**串行IO接口程序设计**

通信2002班 涂增基 U202013990

1. **实验任务**

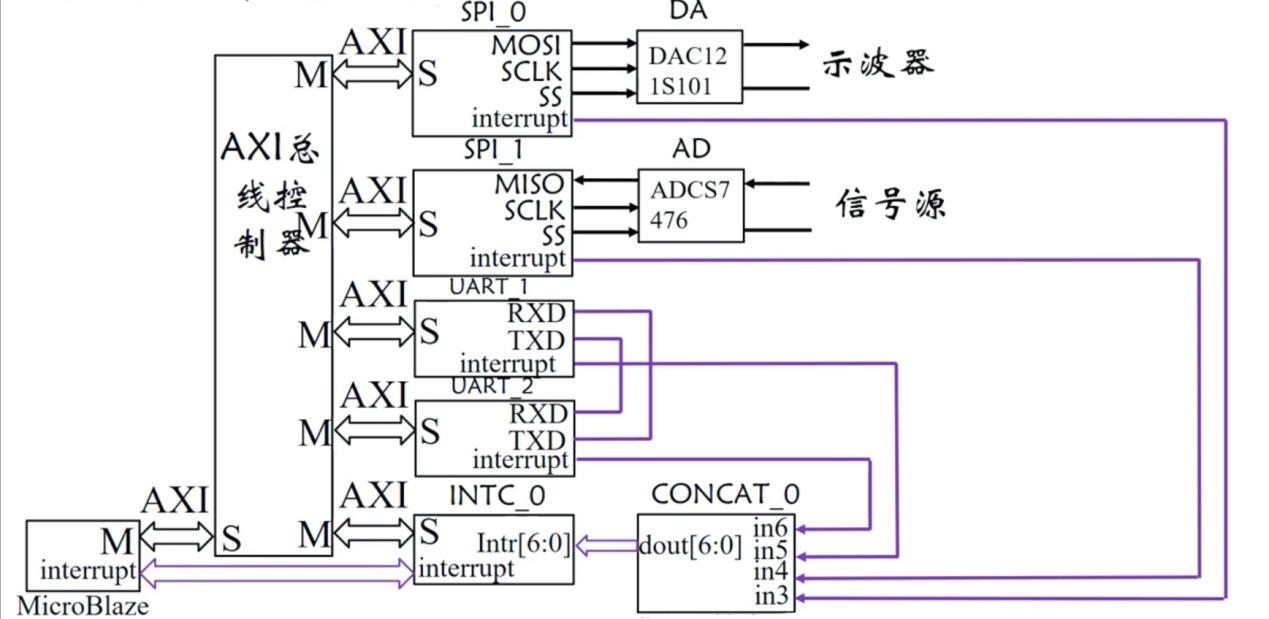
利用SPI IP 核，timer IP 核、GPIO IP核以及DA 模块，控制DA 模块输出周期可变锯齿波，且锯齿波周期由switch 控

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。

1. **实验目的**

* 理解UART串行通信协议以及接口设计
* 理解SPI串行通信协议
* 掌握UART串行接口设计
* 掌握SPI串行接口设计
* 掌握串行DA接口设计
* 掌握串行AD接口设计

**三、硬件电路框图**

****

**四、软件流程图**

****



1. **软件源代码**

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

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

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

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

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

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

**#include** "xparameters.h"

**int** RESET\_VALUE = 100000000/0xfff-2;

**void** **My\_ISR**() **\_\_attribute\_\_**((interrupt\_handler));

u16 volt=0;

**void** **switchHandler**(); //开关中断

**void** **timerHandler**(); //按键中断

**int** **main**()

{

RESET\_VALUE = 100000000/0xfff-2;

//设定SPI接口的通信模式，设定SPI为主设备，CPOL=1,CPHA-0,时钟相位180°，自动方式，高位优先传送

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);

//设定SSR寄存器

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

//开放SPI发送寄存器空中断

Xil\_Out32(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR+XSP\_IIER\_OFFSET,XSP\_INTR\_TX\_EMPTY\_MASK);//中断源为SPI接口发送完数字信号则产生中断

Xil\_Out32(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR+XSP\_DGIER\_OFFSET,XSP\_GINTR\_ENABLE\_MASK); //开启SPI接口的中断输出

//GPIO中断使能

Xil\_Out32(XPAR\_AXI\_GPIO\_0\_BASEADDR+XGPIO\_TRI\_OFFSET,0xffff);//开关switch设置为输入

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR+XGPIO\_TRI2\_OFFSET,0X0); //LED设置为输出

Xil\_Out32(XPAR\_AXI\_GPIO\_0\_BASEADDR+XGPIO\_IER\_OFFSET,XGPIO\_IR\_CH1\_MASK);//GPIO\_0中断使能

Xil\_Out32(XPAR\_AXI\_GPIO\_0\_BASEADDR+XGPIO\_GIE\_OFFSET,XGPIO\_GIE\_GINTR\_ENABLE\_MASK);//GPIO\_0全局中断使能

//定时器初始化

Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR+XTC\_TCSR\_OFFSET

,Xil\_In32(XPAR\_AXI\_TIMER\_0\_BASEADDR+XTC\_TCSR\_OFFSET)&~XTC\_CSR\_ENABLE\_TMR\_MASK);//写TCSR，停止定时器

Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR+XTC\_TLR\_OFFSET,RESET\_VALUE);//写TLR，预置计数初值

Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR+XTC\_TCSR\_OFFSET

,Xil\_In32(XPAR\_AXI\_TIMER\_0\_BASEADDR+XTC\_TCSR\_OFFSET)|XTC\_CSR\_LOAD\_MASK);//装载计数初值

Xil\_Out32(XPAR\_AXI\_TIMER\_0\_BASEADDR+XTC\_TCSR\_OFFSET

,(Xil\_In32(XPAR\_AXI\_TIMER\_0\_BASEADDR+XTC\_TCSR\_OFFSET)&~XTC\_CSR\_LOAD\_MASK)\

|XTC\_CSR\_ENABLE\_TMR\_MASK|XTC\_CSR\_AUTO\_RELOAD\_MASK|XTC\_CSR\_ENABLE\_INT\_MASK|XTC\_CSR\_DOWN\_COUNT\_MASK);//开始计时 自主获取允许中断减计数 \*/

//中断控制器intr0中断源使能

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

XPAR\_AXI\_TIMER\_0\_INTERRUPT\_MASK|

XPAR\_AXI\_QUAD\_SPI\_1\_IP2INTC\_IRPT\_MASK|

XPAR\_AXI\_GPIO\_2\_IP2INTC\_IRPT\_MASK); //开放定时器T0及SPI中断

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

//处理器中断使能

microblaze\_enable\_interrupts();

//启动传输，发送数据0

Xil\_Out16(XPAR\_AXI\_QUAD\_SPI\_0\_BASEADDR+XSP\_DTR\_OFFSET,0); //启动SPI传输，产生时钟和片选信号

//while(1);

**return** 0;

}

**void** **My\_ISR**()

{

**int** status;

status=Xil\_In32(XPAR\_AXI\_INTC\_0\_BASEADDR+XIN\_ISR\_OFFSET); //读入中断状态

**if**((status&XPAR\_AXI\_GPIO\_0\_IP2INTC\_IRPT\_MASK)==XPAR\_AXI\_GPIO\_0\_IP2INTC\_IRPT\_MASK)

{

switchHandler(); //如果开关产生了中断，则进入开关中断服务函数

}

**else** **if**((status&XPAR\_AXI\_TIMER\_0\_INTERRUPT\_MASK)==XPAR\_AXI\_TIMER\_0\_INTERRUPT\_MASK)

{

timerHandler(); //如果定时器产生了中断，则进入定时器中断服务函数

}

Xil\_Out32(XPAR\_AXI\_INTC\_0\_BASEADDR+XIN\_IAR\_OFFSET,status);

}

**void** **switchHandler**() //开关中断服务程序

{

**int** sw;

sw = Xil\_In16(XPAR\_AXI\_GPIO\_0\_BASEADDR+XGPIO\_DATA\_OFFSET); //读入开关值

Xil\_Out32(XPAR\_AXI\_GPIO\_0\_BASEADDR+XGPIO\_DATA2\_OFFSET,sw); //把开关的状态反映到LED上

Xil\_Out16(XPAR\_AXI\_GPIO\_0\_BASEADDR+XGPIO\_ISR\_OFFSET,0x01);

**int** min=6000000; //最短时间60ms

RESET\_VALUE=((sw&0x0000ffff)\*1434+min)/0xfff-2; //步进值1434=（最大时长100000000-最小时长6000000）/2^16（=65536） 每拨动一个开关加一个步进时长

//读入的开关值sw一定要与上0x0000ffff保存低16位，否则会自动有符号数扩展，装载进去的值就会是个负的

//主程序中的是定时器初始化，此处开关改变了定时器的预置值，故需要重新装载

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

Xil\_Out32(XPAR\_TMRCTR\_0\_BASEADDR+XTC\_TCSR\_OFFSET,status&(~XTC\_CSR\_ENABLE\_TMR\_MASK));

Xil\_Out32(XPAR\_TMRCTR\_0\_BASEADDR+XTC\_TLR\_OFFSET,RESET\_VALUE); //为定时器装载改变后的预置值

Xil\_Out32(XPAR\_TMRCTR\_0\_BASEADDR+XTC\_TCSR\_OFFSET,Xil\_In32(XPAR\_TMRCTR\_0\_BASEADDR+XTC\_TCSR\_OFFSET)|XTC\_CSR\_LOAD\_MASK);

status=(status&(~XTC\_CSR\_LOAD\_MASK))|XTC\_CSR\_ENABLE\_TMR\_MASK;

Xil\_Out32(XPAR\_TMRCTR\_0\_BASEADDR+XTC\_TCSR\_OFFSET,status);

}

**void** **timerHandler**() //锯齿波形成

{

volt++; //输出锯齿波，每中断一次，输出的数字信号＋1

Xil\_Out32(XPAR\_TMRCTR\_0\_BASEADDR+XTC\_TCSR\_OFFSET,Xil\_In32(XPAR\_TMRCTR\_0\_BASEADDR+XTC\_TCSR\_OFFSET));//清定时器中断，不然一直中断周期会不对

Xil\_Out32(XPAR\_INTC\_0\_BASEADDR+XIN\_IAR\_OFFSET,0x8); //普通中断模式，手动清中断

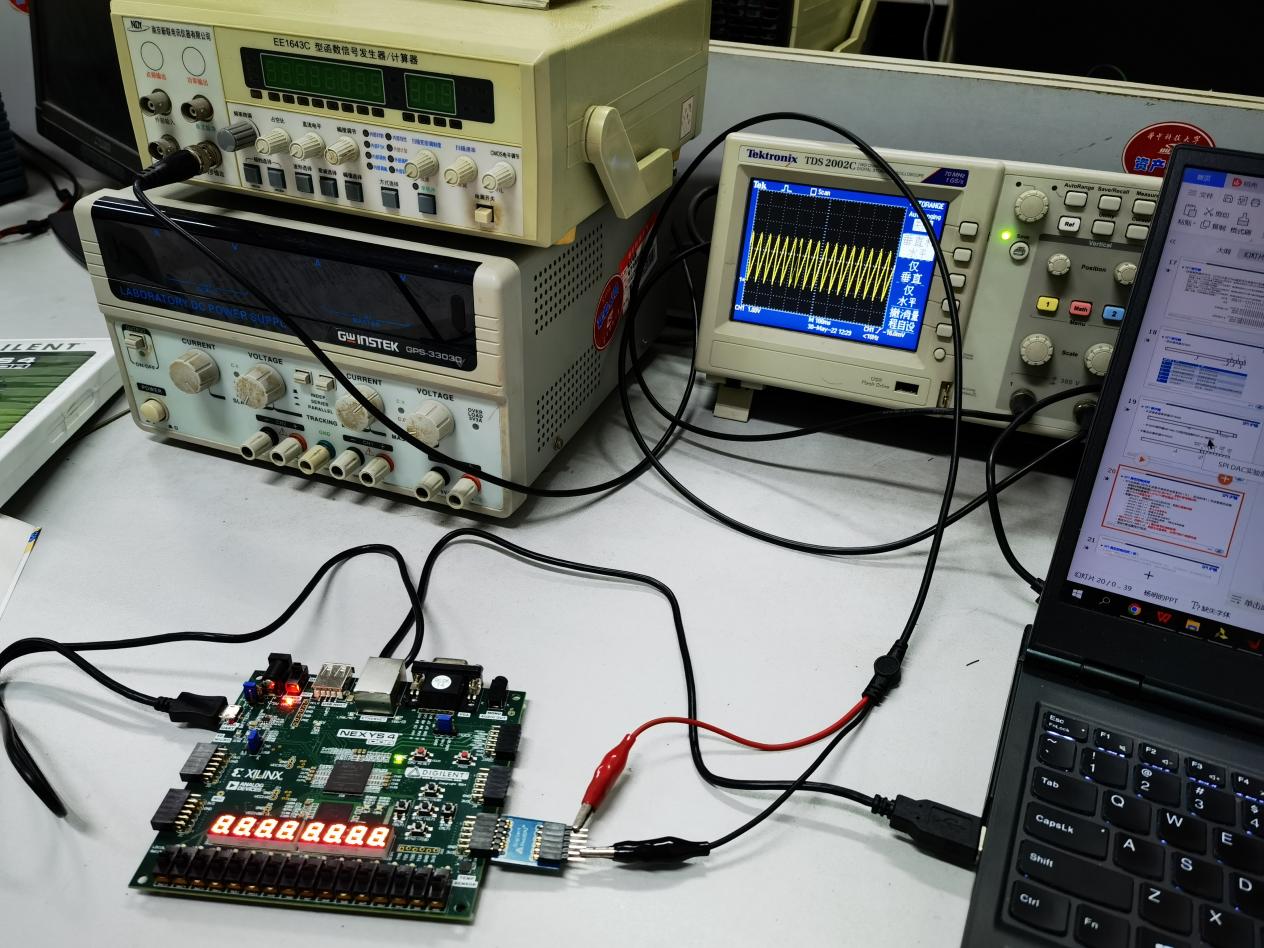
Xil\_Out16(XPAR\_SPI\_0\_BASEADDR+XSP\_DTR\_OFFSET,volt&0xfff);//启动SPI传输，产生时钟和片选信号，发送数据，有效数据为低12位

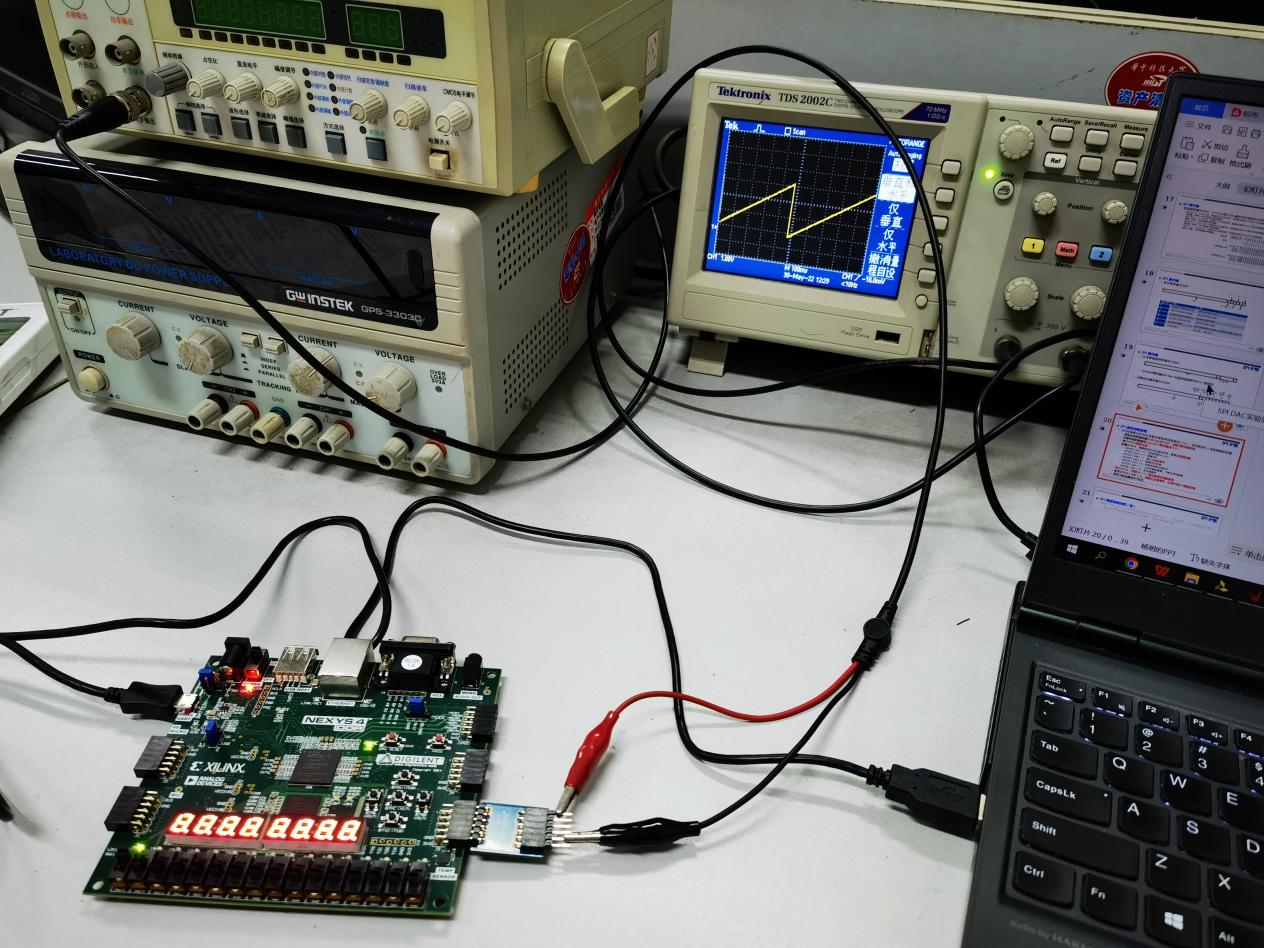
}

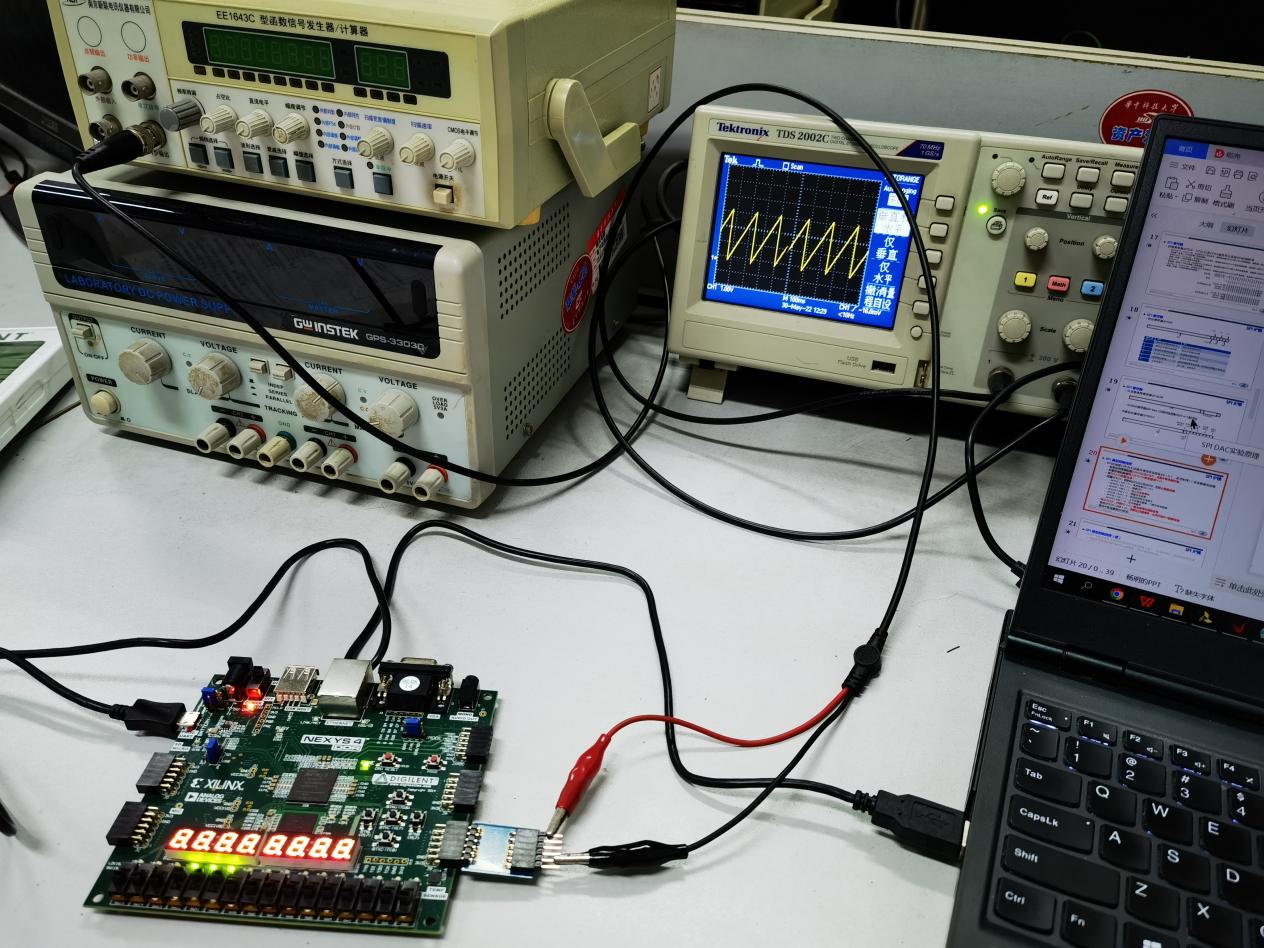
**六、实验结果**

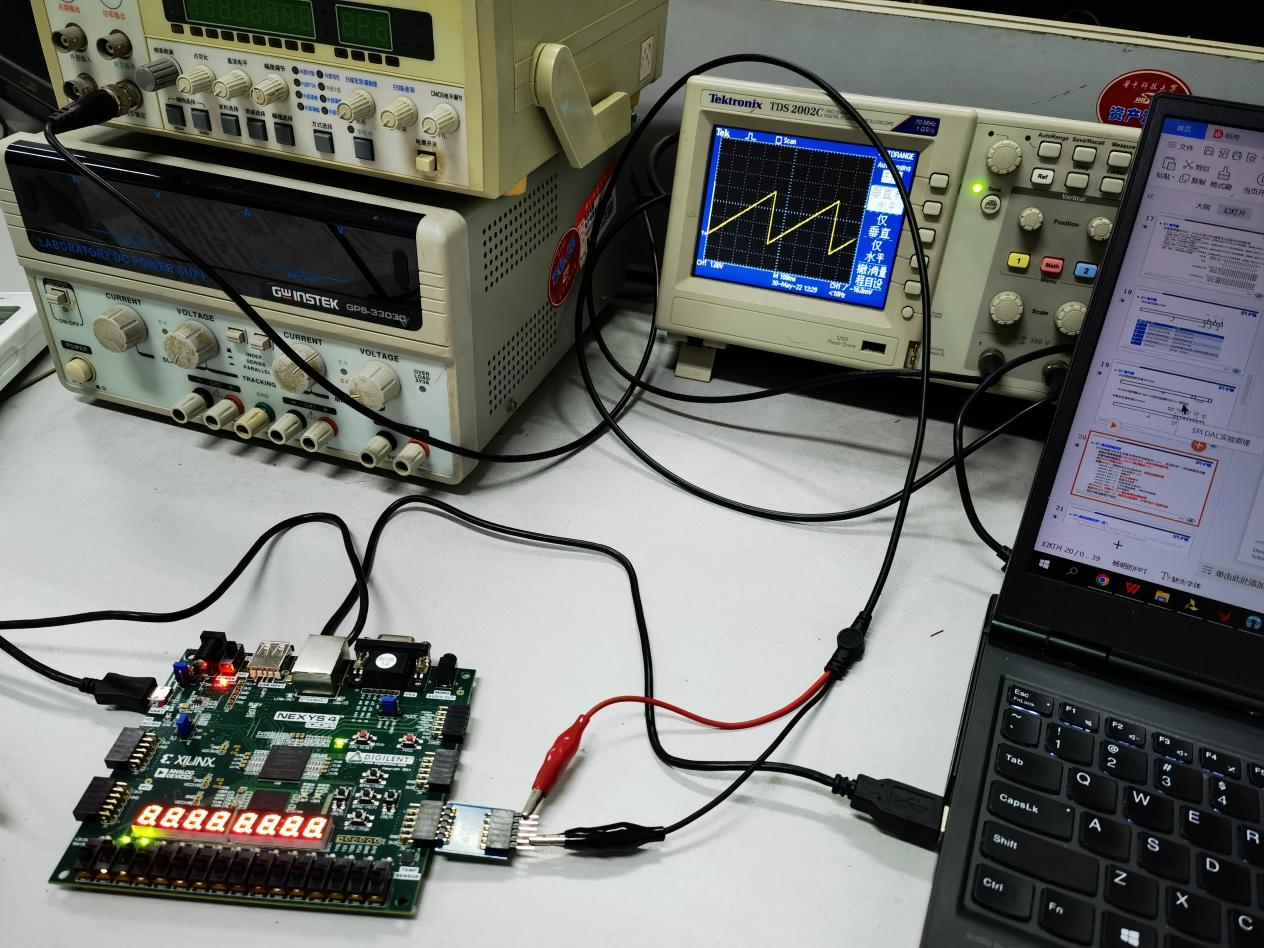
用DA转换器输出波形到示波器，得到如下测试结果：

拨动开关，可以输出不同周期的锯齿波。

****

****

****

****

**七、心得体会**

在本次实验中，由于理论课上对UARTLite和SPI都学的不是很透彻，一开始做这个实验的时候很是困难。对着实验书学习了很久，才开始写代码。

平台建立是遇到了一些问题的，一开始我跟着学习通上的视频建立平台，最终导出bit流文件总是出错，原来是引脚约束出了点问题。可能是版本不一样的问题，我用2019.2版本按照视频上的做法无法约束引脚，最后只好自己在IOPort里面进行引脚约束。

中断服务函数中，那个周期的设置并不简单，遇到了不少的问题，sw的值一定要&上0x0000ffff，否则会自动有符号扩展。然后一定要记得清除中断！本次实验中我在写计时器的时候，最后忘了清除中断，最终的周期就是错误的结果。

本次实验对中断和串行输入输出有了更深的理解，收获良多。