MicroBlaze Digital Clock and the Embedded Development Kit (EDK)

Learning Objectives

- Use the Xilinx Embedded Development Kit (EDK) to develop an embedded system
- Configure a MicroBlaze soft-core processor with associated peripherals
- Use a Timer peripheral to create accurate timing intervals
- Use an interrupt service routine
- Use General Purpose I/O (GPIO) to send ASCII characters to an LCD, and receive pushbutton input from Nexys board.

Introduction and Overview

Experiment 2 provided an introduction to both the EDK and to some of the particulars of working with LCD devices. In that experiment, firmware was written using C and was used to control the required operations of the LCD. The requirements and most of the specifications for this experiment are the same as the VHDL-based digital clock in Experiment 1. Whereas Experiment 1 required you to design and build the digital clock using hardware, this experiment requires you to design and implement a digital clock using a microcontroller running firmware. The digital clock design in this experiment once again involves a soft-core microprocessor-based design (MicroBlaze) which is downloaded onto the fabric of the FPGA. This design interfaces LCD module and other peripherals on the Nexys board.

Digital Clock System Requirements

1. The clock should display the current time in one of the formats shown in Figure 1 (dependent upon the actual time). The time output is displayed in a 12-hour display mode; the proper time of day is indicated by using the am/pm portion of the display.

   hh:mm:ss am    hh:mm:ss pm

   Figure 1: Output display format for digital clock.

2. Upon power-up, the clock output should be 12:00:01 am and begin keeping time.
3. The clock’s hours and minutes setting should have the ability to be changed by the user in order to set the clock’s output to the appropriate time.

4. Use the minimum code space (number of words in BRAM) and the minimum number of FPGA programmable logic resources (slices).

**Digital Clock Specification**

Implement an embedded computer system on the Nexys board that can output the digital clock requirements on the LCD screen. The embedded computer system should be centered about a MicroBlaze soft-core processor. Configure the MicroBlaze processor, write the firmware, and download the hardware and firmware into the Spartan 3 FPGA on the Nexys board. The embedded system will send the time output requirements to the LCD on the Nexys board.

The time output should be manually changeable with use of the three buttons (BTN0, BTN2, BTN3) on the Nexys board. Setting the time manually is referred to as the *time-set* mode. The signals used for the time-set mode, a description of their function, and the associated I/O devices are listed in Table 1. These signals are used to put the digital clock into a time-set mode and to manually configure the data associated with the minutes and hours. The minutes and hours display can only be manually changed when the digital clock is in the time-set mode.

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
<th>Nexys board</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET</td>
<td>Puts digital clock in time-set mode; disables seconds output</td>
<td>BTN0</td>
</tr>
<tr>
<td>MN_SET</td>
<td>Increments the minutes output</td>
<td>BTN0</td>
</tr>
<tr>
<td>HR_SET</td>
<td>Increments the hours output</td>
<td>BTN3</td>
</tr>
</tbody>
</table>

**Table 1: Display control functions and configurations of digital clock.**

- **Time-set mode:** The time-set mode is entered by asserting the SET input. When in time-set mode the seconds should not increment.

- **Setting the minutes:** to modify the minutes display, the SET input should be asserted simultaneously with the MN_SET input which causes the minutes display to increment at ¼ second intervals. The minutes display should increment from 59 to 00.

- **Setting the hours:** to modify the hours display, the SET input should be asserted simultaneously with the HR_SET input which causes the hours display to increment at ¼ second intervals. The hours display should increment until reaching 12 and then rolls over to 1 and continues incrementing. The am/pm indicator should change appropriately with the hours display value.

**Digital Clock System Architecture**

The minimum hardware configuration for the Digital Clock consists of a MicroBlaze soft-core processor, program memory, data memory, general purpose I/O peripherals, a timer/counter peripheral (OPB Timer), and the Nexys board. The hardware architecture is shown in Figure 2. Note that BTN1 is reserved for the system reset leaving the other three buttons for the functions in Table 1.

The Timer/Counter input is connected via the on-chip peripheral bus (OPB) while the output is connected to the MicroBlaze Interrupt input. The Timer/Counter requires a clock input and produces an interrupt output. The Timer/Counter interrupt output should be connected to the MicroBlaze interrupt input. You
associate this input/output pair by connecting them to a signal of the same name (both of which have an internal attribute).
The timer should be initialized to generate an interrupt every second. After initialization the foreground task will send the current time to the LCD display and poll the pushbuttons to determine if the clock time-set mode has been entered.

There are two main algorithms required for this code: the mainline code (also called the foreground task code) and the interrupt handler. A possible diagram for implementing the task code is shown in Figure 3(a). This algorithm includes some of the overhead tasks associated with setting up the timer as well as a possible approach to handling the time-keeping functions. To set the interrupt enable bit in the MicroBlaze Machine Status Register (MSR) you will need to use the microblaze_enable_interrupts() function. The microblaze_enable_interrupts function prototype can be found in the my_mblaze\include\mb_interface.h file directory under your project. Your implementation should only write to the LCD when the current time value has changed. Your algorithm should ensure the time data being written to the LCD does not change while it is being written. Your final code should be well structured with function calls handling the major tasks.

The interrupt service routine should initiate the time-keeping function’s time by incrementing the clock one second at each interrupt. You should keep the interrupt handler code as short and efficient as possible. A possible algorithm for the interrupt service routine algorithm is shown in Figure 3(b). You can associate the interrupt service routine with the timer module by accessing the software menu from the Xilinx Platform Studio window and clicking on the “Software platform settings”; then click “Interrupt Handler” on the left side of the window and then provide your signal name under Interrupt Handler. In the Peripheral Options window, add the name of the interrupt service routine in the Interrupt Handler Function box. When entering the name of the interrupt service routine into the dialog box, you only need to enter the name of the function not the return values or arguments (or even the parenthesis).

For documentation on the OPB timer/counter, right click on name and then select VIEW PDF Datasheet. Note that this procedure can be done to get documentation on any IP Catagor component listed in its window.
Digital Clock System Components

The two basic components of the Digital Clock system are the PLD design implemented on the FPGA and the hardware on the Nexys board. The subcomponents of the PLD design include the embedded computer system on the FPGA and its associated firmware. The hardware component on the Nexys board consists of the LCD module and the peripherals on the Nexys board.

Digital Clock System Integration

You must make the proper pin assignments for the GPIO modules. The bit ordering in the EDK version 8.2i must be interfaced with the proper external signal lines; see Lab 2 for review of this pin assignment issue. The system.ucf constraints file is shown in Table 2, and can be modified as you deem appropriate for your design (it is sufficient to satisfy the Lab 2 requirements without any added features).

Digital Clock Design Suggestions

1. Make your design modular. Use functions to handle major portions of the code.
2. Optimize your code. If you waste code space the compiled code might exceed the 8K of BRAM memory allocated in the Spartan 3.
3. Use a flowchart to describe the approach your design will take. Have this completed before you start writing code. If you need help during lab, you will need to show the instructor or TA an up-to-date flowchart describing your design approach.
4. As stated in the report, keep your Interrupt Service Routine (ISR) short. The diagram in the experiment description is one approach to designing a succinct ISR.
5. Don’t write to the LCD unless the time displayed on the LCD needs to be updated.
6. Come up with an incremental implementation procedure. The debug tools are very limited in the EDK environment. An example of an incremental approach would be as follows:
   a. Create a function called Print_message() and re-implement Experiment 2 by calling this function.
b. Write your program such that it keeps time based on a simple delay loop (without setting controls)
c. Add in the controls to your program (the various buttons)
d. Implement a timer and use the ISR to create a one second interval used for time keeping.

#system.ucf file for cpe 329 lab 2 (note that # identifies a comment)

#lcd control signal I/O pin assignments
#pin<2> = RS signal, pin<1> = R/W signal, and pin<0> = E (or R) signal
NET "<signal name>_GPIO_IO_pin<2>" LOC = "P15" ;
NET "<signal name>_GPIO_IO_pin<1>" LOC = "T7" ;
NET "<signal name>_GPIO_IO_pin<0>" LOC = "R5" ;

#lcd data signal I/O pin assignments
NET "<signal name>_GPIO_IO_pin<7>" LOC = "N15" ;
NET "<signal name>_GPIO_IO_pin<6>" LOC = "J16" ;
NET "<signal name>_GPIO_IO_pin<5>" LOC = "K16" ;
NET "<signal name>_GPIO_IO_pin<4>" LOC = "K15" ;
NET "<signal name>_GPIO_IO_pin<3>" LOC = "L15" ;
NET "<signal name>_GPIO_IO_pin<2>" LOC = "M16" ;
NET "<signal name>_GPIO_IO_pin<1>" LOC = "M15" ;
NET "<signal name>_GPIO_IO_pin<0>" LOC = "N16" ;

#nexys buttons: pin<2> = btn(0)[BTN3], pin<1> = btn(1)[BTN2],
#and pin<0> = btn(2)[BTN0]
NET "<signal name>_GPIO_IO_pin<2>" LOC = "K12" ;
NET "<signal name>_GPIO_IO_pin<1>" LOC = "K13" ;
NET "<signal name>_GPIO_IO_pin<0>" LOC = "J13" ;

#system clock source from nexys board
NET "sys_clk_pin" LOC = "A8" ;

#reset signal from btn1 on nexys board
NET "sys_rst_pin" LOC = "K14" ;

Table 2: system.ucf constraints file.
Project Demonstration and Lab Report Submission

When your project is working properly and your lab report is ready to submit, zip up the project and lab report into one file and submit it electronically for grading. Your lab report should also include the following design sections that describe the stages of the standard design methodology as they relate to this experiment:

1. System Requirements
2. Specification
3. Architecture
4. Components
5. System Integration

Your report should include flowcharts and written descriptions of the algorithms you used to implement your design. The flow chart and written description should readily communicate the approach you used to design the digital clock. Flow charts are used to communicate software designs to engineers since other engineers should not have to read the code to determine how a system was designed. Your final write-up should also contain comments indicating that you understood overall concept of the EDK software and design methodology.

Your report should be clear and concise in its description of the digital clock design. Remember, diagrams are a viable method to conveying important design information; consider using them in conjunction with written descriptions of your design.

You lab report should also include a Conclusions section. This section should not be a summary of the procedures or a description of what was done in the experiment. This section should include an analysis of the final system and may include topics such as: were the system requirements met? Is there any error in the final system and if so estimate it? Could you have met the system requirements with a better design or an alternate approach? If so how? What tradeoffs were made in the areas of cost, performance, accuracy, design time, power consumption or reliability (if any of these are appropriate)?

Grading

Successful completion of the system requirements and a well written lab report will earn a maximum grade of 9 out of 10 points. In order to achieve a grade above 9 on this lab, you need to demonstrate a complete understanding of the lab by adding additional features/functionality to it. Some suggested lab enhancement ideas are listed below. Higher complexity will earn a higher grade. Adding other features/functionality is permitted as long as they are approved by an instructor and documented in an “Added Features” section following the conclusion of your report. Any changes made to the original specification due to added features must be listed along with an explanation of why the enhancement was made.

Suggested added features:
- Ability to switch between standard and 24-hour time representation
- Minute and hour decrement buttons
- Add a stopwatch (either count up or count down) on the second LCD line
- Add an alarm that flashes the LEDs when the current time equals the alarm time