Laboratory Exercise 6

Using UART and Timer Circuits

The purpose of this exercise is to learn how to send and receive data to/from I/O devices by using a UART circuit, and how to time events by using a timer circuit. We will use the DE-series Basic Computer downloaded onto an Altera DE-series Board. The DE-series Basic Computer includes the necessary UART and timer circuits.

Background

A simple and commonly used scheme for transferring data between a processor and an I/O device is known as the *Universal Asynchronous Receiver Transmitter (UART)*. A UART interface (circuit) is placed between the processor and the I/O device. It handles data one 8-bit character at a time. The transfer of data between the UART and the processor is done in parallel fashion, where all bits of a character are transferred at the same time using separate wires. However, the transfer of data between the UART and the I/O device is done in bit-serial fashion, transferring the bits one at a time.

Altera's Quartus II software includes the Qsys Tool which can be used to implement Nios II systems on FPGA devices. This tool was used to implement the DE-series Basic Computer. It provides an interface of the UART type, called JTAG UART, which establishes a connection between a Nios II processor and the host computer connected to the DE-series board. Figure 1 shows a block diagram of the JTAG UART circuit. On one side the JTAG UART connects to the Avalon switch fabric, which interconnects the Nios II processor, the memory chips, and the I/O interfaces. On the other side it connects to the host computer via the USB-Blaster interface. The JTAG UART core contains two registers: *Data* and *Control*, which are accessed by the processor as memory locations. The address of the *Control* register is 4 bytes higher than the address assigned to the *Data* register. The core also contains two FIFOs that serve as storage buffers, one for queueing up the data to be transmitted to the host and the other for queueing up the data received from the host. Figure 2 gives the format of the registers.

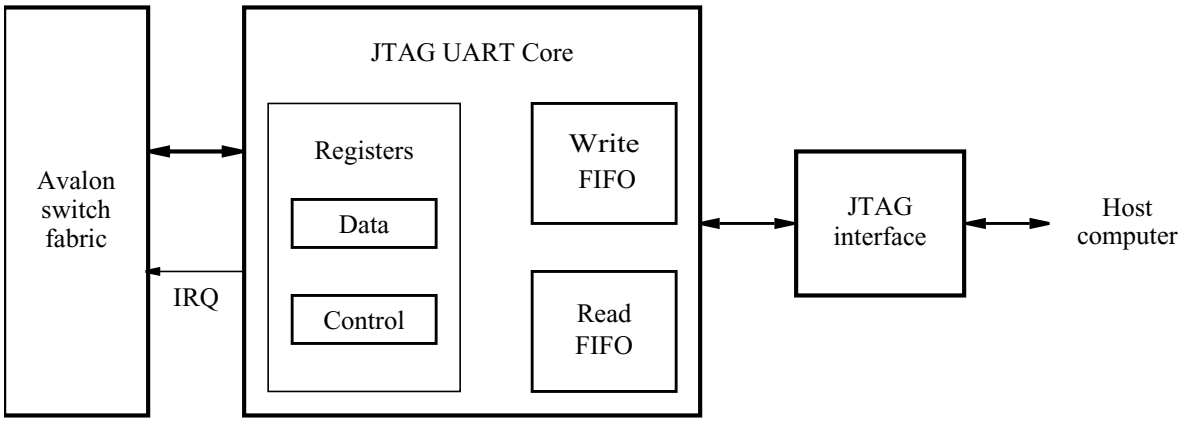

Figure 1. A block diagram of the JTAG UART circuit

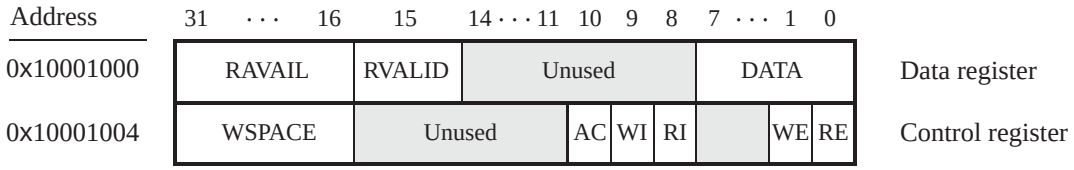

Figure 2. Registers in the JTAG UART

The fields in the *Data* register are used as follows:

- b_{7-0} (DATA) is an 8-bit character to be placed into the Write FIFO when a Store operation is performed by the processor, or it is a character read from the Read FIFO when a Load operation is performed.
- b_{15} (RVALID) indicates whether the DATA field contains a valid character that may be read by the processor. This bit is set to 1 if the DATA field is valid; otherwise it is cleared to 0.
- b_{31-16} (RAVAIL) indicates the number of characters remaining in the Read FIFO (after this read).

The fields in the *Control* register are used as follows:

- b_0 (RE) enables the read interrupts when set to 1.
- b_1 (WE) enables the write interrupts when set to 1.
- b_8 (RI) indicates that a read interrupt is pending if the value is 1. Reading the *Data* register clears the bit to 0.
- b_9 (WI) indicates that a write interrupt is pending if the value is 1.
- b_{10} (AC) indicates that there has been JTAG activity (such as the host computer polling the JTAG UART to verify that a connection exists) since the bit was cleared. Writing a 1 to AC clears it to 0.
- b_{31-16} (WSPACE) indicates the number of spaces available in the Write FIFO.

More information on the JTAG UART may be found in Chapter 5 of the *Altera Embedded Peripherals Handbook*.

In this exercise, we will use the JTAG UART to transfer ASCII-encoded characters between a Nios II processor implemented on the DE-series board and the host computer. We will also make use of an "interval timer" circuit to provide fixed delays. A block diagram of the hardware that we want use is shown in Figure 3. This hardware is included in the DE-series Basic Computer. Check the appropriate document *Basic Computer System for the Altera DE-series Board* for your board to see how the system in Figure 3 fits into the complete DE-series Basic Computer.

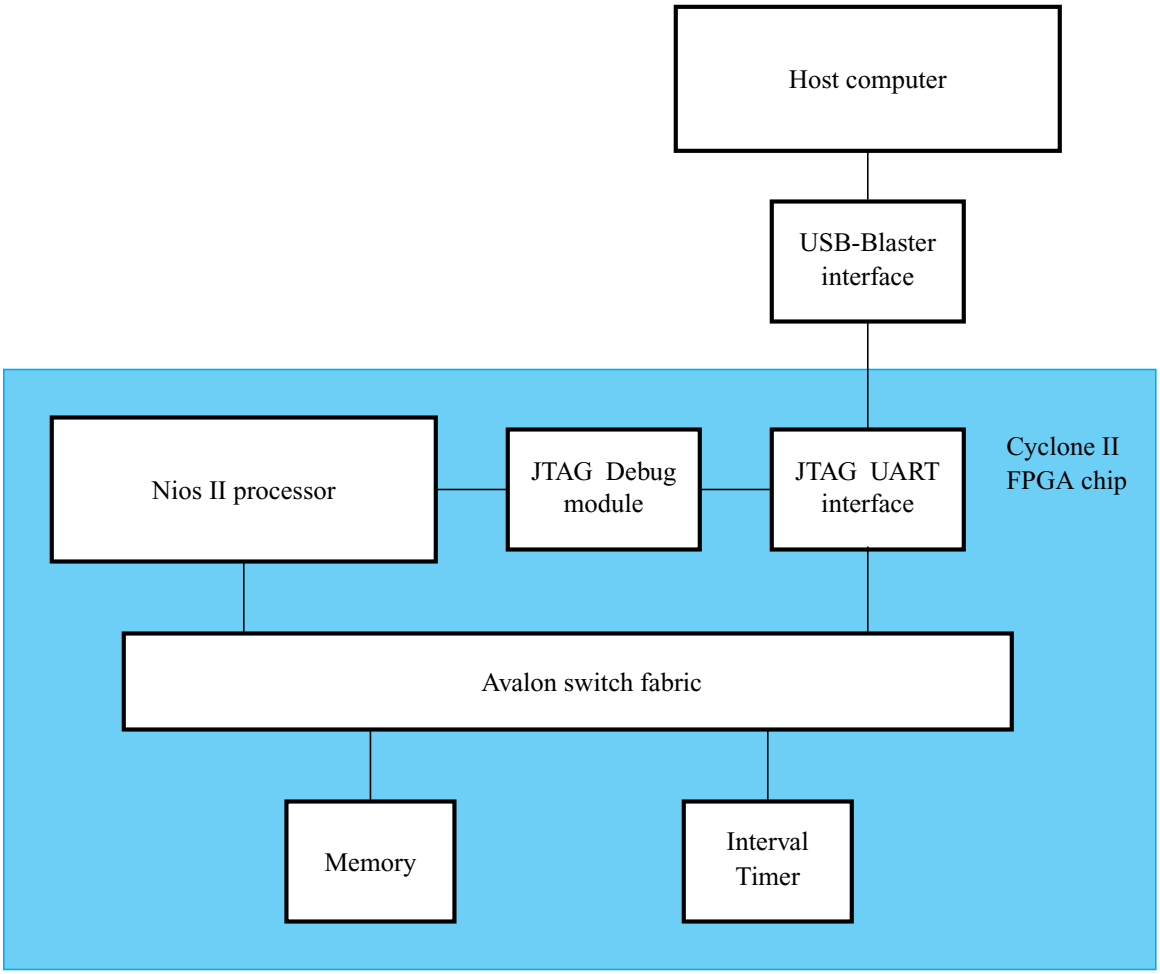

Figure 3. The desired Nios II system shown on a Cyclone II FPGA chip

Part I

The JTAG UART can send ASCII characters to the Altera Monitor Program, which will display these characters in its terminal window. When the WSPACE field in the *Control* register of the JTAG UART has a non-zero value, the JTAG UART can accept a new character to be written to the Altera Monitor Program. To write a character to the Monitor Program, poll (continuously read) this register until space is available. Once space is available the ASCII character can be written into the *Data* register of the JTAG UART.

Write a Nios II assembly-language program to display the letter Z approximately every half second in the terminal window of the Monitor Program. Create and execute the program as follows:

1. Using the Nios II assembly language, write a loop which reads the *Control* register in the JTAG UART and keeps looping until there is some write space available.
2. Write the letter Z to the *Data* register.
3. Using the Monitor Program, compile and load the assembly-language program.
4. Run this program using the single step feature only. If you run this program using the Continue mode, the character will be sent to the terminal window faster than the Monitor Program can handle.
5. In the assembly-language code, create a delay loop so that characters are only printed approximately every half second.

6. Recompile, load and run the program.

Part II

The JTAG UART can receive ASCII characters from the terminal window, as well as write them. The RVALID bit, b_{15}, in the *Data* register indicates whether or not a value in the DATA field is a valid received ASCII character. If more characters are still waiting to be read, the RAVAIL field will have a non-zero value.

Write a program that implements a "typewriter-like" task; that is, read each character that is received by the JTAG UART from the host computer and then display this character in the terminal window of the Monitor Program. Use polling to determine if a new character is available from the JTAG UART.

Note: the cursor must be in the terminal window of the Monitor Program to write characters to the JTAG UART's receive port.

Part III

Polling the JTAG UART is inefficient, due to the overhead of reading its registers to determine the UART's state. The overhead of determining if a new character is available significantly impacts the performance of the program. Instead of polling, it is possible to use the interrupt mechanism, which allows the processor to do useful work while it is waiting for an I/O transfer to take place.

Create an interrupt-service routine to read characters recieved by the JTAG UART from the host computer. Place the interrupt-service routine at the hex address 0x20, which is the location assigned for the *exception handler* in the DE-series Basic Computer. The exception return address in the *ea* register must be decremented by 4 for external interrupts.

Nios II's control register *ctl3*, also referred to as *ienable*, enables interrupts on an individual basis. In the DE-series Basic Computer the JTAG UART is assigned the interrupt level 8 (see the document *Basic Computer System for the Altera DE-series Board* for your board). This means that in the control register *ctl3*, bit $ctl3_8$ must be set to 1 to enable the JTAG UART's interrupts. In addition, control register *ctl0*, which is also referred to as *status*, must have its bit $ctl0_0$ set to 1. This is the processor interrupt-enable bit - setting it to 1 allows external interrupts to be accepted.

Perform the following:

1. Write an interrupt-service routine to read a character from the JTAG UART. Note that
   - The interrupt service routine must be placed at the memory address 0x20.
   - To enable interrupts, appropriate values must be written to the *Control* register of the JTAG UART, and the Nios II's control registers *ctl0* and *ctl3*.
2. In your interrupt service routine, use the polling approach to display the characters received from the host computer in the terminal window of the Monitor Program.
3. Compile, load and run your program.

If your program does not work at a first try, you will have to debug it and fix the errors. One aid in debugging is the single-step feature of the Altera Monitor Program, which allows the user to observe the flow of execution and the contents of Nios II registers as each instruction is being executed. However, this approach cannot be used when interrupts are involved, because interrupts are automatically disabled when single stepping through a program. Therefore, use breakpoints as a debugging aid.

Note also that interrupts are automatically disabled when the execution of an interrupt-service routine begins and re-enabled upon exit from this routine. This means that if some application requires nested interrupts, the

interrupts have to be re-enabled within the interrupt-service routine.

Part IV

In this part we wish to write a program that uses interrupts to read characters received by the JTAG UART from the host computer and displays the last character received repeatedly every 500 milliseconds. In Part I we used a delay loop to generate an approximate time interval of this length. Now, we want to use the Interval Timer circuit for this purpose. The Interval Timer should interrupt the processor every 500 ms at which point a character should be written to the Monitor Program's terminal window.

The Interval Timer has an internal counter which is set to a specified value and then decremented in each clock cycle. When the counter reaches 0, a "timeout" event is said to have occurred. At this point the Interval Timer can raise an interrupt request and the counter can be reset to the specified value. The Interval Timer has a set of 16-bit registers that can be accessed as memory locations, similar to the JTAG UART. These registers are shown in Figure 4. The address of the *Status* register is 0x10002000, which is the base address assigned to the Interval Timer. The *Control* register is at address 0x10002004. The starting value for the counter is specified in registers at addresses 0x10002008 (low-order 16 bits of the value) and 0x1000200C (high-order 16 bits of the value).

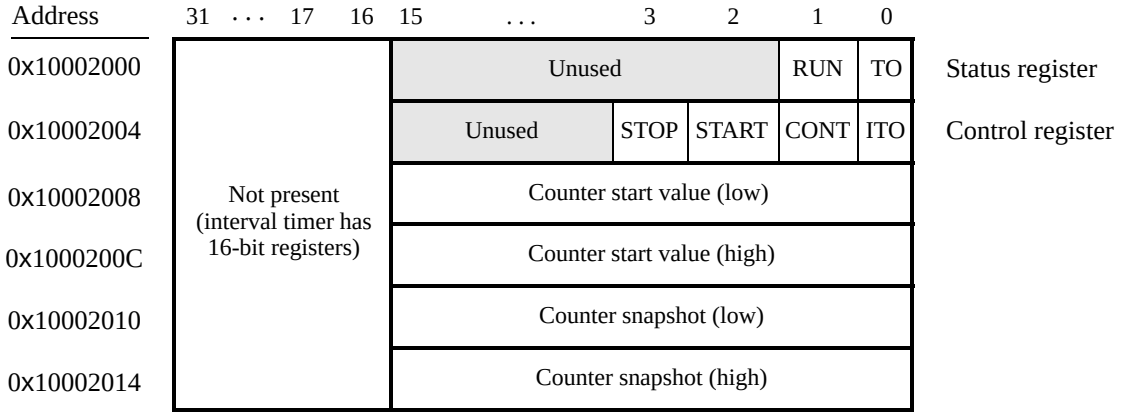

Figure 4. Registers in the Interval Timer

The bits in the *Status* register are used as follows:

- b_0 (TO) is the timeout bit. It is set to 1 when the internal counter in the Interval Timer reaches 0. It remains set until explicitly cleared by the processor writing a 0 to it, which must be done to clear an existing interrupt request.
- b_1 (RUN) is equal to 1 when the internal counter is running; otherwise, it is equal to 0. This bit is not changed by a write operation to the *Status* register.

The bits in the *Control* register are used as follows:

- b_0 (ITO) enables the Interval Timer interrupts when set to 1.
- b_1 (CONT) determines how the internal counter behaves when it reaches 0. If CONT = 1, the counter runs continuously by reloading the specified initial count value; otherwise, it stops when it reaches 0.
- b_2 (START) causes the internal counter to start running when set to 1 by a write operation.
- b_3 (STOP) stops the internal counter when set to 1 by a write operation.

More information on the Interval Timer may be found in chapter 12 of the *Altera Embedded Peripherals Handbook*.

To enable both interrupts, from the Interval Timer and the JTAG UART for reading characters (from Part III), the bits b_8 and b_0 of the control register *ctl3* must both be set to 1. The control register *ctl4*, also referred to as *ipending*, can be used to determine which interrupt has occurred. If an interrupt is disabled using the control register *ctl3*, it will not cause the interrupt-service routine to execute, nor will it show as being triggered in the control register *ctl4*, even if the device is driving its interrupt-request line to 1.

Perform the following steps:

1. Modify the program from Part III, so that the main program sets the desired count period in the Interval Timer, enables interrupts, and then waits in an infinite loop.
2. Modify the interrupt-service routine to handle both the Interval Timer and the JTAG UART's read interrupts.
3. To enable interrupts, appropriate values must be written to the *Control* register of the JTAG UART, the *Control* register of the Interval Timer and the Nios II control registers *ctl0* and *ctl3*.
4. Compile, load and run your program.

Preparation

Write the assembly-language programs for parts I to IV.