Experiment 4

Extending the 8051 with External Hardware: An Address Latch and Memory-Mapped Port

INTRODUCTION:

In this lab, you play the role of a hardware designer developing components to work with the 8051 and to extend the 8051's capabilities. The software-design team has not yet finished their design, but they have created a software model that implements the most important features of that design. You will use this model to test hardware you develop. By simulating hardware and software early, you have the opportunity to detect and solve problems in your hardware design before that design is set in stone (or set in silicon, as the case may be). Waiting for a hardware prototype to be finished before testing software could add to your troubles. The hardware may be too far along to make changes cost effectively, and some corrections may be impossible to make with software alone. Changing the hardware late in the design could add significant costs to development and could significantly extend the time-to-market, which could be disastrous to your product's success. By testing the design early and often, you can quickly and easily correct the problems that occur. Of course, testing should never replace the process of carefully laying out and developing the design in the first place – a design which is well thought out before any implementation is done is bound to be the best.

You will design hardware to allow the 8051 on the Altera DE-2 board to use ROM module that stores the instructions. The 8051 module does not contain internal code memory. To access an external IO, it requires an external address latch as the lower address byte and the data are multiplexed on port 0. You will incorporate this external latch in the FPGA on the Altera DE-2 board using the eight-bit latch designed in lab 1. You will also create a memory mapped output port for the seven segment display from another instance of this same eight-bit latch.

Schematics will be developed in Quartus-II software and will be simulated using the same. An EDIF module of the 8051 microcontroller will be provided for you by the instructor. Once your design has been verified through simulation, you will verify your design in hardware.

OBJECTIVES:

1. Improve hardware-software design skills.
2. Learn how to de multiplex the 8051 address/data bus.
3. Observe the timing of signals involved in a code-fetch memory cycle.
4. Introduce a simulation model of the 8051 microcontroller.
5. Learn how to add a memory mapped output port to the 8051.

MATERIALS REQUIRED:

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

BACKGROUND:

The 8051 requires external code memory because it contains no internal code space. A latch is needed to latch the lower eight bits of the address from port 0 because this port is multiplexed with both data and addresses. Figure 1(a) shows a symbol for the 8051 with external memory and an address latch interfaced to it. The schematic shows that port 0 is connected both to the memory address lines and to memory data lines.

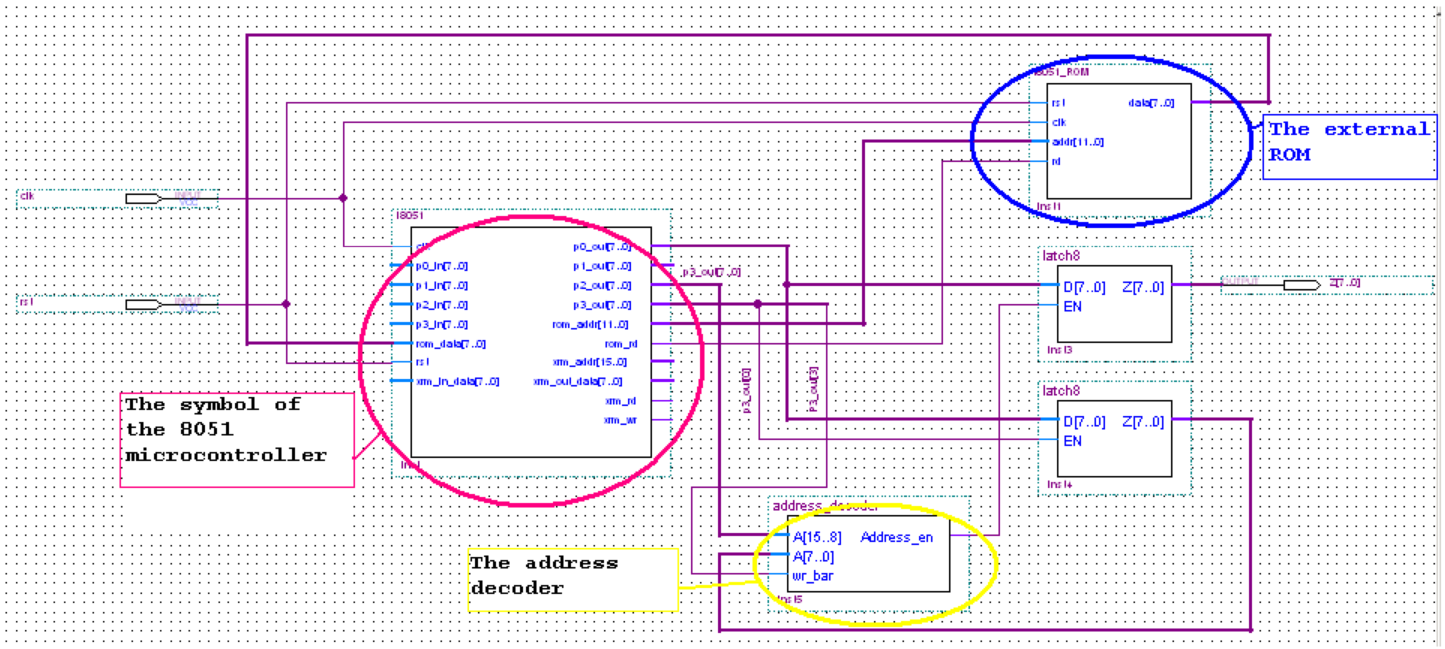


Figure 1(a): Top level Schematic for Lab 4.

Above figure also gives the complete schematic of the entire system.

Figure 1(b) is a timing diagram for an external memory access. A memory access proceeds as follows: 1) The 8051 writes the lower and upper address byte out port 0 and port 2, respectively; 2) On the first falling edge of ALE (address latch enable), the low byte (PCL) from Port 0 is latched. Once latched, port 0 is available to receive data from memory; 3) PSEN (program store enable) goes low, enabling memory to send data back to the 8051. 4) Data – the opcode of the next instruction – is received by the 8051. This process is illustrated in the timing diagram in Figure 1(b). Notice that Port 2 does not need a latch because its value does not change during a single bus cycle.

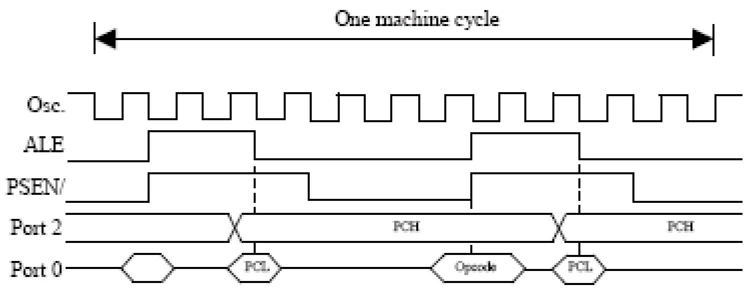


Figure 1(b) 8051 machine cycle

To free up ports on the 8051, external devices can be mapped within the microcontroller’s data memory address space. In this case, you can read or write the device by using a MOVX instruction. In this lab, we would like to map the seven-segment display to memory location 0xAA55. We will put a latch there so that data we write will be shown on the seven-segment display, even after the microcontroller has gone on to other tasks. The block diagram is illustrated in Figure 2.

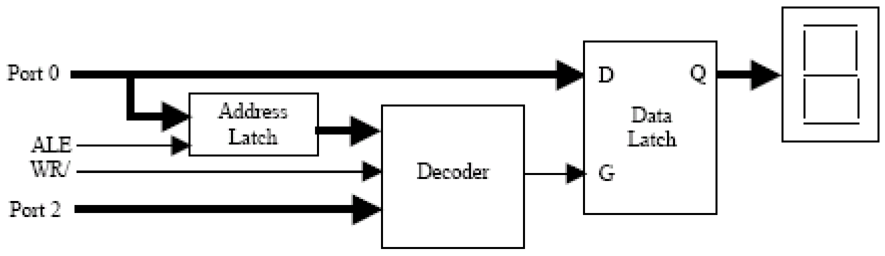


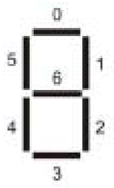
Figure 2 Memory mapped output port

The address latch, just as before, the 8051 will first write an ad address, which will be stored at this latch. During a data write instruction, the 8051 would then write data, which would be stored by the data latch if the address was 0xAA55. Data is “written” when the WR/ line from the 8051 goes low. Data is latched by the data latch, then, when both the WR/ line is low and the address is 0xAA55, indicating that the 8051 has performed a MOVX to external data memory location 0xAA55. As far as the 8051 is concerned, the seven segment display is just another external memory location.

PROCEDURE:

Summary: Create an address latch, address decoder and external ROM in Quartus-II software and integrate it with the 8051 module as shown in the above figure. Then display a word HELLO on the Seven Segment display of the Altera DE-2 board corresponding to the Address 0xAA55. This code is stored in the external ROM.

1. Create the address decoder for the seven-segment display. Draw the logic for your decoder within the sheet in the Quartus-II. Connect the inputs and outputs from the decoder to port in and port out ports. Using ports will allow you to create a decoder “symbol”.
2. Similarly create the symbols for the ROM which contains the code for the displaying HELLO when the address 0xAA55 is detected and connect it to the 8051 and the 8051 as shown in the figure above.
3. The code for the ROM is first written in C and is compiled in the Keil Microvision software. The format for writing the code has been described in the Figure 1(b). The C code should be in the accordance with the waveform shown.
4. Now the data that will correspond to the address specified that is 0xAA55 and that has to be displayed on the 7 segment display is the word HELLO. The corresponding code for the word HELLO can be calculated depending on how the consecutive words would be displayed on the 7 segment display as shown in the figure below. The code for the word ‘H’ has been formulated here to give an example of the method.



Now ‘H’ will be displayed on the 7 segment display when the corresponding segments 1,2,4,5 and 6 are all turned 0. Now since the data is 8 bits therefore keeping the 8th bit as a ‘0’ always, thus the data code for displaying ‘H’ will be 00001001 that is 0x09. In the following manner all the data codes can be formulated corresponding to the words.
The next step is to convert this C code into the corresponding vhd format for making it compatible to be synthesized in the Quartus-II software which is done by executing the ‘a.out’ file given to the students by the instructor.

5. Then integrate entire system with the 8 bits latches as shown in the Fig.1(a) and compile the design in Quartus-II.
6. Finally assign input and output pins to the necessary ports and download the design onto the Altera DE-2 FPGA board using the Programmer tool in Quartus-II.
7. Observe the output and show the hardware implementation to the instructor and make the corrections specified if any.

QUESTIONS:

1. How many Logic Blocks in the FPGA were needed by your design?
2. Explain how the address/data multiplexing takes place in your design.
3. Why the data port is called “memory mapped port”?