Lab 3 - Timer Peripheral

Introduction

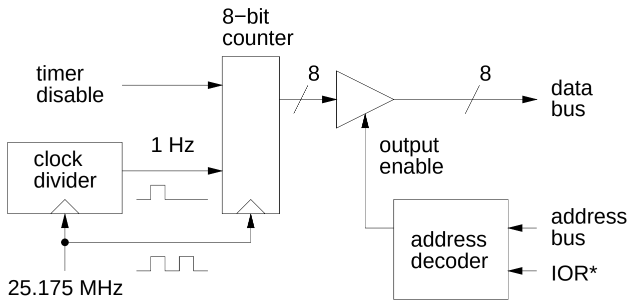
In this lab you will design a timer peripheral chip. These chips are often used in computers so that programs can keep track of the elapsed time (e.g. to keep track of the time of day or to generate interrupts to implement multi-tasking).

You will design an interface to the SBC's bus that will allow the the timer to be read over the PC-104 bus. The timer is visible to the CPU as one 8-bit register that increments once per second. The timer also has an external disable input which is controlled by a push-button on the FPGA board. The clock signal into the timer will come from a 25.175 MHz oscillator that is already connected to one of the FPGA pins.

You'll also write an assembly-language demonstration program that will run on the SBC. This program should continuously read the contents of the timer register and displays the value on the SBC console as a 2-digit hexadecimal value.

Hardware Description

The diagram below shows the internal structure of the timer:



Clock Divider

The clock divider circuit is a counter that counts from 0 to 25,174,999 (the clock frequency-1) and then starts at 0 again. An output from the divider is asserted during the clock cycle where the count is equal to 0. The frequency of the clock divider output is thus 1 Hz and the period is 1/25,175,000 s. Design this part of the circuit in VHDL as an entity which can be included as a component in the rest of the design.

Timer

The timer circuit is an 8-bit counter that is incremented when the clock divider output is asserted and the disable input is not asserted. It thus counts up at 1 Hz. Note that the 1 Hz signal does *not* act as a clock signal – it is simply another signal that enables/disables the counter. The 25.175 MHz signal should be the *only* clock signal in your design.

CPU Interface

The SBC should be able to read the count held by the timer circuit at location 220H. You should therefore design the CPU interface so that it places the contents of the timer register on the data bus in response to a read cycle to I/O memory (port) address 220H as indicated by the address on the address bus and the PC-104 IOR* signal. A logic circuit on the SBC generates this IOR* signal by OR'ing the 386SX CPU's M/IO* and W/R* signals to create a signal (strobe) that is low only during a read cycle to the I/O space.

Your circuit must tri-state it's data bus pins except during an I/O read to memory location 220H to avoid contention between the different devices that share the data bus.

Note that the IOR* signal is not treated as a clock. Unlike the the CPU interface to write to a register, the CPU interface to read a register requires only combinational logic.

Again, your design should be *synchronous* (use only one clock).

The FPGA connections to the data bus, address bus, IOR* and pushbutton were given in previous labs. The 25.175 MHz clock is permanently connected to the FPGA chip on pin 91.

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:

- inputs a byte from memory location 220H
- prints the value as two hex characters
- prints a carriage return (ASCII decimal 13)
- repeats these steps indefinitely

Create an executable .COM file as described in the previous lab. When you run this program it should repeatedly print a 2-digit hex number at the left margin. Press control-break to interrupt the program.

VHDL Description

Write, compile and test by simulation a VHDL description for the timer circuit described above. Create simulation test waveforms that demonstrate the following[1]:

- a read from address 221H leaves the bus in a high-impedance state
- a read from address 220H puts a value of zero on the data bus only when IOR* is asserted

The instructions for compiling and simulating your VHDL description are given 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.

[1] Unfortunately, it is not practical to simulate the operation of the device over approximately 25.175 million clock cycles.

Lab Procedure

Use short jumpers to connect the PC-104 address, data bus, and IOR* signals 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.

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

Run the Windows Hyperterm program. Reset the SBC and download your program.

Run your program on the SBC. It should continuously display the timer contents which should be incrementing once per second. Hold down the pushbutton button to disable counter and verify that it is no longer incrementing.

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