上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

本科课程项目报告

UNDERGRADUATE COURSE PROJECT REPORT

HW2: 嵌入式系统开发

姓　　名:	Brian Li
学　　号:	521030990021
课　　程:	MST4311-智能芯片与系统设计
任课教师:	刘婷
学院(系) :	电子信息与电气工程学院
开课学期:	2024年 (秋季)

2024 年 12 月 26 日

目 录

第 1 章　引言

1.1　实验目的

- 学会使用双通道 GPIO
- 学会使用中断
- 理解 IP 的设置和使用
- 实现简单的嵌入式系统的开发

1.2　实验内容

在 lab5 基础上，计数周期的计算改为 ONE_TENTH * (2 bit SW + 2bit BTN)

- **双通道 GPIO 设计**: 将 2 个 switch 按键 (SW0, 1) 和两个 button 按键 (BTN2, BTN3) 采用同一个 GPIO 接口挂在 AXI 总线上
- **中断程序设计**: 捕捉来自 PL BTN0 和 BTN1 的中断，使得按下按键0(1)时，通过 UART 串口输出 "Interrupt valid, Button 0(1) is pressed"
- **中断屏蔽设计**: 一次中断触发后，1 s 内不响应第二次中断

本实验基于 Vivado 2018.2 实现。

第 2 章　实验设计

本实验的系统框图如图2.1所示。

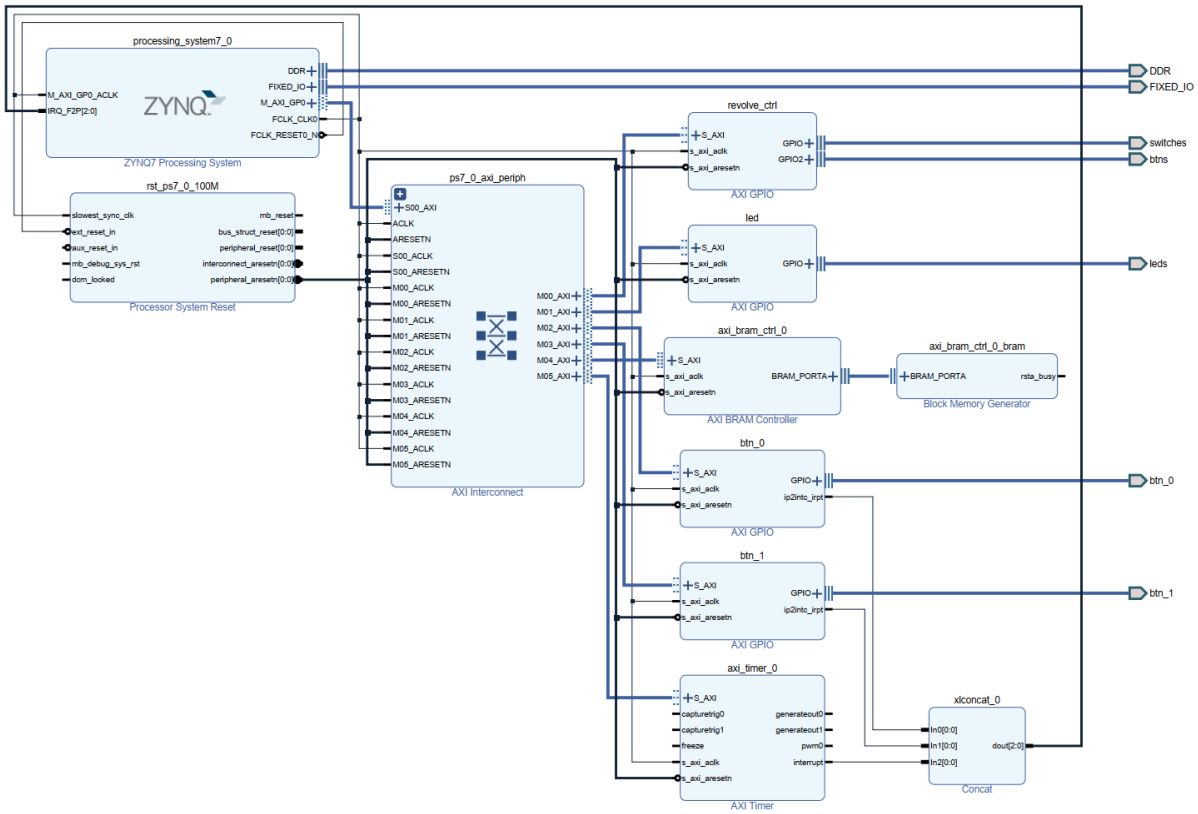

图 2.1　Block Design

考虑到实验要求中断的实现**尽量使用添加硬件管脚后利用软件驱动的方式**，这里不添加 AXI Interrupt Controller，而是在 Zynq 核上添加 PL-PS 中断端口，后续通过编写 SDK 中的 .c 文件实现中断初始化、中断处理程序、中断屏蔽等。

图2.1中，按键 SWITCH 和 BUTTON 作为 AXI GPIO 的输入，LED 作为 AXI GPIO 的输出。在执行 lab5 跑马灯程序的同时，当 AXI GPIO 检测到按键0或1状态发生变化时，AXI GPIO 就会产生一个中断信号，直接传入 Zynq 核上的 PL-PS 中断端口，Zynq 核通过接收到的中断信号控制 UART 端口输出指定信息。

2.1　Vivado 工程设计

2.1.1　Zynq 核

创建一个空白 Zybo 工程，并新建 Block design。添加 IP: ZYNQ7 Processing System，执行 *Run Block Automation*，并按照 lab1 *Configure the processing block with just UART 1 peripheral enabled* 节中的指导，去掉如下端口:

- ENET
- USB 0
- SD 0
- GPIO MIO
- Quad SPI Flash
- Timer 0

之后如图2.2所示，添加 PL-PS 中断端口。

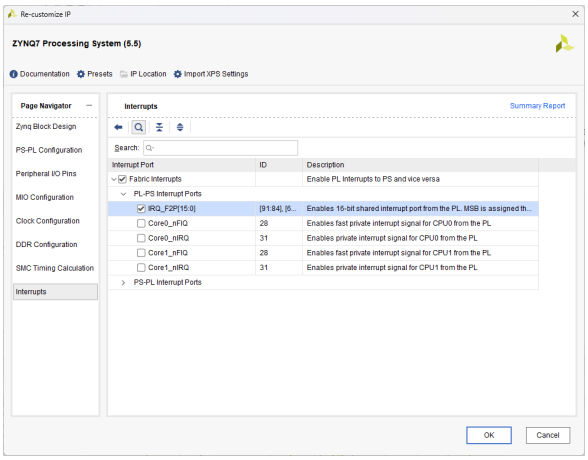

图 2.2　PL-PS 中断端口设置

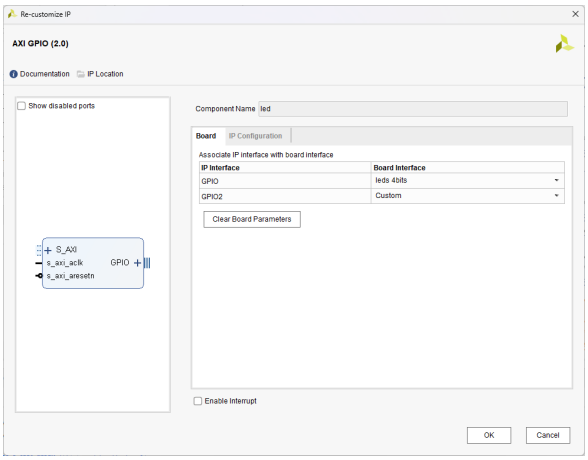

图 2.3　LED GPIO 端口设置

2.1.2　GPIO

接下来添加 IP: AXI GPIO。共需要添加4个 AXI GPIO，分别用于 LED0-3 (走马灯)，SW0-1 & BTN2-3 (走马灯控制信号)，BTN0 (生成中断)，BTN1 (生成中断)。

如图2.3所示，设置 LED 的 AXI GPIO 端口。

如图2.4所示，设置 SW0-3 和 BTN2-3 的 AXI GPIO 端口。该 GPIO 为双通道，通道一用于 SW0-3，通道二用于 BTN2-3。由于 BTN0 和 BTN1 需要另外用于产生中断，因此通道二不能设置为 btns_4bit，需要先设置为 custom 2 bit，后续在2.1.5节设置管脚约束。

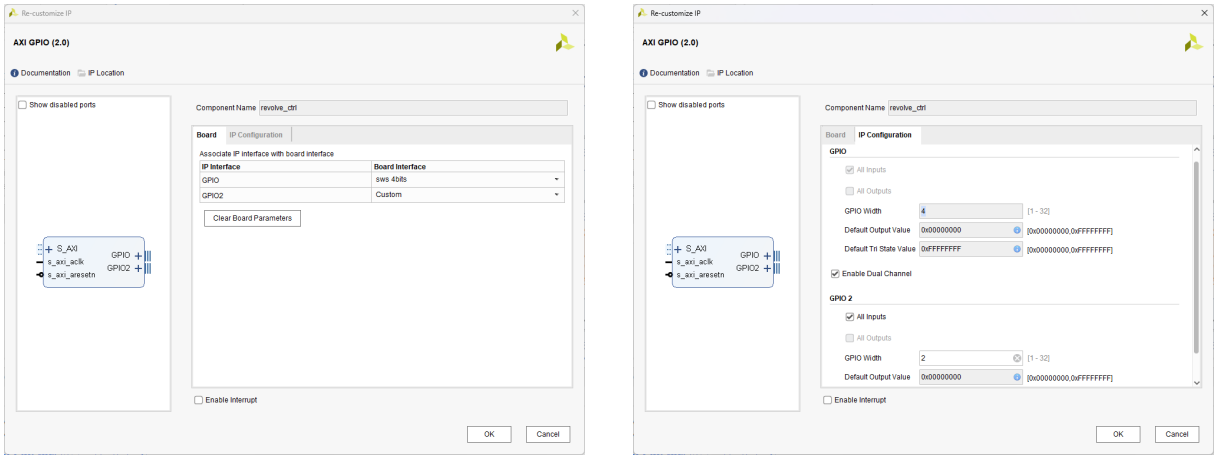

图 2.4　SW0-3 和 BTN2-3 的 AXI GPIO 端口设置

如图2.5所示，设置 BTN0，1 的**两个** AXI GPIO 端口。通道设置为 custom 1 bit。此处应勾选*Generate Interrupt*以使该模块在 GPIO 信号改变时产生中断信号。

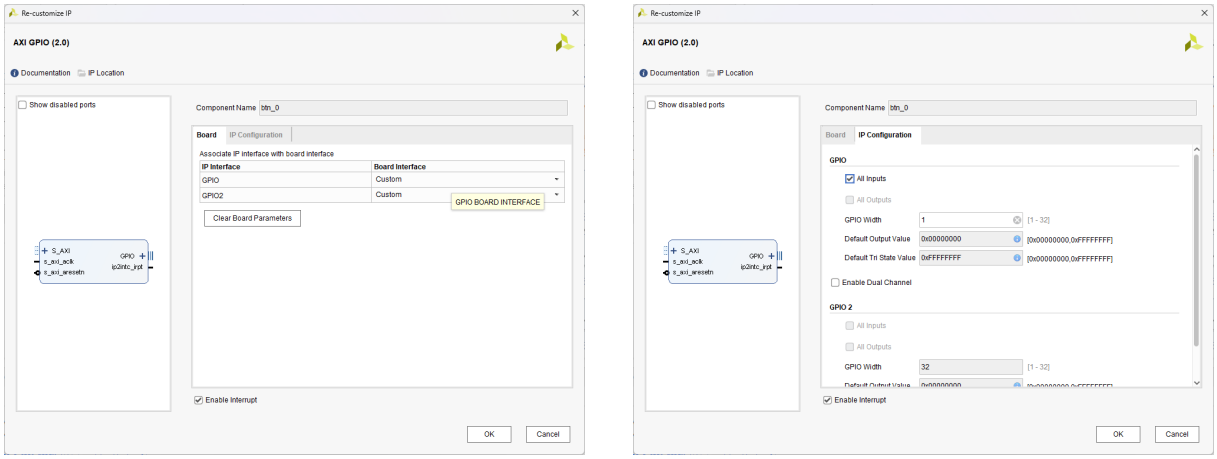

图 2.5　BTN0，1 的 AXI GPIO 端口设置

2.1.3　其它 IP

添加 IP: AXI Timer，用于为中断屏蔽提供定时器，如图2.6所示。

添加 IP: Concat，用于将来自 BTN0，1，以及 AXI Timer 的中断信号组合为 3 bit 的总线，并输出给 Zynq 核，如图2.7所示。

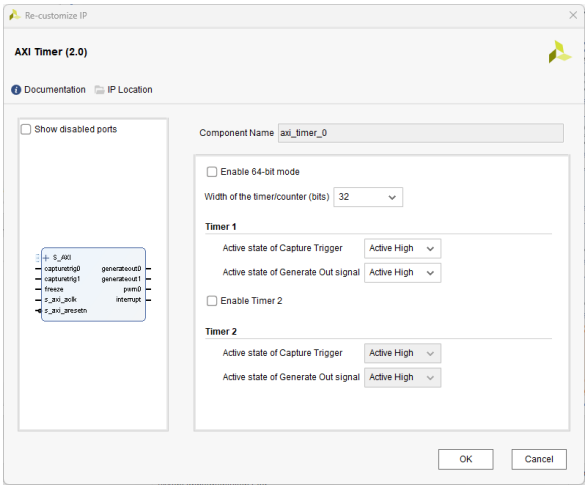

图 2.6　Timer IP 设置

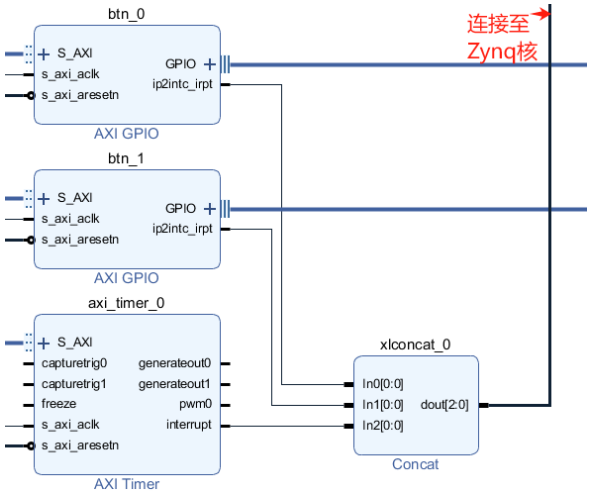

图 2.7　Concat 端口连线

2.1.4　硬件综合

点击 *Run Connection Automation*，完成连接，至此 Block design 设计完成， Block design 应如图2.1所示。之后，按照 lab1 *Generate Top-Level and Export to SDK* 节中的指导，执行 *Generate Output Products*，生成 HDL Wrapper，并执行 *Run Synthesis*。

2.1.5　IO 规划

打开 Synthesized Design，切换到 I/O Planning 视图，设置管脚约束。SW0-3 以及 LED0-3 的管脚已设置好，接下来设置 BTN0-3 的管脚约束，如图2.8所示，将管脚约束保存为 .xdc 文件。

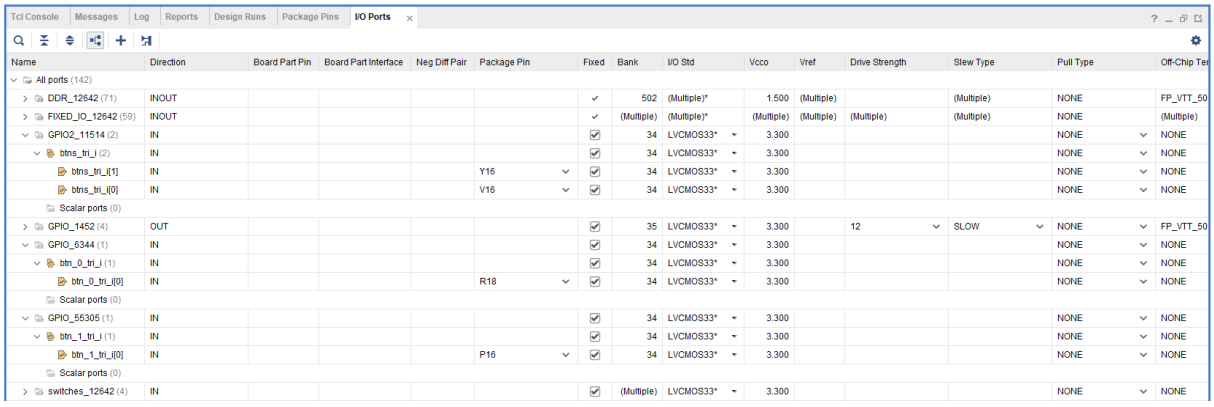

图 2.8　BTN0-3 管脚约束设置

2.2　SDK 编程

执行 *Generate Bitstream*。在 Vivado 中点击 *File -> Export -> Export Hardware*，选择 *Include Bitstream*，点击 *OK*。之后点击 *File -> Launch SDK*，在 SDK 中创建一个新的工程，选择 *Empty Application*。

以下代码主要参考2014年柏林应用科技大学 (BHT) 的 Adam P. Taylor 教授编写的讲义 *How to Use Interrupts on the Zynq SoC* 以及 SDK BSP 中提供的示例代码。

2.2.1　GIC 初始化

该部分代码可复制自 SDK BSP 中的 *xscugic_example.c* 文件，将部分函数参数修改为本实验所用的对象即可。

2.2.2　GPIO 配置

本实验中，GPIO 模块包括:

- BTN0，BTN1: 用于产生中断
- BTN2-3 和 SW0-1 用于控制 LED 走马灯周期
- LED0-3: 走马灯

各模块的初始化代码可复制自 SDK BSP 中的 *xgpio_intr_example.c* 文件。

由于 BTN0 和 BTN1 分别引出一条 IRQ_F2P 中断输入 Zynq 核，需要各配置一个中断处理程序。先关闭两个按键的中断响应，然后在中断处理程序中输出指定信息，最后设置 AXI Timer 开始倒计时 1 s，使得 1 s 后再次开启中断响应。以下以 BTN0 为例，展示中断处理程序的代码。

```
1  void BTN0_Handler(void *CallbackRef)
2  {
3      XGpio *GpioPtr = (XGpio *)CallbackRef;
4
5      /* Clear and disable the Interrupt */
6      XGpio_InterruptClear(GpioPtr, GPIO_CHANNEL_MASK);
7      XGpio_InterruptGlobalDisable(GpioPtr);
8      XGpio_InterruptGlobalDisable(&btn1);
9
10     /* Export message to UART terminal */
11     xil_printf("Button 0 pressed\r\n");
12
13     /* Restart the AXI timer for interrupt blocking */
```

```
14      XTmrCtr_Stop(&TmrCtr, 0);
15      XTmrCtr_Reset(&TmrCtr, 0);
16      XTmrCtr_Start(&TmrCtr, 0);
17  }
```

BTN2-3 和 SW0-1 通过 GPIO 读取状态，LED0-3 通过 GPIO 输出状态。

2.2.3　AXI Timer 配置

本实验中，AXI Timer 用于实现中断屏蔽功能。其初始化代码可复制自 SDK BSP 中的 *xtmrctr_intr_example.c* 文件。其中计时器在按钮按下后复位的值应设置为 *xtime_l.h* 中 1 s 的计数值 *COUNTS_PER_SECOND*，如下所示。

```
1  XTmrCtr_SetResetValue(&TmrCtr, TIMER_CNTR_0, COUNTS_PER_SECOND);
```

在 AXI Timer 的中断处理程序中，则需要重新使能 BTN0 和 BTN1 的中断响应。

```
1  void TmrCtrHandler(void *CallBackRef, u8 TmrCtrNumber)
2  {
3      xil_printf(" Button 0 and 1 interrupt re-enabled\r\n");
4
5      /* Enable the GPIO channel interrupts */
6      XGpio_InterruptGlobalEnable(&btn0);
7      XGpio_InterruptGlobalEnable(&btn1);
8  }
```

2.2.4　主循环

在 lab5 的基础上，由于走马灯周期改为了 *ONE_TENTH * (2 bit SW + 2bit BTN)*，因此需要在主循环中读取 SW0-1 和 BTN2-3 的状态，计算走马灯周期，同时去掉“按下按钮退出循环”的功能。

```
1  // Read led control switches & buttons
2  dip_check = XGpio_DiscreteRead(&ctrl, 1);
3  psb_check = XGpio_DiscreteRead(&ctrl, 2);
4  if (dip_check != dip_check_prev || psb_check != psb_check_prev)
5  {
6    xil_printf("DIP Switch Status \%x, \%x\r\n",
7          dip_check_prev, dip_check);
8    xil_printf("Push Button Status \%x, \%x\r\n",
9          psb_check_prev, psb_check);
```

```
10    dip_check_prev = dip_check;
11    psb_check_prev = psb_check;
12    // load timer with the new switch settings
13    XScuTimer_LoadTimer(TimerInstancePtr,
14              ONE_TENTH * (dip_check + psb_check));
15    count = 0;
16  }
```

另外，由于 LED 模块相比 lab5，改为了使用 GPIO 模块实现，其写入函数应改为:

```
1  XGpio_DiscreteWrite(&led, 1, count);
```

第 3 章　总结

本实验中，我学会了使用双通道 GPIO 和 AXI Timer，实现了中断的初始化、中断处理程序、中断屏蔽等功能。

通过实验，我们对于 Zynq SoC 的中断机制有了更深入的理解，掌握了嵌入式开发的基本方法和 Vivado 使用相关技能。

实验中遇到了许多困难，感谢助教老师们的无私帮助，使得我能够顺利完成本实验。