# 目 录

目 录 2

实验一 简易频率计 3

一． 实验目的 3

二． 实验所用器材和设备 3

三． 实验内容 3

四． 设计方案 3

五． 代码 5

六． 问题及解决方案 11

七． 心得体会 12

八． 可改进之处 12

实验二 电子钟显示 13

一． 实验目的 13

二． 实验所用器材和设备 13

三． 实验内容 13

四． 设计方案 13

五． 代码 15

六． 问题及解决方案 23

七． 心得体会 23

八． 可改进之处 23

实验三 药片瓶装系统 24

一． 实验目的 24

二． 实验所用器材和设备 24

三． 实验内容 24

四． 设计方案 24

五． 代码 26

六． 问题及解决方案 30

七． 心得体会 30

八． 可改进之处 30

附录 ：实验每日调试日志 31

# 实验一 简易频率计

### 实验目的

⑴掌握较复杂逻辑的设计和调试。

⑵掌握用VHDL语言设计数字逻辑电路。

⑶掌握ispLEVER软件的使用方法。

⑷掌握ISP器件的使用。

⑸了解频率计的初步知识。

### 实验所用器材和设备

在系统可编程逻辑器件ISP1032 一片

示波器 一台

万用表或逻辑笔 一只

TEC-5实验系统 一台

### 实验内容

设计一个简易频率计，用于测量1MHz以下数字脉冲信号的频率。闸门只有1s一档。测量结果在数码管上显示出来。不测信号脉宽。用一片ISP芯片实现此设计，并在实验台上完成调试，建议设计采用VHDL语言编写

### 设计方案

**1. 整体思路**：求待测信号的频率，实际上是在1s内对脉冲进行计数。故先对5kHz分频产生1s的闸门信号，由此产生计数使能信号，计数清零信号（有锁存要求的也要产生锁存信号要求）；六位十进制计数器在计数使能和计数清零信号的控制下计数。若有显示频率值固定的要求，则由锁存信号控制寄存器锁存稳定的频率值。

**2. 模块划分**：采用“自顶向下，逐步细化”的设计思想，顶层文件定义总输入输出端口及各模块的输入输出端口，并进行端口匹配。底层分为三个模块：分频模块，计数模块和锁存模块，分别完成各自功能。

（1）分频模块：输入为5kHz基准时钟信号，输出为计数使能信号,计数清零信号(和锁存信号)

共有四个线程,

p1: process (clk01) 用于将5kHz时钟基准信号2500分频产生2Hz时钟信号（具体实现方式是 设置计数变量，每计到2499时2Hz时钟信号clk2产生一个脉冲）；

p2: process ( clk2 ) 用于将2Hz时钟信号clk2 2分频，产生1s的闸门信号（实现方法是 clk2每出现一个脉冲，clk1的值反转一次。这样两次分频产生的闸门信号是方波，而将5kHz直接5000分频得到的输出信号占空比不是50%）；

p3: process ( clk1 ) 用于将闸门信号clk1再次2分频从而产生计数使能信号en（和锁存信号lock）；

p4: process ( clk1,clk0\_5 ) 用于产生计数清零信号clr（计数使能信号无效的后0.5s计数器清零）。

（2）计数模块：输入为待测脉冲频率，计数使能和计数清零信号，输出为27位（前7位对应十万位7段译码管g~a，后20位对应万位~各位的8421码输出）。采用异步清零。

计数器的原理入下：

a.若个位小于9，则个位加1，其余位不变；

b.若个位为9，十位小于9，则十位加1，个位归0，其余位不变；

c.若个位，十位均为9，百位小于9，则百位加1，十位，个位归0，其余位不变；

d.若个，十，百位均为9，千位小于9，则千位加1，百位，十位，个位归0，其余位不变；

e.若个，十，百，千位均为9，万位小于9，则万位加1，千位，百位，十位，个位归0，其余位不变；

f.若个，十，百，千,万位均为9，十万位小于9，则十万位加1，万位，千位，百位，十位，个位归0，其余位不变；

g.若这6位均为9，则各位均归0（此时待测频率超出0~999999Hz量程，频率计不能正确显示）。

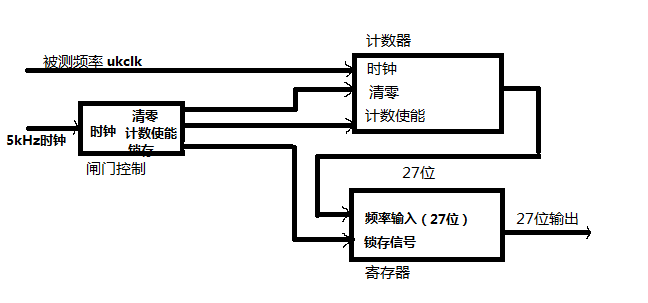
计数器的数值一定是上述情况中的一种，因此一定会进入某个if条件中执行相应语句。

\*（3）锁存模块

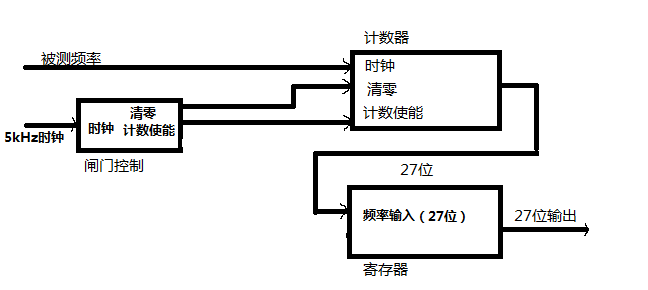
有锁存要求时， 由锁存信号控制在第2s初（上升沿）锁存计数值，因此当输入待测信号稳定时，输出总是某个恒定值。没有锁存要求时，不需锁存信号控制，锁存模块的输出就是本模块的输入。

**3. 模块框图**：

（1）有锁存信号



（2）无锁存信号



### 代码

**1.顶层文件**：

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_UNSIGNED.ALL;**

**ENTITY cymometer is**

**Port(ukclk: in std\_logic;--待测时钟输入信号**

**clk :in std\_logic;--5kHz基准时钟信号**

**q: out STD\_LOGIC\_VECTOR(26 DOWNTO 0) );--27位显示输出**

**end cymometer;**

**architecture art of cymometer is**

**COMPONENT divider**

**PORT (clk01:in std\_logic;-- 5kHz基准时钟信号输入**

**clr1: out std\_logic;--计数清零信号输出**

**en1: out std\_logic);--计数使能信号输出**

**END COMPONENT;**

**COMPONENT count10**

**PORT ( ukclk2: in std\_logic; --待测时钟输入信号**

**clr2: in std\_logic; --计数清零信号输入**

**en2: in std\_logic; --计数使能信号输入**

**qout:out STD\_LOGIC\_VECTOR(26 DOWNTO 0)); --27位输出**

**END COMPONENT;**

**COMPONENT latch6**

**PORT ( qin: in STD\_LOGIC\_VECTOR(26 DOWNTO 0);**

**fre:out STD\_LOGIC\_VECTOR(26 DOWNTO 0));**

**END COMPONENT;**

**signal tempq:STD\_LOGIC\_VECTOR(26 DOWNTO 0);**

**signal tempclr:std\_logic;**

**signal tempen:std\_logic;**

**BEGIN**

**u1 : divider PORT MAP ( clk01=>clk,clr1=>tempclr,en1=>tempen);**

**u2:count10 PORT MAP ( ukclk2=>ukclk,clr2=>tempclr,en2=>tempen,qout=>tempq );**

**u3 : latch6 PORT MAP ( qin=>tempq,fre=>q);**

**end art;**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**2.底层文件**

**（1）分频模块**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**entity divider is**

**PORT (clk01:in std\_logic;**

**clr1: out std\_logic;**

**en1: out std\_logic;**

**lock1: out std\_logic );**

**end;**

**architecture art of divider is**

**signal clk2 : std\_logic ;**

**signal clk1 : std\_logic ;**

**signal clk0\_5 : std\_logic ;**

**begin**

**p1: process (clk01)**

**variable count1 : integer range 0 to 2500;**

**begin**

**if clk01'event and clk01='1' then**

**if count1 = 2499 then**

**count1 := 0;**

**clk2 <='1';**

**else count1 :=count1+1;**

**clk2<='0';**

**End if;**

**End if;**

**end process;**

**p2: process ( clk2 )**

**variable count2 : std\_logic ;**

**begin**

**if clk2'event and clk2='1' then**

**count2 := not count2;**

**if count2 = '1'then**

**clk1 <='1';**

**else**

**clk1 <='0';**

**end if;**

**end if;**

**end process;**

**p3: process ( clk1 )**

**variable count3 : std\_logic ;**

**begin**

**if clk1'event and clk1='1' then**

**count3 := not count3;**

**if count3 = '1'then**

**clk0\_5<='1';**

**lock1<='0';**

**else**

**clk0\_5<='0';**

**lock1<='1';**

**end if;**

**end if;**

**en1<=clk0\_5;**

**end process;**

**p4: process ( clk1,clk0\_5 )**

**begin**

**if clk1='0' and clk0\_5='0' then**

**clr1 <='1';**

**else**

**clr1 <='0';**

**end if;**

**end process;**

**end art;**

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

**（2）计数模块**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**use ieee.std\_logic\_unsigned.all;**

**entity count10 is**

**PORT ( ukclk2: in std\_logic;**

**clr2: in std\_logic;**

**en2: in std\_logic;**

**qout:out STD\_LOGIC\_VECTOR(26 DOWNTO 0));**

**end count10;**

**architecture art of count10 is**

**SIGNAL temp1:STD\_LOGIC\_VECTOR(3 DOWNTO 0);**

**SIGNAL temp2:STD\_LOGIC\_VECTOR(3 DOWNTO 0);**

**SIGNAL temp3:STD\_LOGIC\_VECTOR(3 DOWNTO 0);**

**SIGNAL temp4:STD\_LOGIC\_VECTOR(3 DOWNTO 0);**

**SIGNAL temp5:STD\_LOGIC\_VECTOR(3 DOWNTO 0);**

**SIGNAL temp6:STD\_LOGIC\_VECTOR(3 DOWNTO 0);**

**SIGNAL temp:STD\_LOGIC\_VECTOR(6 DOWNTO 0);**

**begin**

**process(ukclk2,clr2,en2)**

**begin**

**if(clr2='1')then**

**temp1<="0000";**

**temp2<="0000";**

**temp3<="0000";**

**temp4<="0000";**

**temp5<="0000";**

**temp6<="0000";**

**temp<="0111111";**

**elsif(en2='1')then**

**if(ukclk2'event and ukclk2='1')then**

**if(temp1<9) then**

**temp1<=temp1+1;**

**end if;**

**if(temp2<9 and temp1="1001") then**

**temp2<=temp2+1;**

**temp1<="0000";**

**end if;**

**if(temp3<9 and temp2="1001" and temp1="1001") then**

**temp3<=temp3+1;**

**temp1<="0000";**

**temp2<="0000";**

**end if;**

**if(temp4<9 and temp3="1001" and temp2="1001" and temp1="1001") then**

**temp4<=temp4+1;**

**temp1<="0000";**

**temp2<="0000";**

**temp3<="0000";**

**end if;**

**if(temp5<9 and temp4="1001" and temp3="1001" and temp2="1001" and temp1="1001") then**

**temp5<=temp5+1;**

**temp1<="0000";**

**temp2<="0000";**

**temp3<="0000";**

**temp4<="0000";**

**end if;**

**if(temp6<9 and temp5="1001"and temp4="1001" and temp3="1001" and temp2="1001" and temp1="1001") then**

**temp6<=temp6+1;**

**temp1<="0000";**

**temp2<="0000";**

**temp3<="0000";**

**temp4<="0000";**

**temp5<="0000";**

**end if;**

**if(temp6="1001" and temp5="1001"and temp4="1001" and temp3="1001" and temp2="1001" and temp1="1001") then**

**temp6<="0000";**

**temp1<="0000";**

**temp2<="0000";**

**temp3<="0000";**

**temp4<="0000";**

**temp5<="0000";**

**end if;**

**end if;**

**end if;**

**case temp6 is**

**WHEN "0000"=>temp<="0111111";**

**WHEN "0001"=>temp<="0000110";**

**WHEN "0010"=>temp<="1011011";**

**WHEN "0011"=>temp<="1001111";**

**WHEN "0100"=>temp<="1100110";**

**WHEN "0101"=>temp<="1101101";**

**WHEN "0110"=>temp<="1111100";**

**WHEN "0111"=>temp<="0000111";**

**WHEN "1000"=>temp<="1111111";**

**WHEN "1001"=>temp<="1101111";**

**WHEN OTHERS=>temp<="0000000";**

**END case;**

**end process;**

**qout<=temp & temp5 & temp4 & temp3 & temp2 & temp1;**

**end art;**

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

**（3）锁存模块**

**带有锁存功能**

**LIBRARY IEEE;**

**USE IEEE.STD\_LOGIC\_1164.ALL;**

**ENTITY latch6 IS**

**PORT(lock3:IN STD\_LOGIC;**

**qin: IN STD\_LOGIC\_VECTOR(26 DOWNTO 0);**

**fre:OUT STD\_LOGIC\_VECTOR(26 DOWNTO 0) );**

**END latch6;**

**ARCHITECTURE art OF latch6 IS**

**BEGIN**

**PROCESS(lock3)**

**BEGIN**

**IF (lock3'event AND lock3='1' )THEN**

**fre<=qin;**

**END IF;**

**END PROCESS;**

**END art;**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**不带有锁存功能：**

**LIBRARY IEEE;**

**USE IEEE.STD\_LOGIC\_1164.ALL;**

**ENTITY latch6 IS**

**PORT( qin: IN STD\_LOGIC\_VECTOR(26 DOWNTO 0);**

**fre:OUT STD\_LOGIC\_VECTOR(26 DOWNTO 0) );**

**END latch6;**

**ARCHITECTURE art OF latch6 IS**

**BEGIN**

**fre<=qin;**

**END art;**

### 问题及解决方案

1. 顶层设计时，最初为了简便将各模块对应端口直接MAP起来，而没有设置中间变量；后来怀疑这种写法是否规范，咨询同学后决定还是在端口间设置signal，将各端口与对应signal匹配。
2. if-else 需出现在process中，不可直接出现在architecture中。
3. 分频器中三个输出信号的产生逻辑顺序：设置了4个进程，分别产生2Hz时钟信号， 1Hz闸门信号，0.5Hz时钟信号（计数使能信号）和计数器清零信号。这样虽然麻烦了一点，但是思路较为清晰。
4. 除了技术性问题外，还有一些由于粗心造成的错误，如if后缺少then, 拼写错误，忘记包含use ieee.std\_logic\_unsigned.all 头文件，带有标号的信号名称忘记打标号等等。
5. variable 和 signal的赋值语句形式不同
6. 由于试验台的6个7段译码管中，有5个是4位8421码输入，1个是7位输入，之前没有注意到这个细节，误以为都是4位输入。发现问题后修改了锁存器的代码（24位锁存器->27位锁存器），计数器代码（增添8421码 到 7段译码管的转换）及顶层文件代码（输出位数变为27位）。由此也有不小的心得与体会：再设计前一定要先仔细考察硬件电路，一丁点考虑不周导致的后果可能是程序结构，代码的大量修改。
7. 系统不能清零:计数器清零信号clr的产生依赖于闸门信号clk1和计数使能信号en，最初设计clk1二分频得到clk0\_5,再将clk0\_5赋给en，并由clk0\_5作为敏感信号触发产生clr信号的进程，系统可以清零。后来为了使代码更加简洁，将clk0\_5省去，clk1二分频直接得到en，并由en作为敏感信号触发产生clr信号的进程，但系统反而不能正确清零，后来迫不得已又改回了原版。由输出信号直接控制进程，细细想来确实不太合适。这让我认识到中间缓冲信号的作用与价值。
8. 芯片管脚的问题：6个七段译码管中，后5个均可正常显示，只有第一个的c管和f管不亮。于是换了c管和f管对应的管脚，可还是不行。用逻辑笔测试了一下，发现接入c管和f管的导线都是高电平，但插入c口和f口后变成了低电平，于是怀疑是不是LED灯坏了。借了别组的试验箱上的七段译码管，发现还是同样的问题，联想到接入c管和f管的导线插入小LED灯，灯发光但没有正常的亮，就又把故障锁定在管脚上。这次认真确认了所接管脚都是没有问题的，有烧了一次，果然程序可以正确运行了。之后测试了所有管脚并把失效管脚记录在案，以免后续实验在同样的地方栽跟头。

### 心得体会

1. 对VHDL语言有了更加深刻的认识。从前只是知道某些语句可以实现某些功能，而并真正未从一种编程语言的角度来看待它。之前的数字逻辑实验要求设计的常常是具有单一功能的元器件，而这次实验设计则比较综合，这就要求我们掌握“自顶向下，逐步细化”的设计思想，先将总功能的实现划分成多个模块，由顶层声明各模块及其管脚；具体的功能在各个模块内部实现。
2. 在编程过程中，发现VHDL硬件语言与以前使用的C语言,C++语言有很多相似（如语句方面：if-else；case语句；设计思想方面：自顶向下，逐步细化，功能划分成多个子模块来实现），也有许多不同（如VHDL中同一进程中的if语句是并行进行的，而C语言和C++则是顺序进行的，由此C中的一些设计技巧在VHDL中无法进行）。
3. 建立多个线程可使子模块继续“分块”，程序思路也更加清晰。但有时在多个线程中对同一个变量赋值是不允许的，这就给线程的划分带来了困难。一些逻辑上不可分的功能只能在同一个进程中实现。
4. 模块合理划分的重要性：编程后期深刻的体会到，在编码前一定要先做好分析设计，将功能合理的划分成一个个的子模块，考虑清楚每个模块的输入，输出端口。模块分得好，事半功倍，后面的编码也会轻松许多；模块分得混乱，编码时很难思路清晰，有时也会造成代码冗余和重复。

### 可改进之处

频率计可以设计成多倍率，以适应不同的频率范围使用。由于时间原因没有实践，思路如下：增加倍率选择输入端（可设计为两位，实现四种倍率选择）通过倍率选择输入端控制分频。如倍率为1时，5kHz的基准信号分频得1s的闸门信号；7段译码管显示的即为输入待测频率；而倍率为10 时，将5kHz的基准信号分频得0.1s的闸门信号，7段译码管显示的数值乘以10即为待测频率。

# 实验二 电子钟显示

### 实验目的

（1）掌握复杂的逻辑设计和调试

（2）学习用原理图+VHDL语言设计逻辑电路。

（3）学习数字电路模块层次设计。

（4）掌握ispLEVER软件的使用方法。

（5）熟悉ISP器件的使用

### 二．实验所用器材和设备

|  |  |
| --- | --- |
| 在系统可编程逻辑器件ISP1032E | 一片 |
| 逻辑笔 | 一只 |
| TEC-5实验系统，或TDS-2B数字电路实验系统 | 一台 |
| 连接线路导线 | 若干 |

### 实验内容

（1）设计并用ISP1032实现一个电子钟。电子钟具有下述功能：

a) 试验台上的6个数码管显示时、分、秒。

b) 能使电子钟复位（清零）。

c) 能随时启动或者停止电子钟运行。

d) 修改时、分、秒的值。

e) 具有报时功能，整点时喇叭鸣叫。

（2）要求整个设计分为若干模块。顶层模块用原理图设计，底层模块用VHDL语言设计。

（3）在试验台上调试设计。

### 设计方案

1.整体思路：电子钟的程序分为四个模块，由于时钟以秒为单位变化，因此首要任务是利用5kHz的脉冲产生1Hz的时钟脉冲，然后需要为脉冲计数。我认为计数是我们这个程序最大的闪光点，简洁的计数方式让整个程序思路更清晰明了，详细方法在模块分析中具体说明。对于校时部分，我的构想是时，分，秒的两位共六位能够单独调节，并且限定在合理的范围以内，例如”时”的第一位数只能是0、1、2，六个位数用三位二进制数选择，用单次脉冲进行加法以调节。由于时钟要从调节好的时间开始继续往前走，因此要将调好的时间送给计数器，继续计时。对于整点报时部分，在最终输出时进行控制，由于实验台的限制，也需要将第一个数字的输出转化为七段译码。

2.模块划分：

顶层文件定义总输入输出端口及各模块的输入输出端口，端口匹配。底层分为三个模块：分频模块divider，计数模块counter，时间重置模块set和输出显示模块screen。counter只有计数功能，stop、clr做为使能。Set是校时时用来设定时间的,Set将设定好的时间传输给counter，以便设好时间后从预期的时间开始走。Screen将六个8421码中的第一个数值变成7段译码，并且控制整点闹铃。

1. 分频模块：输入5000hz的脉冲，先将其进行2500分频，变成了一秒两次的脉冲，然后利用内部变量对它进行二分频，既遇到上升沿时反转，这样就产生了每秒一个上升沿的方波，将其输出做为计时的标准。
2. 时间重置模块：

set是校时的时候用来设置时间的模块，有choice（2：0）做为输入，选择调节六个数中的某一位，000代表时的第一位，001代表时的第二位，以此类推。由于设置好的时间必须是合理的范围，因此第一位数必须在零到二之间，当它为二时，再给add脉冲是它变为0，否则就加一，第二位数需要看第一位的情况，如果第一位数是零或一，第二位可以是零到九，如果第一位是二，那么第二位只能是0到3之间的数，同理，第三位只能是0到5，以此类推。设置好的数用一个24位的变量qout1输送给counter，counter中的settime使能决定是否接收该数。

1. 计数模块

counter子程序有settime，clr两个使能信号，有1hz脉冲和set中传输的时间做为输入，并且还有六个8421码的数字做为中间变量，分别代表时分秒六位数。当clr为1时，六个中间变量均变为零，当settime变为1，中间变量变为set传过来的数，每次计时从中间变量的当前值开始，并且将当前值输出。计数时未使用模块间的相互进位，全部在一个进程中完成,计数原理如下。经过尝试发现将clk做为计数的控制条件，放在最外层的时候时钟脉冲有效。

采用同步的方式进行时钟清零与时间重置。

计数原理：以h1h2m1m2s1s2表示小时、分钟、秒的首位和末位。从s2开始从后向前依次判断各位,

s2<9时，只需s2加1；

s2=9且s1<5时，s1加1，s2回0；

s1=9且 s2=5且 m2<9时，m2加1，s1，s2回0；

s1=9且 s2=5且 m2=9且m1<5时，m1加1，m2、s1、s2回0；

s1=9且 s2=5且 m2=9且m1=5且h1<2且h2<9时，h2加1，m1、m2、s1、

回0；

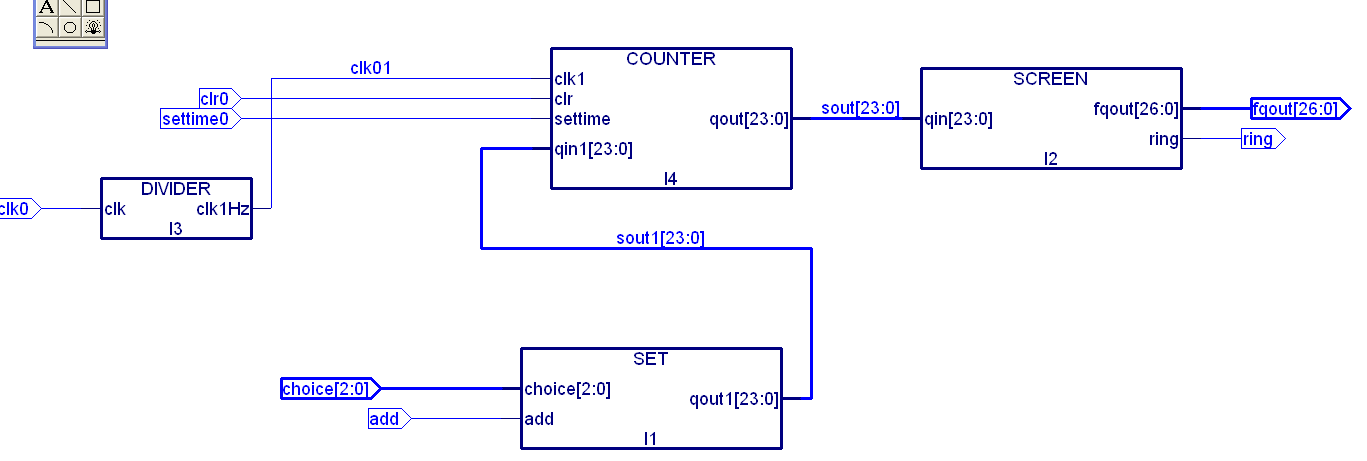
s1=9且 s2=5且 m2=9且m1=5且h1<2且h2=9时，h1加1，h2、m1、m2、s1、s2回0；

s1=9且 s2=5且 m2=9且m1=5且h1=2且h2<3时，h2加1， m1、m2、s1、s2回0；

s1=9且 s2=5且 m2=9且m1=5且h1=2且h2=3时，h1、h2、m1、m2、s1、s2回0。

1. 输出显示模块：screen子程序的功能有两个，一是把原本六个8421码中的第一个8421码改为七段译码，方法是将传过来的qin参数的0-19位保留，然后将20-23位变成七段译码，最后合并成一个27位的二进制数输出；第二个功能是整点报时，判断qin的最后16位，当最后16位的值在1和5之间时令代表ring的中间变量变为1，然后将中间变量的值赋给ring，如果不设置中间变量的话闹铃会一直响。

3.原理图



### 代码

**1.顶层文件**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**use ieee.std\_logic\_unsigned.all;**

**entity clock is**

**port(clk0:in std\_logic;--时钟输入信号**

**clr:in std\_logic;--电子钟清零使能信号**

**settime:in std\_logic;-- 电子钟重置使能信号**

**choice:in std\_logic\_vector(2 downto 0);-- 电子钟重置位选择**

**add:in std\_logic;-- 单次脉冲（控制重置的时间设置）**

**fqout:out std\_logic\_vector(26 downto 0); --显示输出**

**ring:out std\_logic);--整点报时：响铃输出**

**end;**

**architecture art of clock is**

**component divider**

**port (clk:in std\_logic;**

**clk1Hz: out std\_logic );**

**end component;**

**component counter**

**port(clk1:in std\_logic;**

**clr:in std\_logic;**

**settime:in std\_logic;**

**qin1:in std\_logic\_vector(23 downto 0);**

**qout:out std\_logic\_vector(23 downto 0));**

**end component;**

**component set**

**port(choice:in std\_logic\_vector(2 downto 0);**

**add:in std\_logic;**

**qout1:out std\_logic\_vector(23 downto 0));**

**end component;**

**component screen**

**port(qin:in std\_logic\_vector(23 downto 0);**

**fqout:out std\_logic\_vector(26 downto 0);**

**ring:out std\_logic);**

**end component;**

**signal clk01:std\_logic;**

**signal sout1:std\_logic\_vector(23 downto 0);**

**signal sout:std\_logic\_vector(23 downto 0);**

**begin**

**u1:divider port map(clk=>clk0,clk1Hz=>clk01);**

**u2:counter port map(settime=>settime,qout=>sout,clr=>clr,clk1=>clk01,qin1=>sout1);**

**u3:set port map(choice=>choice,add=>add,qout1=>sout1);**

**u4:screen port map(qin=>sout,fqout=>fqout,ring=>ring);**

**end art;**

**2.底层文件**

**（1）分频模块**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**entity divider is**

**port (clk:in std\_logic;**

**clk1Hz: out std\_logic );**

**end;**

**architecture art of divider is**

**signal clk2Hz : std\_logic ;**

**begin**

**p1: process (clk)**

**variable count1 : integer range 0 to 2500;**

**begin**

**if clk'event and clk='1' then**

**if count1 = 2499 then**

**count1 := 0;**

**clk2Hz <='1';**

**else count1 :=count1+1;**

**clk2Hz<='0';**

**End if;**

**End if;**

**end process;**

**p2: process ( clk2Hz )**

**variable count2 : std\_logic ;**

**begin**

**if clk2Hz'event and clk2Hz='1' then**

**count2 := not count2;**

**if count2 = '1'then**

**clk1Hz <='1';**

**else**

**clk1Hz <='0';**

**end if;**

**end if;**

**end process;**

**end art;**

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

（2）时间设置模块

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**use ieee.std\_logic\_unsigned.all;**

**entity set is**

**port(choice:in std\_logic\_vector(2 downto 0);**

**add:in std\_logic;**

**qout1:out std\_logic\_vector(23 downto 0));**

**end;**

**architecture art of set is**

**signal temph12:std\_logic\_vector(3 downto 0);**

**signal temph22:std\_logic\_vector(3 downto 0);**

**signal tempm12:std\_logic\_vector(3 downto 0);**

**signal tempm22:std\_logic\_vector(3 downto 0);**

**signal temps12:std\_logic\_vector(3 downto 0);**

**signal temps22:std\_logic\_vector(3 downto 0);**

**signal tempqout:std\_logic\_vector(23 downto 0);**

**begin**

**process(choice,add)**

**begin**

**if(add'event and add='1')then**

**if(choice="000")then**

**if(temph12="0010")then**

**temph12<="0000";**

**else**

**temph12<=temph12+"0001";**

**end if;**

**end if;**

**if(choice="001") then**

**if((temph12<"0010"and temph22="1001")or(temph12="0010" and temph22="0011"))then**

**temph22<="0000";**

**else**

**temph22<=temph22+1;**

**end if;**

**end if;**

**if(choice="010" )then**

**if(tempm12="0101")then**

**tempm12<="0000";**

**else**

**tempm12<=tempm12+1;**

**end if;**

**end if;**

**if(choice="011")then**

**if(tempm22="1001")then**

**tempm22<="0000";**

**else**

**tempm22<=tempm22+1;**

**end if;**

**end if;**

**if(choice="100")then**

**if(temps12="0101")then**

**temps12<="0000";**

**else**

**temps12<=temps12+'1';**

**end if;**

**end if;**

**if(choice="101")then**

**if(temps22="1001")then**

**temps22<="0000";**

**else**

**temps22<=temps22+1;**

**end if;**

**end if;**

**end if;**

**tempqout<=temph12&temph22&tempm12&tempm22&temps12&temps22;**

**qout1<=tempqout;**

**end process;**

**end art;**

（3）计数模块：

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**use ieee.std\_logic\_unsigned.all;**

**entity counter is**

**port( clk1:in std\_logic;**

**clr:in std\_logic;**

**settime:in std\_logic;**

**qin1:in std\_logic\_vector(23 downto 0);**

**qout:out std\_logic\_vector(23 downto 0));**

**end;**

**architecture art of counter is**

**signal temph1:std\_logic\_vector(3 downto 0);**

**signal temph2:std\_logic\_vector(3 downto 0);**

**signal tempm1:std\_logic\_vector(3 downto 0);**

**signal tempm2:std\_logic\_vector(3 downto 0);**

**signal temps1:std\_logic\_vector(3 downto 0);**

**signal temps2:std\_logic\_vector(3 downto 0);**

**signal flag:std\_logic;**

**begin**

**p1:process(settime,clr,clk1,qin1)**

**begin**

**if( clk1'event and clk1='1')then**

**if(clr='1')then**

**temph1<="0000";**

**temph2<="0000";**

**tempm1<="0000";**

**tempm2<="0000";**

**temps1<="0000";**

**temps2<="0000";**

**elsif(settime='1' )then**

**temph1<=qin1(23 downto 20);**

**temph2<=qin1(19 downto 16);**

**tempm1<=qin1(15 downto 12);**

**tempm2<=qin1(11 downto 8);**

**temps1<=qin1(7 downto 4);**

**temps2<=qin1(3 downto 0);**

**else**

**if(temps2<"1001")then**

**temps2<=temps2+1;**

**end if;**

**if(temps1<"0101" and temps2="1001" )then**

**temps1<=temps1+1;**

**temps2<="0000";**

**end if;**

**if(tempm2<"1001" and temps1="0101" and temps2="1001" )then**

**tempm2<=tempm2+1;**

**temps1<="0000";**

**temps2<="0000";**

**end if;**

**if(tempm1<"0101" and tempm2="1001" and temps1="0101" and temps2="1001" )then**

**tempm1<=tempm1+1;**

**tempm2<="0000";**

**temps1<="0000";**

**temps2<="0000";**

**end if;**

**if(temph1<"0010" and temph2<"1001" and tempm1="0101" and tempm2="1001" and temps1="0101" and temps2="1001")then**

**temph2<=temph2+1;**

**tempm1<="0000";**

**tempm2<="0000";**

**temps1<="0000";**

**temps2<="0000";**

**end if;**

**if(temph1<"0010" and temph2="1001" and tempm1="0101" and tempm2="1001" and temps1="0101" and temps2="1001" ) then**

**temph1<=temph1+1;**

**temph2<="0000";**

**tempm1<="0000";**

**tempm2<="0000";**

**temps1<="0000";**

**temps2<="0000";**

**end if;**

**if(temph1="0010" and temph2<"0011" and tempm1="0101" and tempm2="1001" and temps1="0101" and temps2="1001" )then**

**temph2<=temph2+1;**

**tempm1<="0000";**

**tempm2<="0000";**

**temps1<="0000";**

**temps2<="0000";**

**end if;**

**if(temph1="0010" and temph2="0011" and tempm1="0101" and tempm2="1001" and temps1="0101" and temps2="1001" )then**

**temph1<="0000";**

**temph2<="0000";**

**tempm1<="0000";**

**tempm2<="0000";**

**temps1<="0000";**

**temps2<="0000";**

**end if;**

**end if;**

**end if;--clr**

**qout<=temph1&temph2&tempm1&tempm2&temps1&temps2;**

**end process;**

**end art;**

（4）输出显示模块

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**use ieee.std\_logic\_unsigned.all;**

**entity screen is**

**port (qin:in std\_logic\_vector(23 downto 0);**

**fqout:out std\_logic\_vector(26 downto 0);**

**ring:out std\_logic);**

**end;**

**architecture art of screen is**

**signal temp\_ring:std\_logic;**

**begin**

**process(qin)**

**begin**

**if(qin(15 downto 0)>="0000000000000000" and qin(15 downto 0)<="0000000000000101")then**

**temp\_ring<='1';**

**else**

**temp\_ring<='0';**

**end if;**

**case qin(23 downto 20) is**

**WHEN "0000"=>fqout(26 downto 20)<="0111111";**

**WHEN "0001"=>fqout(26 downto 20)<="0000110";**

**WHEN "0010"=>fqout(26 downto 20)<="1011011";**

**WHEN "0011"=>fqout(26 downto 20)<="1001111";**

**WHEN "0100"=>fqout(26 downto 20)<="1100110";**

**WHEN "0101"=>fqout(26 downto 20)<="1101101";**

**WHEN "0110"=>fqout(26 downto 20)<="1111100";**

**WHEN "0111"=>fqout(26 downto 20)<="0000111";**

**WHEN "1000"=>fqout(26 downto 20)<="1111111";**

**WHEN "1001"=>fqout(26 downto 20)<="1101111";**

**WHEN OTHERS=>fqout(26 downto 20)<="0000000";**

**end case;**

**fqout(19 downto 0)<=qin(19 downto 0);**

**ring<=temp\_ring;**

**end process;**

**end art;**

### 问题及解决方案

1. 首先是GLB的问题，程序使用GLB是46%，后来将counter中if嵌套的条件附加，减少了一层循环，GLB减少到33%，还是不能通过编译。接着想到把counter中process的内容拆分成两部分，但是尝试过之后还是不成立，因为两个进程不能给同一个量赋值。最后将6个8421码的输出改成1个24位数的输出，还有speaker中的循环也减少了一层，GLB才通过了编译。
2. 编译通过后插线试运行，发现时钟不走动，想了几种办法，最后将清零和暂停改为同步清零和同步暂停，将clk上升沿做为if循环的条件放到最外层，问题才得到解决。
3. 整点报时信号ring一直高电平，闹铃不停地响，然后为ring输出设了signal，才使闹铃正常工作。
4. 开始时在counter中没有设置signal中间变量，直接改变输出的值，导致h1<=h1+1这类语句全部出错，于是设置了中间变量，在process的最后再给外部变量赋值。

### 心得体会

这个小学期中我学会了将VHDL语言熟练运用，能将所学的知识运用于实际，虽然编写程序、调试程序的过程非常艰辛苦闷，但是成功之后让我非常有成就感，从中领略到了硬件软件学习的乐趣。VHDL语言与之前学的c语言、c++相比，多的是与硬件的结合，这一点让我觉得十分有趣，而相对缺少的是语言的灵活性、功能，句式的种类和以前学的语言相比没有那么多，使得在编写程序的时候采用的表达方式有限。另外我感触比较深的就是，硬件和软件不同在于硬件本身有局限性，因此虽然有时我们觉得思路没有问题，但却不能保证硬件能够实现我们预期的功能，调试程序的过程比编写的过程更具有挑战性。还有一个收获就是要不停地尝试，不能放弃，不能理所当然地觉得“这样改肯定不对，不试了”，在成功之前，应该将能想到的方法都付诸实践，这样才有成功的可能。

### 可改进之处

这次实验总体来说比较满意，让我有很多的收获，但遗憾的是电子钟的程序里没有能实现定时闹钟的功能，有机会希望能使用更好的机器来实现开始的构想。

# 实验三 药片瓶装系统

### 实验目的

掌握较复杂的逻辑设计和调试。

学习用原理图+VHDL语言设计逻辑电路

学习数字电路模块的层次设计。

掌握ispLEVER软件的使用方法。

熟悉ISP期间的使用。

### 实验所用器材和设备

在系统可编程逻辑期间ISP1032 一片

示波器 一台

万用表火逻辑笔 一只

TEC-5实验系统，或TDS-2B数字电路实验系统 一台

### 实验内容

药片由输送管送入漏斗装置中，后者颈部每次只允许一粒药片掉进传送带上的瓶子里。漏斗的颈部有一个光传感器，它**探测到每一粒药片后产生一个电脉冲信号**。这个脉冲传送到计数器中，使其计数加1，这样在药片装入瓶子过程的任一时刻，计数器都保存着瓶子中药片数量的二进制数。这个二进制数以计数器通过并行导线传送到比较器的输入端B。

另一方面，每个瓶子中要装入的固定药片数量（例如50片）通过键盘手动设置。按键信号经过编码器编码后送到寄存器A保存，而代码转换器A将寄存器A中的BCD数变成二进制数送到比较器输入端A。

假设每个瓶子要装50粒药片，当计数器的数值达到50后，比较器的A=B输出端出现高电平，指示瓶子已装满，立即关闭漏斗颈上的阀门使药片停止下落，与此同时它使传送带移动下一个瓶子到漏斗的下面。当瓶子到达漏斗颈正下方时，传送带的控制电路产生一个脉冲信号使计数器清0，比较器A=B输出端变成低电平，打开漏斗阀门，重新开始药片滴落。

结合上面的药片装瓶系统设计实例，采用VHDL设计，并用ISP1032E大容量器件实现如图所示的药片装瓶系统。

⑴实验台上的5个数码管作为显示系统，显示每瓶药片及总药片的数量。

⑵用实验台的红绿发光二极管来模拟对机电装置系统的输出，绿色灯亮表示启动机电装置，装瓶进行中；红色灯亮表示装瓶完成，机电装置关闭。

⑶**输入子系统为包括BCD码每瓶装药数输入与装瓶开始脉冲输入，设计要求每瓶最大药片数50粒，最多装18瓶**。

⑷启动装瓶开始脉冲后，如果输入数量超出最大装瓶数或者为零，要求显示系统出现告警提示。

⑸漏斗感应器送来的药片装瓶信号用2s信号模拟，可以用实验台提供的5kHz的时钟分频产生。

⑹在实验台上调试设计。

### 设计方案

**（1）整体思路：**

1.整体分为两个模块，一个用于分频，另一个用于实现药片装瓶系统的所有功能。

2. 由于输出显示管采用8421码，所以输入及中间信号也采用8421码。

3. 中间信号的计数并没有采用计数器，而是用逻辑关系，实现8421码的相加。例如，一个共12位的信号，则可以表示成一个3位的十进制数。那我先判断这三位是不是“999”，若是，则加上“999”对应的8421码的补码；若不是，则再判断除去最高位的后两位是不是“99”，若是，则加上“99”对应的8421码的补码；若不是，则再判断，除去最高两位的数是不是“9”，若是，则加上“9”对应的8421码的补码；若还不是，那就说明这个二进制码可以进行简单的加一，而不涉及到进位的问题。通过此方法，实现8421码的加一。

**（2）模块细分：**

1. divider模块：

该模块用于将5K Hz的脉冲分成0.5Hz的脉冲。具体做法是，先将5K Hz的脉冲分成1Hz的脉冲，然后再将其分成0.5Hz的方波。

2. pill模块：

该模块用于实现题目中的所有要求，包括以下功能：

1）药片装瓶计数

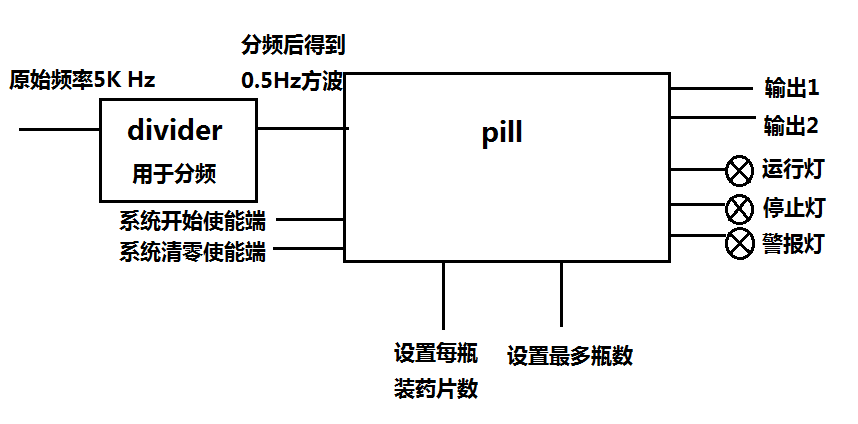
2）显示已装完的药片及瓶数

3）系统的灯的输出

4) 操作人员可自行设定每瓶装药片数及最大装瓶数

5）超出规定时系统停止并报警

**（3）结构框图：**



### 代码

1. **顶层文件**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**use ieee.std\_logic\_unsigned.all;**

**entity total is**

**port(clk5k:in std\_logic;--5kHz基准时钟信号**

**sta:in std\_logic;--外部启动使能**

**rs:in std\_logic;--清零信号**

**pillnum:in std\_logic\_vector(7 downto 0);--待输入的每瓶药片数（不大于50）**

**maxbotnum:in std\_logic\_vector(7 downto 0);--待输入的装瓶数（不大于18）**

**print1:out std\_logic\_vector(7 downto 0);--输出显示：装瓶时显示每瓶药片数；**

**暂停时显示已装瓶数**

**print2:out std\_logic\_vector(11 downto 0);--输出显示：已装总药片数**

**golight:out std\_logic;--绿灯，正在装瓶时亮**

**stoplight:out std\_logic;--红灯，暂停和装瓶结束时亮**

**warnlight:out std\_logic);--黄灯，输入不符合要求时亮**

**end total;**

**architecture art of total is**

**component divider**

**port (clk0:in std\_logic; --5kHz基准时钟信号**

**clk0\_5: out std\_logic );--分频得到的0.5Hz时钟信号**

**end component;**

**component pill**

**port(clk:in std\_logic; --分频得到的0.5Hz时钟信号**

**start:in std\_logic; --外部启动使能**

**rst:in std\_logic; --清零信号**

**pill\_num:in std\_logic\_vector(7 downto 0); --待输入的每瓶药片数（不大于50）**

**max\_bot\_num:in std\_logic\_vector(7 downto 0); --待输入的装瓶数（不大于18）**

**dout1:out std\_logic\_vector(7 downto 0);- -输出显示：装瓶时显示每瓶药片数；**

**暂停时显示已装瓶数**

**dout2:out std\_logic\_vector(11 downto 0); --输出显示：已装总药片数**

**go\_light:out std\_logic; --绿灯**

**stop\_light:out std\_logic; --红灯，**

**warn\_light:out std\_logic); --黄灯**

**end component;**

**signal temp\_clock:std\_logic;**

**begin**

**u1 : divider PORT MAP (clk0=>clk5k, clk0\_5=>temp\_clock);**

**u2:pill PORT MAP(clk=>temp\_clock, start=>sta, rst=>rs, pill\_num=>pillnum, max\_bot\_num=>maxbotnum, dout1=>print1,dout2=>print2, go\_light=>golight, stop\_light=>stoplight,warn\_light=>warnlight);**

**end art;**

1. **底层文件**
2. **分频模块**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**entity divider is**

**PORT (clk0:in std\_logic;**

**clk0\_5: out std\_logic );**

**end;**

**architecture art of divider is**

**signal clk1 : std\_logic ;**

**begin**

**p1: process (clk0)**

**variable count1 : integer range 0 to 5000;**

**begin**

**if (clk0'event and clk0='1') then**

**if (count1 = 4999) then**

**count1 := 0; --4999就回到0再自增**

**clk1 <='1';**

**else count1 :=count1+1; --每5000次，clk1变成0一次**

**clk1<='0';**

**end if;**

**end if;**

**end process p1;**

**p2: process ( clk1 )**

**variable count2 : std\_logic ;**

**begin**

**if (clk1'event and clk1='1') then**

**count2 := not count2; --clk1从0变成1时，count2反转**

**if count2 = '1'then**

**clk0\_5 <='1';**

**else**

**clk0\_5 <='0'; --方波，相当于每5000\*2次周期对应一次输出**

**end if;**

**end if;**

**end process p2;**

**end art;**

1. **药品装瓶控制模块**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**use ieee.std\_logic\_unsigned.all;**

**entity pill is**

**port(clk:in std\_logic;**

**start:in std\_logic;**

**rst:in std\_logic;**

**pill\_num:in std\_logic\_vector(7 downto 0);--每瓶药片数**

**max\_bot\_num:in std\_logic\_vector(7 downto 0);--最大瓶数**

**dout1:out std\_logic\_vector(7 downto 0);**

**dout2:out std\_logic\_vector(11 downto 0);**

**go\_light:out std\_logic;**

**stop\_light:out std\_logic;**

**warn\_light:out std\_logic);**

**end;**

**architecture arc of pill is**

**signal count:std\_logic\_vector(7 downto 0);**

**signal bot\_c:std\_logic\_vector(7 downto 0);**

**signal pill\_c:std\_logic\_vector(11 downto 0);**

**signal stop:std\_logic;--一些中间变量，比如这个stop就是指示暂停模式的**

**begin**

**process(clk,start,rst)**

**begin**

**if(rst='1')then**

**count<="00000000";--rst进行重设，count表示当前瓶的药片数**

**bot\_c<="00000000";--当前装好的瓶数**

**pill\_c<="000000000000";--当前用的总药片数**

**stop<='0';**

**warn\_light<='0';**

**go\_light<=start;**

**stop\_light<='0';**

**else**

**if(start='1')then**

**if( (pill\_num="000000000000") or (pill\_num>"01010000") or (max\_bot\_num>"00011000") or (max\_bot\_num="00000000"))then**

**warn\_light<='1';--出现了输入违法的情况：总药片存完、本瓶药片过多、最大瓶数过多、最大瓶数为0**

**stop<='1';**

**elsif(bot\_c=max\_bot\_num)then**

**stop<='1';--达到最大值，结束**

**warn\_light<='0';**

**end if;**

**if((pill\_num>"000000000000") and (pill\_num<"01010000") and (max\_bot\_num<"00011000") and (max\_bot\_num>"00000000") and stop='0')then**

**go\_light<='1';--正常运行的时候**

**stop\_light<='0';**

**if(clk'event and clk='1')then**

**if(count=pill\_num)then**

**count<="00000000"; --clk输入，模拟一个药片落入了，而且该换瓶了**

**if(bot\_c="10011001")then**

**bot\_c<=bot\_c+"01100111";--99加了以后就清零了**

**elsif(bot\_c(3 downto 0)="1001")then**

**bot\_c<=bot\_c+"0111";--进位**

**else bot\_c<=bot\_c+1;**

**end if;**

**else**

**if(count="10011001")then**

**count<=count+"01100111";--同样的清零**

**elsif(count(3 downto 0)="1001") then**

**count<=count+"0111";**

**else count<=count+1;**

**end if;**

**if(pill\_c="100110011001")then**

**pill\_c<=pill\_c+"011001100111";--999清零**

**elsif(pill\_c(7 downto 0)="10011001")then**

**pill\_c<=pill\_c+"01100111";--进位**

**elsif(pill\_c(3 downto 0)="1001")then**

**pill\_c<=pill\_c+"0111";--进位**

**else pill\_c<=pill\_c+1;**

**end if;**

**end if;--end count=pill\_num，count等于进入的片数**

**end if;--end clk**

**else --system pause**

**go\_light<='0';--不是正常状态，就是停止了**

**stop\_light<='1';**

**end if;--end stop**

**else --compared with start**

**go\_light<='0';--start=0，运行使能为0就停止**

**stop\_light<='1';**

**end if; --end start**

**end if;--end rst**

**if(start='1') then**

**dout1<=count; --输出模块，如果还在运行，就输出本瓶的片数**

**else**

**dout1<=bot\_c;--不然就输出装好的瓶数**

**end if;**

**dout2<=pill\_c;--总片数用dou2输出**

**end process;**

**end arc;**

### 问题及解决方案

1. 最开始编译时，系统总在一个定义信号的语句处报错，这令我百思不得其解。试了很多办法都不行，最后把照片拍下来传到了论坛，得到了大神的解答。原来同一个signal信号不可以在两个进程中赋值。我将stop信号的赋值语句改在同一个进程中后，总算通过了编译。

2. 在编译过程中，我的一个while语句一直报错，可是我又查不出问题。后来咨询老师，老师说可能是系统的缺陷。于是我把while改成了if，最终编译通过。

3. 编译好后，在实验台上运行，发现示数无法改变。我们仔细检查程序，也没有发现逻辑上的问题。于是我们猜测是有些管脚不好用，导致系统无法正常运行。所以我们用逻辑笔一个一个地测试管脚，果然很多管脚坏掉了。我们把坏的管脚标志出来。当确保所有管脚都好用后，示数终于随着时钟变化了。

4. 起初时清零一直有问题，困扰了我好久。当时pill模块中有3个进程。由于受到第一个问题的启发，我想会不会又是进程出了问题。于是我将三个进程合并成一个进程，果然清零功能正常了。

### 心得体会

为期两周的数字逻辑课程设计已经接近尾声了，在这两周里我收获颇丰。

我负责的是药片装瓶系统。刚拿到这个课题的时候，我简直头都大了，因为以前学的东西都忘得差不多了。后来通过自学，总算把程序写出来了，但编译的时候就是会出现各种各样的报错，有的甚至会让我觉得莫名其妙。编译过了之后，测试过程中还会出现让人意想不到的问题。有时候我甚至都觉得要绝望了，不知道问题出在哪里。代码应该是对的，但是运行的结果就是不对，一点思绪都没有，真的想要放弃了。但是过一会儿我又还是会接着做。因为这不仅是为了完成小学期的任务，更是一种对目标的坚持，我坚信努力不会白费。所以当我看到自己编写的程序写入芯片，然后实验台的数码管上出现了我所期待的结果，那种喜悦简直是无法言喻的。

再讲一讲我在硬件，VHDL语言和数字逻辑上面的收获吧。首先，我了解了分频和计数的原理，这个是我这个程序的一个基础。此外，我第一次知道了，在VHDL中if从句是并行执行的，这与我们在C语言中培养出的逻辑习惯完全不同。通过本次实验，我还学会了原理图的制作方法。总之收获很大。

另外，通过这次的编程，我也更加重视培养严谨的编程态度。有时，只是一个小的编程疏忽就可能要调试很长时间，而这浪费的不仅仅是自己的时间，也是全组成员的时间。这也让我们更加认识到团队合作的重要性，培养了我们严谨认真、耐心细致的实验态度，提高了我们发现问题、查找原因、解决问题的能力。

### 可改进之处

原本我们的程序结果是，无论系统运行与否，两个输出分别是每瓶装药片数和已装好的总药片数。在老师的帮助下，我们将程序作了改进：若系统运行，则两个输出分别为当前瓶已装的药片数和已装好的总药片数；若系统停止，则两个输出分别为已装好的总瓶数和已装好的总药片数。

**附录 ：实验每日调试日志**

**~~~~日期 ：8.27~~~~**

**进程**

上午:根据已有源代码，完成电子琴系统，不仅复习了VHDL的基本语法语句及isplever软件的使用方法，也理解了分频的原理，对自顶向下，逐步细化的设计思想有了初步而深刻的认识。

下午：参照电子琴系统的例子，开始简易频率计的设计，并完成顶层设计及分频，锁存的代码编写。

晚上：编译已编写代码使通过。

**设计思路**

频率计功能的实现主要依靠三部分：分频器（产生闸门信号），计数器（六位十进制）和锁存器（防止显示的频率值不断闪烁）。因此底层设计中将包含这三个子程序，顶层中包含这三个COMPONENT即可。

1分频器的设计思路：分频器最终要产生3个信号输出：计数使能信号，锁存器锁存信号和计数器清零信号。为了产生这三个信号，可先产生1Hz的闸门信号clk1——将5kHz的基准时钟信号先2.5k分频，再2分频(这样最终得到1Hz的闸门信号是方波，而直接5k分频得到的不是方波)。

1.1计数使能信号en的产生：由于题目要求1s计数1s显示，使能信号应是0.5Hz的方波，其产生可以是每到闸门信号clk1上升沿，计数使能信号便反转一次。

1.2锁存器锁存信号lock的产生：从原理上讲，前一秒计数使能信号有效，后一秒上升沿锁存信号有效；从波形上讲，lock信号与en的波形是相反的，因此只需?±lock<=not en?±即可得到lock.

1.3计数器清零信号clr的产生：默认在第2秒的后0.5s计数器清零，即计数使能无效且闸门信号为低电平时，clr有效。

2.计数器的设计思路：用多层if-else 嵌套的方式较易于设计，但这样有可能使系统的GLB过高，暂时没有较好的解决方法。

3.锁存器的设计思路：当锁存器锁存信号lock上升沿时，将输入锁存，赋给输出。

**遇到的问题及解决**

1.顶层设计时，最初为了简便将各模块对应端口直接MAP起来，而没有设置中间变量；后来怀疑这种写法是否规范，咨询同学后决定还是在端口间设置signal，将各端口与对应signal匹配。

2.if-else 需出现在process中，不可直接出现在architecture中。

3.分频器中三个输出信号的产生逻辑顺序：设置了4个进程，分别产生2Hz时钟信号， 1Hz闸门信号，0.5Hz时钟信号（计数使能信号）和计数器清零信号。这样虽然麻烦了一点，但是思路较为清晰。

4.除了技术性问题外，还有一些由于粗心造成的错误，如if后缺少then, 拼写错误，带有标号的信号名称忘记打标号等等。

5.variable 和 signal的赋值语句形式不同

**小组讨论**

针对计数器的设计进行讨论，如何能够尽量避免多层if-else的嵌套。尽管一些同学使用多层if-else嵌套的方式设计运行起来并没有什么问题，但我们认为程序还是尽量规范为好，潜在威胁还是越少越好。

**~~~~日期8.28~~~~**

**进程**

上午：完成计数器部分的编码，修改锁存器编码，编译通过后将程序烧入芯片，开始调试。

下午：继续调试工作

晚上：调试成功

**设计思路**

经过不断的思考，想到了一个可以避免多层if-else嵌套的方法：用6个signal存储六位十进制计数器的每一位，然后从个位开始逐渐增加判断位数：

若个位小于9，则个位加1，其余位不变；

若末尾为9，十位小于9，则十位加1，末位归0，其余位不变；

若末尾为9，十位也为9，百位小于9，则百位加1，十位，个位归0，其余位不变；

……

若这6位均为9，则各位均归0（此时待测频率超出0~999999Hz量程，频率计不能正确显示）。

计数器的数值一定是上述情况中的一种，因此一定会进入某个if条件中，于是避免了多层if-else的嵌套。

**遇到的问题及解决**

1.由于试验台的6个7段译码管中，有5个是4位8421码输入，1个是7位输入，之前没有注意到这个细节，误以为都是4位输入。发现问题后修改了锁存器的代码（24位锁存器->27位锁存器），计数器代码（增添8421码 到 7段译码管的转换）及顶层文件代码（输出位数变为27位）。由此也有不小的心得与体会：再设计前一定要先仔细考察硬件电路，一丁点考虑不周导致的后果可能是程序结构，代码的大量修改。

2.系统不能清零:计数器清零信号clr的产生依赖于闸门信号clk1和计数使能信号en，最初设计clk1二分频得到clk0\_5,再将clk0\_5赋给en，并由clk0\_5作为敏感信号触发产生clr信号的进程，系统可以清零。后来为了使代码更加简洁，将clk0\_5省去，clk1二分频直接得到en，并由en作为敏感信号触发产生clr信号的进程，但系统反而不能正确清零，后来迫不得已又改回了原版。由输出信号直接控制进程，细细想来确实不太合适。这让我认识到中间缓冲信号的作用与价值。

3.芯片管脚的问题：6个七段译码管中，后5个均可正常显示，只有第一个的c管和f管不亮。于是换了c管和f管对应的管脚，可还是不行。用逻辑笔测试了一下，发现接入c管和f管的导线都是高电平，但插入c口和f口后变成了低电平，于是怀疑是不是LED灯坏了。借了别组的试验箱上的七段译码管，发现还是同样的问题，联想到接入c管和f管的导线插入小LED灯，灯发光但没有正常的亮，就又把故障锁定在管脚上。这次认真确认了所接管脚都是没有问题的，有烧了一次，果然程序可以正确运行了。之后测试了所有管脚并把失效管脚记录在案，以免后续实验在同样的地方栽跟头。

**小组讨论**

总结了一下简易频率计的设计思路与功能实现，尤其将分频原理理解透彻，以利于后续其他系统的设计；并且对药片装瓶系统的原理图进行研究与分析，为明天的编程做准备。

**~~~~日期8.29~~~~**

开始编写药丸的控制系统，先研究了书上的内容，按照PPT提供的框图进行构思，在纸上画出了模块图并写出了初步的伪代码。然后又借鉴了同组同学分频计的思想。由于我的程序比计数器先开始编写，所以我并没有运用到计数器的相关内容。

我大概一个上午就将代码写完了，但是我知道编译和调试才是最有挑战的。

下午的编译过程遇到很多问题，比如变量与常量的定义方式的不同，integer类型的数据取值范围的确定使用range to 语句。令我印象最深的就是，一个三态门信号的定义语句一直被报错。我咨询了班里的很多同学大家都不知道原因出在哪里。后来上网问别人，才知道同一个singal不可以在两个process中被赋值。其他在编译中遇到的问题大多都是拼写错误，或者是一些比较低级的错误，经过上网搜索、询问同学等方法大多敲定。

**~~~~日期8.30~~~~**

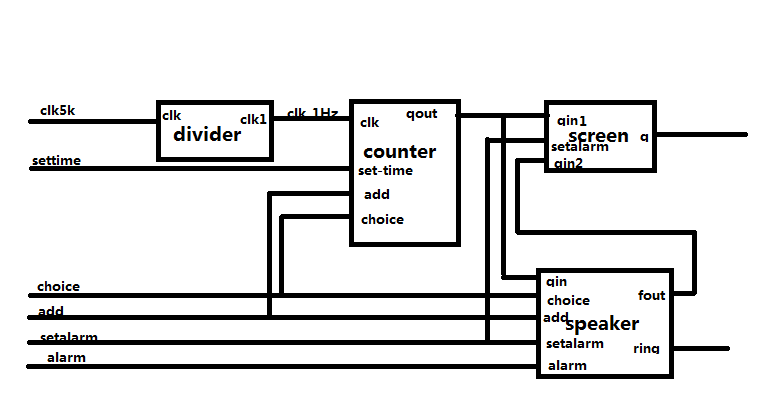
编译还未完全通过，这次报错的原因是一个while语句。系统认为它无法跳出循环。可我的逻辑明明是对的，这绝对不是一个死循环！我询问了好多人，最后咨询了老师。老师给出的答案是，有可能是系统自身的缺陷无法编译通过。于是，我将while循环改成if条件句，终于就可以通过了。

但是将程序写入芯片后，发现示数并不随时钟变化。于是我怀疑是管脚出了问题，就用逻辑笔一个一个管脚进行测量。找出所有坏的管脚。最终连接成功，示数开始变化。

但是问题又出现了，清零功能不起作用。改了程序的好多地方无果，于是怀疑是不是进程之间出现了问题。于是把所有的进程合并到了一起，最终清零功能就正常了。

**~~~~日期8.31~~~~（何芊慧）**

上午开始构思，根据照ppt上边的要求画原理图，给电子钟设置了校时，整点报时，设定定时闹钟等功能，大致如下。



下午正式开始编写程序，按照原理图，先写子程序。

divider是分频计，先将5kHz进行2500分频，然后遇到上升沿即反转，变成一秒一个上升沿的时钟脉冲；

counter是计数器，没有采用24进制和60进制的计数器，没有使用if嵌套，根据进位的规律分成多个if模块，并且counter函数可设置时间，用choice（2:0）来选择需要修改六位数中的哪一位，用add（单次脉冲）来控制增加的量。Counter函数将时间通过screen输出；

speaker用来设定定时闹钟的时间，与counter同理使用choice和add来控制。将counter与speaker分开的原因是在设定闹钟的同时，时钟必须继续往前走，因此counter的计数功能不能停止，alarm对speaker使能，当alarm为1时将counter传输过来的当时时间与设定好的时间进行比较，当它们相等的时候ring变为1，闹钟响；

screen用一个setalarm控制，当setalarm为0（即不设定闹钟）时输出counter中的时间，setalarm为1时输出speaker中的时间，以便使用者知道自己所设定的时间是多少，当闹钟已设定好，setalarm变回0，继续显示counter中的时间。当setalarm、时、分均为0的时候整点报时。

**~~~~日期9.1~~~~**

对昨天写的程序进行调试。出现的第一个问题是counter中没有设置signal中间变量，导致h1<=h1+1这类语句全部出错，于是设置了中间变量，在process的最后再给外部变量赋值。在解决语法问题后编译通过，在tec5中试运行，发现数字灯不亮，排查管脚和导线的问题，换了几个坏掉的管脚和接触不良的线，电子钟基本都有显示，但是显示的是乱码而且没有变化。由于电子钟不动，考虑到是计数器的问题，于是将其他无关部分注释掉，发现电子钟能顺利地变化和进位，于是排查其他问题。问题还没解决好的时候看到了课本上对电子钟程序的要求，发现定时闹钟功能不做要求，但是要使它能够暂停、清零。于是给电子钟增加了暂停、清零的功能，在counter中用stop、clr实现。加了stop、clr两层嵌套之后编译不通过了……原因是GLB超标，使用了49个。想尽办法减少循环，但是还是没有解决GLB的问题，猜想原因是counter的功能太多，想不出结构的改进方式。而且不知道怎么控制定时闹钟的铃音只响五秒，所以决定放弃定时闹钟的功能，在原来的基础上重新编写，画新的原理图。

**~~~~日期9.2~~~~**

调试1日写的新程序，问题多多。首先是GLB的问题，新的程序使用GLB是46个，后来将counter中if嵌套的条件附加，减少了一层循环，GLB减少到33个，还是不能通过编译。接着想到把counter中process的内容拆分成两部分，但是尝试过之后还是不成立，因为两个进程不能给同一个量赋值。最后将6个8421码的输出改成1个24位数的输出，还有speaker中的循环也减少了一层，GLB就通过了编译。编译通过后插线试运行，发现时钟不走动，想了几种办法，最后将清零和暂停改为同步清零和同步暂停，将clk上升沿做为if循环的条件放到最外层，问题才得到解决。解决了这些问题以后发现ring一直高电平，闹铃不停地响，然后为ring输出设了signal，才使闹铃正常工作。

程序运行成功后画原理图，仔细地对照接口名称等细节内容后编译，没有出现什么问题。

**~~~~日期9.3~~~~**

别的组在验收时，老师说分频计尽量不要用锁存器，于是对之前的程序（利用锁存器）进行改进。方便起见，只是将锁存器的锁存信号输入端删去，将锁存器的输入直接赋给输出。期间没有遇到问题，一次成功。并开始整理代码，准备文档。

**~~~~日期9.4~~~~**

原本我们的程序结果是，无论系统运行与否，两个输出分别是每瓶装药片数和已装好的总药片数。但听到老师的新要求，我将程序作了改进：若系统运行，则两个输出分别为当前瓶已装的药片数和已装好的总药片数；若系统停止，则两个输出分别为已装好的总瓶数和已装好的总药片数。

下午，我们三个各自检查自己的程序无误，验收。

**~~~~日期9.5~~~~**

每个组员写自己的调试日志和收获心得。

**~~~~日期9.6~~~~**

每个组员写自己的调试日志和收获心得，最后由温丽云整理、排版。