Lab 3 - Timer Peripheral

Introduction

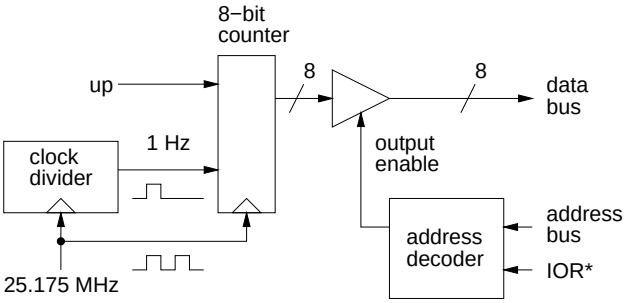
In this lab you will design a timer peripheral chip. These chips are often used in computers so that programs can keep track of the elapsed time (e.g. to keep track of the time of day, to generate interrupts and to implement multi-tasking).

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 or decrements once per second. The timer has an external up/down control input, `up`, which is controlled by a push-button. 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 timer register and display the timer value on the SBC console (via Hyperterm).

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. Use generics to specify the divider ratio (25,175,000 in this case) and the number of bits required for the counter state (you'll need to calculate this). You will re-use this component in a Lab 4.

Timer

The timer circuit is an 8-bit counter that is incremented when the `up` control is asserted and decremented when `up` is not asserted. It thus counts up or down at 1 Hz. Your design should be synchronous. The clock divider output may *not* to be used as a clock. 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 224H. 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 224H 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 224H 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* (only one clock for all registers).

The FPGA connections to the data bus, address

bus, IOR* and pushbutton 1 were given in previous labs. The 25.175 MHz clock is permanently connected to the FPGA chip on pin 91.

VHDL Description

Write, compile and test by simulation a VHDL description of the timer circuit described above. Set the divider value to 2 for testing. Create simulation test waveforms that demonstrate the following:

- not asserting IOR* with and address of 224H leaves the bus in a high-impedance state
- asserting IOR* when the address is 220H leaves the bus in a high-impedance state
- asserting IOR* with an address of 224H puts the timer value on the data bus
- the counter increments every two clock cycles when `up` is asserted and decrements every two clock cycles when it's not asserted

The instructions for compiling and simulating your VHDL description are given in the previous lab.

Software Description

Write an 8088 assembly-language program that continuously displays the value of the timer register by doing the following:

- inputs a byte from memory location 224H
- if the value is between 0 and 9, it prints it as a decimal digit ('0' - '9') otherwise it prints an asterisk (*)
- prints a carriage return (ASCII code decimal 13)
- exits to DOS if the value is equal to 10 (decimal)
- loops back to the start if it's not

Create an executable .COM file as described in the previous lab. When you run this program it should repeatedly print a 1-digit number (or *) at the left margin. Press control-break to interrupt the program.

Pre-Lab Assignment

Before the lab you must write, assemble and test (to the extent possible) the utility program. You must also test your VHDL code 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.

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

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 counting up once per second. Hold down the pushbutton button to make the counter count down. Let the counter reach 10 (*) and make sure your program exits.

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.

Hook up the logic analyzer if you need help debugging your design.

Report

Submit a short report documenting your design. It must include:

- a block diagram showing the connections between the PC-104 bus, the FPGA, the oscillator and the pushbutton
- a block diagram corresponding to your VHDL description
- a listing of your assembly-language program
- a listing of the VHDL code
- a printout of the simulation waveforms that demonstrate correct operation of your device.

Your documentation should conform to the standards given on the course web page.