北京邮电大学课程设计报告

课程设计名称	数字逻辑与数字系统课程设计	学 院	计算机学院	指导教师	靳秀国
班 级	班内序号	学 号	学生姓名	成绩	
2021211304		2021212171	杨晨		
2021211304		2021212484	张梓良		
2021211304		2021211899	蒋礼特		
2021211303		2021211008	芦枷旭		

课程设计内容

实验一、电子钟系统设计

实验目的：熟悉 CPLD 与 VHDL 编程设计，提升团队合作开发能力，培养工程思维。

基本内容：设计一个 24 小时制的时钟并添加设置时间、闹钟等基本功能。

实验方法：VHDL 文本+原理图

团队分工：

1. 系统整体设计：杨晨、张梓良、蒋礼特
2. 资源优化：张梓良

实验二、药片装瓶系统设计

实验目的：熟悉 CPLD 与 VHDL 编程设计，提升团队合作开发能力，培养工程思维。

基本内容：设计一个药片装瓶系统，并添加设置瓶数和每瓶药片数、非法输入检测等基本功能。

实验方法：VHDL 文本+原理图

团队分工：

3. 系统整体设计：全体成员
4. 资源优化：张梓良

实验三、VGA 接口自主探索实验

实验目的:熟悉 CPLD 与 VGA 接口的应用,掌握 CPLD 实现简单图像处理和输出的方法。

基本内容:

1. 理解 VGA 接口的时序和彩色显示原理,掌握通过 VGA 接口输出不同颜色和简单图像的方法。
2. 学习使用 VHDL 描述数字系统的方法,掌握 VHDL 描述 CPLD 的结构和时序的方法。
3. 学习使用 Quartus II 集成开发环境,掌握 Quartus II 用于 CPLD 设计的流程和方法。

实验方法:

1. 分析 VGA 接口的时序参数,确定行同步脉冲周期、场同步脉冲周期和像素周期。
2. 使用 VHDL 描述 CPLD 的结构,包括行列计数器和 VGA 输出接口。
3. 使用 Quartus II 编译 Verilog HDL 代码,进行仿真验证和下载到开发板。
4. 通过修改 VHDL 代码和颜色参数,输出不同的图像效果。

团队分工:

1. 理解 VGA 接口原理和参数:杨晨、张梓良

	2. Verilog HDL 代码编写,Quartus II 工程创建和仿真:杨晨 3. 参数调试:杨晨、张梓良 4. 编写实验报告:杨晨
学生 课程设计 报告 （附页）	
课 程 设 计 成 绩 评 定	遵照实践教学大纲并根据以下四方面综合评定成绩： 1、课程设计目的任务明确，选题符合教学要求，份量及难易程度 2、团队分工是否恰当与合理 3、综合运用所学知识，提高分析问题、解决问题及实践动手能力的效果 4、是否认真、独立完成属于自己的课程设计内容，课程设计报告是否思路清晰、文字通顺、书写规范 **评语**: **成绩**: 指导教师签名： 年　　月　　日

注：评语要体现每个学生的工作情况，可以加页。

目录

实验一 电子钟系统设计

一、 硬件环境描述

- 基于 TEC-8 系统平台
- MAX7000 系列

  目前工业界中速度最快且高度集中的可编程器件系列，一般集成度为 600-5000 个可用门，有 32-256 个宏单元，32-155 个 I/O 引脚。
- CPLD

  复杂可编程逻辑器件，是以乘积项结构方式构成的逻辑器件
- EPM7128SLC84-15

  84 个引脚，其中 5 根用于 ISP（InSystemProgrammable）下载。器件内有 8 个逻辑阵列块， 有 128 个宏单元，每个宏单元都有独立的可编程电源控制， 宏单元内的寄存器具有单独的时钟和复位等信号；有 60 个可 用 I/O 口，可单独配置为输入、输出及双向工作方式；有 2 个全局时钟及一个全局使能端和一个全局清除端。

二、 题目分析

（一） 题目要求

a) 基本功能要求
   i. 实现 24 小时制时钟功能
   ii. 实现整点报时功能
   iii. 实现时间设置功能
   iv. 实现随意切换设置和正常工作模式

b) 附加功能
   i. 实现设置状态时闪烁显示
   ii. 实现音乐整点报时
   iii. 实现闹钟功能

（二） 需求分析

a) 基本功能分析
   i. 实现三个计数器：分别用以时分秒的计数。三个计数器的功能相似，可以考虑设计一个组件复用实现。
   ii. 整点报时功能的实现可以从两方面去考虑：
      1. 每当分钟和秒钟为 59:59 时，下一秒进行整点报时
      2. 每当小时+1 时，进行整点报时

      因此只需要在上述时刻向报时模块传递一个使能信号，让其进行整点报时即可。但需要注意，该使能信号只在整点的那一刻有效，一旦不为整点便无效，所以整点报时模块需要能够做到在收到一个使能信号后，能够持续一段时间报时，这就需要在整点报时模块实现一个计数器来完成。
   iii. 时间设置功能的核心在于能够修改用于计时的三个计数器的计数数

值，所以该功能应该嵌套在计数器模块，通过拨动开关，以开关的高低电平来代表输入的时间的 8421BCD 码。

iv. 工作模式的切换可以通过一个控制模块利用译码器来实现。通过开关的输入，译码出不同的使能信号控制工作状态。

b) 附加功能分析

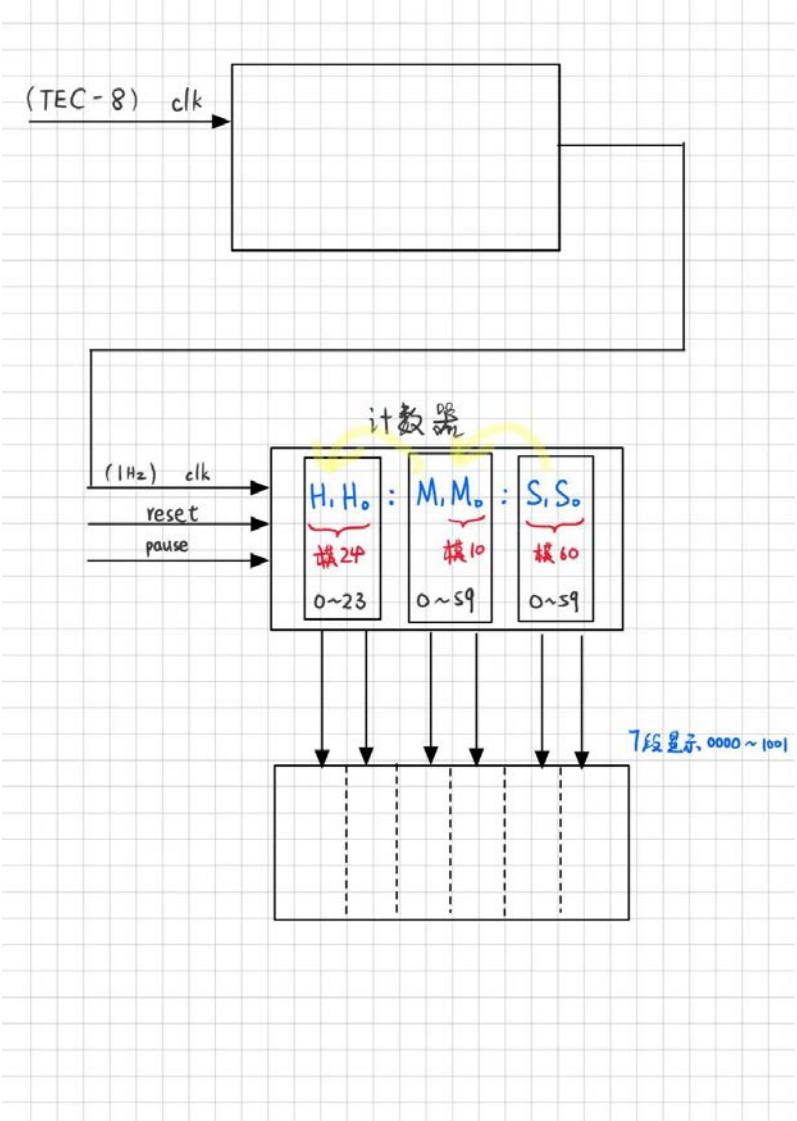
i. 闪烁功能的核心在于能够让数码管交替处于显示状态和不显示状态.，因此需要使用时序逻辑，将数码管的显示状态和时间相关联。一种设计思路是：传入一个 1Hz 的时钟信号，在时钟信号为高电平时正常显示，在时钟信号为低电平时不显示。而让数码管什么都不显示的办法有两种：1. 输出高阻状态“ZZZZ”；2. 输出 8421BCD 码冗余码。

ii. 音乐整点报时功能是基于整点报时功能。二者都是通过将脉冲信号输出到蜂鸣器从而实现报时功能。二者的不同之处在于，音乐整点报时功能能够输出频率变化的脉冲信号，不同的频率能够播放出不同的音符，从而实现播放音乐。

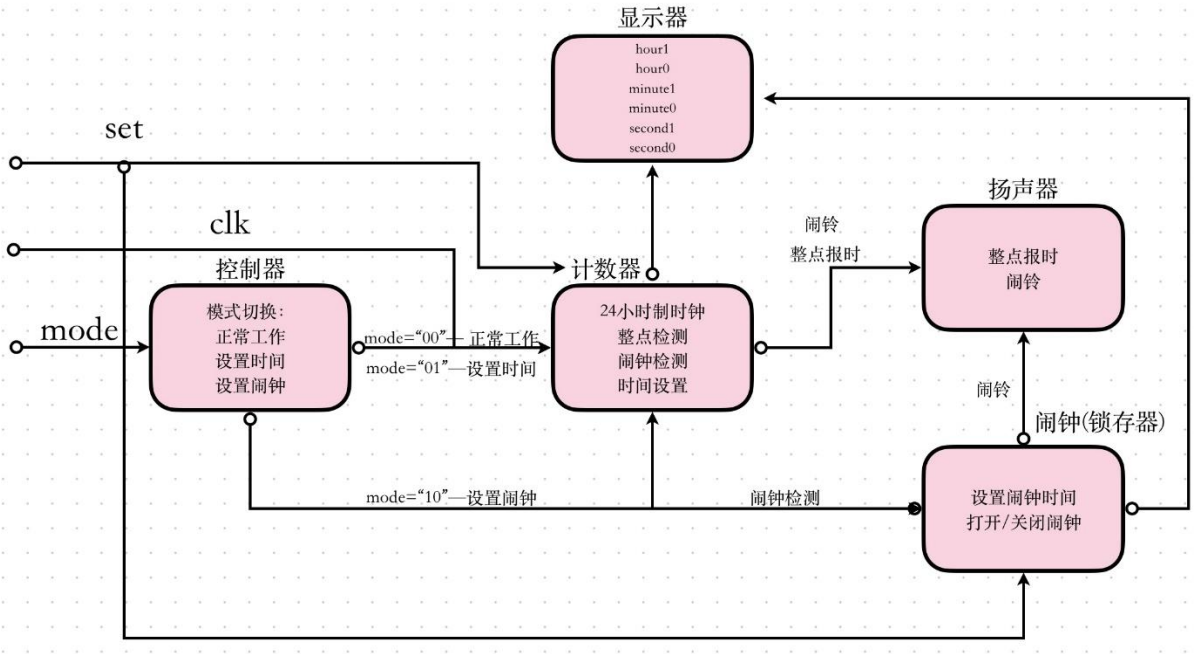
iii. 闹钟功能是依附于时钟功能的，闹钟的设置和时钟的设置类似，同时闹钟一般是无需对秒钟进行设置，只需要精确到分钟即可。闹钟的检测需要通过将置入的闹钟和时钟进行比较，当二者相同时，播放闹铃。此处闹铃可以和音乐整点报时复用。

三、 设计详解

通过需求分析我们可以将电子钟系统大致分为以下几个模块：



在此基础上我们进一步细化功能和模块得到更为详细的设计图：



每个模块便是一个小组件，模块之间通过 siganl 信号传递信息、进行交互。顶层设计将所有小组件按照对应的逻辑连接起来，构成整个电子钟系统设计。经过反复尝试和优化，最终得到以下顶层设计：

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity clock_24 is
port(clk:in std_logic;--100Hz
    clr:in std_logic;--清零 高有效
    alarm_state:in std_logic;--打开闹钟/关闭闹钟
    mode:in std_logic_vector(1 downto 0);--00 正常工作  01 设置时间  11 设置闹钟
    set:in std_logic_vector(1 downto 0);--设置哪一位
    set_vector:in std_logic_vector(6 downto 0);--设置输入
    cp_in:in std_logic;--扬声器输入频率 10000Hz
    cp_out:out std_logic;--扬声器输出频率
    ledout1:out std_logic_vector(6 downto 0);--驱动 7 段发光管
    ledout2:out std_logic_vector(19 downto 0)--5 个七段显示译码器
    );
end clock_24;



architecture func of clock_24 is

component control--模式切换
port(mode:in std_logic_vector(1 downto 0);
    set:in std_logic_vector(1 downto 0);--设置哪一位
    ENs:out std_logic_vector(2 downto 0);--闹钟设置、时间设置、正常工作
    Sets:out std_logic_vector(2 downto 0)--设置使能
    );
end component;

component counter1
port(cp:in std_logic;--100Hz
    clr:in std_logic;--清零 高有效
    ENwork:in std_logic;--正常工作
    ENset:in std_logic;--设置时间
    set_vector:in std_logic_vector(6 downto 0);--设置输入
    counter_time1:out std_logic_vector(3 downto 0);--当前时间
    counter_time2:out std_logic_vector(6 downto 0);--七段数码管
    co1:out std_logic;--进位信号
    check:out std_logic--检测信号
```

```
    );
end component;

component counter2
port(limit1:in integer range 9 downto 0;
    limit2:in integer range 5 downto 0;
    cp:in std_logic;--100Hz
    clr:in std_logic;--清零 高有效
    ENwork:in std_logic;--正常工作
    show:in std_logic;--显示选择
    ENset:in std_logic;--设置时间
    ENalarm:in std_logic;--设置闹钟
    set_vector:in std_logic_vector(6 downto 0);--设置输入
    EN_alarm_check:in std_logic;--是否进行闹钟检测
    EN_loudspeaker:out std_logic;--整点报时/闹钟
    counter_time:out std_logic_vector(7 downto 0);--当前时间
    co1:out std_logic;--进位信号
    check:out std_logic--检测信号
    );
end component;

component loudspeaker--扬声器
port(en_music:in std_logic;
    clk:in std_logic;   --1hz
    clk_i:in std_logic; --10000hz
    clk_o:out std_logic
    );
end component;

component div100--100 分频
port(clk_100:in std_logic;
    clk_1:out std_logic
  );
end component;

signal ENs:std_logic_vector(2 downto 0);--状态使能信号
signal Sets:std_logic_vector(2 downto 0);--设置使能信号
signal en_lp2:std_logic;--扬声器使能
signal en_lp3:std_logic;--扬声器使能
signal cp:std_logic;--1Hz
signal co11:std_logic;
signal co12:std_logic;
signal co21:std_logic;
signal co22:std_logic;
```

```
signal co31:std_logic;
signal co32:std_logic;
signal div:std_logic_vector(3 downto 0);

begin
u1:control port map(mode,set,ENs,Sets);
u2:counter1 port map(cp,clr,ENs(0),ENs(1) and Sets(0),set_vector,ledout2(3 downto 0),ledout1
,co11,co12);
u3:counter2 port map(9,5,cp,clr,co11,ENs(2),ENs(1) and Sets(1),ENs(2) and Sets(1),set_vector
,alarm_state,en_lp2,ledout2(11 downto 4),co21,co22);
u4:counter2 port map(3,2,cp,clr,co21,ENs(2),ENs(1) and Sets(2),ENs(2) and Sets(2),set_vector
,alarm_state,en_lp3,ledout2(19 downto 12),co31,co32);
u5:div100 port map(clk,cp);
u6:loudspeaker port map(en_lp3 or (co32 and co22 and co12),cp,cp_in,cp_out);
end func;
```

下面对其中各部分的设计和功能进行详细叙述：

（一）control 部分：控制部件

a) 输入：
   i. mode：用户选择的模式。
   ii. set：处于设置模式时用于确定设置哪一位（时、分、秒）。

b) 输出
   i. ENs：对于不同的模式输出不同的使能信号。
   ii. Sets：对于设置不同的位时输出对应的设置使能。

（二）counter1 部分：秒钟计时部件

a) 输入：
   i. cp：输入的 1Hz 主时间信号。由 div100 部件将 100Hz 频率信号 100 分频得到。
   ii. clr：计数器清零信号。
   iii. ENwork：计数器工作使能信号。
   iv. ENset：设置时间使能信号。
   v. set_vector：设置状态下的输入，通过开关电平表示 8421BCD 码进行输入和信息读取。

b) 输出
   i. counter_time1：秒钟的十位输出。
   ii. counter_time2：秒钟的个位输出。由于个位需要用最低位的数码管显示，因此需要进行七段显示译码，将 8421BCD 码转换成七段数码管对应的电平信号。
   iii. co1：进位输出。作为分钟的计数使能。
   iv. check：闹钟匹配输出。当秒钟和设置的闹钟的秒相等时输出‘1’。

（三）counter2 部分：分钟和时钟计时部件（进行复用）

a) 输入：
   i. limit1：分钟和时钟的个位的限制
   ii. limit2：分钟和时钟的十位的限制

iii. cp：输入的 1Hz 主时间信号。由 div100 部件将 100Hz 频率信号 100 分频得到。
iv. clr：计数器清零信号。
v. ENwork：计数器工作使能信号。
vi. show：在设置闹钟时要显示所设置的闹钟时间，由于闹钟是内置于时钟部件的，所以要做时钟和闹钟之间的显示切换，通过 show 的值来决定显示时钟还是闹钟。
vii. ENset：设置时钟使能。
viii. ENalarm：设置闹钟使能。
ix. set_vector： 设置时钟或闹钟时的输入。
x. EN_alarm_check：开启闹钟使能。

b) 输出：

i. EN_loudspeaker：蜂鸣器使能。整点报时或者闹钟响铃时为‘1’。
ii. counter_time：接两根数码管，显示分钟时间或者时钟时间。
iii. co1：分钟的进位输出。作为时钟的计数使能。
iv. check：闹钟匹配输出。当分钟/时钟和设置的闹钟的分钟/时钟相等时输出‘1’。

（四）loudspeaker 部分：蜂鸣器部件。用以整点报时和闹钟提醒。

a) 输入：

i. en_music：蜂鸣器使能信号。
ii. clk：输入的 1Hz 时钟信号。用以实现一个模 6 的计数器，控制播放音乐的时间以及确定每 1s 播放哪种频率的音符。
iii. clk_i：输入的 10000Hz 的脉冲信号。将该脉冲信号进行不同的分频得到不同频率的音符。

b) 输出：

i. clk_o：输出的不同频率的音符。实现音乐播放的效果。

（五）div100 部分：100 分频器。将 100Hz 的脉冲信号 100 分频得到 1Hz 的主时钟信号。

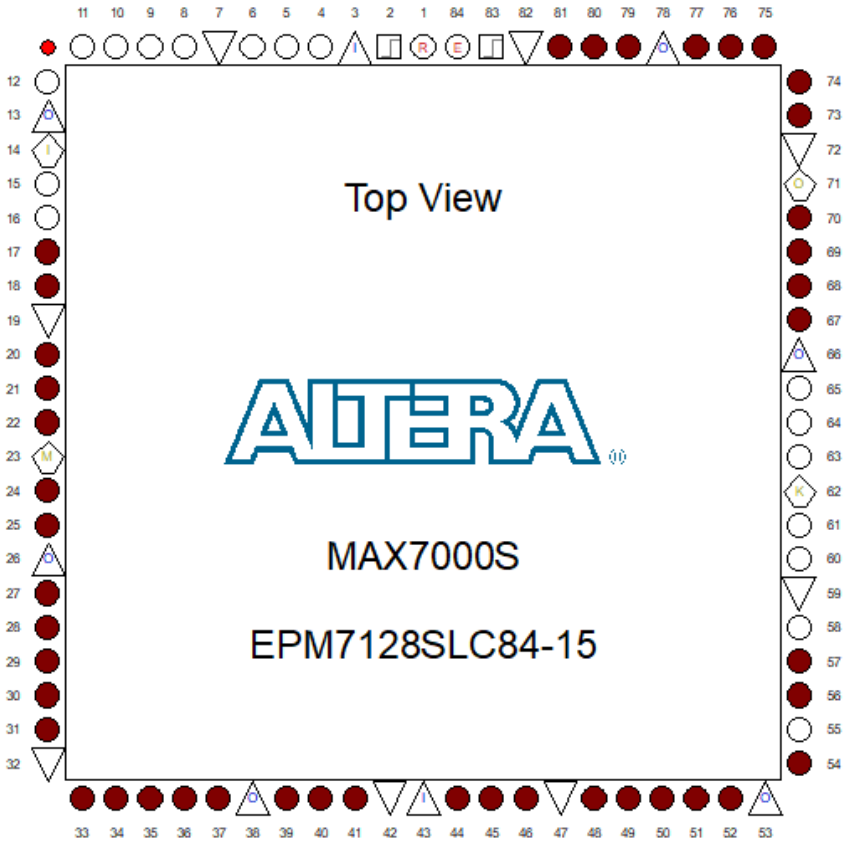
a) 输入：

i. clk_100：输入的 100Hz 脉冲信号。

b) 输出：

i. clk_1：输出的 1Hz 时钟信号。

以上各个部件通过 signal 信号进行联系和交互，协同完成电子钟系统功能。最后附上引脚分配图：



四、 团队分工

（一）张梓良

a) 设计图（第 2 版）
b) 顶层设计和维护
c) 完成时钟设计
d) control 部分设计
e) loudspeaker 部分设计和闹钟检测功能设计
f) 设置时间功能设计
g) 优化程序架构添加闹钟功能
h) 音乐播放的设计
i) 资源优化

（二）杨晨

a) 设计图（第 1 版）
b) 完成分钟和小时设计
c) 闹钟部分设计
d) 时钟部分改写和优化
e) 资源优化

（三）蒋礼特

a) display 部分设计
b) 闪烁功能实现

c) 音乐播放的设计
d) 100 分频器的设计

五、 实现功能

基本实现所有基本功能和附加功能，同时有一些添加的额外功能。

1. 基本功能的实现：
   i. 实现 24 小时制时钟功能
   ii. 实现整点报时功能
   iii. 实现时间设置功能.
   iv. 实现随意切换设置和正常工作模式
2. 附加功能的实现：
   i. 实现设置状态时闪烁显示
   ii. 实现音乐整点报时
   iii. 实现闹钟功能
3. 额外功能的实现：
   i. 实现闹钟的开启和关闭
   ii. 实现设置闹钟和时钟并发进行。设置闹钟并不会影响到时钟正常工作。

六、 所遇问题及解决方案

1. 问题：仿真时报错 error: zero-time oscillation in node "|counter|lpm_add_sub:add5|addcore:adder|addcore:adder[0]|_~1" at time 302.5 ns. check the design or vector source file for combinational loop
   解决方案：这是由于在小时和分种的进程中未加 cp 敏感信号而导致错误，添加上 cp 信号即可。
2. 问题：分钟达到 59 时，时钟持续增加。
   解决方案：未添加秒钟是否达到 59 的判断，在时钟使能的条件中加上这一判断。
3. 问题：编译时报错 Error: Quartus II Analysis & Synthesis was unsuccessful. 1 error, 6 warnings Error: Peak virtual memory: 242 megabytes Error: Processing ended: Wed Mar 15 10:02:46 2023 Error: Elapsed time: 00:00:01 Error: Total CPU time (on all processors): 00:0
   解决方案：例化语句中 std_logic_vector 的位宽与调用时传入的位宽不一致，修改传入信号的位宽。
4. 问题：程序无法烧入芯片。
   解决方案：笔记本电脑上是 64 位的 quartus II，需要使用实验室中的 32 位的进行烧录。
5. 问题：7 段数码管译码错误，导致将程序烧入实验箱后，秒钟的个位无法正常显示。
   解决方案：不同实验箱上的 7 段数码管实际上采取的不同译码方式，应该根据实验箱来调整自己的译码方式，因此可能出现在某一个实验箱上秒钟的个位无法正常显示（有两段数码管错位），而在另外一个实

验箱上却可以正常显示的现象。

6. 问题：秒钟的十位无法闪烁。
   解决方案：实验箱上倒数第二个数码管无法采用输出高阻"ZZZZ"的方式使得其熄灭，需要采用输出 8421BCD 码的冗余码的方式使得其熄灭。
7. 问题：程序烧录到实验箱上出现一些未知的错误。
   解决方案：未将 TEC-8 实验箱上的控制转换开关拨到硬连线处，拨至硬连线处即可。
8. 问题：整点无法报时。
   解决方案：未将蜂鸣器的输入 cp 加入敏感信号表中，使得 cp 变化后无法进入 process 中。同时若实验箱上的控制转换开关未拨到硬连线处，蜂鸣器也有可能播放不出声音。
9. 问题：在两个进程同时对一个信号或输出赋值，编译时报错。
   解决方案：进程之间是并行的，不能在两个进程同时对一个信号或输出赋值。
10. 问题：编译时报错 Error:Location assignment for nodes in LAB_E contains 42 programmable interconnect arrays (PIA),but the LAB cannot contain more hean 36 PIAs; Error:Can’t place 8 sharable expanders in LAB LAB_E because the LAB can contain only 5 sharable expanders
    解决方案：芯片被使用的逻辑块过多，可以共享的拓展数量不够，尝试=整合模块从而减少逻辑块的使用。
11. 问题：资源使用超限。
    解决方案：
    i. 优化程序架构。
    ii. 组件复用。
    iii. 减少位向量的个数和每个位向量的位数。
    iv. 消减进程中不必要的敏感变量。
    v. 采用 when-else 语句替换进程内部的 if-else 语句对信号进行赋值。
    vi. 七段译码管的功能内置于秒计数器，从而减少位向量的传递。
    vii. 用 case 语句替换 if 语句，由于进程里的 if 语句存在优先级，而 case 语句各种情况之间都是并行的，if 语句消耗更多的芯片资源（实际上 case 语句也不一定能够比 if 语句节省资源，还是需要去尝试）。
    viii. 闹钟功能内置于时钟部件，从而消减向时钟部件传递闹钟位向量的开销。
    ix. 减少分频开销，尽量用合适频率作为原始输入频率。
    x. 资源共享。减少相同模块的反复调用。
    xi. 在用 case 语句描述组合逻辑电路时，应尽量避免使用 null 来表示未用条件下的操作行为，因为这样会综合出锁存器，消耗更多的资源。

七、 设计与调试小结

电子钟系统的总体逻辑并不复杂，整体设计的难点在于如何在有限的资源下

实现所有的功能。这就需要我们高度优化每个组件内部的逻辑，在实现相同的功能的前提下，尽量减少资源的占用。设计时应注意不能一味的为了整合模块而整合，也不能为了划分模块而划分，两个模块是否需要整合，或者一个模块是否需要划分成多个模块的出发点都在于是否能够节省更多的资源，这需要反复的思考和推敲，需要学会从硬件层面去思考问题，要能够分析实现一个功能会需要哪些门电路和触发器，怎样设计能够减少触发器的数量从而节省资源。我们的设计要尽量符合规范，使得综合出来的电路不会含有冗余项，要尽量去使用 quartus 自带组件从而减少 macrocells。

在设计的过程中要去大胆尝试，小心求证。硬件层面的设计存在很多不确定性，但这并不意味着是依靠运气去尝试，应该是在不够理想的设计上针对相应问题作出优化和调整，一切的调试和优化都应该是在有逻辑和理论知识的支撑上进行的操作。

同时要学会去分析实验箱的工作行为，当出现和自身设想不一致的行为时，要分别从自身理论设计和实验箱硬件是否存在问题两方面去考虑。当是自身理论设计存在问题时，就应调整设计思路，根据程序在硬件上错误表现行为对代码进行相应调整，优化设计。当是实验箱硬件层面的问题时，就应先去分析可能是哪部分硬件存在问题，找出问题源头所在之处，判断能否依靠更换实验台上一两个硬件器件从而解决问题，若不能则更换实验箱。

总体而言，设计和调试是相辅相成的，每一次优化的设计会有更好的调试结果，而每一次调试结果中的问题又推动设计的优化。理论的设计需要通过实践验证其正确性、健壮性。需要学会记录和分析调试过程中遇到的问题，逐渐培养工程师素养。

八、 创新点

电子中系统广泛应用于我们的日常生活中，因此在设计的过程中，我们应该在实验的要求上结合实际进行设计和创新。以下是我们所设计的电子中系统中的创新点：

1. 设置时间采用 8421BCD 码的方式，而不是按下 QD 而加 1 的方式，这样更加高效、让用户能够有更好的体验。
2. 设置闹钟和时钟正常工作能够并发。结合实际而言，在设置闹钟时，时间并不应该停止，所以应该让两个模式能够并发工作。
3. 闹钟能够开启和关闭。用户设置的闹钟并不一定在每一天都需要，因此需要考虑闹钟能够开启和关闭。当不需要时，关闭闹钟，需要时，打开闹钟。
4. 设置时间时，只有正在设置的位会闪烁，其余还未设置的位正常显示。这样方便用户判断自己正在设置哪一位，提升用户体验。

实验二 药片装瓶系统设计

一、硬件环境描述

- 基于TEC-8系统平台
- MAX7000系列

> 目前工业界中速度最快且高度集中的可编程器件系列，一般集成度为 *600-5000* 个可用门，有*32-256*个宏单元，*32-155*个*I/O*引脚。

- CPLD

> 复杂可编程逻辑器件，是以乘积项结构方式构成的逻辑器件

- EPM7128SLC84-15

> *84*个引脚，其中*5*根用于*ISP*（*InSystemProgrammable*）下载。器件内有*8*个逻辑阵列块， 有*128*个宏单元，每个宏单元都有独立的可编程电源控制， 宏单元内的寄存器具有单独的时钟和复位等信号；有*60*个可 用*I/O*口，可单独配置为输入、输出及双向工作方式；有*2*个全局时钟及一个全局使能端和一个全局清除端。

二、题目分析

（一）题目要求

1. 基本功能
    a. 实现药片装瓶功能
    b. 能够同时显示药瓶以及药片数量
    c. 有工作状态以及告警指示
    d. 有清零状态、设置状态和工作状态，并实现状态间正确切换设置
2. 附加功能
    a. 实现设置状态时闪烁显示
    b. 工作状态时可以切换显示初始设置

c. 可以设定每瓶药片数以及总药片数量限定

（二）需求分析

1. 基本需求分析

a. 实现三个计数器

i. 计数器1记录每瓶药片数量

ii. 计数器2记录已装瓶数

> 计数器*2*和计数器*1*功能相似，可以进行复用，以减少芯片资源开销。
>
> 同时计数器*2*以计数器*1*的进位信号为计数使能信号，二者存在级联关系。

iii. 计数器3记录总药片数量

> 计数器*3*和计数器*1*同步计数。

b. 实现数码管的显示切换

> 由于*TEC-8*系统平台只提供了*6*个数码管，因此无法同时显示出每瓶药片数量（占用*2*个数码管）、总药片数量（占用*3*个数码管）、已装瓶数（占用*2*个数码管）。因此必须实现显示切换，让相同的数码管在不同的显示状态下显示不同的信息。

c. 用LED实现工作状态和告警状态的区分

> 发光二极管模拟机电装置的输出和告警，绿灯点亮表示启动装瓶进行中；红灯点亮表示装瓶完成，机电装置关闭；红灯闪烁表示输入错误或者其他故障显示。

d. 实现一个状态控制器

> 利用实验台上的两个开关的四种组合来表示四种不同的状态：设置状态、工作状态、显示初始设置状态、显示已装瓶数状态。开关处于哪一种组合就发出对应的状态使能信号。
>
> 同时利用一个开关来表示*clr*是否使能，控制计数器是否清零。

2. 附加需求分析

a. 实现设置时闪烁显示

> 类似于实验一，将时钟信号加入显示模块，实现时钟信号高电平时正常显示，低电平时熄灭数码管以实现闪烁功能。

b. 工作状态显示初始设置

> 是基本需求分析中的*b*、*d*两点的综合，在*d*中状态控制器切换到相应的状态下，控制显示模块数码管进行显示切换。

c. 设定每瓶药片数以及总药片数量限定

> 由于如果采用*8421BCD*码的方式直接设置总药片数量，至少需要*12*个开关（因为总药片数量上限为*900*），太浪费开关资源，而如果选择设置每瓶药片数量以及装瓶数量的限定，不仅能够达到相同的效果（*3*者之间存在等式关系：每瓶药片数量*装瓶数量=总药片数量，因此知道其中任意*2*个，可以推出第*3*个），而且还只需要*8*个开关便可以实现，节省了开关资源。同时将设置每瓶药片数和瓶数的功能内置于计数器*1*、*2*，可以减少传递位向量的开销。

（三）场景应用

1. 应用场景

> 药片装瓶系统是一种自动化装瓶流水线，这种系统广泛应用于制药厂、医院等。例如，在制药厂中，药片装瓶系统可以大幅度提高生产效率和药品质量（降低药物污染的风险），同时减少人工操作的错误和工作量。

2. 容错设计思考

> 在药品生产中，错误可能导致严重后果，因此药片装瓶系统需要具备高度的容错能力。基于*VHDL*实现的药片装瓶系统可以采用以下容错策略：

a. 冗余设计

> 在系统中添加冗余电路，以防止单个电路出现故障导致整个系统崩溃。

b. 状态监测

> 通过监测系统各个模块的状态，及时检测并处理异常情况，避免故障蔓延。

c. 纠错编码：

> 采用纠错编码算法，对数据进行校验和修复，保证数据的正确性。

3. 安全设计思考

> 药品生产涉及到人类的健康和安全，因此安全设计是非常重要的。基于*VHDL*实现的药片装瓶系统可以采用以下安全策略：

a. 权限控制

> 设置不同层级的权限，只有授权人员才能对系统进行修改和操作。

b. 安全防范

> 在系统中添加安全措施，如安全门、防火墙等，保证生产过程中的安全性。

c. 环境监测

> 在药品生产过程中，监测环境参数，如温度、湿度、光照等，及时发现并处理异常情况，保证生产质量和安全。

4. 用户体验

a. 界面设计

> 采用友好的界面设计，使操作人员易于理解和操作，减少人为操作误差。

b. 自适应控制

> 根据系统的反馈信息和环境参数，自动调整药片装瓶的速度和瓶子的数量，保证生产效率和质量。

c. 数据分析

> 采集和分析生产数据，为生产过程的优化和质量控制提供数据支持。

三、设计详解

基于二中对题目的分析，再结合理论课讲解的顶层设计：图***1***和图***2***。我们将整体设计模块大致分为：

- 药片总量计数器模块
- 单瓶药片计数器模块
- 装瓶量计数器模块
- 控制模块
- 显示模块

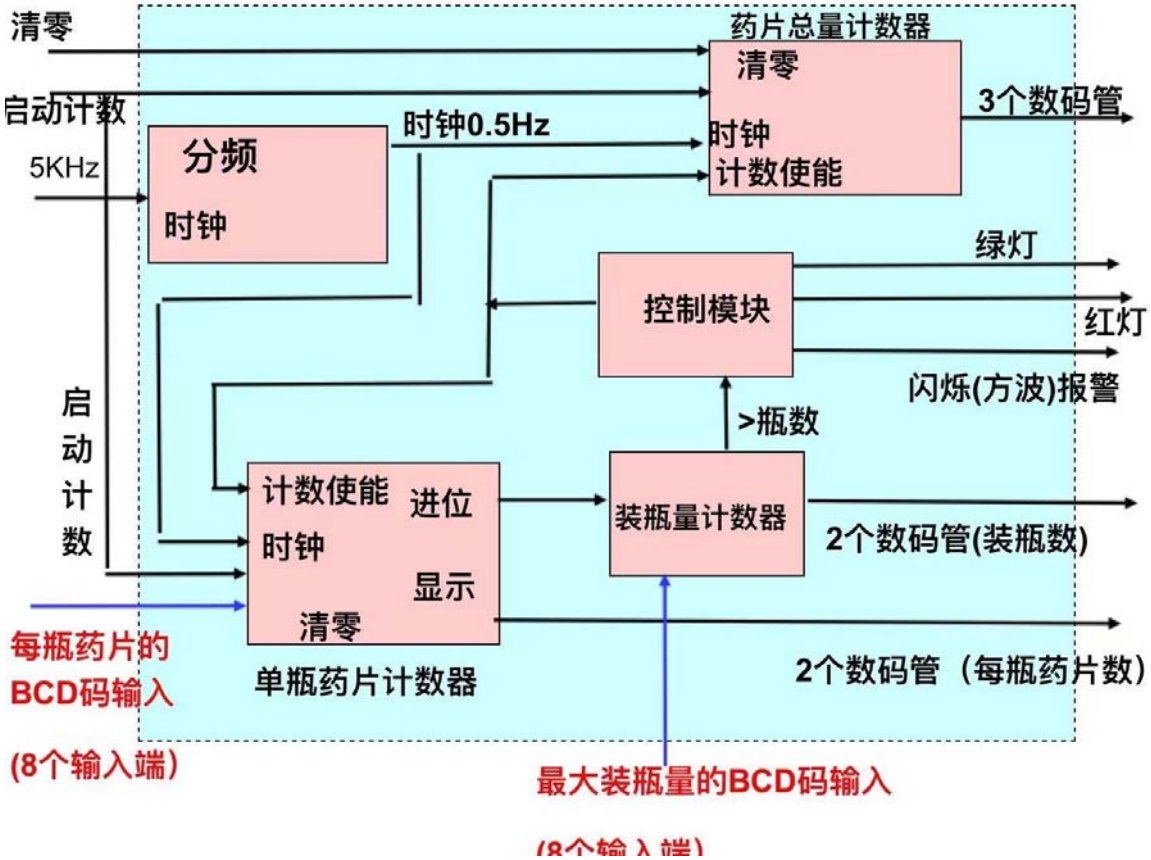


图1 理论课顶层设计原理图

图2 生产线装瓶显示系统说明

每个模块就是一个底层元件，由顶层设计将各个模块统一起来，并通过`signal`实现各个模块的交互，由此我们完成了第***1***版顶层设计图，如下图所示：

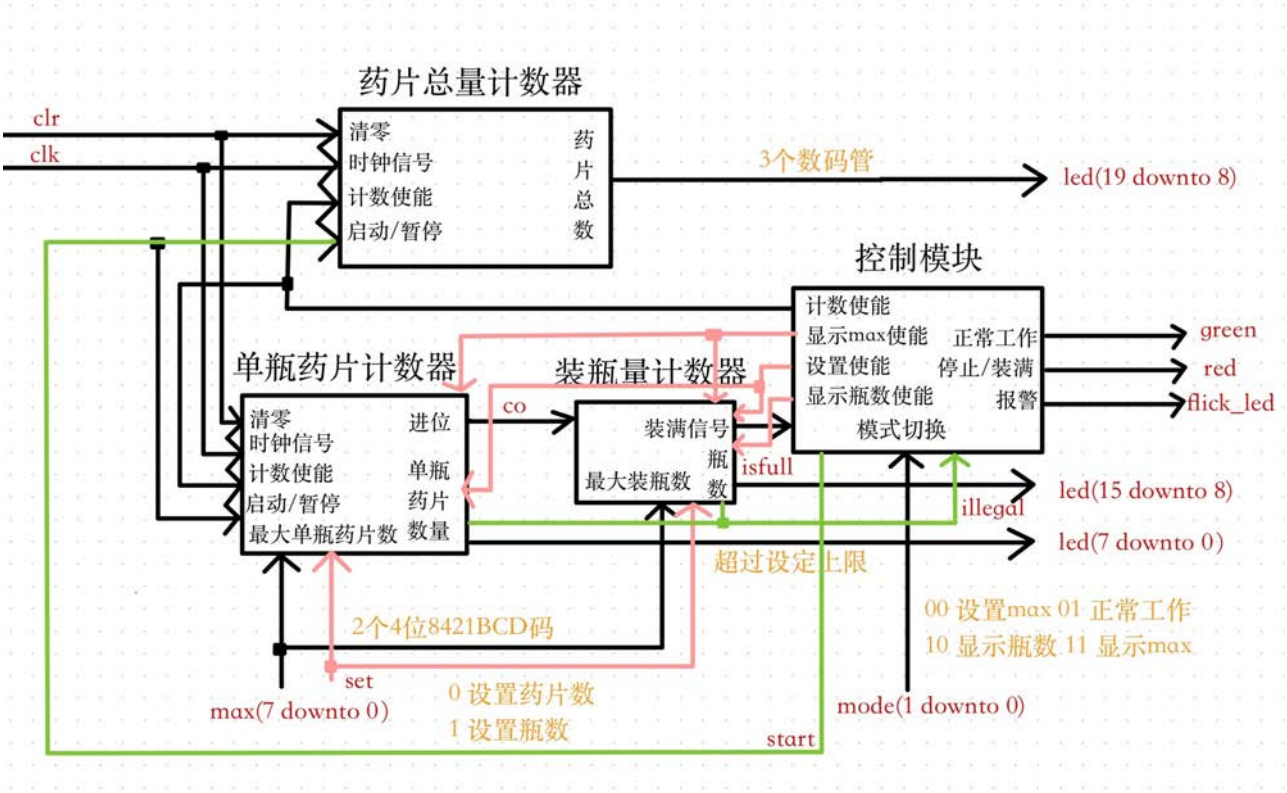


图3 第1版顶层设计

其中单瓶药片计数器和装瓶量计数器模块由于功能相似，因此采用相同的底层元件复用实现以节约宝贵的芯片资源。

在第1版的顶层原理图设计上我们发现了一个问题：如果采用将数量信息作为`signal`传递给显示模块，这样综合出来的设计会占用大量的芯片资源，这是因为在模块与模块间传递信息的过程中，`signal`是`std_logic_vector`类型的，且位数较多，所以占用资源较多。但如果省略掉显示模块，又无法实现显示的切换功能。

为此我们采用部分内置显示、部分通过显示模块显示的方法来解决资源占用和显示切换功能之间的矛盾。我们将每瓶药片数量内置显示，将总药片数量和已装瓶数通过显示模块显示，显示模块根据状态信息，选择二者之一进行显示。这样设计既降低了资源的占用，同时又保留了显示切换的功能。综上所述，我们设计了第***2***版顶层设计，如下图所示：

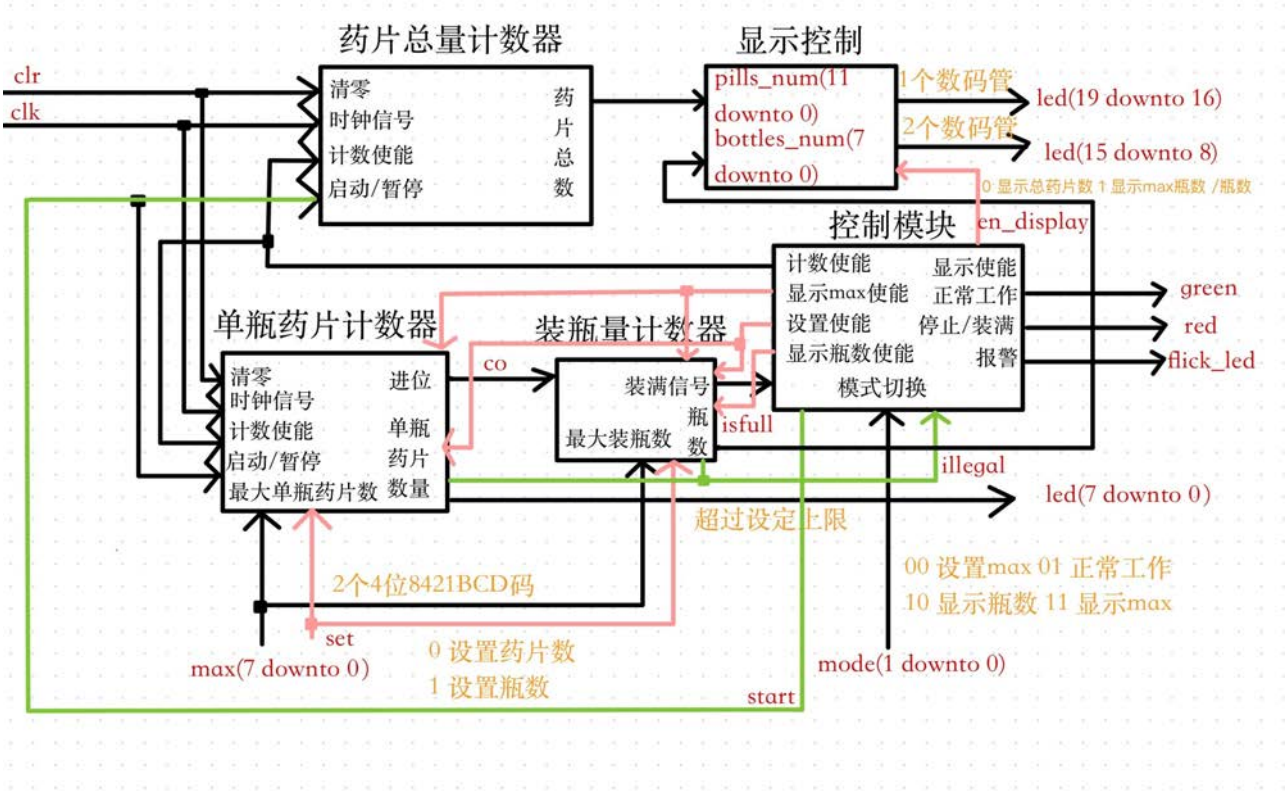


图4 第2版顶层设计

在原理图的基础上，我们搭建出了整体的顶层框架：

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity pill is
port(cp:in std_logic;--100Hz
    clk:in std_logic;--时钟信号
    clr:in std_logic;--清零信号
    mode:in std_logic_vector(1 downto 0);--模式切换
    max:in std_logic_vector(7 downto 0);--设定上限
    set:in std_logic;--0 设置药片数 1 设置瓶数
    start:in std_logic;--开始/暂停
    led:out std_logic_vector(19 downto 0);--数码管显示
    green:out std_logic;--绿灯
    red:out std_logic;--红灯
    loudspeaker:out std_logic--报警
);
end pill;

architecture func of pill is

```

```
--00 显示设置的每瓶药片数最大值 和 设置的瓶数最大值（闪烁）
--01 显示当前药片总数 和 当前单瓶药片的数量
--10 显示当前瓶数
--11 显示设置的每瓶药片数最大值 和 设置的瓶数最大值
component control
port(cp:in std_logic;--100Hz;
    clk:in std_logic;--闪烁时钟信号
    mode:in std_logic_vector(1 downto 0);--模式切换
    isfull:in std_logic;--装满信号
    illegal:in std_logic;--非法输入
    start:in std_logic;--开始/暂停
    green:out std_logic;--绿灯
    red:out std_logic;--红灯
    en:out std_logic_vector(3 downto 0);--使能信号 00-0001 01-
0010 10-0100 11-1000
    en_display:out std_logic;--显示使能
    loudspeaker:out std_logic--报警
);
end component;

component total_pills
port(clr:in std_logic;--清零信号
    clk:in std_logic;--时钟信号
    en:in std_logic;--使能信号
    pills_num:out std_logic_vector(11 downto 0)--总药片数
);
end component;

component per_bottle
port(clr:in std_logic;--清零信号
    clk:in std_logic;--时钟信号
    en:in std_logic;--使能信号
    max:in std_logic_vector(7 downto 0);--设定上限
    en_max:in std_logic;--显示max使能
    en_set:in std_logic;--设置使能
    set:in std_logic;--0 设置药片数
    co:out std_logic;--进位信号
    led:out std_logic_vector(7 downto 0);--数码管显示单瓶药片数
    illegal:out std_logic--非法输入
);
end component;

```

```
component display
port(pills_num:in std_logic_vector(11 downto 0);--总药片数
    bottles_num:in std_logic_vector(7 downto 0);--瓶数/最大瓶数
    en_display:in std_logic;--显示使能
    led:out std_logic_vector(11 downto 0)--数码管显示
);
end component;

signal isfull1:std_logic;--装满信号
signal co:std_logic;--进位信号
signal illegal1:std_logic;--非法输入
signal illegal2:std_logic;--非法输入
signal en1:std_logic_vector(3 downto 0);--使能信号
signal en_display1:std_logic;--显示使能
signal pills_num1:std_logic_vector(11 downto 0);--总药片数
signal bottles_num1:std_logic_vector(7 downto 0);--瓶数/最大瓶数


begin

u1:control port map(cp,clk,mode,isfull1,illegal1 or
illegal2,start,green,red,en1,en_display1,loudspeaker);
u2:total_pills port map(clr,clk,en1(1),pills_num1);
u3:per_bottle port
map(clr,clk,co,max,en1(3),en1(0),not(set),isfull1,bottles_num1,i
llegal2);
u4:per_bottle port
map(clr,clk,en1(1),max,en1(3),en1(0),set,co,led(7 downto
0),illegal1);
u5:display port map(pills_num1,bottles_num1,en_display1,led(19
downto 8));

end func;
```

在该顶层设计下，我们只需要实现在其中声明的4个底层元件便可大致实现药片装瓶系统。这体现出了VHDL层次化、结构化编程的优势：结构清晰，任务细化，便于实现。下面给出整个工程的具体结构

- pill(顶层)
    - control(控制模块)
    - total_pills(总药片数模块)

- per_bottle(单瓶药片数&瓶数模块)
- display(显示切换模块)

下面详细介绍各个模块的功能和输入输出：

1. control部分：控制部件。
    a. 输入：
        i. cp：输入的100Hz脉冲信号。用于蜂鸣器发出报警。
        ii. clk：输入的1Hz主时间信号。用于控制状态灯闪烁。
        iii. mode：用户选择的模式。
        iv. isfull：装满提示灯使能信号。用于控制装满时红灯点亮提示用户已经装满。
        v. illegal：非法输入提示灯使能信号。用于控制输入非法数据时红灯闪烁提示用户输入了非法数据。
        vi. start：开始/暂停。用于控制系统开始/暂停工作。
    b. 输出：
        i. green：绿灯使能信号。用于控制绿灯是否点亮。
        ii. red：红灯使能信号。用于控制红灯是否点亮。
        iii. en：对于不同的模式输出不同的使能信号。
        iv. en_display：显示切换使能信号。用于控制七段数码管的显示切换。
        v. loudspeaker：蜂鸣器报警使能信号。用于控制蜂鸣器以1s为周期间歇发出100Hz报警信号。
2. total_pills部分：药片总片数计数部件。
    a. 输入：
        i. clr：清零信号。用于计数器清零。
        ii. clk：1Hz的主时钟信号。用于控制计数。
        iii. en：使能信号。用于控制开始/暂停计数。
    b. 输出：
        i. pills_num：总药片数。将总药片数的8421BCD码输出到数码管进行显示。
3. per_bottle部分：每瓶药片数计数部件/已装瓶数计数部件（进行复用）。
    a. 输入：
        i. clr：清零信号。用于计数器清零。
        ii. clk：1Hz的主时钟信号。用于控制计数。
        iii. en：使能信号。用于控制开始/暂停计数。
        iv. max：设定瓶数和每瓶药片数。设置状态下的输入，通过开关电平表示8421BCD码进行输入和信息读取。
        v. en_max：显示初始设定使能。控制显示模块进行显示切换，显示初始设定的瓶数和每瓶药片数。

vi. en_set：设置使能。
vii. set：等于'0'时设置药片数，等于'1'时设置瓶数。

b. 输出：

i. co：每瓶药片数计数部件的进位输出。作为已装瓶数计数部件的使能信号。
ii. led：计数结果输出。每瓶药片数计数部件直接接数码管进行显示，已装瓶数计数部件将其传入显示部件进行显示。
iii. illegal：非法输入的使能输出。传递给控制模块，用于控制是否处于非法输入状态。

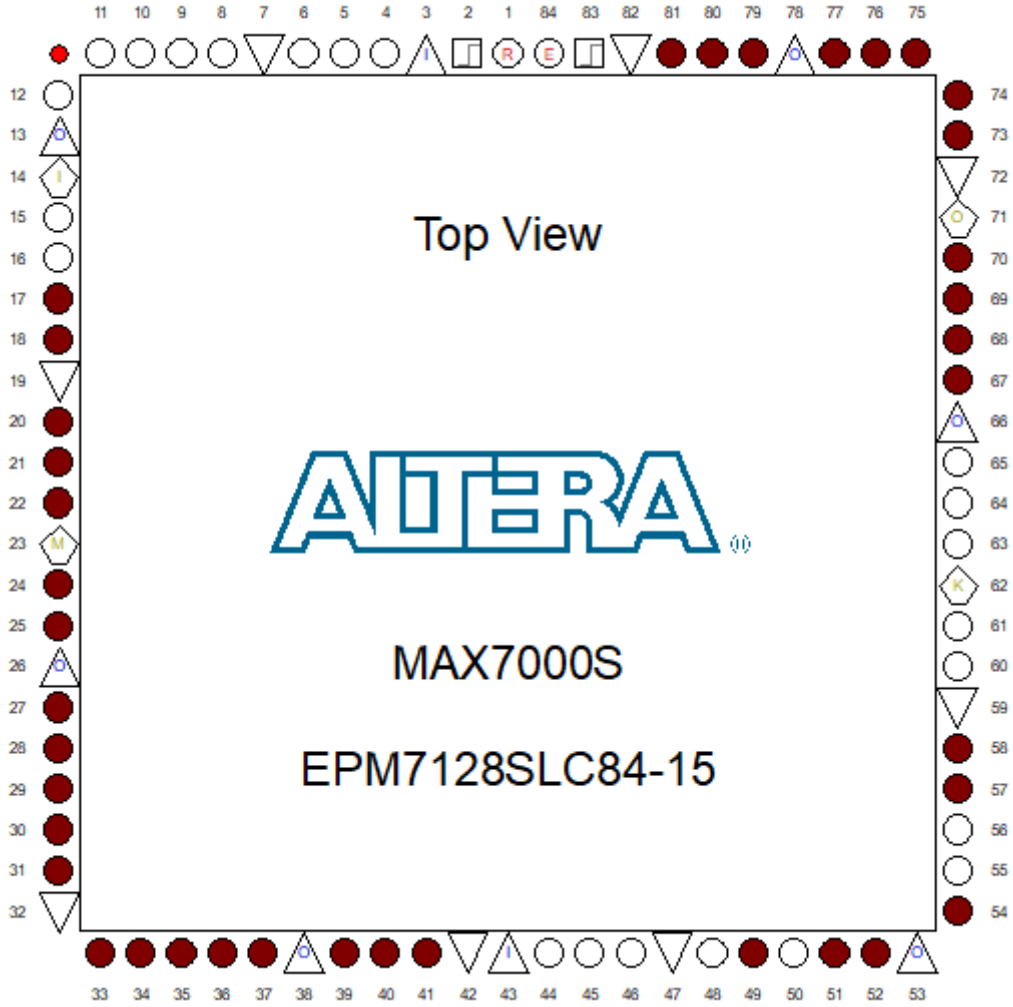
4. display部分：显示部件。在传入的显示内容中根据不同状态选取相应内容在高三位数码管上进行显示。

a. 输入：

i. pills_num：药片总片数计数部件传入的药片总数的8421BCD码。
ii. bottles_num：已装瓶数计数部件传入的瓶数/最大瓶数的8421BCD码。
iii. en_display：显示切换使能信号。默认显示pills_num，当使能时，显示bottles_num。从而实现显示的切换。
iv. led：传递给高三位数码管显示内容的8421BCD码。

以上各个部件通过signal信号进行联系和交互，协同完成药片装瓶系统功能。

最后附上引脚分配图：



四、团队分工

张梓良

- 设计图（第1版）和设计图（第2版）
- 顶层设计（第2版）
- 单瓶药片计数器设计（第2版）和装瓶量计数器设计（第3版）
- 非法输入检测设计和报警提示设计
- 调试和优化

杨晨

- 药片总量计数器设计（第1、2版）
- 调试和优化

蒋礼特

- 单瓶药片计数器设计（第1版）
- 装瓶量计数器设计（第1版）

芦枷旭

- 顶层设计（第1版）

五、实现功能

基本实现所有基本功能和附加功能，同时有一些添加的额外功能。

1. 基本功能的实现：
    a. 实现药片装瓶功能
    b. 能够同时显示药瓶以及药片数量
    c. 有工作状态以及告警指示
    d. 有清零状态、设置状态和工作状态，并实现状态间正确切换设置
2. 附加功能的实现：
    a. 实现设置状态时闪烁显示
    b. 工作状态时可以切换显示初始设置
    c. 可以设定每瓶药片数以及总药片数量限定
3. 额外功能的实现：
    a. 实现对8421BCD码的冗余码的非法检测，当用户输入冗余码时系统会认定该输入无效，会丢弃输入信息，并发出报警提示。
    b. 实现对设定时超限的提示。系统允许最多设定30瓶，最多设定每瓶装30片。当超限时系统会闪烁红灯并发出报警提示。

六、所遇问题及解决方案

日期	问题	解决方案
4.3	将两段8421BCD码直接与整数进行比较，导致意外错误	将两段8421BCD码拆开后分别与整数进行比较
4.3	在一个process中对一个out引脚多次赋值	1.采用先对一个signal赋值，再将该signal的值赋给out 2.采用条件赋值when-else语句并行对out赋值

日期	问题	解决方案
4.7	倒数第二段数码管无法闪烁	无法用输出高阻"ZZZZ"的方式使得第二段数码管熄灭。采用输出8421BCD码的冗余码的方式使得其熄灭
4.7	装满后仍然继续装瓶	装满后，未将装瓶使能信号设置为低电平（高有效）。 在装瓶信号的赋值表达式中与上装满信号的非：`en(1) <= '1' and start and (not isfull)`
4.7	错误的将每瓶药片数装满后再收到一个激励信号后药片数设置成0，导致每瓶药片数和总药片数在时间上不同步	采用模值的方式来表示，例如每瓶最多能装20片，我们将计数器设置为模20的计数器，当计数到19时，下一个计数应为0，用0来代替最大值，最大值和0相互重叠。
5.4	无法存储住初始设定的瓶数上限和每瓶药片数上限	实验箱寄存器存在问题，更换实验箱得以解决

七、设计与调试小结

药片装瓶系统的总体设计难点在于如何利用有限的数码管显示资源实现多种信息的显示，这就必须要实现显示的切换。与此同时又带来一个与之相矛盾的问题，当显示信息比较多时，需要使用较多的位向量，将位向量传递给显示模块又需要占用大量的资源。这个问题我们在电子钟系统的设计中也曾遇到过，可以采取类似的处理办法，将显示内置到计数器模块。在该实验中，我们采取了一种折中的办法，将每瓶药片数的显示内置到其计数器部件内，直接将计数器部件的输出接2段数码管。将总药片数和已装瓶数先传输到显示模块，显示模块根据所处工作状态在二者中选择其一进行显示。显示模块起到了一个中转、判断和选择的作用。采取这样折中的办法既能节省资源，同时也能够以一种比较简易、用户友好的方式进行显示切换。

同时药片装瓶系统的设计需要考虑实际生产线的真实情况。需要有容错设计和安全设计，且能够提供给用户比较好的操控体验。所以需要对非法输入进行检测，既不能允许用户输入8421BCD码的冗余项，也不能允许用户输入超出上限的初始设定，同时能够以蜂鸣器报警、红灯闪烁的方式来提醒用户发生了非法输入的情况。且系统的操作应该通俗易懂，能够让用户快速上手，提供较为良好的用户体验。

总而言之，药片装瓶系统相较于电子钟系统而言，在资源优化上没有那么高的要求。但需要我们更多的结合实际，考虑生产线上的真实情况，去设计我们的系统。操作简便，防呆防错等问题都应该在设计时仔细思考， 应努力贴近实际应用去设计我们的系统。

八、创新点

1. 设置每瓶药片数和总瓶数时采用8421BCD码的方式，而不是按下QD而加1的方式，这样更加高效、让用户能够有更好的体验。
2. 具有检错和容错能力。用户输入8421BCD码的冗余项和非法数据时能够报警提示。
3. 设置每瓶药片数和总瓶数时，只有正在设置的位会闪烁，其余还未设置的位正常显示。这样方便用户判断自己正在设置哪一位，提升用户体验。
4. 停止状态灯有点亮、熄灭、闪烁三种工作状态，分别代表停止工作、正在工作以及发生错误三种情况。

实验三 VGA 接口自主探索实验

1 实验内容

1.1 实验目的

(1) 学习 VGA 接口的工作原理和在显示器上显示某种特定图形的方法。

(2) 掌握数字逻辑系统的设计方法。

(3) 掌握 EDA 软件 Quartus II 的基本使用方法。

(4) 掌握用 VHDL 语言设计复杂数字电路的方法。

1.2 实验内容及要求

(1) 在 VGA 接口显示器上显示出下列图形：横彩条、竖彩条、彩色方格和全屏同一彩色。其中横彩条要包括黑、黄、红、品红、绿、青、黄、白 8 种颜色，每种颜色彩条宽度基本相等。同样竖彩条也要包括黑、黄、红、品红、绿、青、黄、白 8 种颜色，每种颜色彩条宽度基本相等。

(2) 内部设置一个 2 位的模式计数器。当 CLR 为低电平时，模式计数器复位为 00，当 QD 的上升沿到来后，模式计数器加 1。当模式计数器为 00 时，显示横彩条；当模式计数器为 01 时，显示竖彩条；当模式计数器为 10 时，显示彩色方格；当模式计数器为 11 时，显示同一种颜色。

2 VGA 接口设计

2.1 VGA 原理介绍

VGA 彩色显示器 (640×480/60Hz) 显示过程中所必需的信号，除 R、G、B 三基色信号外，行同步 HS 和场同步 VS 也是非常重要的两个信号。在显示器显示过程中，HS 和 VS 的极性可正可负，显示器内可自动转换为正极性逻辑。现以正极性为例，说明 CRT 的工作过程：R、G、B 为正极性信号，即高电平有效。当 VS=0，HS=0，CRT 显示的内容为亮的过程，即正向扫描过程约为 26μS，当一行扫描完毕，行同步 HS=1，约需 6μS；其间，CRT 扫描产生消隐，电子束回到 CRT 左边下一行的起始位置 (X=0，Y=1)；当扫描完 480 行后，CRT 的场同步 VS=1，产生场同步使扫描线回到 CRT 的

第一行第一列 (X=0，Y=0) 处 (约为两个行周期)，HS 和 VS 的时序如图 1 所示。

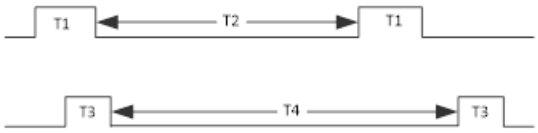


图 1　HS 和 VS 时序图

2.2 VGA 颜色编码表

表 2　VGA 颜色编码表

颜色	黑	绿	蓝	青	红	黄	粉	白
R	0	0	0	0	1	1	1	1
G	0	0	1	1	0	0	1	1
B	0	1	0	1	0	1	0	1

2.3 VGA 接口驱动

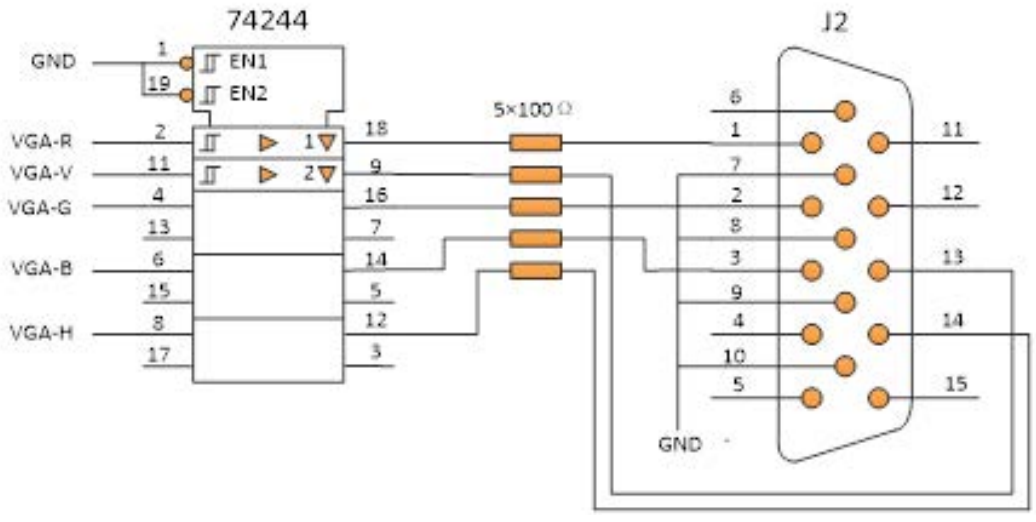


图 2　VGA 接口驱动电路

图 3 中，J2 是一个 15 芯的插座，与个人计算机 PC 上的显示器插座相同。VGA 接口的控制信号 VGA-R(红)、VGA-G(绿)、VGA-B(蓝)、VGA-H(行同步)、VGA-V(场

同步) 经 74244 驱动后通过 100 欧姆电阻送往插座 J2

3 实验设备

1. 个人计算机 1 台
2. TEC-8 实验系统 1 台
3. 个人计算机 PC 用的显示器 1 台
4. 双踪示波器 1 台
5. 万用表 1 只

4 软件设计

4.1 主程序设计

主程序流程图如图 3 和图 4所示, 首先根据时钟脉冲产生水平计数器和垂直计数器, 再根据计数器，设置好合适的同步信号。之后根据计数器将屏幕划分为均匀的区间，在不同的区间里，输出不同的颜色信号

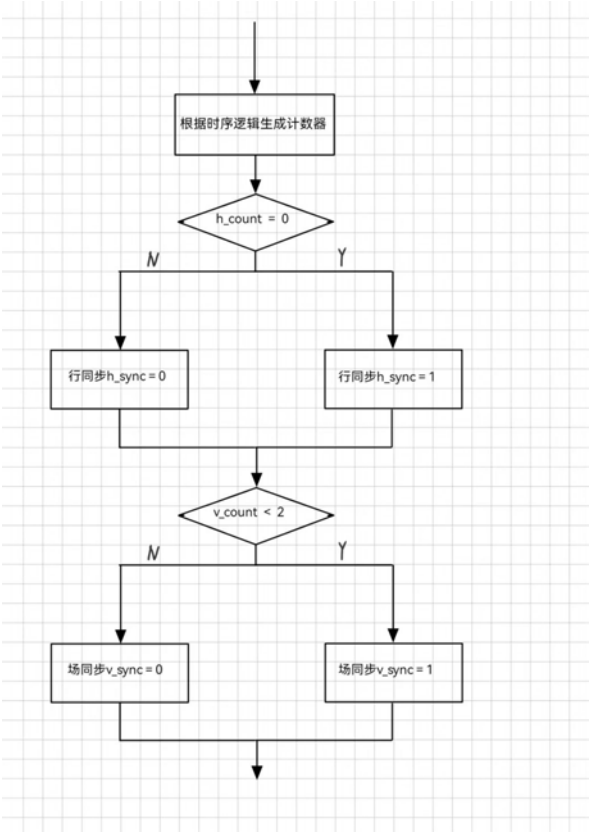


图 3　主程序流程图

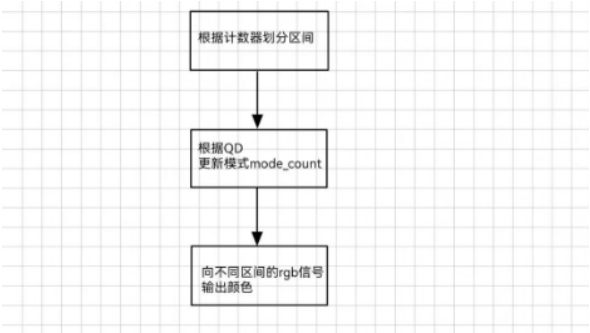


图 4　主程序流程图

4.2 创建计数器和区间划分

创建计数器和区间划分如图 5 所示。

首先，在时钟脉冲 clk 的上升沿，对水平计数器进行累加。将v_count设置为模 32 计数器的方式 (5 位标准逻辑矢量型)，之后在进位的时候对v_count做累加运算，设置v_count为模 525 计数器。h_count和v_count的初值均为 0。

根据水平计数器的范围，去掉行同步区间后，将水平计数器划分为 10 个区间；根据垂直计数器的范围，去掉场同步区间后，将垂直计数器划分为 8 个区间

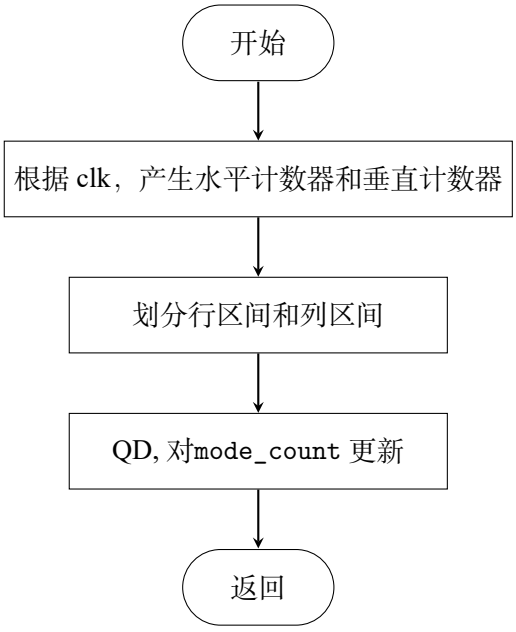


图 5　创建计数器和区间划分

4.3 纯色显示设计和同步周期确定

任何颜色的正确输出，都是建立在正确的行同步周期和场同步周期上的。所以必须调整到一个合适的行同步信号和场同步信号。所有颜色显示中，纯色显示最为简单。不妨以某一种纯色为基准，反复调整行同步信号和场同步信号的周期和周期宽度，直至纯色能够正常输出。

在调试过程中，关键在于必须将行同步信号和场同步信号放在最前方。

在 VGA 显示器出现之前，CRT（阴极射线管）显示器是主流的显示技术。在 CRT 显示器中，电子束从屏幕的左上角开始扫描，然后从左到右扫描整个行，然后下移一行并再次从左到右扫描，如此往复，直到扫描整个屏幕。为了保证显示的正确性，需要在每行结束和每个场结束时进行同步。

在 VGA 彩条显示中，场同步和行同步信号被放置在开始位置是为了保持与 CRT 显示器的兼容性。此外，由于 VGA 信号是模拟信号，受到干扰的可能性比数字信号更大，放置同步信号在开始位置可以更好地保证信号的稳定性和正确性。

因此，将场同步和行同步信号放置在开始位置是为了与 CRT 显示器保持兼容性，并确保 VGA 显示的稳定性和正确性，以确保 VGA 信号在各种显示器上都能够正常工作。

在 VGA 显示时，虽然标准的 640×480 分辨率是常用的分辨率之一，但是实际上在 VGA 显示时，使用的是 800×525 的分辨率。

这是因为 VGA 显示时，需要考虑到显示器的同步信号和帧率等因素，因此需要一定的空间来容纳这些信号。具体来说，每个 VGA 帧包含两个同步信号（水平同步和垂直同步），以及一些前后的黑色空白区域，以确保同步信号的正确性和稳定性。因此，为了容纳这些信号和空白区域，VGA 显示器实际上需要比标准分辨率更高的分辨率来显示图像。

由于实验室的主时钟频率是 1MHz，而标准的 640×480 的 60Hz 显示器，需要的时钟频率是 20MHz。故水平计数器的扫描周期要根据实验室的情况设置成 32μs（$\frac{1}{60} \times \frac{1}{525} \approx 32$）。在实际调试的过程中，我们还发现行同步的时间必须控制在 1 2μs 的时间内，而非 6μs，否则无法正常显示

由于每种颜色的输出时机不一致，且不同的颜色输出顺序会导致不同的颜色输出时机。因此必须确定一个合适的颜色的输出顺序。

4.4 彩条显示设计

彩条区间划分的关键代码如图下所示。

```
--根据水平计数，进行列划分
process(h_count, v_count)
begin
    if h_count = "00000" then
        h_end <= '1';
        h_part <= "0000";
    else
        h_end <= '0';
        if h_count >= "00001" and h_count <= "00011" then
            h_part <= "0001";
        elsif h_count >= "00100" and h_count <= "00110" then
            h_part <= "0010";
        elsif h_count >= "00111" and h_count <= "01001" then
            h_part <= "0011";
        elsif h_count >= "01010" and h_count <= "01011" then
            h_part <= "0100";
        elsif h_count >= "01100" and h_count <= "01110" then
            h_part <= "0101";
        elsif h_count >= "01111" and h_count <= "10001" then
            h_part <= "0110";
        elsif h_count >= "10010" and h_count <= "10011" then
            h_part <= "0111";
        elsif h_count >= "10100" and h_count <= "10110" then
            h_part <= "0001";
        elsif h_count >= "10111" and h_count <= "11001" then
            h_part <= "1000";
        elsif h_count >= "11010" and h_count <= "11111" then
            h_part <= "0001";
        end if;
    end if;
    LinePart <= h_part;
```

```
--根据垂直计数，做行划分
    if v_count < 2 then
        finish <= '1';
        v_part <= "0000";
    else
        finish<= '0';
        if v_count < 65 then
            v_part <= "0001";
        elsif v_count < 128 then
            v_part <= "0010";
        elsif v_count < 191 then
            v_part <= "0011";
        elsif v_count < 254 then
            v_part <= "0100";
        elsif v_count < 319 then
            v_part <= "0101";
        elsif v_count < 392 then
            v_part <= "0110";
        elsif v_count < 456 then
            v_part <= "0111";
        elsif v_count < 525 then
            v_part <= "1000";
        end if;
    end if;
    RowPart <= v_part;
end process;
```

VGA 信号传输方式通常有模拟信号和数字信号两种方式。模拟信号传输方式下，彩条的顺序可能会受到传输线路的影响，导致不同的彩条顺序。而数字信号传输方式下，彩条的顺序通常是固定的，因为数字信号传输方式下，彩条的顺序由显示器解码器中的固定算法来确定。

实验室中的 VGA 信号传输是模拟信号，因此颜色的顺序和出现时机会尤为重要。经过尝试，我们确定了将水平计数器划分为 10 个区间，将垂直计数器划分为 8 个区间，这样可以得到 80 个时隙来产生不同的彩条组合。在每个时隙内，我们可以

选择显示红、绿、蓝三种基本颜色的任意组合。例如，在第一个时隙我们可以选择显示红色，在第二个时隙显示绿色，第三个时隙显示蓝色，第四个时隙显示黄色 (红 + 绿), 第五个时隙显示品红 (红 + 蓝) 等等。通过控制这 80 个时隙内的颜色组合，我们可以产生众多组合，从而构成颜色丰富的彩条。

在实验中, 我们通过烧录 VHDL 程序设计好的彩条图像数据到 TEC-8 实验台，实验台再根据这些数据在对应的时隙内选择显示相应的颜色，从而在屏幕上显示出预定的彩条图像。

当颜色切换时，如果显示器响应速度不够快，则可能会出现颜色失真或错位的问题。所以在图像数据的产生过程中，我们要确保不同彩条之间的颜色与顺序是正确且合适的，否则在 LCD 屏幕上显示的彩条图像可能会出现颜色错乱的情况，甚至不能正常显示。

经过尝试，我们确定了横彩条的颜色显示顺序为黑、红、粉、绿、青、黄、白、黑、蓝、黑；竖彩条的颜色显示顺序为黑、红、粉、绿、青、黄、白、蓝；纯色显示为粉色。

选择这个彩条颜色显示顺序有以下几个原因:

包含白色和黑色, 可以测试彩屏的颜色显示范围。

包含相近色彩如红色和粉红色, 可以保证彩屏在显示相近色彩时过渡平滑，防止出现色块和色斑。

4.5 关键算法

```
case mode_count is
when "00" :
if RowPart = "0001" then
    ...(根据LinePart赋值temp_color)
elsif  RowPart = "0010" then
    ...(根据LinePart赋值temp_color)
elsif RowPart = "0011" then
    ...(根据LinePart赋值temp_color)
...
else
    temp_color <= "000";
```

```
end if;
when "01" :
if LinePart = "0001" then
    temp_color <= black;
elsif LinePart = "0010" then
    temp_color <= red;
...
end if;
when "10" :
if RowPart = "0001" then
    ...(根据LinePart赋值temp_color)
elsif RowPart = "0010" then
    ...(根据LinePart赋值temp_color)
...
else
    temp_color <= "000";
end if;
when "11" :
if finish = '0' and h_end = '0' then
    temp_color <= pink;
else
    temp_color <= "000";
end if;
end case;
end process;
```

5 调试心得总结

将每次能够正确工作的代码保存为一个版本,并添加版本号及日期,如v0.1_20230515等。这样在后续调试中出现问题时，我们可以回退到上一个工作版本，继续进行调试。

在修改代码前，应对当前工作版本进行备份，这样如果修改后的代码不能工作，我们可以直接还原到修改前的状态，避免大范围修改导致难以调试的情况出现。

修改代码时，应逐步进行修改和测试，而非一次性完全重写。逐步修改可以快速定位问题代码，便于调试。完全重写的代码如果不能工作，调试难度会大大增加。

应记录每次修改的详细内容，如果新修改的代码不能工作，我们可以根据修改记录逐步还原，而不是盲目还原，提高调试效率。

6 实验仿真电路图

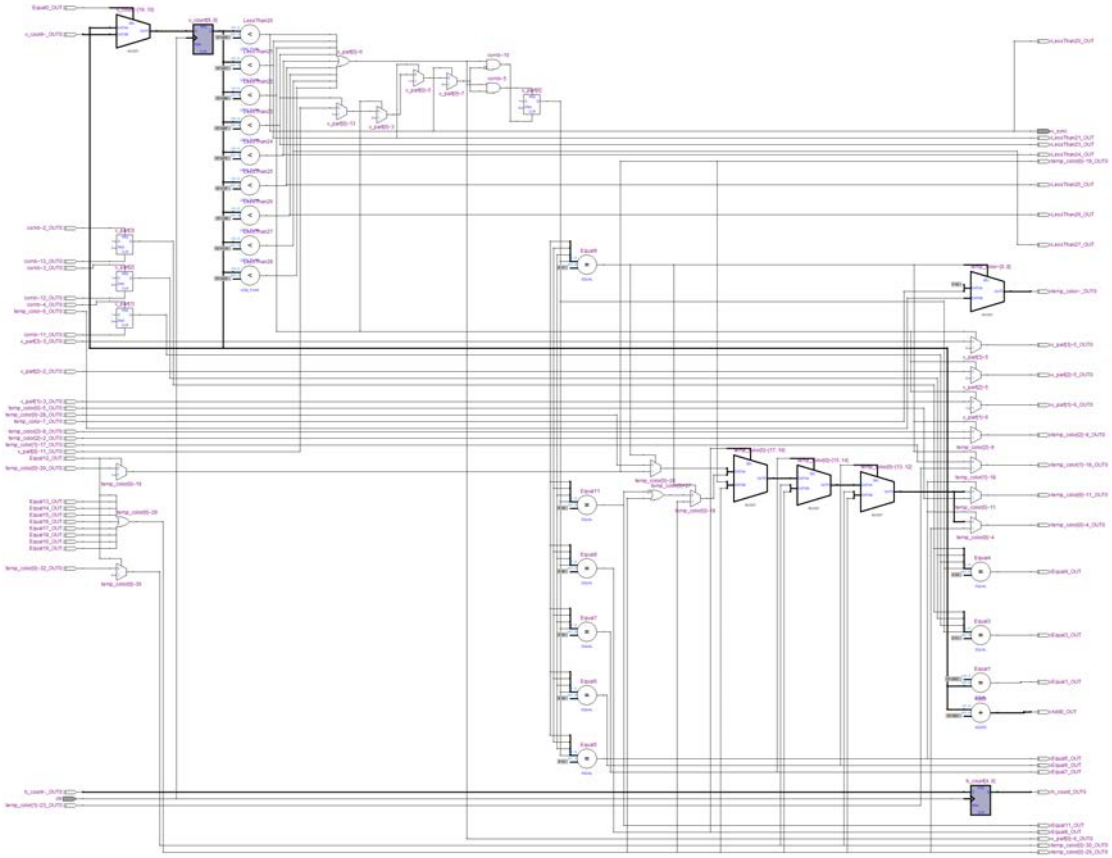


图 6　实验仿真电路图

调试日志

3.11 实验一第一次讨论

选题

两个必做　时限: 3 周
三个选做 ：选 1-2 个　时限: 4 周

第一个实验：电子钟

vhdl + 原理图混编(顶层) 3.12 完成

- 模式切换
- 计数器
  - 24 小时时钟
  - 整点检测
    - 音乐播放
  - 闹钟检测
    - 音乐播放
  - 时间设置
    - 闪烁
- 闹钟
  - 设置
  - 打开/关闭
- 扬声器
- 显示器
- 顶层(2 人) 3.12 完成
  - 计数器(时 分 秒)(1-2 人)(张，杨)
    - 功能
      - 闹钟功能
      - 整点报时
    - in
      - clk(std_logic 1Hz)
      - ckr(std_logic 低有效)
      - pause/continue(暂停/继续)
      - set(std_logic_vector)
        - 000 设置秒的个位 个位闪烁
        - 001 设置秒的十位 十位闪烁
        - 010 设置分的个位
        - 011 设置分的十位
        - 100 设置时的个位
        - 101 设置时的十位
        - 冗余码 全部闪烁
      - QD(std_logic)
        - 快进一下

        - out
            - hour1(std_logic_vector)
            - hour0
            - minute1
            - minute0
            - second1
            - second0
    - 控制器(1 人)(芦)
        - 功能
            - 模式切换
        - in
            - mode(std_logic_vector)
                - 00 正常工作
                - 01 设置时间
                - 10 设置闹钟
        - out
            - ENTime
            - ENSet
            - ENClock
    - 显示器(显示状态 六段数码管)(1 人)(蒋)
        - in
            - hour1(std_logic_vector)
            - hour0
            - minute1
            - minute0
            - second1
            - second0
        - out
            - led5(std_logic_vector)
            - led4
            - led3
            - led2
            - led1
            - led0(译码)
    - 闹钟
    - 扬声器

3.13 实验一第二次讨论

3.14-3.15 任务

- control
- counter 部分实现
- display

任务分配

- counter (张、杨)
- display(蒋)
- control(芦)

3.16 实验一第三次讨论

实验一剩余任务

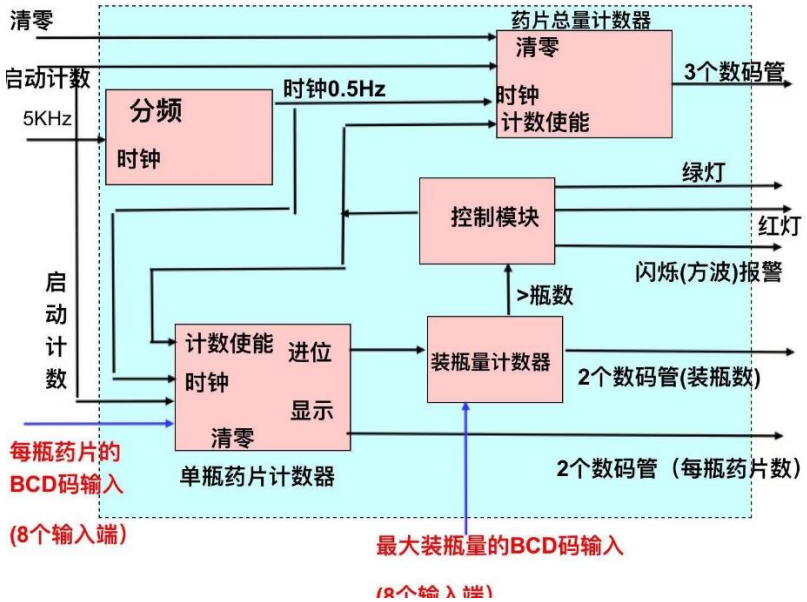
- 设置时间功能
- 整点报时
- 闪烁功能
- *音乐

任务分配

- 设置时间功能(张)
- 整点报时(杨)
- 闪烁(蒋)
- 实验二需求分析，设计原理图(芦)(ps: 可参考实验一的原理图)

3.27 实验二第一次讨论

实验二模块划分



任务分工

- 顶层设计 （芦）
- 药片总量计数器（杨）

- 单瓶药片计数器（蒋）
- 装瓶量计数器（蒋）
- 实验一整理、报告编写（张）

3.30 实验二第二次讨论

顶层设计讲解

任务分配
张：control
杨：total_pills, display
蒋：per_bottle
芦：bottles
注意：自己的模块写完后，一定要仿真看看是否能够实现模拟功能

开发中遇到问题

实验一

- 3.14
  - 在小时和分种的进程中未加 cp 敏感信号导致错误(仿真时)
    - error: zero-time oscillation in node "|counter|lpm_add_sub:add5|addcore:adder|addcore:adder[0]|_~1" at time 302.5 ns. check the design or vector source file for combinational loop.
  - 小时进程中未检测秒达到 59 秒，导致达到 59 分钟后，小时一直进位
  - 分钟进程中对于个位达到 9 分钟后未归为 0
- 3.15
  - 例化语句中位宽与调用位宽不一样，导致编译错误
    - Error: Quartus II Analysis & Synthesis was unsuccessful. 1 error, 6 warnings Error: Peak virtual memory: 242 megabytes Error: Processing ended: Wed Mar 15 10:02:46 2023 Error: Elapsed time: 00:00:01 Error: Total CPU time (on all processors): 00:0
  - 程序无法烧入芯片
    - 因为笔记本电脑上是 64 位的 quartus II，需要用 32 位的
  - 7 段数码管译码错误
- 3.20
  - 资源使用超限
    - 优化各模块语句，减少 display 模块的调用，减少 if 语句的使用
    - 压缩时间存储，对时间的存储用二进制码代替 8421BCD 码，节省 7 个引脚的使用
    - 串行语句修改为并行语句

- 3.22
  - 秒的十位无法闪烁
    - 无法用输出高阻"ZZZZ"的方式使得显示秒的十位的数码管熄灭
    - 采用输出 8421BCD 码的冗余码的方式使得其熄灭
  - 秒的个位和十位无法设置时间
    - 未将 TEC-8 实验台上的控制转换开关拨到硬连线处
  - 整点无法报时
    - 未将喇叭的输入 cp 加入敏感信号表中，使得 cp 变化后无法进入 process 中
    - 未将 TEC-8 实验台上的控制转换开关拨到硬连线处

实验二

- 4.3
  - 将两段 8421BCD 码直接与整数进行比较，导致意外错误
    - 将两段 8421BCD 码拆开后分别与整数进行比较
  - 在一个 process 中对一个 out 引脚多次赋值
    - 采用先对一个 signal 赋值，再将该 signal 的值赋给 out
    - 采用条件赋值 when-else 语句并行对 out 赋值
- 4.7
  - 秒的十位无法闪烁
    - 无法用输出高阻"ZZZZ"的方式使得显示秒的十位的数码管熄灭
    - 采用输出 8421BCD 码的冗余码的方式使得其熄灭
  - 装满后仍然继续装瓶
    - 装满后，未将装瓶使能信号设置为低电平（高有效）
    - 在装瓶信号的赋值表达式中与上装满信号的非：en(1) <= '1' and start and (not isfull)
  - 错误的将每瓶药片数装满后再收到一个激励信号后药片数设置成 0
    - 应设置为 1
- 4.14
  - 每瓶药片数装满后瓶数并没有+1
  - 所有瓶都装满后没有停止装瓶

心得总结

张梓良 2021212484

通过本次数字逻辑课程设计，让我在问题分析、编程设计、团队合作等方面都有了很大提升。在设计的过程中我深入了解了数字逻辑的基本原理和设计方法，培养了我在数字电路设计和分析方面的技巧和能力。在这个过程中，我积累了以 下的心得和收获：

首先，数字逻辑课程设计提高了我的问题分析和解决能力。在每个设计任务中，我们都需要仔细的阅读设计要求，根据设计需求并结合应用场景设计出相应的数字系统，并优化该数字系统电路的性能和延迟。在解决这些问题的过程中，我学会了如何分析问题、提出解决方案和评估设计的正确性和可行性的方法，同时也锻炼了我的逻辑思维和解决实际问题的能力。

其次，数字逻辑课程设计提供了实践操作的机会。通过使用 Quartus II 设计平台和 TEC-8 实验箱完成本次课程设计，不仅巩固了我对数字逻辑理论知识的理解和掌握，还增强了我的动手能力和实际操作经验。我学会了正确使用设计工具、进行仿真和调试，并能够将设计结果烧录到实验箱上进行实际测试。

此外，在数字逻辑课程设计中，我还学到了一些重要的实践经验。首先，设计过程中的规范性和准确性至关重要。每一步设计都需要仔细思考和验证，确保电路的正确性和可靠性。其次，对于复杂的电路设计，分模块和层次化设计是非常有效的方法。将整个电路划分为几个子模块，分别进行设计和测试，最后再进行整合，能够提高设计效率和降低错误率。此外，充分利用数字电路模拟软件进行仿真和调试，能够帮助发现问题并优化电路性能。

再者，数字逻辑课程设计培养了我的团队合作和沟通能力。课程要求多人共同完成复杂的逻辑电路的设计和实现。这要求我们有效地分工合作、交流思路和解决合作中的问题。通过这样的合作经历，我学会了团队合作的重要性，提高了沟通和协调的能力。

同时，通过数字逻辑课程设计，我也意识到数字逻辑在现代科技中的广泛应用。数字电路和逻辑电路设计是数字系统、计算机硬件和通信领域的基础，涵盖了从简单的逻辑门到复杂的处理器和通信协议的设计。掌握数字逻辑的知识和技能，将对我的未来学习和职业发展产生积极的影响。

最后，本次课程设计给我留下的最深刻的记忆便是资源优化的过程。如何在实现相同的功能的前提下，尽量减少资源的占用是伴随着数字逻辑课程设计的始终。通过一次次的资源优化的经历，我总结出了以下经验：

设计时应注意不能一味的为了整合模块而整合，也不能为了划分模块而划分，两个模块是否需要整合，或者一个模块是否需要划分成多个模块的出发点都在于 是否能够节省更多的资源，这需要反复的思考和推敲，需要学会从硬件层面去思 考问题，要能够分析实现一个功能会需要哪些门电路和触发器，怎样设计能够减 少触发器的数量从而节省资源。

同时我们的设计要尽量符合规范，使得综合出来的电路不会含有冗余项，要尽量去使用 Quartus II 自带组件从而减少 macrocells。在设计的过程中要去大胆尝试，小心求证。硬件层面的设计存在很多不确定性，但这并不意味着是依靠运气去尝试，应该是在不够理想的设计上针对相应问题作出优化和调整，一切的调试和优化都应该是在有逻辑和理论知识的支撑上进行的操作。同时要学会去分

析实验箱的工作行为，当出现和自身设想不一致的行为时，要分别从自身理论设计和实验箱硬件是否存在问题两方面去考虑。当是自身理论设计存在问题时，就应调整设计思路，根据程序在硬件上错误表现行为对代码进行相应调整，优化设计。当是实验箱硬件层面的问题时，就应先去分析可能是哪部分硬件存在问题，找出问题源头所在之处，判断能否依靠更换实验台上一两个硬件器件从而解决问题，若不能则更换实验箱。同时设计和调试是相辅相成的，每一次优化的设计会有更好的调试结果，而每一次调试结果中的问题又推动设计的优化。理论的设计需要通过实践验证其正确性、健壮性。需要学会记录和分析调试过程中遇到的问题，逐渐培养工程师素养。

总而言之，数字逻辑课程设计是一门富有挑战性和实践性的课程，对我在数字逻辑领域的学习和成长起到了重要作用。如果要用四个字来总结本次数字逻辑课程设计的特点，那便是“多、快、好、省”——实现尽可能多的功能，设计的电路的响应速度尽可能的快，提供给用户的体验尽可能的好，占用芯片的资源尽可能的省。通过本次课程设计，我不仅深入了解了数字电路的原理和设计方法，还提高了问题解决能力、动手实践能力和团队合作能力。这门课程为我今后在数字逻辑和相关领域的学习和职业发展打下了坚实的基础，让我受益匪浅！

心得总结

蒋礼特 2021211899

我在实验一中参与了显示器的实现部分，翻译了七段译码器。在此过程中我发现，实验台的七段对应方式与数字逻辑课程中给出的七段方式不同，七段顺序是倒过来的。在设置数码管不亮的过程中，设为高阻态可以实现多个数码管的不亮。但对于倒数第二个数码管，使用高阻态的效果不好，最终改为了无效信号 1111。

通过这段实验过程，我明白一个道理，实验的内容跟书本上的内容很可能会有所不同，总是按照书上的内容不如自己动手体验一下。同样地，仿真实验也只是一个模仿实验，
同样无法了解到与真实实验之间可能存在的差别。

此外，我在实验一种还参与了扬声器与分频器的设计。将一万 hz 的时钟信号按不同情况分频为三种不同的频率，传给扬声器后可以发出三种不同的声音。同时，捕捉短时间的使能信号也是最大的问题之一。按照一般的设计模式，很容易造成使能信号消失后声音不再继续的情况。为此，我将捕捉使能信号与一个变量的清零关联起来，同时对该变量进行实时监测，每次清零后就开始音乐播放。这样即使使能信号很短，也能每次都将要播放的音乐播放完毕。通过音乐播放功能的实现过程，我学会了检测短时间的使能信号，也了解了音乐的基本实现过程。

在实验二过程中，我首先参与了第一版的装瓶系统和计数系统的设计。设计过程中出现了较多错误，也没能修改完成。因为由于顶层设计的些许简陋，我们制作了第二版的顶层系统，为此选择重新设计各个模块。我在第二版设计过程中再次参与设计了显示器模块，主要内容大概与实验一相同。虽然显示器内容没有什么更多的收获，但有过第一版的系统设计过程，也加深了我对计数系统等模块的理解。

除此之外，我与另一位组员一起尝试了 VIVADO 平台上的实验一。在使用 Verilog 语言重新编成实验一的内容后，我们一起进行了初步的仿真探索，但由于时间紧迫，并没有取得较为明显的成果，最后也没能尝试对应的硬件平台。如此，虽然没有探索出有用的成果，但探索的过程与我也有不小的价值。我通过这次的尝试，接触了硬件语言 Verilog 的语法与结构，同时也初步了解了 VIVODA平台的使用方法。

通过实验，我学到了实践比书本重要，要灵活应用不同方法解决问题。学会了如何处理短时间使能信号的问题，即关联一个变量清零和实时监测来确保活动完整进行。在实验二中，重新设计模块和参与显示器设计，加深了我对计数系统的理解。我对在 VIVADO 平台上重新编码实验一的尝试，使我了解了 Verilog 语言和 VIVADO 的使用。这些经验对我的学习和职业发展有利无弊。

心得总结

芦枷旭 2021211008

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。

5. 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。

6. 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。

7. 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。（药片装瓶）

心得总结

杨晨 2021212171

实验一、电子钟系统设计：

在这个实验中，我主要负责系统的整体设计、分钟和小时设计、时钟部分改写和优化、资源优化，包括分析时钟系统的功能需求，提出实现方案，确定系统的结构和工作原理。首先需要分析电子钟的基本功能，包括时间显示、时分秒计数、时间设置和闹钟等。然后根据这些功能提出系统的框图，确定使用哪些模块如计数器、分频器、译码器等来实现。再使用 VHDL 描述各个模块，并在 Quartus II 中进行综合、仿真验证每个模块的正确性。最后进行顶层设计，把各个模块集成，完成全系统的仿真和下载。

通过这个实验，我对 CPLD 的编程方法和 VHDL 的语法有了进一步的理解和运用，熟练掌握了 VHDL 中描述数字系统各种模块的方法，并能熟练使用 Quartus II 完成从设计输入、综合、仿真到下载的全过程。在描述较复杂模块和系统时，要注重 VHDL 代码的可读性，采用合理的变量命名和注释，编写规范和优雅的代码。我也需要进一步学习 Quartus II 中的高级功能，例如顶层框架设计和仿真分析等，利用其提供的功能提高调试效率。

在分析系统功能和提出实现方案时，我考虑到系统的扩展性和灵活性，要选择合适的数据类型作为计数器的输出，如 unsigned 类型可以表示正数，而 signed 类型既可以表示正数也可以表示负数，更适合倒计数的情况。计数器设计时要考虑各种边界条件，如在使能取消时计数器的值应该如何处理。这需要使用状态机来对整个计数过程进行有效控制，判断各种条件并作出正确响应。要考虑接口匹配的问题，保证计数器的输出可以正确地作为后续数字显示或译码模块的输入。这需要综合考虑接口参数如位数、8421BCD 编码方式、7 段显示数码管的译码方式等。我也密切关注资源利用率，采用更经济的模块来实现相同功能，缩短综合时间，减小芯片资源的浪费。

实验二、药片装瓶系统设计：

在这个实验中，我参与了计数器的方案设计，主要负责功能需求分析和实现方案的提出，并积极参与了后续的调试和优化。首先需要分析装瓶机的基本功能和输入输出，然后根据这些提出系统的框图，确定使用哪些模块如计数　器、比较器、译码器等来实现各功能模块。在 VHDL 中描述各个模块，并使用 Quartus II 进行仿真验证。然后进行顶层综合，整合各模块，完成全系统设计并下载至 CPLD 中运行。

在开发和调试的过程中，我们遇到了输入比较、同一输出多重赋值、显示器熄灭、计数同步等问题，但最终都给出了比较合理的解决方案。这也增强了我对硬件系统和硬件电路的敏感性。

通过这个实验，我对 VHDL 中描述数字系统的方法有了进一步的运用，能较为熟练地描述基本模块如计数器、译码器等，但是对更复杂的系统级设计还需要提高。特别是在模块间的接口连接和定时同步方面，代码规范、资源优化和总结改进等方面还需要加强

实验三、VGA 接口自主探索实验：

在这个实验中，我主要负责理解 VGA 接口的工作原理，确定 VGA 的时序参数，并在 VHDL 中描述 VGA 的输出接口。首先需要通过资料理解 VGA 接口的结构和显示原理，分析像素时钟、同步信号的产生方法和时序。然后在 VHDL 中编写 VGA 接口的代码，产生行复位、场复位和像素时钟，并根据输入的图像数据产生红绿蓝三个数据通道的输出。

在开发过程中,我采用版本控制和逐步修改的方法。我会将每次能正常工作的代码保存为一个版本,并添加版本号和日期,如 v0.1_20230515 等。这样在后续调试中出现问题时,我们可以回退到上一个工作版本,继续进行调试。在修改代码前,我会对当前工作版本进行备份,这样如果修改后的代码不能工作,我们可以直接还原到修改前的状态,避免大范围修改导致难以调试的情况出现。在修改和测试过程中，我是逐步进行修改和测试,而非一次性完全重写,这样的好处是，可以快速定位问题代码,便于调试。我们也记录每次修改的详细内容,如果新修改的代码不能工作,根据修改记录逐步还原,提高调试效率。

在 Quartus II 中进行仿真验证 VGA 接口模块的正确性，观察到行场同步信号和像素输出的时序。烧录代码到 CPLD 开发板后，通过对纯色显示的测试，确保时序正确。然后修改 VHDL 代码和颜色参数，输出一个简单的条纹图像以验证显示效果。这个过程中可能会出现各种问题，如同步信号的抖动、图像数据和实际显示的不一致等，需要通过仿真和实测的对比，逐步调试优化代码，最终实现预期的显示效果。

通过这个实验，我对 VHDL 的应用有了较大提高，特别是在描述时序和接口逻辑方面，熟练掌握了描述并产生复杂定时信号的方法。也对 VGA 接口有了较深入的了解，掌握了通过 VGA 显示图像的基本原理和方法。

综合这个三个实验，让我对 CPLD 的设计流程和方法有了较为系统和深入的学习和运用。从需求分析、方案设计到 VHDL 描述，再到 Quartus II 中的仿真调试，最终实现下载和调试，全面锻炼和提高了我的数字系统设计的能力。但相应的，更高级的部分，如顶层设计等高级设计技能，以及 VHDL 代码的规范和复杂度，还需要进一步学习和实践。这些实验为我后续的学习提供了很好的基础，也培养了解决实际问题的工程思维，收获颇丰。