# 

# 微

# 机

# 原

# 理

# 实

# 验

# 报

# 告

# 学院：电子信息与通信学院

# 班级：提高2101班

# 姓名：杨筠松

# 学号：U202115980

# 实验时间：2022年4月26日

串行IO接口实验

**1.实验任务**

利用SPI IP核，timer IP核以及DA模块，控制DA模块输出周期可变锯齿波，且锯齿波周期由switch控制。锯齿波周期最长约为1s，最短约为60ms。

►提示：switch 输入的数据，控制定时计数器的定时时间，定时计数器定时时间到，输出一个新数据到DA转换器。

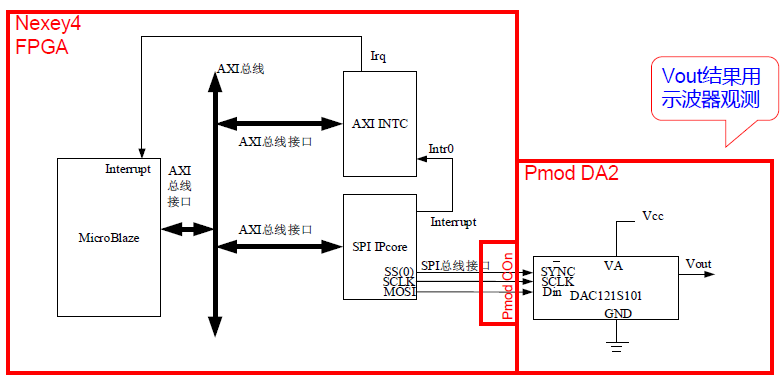
**2.实验目的**

1. 理解UART串行通信协议以及接口设计
2. 理解SPI串行通信协议
3. 掌握UART串行接口设计
4. 掌握SPI串行接口设计
5. 掌握串行DA接口设计
6. 掌握串行AD接口设计

**3.实验环境**

1. Windows 11操作系统；
2. 编辑工具：Vivado，SDK；
3. 实验开发板：Nexys 4 DDR；
4. 示波器。

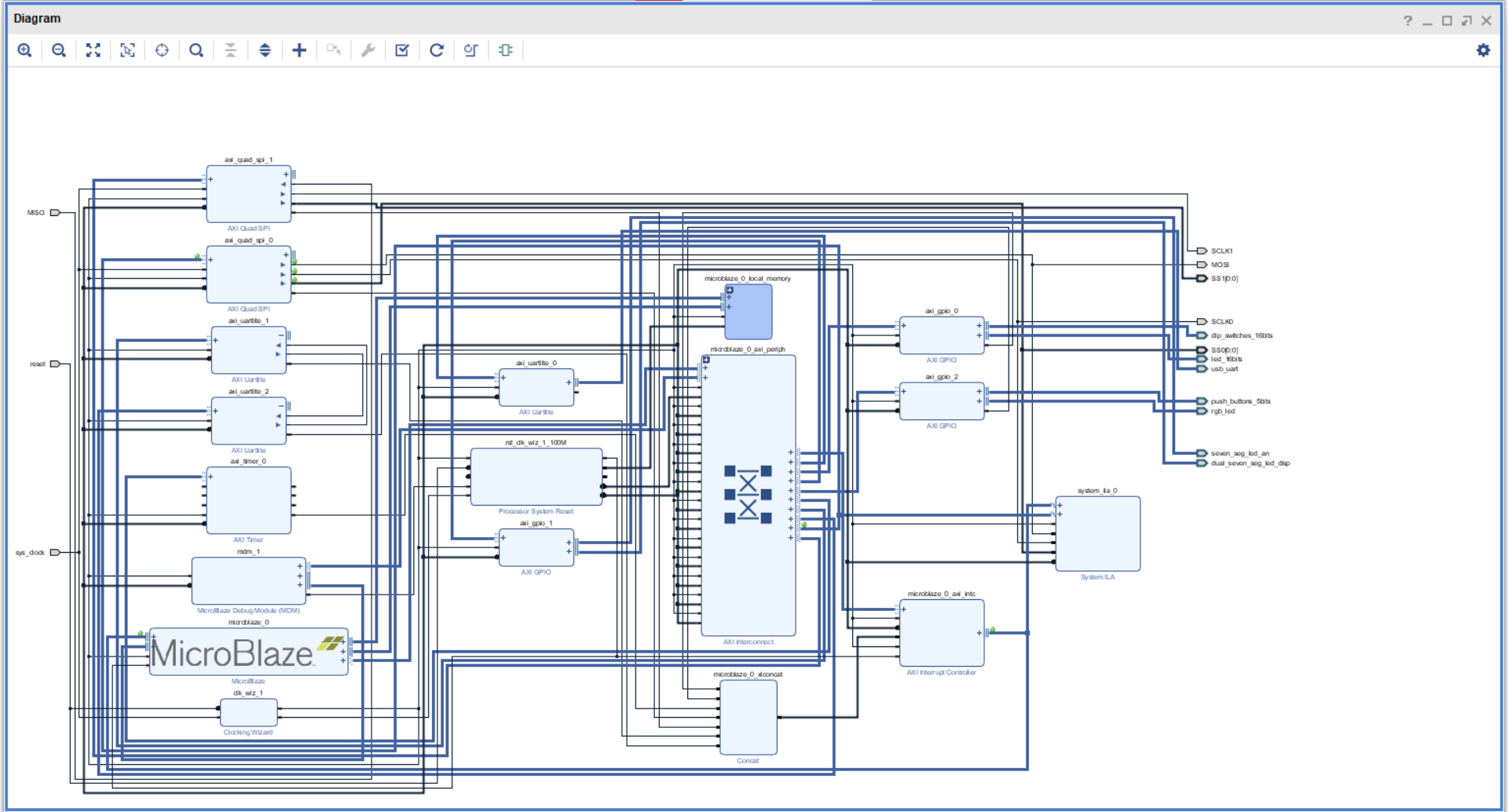
**4.设计方案**

硬件框图：

Nexys 4 DDR开发板的16个独立开关SW接GPIO\_0的第一个端口，设定SPI、TIMER、GPIO中断。

首先要通过中断读取按键健值，读入数据用Xilinx C语言提供的基本IO读函数。设置变量RESET\_VALUE0表示保存的16位独立开关状态二进制数据，再通过计时器中断，装载计时器的初始值为RESET\_VALUE0，实现不同频率锯齿波的输出。

**5.实现过程**

1. 硬件平台：
2. 源代码及注释：

**#include "stdio.h"**

**#include "xparameters.h"**

**#include "xil\_exception.h"**

**#include "xil\_io.h"**

**#include "xio.h"**

**#include "xgpio\_l.h"**

**#include "xtmrctr\_l.h"**

**#include "xintc\_l.h"**

**#include "xspi\_l.h"**

**#define T0\_RESET\_VALUE\_MIN 1000-2 // 60ms**

**#define T0\_RESET\_VALUE\_MAX 20000-2 // 1.2S**

**void switch\_handler() \_\_attribute\_\_ ((fast\_interrupt)); // intr(0)**

**void timer\_handler() \_\_attribute\_\_ ((fast\_interrupt));// intr(2)**

**//void DA\_handler() \_\_attribute\_\_ ((fast\_interrupt));//intr(3)**

**u16 volt, lastvolt;**

**int main()**

**{**

**//GPIO0 switch and led**

**Xil\_Out32(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_TRI\_OFFSET, 0x0000ffff);// set switches input and led output**

**Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_TRI2\_OFFSET, 0x0);// set segment display**

**Xil\_Out32(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_ISR\_OFFSET, XGPIO\_IR\_CH1\_MASK|XGPIO\_IR\_CH2\_MASK);// CLEAR INTERRUPT**

**Xil\_Out32(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_IER\_OFFSET, XGPIO\_IR\_CH1\_MASK);// ENABLE INTERRUPT OF CH1**

**Xil\_Out32(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_GIE\_OFFSET, XGPIO\_GIE\_GINTR\_ENABLE\_MASK);//ENABLE GLOBAL INTERRUPT**

**//SPI**

**//RESET**

**Xil\_Out32(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR + XSP\_SRR\_OFFSET,XSP\_SRR\_RESET\_MASK );**

**//SPI MASTER MODE CPOL = 1 CPHA = 0**

**Xil\_Out32(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR + XSP\_CR\_OFFSET, XSP\_CR\_ENABLE\_MASK|\**

**XSP\_CR\_MASTER\_MODE\_MASK|XSP\_CR\_CLK\_POLARITY\_MASK);**

**//SET SSR to select slave0**

**Xil\_Out32(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR + XSP\_SSR\_OFFSET, 0xfffffffe);**

**//ENABLE SPI INTERRUPT**

**Xil\_Out32(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR + XSP\_IIER\_OFFSET, XSP\_INTR\_TX\_EMPTY\_MASK);**

**Xil\_Out32(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR + XSP\_DGIER\_OFFSET, XSP\_GINTR\_ENABLE\_MASK);**

**//TIMER**

**int tcsr0;**

**tcsr0 = Xil\_In32(XPAR\_AXI\_TIMER\_0\_BASEADDR + XTC\_TCSR\_OFFSET);// TO GET THE VALUE 0F TCSR**

**Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR + XTC\_TCSR\_OFFSET,tcsr0&~XTC\_CSR\_ENABLE\_TMR\_MASK);// STOP THE COUNTER**

**Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR + XTC\_TLR\_OFFSET,T0\_RESET\_VALUE\_MIN);//preset value**

**Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR + XTC\_TCSR\_OFFSET,tcsr0|XTC\_CSR\_LOAD\_MASK);// LOAD THE PRESET VALUE**

**Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR + XTC\_TCSR\_OFFSET,(tcsr0&~XTC\_CSR\_LOAD\_MASK)|XTC\_CSR\_ENABLE\_TMR\_MASK|XTC\_CSR\_AUTO\_RELOAD\_MASK|\**

**XTC\_CSR\_ENABLE\_INT\_MASK|XTC\_CSR\_DOWN\_COUNT\_MASK|XTC\_CSR\_INT\_OCCURED\_MASK);// NO LOAD;ENABLE COUNT;AUTO RELOAD;ENABLE INT; CLEAR INT**

**//INTC**

**Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IAR\_OFFSET, \**

**XPAR\_AXI\_QUAD\_SPI\_0\_IP2INTC\_IRPT\_MASK|XPAR\_AXI\_GPIO\_0\_IP2INTC\_IRPT\_MASK|XPAR\_AXI\_TIMER\_0\_INTERRUPT\_MASK );//clear interrupt**

**Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IMR\_OFFSET, \**

**XPAR\_AXI\_QUAD\_SPI\_0\_IP2INTC\_IRPT\_MASK|XPAR\_AXI\_GPIO\_0\_IP2INTC\_IRPT\_MASK|XPAR\_AXI\_TIMER\_0\_INTERRUPT\_MASK);//fast interrupt**

**Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IER\_OFFSET, \**

**XPAR\_AXI\_QUAD\_SPI\_0\_IP2INTC\_IRPT\_MASK|XPAR\_AXI\_GPIO\_0\_IP2INTC\_IRPT\_MASK|XPAR\_AXI\_TIMER\_0\_INTERRUPT\_MASK);// interrupt enable**

**Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_MER\_OFFSET,XIN\_INT\_MASTER\_ENABLE\_MASK|XIN\_INT\_HARDWARE\_ENABLE\_MASK);// master enable**

**Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IVAR\_OFFSET, (int)XPAR\_INTC\_0\_SPI\_0\_VEC\_ID);**

**Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IVAR\_OFFSET + 4\*XPAR\_INTC\_0\_GPIO\_0\_VEC\_ID,(int)switch\_handler);//**

**Xil\_Out32(XPAR\_INTC\_0\_BASEADDR + XIN\_IVAR\_OFFSET + 4\*XPAR\_INTC\_0\_TMRCTR\_0\_VEC\_ID,(int)timer\_handler);//**

**microblaze\_enable\_interrupts();**

**Xil\_Out16(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR + XSP\_DTR\_OFFSET, 0);**

**return 0;**

**}**

**//void DA\_handler()**

**//{**

**// // Clear SPI interrupt**

**// Xil\_Out32(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR + XSP\_IISR\_OFFSET, Xil\_In32(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR + XSP\_IISR\_OFFSET));**

**// volt++;**

**// Xil\_Out32(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR + XSP\_DTR\_OFFSET, volt&0xfff);**

**//}**

**void timer\_handler()**

**{**

**int status;**

**status =Xil\_In32(XPAR\_AXI\_TIMER\_0\_BASEADDR + XTC\_TCSR\_OFFSET);**

**if ((status&XTC\_CSR\_INT\_OCCURED\_MASK )== XTC\_CSR\_INT\_OCCURED\_MASK)**

**{**

**volt++;**

**Xil\_Out32(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR + XSP\_DTR\_OFFSET, volt&0xfff);**

**Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR + XTC\_TCSR\_OFFSET,status);**

**}**

**}**

**void switch\_handler()**

**{**

**unsigned short hex = Xil\_In16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA\_OFFSET);**

**int reset\_value =(int)((T0\_RESET\_VALUE\_MAX - T0\_RESET\_VALUE\_MIN)/65000.0 \* hex)+ T0\_RESET\_VALUE\_MIN;**

**int tcsr0 = Xil\_In32(XPAR\_AXI\_TIMER\_0\_BASEADDR + XTC\_TCSR\_OFFSET);// TO GET THE VALUE 0F TCSR**

**Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR + XTC\_TCSR\_OFFSET,tcsr0&~XTC\_CSR\_ENABLE\_TMR\_MASK);// STOP THE COUNTER**

**Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR + XTC\_TLR\_OFFSET,reset\_value);//preset value**

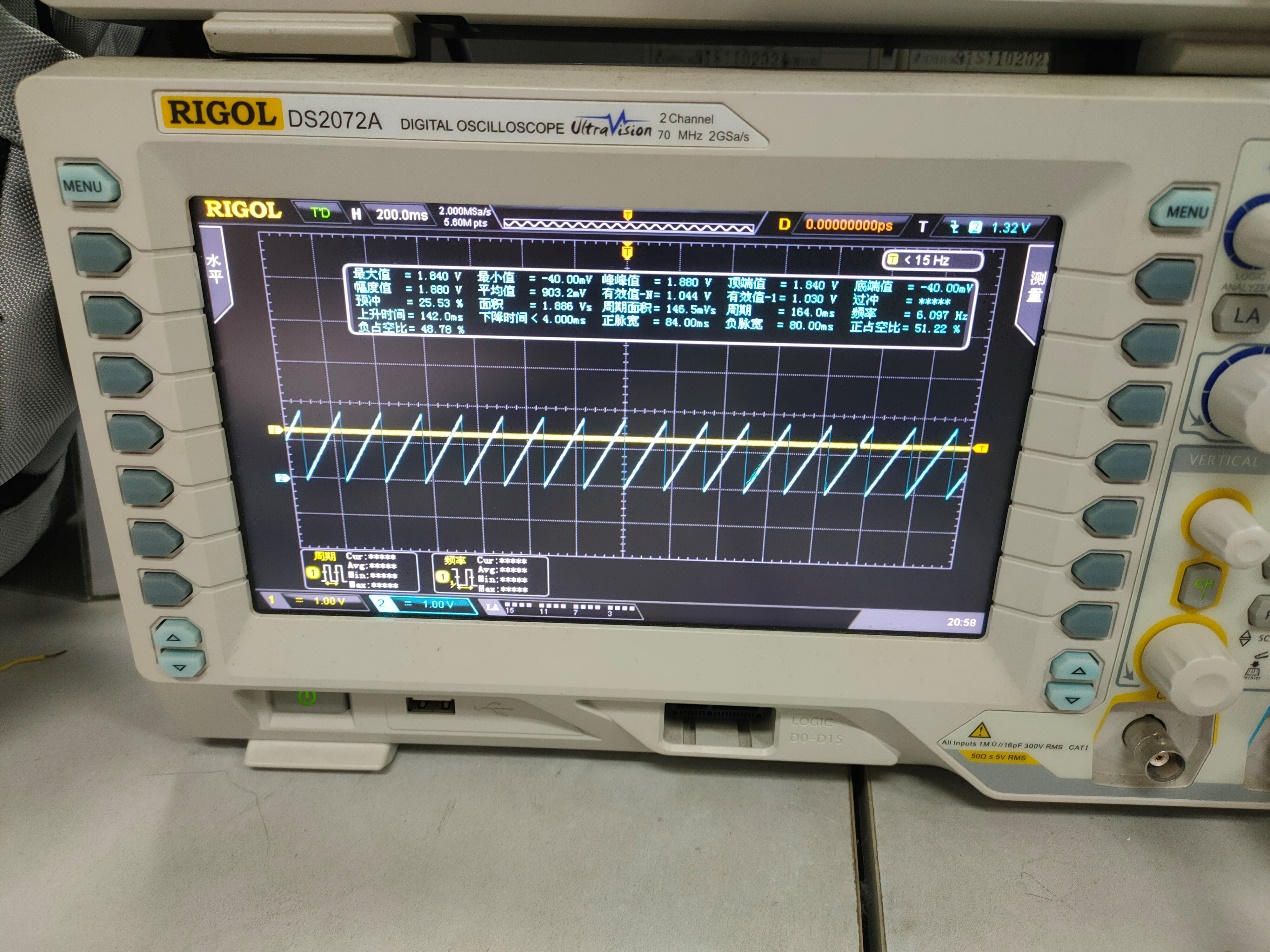
**Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR + XTC\_TCSR\_OFFSET,tcsr0|XTC\_CSR\_LOAD\_MASK);// LOAD THE PRESET VALUE**

**Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR + XTC\_TCSR\_OFFSET,(tcsr0&~XTC\_CSR\_LOAD\_MASK)|XTC\_CSR\_ENABLE\_TMR\_MASK|XTC\_CSR\_AUTO\_RELOAD\_MASK|\**

**XTC\_CSR\_ENABLE\_INT\_MASK|XTC\_CSR\_DOWN\_COUNT\_MASK|XTC\_CSR\_INT\_OCCURED\_MASK);// NO LOAD;ENABLE COUNT;AUTO RELOAD;ENABLE INT; CLEAR INT**

**Xil\_Out32(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_ISR\_OFFSET,XGPIO\_IR\_CH1\_MASK);**

**}**

**6.实验结果**

**七、实验总结**

通过这个实验，我还学会了基本串行IO的使用方法，与示波器结合了起来，更加熟练了Vivado和SDK的使用方法以及加深了对NEXYS 4 DDR 开发板认识也对接口的利用，认识上有了更多的了解，加深了对中断模式的印象。