第六章 接口技术

并行IO接口

华中科技大学
电子信息与通信学院
School of Electronic Information and Communications

学习目标

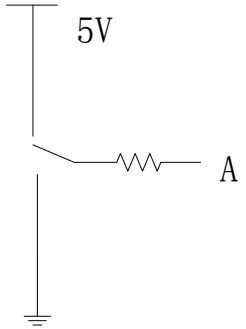
- 理解简单并行IO接口设计原理
- 掌握独立开关、LED、7段数码管、矩阵式键盘以及并行AD转换器接口设计
- 掌握基于GPIO控制器、外设控制器的接口设计

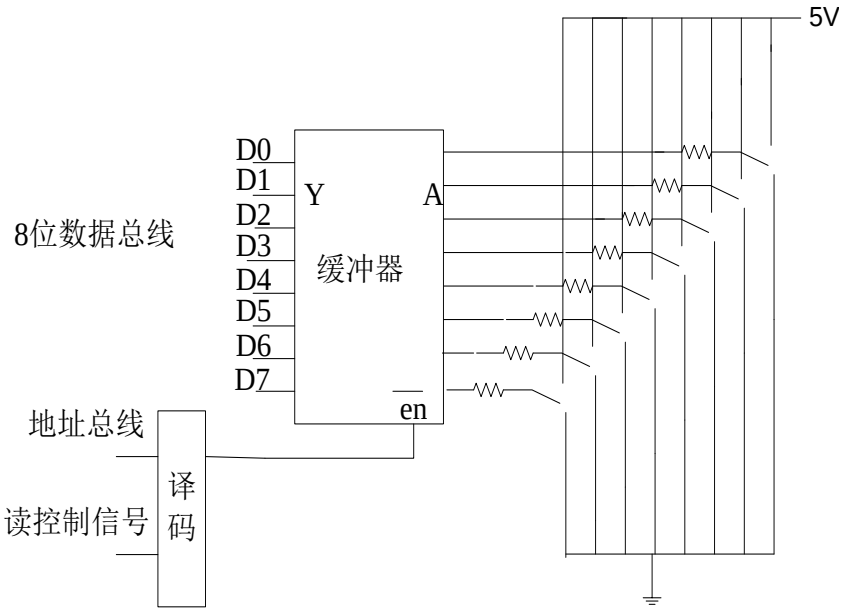
并行IO接口设计

- 独立开关输入接口
- 发光二极管输出接口
- 矩阵式键盘接口
- 七段数码管动态显示接口
- AD转换器ADC1210接口
- GPIO控制器
- 外设控制器（EPC）

独立开关输入接口

- RC吸收电路或RS触发器组成的闩锁电路来消除按键抖动；
- 采用软件延时方法消除抖动

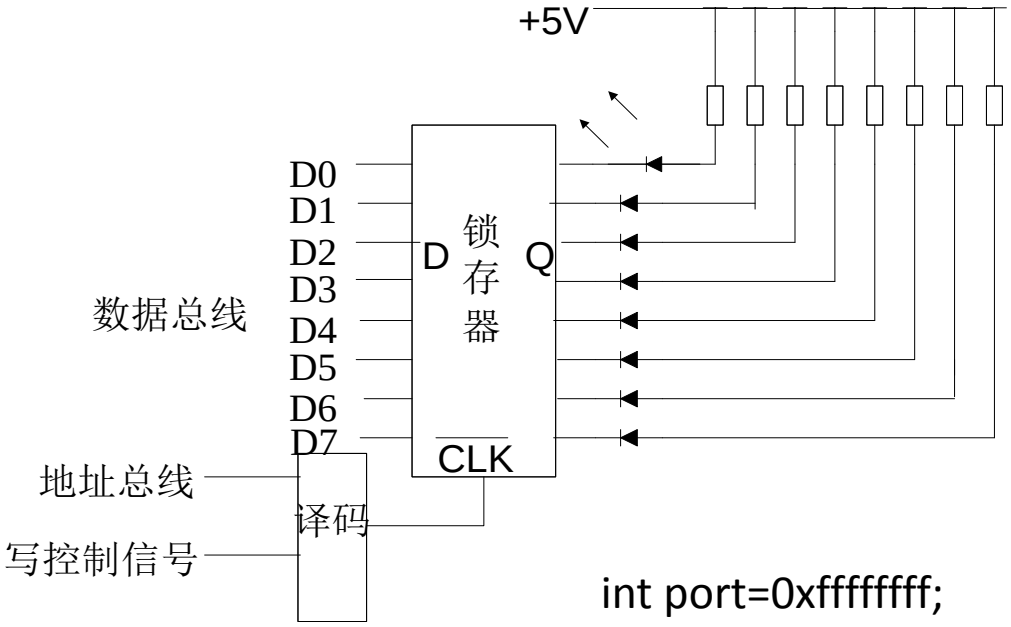


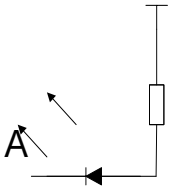


int port=0xffffffff;
char Switch=Xil_In8(port);8位开关的状态保存在Switch中

发光二极管输出接口

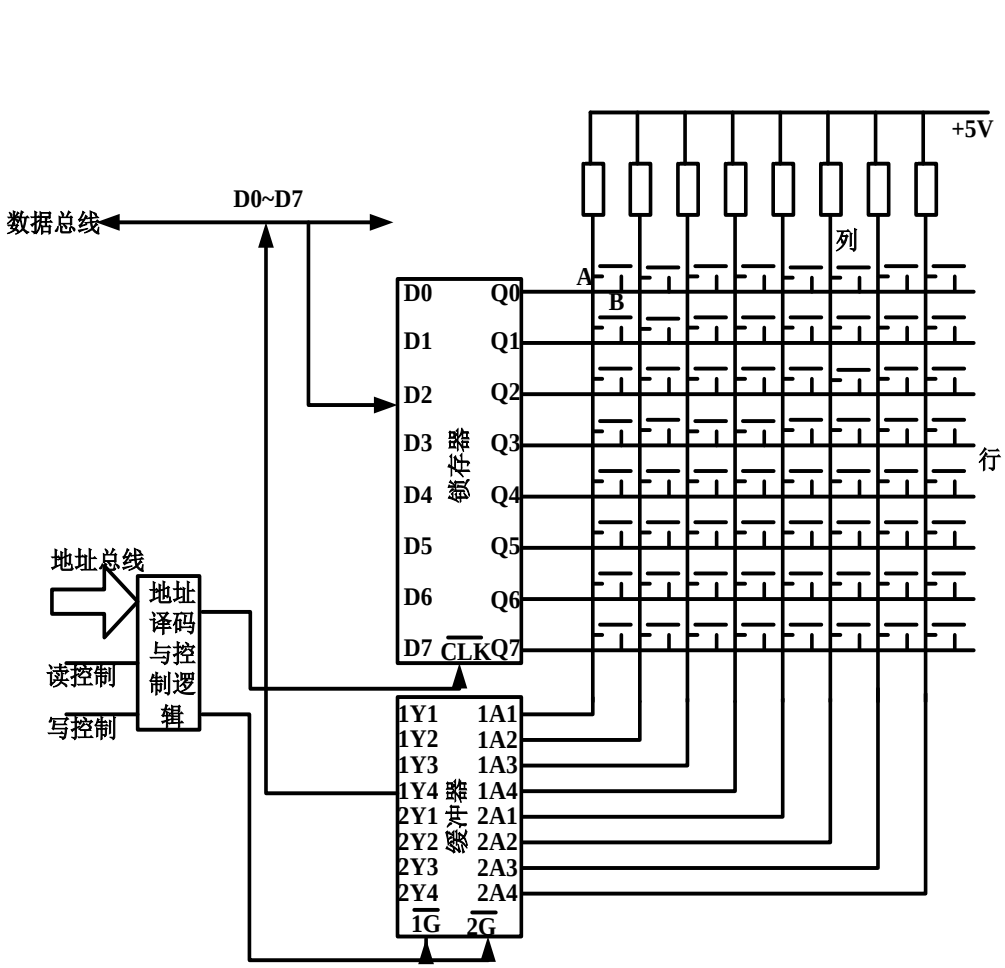
- 必须具有数据锁存功能

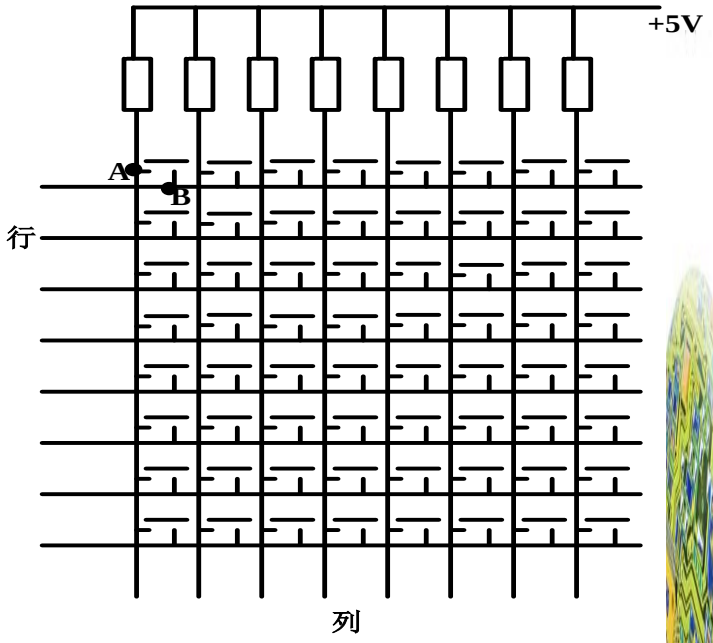





```
int port=0xffffffff;
Xil_Out8( port,0x0);
```

矩阵式键盘接口


+5V
D0~D7
数据总线
列
A
B
行
D0
D1
D2
D3
D4
D5
D6
D7
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
CLK
锁存器
地址总线
地址译码与控制逻辑
读控制
写控制
1Y1
1Y2
1Y3
1Y4
2Y1
2Y2
2Y3
2Y4
1A1
1A2
1A3
1A4
2A1
2A2
2A3
2A4
1G
2G
缓冲器


+5V
A
B
行
列

微处理器检测矩阵式键盘的状态流程

1. 首先使所有行都输出低电平；
2. 读取键盘列信号；
3. 检测是否有键按下，若没有键按下，获得0xFF，重复步骤2）3）；当有键按下时，此时获得的8位数据不等于0xff，从第一行开始，进入步骤4）；
4. 输出该行为低电平，其余行为高电平
5. 读取键盘列信号；
6. 检测是否有键按下，若没有键按下，获得0xff，指向下一行，重复步骤4）~6）；当有键按下时，此时获得的8位数据不等于0xff，进入步骤7)
7. 利用输出的行信号以及读取的列信号，从而确定属于该行该列交叉处的按键被按下。

矩阵式键盘按键编码表

	列0	列1	列2	列3	列4	列5	列6	列7
行0	0xfefe	0xfefd	0xfefb	0xfef7	0xfeef	0xfedf	0xfebf	0xfe7f
行1	0x fdfe	0xfdfd	0xfdfb	0xfdf7	0xfdef	0xfddf	0xfdbf	0xfd7f
行2	0x fbfe	0xfbfd	0xfbfb	0xfbf7	0xfbef	0xfbdf	0xfbbf	0xfb7f
行3	0x f7fe	0xf7fd	0xf7fb	0xf7f7	0xf7ef	0xf7df	0xf7bf	0xf77f
行4	0x effe	0xeffd	0xeffb	0xeff7	0xefef	0xefdf	0xefbf	0xef7f
行5	0x dffe	0xdffd	0xdffb	0xdff7	0xdfef	0xdfdf	0xdfbf	0xdf7f
行6	0x bffe	0xbffd	0xbffb	0xbff7	0xbfef	0xbfdf	0xbfbf	0xbf7f
行7	0x 7ffe	0x7ffd	0x7ffb	0x7ff7	0x7fef	0x7fdf	0x7fbf	0x7f7f

采用Xilinx C语言来读取按键的行列信息

```
int AddressPort=0xfffffffe,DataPort=0xffffffff;
        char col, row, rowtemp=0x01;
        Xil_Out8(AddressPort,0x00);
        While ((col=Xil_In8(DataPort))==0xff);
        row=~rowtemp;
        Xil_Out8 (AddressPort,row);
while ((col=Xil_In8 (DataPort))==0xff)
                { rowtemp=rowtemp<<1;
                  row=~rowtemp;
                  Xil_Out8 (AddressPort,row);
}
```

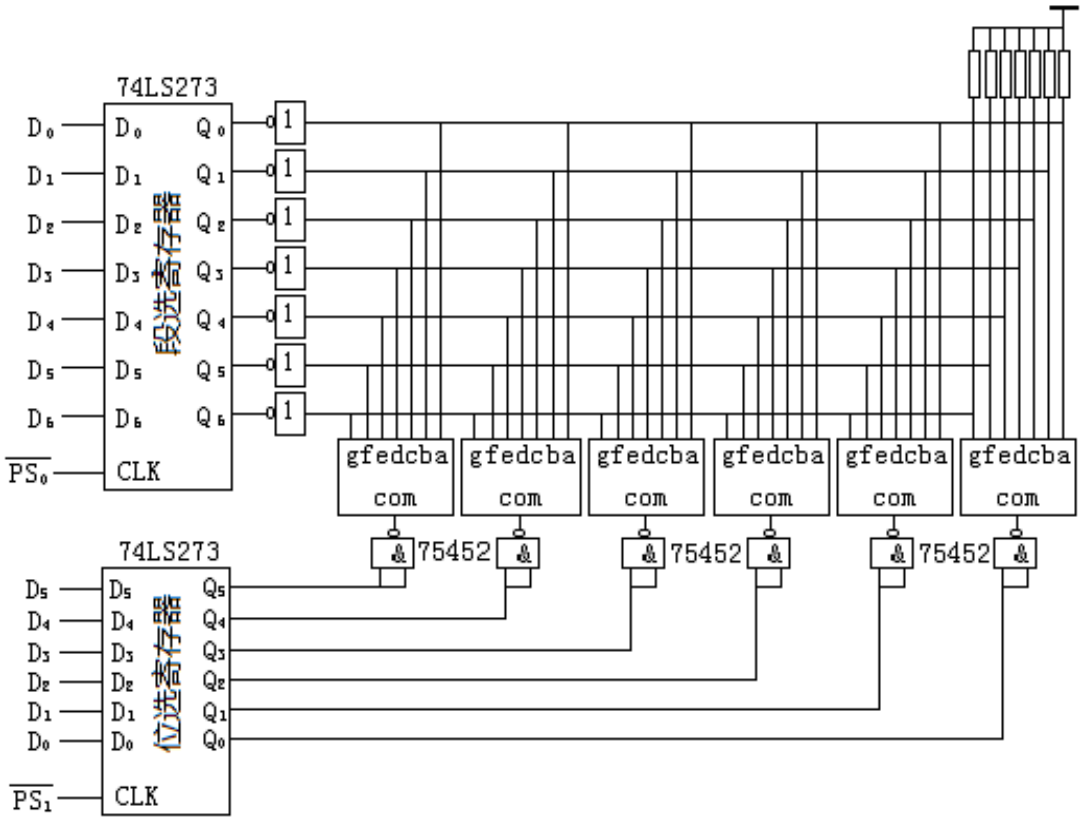
- 按键行和列信息分别保存在变量row和col中。

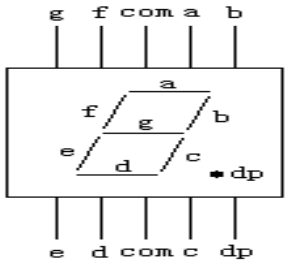
键值与编码关系

```
short int scancode[64]=
{0xfefe, 0xfefd, 0xfefb, 0xfef7, 0xfeef, 0xfedf, 0xfebf, 0xfe7f,
0xfdfe, 0xfdfd, 0xfdfb, 0xfdf7, 0xfdef, 0xfddf, 0xfdbf, 0xfd7f,
0xfbfe, 0xfbfd, 0xfbfb, 0xfbf7, 0xfbef, 0xfbdf, 0xfbbf, 0xfb7f,
0xf7fe, 0xf7fd, 0xf7fb, 0xf7f7, 0xf7ef, 0xf7df, 0xf7bf, 0xf77f,
0xeffe, 0xeffd, 0xeffb, 0xeff7, 0xefef, 0xefdf, 0xefbf, 0xef7f,
0xdffe, 0xdffd, 0xdffb, 0xdff7, 0xdfef, 0xdfdf, 0xdfbf, 0xdf7f,
0xbffe, 0xbffd, 0xbffb, 0xbff7, 0xbfef, 0xbfdf, 0xbfbf, 0xbf7f,
0x7ffe, 0x7ffd, 0x7ffb, 0x7ff7, 0x7fef, 0x7fdf, 0x7fbf, 0x7f7f};
```

```
KeyBtn=(short int)row;
KeyBtn=(KeyBtn<<8)|col;
for(i=0;i<64;i++)
{
if(scancode[i]==KeyBtn)
break;
}
xil_printf("Key %d is pressed\n\r",i);
```

七段数码管动态显示接口





10个十进制数的字形代码分别是0x40，0x79，0x24，0x30，0x19，0x12，0x02，0x78，0x00H，0x18

- PS0表示端口地址0x80，而PS1表示端口地址0x81，在6位显示器上分别显示1,2,3,4,5,6等字符，那么其Xilinx C语言程序段为如下所示：

```
char segment[5]={ 0x79,0x24,0x30,0x19,0x12,0x02};
int AddressPort=0x81,DataPort=0x80;
char position;
Xil_Out8(AddressPort,0x0);//使所有的七段数码管熄灭
While(1)
{               position=0x20;
for(int i=0;i<6;i++)
{
                        Xil_Out8(DataPort, segment[i]); //输出第一位的段码
                        Xil_Out8(AddressPort,position);//点亮第一位7段数码管
                        delay; //延时
                        position=position >> 1;//控制下一个7段数码管
}
}
```

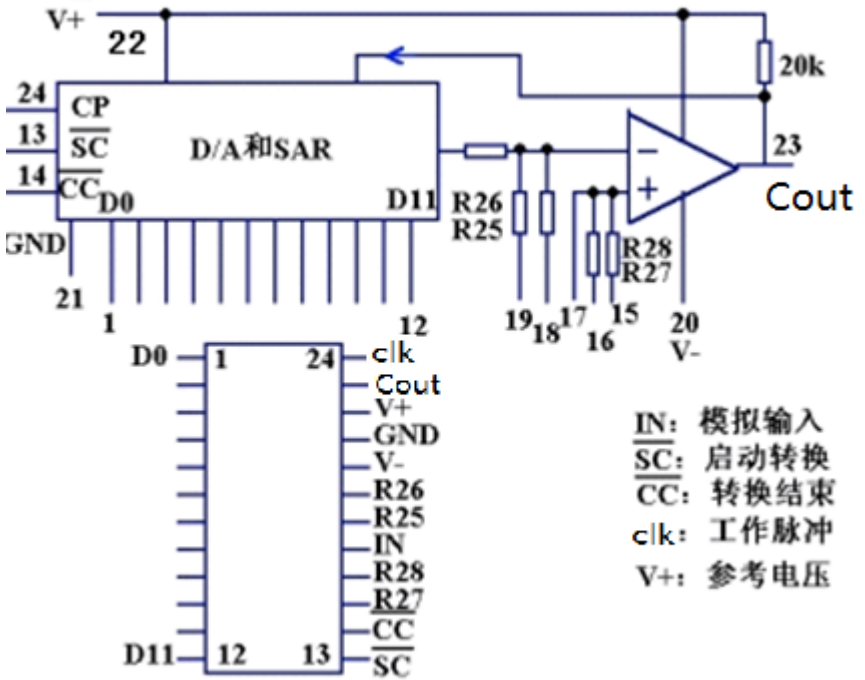
软件延时

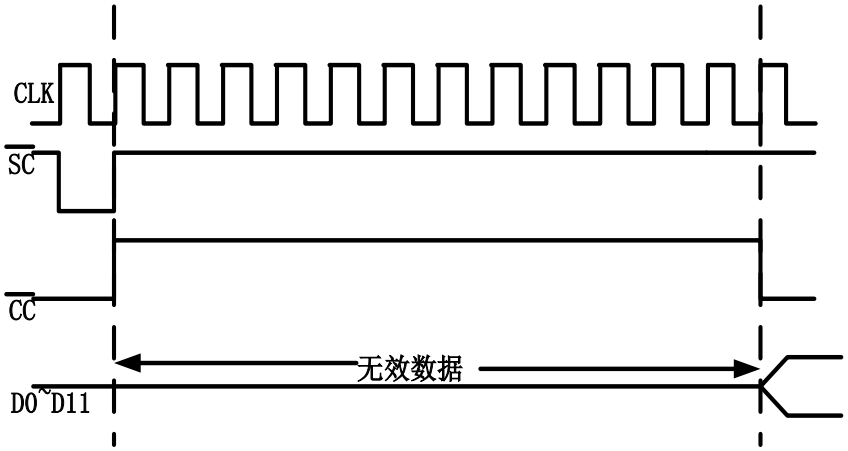
软件延时的例子：
```
for(int i=0;i<0x10000;i++);
```

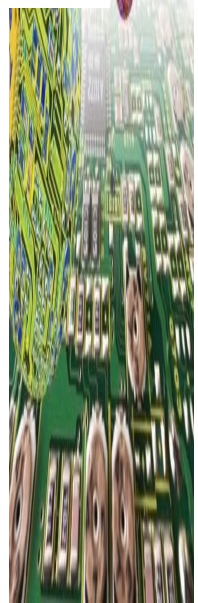
$$d = (N_a + N_c + N_j) * T * M$$

- 执行加法指令所需的时钟周期个数为N_a，
- 执行比较指令所需的时钟周期个数为N_c，
- 执行条件跳转指令所需的时钟周期个数为N_j，
- 微处理器时钟周期为T，
- 循环次数为M

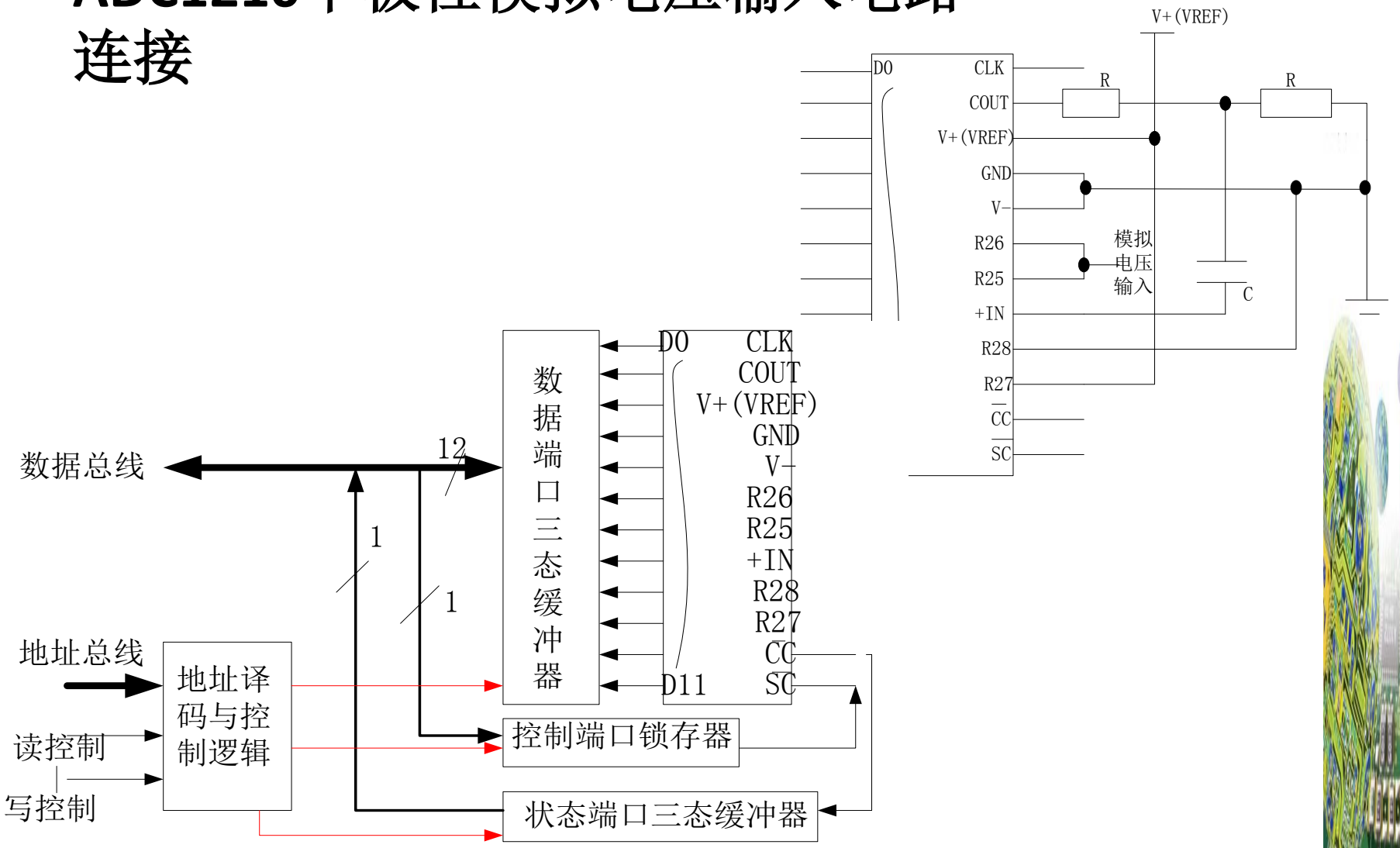
AD转换器ADC1210接口



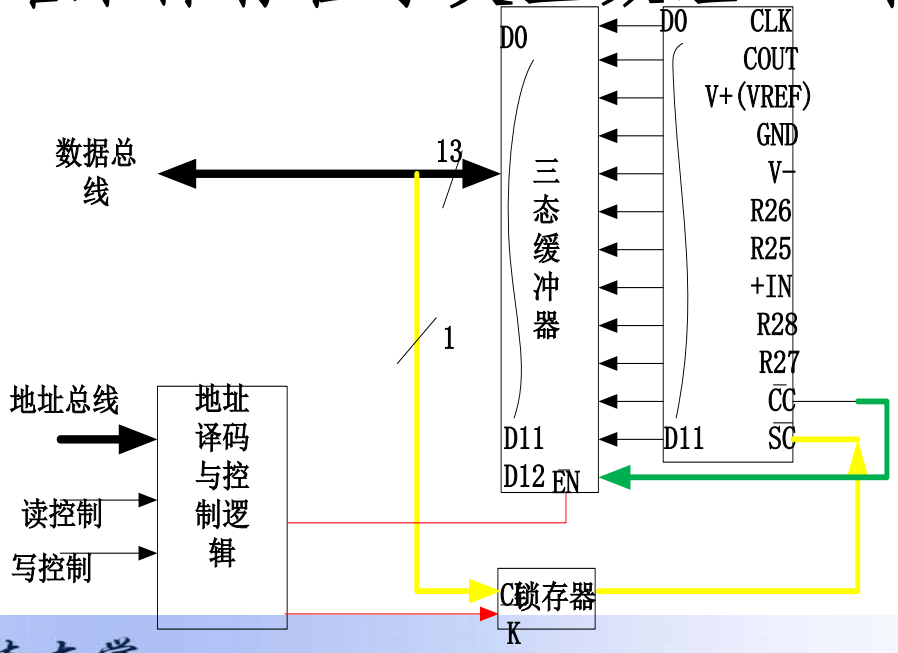




ADC1210单极性模拟电压输入电路连接



- 查询方式设计ADC1210与32位MicroBlaze微处理器的接口电路原理图，要求该接口电路仅占据一个IO端口地址，且其地址为0x80000000。并基于Xilinx C语言编写控制该接口电路转换100个数据，且将转换结果保存在字类型数组volt中的程序段

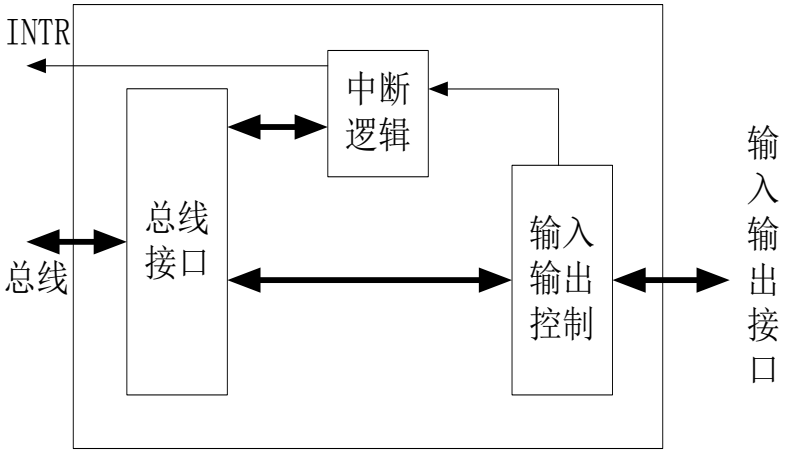


控制程序段

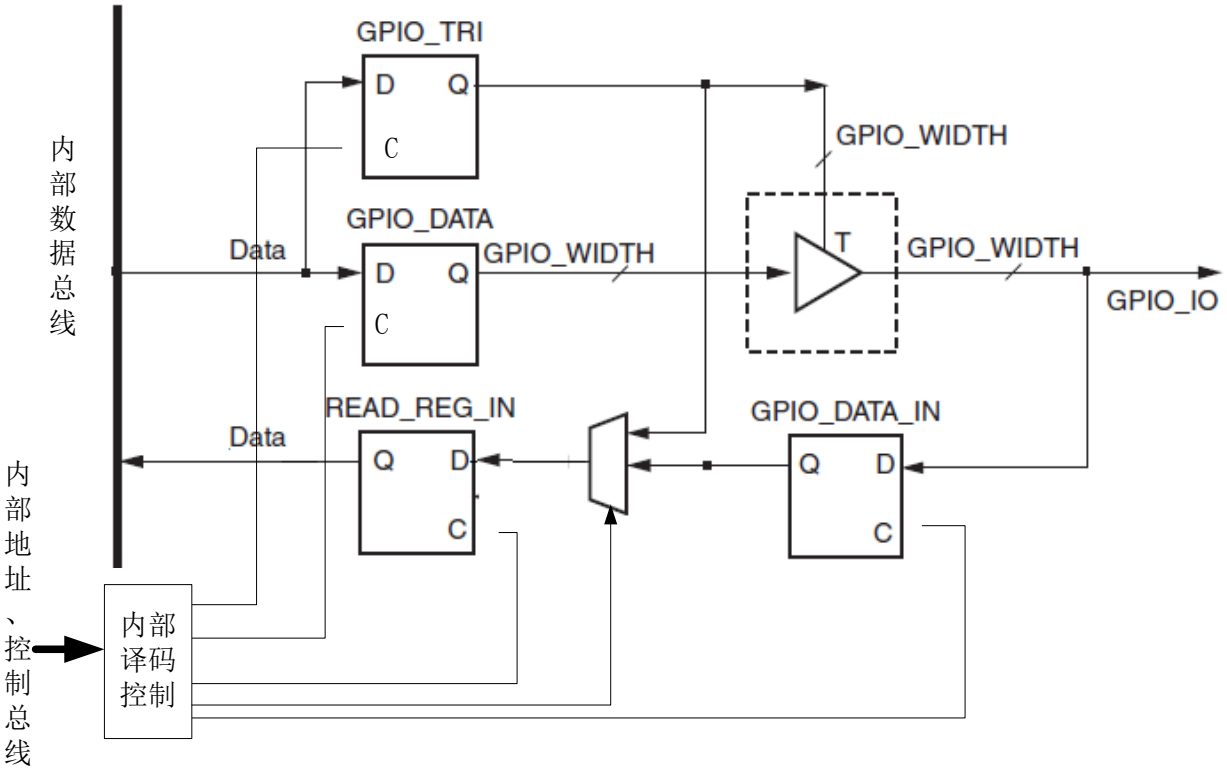
```
int volt[100]， i;
int port=0x80000000;
for(i=0;i<100,i++)
{
Xil_Out8(port,0x01);//输出数据使得SC为高电平
Xil_Out8(port,0x00);//输出数据使得SC为低电平
Delay;//延时一个ADC1210的时钟周期
Xil_Out8(port,0x01);//输出数据使得SC为高电平，从而产生启动转换信号
While ((Xil_In16(port)&0x1000)!=0);//查询状态
Volt[i]=Xil_In16(port)&0x0fff;//读取转换结果，且仅保留低12位有效数据
}
```

GPIO控制器

- GPIO（general purpose IO）是通用并行IO接口的简称。它将总线信号转换为IO设备要求的信号类型，实现地址译码、输出数据锁存、输入数据缓冲的功能

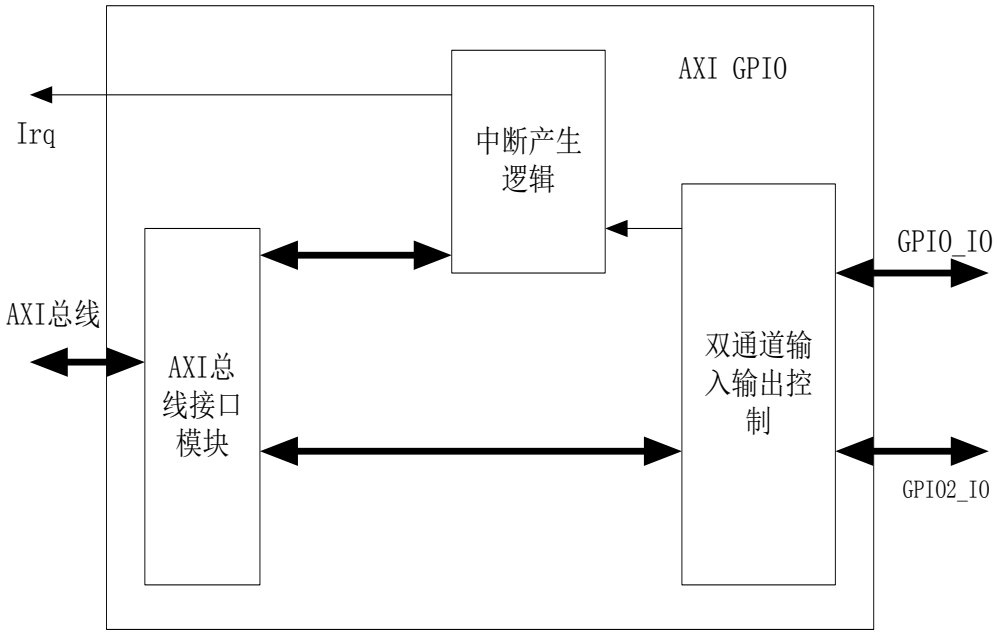


输入输出控制模块内部框图



Xilinx AXI总线GPIO IP核

- 每个通道都可以支持1~32位的数据输入输出，可以配置为单输入、单输出或双向输入输出

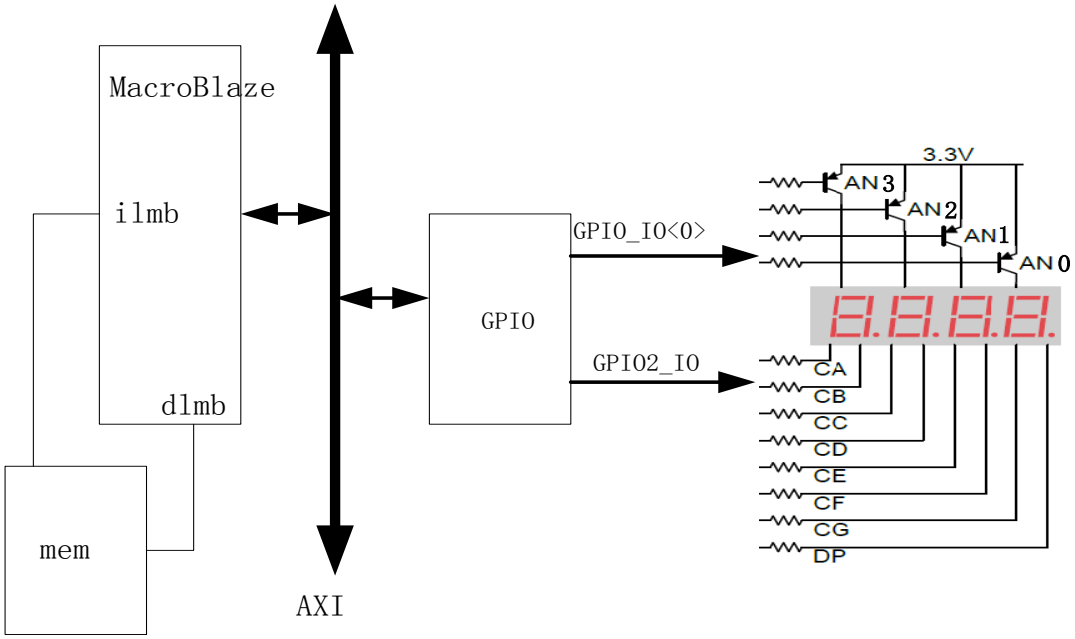


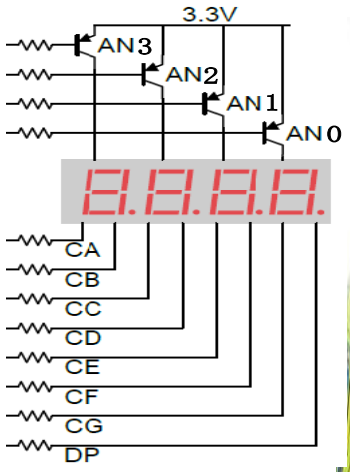
GPIO内部寄存器

寄存器名称	偏移地址	初始值	含义	读写操作
GPIO_DATA	0X0	0	通道1数据寄存器	通道1数据
GPIO_TRI	0X4	0	通道1三态控制寄存器	写控制通道1传输方向
GPIO2_DATA	0X8	0	通道2数据寄存器	通道2数据
GPIO2_TRI	0XC	0	通道2三态控制寄存器	写控制通道2传输方向

当GPIO_TRI某位为0时，GPIO相应的IO引脚配置为输出；
当GPIO_TRI某位为1时，GPIO相应的IO引脚配置为输入

采用GPIO控制器控制该四位7段数码管显示字符PASS

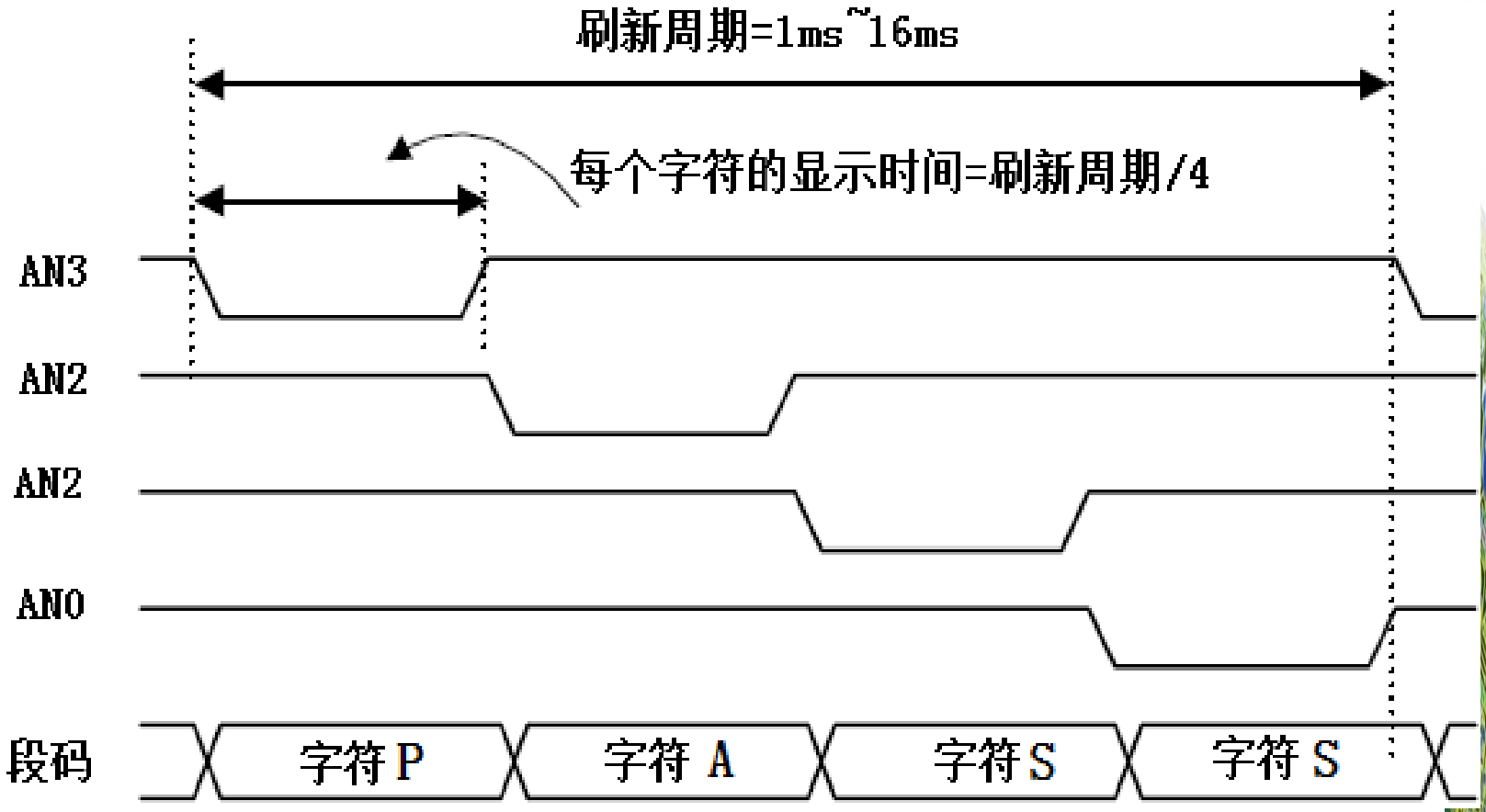




GPIO_IO<0>连接AN0，GPIO_IO<1>连接AN1，
GPIO_IO<2>连接AN2，GPIO_IO<3>连接AN3；
GPIO2_IO<0>连接CA，GPIO2_IO<1>连接CB，
GPIO2_IO<2>连接CC，GPIO2_IO<3>连接CD，
GPIO2_IO<4>连接CE，GPIO2_IO<5>连接CF，
GPIO2_IO<6>连接CG，GPIO2_IO<7>连接DP。

- GPIO控制器的基地址为XPAR_SEG_0_BASEADDR，那么GPIO_DATA地址为XPAR_SEG_0_BASEADDR，GPIO2_DATA地址为XPAR_SEG_0_BASEADDR+0x8，GPIO_TRI地址为XPAR_SEG_0_BASEADDR+0x4，GPIO2_TRI地址为XPAR_SEG_0_BASEADDR+0xC。
- 字符串“PASS”的段码为：0x8c,0x88,0x92,0x92;4位7段数码管从左到右的位码分别为0xf7，0xfb，0xfd，0xfe。

4位7段数码管显示PASS的控制时序

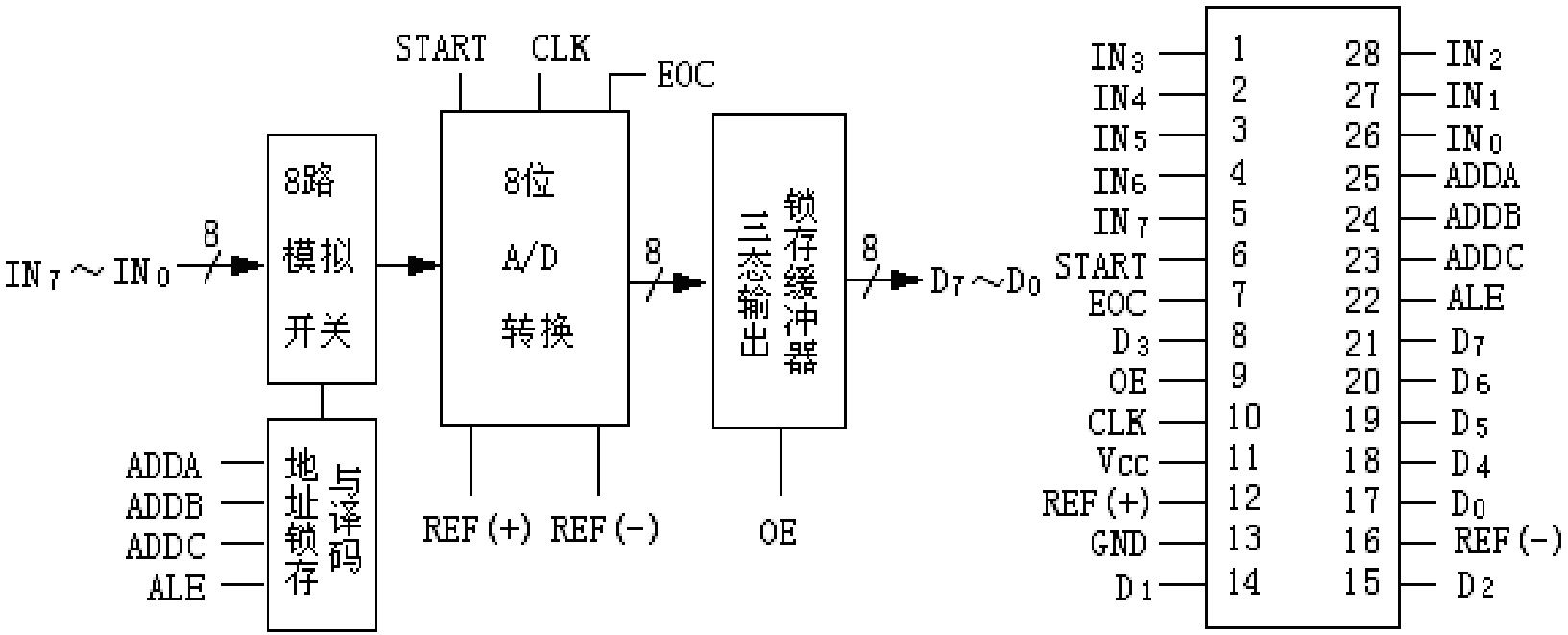


控制程序段

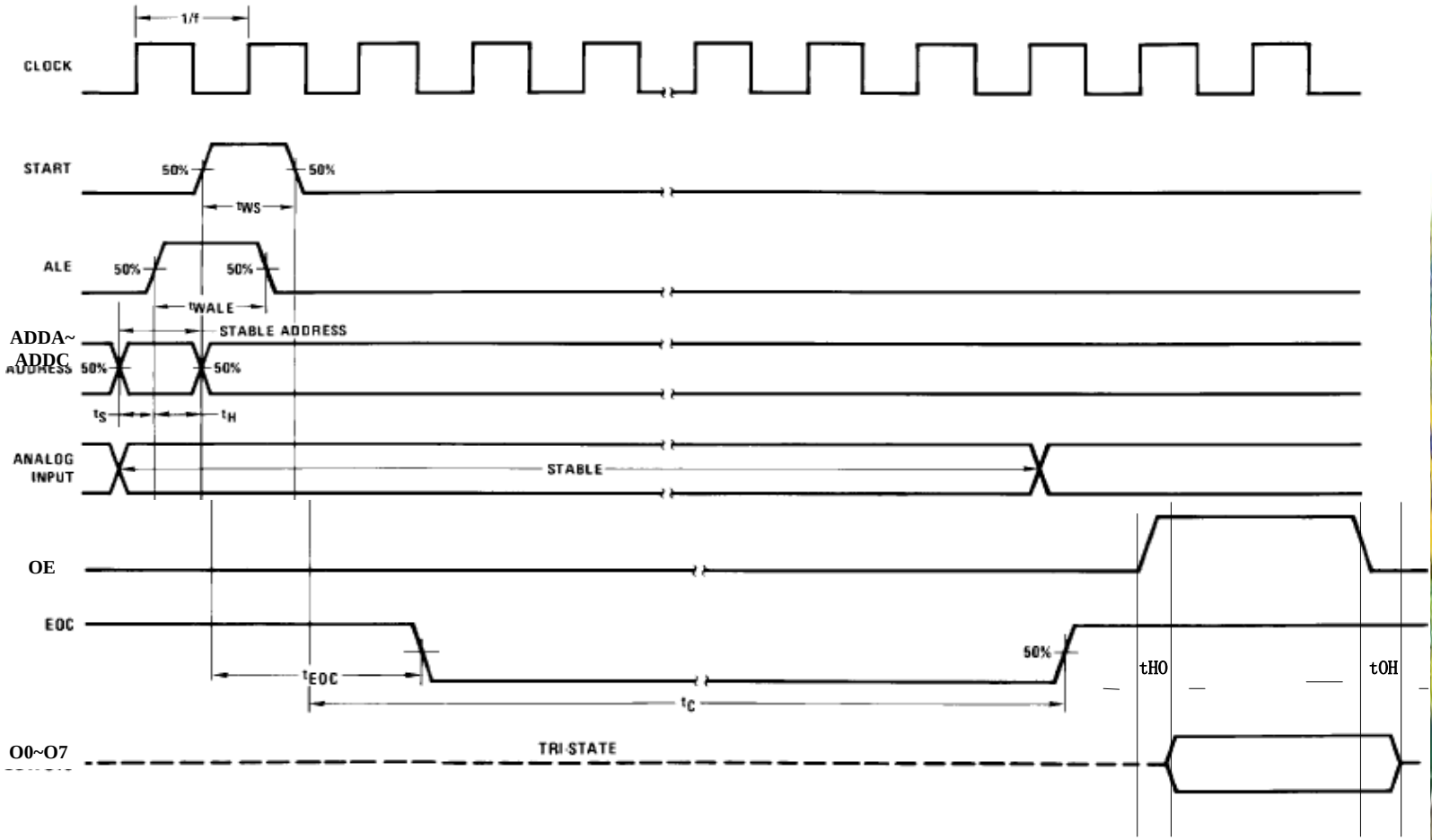
```
char segcode[4]={0x8c,0x88,0x92,0x92};
char pos=0xf7;  //初始化位码
int i,j;
Xil_Out8(XPAR_SEG_0_BASEADDR+0x4,0x0);  //使GPIO_IO通道输出
Xil_Out8(XPAR_SEG_0_BASEADDR+0xc,0x0);  //使GPIO2_IO通道输出
while(1){
        for(i=0;i<4;i++){
                Xil_Out8(XPAR_SEG_0_BASEADDR,pos);  //输出位码
                Xil_Out8(XPAR_SEG_0_BASEADDR+0x8,segcode[i]);//输出段码
                for(j=0;j<10000;j++); //延时，使人眼能正确的看到显示的字符
                pos=pos>>1;
                }
                pos=0xf7;
}
```

- GPIO控制器可以解决微处理器通过AXI总线与简单输入输出设备之间的通信问题
  - 独立开关
  - 独立发光二极管
  - 矩阵式键盘、
  - 7段数码管
  - ADC1210
- GPIO连接的外设数据、地址、控制以及状态信息相对独立，数据传输没有复杂的时序要求
- 若有复杂的时序要求，GPIO能否胜任?

A/D芯片ADC0808/ADC0809



ADC0808时序

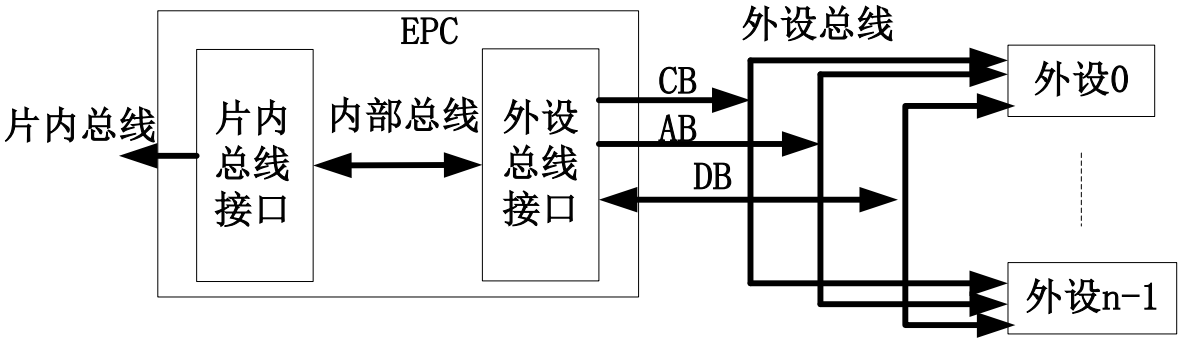

1/f
CLOCK
START
50%
50%
tWS
ALE
50%
50%
tWALE
STABLE ADDRESS
ADDA~
ADDC
ADDRESS
50%
50%
tS
tH
ANALOG
INPUT
STABLE
OE
EOC
50%
tHO
tOH
tEOC
tC
O0~O7
TRI-STATE

ADC0808时序参数

参数名称	含义	最小值	典型值	最大值
tWS	START信号脉宽		100ns	200 ns
tWALE	ALE信号脉宽		100 ns	200 ns
tS	地址信号有效到ALE信号上升沿的时间		25 ns	50 ns
tH	ALE上升沿之后地址信号维持有效的时间		25 ns	50 ns
tc	START信号下降沿到EOC信号有效的时间	90μs	100μs	116 μs
tEOC	START信号上升沿到EOC信号失效的时间	0	8/f+2us	
f	时钟频率	10 kHz	640 kHz	1280 kHz
tOH	OE下降沿到数据维持有效的时间		125 ns	250 ns
tHO	OE上升沿到数据输出有效的时间		125 ns	250ns

外设控制器（EPC）——自学

- 外设控制器是将片内总线转换为外设外部并行总线信号的一种接口控制器。它一方面实现总线信号转换，另一方面还实现高位地址译码，产生外设片选信号，并实现时序控制。



作业

- 12.GPIO控制器，要求采用Nexys4板实验验证（控制程序作业本提交，16周实验课验收）
  - 8.采用某8位独立开关输入十六进制字符（0~9，a~f）的ASCII码，并将该ASCII码表示的十六进制字符通过一位七段显示器显示出来。
  - 9.设计一监视2台设备状态(switch代替）的接口电路和监控程序：若发现某一设备状态异常(由低电平变为高电平)，则发出报警信号(指示灯亮)，一旦状态恢复正常，则将其报警信号撤除。
  - 10.用8个理想开关输入二进制数，8只发光二极管显示二进制数。设输入的二进制数为原码，输出的二进制数为补码。
  - 11.用4个7段数码管显示数字1，2，3，4，且仅占用两个端口地址。