INEL 6048 Advanced Microprocessor Interfacing - Laboratory Exercise II

Prof. Manuel Toledo - ECE Dept. UPRM - Fall 2005

October 5, 2005

• **TASK DESCRIPTION:** Your task is to develop the interface between a *Microblaze* microprocessor running in XILINX’s *Spartan 3 Starter Board* and ATMELE’s AT28BV256 EEPROM. Specifically, you should
  - integrate the EEPROM using XILINX’s implementation of the *On-chip Peripheral Bus (OPB)* and VHDL,
  - write code to access the chip from software,
  - write a test program to use the board’s rs232 port and Window’s *Hyperteminal* program to test your design.

• **Due on October 20 during the course’s lecture.**

• Late demos will be accepted after the deadline, but must take place right after lecture. A penalty for late delivery will be applied as follows:
  1. 15 points if delivered on 10/25.
  2. 30 points if delivered on 10/27.
  3. 50 point if delivered on 11/1.
  4. No late work will be accepted after 11/1.

• **ORGANIZATION:** You should work in groups of two. Each group should deliver two versions of the work, with each student being responsible for developing the hardware/VHDL code of one version, and the software/C-code of the other version.

• To obtain full credit, by the deadline you should:
  1. develop and implement the interface,
  2. demonstrate it in the laboratory,
  3. turn-in a report with the following information:
     (a) textual descriptions of the two versions of your VHDL and C code,
     (b) block and state diagrams describing the functionality of each version,
     (c) source code.

• **SYSTEM’S FUNCTIONALITY:** Your design should enable the user to read/write decimal (base 10) data from/to the EEPROM. Specifically,
  1. If the user enters a read command on *Hyperteminal’s* window, as in *R 0FF*, your system will print the 32-bit data stored at EEPROM’s address 0FF on the computer’s screen, using base 10 to display the quantity.
  2. An optional second number between 1 and 15 can be included to specify the number of data items that should be printed. For example, entering *R 0FF 5* causes five 32-bit values to be printed, starting with the one at address 0FF.
  3. If the user enters *W 0FF 1234*, your system will store the quantity 1234₁₀ in the EEPROM at address 0FF hex, and write a message indicating if the task was properly executed.
  4. A user can also store up to 16 quantities in a single instruction. For example, by issuing command *W 0FF 1234 1235 1236 1237* the four values 1234₁₂₃₅₁₂₃₆₁₂₃₇ will be stored, starting at address 0FF.
  5. From the user’s point of view, all data is 32 bits. However, the EEPROM stores data in bytes. Your design should provide the functionality to make this difference invisible to the user. Thus, for users the device is organized in 8192 32-bit words, although internally the device is organized as 32k bytes of storage. Your code must do the necessary address translation and properly reorganize the data.
  6. Your code should be modular and of high quality, or points will be deducted from your grade.