COE 608 Lab 4a – Data Memory Module

Due Date: Week 6

1. Objectives

The purpose of this lab is to implement and simulate a simple memory unit which is capable of reading and writing data within a single clock cycle. The symbol for the data memory unit is illustrated in Figure 1.

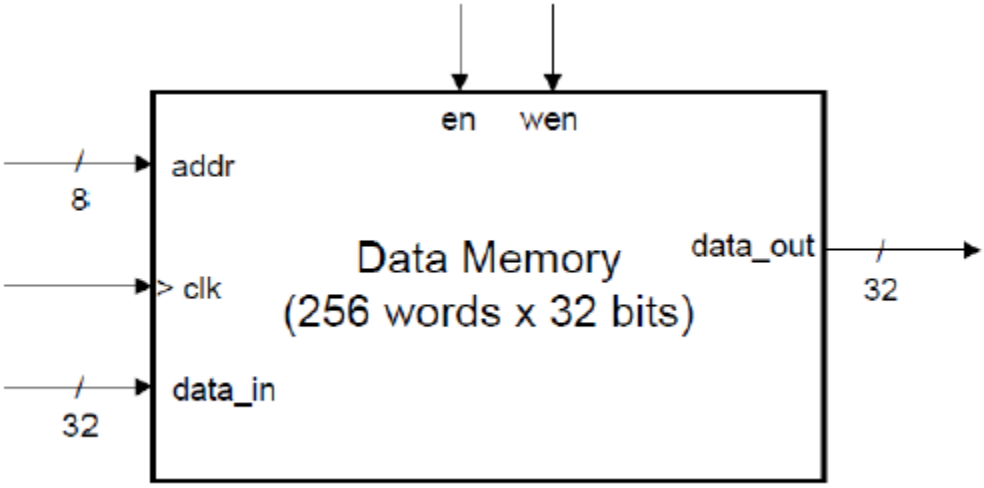

Figure 1: 256-word x 32-bit Data Memory Unit

The memory unit has an 8-bit address input (addr), 32-bit data input and output (data_in and data_out), and other control lines such as clock, write enable (wen), and enable (en). There are many ways to implement a memory module, however memories are typically implemented as a 2D array of registers. The data memory unit designed in this lab will also be simulated on the Cyclone IV device.

COE608's final project requires you to design a semi-RISC processor which is capable of executing various instructions. You will use the memory module designed in this lab to retrieve and store data for the CPU (i.e. data memory). We will be implementing a Harvard based architecture, meaning that two separate memories will be needed - one for reading and writing data, and the other for instruction handling. We will explore an alternate approach to create the instruction memory later in the course.

2. Data Memory Unit Operations

For each clock cycle, the "en" and "wen" input bits will enable the read and write functionality of the memory module for a given input address. These functionalities are described in Table I. Note that the data memory will function on the falling edge of the input clock.

Table I: Data Memory Unit Functions

en	wen	Function	data_out
0	X	N/A	0
1	0	Read	M[addr]
1	1	Write M[addr] <= data_in	0

During a read request (wen = '0', en = '1'), the memory unit will output the contents stored at the inputted address, triggered by the clock's falling edge (after a certain propagation delay). If a read is requested on the next cycle, the unit will be able to output the new contents since it has a single-cycle access latency. In the case of a write (wen = '1', en = '1'), the "wen", "en", "addr" and "data_in" signals must be stable for the falling clock edge in order to guarantee that the contents are successfully written to the memory. This means that all the data and control lines must be setup prior to the falling edge of the clock (a stable t_{setup} for this component will be approximately 20ns).

3. Memory Module Design

Students are required to implement a data memory module in VHDL with the following entity declaration:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY data_mem IS
PORT(
        clk      : IN STD_LOGIC;
        addr     : IN UNSIGNED(7 DOWNTO 0);
        data_in  : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
        wen      : IN STD_LOGIC;
        en       : IN STD_LOGIC;
        data_out : OUT STD_LOGIC_VECTOR(31 DOWNTO 0));
END data_mem;

ARCHITECTURE Description OF data_mem IS
        -- define array type and signal to store data

BEGIN
        --Make it work!
END Description;
```

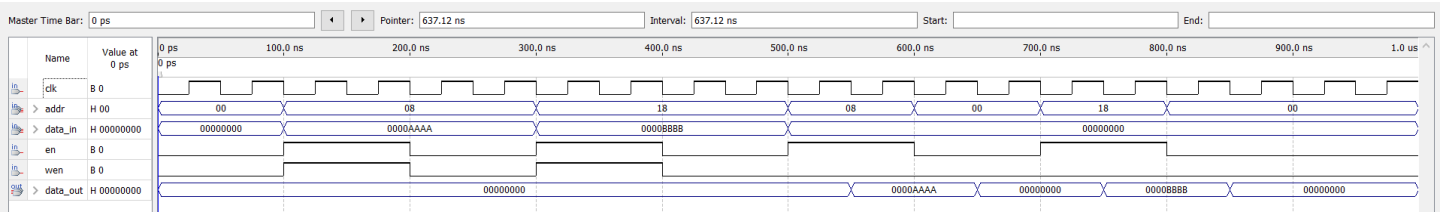

Figure 2: Sample Functional Waveform Simulation of Memory Module

Figure 2 presents a sample functional waveform simulation for the memory module, including reads, writes, and no operation as described in Table I. Regarding the actual VHDL implementation, students may want to research and refer to the following concepts in order to implement the memory module:

- 'type' and 'array'
- SIGNAL implementations for array types
- to_integer() function

4. What to Hand In

To receive full credit for the single-cycle memory unit design, students must submit the following:

- A hard copy listing of your VHDL source code implementation.
- A hard copy printout of the waveform testing file containing testing that you feel is required to demonstrate the correct implementation of your memory module.
- You must also submit a short report (half page) describing the timing characteristics of your memory unit, meaning worst-case delays for reads and writes of various inputs. Students need to perform both functional and timing simulations.

Your lab instructor may also quiz you at the time of this demo.