Experiment 5

Using Random Access Memory with 8051 Microcontroller

**INTRODUCTION:**

In this lab, you will be provided with a Random Access Memory (RAM) module in the form of an EDIF file. The given RAM is double ported i.e. it has separate Read and Write ports, just like in the 8051 Module that we use. You are required to interface this RAM with the 8051 microcontroller. Then you will be given some address locations and their corresponding contents to be written to the RAM. You will write a C program for the microcontroller to execute the above Write operation, and a routine to read those locations back to display the contents on seven-segment displays available on DE-2 FPGA board.

**OBJECTIVES:**

1. Familiarization with Random Access Memories
2. Understanding Read and Write Cycles of RAM

**MATERIALS REQUIRED:**

- Altera DE-2 FPGA board
- Windows-based Computer with an unused USB port
- FTP program
- Keil Microvision software suite
- Altera Quartus-II software suite

**BACKGROUND:**

Random Access Memories are volatile memories used in almost all the modern electronic devices on account of their ever-increasing storage capacity, ease of use, and speed of operation. They are called “Random Access” memories, because any address of a RAM can be accessed randomly at any time and can be read/written any number of times. There are two types of RAMs, viz. Static RAM (SRAM) and Dynamic RAM (DRAM). SRAM is extremely fast and dissipates less power but requires a high silicon area and is very expensive. It exclusively used as Cache Memory in Microprocessors. DRAM, on the other hand, is cheap, area efficient but is slow and requires a frequent refreshing. It is used as the main memory in modern personal computers. The choice between the use of SRAM and DRAM is made on the basis of cost, power and area requirements of a design. In this lab, you will learn the basic interface between a microprocessor and a RAM. You will also learn about the Read and Write cycles of a RAM. The RAM module will be provided to you by the instructor that has an 8-bit address bus and an 8-bit data bus. The read and write cycles are illustrated in Figure 1 and Figure 2.
**PROCEDURE:**

<table>
<thead>
<tr>
<th>Data</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFF</td>
<td>0x00</td>
</tr>
<tr>
<td>0xAB</td>
<td>0x04</td>
</tr>
<tr>
<td>0x05</td>
<td>0x0A</td>
</tr>
<tr>
<td>0x7D</td>
<td>0x0C</td>
</tr>
</tbody>
</table>

Table 1: Data stored in the corresponding memory locations of a RAM

1. Table 1 shows the various address locations of a RAM where the data is to be written and then it has to be read from those locations. For writing and then reading from the RAM you have to first write a C code in the Keil Microvision software.
2. While writing the C code, make sure that you use port 0 of the micro controller as the RAM data port, port 1 as the seven segment display output, port 2 as the RAM address port and port 3 as the control signals such as OE (Output Enable) and WE (Write Enable) pins. Both are active low.
3. The sequence for writing into the RAM is as follows
   a. Set OE’ and WE’ high to ensure no reading or writing is being done.
   b. Send the address via Port 2.
   c. Send the data to be written via Port 0.
   d. Set WE’ low to enable writing to the RAM.
   e. Set WE’ high to latch the data in RAM and disable writing.
4. Similarly the read sequence is as follows
   a. Set OE’ and WE’ to 1.
   b. Send the address out on Port 2.
   c. Set OE’ low which tells the RAM to output the data.
   d. Read the data with Port 0.
5. After the code is ready, compile the code in Microvision to obtain the HEİX file corresponding to your code.
6. Create the VHDL ROM module from the HEİX code using the procedure in Appendix ‘C’.
7. The RAM module will be provided to you by the instructor. Then integrate the components as depicted in the Figure 3 below.

Fig 3: The Top level Schematic

8. Make the Pin assignments as specified in Appendix ‘D’ and program it onto the FPGA. Get the output checked by the instructor.

QUESTIONS:

1. What is the size of the RAM module you were provided with?
2. How many logic blocks in the FPGA were needed by your design?
3. A ROM module does not have one control signal that a RAM module has. Which one? Why?
4. Was ALE signal needed in your design? Why?