第十集

内置 IP 核之 PIO 的实战应用讲解

1.10.5 PIO IP 核的应用实例——流水

介绍完了 PIO IP 核的软件编程，接下来我们再来看看 PIO IP 核的应用实例，对于应用实例的讲解，我们这里需要说明的是，我们将会从以下四个方面进行讲解：

(1) 实验目的：该方面主要是用来讲解，我们的实验将要完成一个怎样的功能，我们从该实验中将会学到什么知识。

(2) 硬件框架：该方面主要是用来讲解，如何构建一个 Qsys 系统，以及我们构建该实验都用到了哪些 IP 核。

(3) 软件工程：该方面主要是用来讲解，如何通过 C 语言程序来完成我们的设计，实现我们的功能。

(4) 板级调试：该方面主要是用来验证，我们的实验是否正确，以及能否在锆石 A4 开发板上正常运行。

(1) 实验目的

首先我们讲解的是第一部分实验目的，该实验主要是利用 PIO IP 核来控制 8 个 LED 进行流水灯显示，在这个实验中，我们将学习到 PIO IP 核在 Qsys 软件中是如何进行配置的，以及我们如何利用 C 语言程序来通过 PIO 输出数据。

(2) 硬件框架

由于之前我们已经给出了详细的工程建立步骤，那么这里我们就不再给出大量的重复步骤，大家只需要根据我们给出的关键性步骤，便能创建出可以用于实验的工程。如果此时你对Qsys软件创建流程还不是很熟悉，那么你可以参考该篇第2章中介绍的内容。下面我们便给出该实验的硬件框架，如图 1.121 所示。

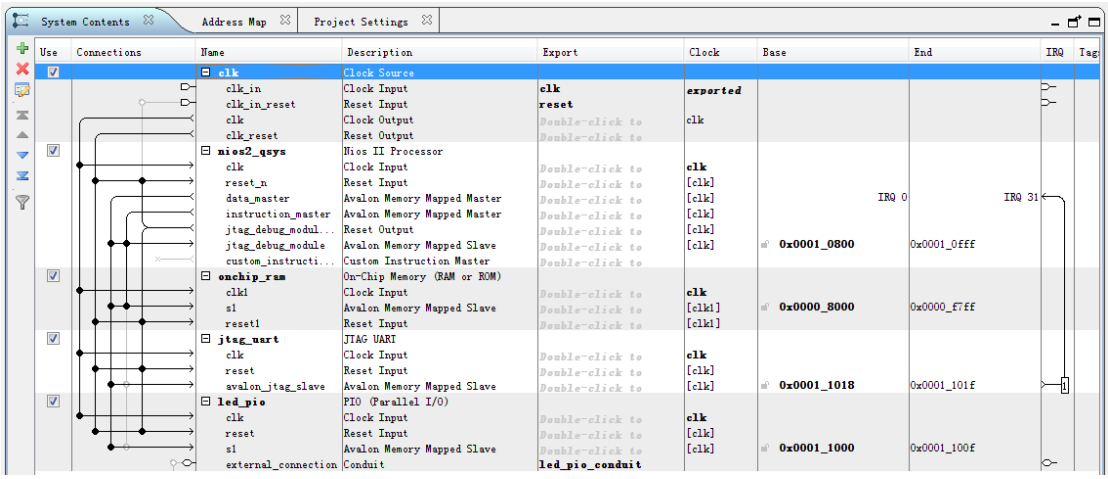

图 1.121 LED 流水灯实验的硬件框架图

从该图中我们可以看出，我们使用了 Nios、Onchip_Memory、Jtag_Uart 和 PIO 四个 IP

核，对于这四个 IP 核，相信大家都不会陌生，我们在之前工程中都有用过，这里我们就不再给出它们的详细配置图。下面我们就对它们的配置进行一个简单的讲解，在这四个 IP 核中，我们 Nios 配置使用的是 Nios II/f 等级；onchip_memory 配置使用的是单端口 RAM，RAM 的大小为 30KB。jtag_uart 配置使用的是默认配置；pio 配置使用的是输出端口，数据宽度为 8。这里要注意的是，由于我们这里没有用到 onchip_rom，只用了一个 onchip_ram，所以读者在设置 Reset Vector 和 Exception Vector 的时候全都选择 onchip_ram 即可。

(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1.6所示。

代码 1.6 Qsys_Led.c 代码

```
//-----------------------------------------------------------------------------
//-- 文件名    :  Qsys_Led.c
//-- 描述      :  通过 PIO 直接控制 8 个 Led 产生流水灯效果
//-- 修订历史  :  2014-1-1
//-- 作者      :  Zircon Opto-Electronic Technology CO.,Ltd.
//-----------------------------------------------------------------------------
#include "system.h"                //系统头文件
#include "unistd.h"                //延迟函数头文件
#include "alt_types.h"             //数据类型头文件
#include "altera_avalon_pio_regs.h"//pio 寄存器头文件

/* 流水灯花样，Led 低电平亮,高电平灭  */
alt_u32 LED_TBL[] = {
   0xFF, 0x00,                                          // 全部熄灭后，再全部点亮
   0xFE, 0xFD, 0xFB, 0xF7, 0xEF, 0xDF, 0xBF, 0x7F,   // 依次逐个点亮
   0xFE, 0xFC, 0xF8, 0xF0, 0xE0, 0xC0, 0x80, 0x00,   // 依次逐个叠加
   0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF8, 0xFC, 0xFE,   // 依次逐个递减
   0x7E, 0xBD, 0xDB, 0xE7, 0xE7, 0xDB, 0xBD, 0x7E,   // 两个靠拢后分开
   0x7E, 0x3C, 0x18, 0x00, 0x00, 0x18, 0x3C, 0x7E    // 从两边叠加后递减
};

//-----------------------------------------------------------------------------
//-- 名称      :  main()
//-- 功能      :  程序入口
//-- 输入参数  :  无
//-- 输出参数  :  无
//-----------------------------------------------------------------------------
int main (void)
{
   alt_u8 i;

   while(1)
   {
```

```
34        for(i=0; i<=41; i++)
35        {
36          //将 LED_TBL 数组中的数值依次赋值给 Led 进行显示
37          IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, LED_TBL[i]);
38          usleep(500000);  //等待 100ms
39        }
40      }
41      return 0;
42  }
```

由于该代码比较简单，并且我们也给出了详细的注释，所以这里我们就不再一句一句的讲解代码了，我们下面主要为大家讲解代码中的一些关键部分。

(1) 为什么要包含system.h文件？这主要是因为system.h头文件是系统硬件信息的宏定义文件，程序中 LED_PIO_BASE 就是从该文件中获取的。

(2) altera_avalon_pio_regs.h寄存器头文件一般有什么用？altera_avalon_pio_regs.h头文件提供PIO内核寄存器访问宏定义，该文件路径为C:\altera\13.1\ip\altera\sopc_builder_ip\altera_avalon_pio\inc\ altera_avalon_pio_regs.h。程序中对 I/O 口操作的宏定义都在这个文件中给出。

(3) altera_types.h头文件的作用是什么？altera_types.h头文件定义了与Nios Ⅱ相关的数据类型，它的路径为 C:\altera\13.1\nios-2eds\components\altera_nios2\HAL\inc\alt_types.h。

(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_Led.sof 下载至我们的 A4 开发板，Qsys_Led.sof 下载完成后，我们还需要在 Eclipse 软件中将 Qsys_Led.elf 文件下载至我们的 A4 开发板，Qsys_Led.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，我们可以看到 A4 开发板上的Led 全部熄灭后，再全部点亮，依次逐个点亮，依次逐个叠加，依次逐个递减，两个靠拢后分开，从两边叠加后递减等等各种流水花样。由于我们的图片不能显示动态效果，所以我们这里就给出一个 LED 全部点亮的图片，如图 1.122 所示。

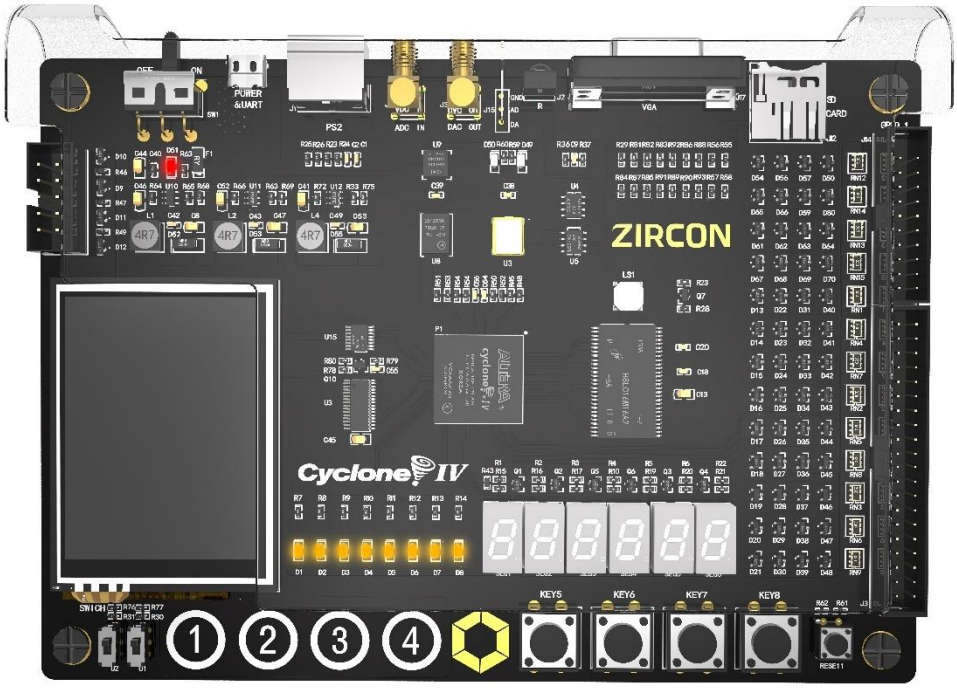

图 1.122 LED 流水灯实验的板级调试图

1.10.6 PIO IP 核的应用实例——按键

(1) 实验目的

介绍完了 PIO 流水实验，接下来我们再来看看 PIO 按键实验，对于 PIO 按键实验，我们将和前面一样，也是从四个方面进行讲解。首先我们讲解的是第一个方面实验目的。该实验主要是利用 PIO IP 核读取按键的值，然后将读到的值输出到 LED 上进行显示。在这个实验中，我们将学习到 PIO IP 核在 Qsys 软件中是如何进行配置，以及我们如何利用 C 语言程序来通过 PIO 读取数据。

(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 1.123 所示。

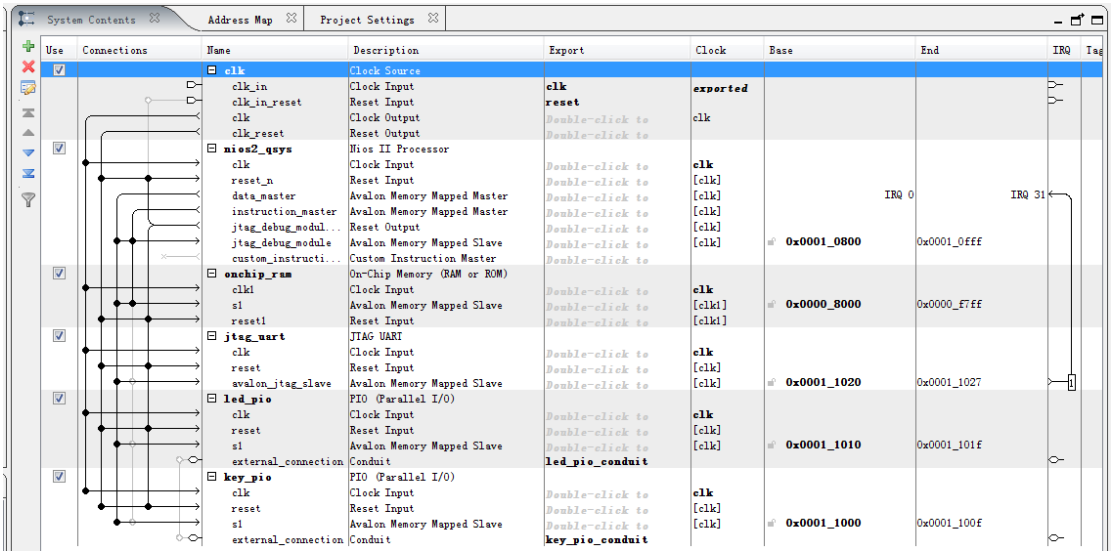

图 1.123 按键实验的硬件框架图

从该图中我们可以看出，我们的 PIO 按键实验在 PIO 流水实验的基础上多添加了一个 PIO IP 核，该 PIO 的配置我们使用的是输入端口，数据宽度为 8，至于其他的 IP 核配置，我们这里和 PIO 流水实验中的配置是一样的，没有做任何改动。这里需要注意的是，我们的按键没有连接内部中断。

(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1.7所示。

代码 1.7 Qsys_Key.c 代码

```
//-----------------------------------------------------------------------------
//-- 文件名    :  Qsys_Key.c
//-- 描述      :  按下不同的按键，点亮相对应的 Led。
//-- 修订历史  :  2014-1-1
//-- 作者      :  Zircon Opto-Electronic Technology CO.,Ltd.
//-----------------------------------------------------------------------------
#include "system.h"                //系统头文件
#include "alt_types.h"             //数据类型头文件
#include "altera_avalon_pio_regs.h"//pio 寄存器头文件

//-----------------------------------------------------------------------------
//-- 名称      :  main()
//-- 功能      :  程序入口
//-- 输入参数  :  无
//-- 输出参数  :  无
//-----------------------------------------------------------------------------
int  main(void)
{
   alt_u32 key_state,led_state;   //Key 和 Led 缓存变量

   while(1)
   {
      //读取按键的值，并赋值给 key_state。
      key_state = IORD_ALTERA_AVALON_PIO_DATA(KEY_PIO_BASE);
      //注意：Key 悬空为低电平，按下为高电平。Led 低电平亮，高电平灭。
      //为了保证按下不同按键点亮相对应 Led，因此，我们将按键的值取反后，再赋值给 led_state。
      led_state = ~key_state;
      //将 led_state 中的值输出到 Led 上进行显示。
      IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, led_state);
   }

   return(0);
}
```

(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_Key.sof 下载至我们的 A4 开发板，Qsys_ Key.sof 下载完成后，我们还需要在 Eclipse 软件中将 Qsys_ Key.elf 文件下载至我们的 A4 开发板，Qsys_ Key.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，我们可以看到 A4 开发板上的 Led 全部熄灭了，当我们按下 KEY1 时，D1 就会被点亮，如图 1.124 所示。

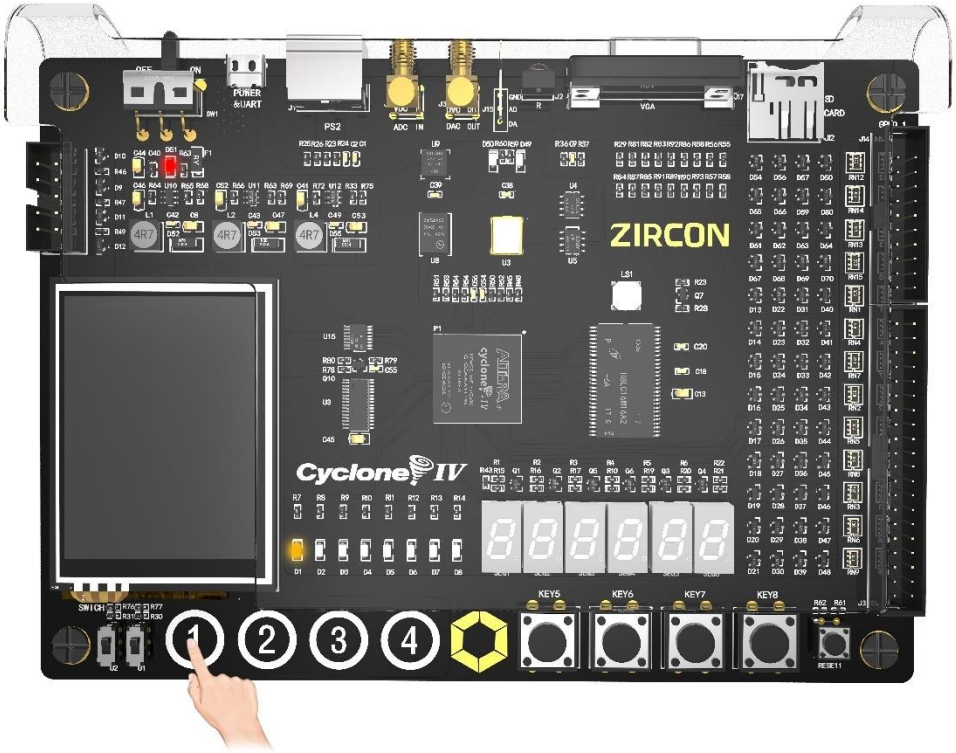

图 1.124 按键实验的板级调试图

从该图中可以看出，我们的按键实验程序是正确的，我们在A4开发板上实现了按下不同的按键，点亮相对应的 Led 功能。

1.10.7 PIO IP 核的应用实例——中断

(1) 实验目的

介绍完了 PIO 按键实验，接下来我们再来看看 PIO 中断实验，该实验主要是利用 PIO IP 核内部中断来实现控制 LED 灯亮灭。中断信号由按键提供，产生中断后，在中断事件中将读取按键的值并将值输出到 LED 上进行显示。在这个实验中，我们将学习到 PIO IP 核中断在 Qsys 软件中是如何进行配置，以及中断服务程序的编写、注册和调试方法。

(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 1.125 所示。

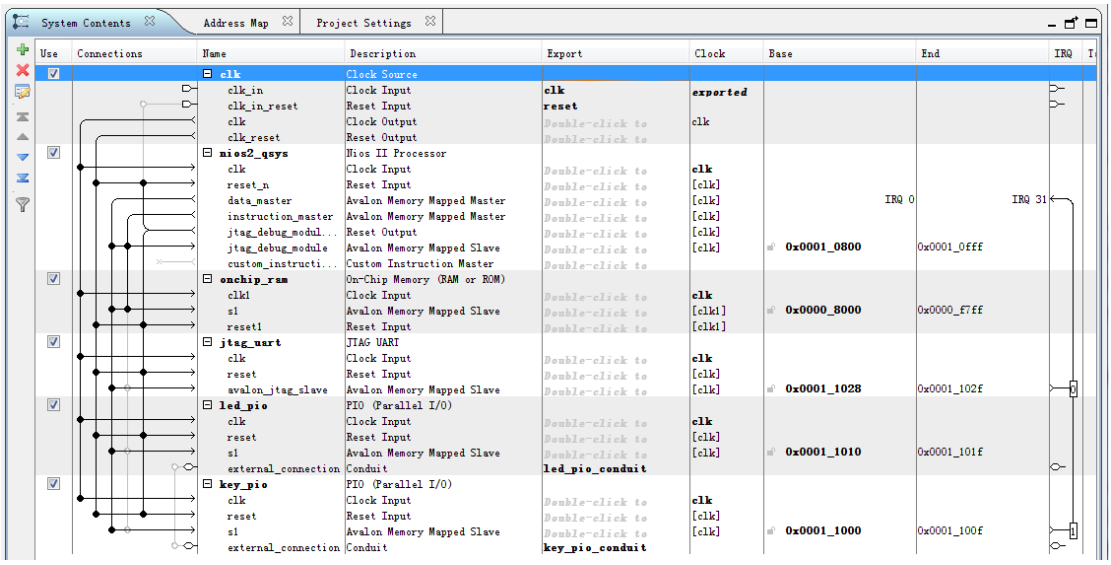

图 1.125 中断实验的硬件框架图

从该图中我们可以看出，我们PIO中断实验与我们PIO按键实验的硬件框架结构是一样的，没有添加任何组件，唯一不同的是，我们改动了 key_pio 的配置，我们为 key_pio 添加了中断选项，它的配置如图 1.126 所示。

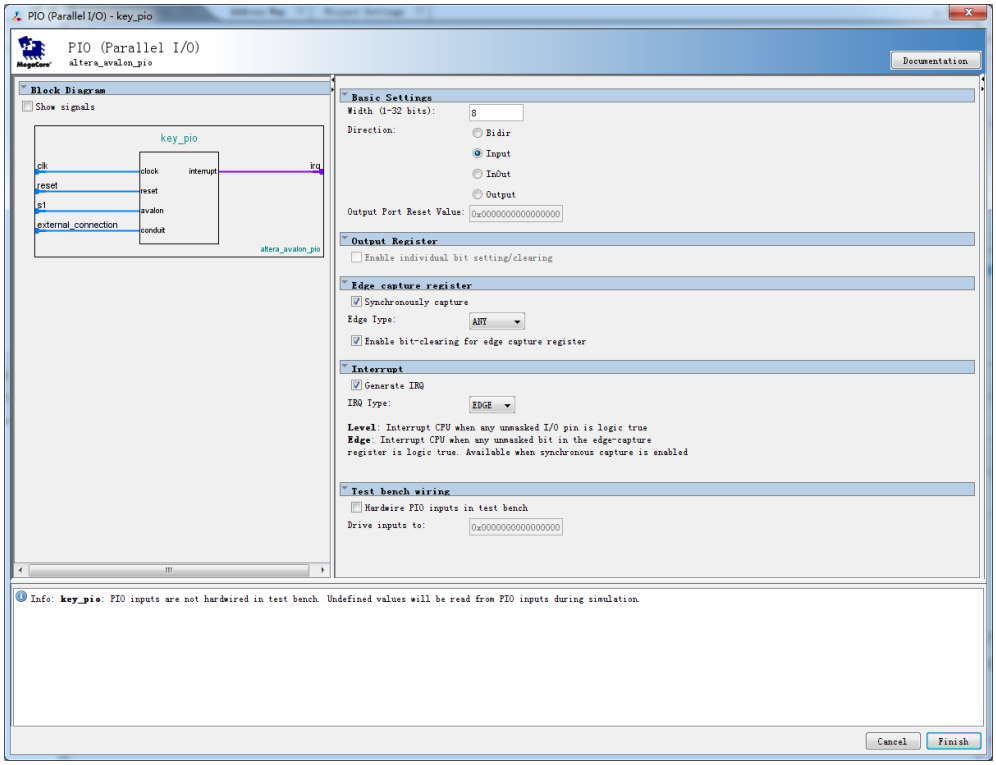

图 1.126 中断实验的 PIO 配置页面

从该图中可以看出，我们选择的是边沿上升沿触发或边沿下降沿触发中断。这里我们需要注意的是，我们还选择了 Enalbe bit-clearing for edge capture register 这个选项，如果大家还记得我们之前讲过的内容，那么大家肯定知道这个选项是用来干什么的，如果我们选中了该选项，，那么对于 edge capture 就应该是写 1 清中断；如果我们没有选择 enable bit-clearing for edge

capture register，则是写任意数清中断。

(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1.8所示。

代码 1.8 Qsys_Key_Interrupt.c 代码

```
//-----------------------------------------------------------------------
//-- 文件名    :  Qsys_Key_Interrupt.c
//-- 描述      :  按下不同的按键，点亮相对应的 Led。
//-- 修订历史  :  2014-1-1
//-- 作者      :  Zircon Opto-Electronic Technology CO.,Ltd.
//-----------------------------------------------------------------------
#include "system.h"                     //系统头文件
#include "altera_avalon_pio_regs.h"     //pio 寄存器头文件
#include "sys/alt_irq.h"                //中断头文件
#include "unistd.h"                     //延迟头文件

void IRQ_Init();                        //中断初始化函数
void IRQ_Key_Interrupts();              //中断服务子程序

//-----------------------------------------------------------------------
//-- 名称      :  main()
//-- 功能      :  程序入口
//-- 输入参数  :  无
//-- 输出参数  :  无
//-----------------------------------------------------------------------
int main()
{
   IRQ_Init(); // 初始化 PIO 中断
   IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, 0xFF); //初始化 LED 全灭

   while(1)
   {
      usleep(1000);
   }
}

//-----------------------------------------------------------------------
//-- 名称      :  IRQ_Init()
//-- 功能      :  中断初始化函数
//-- 输入参数  :  无
//-- 输出参数  :  无
//-----------------------------------------------------------------------
void IRQ_Init()
{
```

```
40    IOWR_ALTERA_AVALON_PIO_IRQ_MASK(KEY_PIO_BASE, 0xff);  // 使能中断
41    IOWR_ALTERA_AVALON_PIO_EDGE_CAP(KEY_PIO_BASE, 0xff);  // 清中断边沿捕获寄存器
42    // 注册 ISR
43    alt_ic_isr_register(
44         KEY_PIO_IRQ_INTERRUPT_CONTROLLER_ID,// 中断控制器标号，从 system.h 复制
45         KEY_PIO_IRQ,                        // 硬件中断号，从 system.h 复制
46         IRQ_Key_Interrupts,                 // 中断服务子函数
47         0x0,                                // 指向与设备驱动实例相关的数据结构体
48         0x0);                               // flags，保留未用
49  }
50
51  //-----------------------------------------------------------------------
52  //-- 名称     :  IRQ_Key_Interrupts()
53  //-- 功能     :  中断服务子程序
54  //-- 输入参数  :  无
55  //-- 输出参数  :  无
57  //-----------------------------------------------------------------------
58  void IRQ_Key_Interrupts()
59  {
60    alt_u32  key_state,led_state; //Key 和 Led 缓存变量
61    //读取按键的值，并赋值给 key_state。
62    key_state = IORD_ALTERA_AVALON_PIO_DATA(KEY_PIO_BASE);
63    //注意：Key 悬空为低电平，按下为高电平。Led 低电平亮，高电平灭。
64    //为了保证按下不同的按键点亮相对应 Led，因此，我们将按键的值取反后，再赋值给 led_state。
65    led_state = ~key_state;
66    //将 led_state 中的值输出到 Led 上进行显示。
67    IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, led_state);
68    IOWR_ALTERA_AVALON_PIO_EDGE_CAP(KEY_PIO_BASE, 0xff);  //清中断边沿捕获寄存器
69  }
```

下面我们就来给大家讲解一下该程序中的关键点。通过前面的学习我们知道,当中断来了以后，我们的程序会跳转到异常处理地址，这时，Nios II 处理器开始执行一段硬件抽象层(HAL)插入的代码，并判断中断源和中断优先级，然后再跳转到我们编写的中断服务子程序(ISR)中，整个过程不需要我们任何操作，我们只需要将中断服务子程序的信息告知给硬件抽象层就可以了，具体如何将中断服务子程序的信息告知给硬件抽象层，我们需要完成以下两个步骤:

- 参照 void isr_name(void *context,alt_u32 id)函数原型编写中断服务子程序。其中 isr_name 是用户给 ISR 取的函数名；void *context 是指向传递给 ISR 的信息的全局变量指针（通常是寄存器信息）；id 是硬件中断号，在 system.h 中声明，比如我们这里的 KEY_PIO_IRQ。
- 调用 alt_irq_register(alt_u32 id,void *context,void (*isr)(void *,alt_u32)) 或者是 alt_ic_isr_register(alt_u32 ic_id, alt_u32 irq, alt_isr_func isr, void *isr_context, void *flags);函数向硬件抽象层登记中断服务子程序。其中 alt_u32 id 和 alt_u32 irq 是硬件

中断号；void *context 和 void *isr_context 是指向传递给 ISR 的信息的全局变量指针（通常是寄存器信息）; void (*isr)(void *,alt_u32)和 alt_isr_func isr 是指向 ISR 的函数指针；alt_u32 ic_id 是中断控制器标号，*flags 保留未用。

(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_Key_Interrupt.sof 下载至我们的 A4 开发板，Qsys_Key_Interrupt.sof 下载完成后，我们还需要在Eclipse软件中将Qsys_Key_Interrupt.elf文件下载至我们的 A4 开发板，Qsys_Key_Interrupt.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，我们可以看到 A4 开发板上的 Led 全部熄灭了，当我们按下 KEY3 时，D3 就会被点亮，如图 1.127 所示。

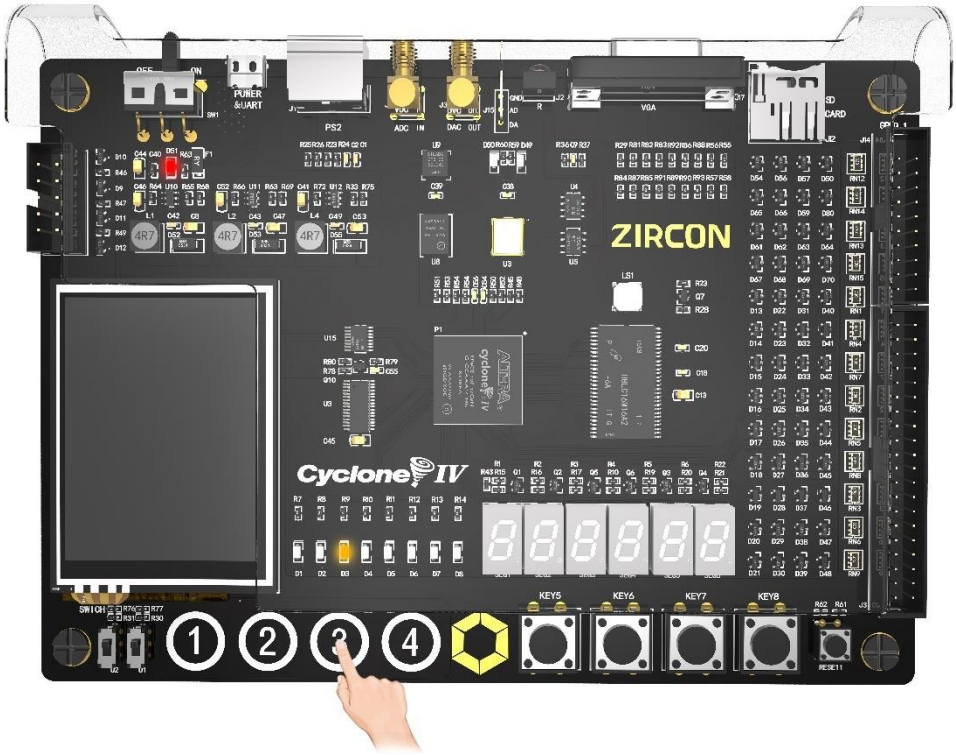

图 1.127 中断实验的板级调试图

从该图中可以看出，我们的中断实验程序是正确的，我们在A4开发板上利用中断实现了按下不同的按键，点亮相对应的 Led 功能。