一、实验室名称： 基础实验楼 331

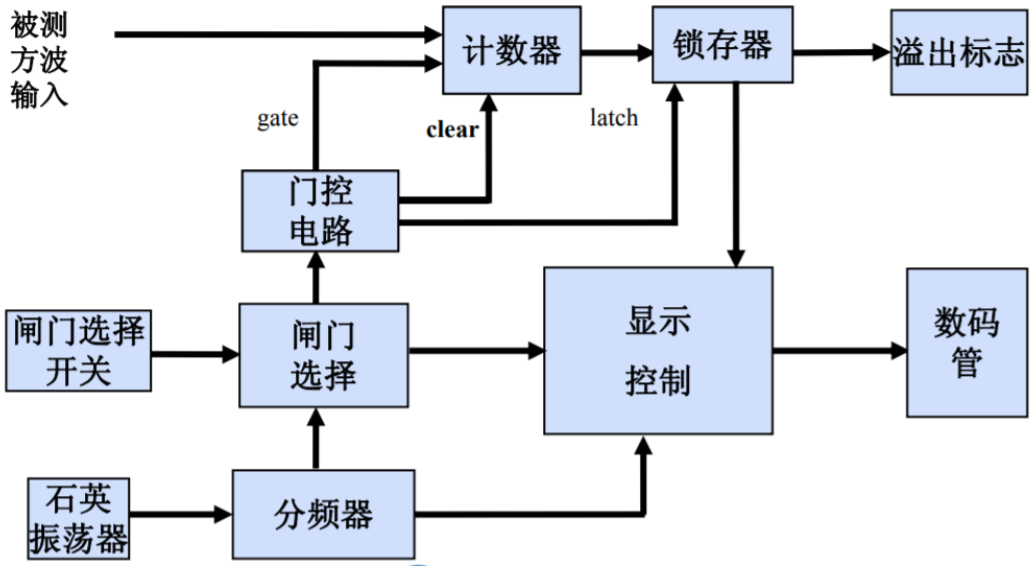
二、实验项目名称：现代电子技术综合实验

三、实验学时：48

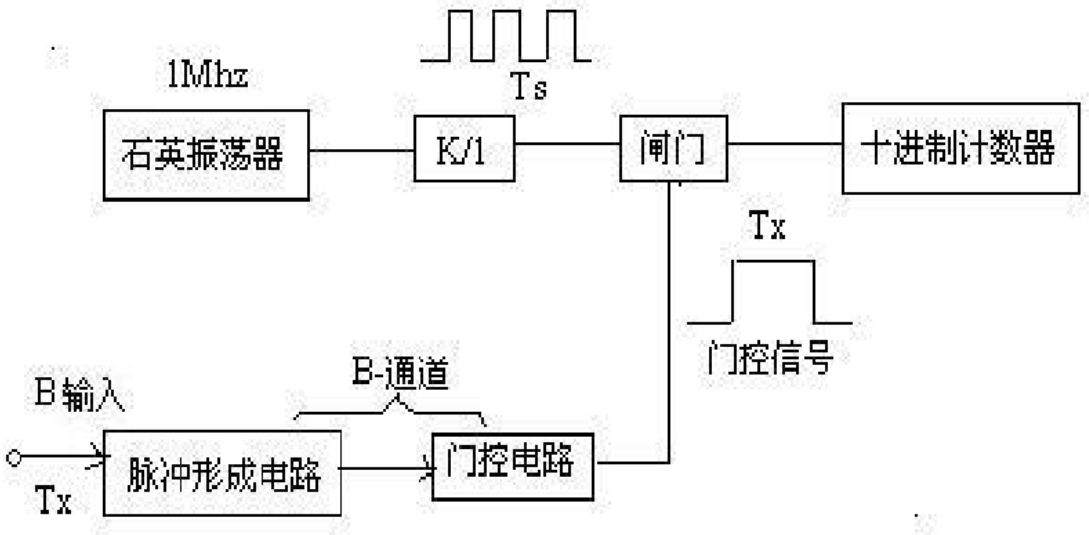
四、实验原理：

1. 测频原理

测频法：在确定的闸门时间 *TW* 内，记录被测信号的变化周期或脉冲个数 *NX*，则被测信号的频率 $fx=NX/TW$ 。其工作原理如下图所示，首先被测信号①经过放大整形后转变为方波信号②，将其加到闸门的输入端。由一个高稳定的晶体振荡器和数字分频器组成时基信号发生器，它输出时基信号③去控制门控制电路形成门控信号④，其控制闸门的开与闭，只有闸门在开通时间内，方波信号②才能通过闸门成为被计数的脉冲⑤，通过计数器计数。

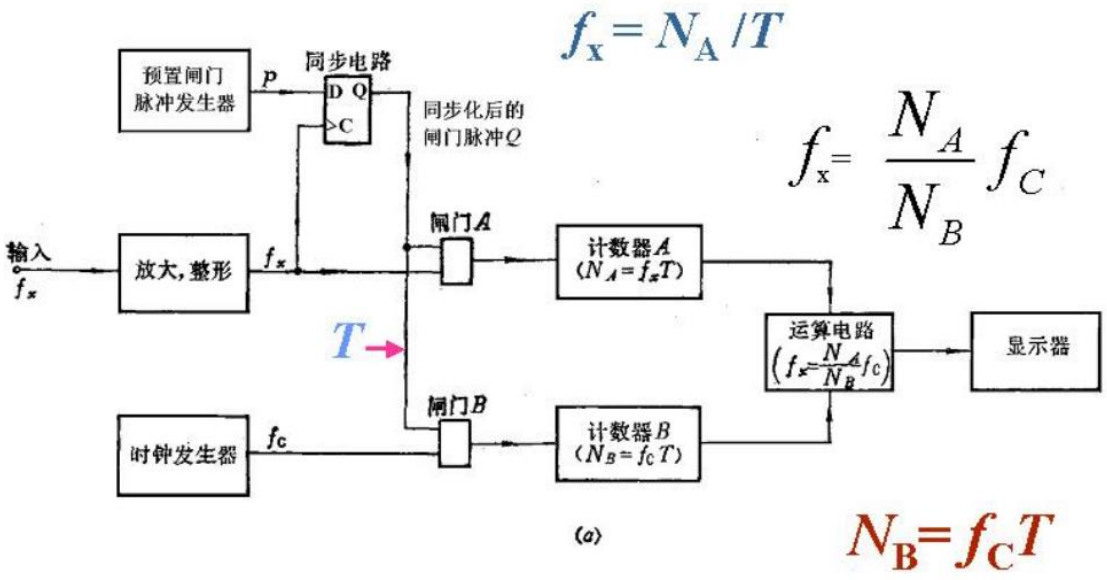


2. 测周原理



3. 等精度测量

等精度频率测量也称为多周期同步测量，与传统的频率测量原理相比，其优点是可在整个测频范围内获得同样高的测试精度和分辨率。其测量原理如图 1 所示，其工作时间波形图如图 2 所示。其中，fx 为输入信号的频率，f0 为基准信号的频率。A、B 2 个计数器在同一个闸门时间 T 内分别对 fx 和 f0 进行计数，计数器 A 的计数值 Nx=fxT，计数器 B 的计数值 NO=fOT。因此，被测信号的频率 fx 公式为：



五、实验目的：

1. 掌握频率检测的方法，学会查阅资料并分析误差结果。

2. 学习分析和设计系统需求并提出结局方案的过程。

3. 学习 MC8051 原理及用其实现部分功能的方法。

4. 学习 FPGA 和 VHDL 语言及用其实现频率计等功能的能力

六、实验内容：

5. 理解计数式频率计基本原理

6. 掌握 VHDL 语言的基本应用

7. 掌握 ISE,Modelsim 等 等 等 EDA 工具的使用流程

8. 掌握 MCU 及 及 C C 语言的基本应用 语言的基本应用

9. 掌握基本逻辑模块的设计

10.掌握小型电路系统的仿真

七、实验器材（设备、元器件）： ISE 、Modelsim 和 KEIL 软件、Smart SOPC 多功能教学实验开发平台主板、数字信号发生器

八、实验过程：

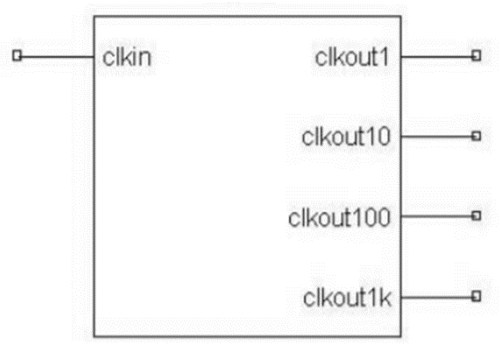
1. EDA

1）分频器：

对石英振荡器产生的信号进行分频，得到 1Hz、10Hz、100Hz 和 1KHz 基准频率，提供标准闸门时间控制信号以精确控制计数器的开闭；同时将 1KHz 的信号作为扫描显示译码模块的时钟，以产生扫描选择信号。

分频器的功能是由于闸门时间只有 1S，0.1S，0.01S 三档，且

在数码管显示时采用动态扫描的方法，需要产生 1kHz 的扫描信号，由于本设计将下载到开发板上，其提供的标准时间是 50MHz，所以要对系统的 50MHz 时钟信号进行分频，以产生符合要求的各频率信号：先由系统时钟

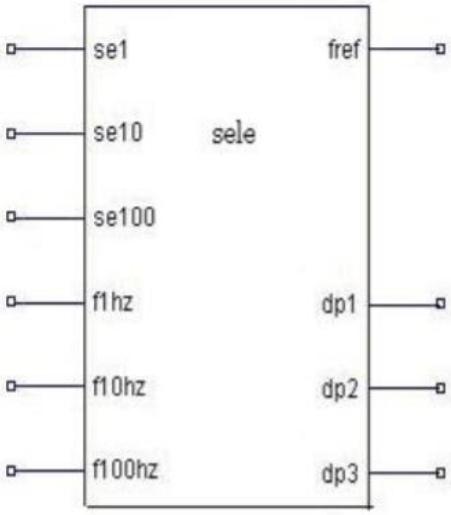


50MHz 分频出 1kHz 作为数码管显示的动态扫描信号，同时产生 0.01S 的计数闸门信号脉冲，再由 1kHz 分频出 100Hz 产生 0.1S 的计数闸门信号脉冲，由 100Hz 分频出 10Hz 产生 1S 的计数闸门信号脉冲。

2）闸门选择：

实现对输入的几个闸门信号的手动选择，将选择的闸门信号有 fref 输出到下一个模块，同时输出小数点的控制信号 dp1。

采用数字电路里所学的多路复用器原理，s(2:0)为选择端，f1,f10,f100 为被选时基信号输入端，通过选择端 s 来控制哪一个时基型号被选择输出：当 s(2:0)为 100 时，f1 的输入时基信号被选中，被赋值给输出端口 fref 输出，此时 dp1(2)有效，其余的无效，点亮对应连接的小数点；当 s(2:0)为 010 时，f10 时基信号被选中，被赋值给输出端口 fref 输出，此时 dp1(1)有效，其余的无效，点亮对应连接的小数点；最后当 s(2:0)为 001 时，f100 时基信号被选中，被赋值给输出端口 fref 输出，此时 dp1(0)有效，其余的无效，点亮对应)连接的小数点。



3）门控电路：

控制整个频率计各模块进行时序工作的控制装置，它对输入的标准时钟信号进行变换，产生我们所需要的三个信号闸门信号 gate，锁存信号 latch 以及清零信号 reset。

测量过程中，产生的上个闸门信号有着先后顺序。根据我们的 测量需要，可以得出三个信号的先后为：gate、latch、reset。同时为了规避冒险，我们需要将三个信号的上升沿错开。

4）计数器：

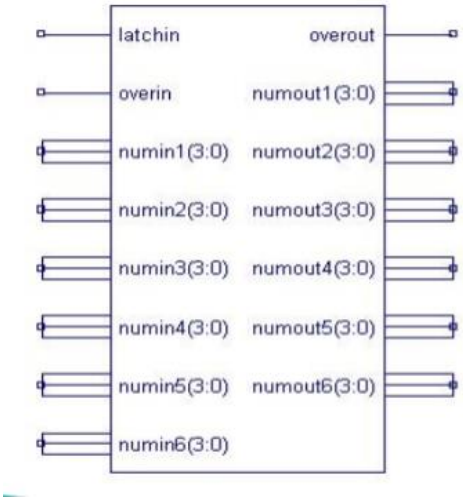
在门控电路的 gate 的控制下对输入的信号进行计数，即完成 f=N/T 公式中的 N 的测量。

实现：采用数字电路所学的十进制计数器，并对计数器采用级联的方 法

扩展为 6 位十进制计数器。

5）锁存器：

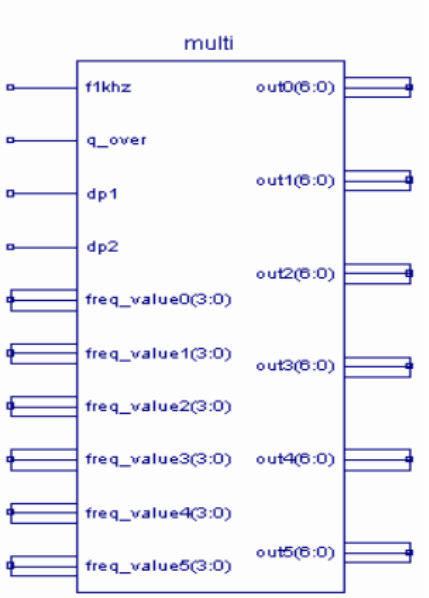
对计数器送来的六位计数结果和溢出信号 ove_out 进行锁存，保证显示数码管的数字不会跳动，方便读取数据。



6）扫描显示：

对锁存器送来的数据进行动态扫描（即位选）、将六位计数结果分别翻译成七段数码管所能识别的数据（即段选）。

实现：通过用一个频率 1KHz 的信号来扫描一个多路选择器，实现对六位已经锁存的计数结果的扫描输出，由于 1KHz 相对了人眼的暂留效应已经很高了，所以显示结果不会让人感觉到闪烁。这样就可以用一个译码器来实现对六个 4 位二进制数的译码。译码结果再连接到一个多路选择器的输入端，同样由 1KHz 的信号来同步扫描选通。实现最终结果的数字显示。六位十进制数的 BCD 码经 7 段译码输出，显示十进制数。并根据档位信号确定小数点显示的位置以实现档位的显示,最后的输出全部通过下载前的固定引脚连接到 LED 显示管上。



2. 51 单片机

整形后的被测信号通过计数器 T0 的外部接口（DCMotorSpeed）输入，由单片机发起标准的计数启、停控制，看通过按键或拨码开关切换设置计数闸门控制宽度为 1s、100ms、10ms。其中，闸门时间可直接用定时器 T1 实现。s 的闸门可用软件定时或多次使用定时器 T1 来实现。已知单片机主频，根据在闸门内的计算结果可算得被测频率值。

本次实验由于时间原因未完成自动切换量程等功能，采用三个档位，用拨码开关切换，采用测频法。

九、实验数据及结果分析：

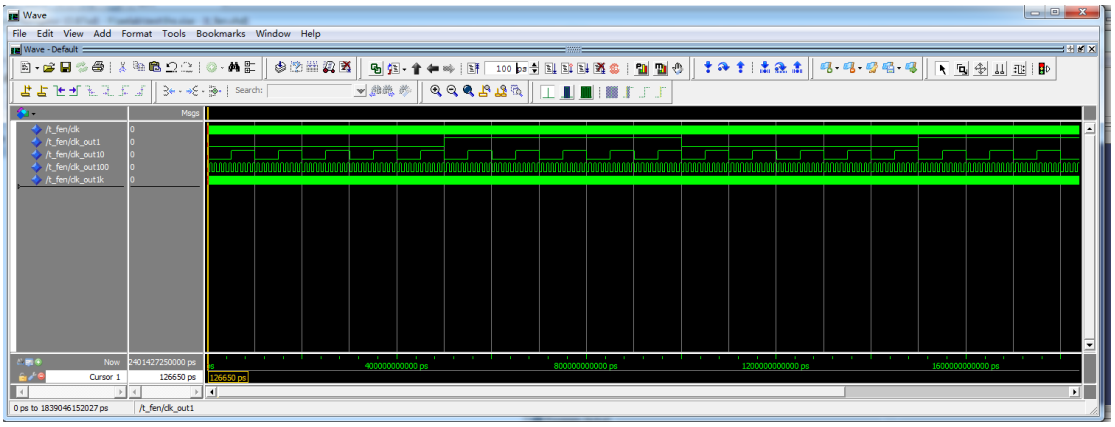
1. EDA 频率计

采用测频法 ,设有三个档位可以手动切换，闸门时间分别为 1s，0.1s，0.01s，量程在 1—10Mhz。实现无意义零的消隐。输入测试信号为占空比 50%的脉冲信号是，在低频段（1khz 以下），三个档位均可可以准确测量，绝对误差小于 1hz。在频率高于 100khz 时，出现误差，当闸门时间选为 0.01s 时，小数点后一位出现误差。闸门为 1s 时误差依然小于 1hz。测量频率更高时闸门时间为

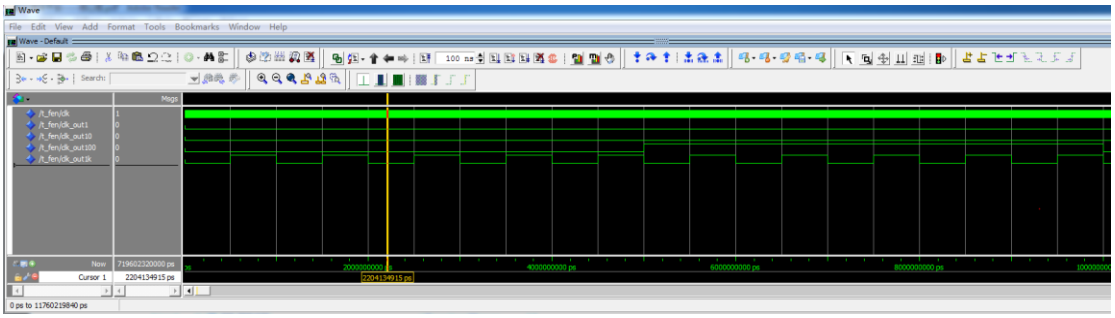
0.1s 时最后一位（十位）也出现误差，当闸门时间选为 0.01s 时依然是小数点后一位出现误差。测量精度在要求频段内可以达到千分之一。

测频方法的误差主要是通过闸门时可能误检到±1 个脉冲通过，闸门时间越长相对误差越小，频率越高相对误差越小。 进一步提高测量精度可以考虑采用等精度的测量方法。在频率很高时，晶振信号源的误差也会变得不可忽略，但是本设备 10Mhz 以内影响不明显。此外，还可以对 BCD 码转换的除法等运算过程进行优化以加快处理响应速度。

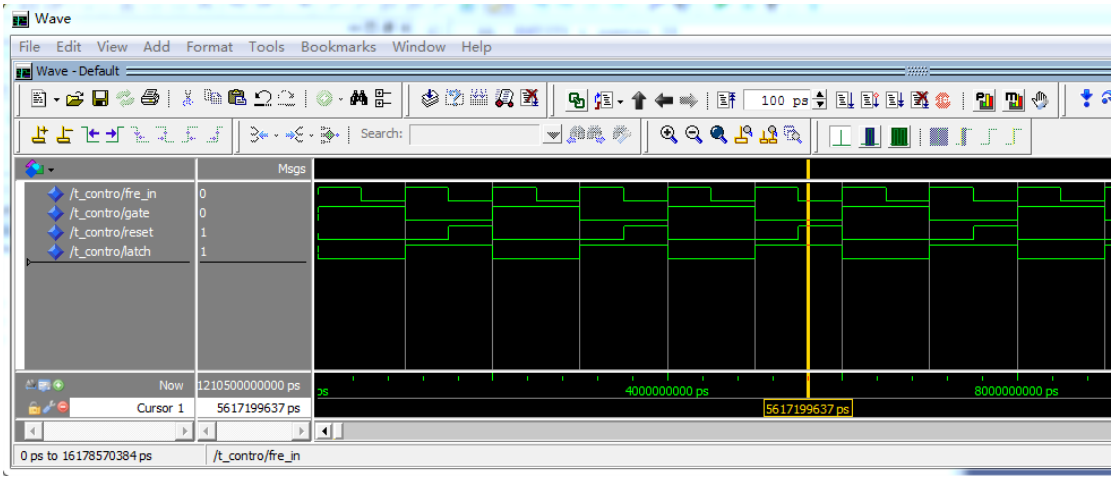
各模块仿真结果如下



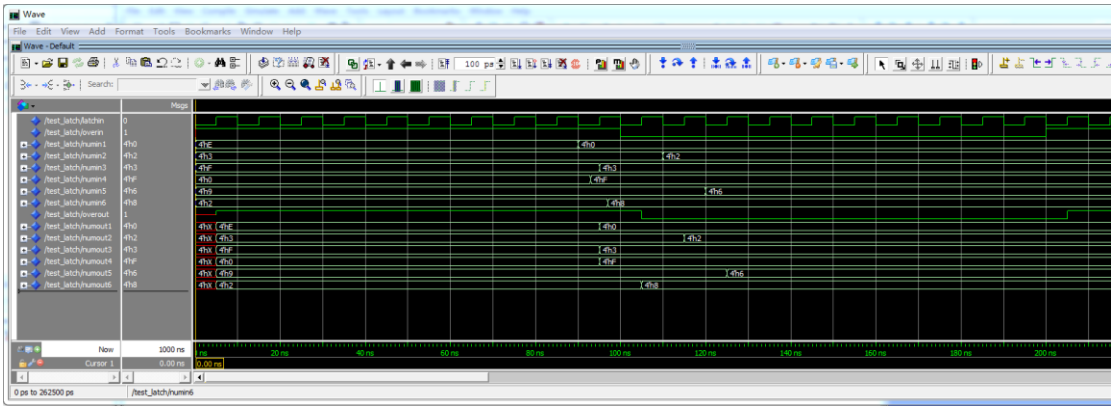
1hz 和 10hz 分频



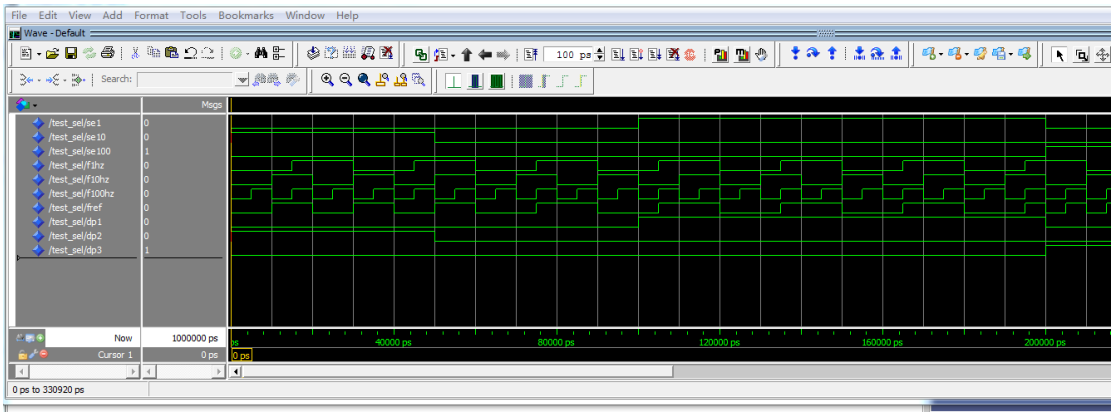
1khz 分频



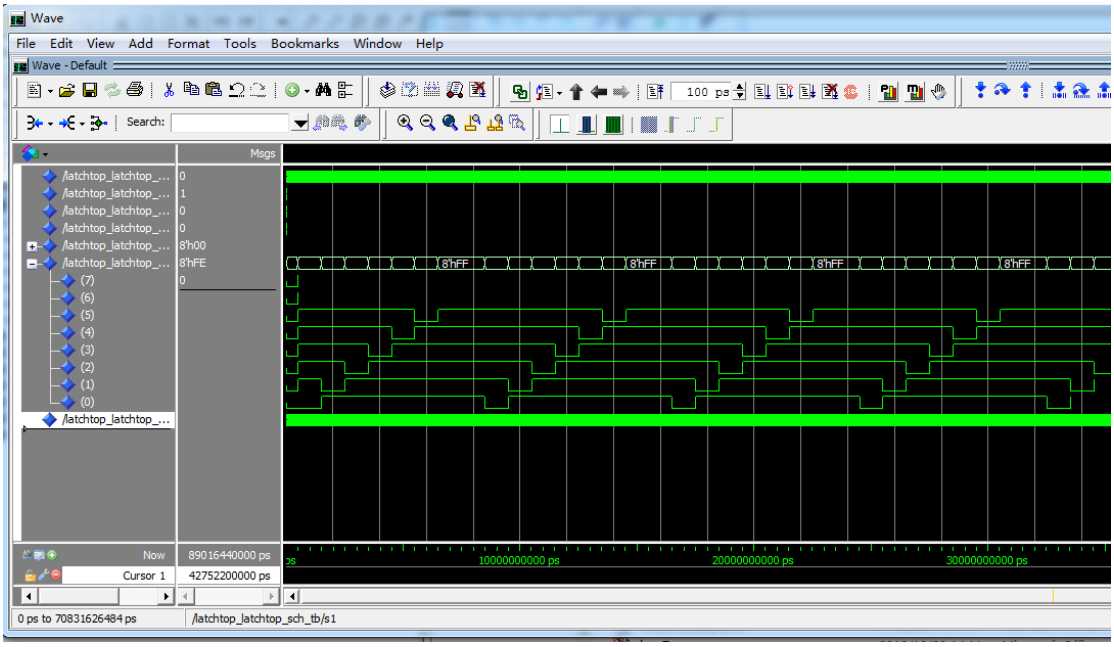
闸门控制模块



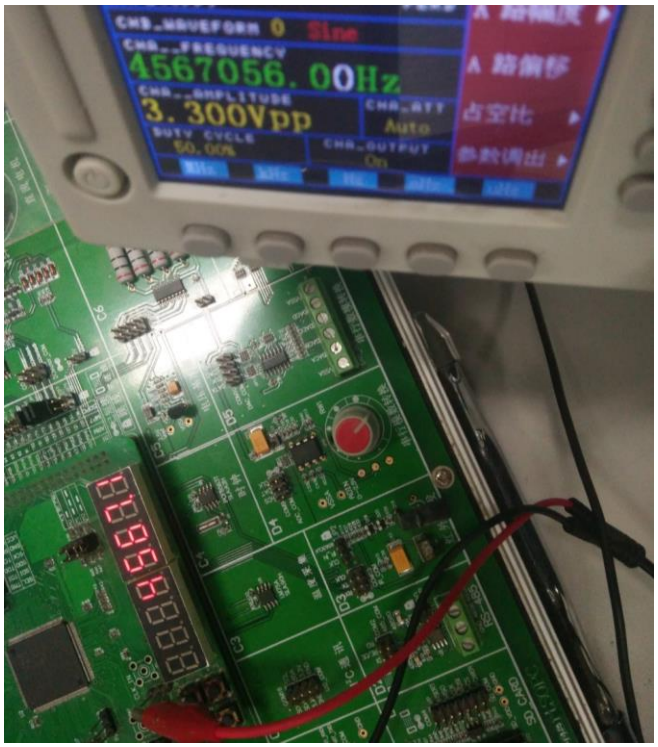
锁存器



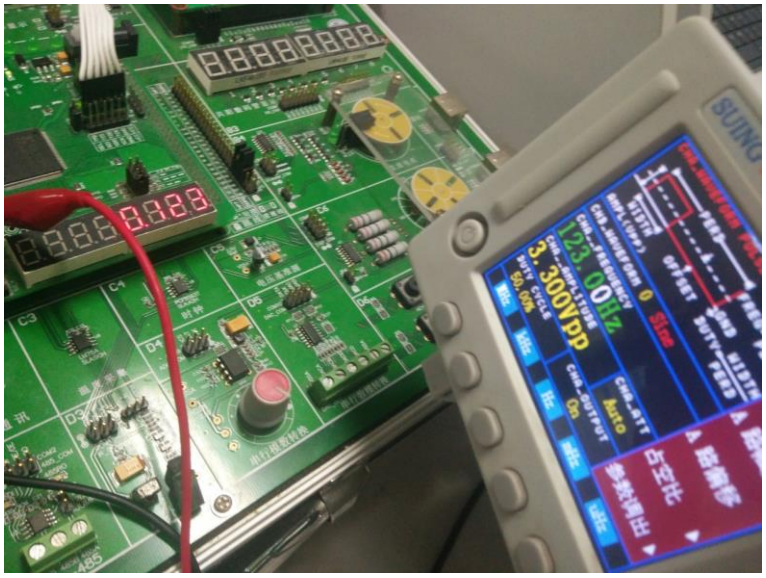
选择模块



整体仿真



EDA 实测（较高频率）



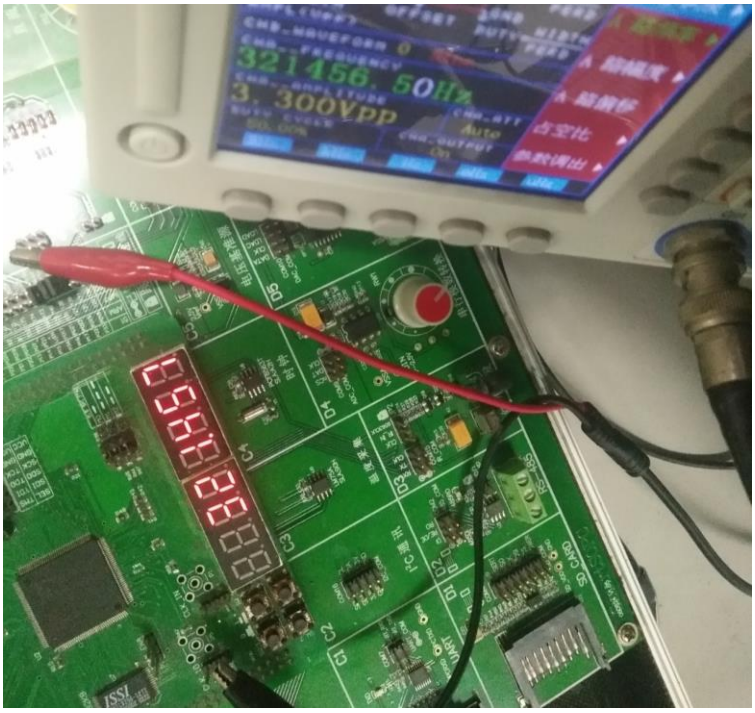
EDA 实测（较低频率）

2. C51 频率计

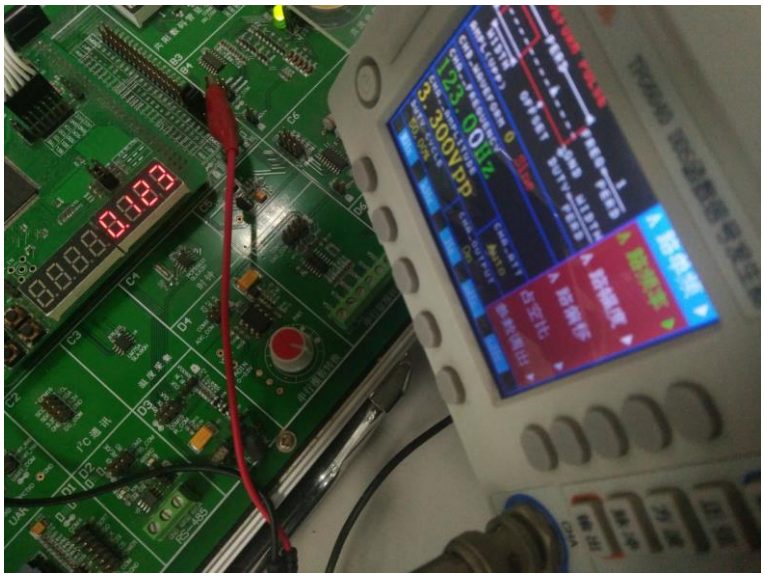
同样采用三个档位，用拨码开关切换，采用测频法，理论计算得到的频率在高频段有较大误差，经实际测试发现误差和实际输入值呈近似线性关系，所以改变系数进行校准，校准后低频段 1000hz 以下显示数字和函数发生器测试信号相同，输入信号频率在 300khz 时，闸门时间为 1s 和时最后一位存在误差，和函数发生器显示的频率不同。在输入信号高于 400khz 时非线性影响较强，进一步改进可考虑分段校准。

理论上该单片机可以测量上限在 700khz 以上，实际只能测到 400khz 可能是由于信号的较高频段采样时失真较大，丢失高频信号所致。进一步提高测量范围可以考虑提高最大工作频率，工作频率越高，N0 的±1 计数误差的相对值 ±1/N0 越趋于 0。输入高频信号时，对输入信号进行预分频也可以提高频率分辨率。

C51 实测图如下：



c51 实测（较高频率）



C51 实测（较低频率）

十、总结及心得体会：

本实验在为期两周的时间内进行了 FPGA 和 51 单片机的频率计制作，实现了量程切换、数显、无意义零的消隐等功能。FPGA 频率计可以实现要求的 1-10Mhz 的测量量程和精度。单片机的精度低于 FPGA，

且在高于 400khz 有较大误差，由于时间有限和个人能力不足，未进行量程自动切换和更高精度的改进实现以及尝试等精度测量方法。本次实验也没有使用液晶屏代替数码管以及对中断进行深入了解，未能完整利用开发板的性能，对于数码管的扫描和锁存也没有完全利用整个扫描周期来使数码管显示的更加清晰。

在这次实验中，了解单片机和 FPGA 各自的特点，培养了工程能力和解决问题、积极探索的意识。本次实验还体验了自顶向下分解问题和设计的模式。由于时间原因，本次实验也没有采用 SOPC 的方法制作频率计，希望以后有机会可以进一步了解 SOPC。 以后面对问题时，能够更加主动地探索和寻求解决方案。最后感谢实验室和老师们提供的平台和支持。

十一、对本实验过程及方法、手段的改进建议：

希望能够延长这种实验课的课时或者实验室开放时间，以便做出更加完善的作品。

基础实验楼的开水间没有品学楼和立人楼那样的烧水机，如果后勤还有经费，希望可以每层楼放一台。

附录：关键部分主要代码

Ｃ５１：

```
#include <reg51.h>
#include <absacc.h>
#include <ctype.h>
```

```
//定义按键
sbit KEY2 = P2^0;       //选择 1 档位
sbit KEY3 = P2^1;       //选择 2 档位
sbit KEY4 = P2^2;       //选择 3 档位

sbit LED1 = P0^0;
sbit LED2 = P0^1;
sbit LED3 = P0^2;

//定义显示缓冲区（由定时中断程序自动扫描）
unsigned char DispBuf[8];
unsigned long T0_count=0;      //计数器 T0 溢出次数
unsigned char key_num=2;              //闸门序号
unsigned long dp_num=0;               //小数点位置
unsigned char k;           //定义键值变量

unsigned char KeyScan()                    //键盘扫描
{
    unsigned char k = '\0';
    if ( KEY2== 0 )       k='1';
    if ( KEY3== 0 )       k='2';
    if ( KEY4== 0 )       k='3';
    return k;
}

//设置 T0 中断
void T0INTSVC() interrupt 1
{
    T0_count++;
}

void DispClear()
{
    unsigned char i;
    for ( i=0; i<8; i++ )
    {
        DispBuf[i] = 0x00;
    }
}

/*
函数：ByteToStr()
功能：字节型变量 c 转换为十进制字符串
```

```
*/

void ByteToStr(unsigned char idata *s, unsigned long c)
{
    code unsigned long Tab[] = {10000000,1000000,100000,10000,1000,100,10};
    unsigned char i;
    unsigned char t;
    for ( i=0; i<7; i++ )
    {
        t = c / Tab[i];
        *s++ = '0' + t;
        c -= t * Tab[i];
    }
    *s++ = '0' + c;
    *s = '\0';
}

/*
函数：DispChar()
功能：在数码管上显示字符
参数：
    x：数码管的坐标位置（0～7）
    c：要显示的字符（仅限 16 进制数字和减号）
*/

void DispChar(unsigned char x, unsigned char c )
{
    code unsigned char Tab[] =
    {//定义 0123456789AbCdEF 的数码管字型数据
        0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,
        0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71
    };
    unsigned char t;     //临时变量
//防止显示位置超出范围
    x&=0x07;
    x = 7-x ;
//分析字符 c，取得对应的数码管字型数据
    if ( c == '-' )
    {
        t = 0x40;
    }
    else
    {
        t = toint(c);     //toint()为库函数
```

```
        if ( t < 16 )     //如果是 16 进制字符
        {
            t = Tab[t];      //查表，取得数码管字型数据
        }
        else
        {
            t = 0x00; //如果是其它字符则显示为空白
        }
    }

//检查是否显示小数点
    if ( x==dp_num )
    {
        t |= 0x80;
    }
    else
    {
        t &= 0x7F;
    }

//送到显示缓冲区显示
    DispBuf[x] = t;
}

/*
函数：DispStr()
功能：在数码管上显示字符串
参数：
    x：显示的起始位置（0～7）
    *s：要显示的字符串（内容仅限 16 进制数字和减号）
*/

void DispStr(unsigned char x, unsigned char idata *s)
{
    unsigned char c;
    for (;;)
    {
        unsigned char temp1, temp2,temp3;
        c = *s++;
        if((x%8)==5) temp1=c;
        if((x%8)==4) temp1=c;
        if((x%8)==3) temp1=c;
        if ( c == '\0') break;
        else if ( c=='0' && ((x%8)>=(7-dp_num)))
```

```
            {DispChar(x++,'0');}
            //else if ((c=='0') && ((++c)!='0'
            else if (c=='0'&&(temp1=='0')&&(x%8)==4)
            {DispChar(x++,'+');}
            else if (c=='0'&&(temp1=='0')&&(x%8)==3&&(temp2=='0'))
            {DispChar(x++,'+');}
            else if (c=='0'&&(temp1=='0')&&(x%8)==2&&(temp2=='0')&&(temp3=='0'))
            {DispChar(x++,'+');}
            else if (c=='0')
            DispChar(x++,'+');

            //消影，清除无意义的零
            //{x++;continue;}
            else DispChar(x++,c);

        }
}

void DispInit()
{
    DispClear();   //初始为全灭
    EA = 0;
    TMOD = 0x15;
    TH1 = 0xFA;
    TL1 = 0x24;
    TR1 = 1;
    ET0 = 1;
    EA = 1;
}

void Delay(unsigned int t)
{
    do
    {
        code unsigned char com[] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
        static unsigned char n = 0;
        TH1 = 0xFA;
        TL1 = 0x24;
        TR1 = 1;
        while ( !TF1 );
        TR1 = 0;
        TF1=0;
        TH1 = 0xFA;
        TL1 = 0x24;
```

```
            TR1 = 1;
            XBYTE[0x7800] = 0xFF;            //暂停显示
            XBYTE[0x7801] = ~DispBuf[n];  //更新扫描数据
            XBYTE[0x7800] = ~com[n]; //重新显示
            n++;
            n&=0x07;
        } while ( --t != 0 );
}

/*
函数：SysInit()
功能：系统初始化
*/
void SysInit()
{

    TMOD = 0x15;      //设置定时器 T1 为 16 位定时器  T0 为计数器
    DispInit();           //数码管扫描显示初始化
}

void sele_gate(unsigned char k)
{
    switch ( k )          //判断键值，执行具体功能
        {
            case '1':
                LED1=0;                     //1s 1hz
                LED2=1;
                LED3=1;
                TR0 = 1;
                TH0 = 0x00;
                TL0 = 0x00;
                Delay(1000);
                TR0 = 0;
                break;

            case '2':
                LED1=1;                     //100ms
                LED2=0;
                LED3=1;
                TR0 = 1;
                TH0 = 0x00;
                TL0 = 0x00;
                Delay(100);
                TR0 = 0;
```

```
                break;

            case '3':
                LED1=1;                         //10ms
                LED2=1;
                LED3=0;
                TR0 = 1;
                TH0 = 0x00;
                TL0 = 0x00;
                Delay(10);
                TR0 = 0;
                break;

            default:
                break;
        }

}

void sele_dp()
{
    switch ( k )        //判断键值，执行具体功能
        {
            case '1':
                dp_num=3;
                break;

            case '2':
                dp_num=2;
                break;

            case '3':
                dp_num=1;
                break;

            default:
                break;
        }
}

unsigned long freq_final()
{
    unsigned long T0_cnt_final;  //定义 T0 计数器计算值
    unsigned long freq;      //定义频率变量
```

```
    T0_cnt_final=TH0*256+TL0+T0_count*65536;   //计算脉冲数
    T0_cnt_final=T0_cnt_final*0.9959964+0.6;    //使用频率计校准的修正

    freq=T0_cnt_final;
    sele_dp();

    T0_count=0;         //T0  置零，准备重新开始下一次计数
    return freq;
}

void main()
{
    unsigned char idata s[16];
    unsigned long freq = 5555;          //定义频率变量
    SysInit();                  //系统初始化
    ByteToStr(s,freq);
    DispStr(8,s);         //显示初始值
    Delay(200);
    for (;;)
    {
        for (;;)
        {
            Delay(10);
            k = KeyScan();      //键盘扫描
            if ( k != '\0' ) break;       //如果有键按下，退出循环
        }

            sele_gate(k);      //选择闸门
            freq = freq_final();//计算频率
            sele_dp();
            ByteToStr(s,freq);
            DispStr(8,s);   //显示计数器值


        for (;;)
        {
            Delay(10);
            if ( KeyScan() == '\0' ) break; //如果按键抬起，退出循环
        }
    }

}
```

ＥＤＡ：

分频：

```
entity fdiv is
GENERIC(N:Integer:=10);
Port
(fin:IN STD_LOGIC;
fout:OUT STD_LOGIC
);
end fdiv;

architecture Behavioral of fdiv is
    SIGNAL cnt:Integer RANGE 0 TO N:=0;
    SIGNAL clk:STD_LOGIC:='0';
begin
    process(fin)
    begin
    if rising_edge(fin) then
        if cnt /=N then
            cnt <= cnt + 1;
            else
                cnt <= 1;
                clk <= not clk;
            end if;
        end if;
    end process;
    fout <= clk;
end Behavioral;
```

计数：

```
entity counter is
port (rst,clk : in std_logic;
      carry_in    : in std_logic;
      carry_out : out std_logic;
      count_out         : out std_logic_vector(3 downto 0));
end counter;
architecture Behavioral of counter is
       signal count: std_logic_vector(3 downto 0):="0000";
begin
    process(rst,clk)
        begin
        if rst='1' then
            count <= "0000";
        elsif clk'event and clk= '1' then
```

```
                if carry_in = '1' then
                    if    count < "1001" then
                          count <= count+1;
                    else
                          count <= "0000";
                    end if;
                    else
                null;
                end if;
            end if;
        end process;
          count_out<=count;
          carry_out <= '1' when carry_in = '1' and count = "1001" else '0';
end Behavioral;
```

选择：

```
entity sele is
   port(se1,se10,se100,f1hz,f10hz,f100hz:in std_logic;
          fref,dp1,dp2,dp3:out std_logic:='0');
end sele;

architecture Behavioral of sele is
 signal sel:std_logic_vector(2 downto 0):="000";
begin
 sel<=se100&se10&se1;
 process(sel,f1hz,f10hz,f100hz)is
    begin
        if(sel="001") then
           fref<=f1hz;
            dp1<='1';
            dp2<='0';
            dp3<='0';
           elsif (sel="010") then
          fref<=f10hz;
            dp1<='0';
            dp2<='1';
            dp3<='0';
           elsif (sel="100") then
          fref<=f100hz;
            dp1<='0';
            dp2<='0';
            dp3<='1';
      else
          fref<=f1hz;
```

```
            dp1<='0';
            dp2<='0';
            dp3<='0';
        end if;
end process;
end Behavioral;
```

数显：

```
entity multi is
  port(f1khz,q_over,dp1,dp2,dp3:in std_logic;
        fre1,fre2,fre3,fre4,fre5,fre6:in std_logic_vector(3 downto 0);
          seg: out std_logic_vector(7 downto 0);
          dig: out std_logic_vector(7 downto 0));
end multi;

architecture Behavioral of multi is
signal seg_r:STD_LOGIC_VECTOR(7 DOWNTO 0):="00000000";
signal dig_r:STD_LOGIC_VECTOR(7 DOWNTO 0):="00000000";
signal fre_data: STD_LOGIC_VECTOR(3 DOWNTO 0):="0000";
signal count: std_logic_vector(2 downto 0):="000";
begin
    process(f1khz)is
      begin
        if rising_edge(f1khz) then
              count<=count+1;
        end if;

    end process;

    process(f1khz,dp1,dp2,dp3,count)is
     begin
--         if (f1khz'event and f1khz='1')then
           if rising_edge(f1khz) then
                case count is
                    when "000"=>   fre_data<=fre1;
                    when "001"=>   fre_data<=fre2;
                    when "010"=>   fre_data<=fre3;
                    when "011"=>   fre_data<=fre4;
                    when "100"=>   fre_data<=fre5;
                    when "101"=>   fre_data<=fre6;
                    when others=> fre_data<="0000";
                end case;

                if(fre6="0000")then
                    if(fre5="0000") then
```

```
                    if(fre4="0000")then
                        if(dp2='1')then
                            case count is
                                when "000"=>    dig_r<="11111110";
                                when "001"=>    dig_r<="11111101";
                                when "010"=>    dig_r<="11111011";
                                when "011"=>    dig_r<="11111111";
                                when "100"=>    dig_r<="11111111";
                                when "101"=>    dig_r<="11111111";
                                when others=> dig_r<="11111111";
                            end case;
                        elsif( dp3='1')then
                            if(fre3="0000")then
                                case count is
                                    when "000"=>    dig_r<="11111110";
                                    when "001"=>    dig_r<="11111101";
                                    when "010"=>    dig_r<="11111111";
                                    when "011"=>    dig_r<="11111111";
                                    when "100"=>    dig_r<="11111111";
                                    when "101"=>    dig_r<="11111111";
                                    when others=> dig_r<="11111111";
                                end case;

                        else
                            case count is
                                when "000"=>    dig_r<="11111110";
                                when "001"=>    dig_r<="11111101";
                                when "010"=>    dig_r<="11111011";
                                when "011"=>    dig_r<="11111111";
                                when "100"=>    dig_r<="11111111";
                                when "101"=>    dig_r<="11111111";
                                when others=> dig_r<="11111111";
                            end case;
--                          else
--                              case count is
--                                  when "000"=>    dig_r<="11111110";
--                                  when "001"=>    dig_r<="11111101";
--                                  when "010"=>    dig_r<="11111011";
--                                  when "011"=>    dig_r<="11110111";
--                                  when "100"=>    dig_r<="11101111";
--                                  when "101"=>    dig_r<="11011111";
--                                  when "111"=>    dig_r<="11011111";
--                                  when others=> dig_r<="01111111";
--                              end case;
```

```
                end if;
            else
                case count is
                    when "000"=>    dig_r<="11111110";
                    when "001"=>    dig_r<="11111101";
                    when "010"=>    dig_r<="11111011";
                    when "011"=>    dig_r<="11110111";
                    when "100"=>    dig_r<="11111111";
                    when "101"=>    dig_r<="11111111";
                    when others=> dig_r<="11111111";
                end case;
            end if;
        else
            case count is
                when "000"=>    dig_r<="11111110";
                when "001"=>    dig_r<="11111101";
                when "010"=>    dig_r<="11111011";
                when "011"=>    dig_r<="11110111";
                when "100"=>    dig_r<="11111111";
                when "101"=>    dig_r<="11111111";
                when others=> dig_r<="11111111";
            end case;
        end if;

    else
     case count is
            when "000"=>    dig_r<="11111110";
            when "001"=>    dig_r<="11111101";
            when "010"=>    dig_r<="11111011";
            when "011"=>    dig_r<="11110111";
            when "100"=>    dig_r<="11101111";
            when "101"=>    dig_r<="11111111";
            when others=> dig_r<="11111111";
     end case;
    end if;

else
     case count is
            when "000"=>    dig_r<="11111110";
            when "001"=>    dig_r<="11111101";
            when "010"=>    dig_r<="11111011";
            when "011"=>    dig_r<="11110111";
            when "100"=>    dig_r<="11101111";
            when "101"=>    dig_r<="11011111";
```

```
                when others=> dig_r<="11111111";
            end case;
        end if;
end if;
  end process;

  process(fre_data,q_over,dp1,dp2,dp3,count)is
     begin
   if q_over='0' then
          if dp1='1' then
                case count is
                    when "000"=>    seg_r(7)<='1';
                    when "001"=>    seg_r(7)<='1';
                    when "010"=>    seg_r(7)<='1';
                    when "011"=>    seg_r(7)<='1';
                    when "100"=>    seg_r(7)<='0';
                    when "101"=>    seg_r(7)<='1';
                    when others=> seg_r(7)<='1';
                end case;
                case fre_data is
                    when "0000"=>    seg_r(6 downto 0)<="1000000" ;
                    when "0001"=>    seg_r(6 downto 0)<="1111001" ;
                    when "0010"=>    seg_r(6 downto 0)<="0100100" ;
                    when "0011"=>    seg_r(6 downto 0)<="0110000" ;
                    when "0100"=>    seg_r(6 downto 0)<="0011001" ;
                    when "0101"=>    seg_r(6 downto 0)<="0010010" ;
                    when "0110"=>    seg_r(6 downto 0)<="0000010" ;
                    when "0111"=>    seg_r(6 downto 0)<="1111000" ;
                    when "1000"=>    seg_r(6 downto 0)<="0000000" ;
                    when "1001"=>    seg_r(6 downto 0)<="0010000" ;
                    when others=>    seg_r(6 downto 0)<="1111111" ;
                end case;


          elsif dp2='1' then
                case count is
                    when "000"=>    seg_r(7)<='1';
                    when "001"=>    seg_r(7)<='1';
                    when "010"=>    seg_r(7)<='1';
                    when "011"=>    seg_r(7)<='0';
                    when "100"=>    seg_r(7)<='1';
                    when "101"=>    seg_r(7)<='1';
                    when others=> seg_r(7)<='1';
                end case;
```

```
                case fre_data is
                    when "0000"=>    seg_r(6 downto 0)<="1000000" ;
                    when "0001"=>    seg_r(6 downto 0)<="1111001" ;
                    when "0010"=>    seg_r(6 downto 0)<="0100100" ;
                    when "0011"=>    seg_r(6 downto 0)<="0110000" ;
                    when "0100"=>    seg_r(6 downto 0)<="0011001" ;
                    when "0101"=>    seg_r(6 downto 0)<="0010010" ;
                    when "0110"=>    seg_r(6 downto 0)<="0000010" ;
                    when "0111"=>    seg_r(6 downto 0)<="1111000" ;
                    when "1000"=>    seg_r(6 downto 0)<="0000000" ;
                    when "1001"=>    seg_r(6 downto 0)<="0010000" ;
                    when others=>    seg_r(6 downto 0)<="1111111" ;
                end case;
            elsif dp3='1' then
                case count is
                    when "000"=>    seg_r(7)<='1';
                    when "001"=>    seg_r(7)<='1';
                    when "010"=>    seg_r(7)<='0';
                    when "011"=>    seg_r(7)<='1';
                    when "100"=>    seg_r(7)<='1';
                    when "101"=>    seg_r(7)<='1';
                    when others=> seg_r(7)<='1';
                end case;
                case fre_data is
                    when "0000"=>    seg_r(6 downto 0)<="1000000" ;
                    when "0001"=>    seg_r(6 downto 0)<="1111001" ;
                    when "0010"=>    seg_r(6 downto 0)<="0100100" ;
                    when "0011"=>    seg_r(6 downto 0)<="0110000" ;
                    when "0100"=>    seg_r(6 downto 0)<="0011001" ;
                    when "0101"=>    seg_r(6 downto 0)<="0010010" ;
                    when "0110"=>    seg_r(6 downto 0)<="0000010" ;
                    when "0111"=>    seg_r(6 downto 0)<="1111000" ;
                    when "1000"=>    seg_r(6 downto 0)<="0000000" ;
                    when "1001"=>    seg_r(6 downto 0)<="0010000" ;
                    when others=>    seg_r(6 downto 0)<="1111111" ;
                end case;
            else
               seg_r<=X"8e";
            end if;
        else
             seg_r<=X"00";
        end if;
      end process;
dig<=dig_r;
```

```
    seg<=seg_r;
end Behavioral;
```