Laboratory Exercise 3

Input/Output in a Nios II System

The purpose of this exercise is to investigate the use of devices that provide input and output capabilities for a processor. There are two basic techniques for dealing with I/O devices: program-controlled polling and interrupt-driven approaches. In this exercise, we will use the polling approach. In Laboratory Exercise 4, we will use the interrupt-driven approach. We will use both the Nios II assembly language and the C programming language to implement the necessary software.

As an example of I/O hardware, we will make use of parallel port interfaces in the DE-series Media Computer system implemented on an Altera DE-series board. The background knowledge needed to do this exercise can be acquired from the tutorials: *Introduction to the Altera Nios II Soft Processor* and *Media Computer System for Altera DE-series Board*.

The parallel port interfaces in the DE-series Media Computer were generated by using Altera's Qsys Tool software (which we will use in Laboratory Exercise 5). A parallel port provides for data transfer in either input or output direction. In the Qsys Tool, a parallel port is implemented in the form of a *PIO (Parallel Input/Output)* component. The transfer of data is done in parallel and it may involve from 1 to 32 bits. The number of bits, n, and the type of transfer are specified by the user through the Qsys Tool (at the time a Nios II based system is being designed). The PIO interface can contain the four registers shown in Figure 1.

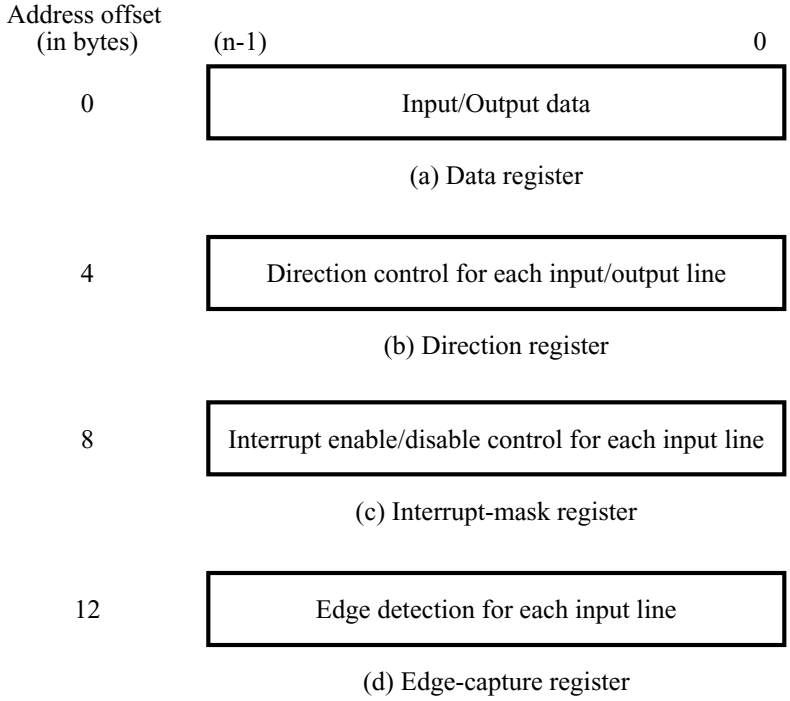

Figure 1. Registers in the PIO interface.

Each register is n bits long. The registers have the following purpose:

- *Data* register holds the n bits of data that are transferred between the PIO interface and the Nios II processor. It can be implemented as an input, output, or a bidirectional register by the Qsys Tool.
- *Direction* register defines the direction of transfer for each of the n data bits when a bidirectional interface is generated.
- *Interrupt-mask* register is used to enable interrupts from the input lines connected to the PIO.
- *Edge-capture* register indicates when a change of logic value is detected in the signals on the input lines connected to the PIO.

Not all of these registers are generated in a given PIO interface. For example, the *Direction* register is included only when a bidirectional interface is specified. The *Interrupt-mask* and *Edge-capture* registers must be included if interrupt-driven input/output is used.

The PIO registers are accessible as if they were memory locations. Any base address that has the four least-significant bits equal to 0 can be assigned to a PIO (at the time it is implemented by the Qsys Tool). This becomes the address of the *Data* register. The addresses of the other three registers have offsets of 4, 8, or 12 bytes (1, 2, or 3 words) from this base address.

The DE-series Media Computer includes several PIOs that are configured for various uses. The details of these PIOs are described in the *Media Computer System for Altera DE-series Board* tutorial.

The application task in this exercise consists of adding together a set of unsigned 8-bit numbers that are entered via the slider switches on the DE-series board. The resulting sum is displayed on the LEDs and 7-segment displays.

Part I

Use 8 slider switches, SW_{7-0}, as inputs for entering numbers. Use the green lights, $LEDG_{7-0}$, to display the number defined by the slider switches. Use the 16 red lights, $LEDR_{15-0}$, to display the accumulated sum. All of these components are connected via parallel ports in the DE-series Media Computer.

Implement the desired task using the Nios II assembly language, as follows:

1. Write a program that reads the contents of the switches, displays the corresponding value on the green LEDs, adds this number to a sum that is being accumulated, and displays the sum on the red LEDs.
2. Create a new directory, *lab3_part1*. Put your program, *lab3_part1.s*, into this directory.
3. Use the *Altera Monitor Program* to create a new project, *part1*, in this directory. Select your program and download the DE-series Media Computer into the FPGA device on the DE-series board. Choose SDRAM as the memory that your program will use. Assemble and download your program.
4. Single-step through the program and verify its correctness by inputting several numbers. Note that single-stepping through the program will allow you to change the input numbers without reading the same number multiple times.

Part II

In this part, we want to add the ability to run the application program continuously and control the reading of new numbers by including a pushbutton switch which is activated by the user when a new number is ready to be read. The desired operation is that the user provides the next number by setting the slider switches accordingly and then pressing a pushbutton switch, KEY_1, to indicate that the number is ready for reading.

To accomplish this task it is necessary to implement a mechanism that monitors the status of the circuit used to input the numbers. A commonly-used I/O scheme is to use a *status flag* which is originally cleared to 0. This flag

is then set to 1 as soon as the I/O device interface is ready for the next data transfer. Upon transferring the data, the flag is again cleared to 0. Thus, the processor can *poll* the status flag to determine when an I/O data transfer can be made.

In our case, the I/O device is the user who manually sets the slider switches. The I/O interface is a parallel port in the DE-series Media Computer. To provide a status flag, we will use the Pushbutton Parallel Port, in which bit position b_1 is connected to KEY_1.

Perform the following steps:

1. Create a new project in a new directory for this part.
2. Modify your application program from Part I to accept a new number when the pushbutton switch is pressed. This action will set the "status flag" bit in the *Edge-capture* register to 1. After adding the number to the accumulated sum, your program has to clear the flag by writing a 0 into the *Edge-capture* register.
3. Download and run your program to demonstrate that it works properly. The program should run continuously and a new number should be added each time the pushbutton switch KEY_1 is pressed.

Part III

In the previous parts the accumulated sum was displayed on the red LEDs. Now, augment your design to display this sum as a hexadecimal number on the 7-segment displays HEX3-HEX0, in addition to the red LEDs. Note that the 7-segment displays are also connected to a parallel port in the DE-series Media Computer.

Part IV

Repeat Part I by writing the necessary program in the C language. After downloading the program into the FPGA device on the DE-series board, place a breakpoint at the start of the loop that reads a new number and adds it to the sum. Then, run the program repeatedly through this loop to verify that it works properly.

Part V

Repeat Part II by writing the necessary program in the C language.

Part VI

Repeat Part III by writing the necessary program in the C language.

Preparation

Your preparation should include the programs for Parts I to VI.