Lab 2 - LED Display Peripheral

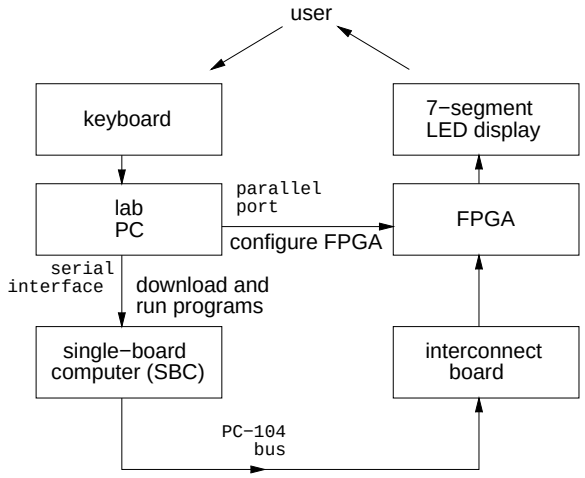
Introduction

In this lab you will design a computer interface that displays a digit between 0 and 4 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 224H. The LS three bits of the register value should be decoded and drive a seven-segment LED display as in the previous lab.

Your utility program will use DOS to read a character ('0' to '4') from the user and write the corresponding two-bit binary value (0 to 4) to I/O port 224H.

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 runs 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, it uses the serial port to communicate with the user. 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, linker, etc). You will edit and assemble programs on the lab PC, transfer ("download") the compiled program (the executable `.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 configure an FPGA to act as 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.

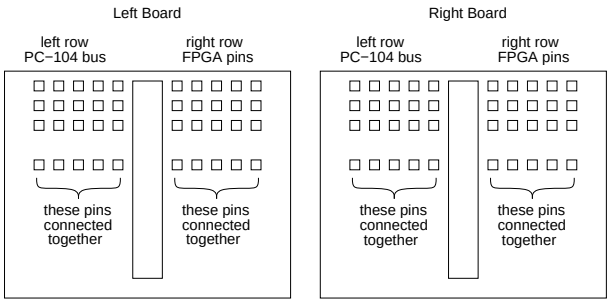


Figure 1: PC-104/FPGA Interconnection.

Interconnection Board

Connections between the PC-104 bus and the FPGA's pins are made by inserting jumper wires into two solderless prototyping boards. As shown in figure 1 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.

Table 1 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.

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

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 or the DOS "Edit" program 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

Left Board			Right Board		
	left row	right row	left row	right row	
Row	PC-104 Signal	FPGA Pin	PC-104 Signal	FPGA Pin	Row
1	D7	113	RESET	90	1
2	D6	114	IRQ9	163	2
3	D5	115	MEMW*	154	3
4	D4	116	MEMR*	156	4
5	D3	117	IOW*	157	5
6	D2	118	IOR*	158	6
7	D1	119	SYSCLK	211	7
8	D0	120	IRQ7	109	8
			Ground		14
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			
31	Ground				

Table 1: PC-104 and FPGA Signal Locations.

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

- uses DOS to read a character from the keyboard. The program loads the code 1 into AH and executes software interrupt 21H. DOS will read a character from the standard input (keyboard) and return it in AL.
- subtracts the ASCII value for the character zero ('0' or 030H) from the returned value to obtain

a value from 0 to 9

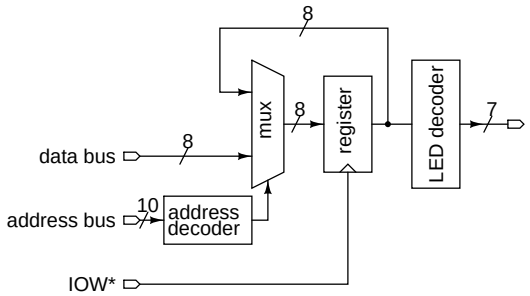
- outputs this value to I/O port 224H
- executes software interrupt 20H to return control back to DOS

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 wait for you to press a key and return to DOS. If the operating system has memory protection or if port 224H is used by a peripheral on your computer, the computer may give an error message or it may crash. In this case comment out the OUT instruction and test the rest of the code.

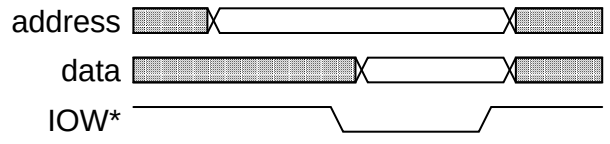
VHDL Code

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.

The following figure shows the behaviour of the address, data and IOW* signals during an I/O write cycle:



[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.

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 224H) or else the current value of the register. The register should be 8 bits wide (although only three 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 and register code from the previous lab.

Create simulation test waveforms consisting of five bus cycles that demonstrate the following:

- an I/O write to address 224H of all five valid values (0-4) changes the LED output to the correct value
- an I/O write of a value to address 21FH does not change the LED output

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 may have to make your own jumpers using the wire and wire strippers supplied in the lab. You will need 14 jumpers: 10 jumpers for the address bus, 3 for the data bus (only the LS three bits are 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.

If you haven't already done so, assemble your assembly code (`asm` file) using `asm` to produce an object (`obj`) file and then link the object file using `val` to produce a binary (`.com`) file. All of these files should be stored in your `c:\max2work\yourname` directory.

Run the Windows Hyperterm program (use the desktop shortcut or see under the `Start|Accessories|Communication` menu). Click on the 379com1 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 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.

Read the logic analyzer description given in the appendix and connect the logic analyzer as follows:

- probes B1, B0, A7–A0 to PC-104 bus address bus signals A9 to A0
- probe B7 to IOW*
- probes C2–C0 to LS 3 bits of the PC-104 data bus

You will need another 14 jumper wires to hook up the logic analyzer. Label probes B1, B0 and A7–A0 as "ADDRESS", B7 as "IOW*" and C1–C0 as "DATA". Set the logic analyzer to trigger on writes to I/O address 224H. Press the logic analyzer "Run" button and switch back to the Hyperterm window.

Run your program on the SBC. It should wait for you to type in a digit, write to the register on the FPGA and then return control back to DOS. The number should be shown on the LED. Switch to the logic analyzer window and verify that the event was captured.

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.

[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).

Report

Submit a short report with a written description of your circuit. Include a block diagram of your VHDL circuit and another 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.

Appendix - The Logic Analyzer

Introduction

In previous lab courses you've used an oscilloscope to measure and display analog signals as voltage versus time. The corresponding instrument for measuring digital logic circuits is called a *logic analyzer*. The logic analyzer displays the values of logic signals as a function of time. This appendix describes how to operate the PC-based logic analyzer that is available in the lab. You can use this instrument to view the signals on the interconnect board and this will help you debug your circuit.

Logic analyzers connect to digital circuits through sampling "pods" that contain buffers. The logic analyzer in the lab has one pod with 24 one-bit inputs. Each input is connected through a short, colour-coded wire to your circuit. The 24 inputs are labelled with a letter and a number (A7 to A0, B7 to B0, and C7 to C0). The colour codes for the wires are the same as the resistor colour codes (listed on the pod). The white wire on each pod should be hooked up to ground.

The logic analyzer can treat groups of signals as buses and display the values of the bus signals in hexadecimal.

The logic analyzer does not sample continuously. Instead, you define a "trigger condition" and press

the “run” button. The logic analyzer waits until the trigger condition is true and then takes 32k samples of the 24 input signals at a rate of up to 50 MHz.

To run the logic analyzer select the PC’s “bi2450P” program menu item. Unfortunately, the logic analyzer software is a DOS program that can only run in full-screen mode. Press alt-Tab to switch between the logic analyzer and other Windows programs.

Connect Probes

First, connect each of the white wires on the pod to ground.

Then connect the probe wires to the signals you want to monitor by putting a short jumper into the connector on the end of the probe wire and connecting the other end of the jumper to the appropriate row on the interconnect board.

If you open the probe assignment dialog box (see next section) while making the connections you will be able to see the current signal level (H(igh), L(ow) or changing) on that signal.

Assigning Channels

Select the menu item `Setup|Probes` to bring up a dialog box where you can assign probes to signals. For each signal you want to observe, type in a signal name, the radix to use when displaying the signal value, and click on the probes to be assigned to that signal. The leftmost probe is taken to be the most-significant bit of a bus. Click on `Done`.

Trigger Conditions

Select the menu item `Setup|Trigger Words` to bring up a trigger word dialog box. In the trigger word “A” fields enter the signal values corresponding to the event that you want the logic analyzer to capture. Use ’X’ as a “don’t care” value.

For example, if you wanted to observe all write cycles to a particular memory location, you would put the desired address as the value in the address signal field, and the asserted value as the value of the write strobe signal while leaving the data bus value set to ‘X’s (the logic analyzer will actually capture data before and after that particular condition). Click on `Done`.

Sample Rate

Click on `Timebase` in the main display window and set the sample rate to 50 MHz.

Save Settings

Select the menu item `Setup|Save Setup` and select a file name (in the `max2work/yourname` directory) to save your settings in case you need to re-load them later.

Capture Signals

Press the `Run` button to start the logic analyzer. When the inputs match the trigger words the logic analyzer will display the signals on the main display window.

The logic analyzer has many more features, some of which will be introduced as necessary.

Select the menu item `File|Quit` to exit the program.