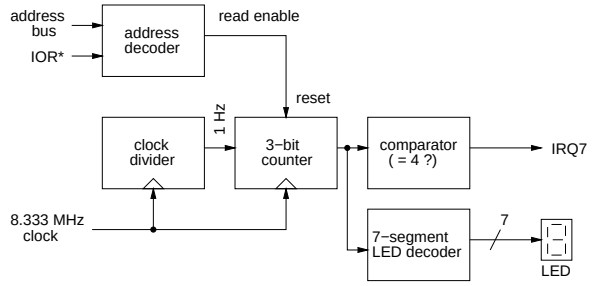
Lab 5 - Interrupt-Driven Timer

Introduction

In this lab you will design and implement a timer device that generates one interrupt every 4 seconds. You will also write a program in 8086 assembly language that initializes an interrupt vector and provides an interrupt service routine (ISR) for the device.

Hardware Description

The following diagram shows a block diagram of the device:



The device has a 3-bit counter that is incremented once per second. The 1 Hz clock is generated using a clock divider as in Lab 3. The count is displayed on a 7-segment LED display as in Lab 2. The port generates an interrupt by asserting the PC-104 bus interrupt request number 7 (IRQ 7) when the counter value reaches 4. This bus signal is connected to IR7 on an 8259 PIC.

The port has a read-only register at I/O address 220H although nothing is put on the bus during a read from this address (to keep things simple). However, reading this register should reset the counter to zero (the read cycle will take longer than the clock period). Thus, if your circuit and ISR are working properly the count should start again from 0 as soon as the count reaches 4.

The address decoder is set up to recognize (and reset the counter) for I/O memory reads of address 220H.

Your design must be synchronous and use only the 8.333 MHz PC-104 bus clock, SYSCLK.

Pre-Lab Assignment

Before the lab you must write and assemble the program described below. You must also design the circuit and test it by simulating its operation (assume a 3 Hz clock rate when testing). You will be asked to hand in your assembler coder, the 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:

- initializes the interrupt vector for IRQ 7 to point to an interrupt service routine included within your program
- enables interrupt number 7 in the PICs interrupt mask register
- executes an infinite loop (a JMP instruction to itself)

The ISR, included as part of your program, should do the following:

- save any registers it will modify
- read the status register at 220H
- re-enable interrupts by issuing the EOI instruction to the PIC
- restore any saved registers
- return from the ISR

Refer to the lecture notes for details on the locations of interrupt vectors, 8086 interrupt service routines and the operation of the 8259 PIC.

VHDL Description

Write, compile and test by simulation a VHDL description of the circuit described above. Use the following tests in your simulation to help verify its correct operation:

- wait for enough clock cycles to elapse so that the count reaches 4. Verify that the IRQ7 request is asserted and that the LED outputs are correct.
- read from memory location 220H. Verify that the IRQ7 output is reset and that the count goes back to zero.
- wait for enough clock cycles to show that the count goes back to 1.

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 address bus signals, IOR*, SYSCLK, and IRQ7 to the FPGA pins on the interconnect board as described in the previous lab. 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.

Observe the LED display. The count displayed on the LED should run from 0 to 7 and wrap around back to 0 when your ISR is not running.

Assemble, link and download your assembly code if you haven't already done so. Run your program on the SBC.

If your device and program are working properly, the LED will now count up to 3 and then get reset to 0 by the ISR.

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 listing of your assembly-language program, the VHDL code and a printout of the simulation waveforms that demonstrate correct operation of your device.