Lab 6: Serial Communication Using JTAG UART

Reading:

- Section 2.11 (JTAG UART) of the *DE10-Standard Computer System with ARM CORTEX-A9* Manual.

Introduction

In this lab exercise, you will develop C programs to interact with the JTAG UART Port. A simple finite state machine (FSM) structure is recommended for the program control flow.

The JTAG UART, shown in Fig. 1, is an input/output port that facilitates serial UART communication between the ARM system on the DE10 board and the host PC, through a JTAG cable such as the Intel USB Blaster. It has two memory-mapped registers shown in Fig. 2. Figs. 3 & 4 describe the purpose of each bit field of these two registers. Characters are encoded in ASCII format.

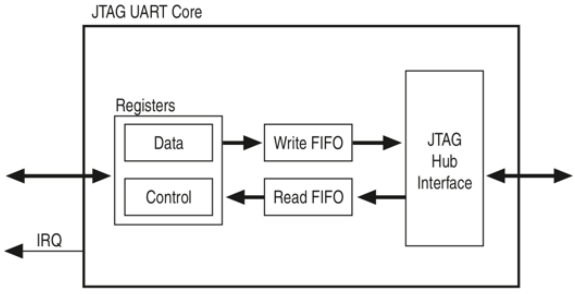

Fig. 1. Intel JTAG UART Port

Address	31 · · · 16	15	14 · · · 11	10	9	8	7 · · · 1	0	
0xFF201000	RAVAIL	RVALID	Unused				DATA		Data register
0xFF201004	WSPACE	Unused		AC	WI	RI		WE	RE	Control register

Fig. 2: DE10 JTAG UART Registers

Bit(s)	Name	Access	Description
[7:0]	DATA	R/W	The value to transfer to/from the JTAG core. When writing, the DATA field holds a character to be written to the write FIFO. When reading, the DATA field holds a character read from the read FIFO.
[15]	RVALID	R	Indicates whether the DATA field is valid. If RVALID=1, the DATA field is valid, otherwise DATA is undefined.
[31:16]	RAVAIL	R	The number of characters remaining in the read FIFO (after the current read).

Fig. 3: DE10 JTAG UART **Data Register**

Bit(s)	Name	Access	Description
0	RE	R/W	Interrupt-enable bit for read interrupts.
1	WE	R/W	Interrupt-enable bit for write interrupts.
8	RI	R	Indicates that the read interrupt is pending.
9	WI	R	Indicates that the write interrupt is pending.
10	AC	R/C	Indicates that there has been JTAG activity since the bit was cleared. Writing 1 to AC clears it to 0.
[31:16]	WSPACE	R	The number of spaces available in the write FIFO.

Fig. 4: DE10 JTAG UART **Control Register**

Assignment: JTAG UART String Processing

In this exercise, you are tasked with writing a C program to get a string from the JTAG UART, to extract and save all the digit characters (0-9) in the string, and then to send the saved digit string to the JTAG UART port. You may not use printf(). All characters are written to and read from the JTAG registers.

The Terminal window of the Intel Monitor Program establishes a link to the JTAG UART port running on the DE10 board. You can send characters to the JTAG UART port by typing them in the Terminal. Characters sent to the JTAG UART will be shown in the Terminal when echoed back with the put_jtag function.

Specifically, perform the following steps:

1. Download and unzip Lab6.zip from the Blackboard. Rename the included lab6_template.c to lab6.c. Make a Monitor project with lab6.c in the folder. On the "Specify system parameters" window, make sure you select "JTAG_UART_for_ARM_0" instead of the default "semihosting" for the Terminal device. Write your name in a comment at the top of the program. The lab6_template.c provides a basic program structure for your reference. The majority of code is for you to work out. Type in the two functions put_jtag() and get_jtag() you learned from the lecture. Read the comments (suggestions) in the code.
2. Use the put_jtag() function to create a function called put_string(char* text_string), which sends a char array text_string or a string of characters included in double quotes to the JTAG UART port. Use this function in your steps below whenever you need to send a string to the JTAG Port. Test it in the Monitor program on the board by sending the string "\n Lab 6 by <*Your name*>. \n".
   - A null character \0 is appended to the string by the compiler by default, which can be used to check the end of a string.
3. Read a string from the JTAG UART and extract digit characters into an array called digit[]. Use a simple state machine to control the flow. A simple FSM structure is included in

lab6_template.c. The comments are suggestions only; You are free to develop your own way of using these states to finish the required work below. Add your code in each state for the following functions.

- Send a string "Press Esc to start:\n" first to prompt the user to start.
- Wait for Esc to be pressed to start getting characters from JTAG data register, and continue getting any characters entered in the terminal until an Enter is received.
- For every character you get, check if it is a digit. In ASCII formatting, digits 0-9 are represented by the number+0x30. If the character is a digit, save the digit to an array called digit[].
- Use the put_jtag () function to send the digits stored in digit[] to the JTAG UART.
- In all states, check if the received character is a '\0'. If so, break that case.

4. Capture a picture of the Monitor program with the Terminal window showing the entire process with the strings from Steps 2 to 4. Save it to the Monitor project folder.
5. **Optional:** You may want to use LEDRs to show the status of the code to help you debug.

What to submit:

Zip the entire Monitor project folder, including the C program, project file, makefile, executable code, and the captured picture in Step 4. Append your last name to the zip file and submit it in the Blackboard.