creating the memory initialization file, known as a .mif file; and (2) modifying your memory VHDL architecture to specify the memory initialization file and including it in the project.

For step one, create a new Memory Initialization File by selecting **File > New > Memory Files > Memory Initialization File**. For at least the first 16 locations, specify unique data at each address, but be careful the data you should input in each box should be unsigned decimal number.

Save this memory initialization file under the filename **ramlpminit.mif** .

For step two, rerun the MegaWizard as you did in Activity #1 and create a new RAM, called **ramlpminit.vhd.** Use the same parameters as you did before, but this time specify **ramlpminit.mif** as the memory initialization file. Creating a force file to run the functional simulation and check the initial values of the memory in the first round of the memory addresses.

Submission checklist:

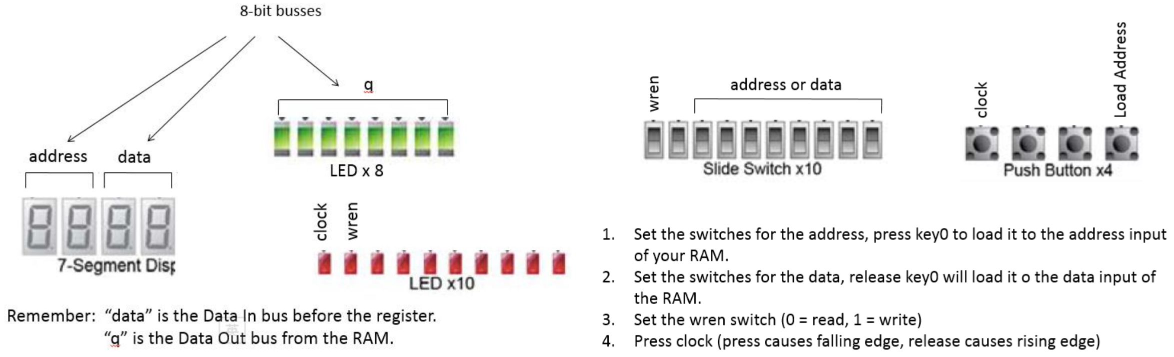
- VHDL code for the ram
- Force file
- ramlpminit.mif
- Functional simulation results

Activity #3 Demo

Create a demo test bench to show initialization, reading, and writing to the memory. Use the test strategy shown below. Program the Cyclone V starter board and demonstrate the memory functionality.

Instructor's checklist:

- Correct operation on the Cyclone V FPGA board for initialization, reading and writing.



Remember: "data" is the Data In bus before the register.
"g" is the Data Out bus from the RAM.

1. Set the switches for the address, press key0 to load it to the address input of your RAM.
2. Set the switches for the data, release key0 will load it o the data input of the RAM.
3. Set the wren switch (0 = read, 1 = write)
4. Press clock (press causes falling edge, release causes rising edge)

Activity #4 Instructor questions (in-class quiz) --- 30pts

To show that each of you are actively involved in the projects, a quiz will be given at the beginning of lab7. Questions will be closely related to the lab tutorial materials and lab practice.