第十七集

内置 IP 核之 DMA 的理论与实战讲解

§1.16 JTAG_UART IP核

1.16.1 JTAG_UART IP 核的综述

介绍完了我们的DMA IP核，接下来我们就来介绍一下我们的JTAG_UART IP核。带Avalon接口的 JTAG_UART 设备实现 PC 和 Nios II 系统间的串行通信。在许多设计中，JTAG_UART 常取代RS-232通信设备，用于字符的输入和输出。JTAG UART 核为用户提供了简单的Avalon-MM 接口映射，简化了 JTAG 接口的复杂性。Nios II 处理器通过读、写 Avalon-MM 接口映射寄存器实现与 JTAG_UART 核的数据交换。JTAG_UART 的系统框图如图 1.169 所示。

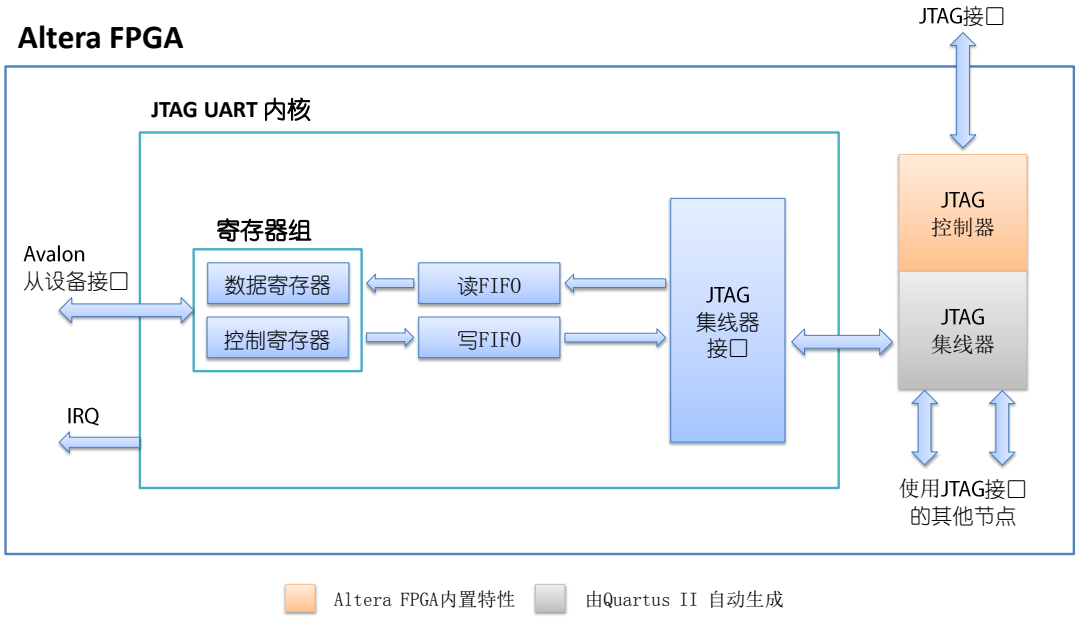

图 1.169 JTAG_UART 的结构框图

从该图中我们可以看出，JTAG_UART 内核通过 Avalon 从控制器接口连接到 Avalon 总线。JTAG_UART 核包含 2 个 32 位寄存器，即数据寄存器和控制寄存器。Nios II 处理器可以通过这两个寄存器与 JTAG_UART 进行数据传输。为了提高 JTAG 连接的传输带宽，JTAG_UART 核还提供了双 FIFO 结构，即读 FIFO 和写 FIFO，FIFO 的存储深度是可以通过编程进行配置的。FIFO 可以用 FPGA 内部的存储器块资源实现，也可以用寄存器资源实现。

1.16.2 JTAG_UART IP 核的寄存器描述

介绍完了 JTAG_UART IP 核的综述，接下来我们再来说一说 JTAG_UART IP 核的寄存器描述。JTAG_UART IP 核的寄存器描述如表 1.24 所示。

表 1.24 JTAG_UART 内核寄存器映射

偏移量	寄存器名称	操作	位描述									
			31…16	15	14	…	10	9	8	7…2	1	0
0	data	读/写	RAVAIL	RVLID	保留					DATA		
1	control	读/写	WSPACE	保留			AC	WI	RI	保留	W	R

从 JTAG_UART IP 核寄存器的描述表格中我们可以看出，JTAG_UART 核共有 2 个 32 位寄存器，Nios II 处理器可以通过这两个寄存器与 JTAG_UART 进行通信。下面我们就对这 2 个 32 寄存器分别进行介绍：

(1) 数据寄存器(Data)

软件通过数据寄存器访问读/写 FIFO。表 1.25 描述数据寄存器每个位的功能。

表 1.25 数据寄存器位

位	名称	操作	描述
0～7	DATA	读/写	传输到或来自 JTAG 内核的值。写操作时，DATA 字段是被写入写 FIFO 的字符。读操作时，DATA 字段是从读 FIFO 中读取的字符
15	RVALID	读	指示 DATA 字段是否有效。如果 RVALID=1，那么 DATA 字段有效，否则 DATA 未定义
16～32	RAVAIL	读	在读 FIFO 中剩余的字符数（读以后）

读数据寄存器获得DATA字段和RAVAIL字段的数据。数据寄存器中仅有DATA字段是可以写的，在执行写数据寄存器的操作时，如果 FIFO 处于满的状态，那么字符将被丢弃。

(2) 控制寄存器(Control)

Nios II处理器可以通过读/写控制寄存器实现对JTAG_UART核的各种控制。读控制寄存器返回读/写 FIFO 的状态，写控制寄存器可以使能禁止中断或者清零 AC 位，表 1.26 为控制寄存器各位的意义。

表 1.26 控制寄存器位

位	名称	操作	描述
0	RE	读/写	读中断的中断使能位
1	WE	读/写	写中断的中断使能位
8	RI	读	表示读中断正在等待
9	WI	读	表示写中断正在等待
10	AC	读/清除	表示自从 AC 位清零后就有 JTAG 活动。写 1 到 AC 将其清零
16～32	WSPACE	读	在写 FIFO 中可用的空间数

RE 和 WE 位分别使能读/写 FIFO 的中断。WI 和 RI 位表示中断源的状态，受中断使能位（WE 和 RE）的限制。软件可通过检查 RI 和 WI 来确定在什么条件下会产生 IRQ。AC 为表示主控制器 PC 上的应用程序已通过 JTAG 接口查询 JTAG_UART 内核。一旦置位，AC 位保持置位直到被清零。写 1 到 AC 位可将其清零。软件可检查 AC 位来确定与 PC 的连接是否存在。

如果连接不存在，软件可选择忽略 JTAG 数据流。当主控制器 PC 没有数据传输时，它可选择每秒查询 JTAG_UART 内核一次。

1.16.3 JTAG_UART IP 核的配置选项

介绍完了我们的 JTAG_UART IP 核的寄存器描述，接下来我们再来看一看 JTAG_UART IP 核在 Qsys 中的配置选项，图 1.170 给出了 JTAG_UART IP 核的配置选项图。

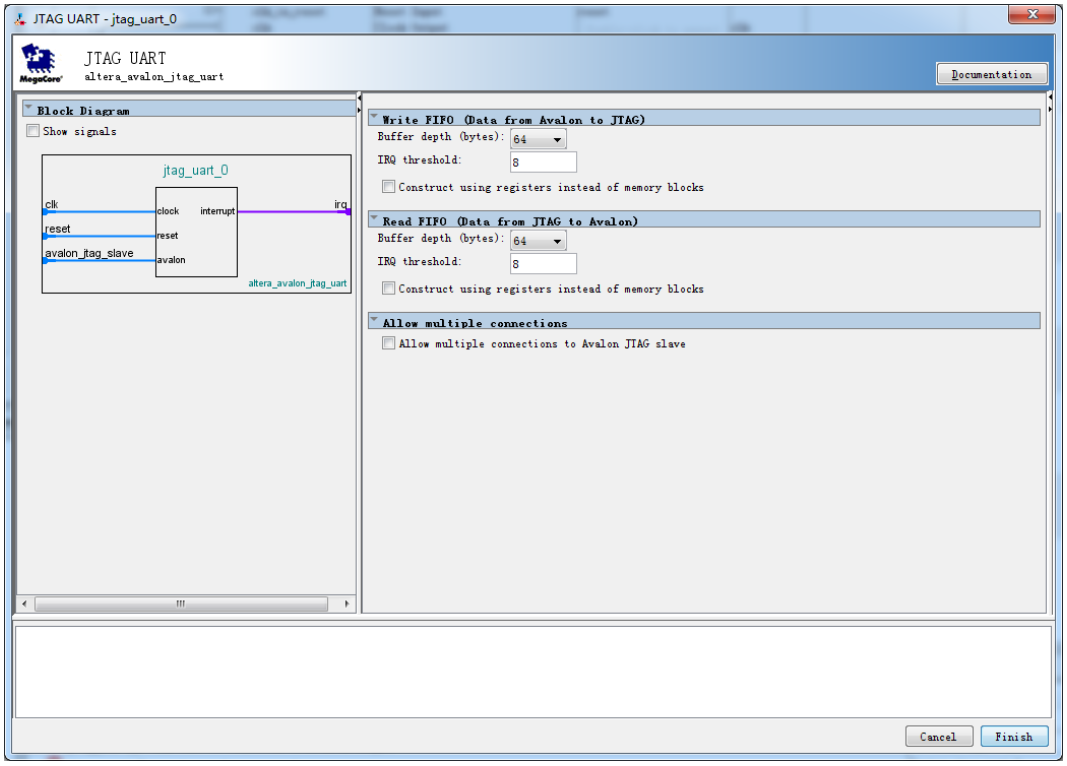

图 1.170 JTAG_UART IP 核的配置选项图

从 JTAG_UART IP 核的配置选项图中我们可以看出，JTAG_UART IP 核有三个选项可以设置，下面我们就对这三个选项分别进行介绍：

(1) 写 FIFO（Write FIFO）

写 FIFO 缓冲区使数据流从 Avalon 接口到主机。各设置的意义为：

- Buffer Depth：写缓冲区 FIFO 的深度可以设置为 8~32768 字节。深度越深，消耗越多的片上存储器资源。深度为 64 时具有最佳的性能，这可以满足大多数应用。这里需要我们注意的是，深度值必须为 2 的 n 次方。例如可以是 8、16、32 等值，但不能是 15、18、50 等类似的数值。
- IRQ Threshold：写 IRQ 阈值控制内核如何提交其 IRQ 来响应 FIFO 清空。在 JTAG 清空写 FIFO 中的数据时，若 FIFO 中剩余的字节数达到该阈值，则 JTAG_UART 核产生中断请求 IRQ 信号。为了得到最有效的传输带宽，处理器应该防止 FIFO 完全清空才触发中断。8 通常是一个最佳的 IRQ 阈值。
- Construct using registers instead of memory blocks：如果打开该选项，那么可以节

省 FPGA 内部的存储器资源。存储一个字节数据大约消耗 11 个逻辑单元（LE）。

(2) 读 FIFO（Read FIFO）

- Buffer Depth：读缓冲区 FIFO 的深度可以设置为 8~32768 字节。深度越深，消耗越多的片上存储器资源。深度为 64 时具有最佳的性能，这可以满足大多数应用。这里需要我们注意的是，深度值必须为 2 的 n 次方。例如可以是 8、16、32 等值，但不能是 15、18、50 等类似的数值。
- IRQ Threshold：读 IRQ 阈值控制内核如何提交其 IRQ 来响应 FIFO 填满。在 JTAG 填满 FIFO 中的数据时，若 FIFO 中剩余的字节数达到该阈值，则 JTAG_UART 核产生中断请求 IRQ 信号。为了得到最有效的传输带宽，处理器应该防止 FIFO 完全填满才触发中断。8 通常是一个最佳的 IRQ 阈值。
- Construct using registers instead of memory blocks：如果打开该选项，那么可以节省 FPGA 内部的存储器资源。存储一个字节数据大约消耗 11 个逻辑单元（LE）。

(3) 允许多个连接（Allow multiple connections）

如果选择该选项，那么就可以连接多个 JTAG 外设，一般我们用不到。

1.16.4 JTAG_UART IP 核的软件编程

介绍完了 JTAG_UART IP 核的配置选项，接下来我们再来看看 JTAG_UART IP 核的软件编程，JTAG_UART IP 核提供了定义硬件的底层接口头文件和HAL驱动驱动程序文件，文件如下所示：

(1) altera_avalon_jtag_uart_regs.h：定义内核的额寄存器映射，提供宏定义来访问底层硬件。该文件中的符号仅由设备驱动程序函数使用。

(2) altera_avalon_jtag_uart.h，altera_avalon_jtag_uart.c：实现 HAL 系统库设备驱动程序。

我们可以通过阅读上述文件来熟悉JTAG_UART IP核的软件访问方法，我们这里建议大家不要修改该文件。

1.16.5 JTAG_UART IP 核的应用实例——C 函数

(1) 实验目的

介绍完了 JTAG_UART IP 核的软件编程，接下来我们再来看看 JTAG_UART IP 核的应用实例，该实验主要利用 ANSI C 标准函数从 JTAG_UART 中读入字符，如果检测到符合要求的字符，则输出提示信息。在这个实验中，我们将学习 printf()、scanf()、fopen()和 fwrite()等函数的用法，以及掌握使用 ANSI C 标准函数访问 JTAG_UART 的方法。

(2) 硬件框架

讲完了实验目的，接下来我们就来讲解硬件框架，该实验的硬件框架，如图 1.171 所示。

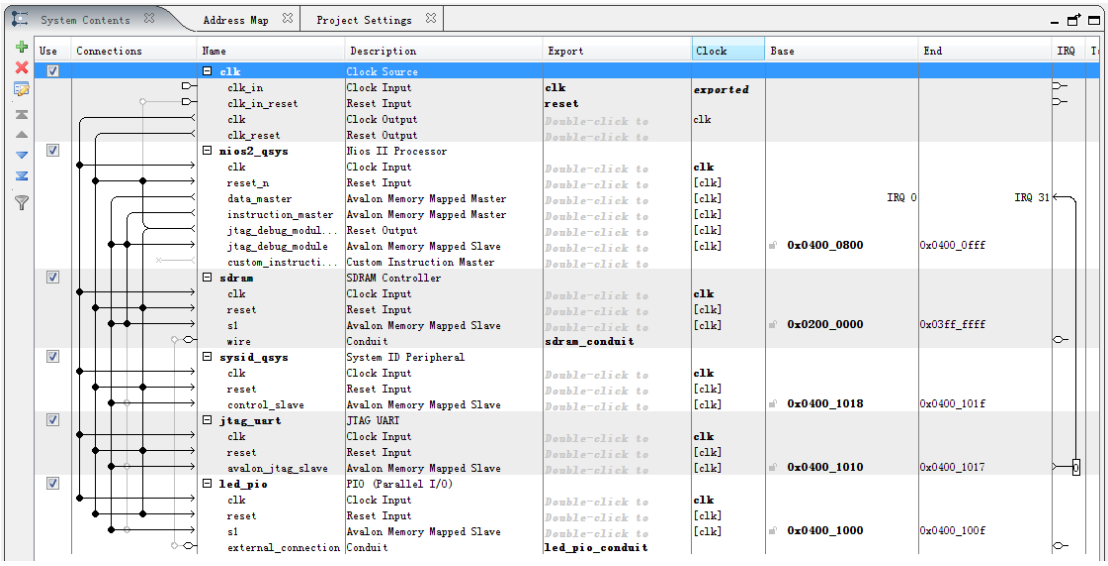

图 1.171 JTAG_UART C 函数的硬件框架图

从该图中我们可以看出，我们JTAG_UART实验的硬件框架与System ID实验的硬件框架是一样的，我们没有添加任何 IP 核，也没有更改任何配置。我们这里的 JTAG_UART 使用的是默认配置。

(3) 软件工程

讲完了硬件框架，接下来我们就来讲解软件工程，该实验的软件工程代码，如代码 1.17 所示。

代码 1.17 Qsys_Jtag_Uart_C.c 代码

```
//-----------------------------------------------------------------------------
//-- 文件名    :  Qsys_Jtag_Uart_C.c
//-- 描述      :  如果发现字符 t，则打印信息到控制台；如果发现字符 v，则退出程序；其它字符忽略
//-- 修订历史  :  2014-1-1
//-- 作者      :  Zircon Opto-Electronic Technology CO.,Ltd.
//-----------------------------------------------------------------------------
#include <stdio.h>
#include <string.h>
#include "system.h"
#include "alt_types.h"

//-----------------------------------------------------------------------------
//-- 名称      :  main()
//-- 功能      :  程序入口
//-- 输入参数  :  无
//-- 输出参数  :  无
//-----------------------------------------------------------------------------
int main (void)
{
   FILE* fp;
```

```c
    char prompt = 0;
    char* msg = "Discover the character 't'.\n";

    printf("Please write character and press Enter... \n");

    fp = fopen(JTAG_UART_NAME, "r+"); //以可读写方式打开设备文件
    if(fp) //成功打开设备文件
    {
      while(prompt != 't')    //循环直至接收到 't'
      {
        prompt = getc(fp);   //从 JTAG UART 中获取字符
        if(prompt == 't')
        {     //如果字符为 't' 则打印 msg 中的信息
          fwrite(msg, strlen (msg), 1, fp);
        }
        if(ferror(fp))
        {
          clearerr(fp); //检查错误并清除错误
        }
      }
      fprintf(fp, "Closing the JTAG UART file handle.\n");
      fclose(fp); //关闭设备文件
    }
    else
    {
      printf("Fail to open file...\n"); //设备文件打开失败
    }
    return 0;
}
```

下面我们就来给大家讲解一下该程序中的关键点。在ANSI C库函数的支持下，我们既可把JTAG_UART 设备当作标准输入/输出设备使用，也可当作文件操作。其实质是通过 ANSI C 库函数调用 JTAG_UART 设备驱动函数访问硬件设备。在其他处理器的集成开发环境中，若开发工具没有提供这项支持，则不能这样访问硬件设备。ferror(fp)用于检查错误是否在JTAG_UART连接上出现，例如JTAG连接断开。如果驱动程序检测到JTAG连接被断开，则报导错误(EIO)并丢弃数据以便后续传输。如果这种错误已经出现，C函数库锁存该错误信息直至使用clearerr()函数将其清除。

(4) 板级调试

讲完了软件工程，接下来我们就将该实验下载至我们的A4开发板进行验证，首先我们需要在 Quartus II 软件中将 Qsys_Jtag_Uart_C.sof 下载至我们的 A4 开发板，Qsys_Jtag_Uart_C.sof 下载完成后，我们还需要在Eclipse软件中将Qsys_Jtag_Uart_C.elf文件下载至我们的 A4 开发板，Qsys_Jtag_Uart_C.elf 下载完成以后，我们的 C 程序将会执行在我们的 A4 开发板上，此时，我们可以在 Eclipse 软件的控制台中看到我们的打印信息，如图

1.172 所示。

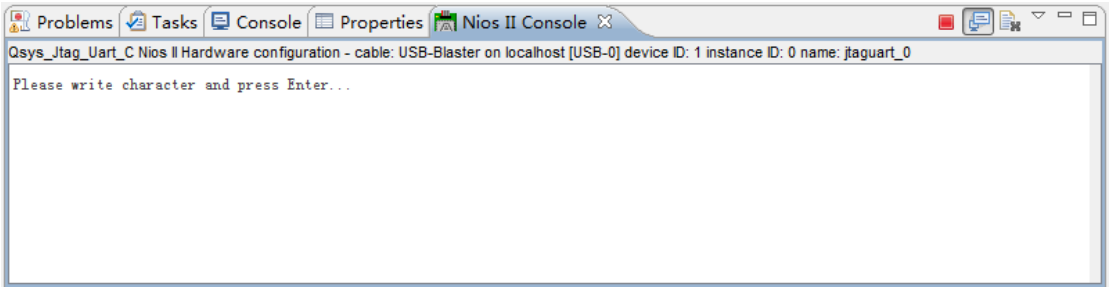

图 1.172 JTAG_UART C 函数的控制台打印信息图

这时，我们在 Eclipse 软件的控制台中输入一串带有 t 字符的字符串，按下回车，然后我们再输入一串带有 v 字符的字符串，按下回车，控制台则会提示如图 1.173 所示内容。

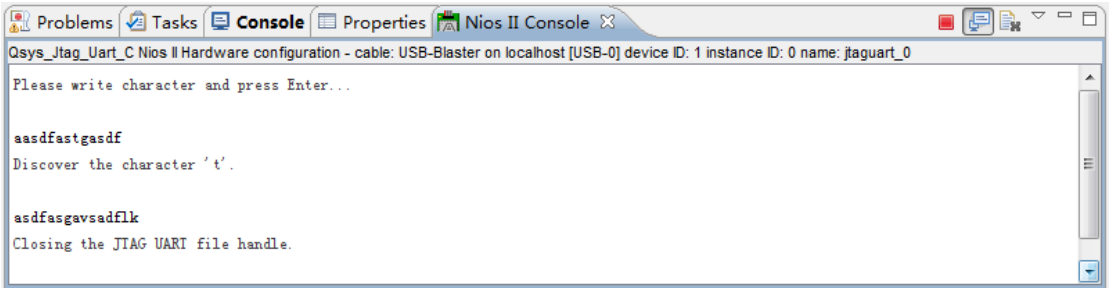

图 1.173 JTAG_UART C 函数的板级调试图

1.16.6 JTAG_UART IP 核的应用实例——API 函数

(1) 实验目的

介绍完了 JTAG_UART C 函数实验，接下来我们再来看看 JTAG_UART API 函数实验，该实验主要利用 HAL API 函数从 JTAG_UART 中读入和输出字符。在这个实例中，我们将学习 open()、close()、read()、write()和 lseek()等函数的用法，以及熟悉并掌握使用 HAL API 函数访问 JTAG_UART 设备的方法。

(2) 硬件框架

讲完了实验目的，接下来我们就来讲解硬件框架，该实验的硬件框架，如图 1.174 所示。

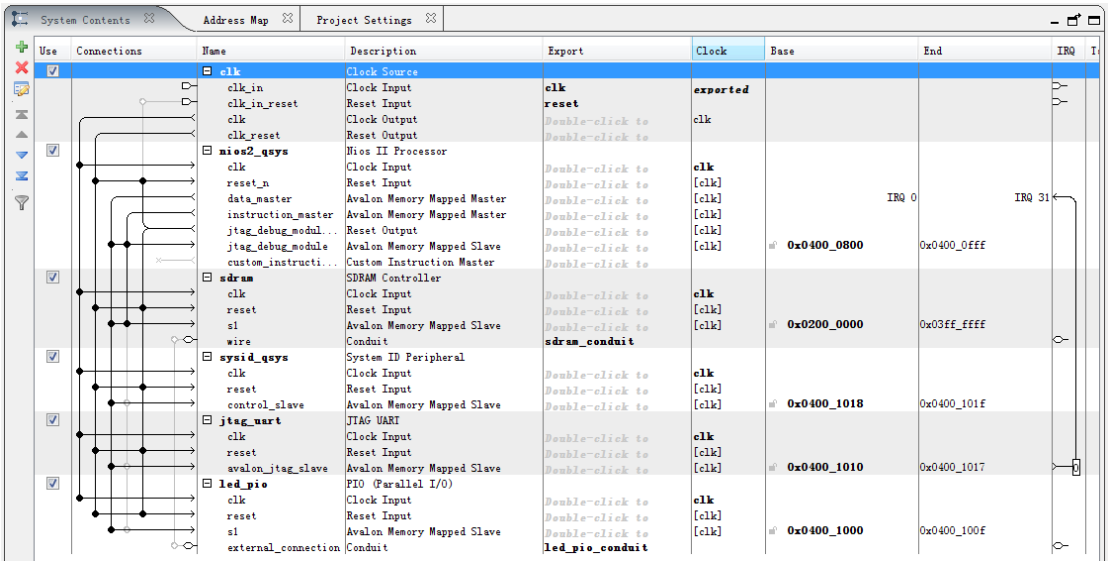

图 1.174 JTAG_UART API 函数的硬件框架图

从该图中我们可以看出，我们的 JTAG_UART API 函数实验的硬件框架与我们 JTAG_UART C 函数的硬件框架是一样的。

(3) 软件工程

讲完了硬件框架，接下来我们就来讲解软件工程，该实验的软件工程代码，如代码 1.18 所示。

代码 1.18 Qsys_Jtag_Uart_API.c 代码

```
//-----------------------------------------------------------------------------
//-- 文件名    :  Qsys_Jtag_Uart_API.c
//-- 描述      :  用 HAL API 函数来访问 JTAG UART；读入一串字符再输出。
//-- 修订历史  :  2014-1-1
//-- 作者      :  Zircon Opto-Electronic Technology CO.,Ltd.
//-----------------------------------------------------------------------------
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include "system.h"
#include "unistd.h"

//-----------------------------------------------------------------------------
//-- 名称      :  main()
//-- 功能      :  程序入口
//-- 输入参数  :  无
//-- 输出参数  :  无
//-----------------------------------------------------------------------------
int main(void)
{
   int fd,count;
```

```
22    char buf[100];
23    char *info = " too! ";
24    char *msg = "Please write the following characters:Nice to meet you \n";
25
26    fd = open(JTAG_UART_NAME, O_RDWR,0666); //以可读写方式打开设备文件
27    if (fd < 0)
28    {
29       printf("Fail to open file...\n ");//设备文件打开失败
30    }
31    else
32    {
33       write(fd,msg,strlen(msg));          //将 msg 中的数据,写到控制台中
34       count = read(fd,buf,16);            //将控制台中的数据,读到 buf 中
35       write(fd,buf,count);                //将 buf 中的数据,写到控制台中
36       write(fd,info,strlen(info));        //将 info 中的数据,写到控制台
37       close(fd);                          //关闭设备
38    }
39    return 0;
40 }
```

下面我们就来给大家讲解一下该程序中的关键点。Nios II的HAL层从宏观上把硬件抽象成3种类型：字符设备、块设备和网络设备。对于字符设备和块设备，HAL 层提供标准的 I/O 接口函数来访问，如图 1.175 所示。

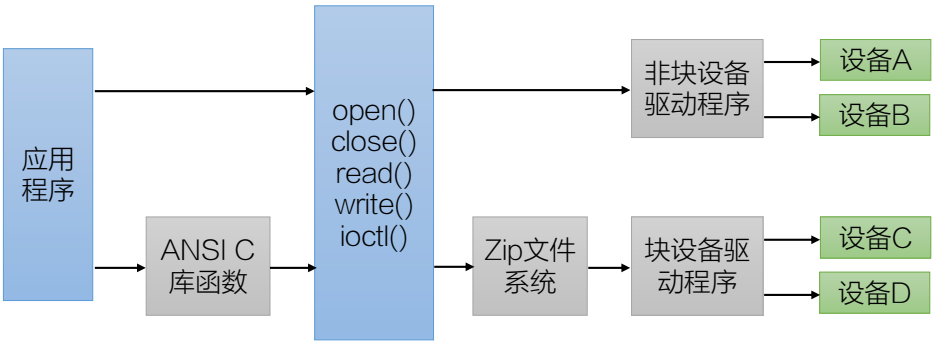

图 1.175 HAL 层 I/O 系统与设备驱动程序

从该图中我们可以看出，JTAG_UART 设备既可以通过 ANSI C 库函数访问，又可以通过 HAL API 函数访问。

(4) 板级调试

讲完了软件工程，接下来我们就将该实验下载至我们的A4开发板进行验证，首先我们需要在 Quartus II 软件中将 Qsys_Jtag_Uart_API.sof 下载至我们的 A4 开发板，Qsys_Jtag_Uart_API.sof下载完成后，我们还需要在Eclipse软件中将Qsys_Jtag_Uart_API.elf文件下载至我们的 A4 开发板，Qsys_Jtag_Uart_API.elf 下载完成以后，我们的 C 程序将会执行在我们的 A4 开发板上，此时，我们可以在 Eclipse 软件的控制台中看到我们的打印信息，如图 1.176 所示。

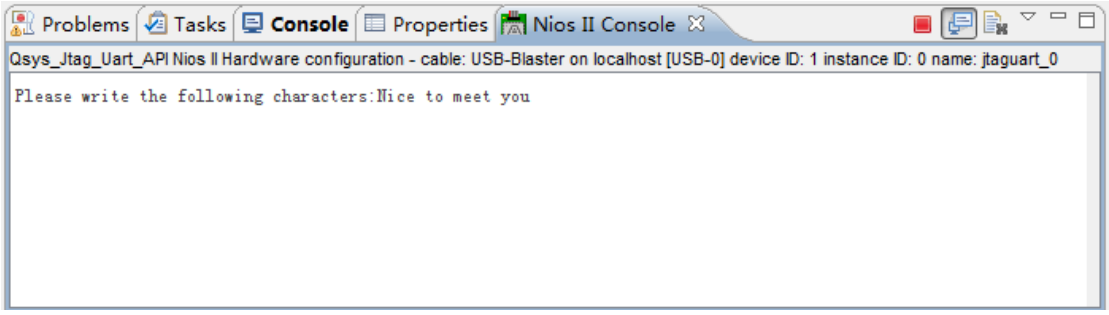

图 1.176 JTAG_UART API 函数的控制台打印信息图

这时，我们按照上面的提示在 Eclipse 软件的控制台中输入 Nice to meet you，按下回车，控制台则会弹出如图 1.177 所示内容。

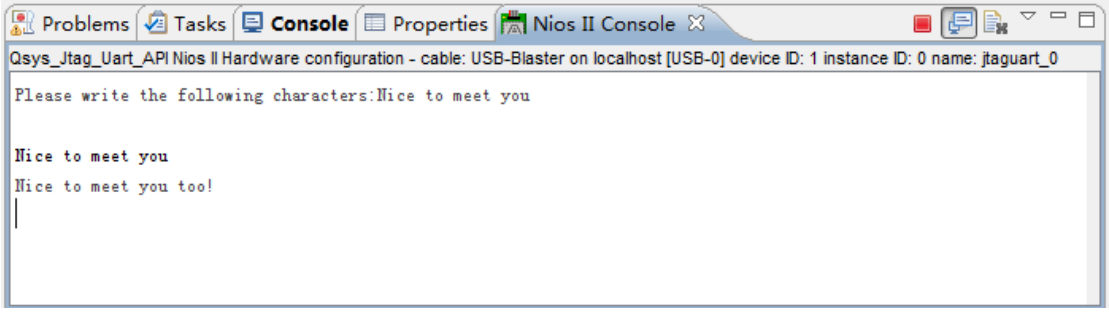

图 1.177 JTAG_UART API 函数的板级调试图