Lab 2 - LED Display Peripheral

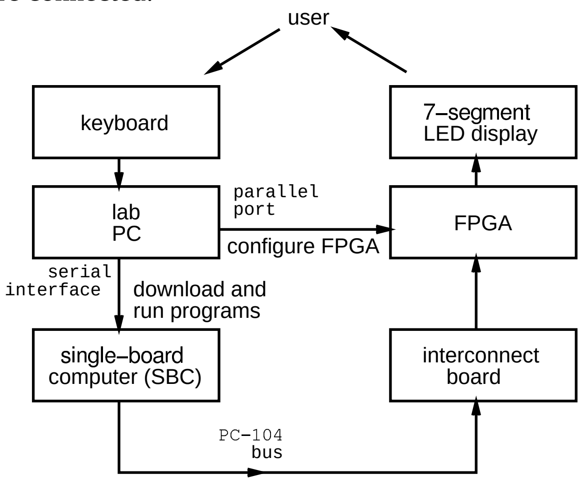
Introduction

In this lab you will design a computer interface that displays a digit from 0 to 3 on an LED. This device will be hooked up to the system bus of a PC-compatible single-board computer (SBC). You'll also write an assembly-language utility to change the displayed digit.

You will use VHDL to design a circuit that loads a register in response to write cycles to I/O memory (port) address 220H. The LS two bits of the register value are decoded to drive a seven-segment LED display as in the previous lab.

Your utility program will use DOS to read a character ('0' to '3') from the user and write the corresponding two-bit binary value (0 to 3) to I/O port 220H.

The diagram below shows how the components are connected:



Hardware Description

The Single-Board Computer

The single-board computer (SBC) in the lab contains a 386EX microprocessor, 2 MB of dynamic RAM (DRAM), 1 MB of "flash" EEPROM used as a virtual disk drive, three PC-compatible interface ports (two serial and one parallel), and several PC-compatible support chips (timers, interrupt- and DMA-controllers and a real-time clock). The SBC boots a modified version of the DOS operating system from the flash disk when it is reset.

The SBC does not have a video display. Instead, you use a "terminal emulator" program such as Windows' Hyperterm to issue DOS commands to the SBC through the serial interface.

Although the SBC has enough memory to run simple DOS programs, it does not have software development tools installed on it (editor, assembler, debugger, etc). You will edit and assemble programs on the lab PC, download the compiled program (the .COM file) through the serial interface to the SBC and then run your programs on the SBC.

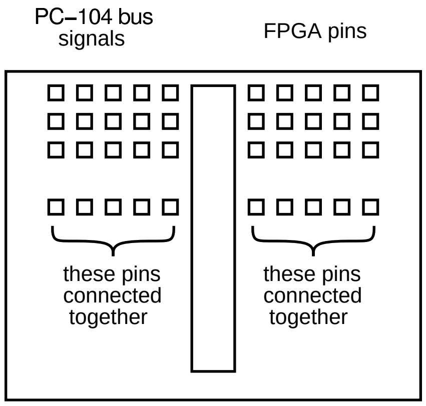
The SBC has a system bus that allows peripheral interface cards such as video displays, LAN cards, analog interfaces, etc to be added to the computer. The system bus used in the SBC is a PC-104 bus which is electrically the same as the ISA bus found in most PCs but with a more compact and robust 104-pin connector. Most of the ISA/PC-104 bus signals are the same as the signals found on the '386SX processor, but the PC-104 bus on the SBC in the lab only supports an 8-bit data bus (D7 to D0) and a 20-bit address bus (A19 to A0).

In this lab you will use an FPGA to develop a peripheral device that interfaces to the SBC through the PC-104 bus. We will only use a subset of these bus signals: the data bus (D7 to D0), the LS 10 bits of the address bus (A9 to A0), and the IOW* signal. The IOW* signal is generated by combining the CPU's M/IO* and W/R* signals to create a signal (strobe) that is low only during a write cycle to the I/O space.

Interconnection Board

Connections between the PC-104 bus and the FPGA's pins are made by inserting jumper wires on two solderless prototyping board. As shown in the diagram below, the five holes on each horizontal row of the prototyping board are connected together.

Each row of five holes is also connected (under the board) to either a PC-104 bus signal or to an FPGA pin. The holes on the left rows connect to PC-104 bus signals and the holes on the right rows connect to FPGA pins.



There are two interconnect boards. The left interconnect board is used to connect the PC-104 data and address bus and the board on the right is used to connect the memory/I/O read/write strobes and the interrupt request lines.

The table below shows the PC-104 signals and the FPGA pins that are connected to each row of the interconnect board. The rows are numbered starting at 1 for the top row.

	Left Board		Right Board	
Row	PC-104 Signal	FPGA Pin	PC-104 Signal	FPGA Pin
1	D7	113		
2	D6	114		
3	D5	115		
4	D4	116		
5	D3	117	IOW*	157
6	D2	118		
7	D1	119		
8	D0	120		
21	A9	138		
22	A8	139		
23	A7	141		
24	A6	142		
25	A5	143		
26	A4	144		
27	A3	146		
28	A2	147		
29	A1	148		
30	A0	149		

The FPGA connections to the LED are given in the previous lab.

The `.acf` configuration file is a text file. It may be easier to edit the this file with a text editor when entering the pin assignments rather than using the dialog box. You can cut-and-paste the pin assignment section from the sample `.acf` file available on the course Web page. Don't edit the `.acf` file while Max+PlusII is running or your changes may be overwritten.

The Lab PC and Software

The lab PC will be used to:

- synthesize the VHDL code and configure the FPGA using the Max+PlusII software
- edit the program using Notepad and assemble it using the free "valarrow" assembler and linker
- download the program to the SBC, run it and enter the digit to be displayed using the Hyperterm terminal emulator

Pre-Lab Assignment

Before the lab you must write, assemble and test (to the extent possible) the utility program. You must also design the circuit and test it by simulating its operation. The TA will ask to see your assembler and VHDL code and the simulation waveforms at the start of the lab.

Assembly Language Program

Write an 8088 assembly-language program that does the following:

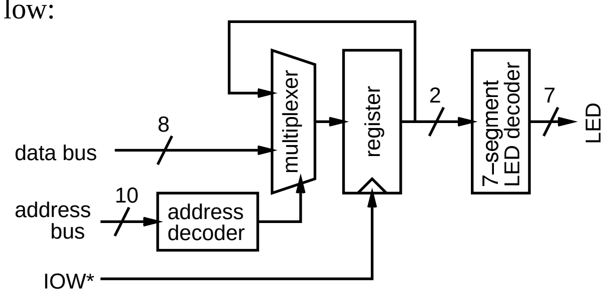
- loads the AH register with the value 1 and executes software interrupt 21H to request that DOS read a character from the standard input (keyboard) and return the ASCII value of the character in the AL register
- subtracts the ASCII value for the character '0' (zero) from the returned value
- outputs this number to I/O port 220H
- executes software interrupt 20H to return control back to DOS

The instructions for downloading and using the free "valarrow" assembler are available on the course web page. Download the assembler from the course web page and assemble the code to create an executable .COM file.

Run the program under DOS. The program should simply return control back to DOS. If your operating system has memory protection (e.g. Windows NT) or port 220H is used by a peripheral on your computer the computer may give you an error message or your computer may crash. In this case comment out the OUT instruction and test the rest of the code.

VHDL Code

Write a VHDL description for the circuit shown below:



The inputs are the LS 10 bits of the address bus[1], the 8 bits of the data bus, and the IOW* write strobe. The outputs are the seven LED segments as in the previous lab.

Your VHDL code should use the IOW* signal as if it were a clock. Use the rising edge of IOW* to load the register. The value loaded into the register should be either the value on the data bus (if the value of the address bus is 220H) or else the current value of the register. The register should be 8 bits wide (although only two bits are actually required and the synthesizer will warn you that the other bits are not being used). Do *not* include the address decoding function in the process statement (i.e. do not "gate the clock"). You can probably re-use the LED decoder code from the previous lab.

Create simulation test waveforms that demonstrate the following:

- an I/O write to address 220H changes the LED output to the correct value
- I/O writes of all four valid values (0-3) generate the correct LED outputs
- an I/O write of a value to address 21FH does not affect the LED output

[1]The original IBM PC design only decoded the LS 10 bits of I/O port addresses so all PC-compatible designs restrict themselves to using only the first 1024 ports.

Compile and simulate your VHDL description as described in the previous lab.

Print and Copy Files

Save the files *projectname*.`asm` (assembly language source code), *projectname*.`com` (DOS executable), *projectname*.`acf` (device and pin assignments), *projectname*.`vhd` (VHDL code), and *projectname*.`scf` (test waveforms) to a floppy disk to bring it with you to the lab. Print out the assembler and VHDL code and the simulator output waveforms.

Lab Procedure

Connect the PC-104 bus signals to the FPGA pins on the interconnect board as described above. Use the short 22-gauge jumper wires provided in the lab. You will need 10 jumpers for the address bus, 8 for the data bus (although only two are really used in this lab), and one for the IOW* strobe. Double-check your connections and turn on the power.

Compile your VHDL code if you haven't already done so, and configure the FPGA as described in the previous lab.

Assemble and link your assembly code if you haven't already done so. All of your files should be stored in the `c:\max2work` directory.

Run the Windows Hyperterm program (under the `Start|Accessories|Communication` menu). Click on the ELEC379 icon[2]. Press the reset button on the interconnect board (top left corner). The SBC will reboot and display a menu. Enter 'x' to exit the start-up menu.

You can now issue DOS commands to the DOS operating system running on the SBC (e.g. DIR, CD, PATH, etc). To download your program to the SBC, run the "dl" (download) command *on the SBC*. The

[2]If there is no such icon, create a new configuration using the Connect option "direct to COM1" with configuration settings of 9600 bps, 8 data bits, no parity, 1 stop bit and xon/xoff flow control).

SBC will output junk to the screen. Use the Hyperterm menu option `Transfer|Send File` to bring up a dialog box. Select your `.com` file and download it. The file will be transferred between the development PC and the SBC over the serial port. The negotiation between the two PCs will take 10 to 15 seconds and the transfer a few seconds more. When the transfer is complete you will be returned to the SBC's DOS prompt.

Run your program on the SBC. It should wait for you to type in a digit, display the digit on the LED and then return control back to DOS.

When your device is working properly, ask the TA to check your work. He will make sure your device works as required and ask you one or two questions to verify your understanding of the material.

Report

Submit a short report with a written description of your circuit. Include a block diagram showing the connections between the PC-104 bus, the FPGA and the LED, a listing of your assembly-language program, the VHDL code and a printout of the simulation waveforms that demonstrate correct operation of your device.