**北京邮电大学课程设计报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **课程设计**  **名称** | **数字逻辑课程设计** | | **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** | |
| **2021211301** | **10** | **2021211298** | | **肖琪臻** |  | |
| **2021211301** | **17** | **2021213586** | | **郭栩源** |  | |
| **2021211304** | **3** | **2021210168** | | **张棋睿** |  | |
| **2021211304** | **33** | **2021213650** | | **姜杨皓** |  | |
| **课**  **程**  **设**  **计**  **内**  **容** | 《数字逻辑课程设计》以巩固《数字逻辑》理论课程中所学知识，培养使用可编程逻辑器件，运用在系统编程技术，利用TEC-8试验台上已有的电子器件，进行具有特定功能的电子产品的设计的能力为目标，要求设计一个电子钟系统和一个药片装瓶系统。  电子钟系统：实现24小时制时钟功能；整点报时功能；时间设置功能；随意切换设置和正常工作模式。附加功能：实现设置状态时闪烁显示；音乐整点报时；闹钟功能等等。  药片装瓶系统：实现药片装瓶功能；能够同时显示药瓶以及药片数量；有工作状态以及告警指示； 有清零状态、设置状态和工作状态，实现状态间正确切换设置。附加功能：实现设置状态时闪烁显示、工作状态时可以切换显示初始设置、可以设定每瓶药片数以及总药片数量限定等。 | | | | | |
| **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。

课程设计主要内容简介

**一、课程设计教学目的**

使用可编程逻辑器件，运用在系统编程技术，利用TEC-8试验台上已有的电子器件，进行具有特定功能的电子产品的设计，并上传到TEC-8试验台上进行验证和测试。

**二、基本内容**

1. 电子钟系统的设计

用硬件描述语言VHDL或（Verilog）和原理，和图混编方式完成设计；实现24小时制时钟功能；实现整点报时功能；实现时间设置功能，并且可以随意切换设置和正常工作模式。 附加功能：实现设置状态时闪烁显示、实现音乐整点报时、闹钟功能等等。

1. 药瓶计数器的设计

基本功能：实现药片装瓶功能；能够同时显示 药瓶以及药片数量；有工作状态以及告警指示； 有清零状态、设置状态和工作状态，实现状态间 正确切换设置。 附加功能：实现设置状态时闪烁显示、工作状态 时可以切换显示初始设置、可以设定每瓶药片数 以及总药片数量限定等。

课题硬件环境：Tec-8，MAX7000S

**三、实验方法**

1. 需求分析

根据老师的要求和自己对产品的理解，列出对这个产品的期望设计和期望工作模式和状态。

1. 顶层设计

根据总体的期望，将相关的功能独立地划分到不同的模块当中。对模块的工作状态进行要求，对模块间的连线和交流进行整合。

1. 模块设计

对具体模块的逻辑进行编程。

1. 分模块测试和整合

对每个模块单独进行测试，保证其按期望工作。

1. 总体测试和整合

将各个模块整合到一起，进行测试。我们还需要将设计上传到TEC-8试验台上，进行接口分配和测试蜂鸣器等外部设备的工作情况。

1. 设计迭代与节约硬件资源

思考新的需求，并重复2-6步。要考虑到芯片有限的资源和外部设备的局限性。

1. 设计固定

保存源码，对其进行高强度的测试，保证在极端和边界工况下的鲁棒性。

**四、团队分工**

肖琪臻：给出电子钟和药瓶计数器顶层设计，熟悉和使用试验台和Quartus软件，为团队的工作做准备。在项目中协助实现药瓶计数器和电子钟的部分功能，参与调试工作。监督项目进度。

郭栩源：负责药瓶计数器的开发工作，给出基本功能的框架和开发思路，负责实现药瓶计数器的附加功能。测试药瓶计数器。编写药瓶计数器报告和调试日志。

姜杨皓：参与完成电子钟报时功能和闹钟功能。参与电子钟测试修改工作。负责电子钟的开发文档的维护和报告的编写。

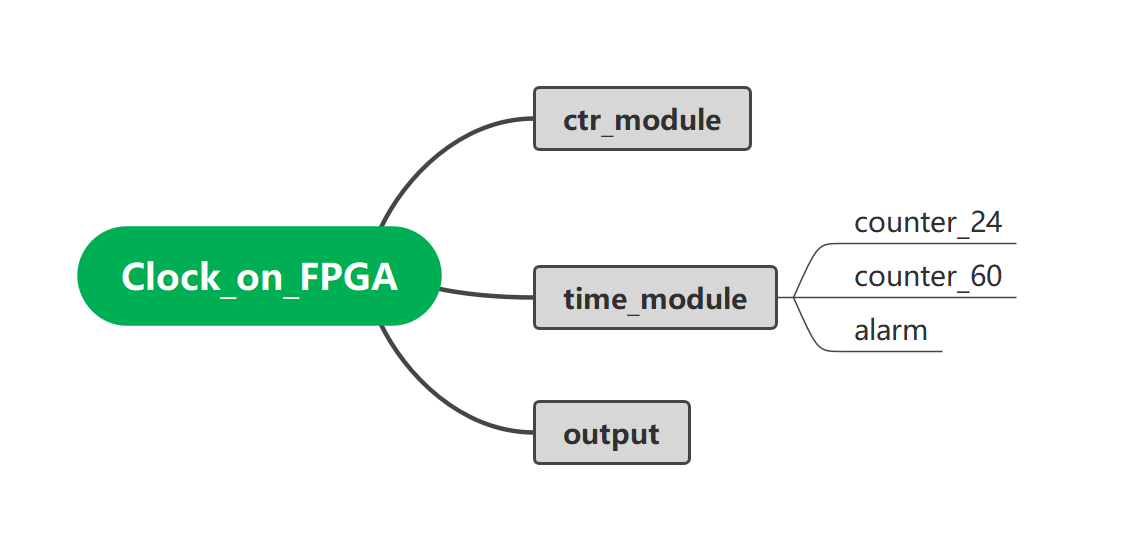
张棋睿：负责电子钟的开发工作，完成基本功能，主导完成附加功能，实现蜂鸣器音乐等功能，协调各功能模块。参与测试修改电子钟功能。完成电子钟调试日志等的编写。

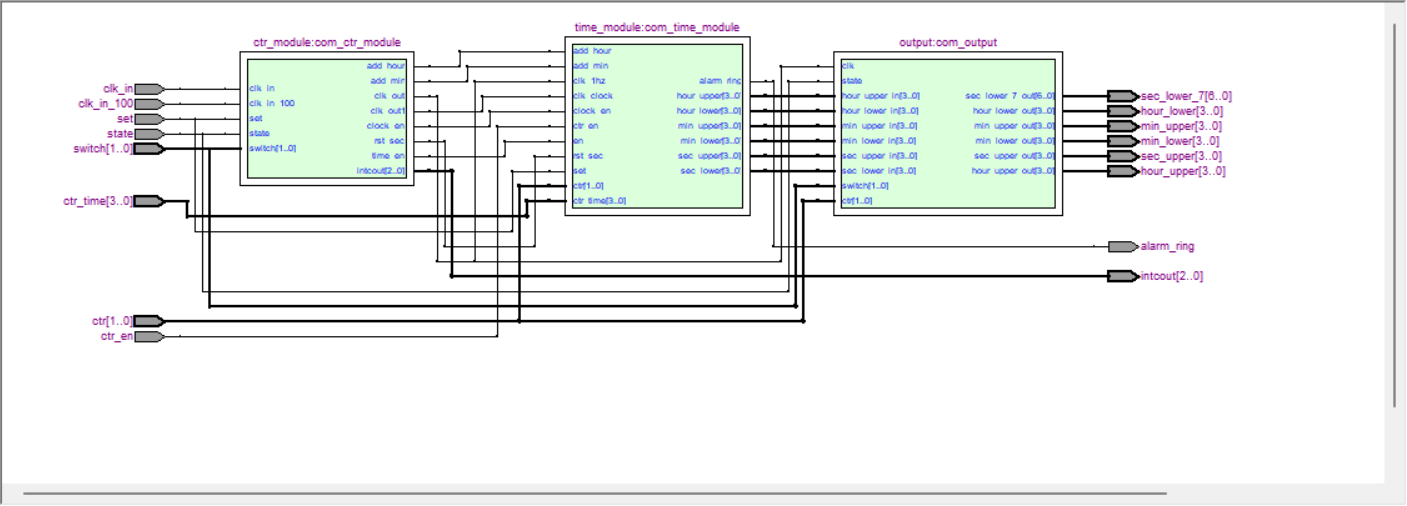
**实验一 电子钟系统**

**一、题目分析**

题目分析：

用VHDL语言，在TEC-8模型机上设计一个电子钟，实现24小时制时钟功能，整点报时功能，时间设置功能，设置状态闪烁功能，音乐整点报时功能，闹钟功能。





模块划分：

（1）Clock\_on\_FPGA顶层模块，完成顶层设计，调用底层模块。

（2）ctr\_module控制模块，负责控制时钟所处的状态是正常运行还是设置时间状态，亦或是设置闹钟状态。同时实现分频，包括时钟所需的1hz频率和响铃所需的不定赫兹频率，以及音乐的频率控制。

（3）output输出模块，负责发光二极管的的显示控制，实现正常运行时输出和设置时闪烁输出

（4）time\_module时间模块，调用底层模块实现时钟计时，整点报时和闹钟的响铃，以及闹钟的设置

（5）alarm响铃模块，在整点报时和闹钟时输出扬声器所接受的方波

（6）counter\_24 24小时计数模块，实现一个模24计数器

（7）counter\_60 60分和秒的计数模块，实现一个模60计数器

**二、设计详解**

（1）顶层设计：Clock\_on\_FPGA顶层模块

输入与输出：

顶层模块的输入主要接受时钟信号（1Hz-100kHz不等）和电平信号（tec-8为k0-k15）；输出主要是把时钟输出到6个数码管上，并输出音频到扬声器。具体可根据实现的功能灵活删减。最终的输入输出如下：

ENTITY Clock\_on\_FPGA IS

port(

clk\_in:in STD\_LOGIC;--输入的10000hz脉冲，用于音乐分频

clk\_in\_100:in STD\_LOGIC;--输入的100hz脉冲，用于时钟分频

set:in STD\_LOGIC;--没有用到的开关K0

switch:in STD\_LOGIC\_VECTOR(1 DOWNTO 0);--K3K2 电子钟的设置状态，00为清零模式，01为设置分模式，10为设置时模式，11为设置闹钟模式

state:in STD\_LOGIC;--K1电子钟的状态，0位正常行走，1为设置模式

ctr:in STD\_LOGIC\_vector(1 DOWNTO 0);--K5K4 控制设置闹钟的位，00为分低位，01为分高位，11为时低位，10为时高位

ctr\_time:in STD\_LOGIC\_vector(3 DOWNTO 0);--K9K8K7K6 设置闹钟的具体数值，为8421码

ctr\_en:in STD\_LOGIC;--K10 闹钟和整点报时有效开关，0时响铃有效，1时禁用响铃

sec\_lower\_7:OUT STD\_LOGIC\_VECTOR(6 downto 0);--秒的低位

hour\_lower:OUT STD\_LOGIC\_VECTOR(3 downto 0);--时的低位

min\_upper:OUT STD\_LOGIC\_VECTOR(3 downto 0);--分的高位

min\_lower:OUT STD\_LOGIC\_VECTOR(3 downto 0);--分的低位

sec\_upper:OUT STD\_LOGIC\_VECTOR(3 downto 0);--秒的高位

hour\_upper:OUT STD\_LOGIC\_VECTOR(3 downto 0);--时的高位

intcout:out INTEGER RANGE 1 TO 6;--调试用

alarm\_ring:out STD\_LOGIC--给扬声器的响铃脉冲

);

END Clock\_on\_FPGA;

顶层模块的设计：

顶层设计时，主要考虑大体上的功能划分。我们采用设置与时间推进分开的总思想，大体上分为完成一个状态控制模块和一个24小时制的时钟模块。状态模块控制主要控制当前时钟的状态，是正常运行还是设置时间状态，亦或是设置闹钟状态。时钟模块实现两个60进制计数器和一个24进制计数器来完成时间的推进和其它设置操作。时钟模块接受控制模块的控制信号，来决定自己当前所要进行的操作。

具体模块设计详见后面。

除了控制模块和时钟模块，我们还在顶层添加了输出模块，输出模块接受时钟模块的当前时间信息，输出到6个数码管上。

具体模块设计详见后面。

（2）控制模块ctr\_module

控制模块ctr\_module包含模式控制和时钟分频，音乐分频，以及音乐控制。

①模式控制：

ctr直接接收顶层的输入对整个电子钟进行模块控制。

输入信号1位state，2位switch

state=0 时钟正常走

state=1 时钟暂停，进入设置模式，switch有效，

switch=00 清零模式

switch=01 设置分钟模式

switch=10 设置小时模式

switch=11 设置闹钟模式

——设置分信号与设置时信号采用的是脉冲信号T3，而不是电平信号，原因在后面问题与解决方案详细介绍

if(state='1') then--state为1，设置模式

en<='1';——时钟正常走的使能信号

CASE(switch) IS

WHEN "00"=>-- switch为00，设置清零模式

sec<=NOT set;——清零信号

min<='0';

hour<='0';

sig\_clock\_en<='0';

WHEN "01" =>-- switch为01，设置分模式

sec<='1';

min<=set;——设置分信号

hour<='0';

sig\_clock\_en<='0';

WHEN "10" =>-- switch为10，设置时模式

sec<='1';

min<='0';

hour<=set;——设置时信号

sig\_clock\_en<='0';

WHEN OTHERS =>-- switch为11，设置闹钟模式

sec<='1';

min<='0';

hour<='0';

sig\_clock\_en<='1';——设置闹钟信号

END CASE;

else --state为0，正常行走模式

en<='0';

sig\_clock\_en<='0';

sec<='1';

min<='0';

hour<='0';

end if;

end process;

②时钟分频

对100hz脉冲进行100分频，用于时钟：

process(clk\_in\_100)--进行计数，每当满100归零

BEGIN

IF(clk\_in\_100'event and clk\_in\_100='1')THEN

IF temp= 99 THEN--进行计数，每当满100归零

temp <= 0;

else

temp <= temp+1;

END If;

END IF;

END PROCESS;

process(clk\_in\_100,temp--进行计数，每当满50进行电平反转

BEGIN

IF(clk\_in\_100'event and clk\_in\_100='1')THEN

if( temp<50) then--进行计数，0~49内电平为0

clk\_out <= '0' ;

else--进行计数，50~99内电平为1

clk\_out <= '1';

end if;

END IF;

END process;

③音乐分频

-- 音乐分频，进行n分频，具体n由变量“ping”控制

process(clk\_in) --进行计数，每当满“ping”归零

BEGIN

IF(clk\_in'event and clk\_in='1')THEN

IF temp1 >= ping THEN--进行计数，每当满“ping”归零

temp1 <= 0;

else

temp1 <= temp1+1;

END IF;

END IF;

END PROCESS;

process(clk\_in,ping,stop) --进行计数，每当满“ping”归零

BEGIN

IF(clk\_in'event and clk\_in='1')THEN

if(stop='1')THE--如果被禁用，则始终高阻态

clk\_out1 <= 'Z';

elsif( temp1<ping\_2) then--进行计数，0~ ping/2-1内电平为0

clk\_out1 <= '0' ;

else--进行计数，ping/2~ping/2-1内电平为0

clk\_out1 <= '1';

end if;

END IF;

END process;

④音乐控制

控制一个音响多久，停多久，

在temp\_3hz为0~49时正常响铃，temp\_3hz为50~69时禁用响铃，temp\_3hz=70时归零并恢复正常响铃，如此0.7秒一个循环

int进行计数，每0.7秒加一，满6循环，每0.7秒控制一种频率

process(clk\_in\_100)

BEGIN

IF(clk\_in\_100'event and clk\_in\_100='1')THEN

IF temp\_3hz = 50 THEN-- temp\_3hz满50，开始禁用响铃

stop<='1';-- 禁用响铃

temp\_3hz<= temp\_3hz+1;

if(int=6)the--对“int”计数，满6后归1

int<=1;

else --对“int”计数，没有满6，正常加1

int<=int+1;

end if;

elsif temp\_3hz =70 then-- temp\_3hz满70，开始启动响铃，并将其归零

stop<='0';--启用响铃

temp\_3hz<= 0;-- temp\_3hz归零

else

temp\_3hz<= temp\_3hz+1;-- temp\_3hz正常计数加一

END If;

END IF;

END PROCESS;

------------------------------------------

-- 改变分频频率

process(int)--int进行计数，每0.7秒加一，满6循环，每0.7秒一种频率

begin

case(int)is

when 1=> ping<= 38 ;--int为1时，ping=38

when 2=> ping<= 30 ; --int为2时，ping=30

when 3=> ping<= 34 ; --int为3时，ping=34

when 4=> ping<= 25 ;-- int为4时，ping=25

when 5=> ping<= 29 ; --int为5时，ping=39

when 6=> ping<= 25 ; --int为6时，ping=25

end case;

end process;

ping\_2<= ping/2;

（3）output模块

负责控制数码管的显示以及设置状态时的闪烁。

ARCHITECTURE arch OF output IS

SIGNAL var : STD\_LOGIC\_VECTOR(6 downto 0);--控制秒低位数码管的信号

BEGIN

——七段显示译码

PROCESS(sec\_lower\_in)—控制秒低位的显示

BEGIN

CASE sec\_lower\_in IS

WHEN "0000" =>--秒低位为0，控制数码管显示0

var <= "1111110";

WHEN "0001" => --秒低位为1，控制数码管显示1

var <= "0110000";

WHEN "0010" => --秒低位为2，控制数码管显示2

var <= "1101101";

WHEN "0011" => --秒低位为3，控制数码管显示3

var <= "1111001";

WHEN "0100" => --秒低位为4，控制数码管显示4

var <= "0110011";

WHEN "0101" =>--秒低位为5，控制数码管显示5

var <= "1011011";

WHEN "0110" => --秒低位为6，控制数码管显示6

var <= "1011111";

WHEN "0111" =>--秒低位为7，控制数码管显示7

var <= "1110000";

WHEN "1000" => --秒低位为8，控制数码管显示8

var <= "1111111";

WHEN "1001" =>--秒低位为9，控制数码管显示9

var <= "1111011";

WHEN OTHERS =>--秒低位为10，控制数码管显示异常符号

var <= "0001000";

END CASE;

END PROCESS;

sec\_lower\_7\_out<=var;--将秒低位显示赋值给输出

sec\_upper\_out<=sec\_upper\_in;--秒高位直接显示

——无闪烁实现

--hour\_lower\_out<=hour\_lower\_in; --时低位直接显示

--hour\_upper\_out<=hour\_upper\_in; --时高位直接显示

--min\_lower\_out<=min\_lower\_in; --分低位直接显示

--min\_upper\_out<=min\_upper\_in; --分高位直接显示

——闪烁实现，思路是根据1Hz的时钟信号，高电平显示，低电平高阻，根据不同的模式实现不同位的闪烁

process(state,switch,ctr)

begin

if(state='0')then—正常行走状态

min\_upper\_out<=min\_upper\_in;

min\_lower\_out<=min\_lower\_in;

hour\_lower\_out<=hour\_lower\_in;

hour\_upper\_out<=hour\_upper\_in;

elsif(state='1')then—设置状态

case(switch)is

when"00"=>

hour\_lower\_out<=hour\_lower\_in;

hour\_upper\_out<=hour\_upper\_in;

min\_lower\_out<=min\_lower\_in;

min\_upper\_out<=min\_upper\_in;

when"01"=>--设置分状态

hour\_lower\_out<=hour\_lower\_in;

hour\_upper\_out<=hour\_upper\_in;

case(clk)is

when '1'=>--时钟脉冲为1时，分为高阻态，不显示

min\_lower\_out<="ZZZZ";

min\_upper\_out<="ZZZZ";

when others=>-时钟脉冲为0时，分正常显示

min\_lower\_out<=min\_lower\_in;

min\_upper\_out<=min\_upper\_in;

end case;

when"10"=>--设置时状态

min\_lower\_out<=min\_lower\_in;

min\_upper\_out<=min\_upper\_in;

case(clk)is

when '1'=>--时钟脉冲为1时，时为高阻态，不显示

hour\_lower\_out<="ZZZZ";

hour\_upper\_out<="ZZZZ";

when others=>--时钟脉冲为0时，时正常显示

hour\_lower\_out<=hour\_lower\_in;

hour\_upper\_out<=hour\_upper\_in;

end case;

when others=>--设置闹钟状态

case(clk)is

when '1'=>----时钟脉冲为1时，时和分为高阻态，不显示

hour\_lower\_out<="ZZZZ";

hour\_upper\_out<="ZZZZ";

min\_lower\_out<="ZZZZ";

min\_upper\_out<="ZZZZ";

when others=>----时钟脉冲为0时，时和分正常显示

min\_lower\_out<=min\_lower\_in;

min\_upper\_out<=min\_upper\_in;

hour\_lower\_out<=hour\_lower\_in;

hour\_upper\_out<=hour\_upper\_in;

end case;

end case;

end if;

end process;

END arch;

（4）时钟模块time\_module

调用底层模块实现时钟计时，整点报时和闹钟的响铃，以及闹钟的设置

具体时钟计时详见模24计数模块和模60计数模块，响铃详见alarm模块，闹钟设置如下：

process(set,clk\_1hz) --set clock

begin

if(clock\_en='1')then——设置临时的闹钟信号

case(ctr)is

when"00"=>--闹钟设置位为00，设置分低位

sig\_tmp\_min1<=ctr\_time;

when"01"=>--闹钟设置位为01，设置分高位

sig\_tmp\_min2<=ctr\_time;

when"11"=>--闹钟设置位为11，设置时低位

sig\_tmp\_hour1<=ctr\_time;

when others=>--闹钟设置位为10，设置时高位

sig\_tmp\_hour2<=ctr\_time;

end case;

if(set'event and set='1')then —当脉冲信号到来时，临时的闹钟信号赋值给实际闹钟信号，实现同步模式

case(ctr)is

when"00"=>--闹钟设置位为00，设置分低位

sig\_clock\_min1<=sig\_tmp\_min1;

when"01"=>--闹钟设置位为01，设置分高位

sig\_clock\_min2<=sig\_tmp\_min2;

when"11"=>--闹钟设置位为11，设置时低位

sig\_clock\_hour1<=sig\_tmp\_hour1;

when others=>--闹钟设置位为10，设置时高位

sig\_clock\_hour2<=sig\_tmp\_hour2;

end case;

end if;

end if;

end process;

（5）模24计数模块

进行小时的模24计数

architecture arch of counter\_24 is

signal cnt\_l: std\_logic\_vector(3 downto 0);--记录低位的信号

signal cnt\_h: std\_logic\_vector(3 downto 0); --记录高位的信号

Begin

process(clk, rst)

begin

if rst = '0' then——清零信号

cnt\_l <= (others => '0');

cnt\_h <= (others => '0');

elsif rising\_edge(clk) then

if en = '0' then

if (cnt\_l = "0011"AND cnt\_h = "0010") the--满24后归零，并产生进位信号

cnt\_l <= "0000";

cnt\_h <= "0000";

co <= '1';

else if(cnt\_l="1001")THE--低位满10进位

cnt\_l <= "0000";

cnt\_h <= cnt\_h + 1;

else—低位正常加一

cnt\_l <= cnt\_l + 1;

co <= '0';

end if;

end if;

end if;

END IF;

end process;

cnt\_0 <= cnt\_l;

cnt\_1 <= cnt\_h;

end arch;

（6）模60计数模块

进行分和秒的模60计数

architecture arc of counter\_60 is

signal cnt\_l: std\_logic\_vector(3 downto 0); --记录低位的信号

signal cnt\_h: std\_logic\_vector(3 downto 0); --记录高位的信号

begin

process(clk, rst)

begin

if rst = '0' then——清零信号

cnt\_l <= (others => '0');

cnt\_h <= (others => '0');

--co <= '0';

elsif rising\_edge(clk) then

if en = '0' then

if (cnt\_l = "1001"AND cnt\_h = "0101") then—满60归零，并产生进位信号

cnt\_l <= "0000";

cnt\_h <= "0000";

co <= '1';--产生进位信号

else if(cnt\_l="1001")THEN—低位满10进位

cnt\_l <= "0000";

cnt\_h <= cnt\_h + 1;

else--正常计数，低位加一

cnt\_l <= cnt\_l + 1;

co <= '0';

end if;

end if;

end if;

end if;

end process;

cnt\_0 <= cnt\_l;

cnt\_1 <= cnt\_h;

end arc;

（7）alarm模块

控制扬声器的响铃

architecture func of alarm is

begin

process(co\_hour,clock\_now)

begin

if(ctr\_en='0')then--K10为0，响铃有效

if(clock\_now='1'or co\_hour='1')then--开始响铃状态

alarm\_ring<=clk\_clock;--将响铃脉冲赋值给扬声器

else --K10为1，响铃禁用

alarm\_ring<='0'; --将扬声器接收脉冲始终为0

end if;

end if;

end process;

end func ;

**三、调试过程中所遇困难及解决方案**

1.vhdl语言问题

（1）不清楚时钟信号的if调用问题。在设计初期时，同时采用时钟的上升沿与下降沿进行if判断，导致编译报错，最后通过查阅资料和老师答疑解决。同样的问题还有时钟沿判断不能使用else，错误与上面相同。

（2）资源不足的问题。到了设计后期，因为要实现音乐分频和闪烁显示，出现了资源不足的问题。为了解决这个问题，我们采取了if语句的case化，把单个信号的多个判断转为case；同时减少了很大的integer类型的使用等

（3）音乐的频率问题。这个问题与前面的资源不足问题是共存的，我们打算实现七个基本的音乐频率，但是资源已近不允许我们分这么多的频了，所以我们再使用（2）优化资源的同时，想到了分频器复用这一个方法。我们控制一个分频器的计数上线，用一个分频器实现了多分频，而计数器的上限，则由100Hz的频率顺带控制。

（4）在最初的设计中，设置闹钟的时或分两位的模式时，控制模块在T3上升沿向时间模块传递了一个由0变成1的设置信号。时间模块识别这个设置信号的上升沿，完成时或分加1的操作。然而这个设置信号无法从1变回0，时间模块无法再次识别这个信号的上升沿，时或分的设置无法继续进行。

我们尝试了很多解决方案，例如在一个T3上升沿，设置信号由0变1，在下一个T3上升沿，设置信号由1变0，这样虽然能完成功能，但是不符合实际情况，因为这样得按两下 QD（两个T3）才能实现位加1。还有一个解决方案就是，不采用设置信号的上升沿，而是把设置信号看成是电平信号，在设置信号为1时实现位加1，但是这样不好控制设置的时间，因为时或分会加得非常快，难以精确到准确要设置得时间。经过思考与讨论，我们得出了一个完美的解决方案，直接把T3信号接入设置信号，这样设置信号随着T3的变动，既能在一个QD中完成由0变1的上升沿传递和由1变0的信号复位。

（5）闹钟设置问题（同步、异步问题）。开始设置闹钟时，采用了异步的方式，只要设置信号一到来，立刻读入当前输入数据设置闹钟，这意味着需要先设定好闹钟数据，再进入设置闹钟模式；需要先离开设置闹钟模式，在撤销设定好闹钟数据。再这与正常的操作习惯是相违背的。

所以我们更换了同步模式，当设置信号来的时候，系统读入数据但不设置，而是等待一个T3脉冲信号才把读入的数据设置成闹钟时间。这样就解决了上述问题。事实上，不只是闹钟设置，本次电子钟设计的所有设置模式都采用了同步的方式，保证了电子钟的体验符合实际。

（6）音乐杂音颤音的问题。分频后扬声器输出的音乐有颤音杂音现象，一部分是因为硬件的问题，但主要还是分频的问题。具体是因为分频后信号的上沿与下沿的长短不一，而长短不一的原因是因为分频器的上限采用了偶数（如0-40,有奇数个整数，导致上下沿时间分配不一致）。当所有分频器的上限采用了奇数后，问题基本解决。

2. 硬件问题

（1）时钟紊乱问题。一开始我们采用10000Hz的时钟信号，采用10000分频分为1Hz，后期再采用10000Hz进行音乐的分频，分频后，时钟信号的1Hz出现紊乱，表现为时钟跳的时快时慢，且整体偏向2Hz。我们猜测这可能是内部硬件实现的问题，在复用10000Hz后，信号出现了干扰。为解决这个问题，我们采用100Hz作为时钟信号，采用100分频分为1Hz，同时保留10000Hz用作音乐分频。实行后，问题得到了有效解决。

（2）要使用T3信号，相应的vhdl信号的管脚应该直接使用T3,而不是使用QD管脚。QD控制T3信号脉冲的产生，但是自己不能作为管脚。

**四、团队分工**

肖琪臻同学完成了基础功能的顶层设计和时间模块的部分。

张棋睿同学完成了基础功能的控制模块，输出模块，完善了时间模块。完成了附加功能的闪烁、闹钟设置等。

张棋睿同学和姜杨皓同学共同完成了整点报时和闹钟音乐功能的实现，以及整个电子钟的调试和debug。

**五、用户手册**

我们使用了开关K1，K2，K3，K4，K5，K6，K7，K8，K9，K10共10个开关。

K1负责控制电子钟的状态，当K1为0时电子钟正常行走，当K1为1时进入设置状态，时钟不再走动，并进行闪烁。

当K1为1时，K2，K3负责控制设置的内容。

K3K2为00时，两个秒位进行闪烁，为控制清零状态，按下QD后时钟会被清零；

K3K2为01时，两个分位进行闪烁，为设置分钟状态，按下QD后分钟会加一；

K3K2为10时，两个时位进行闪烁，为设置小时状态，按下QD后小时会加一；

K3K2为11时，显示当前设置的闹钟，并且所有位一起闪烁，为设置闹钟状态。

当K3K2为11时，K4，K5负责控制设置闹钟的位数，K6，K7，K8，K9用8421码控制设置的值。

当K5K4为00时，设置闹钟分低位，具体值由K9K8K7K6输入，按下QD该值会被写入闹钟的分低位。

当K5K4为01时，设置闹钟分高位，具体值由K9K8K7K6输入，按下QD该值会被写入闹钟的分高位。

当K5K4为11时，设置闹钟时低位，具体值由K9K8K7K6输入，按下QD该值会被写入闹钟的时低位。

当K5K4为10时，设置闹钟时高位，具体值由K9K8K7K6输入，按下QD该值会被写入闹钟的时高位。

K10负责控制整点报时和闹钟的禁用，当K10为0时，整点报时和闹钟正常响铃；当K为1时，整点报时和闹钟不再响铃。

**实验二 药片装瓶系统**

**一、题目分析**

该课题要求使用TEC-8实验系统模拟一个流水线药片装瓶系统。

首先分析现实中的药片装瓶系统的功能：

1. 预设模块：根据需求设置本次装瓶工作参数：“需装总瓶数”和“每瓶需装药片数”，系统中设置“可以设定的每瓶药片数，以及总药片数量的限定”，若输入非法数据，则出现告警显示。
2. 装瓶模块：传送带将药片送至瓶口，且每次只允许一片药片落入瓶中。在药瓶瓶口有一个光传感器，每片药片落入瓶中时产生一个脉冲信号。装瓶过程可以通过按钮随时暂停。
3. 计数模块：收到装瓶模块的脉冲信号后，计数模块中的变量“当前药瓶中药片数”和“总药片数”都加一。然后将“当前药瓶中药片数”与“每瓶需装药片数”进行比较，若相等，则该瓶已装满，此时“已装瓶数”加一，“当前药瓶中药片数”清空。同时，可以通过按钮随时对计数模块中的变量进行异步的置零。
4. 显示模块：具有切换显示功能。在设置时显示设置的工作参数：为便于操作者使用，设置时每个参数的个位与十位分开设置，且正在设置的数位不断闪烁已提醒操作者。在装瓶时有两种显示模式可以通过按钮进行切换：显示“当前药瓶中药片数”+“已装瓶数”或“总药片数”。在装瓶过程暂停的情况下可以查看初始设置。
5. 告警模块：告警模块负责在设置时输入非法数据时发出告警，以及药片装瓶完毕后给出完成提醒。

然后设计如何用TEC-8实验台实验上述功能：

1. 预设模块：使用逐位设置模式，通过8421码分别设置“需装总瓶数”和“每瓶需装药片数”的个位和十位。通过一个4位标准逻辑矢量来输入8421码，通过一个2位的标准逻辑矢量来选定设置哪一位。其中，“需装总瓶数”和“每瓶需装药片数”的个位和十位范围都位0～9，预设最多装药片数为99\*99=9801。
2. 装瓶模块：由于现实中没有药片和光传感器，故简化为用TEC-8实验台提供的10Hz时钟信号来模拟药片装瓶信号。同时将“暂停”与“预设”简化为一个模式，暂停时自动切换为预设模式（显示预设内容）。
3. 计数模块：使用已在电子钟中编写的计数器元件（Component），只需在端口表中提供合适的变量即可。同时用一个标准逻辑型控制异步清零。
4. 显示模块：用一个标准逻辑型来控制显示模式。在“当前药瓶中药片数”+“已装瓶数”模式下，两者显示范围都位00～99；在“总药片数”模式下，显示范围是0000～9999（实际上不超过9801）。
5. 告警模块：正常状态下绿灯亮。在预设模块中输入非8421码0～9的内容时，触发告警，红灯亮，同时蜂鸣器不断发出声音，直到再次输入正确数值。在完成装瓶任务（所有瓶都已装满时），触发提示，红灯亮，蜂鸣器响三声。

**二、设计详解**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity fillpills is

port(

input : in std\_logic\_vector(3 downto 0); --4位标准逻辑矢量来输入8421码

input\_control : in std\_logic\_vector(1 downto 0); --2位的标准逻辑矢量选定设置哪一位

mode : in std\_logic; --'0' 表示设置模式, '1'表示计数模式

working : in STD\_LOGIC; -- ‘0’暂停,‘1’ 工作

display\_mode : in std\_logic; --'0' 显示总片数, '1' 显示当前药片数和瓶数

clr : in std\_logic; --异步置0信号

qd : in std\_logic; --qd脉冲

clk,clk\_buzzer : in std\_logic; --10Hz时钟信号用于计数，100Hz时钟信号用于蜂鸣器

lg2,lg3,lg4,lg5,lg6 : out std\_logic\_vector(3 downto 0); --对应七段显示译码管lg2～lg6

red\_light,green\_light : out std\_logic; --红绿灯，红灯表示装载完毕或设置时出现问题

buzzer : out std\_logic --蜂鸣器，用于告警以及装载完毕提醒

);

end fillpills;

architecture arc of fillpills is

signal start : STD\_LOGIC; --开始信号

signal temp,buzzer\_count: integer range 0 to 4; --temp用于分频，buzzer\_count用于控制蜂鸣器

signal clk\_1hz : std\_logic; --使用分频，从输入的10Hz信号得到1Hz信号

signal finish : std\_logic := '0'; --'1' 表示装载完毕，’0’表示其它情况

signal error : std\_logic := '0'; --‘1’表示存在错误，'0' 不存在错误

signal npo,npt : std\_logic\_vector(3 downto 0); --预设的每瓶装药片数

signal nbo,nbt : std\_logic\_vector(3 downto 0); --预设的装药片瓶数

signal tpo,tpt,tph,tpth : std\_logic\_vector(3 downto 0); --当前总药片片数

signal cbo,cbt : std\_logic\_vector(3 downto 0); --当前装瓶数

signal cpo,cpt : std\_logic\_vector(3 downto 0); --当前瓶中药片片数

signal cocpt, cocbo,cocbt,cotpo,cotpt,cotph,cotpth,cpo\_rst,cpt\_rst,b\_rst,clk\_in:STD\_LOGIC;

--使用的在电子钟中已编写的计数器元件

COMPONENT counter\_10

port(

clk: in std\_logic;

rst: in std\_logic;

en: in std\_logic;

co:out STD\_LOGIC;

cnt:out std\_logic\_vector(3 downto 0)

);

END COMPONENT;

begin

--计数模块

--当前瓶中药片数的计数器

cnt\_cpo:counter\_10 PORT MAP(clk=>clk\_in,rst=>cpo\_rst,en=>start,co=>cocpt,cnt=>cpo);

cnt\_cpt:counter\_10 PORT MAP(clk=>cocpt,rst=>cpt\_rst,en=>start,cnt=>cpt);

--当前瓶数的计数器

cnt\_bo:counter\_10 PORT MAP(clk=>cocbo,rst=>b\_rst,en=>start,co=>cocbt,cnt=>cbo);

cnt\_bt:counter\_10 PORT MAP(clk=>cocbt,rst=>b\_rst,en=>start,cnt=>cbt);

--总药片数的计数器

cnt\_tpo:counter\_10 PORT MAP(clk=>clk\_in,rst=>clr,en=>start,co=>cotpt,cnt=>tpo);

cnt\_tpt:counter\_10 PORT MAP(clk=>cotpt,rst=>clr,en=>start,co=>cotph,cnt=>tpt);

cnt\_tph:counter\_10 PORT MAP(clk=>cotph,rst=>clr,en=>start,co=>cotpth,cnt=>tph);

cnt\_tpth:counter\_10 PORT MAP(clk=>cotpth,rst=>clr,en=>start,cnt=>tpth);

--分频，从输入的10Hz时钟信号，得到1Hz时钟信号

process(clk)

begin

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

if(temp=4) then

temp<=0;

clk\_1hz<=not clk\_1hz;

else

temp<=temp+1;

end if;

end if;

end process;

--设置模块

process(qd)

begin

if(qd'event and qd='1'and mode = '0')then

if(input>"1001") then --在预设模块中输入非8421码0～9的内容时，触发告警

error<='1';

else

error<='0';

case input\_control is

when "00" => npo<=input; --设置“每瓶需装药片数”的个位

when "01" => npt<=input; --设置“每瓶需装药片数”的十位

when "10" => nbo<=input; --设置“需装总瓶数”的个位

when "11" => nbt<=input; --设置“需装总瓶数”的十位

end case;

end if;

end if;

end process;

--控制计数，使计数器在到达设定片数时可以清零并增加一个新的瓶子

process(clk)

begin

if(mode = '1' AND finish = '0')then

clk\_in <= clk;

end if;

if((cpo = npo) AND (cpt = npt))then

cpo\_rst<='0' and clr;

cpt\_rst<='0' and clr;

cocbo <= '1';

else

cpo\_rst<='1' and clr ;

cpt\_rst<='1' and clr;

cocbo <= '0';

end if;

if((nbo = cbo )and (nbt = cbt))then --所有瓶都已装满，finish置一，并停止计数

b\_rst<= clr;

b\_rst<= clr;

finish <= '1';

else

b\_rst<=clr;

b\_rst<=clr;

finish <= '0';

end if;

if(clr = '0')then

finish <= '0';

end if;

end process;

--显示模块

process(mode,display\_mode,start,clk)

begin

if(mode='1') then --在计数模式下

if(display\_mode='0') then --显示“当前药瓶中药片数”+“已装瓶数”

lg6<=cpt;

lg5<=cpo;

lg4<=cbt;

lg3<=cbo;

lg2<="ZZZZ";

else --显示“总药片数”

lg6<=tpth;

lg5<=tph;

lg4<=tpt;

lg3<=tpo;

lg2<="ZZZZ";

end if;

else --在预设模式下，设置哪一位，哪一位就进行闪烁

if(input\_control = "00")Then

if(clk\_1hz = '1')Then

lg5<=npo;

ELSE

lg5<="ZZZZ";

END IF;

lg6<=npt;

lg4<=nbt;

lg3<=nbo;

lg2<="ZZZZ";

elsif(input\_control = "01")Then

if(clk\_1hz = '1')Then

lg6<=npt;

ELSE

lg6<="ZZZZ";

END IF;

lg5<=npo;

lg4<=nbt;

lg3<=nbo;

lg2<="ZZZZ";

elsif(input\_control = "10")Then

if(clk\_1hz = '1')Then

lg3<=nbo;

ELSE

lg3<="ZZZZ";

END IF;

lg6<=npt;

lg4<=nbt;

lg5<=npo;

lg2<="ZZZZ";

else

if(clk\_1hz = '1')Then

lg4<=nbt;

ELSE

lg4<="ZZZZ";

END IF;

lg6<=npt;

lg5<=npo;

lg3<=nbo;

lg2<="ZZZZ";

END IF;

end if;

end process;

--蜂鸣器的控制

process(clk\_1hz)

begin

if(clk\_1hz'event and clk\_1hz='1') then --控制蜂鸣器只响3下

if(finish='1') then

if(buzzer\_count<=3) then

buzzer\_count<=buzzer\_count+1;

end if;

else

buzzer\_count<=0;

end if;

end if;

if(error='1') then --如果出现错误，则蜂鸣器响直到错误消除

buzzer<=clk\_buzzer and clk\_1hz;

elsif(finish='1' and buzzer\_count<=3) then --如果是装载完成，则蜂鸣器只响3下

buzzer<=clk\_buzzer and clk\_1hz;

else

buzzer<='0';

end if;

end process;

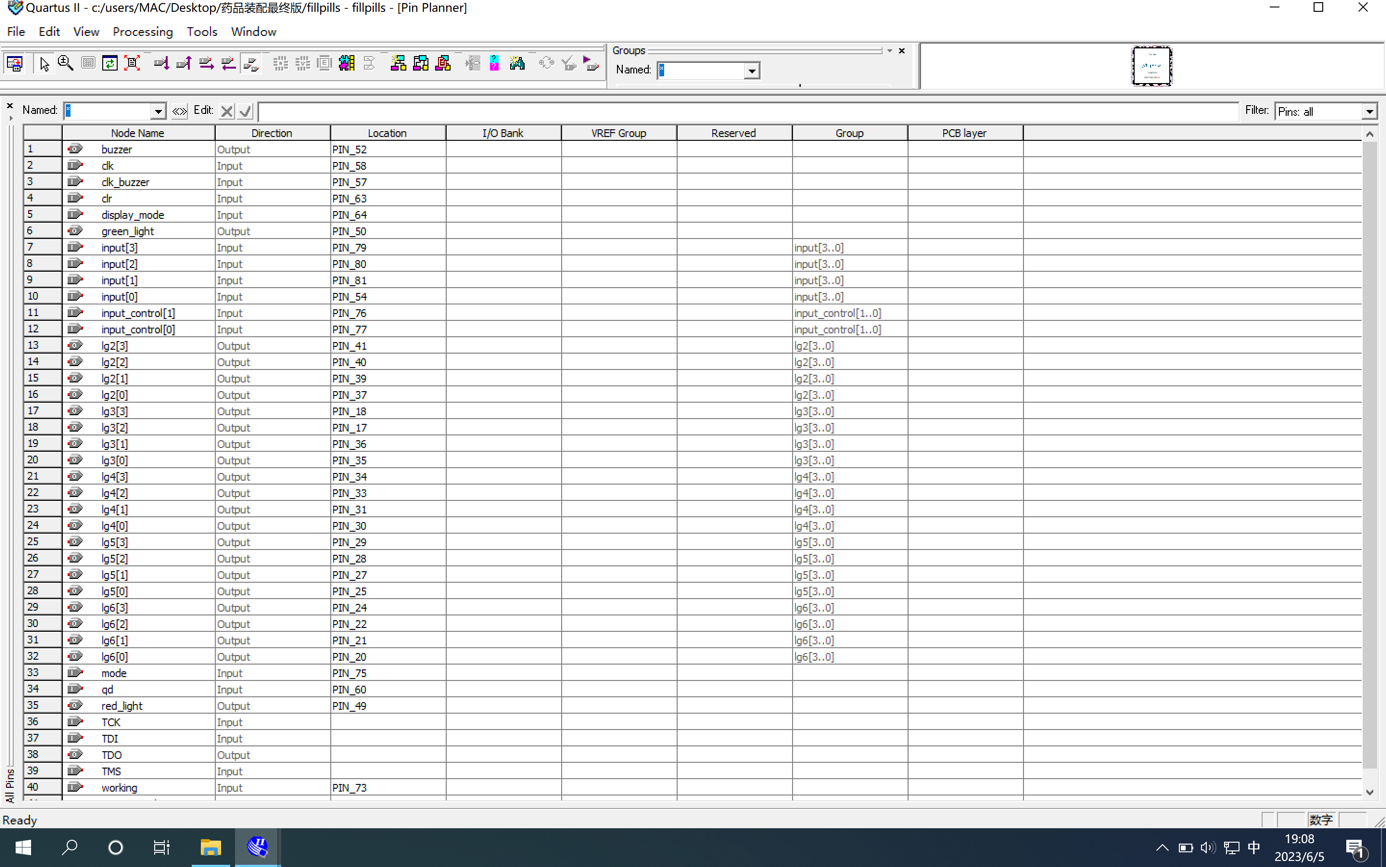
red\_light<=(error or finish); --出错或装载完毕，红灯亮

green\_light<=NOT(error or finish); --正常情况下，绿灯亮

start<= working AND mode;

end arc;

TEC-8引脚分配如下：



图形用户界面, 应用程序, 表格, Excel

描述已自动生成

**三、调试过程中所遇问题以及解决方案**

在编程时我们遇到了很多使编译无法通过的问题。在语法上有：在一个进程中使用两个时钟边沿、在两个不同进程中对同一信号进行赋值等；在分配资源上，有资源超限导致寄存器无法正常分配；这些都是在数字逻辑理论课中所不曾涉及到的问题。

在第一版的程序中，我们实现了基础的预设和装瓶功能，但遇到了预设时无法正确打入数值的问题，后来通过更改预设模式下的敏感信号表，以及将if-else语句改为case语句解决了这个问题。

然后我们又遇到了计数器乱跳的情况：每瓶装片数以及装瓶数在每瓶装药片大于15，装瓶数大于10片的情况下，在装到第4瓶左右时，装瓶速度会突然加快（远大于装瓶信号10Hz），在一秒内装到第8瓶，然后恢复正常。在检查多遍代码的情况下，我们并没有发现代码有什么逻辑问题，于是我们怀疑是实验台出了问题。但更换实验台也并没有解决这个问题。最终我们猜测是由于资源占用过大导致系统无法合理分配寄存器所致。于是我们对程序进行优化，在优化后的第二版程序中，跳变的问题再没有发生过。

第二版程序中，我们遇到了装瓶结束后无法正常停止计数的问题。导致这个问题的原因是，每次瓶中所装片数为（预设片数-1）时，在下一个时钟信号来临时，自动归零，然后瓶数+1。所以在这种情况下，瓶中所装片数永远无法达到预设片数。而装瓶终止条件被设置为“当前所装片数=预设片数，当前所装瓶数=预设瓶数”，所以无法正常终止。意识到这个问题后，我们将程序改为，每次瓶中所装片数为预设片数时，在下一个时钟信号来临时，自动归零，然后瓶数+1——在现实生活中对应装满一瓶后进行封装，并取出下一个空瓶。

随后我们在准备添加“闪烁设置”和“蜂鸣器告警”功能时，又遇到了资源不够的问题。我们在这个问题上浪费了很多时间进行优化，比如将if-else语句改为case语句，以及将一个12位的标准逻辑矢量拆分为三个4位的标准逻辑矢量，但是这些方法都没有解决问题。最后我们意识到可以使用元件例化的方法来节约资源。正好在电子中实验中已经写好的计数器可以直接使用，我们便改写了程序的整体框架，最终所用晶体管由125个降至91个。我们把改写后的程序称为第三版。

在第三版程序的调试中我们没有遇到大的问题，只有一些程序中的逻辑错误。

在最终版（添加“闪烁设置”和“蜂鸣器告警”功能）的程序中，我们也没有遇到令人费解的调试问题，大多数问题都是程序上的逻辑漏洞：比如用一个计数器来控制蜂鸣器响3下，但计数器却没有连接时钟信号等。

**四、团队分工**

药片装瓶系统实验主要由肖琪臻和郭栩源负责。

郭栩源完成了药片装瓶的基础功能，以及蜂鸣器的告警。

肖琪臻在资源不够的情况下使用元件例化的方法重写了程序框架，以及之后完成了闪烁显示功能。

**五、用户手册**

输入区：

K3～K0用于输入4位8421码；

K5～K4用于控制输入哪一位（“需装总瓶数”和“每瓶需装药片数”的个位和十位）：

00控制输入每瓶需装药片数的个位，

01控制输入每瓶需装药片数的十位，

10控制需装总瓶数的个位，

11控制需装总瓶数的十位；

K6用于控制工作模式与暂停模式：1为工作模式，0为暂停模式；

K14用于切换显示模式：

在工作模式下，0显示“当前药瓶中药片数”+“已装瓶数”，

1显示“总药片数”；

K15用于异步清零。

输出区：

LG6～LG3根据工作模式和显示模式分别进行显示；

红灯亮，且蜂鸣器不断发出声音表示“输入非法，需重新输入”；

红灯亮，且蜂鸣器响3下表示药片装瓶完毕；

绿灯亮，表示正常情况。

**调试日志**

实验一调试日志：

3月初，组队完成。

3月中旬，张棋睿和肖琪臻开始第一次讨论，进行第一次接洽，确定了目标和完成时间的规划。

3月20日，开始书写代码，肖琪臻同学完成了顶层设计和简要的输出模块。

3月28日，张棋睿同学完成了控制模块，两人共同完成了时间模块，至此开始基础功能的上机调试和debug。

接下来几天是调试时间，先后解决时钟不跳、时钟乱跳、显示bug、设置bug等问题。

4月初，调试完成，解决了遇到的问题。至此基础功能完成，能实现正常的时钟推进和设置时间，代码更新为第四版。

5月初，张棋睿在已经实现的三大模块上，开始了附加功能的实现，vhdl代码主要由张棋睿书写，调试由张棋睿和姜扬皓共同完成。

5月11日，完成了整点报时（第五版）。

5月13日，完了闹钟功能、闪烁功能（第六版）。

5月18日，实现了同步模式（第七版）。

5月19日，增加了一个清零的功能，为前面所遗忘的基础功能（第八版）。

5月20日-5月23日，实现了音乐的功能。期间出现的问题较多（具体问题在上文已阐述），张棋睿和姜扬皓进行了各种调试，克服了种种未知的软件硬件困难，在5月23日完成。至此所有附加功能和基础功能实现，电子钟完成（第九版）。

5月24日，进行验收。

5月末6月初 ，书写实验报告。

实验二调试日志：

3月初，组队完成。

4月初，郭栩源完成初版代码，实现基础的预设和装瓶功能；

4月中旬，郭栩源和肖琪臻一起简化了程序逻辑结构，且使用case语句大量替换if- else语句，完成第一次优化，解决计数器乱跳的问题；

5月上旬，药片装瓶系统所有基本功能完成，但是使用资源太多，无法支持完成附加功能，肖琪臻使用元件例化的方式修改了代码框架，使用了电子钟系统中已经编写好的计数器，成功将使用晶体管数量由125个降至91个；

5月中旬，肖琪臻实现了显示闪烁功能，郭栩源实现了音乐告警提示功能，并进行初步调试；

5月20日，所有功能调试完毕；

5月24日，进行验收；

5月末6月初 ，书写实验报告。

**各成员心得总结**

肖琪臻:

设计

如何去按照需求去设计产品是一件值得推敲的事情。其实我们在设计的过程中还需要挖掘潜在的，没有直接被提及的需求，并满足这样的需求，这样的设计才是好的设计。同样，好的设计应该对于设计者本人在设计的过程中来说也是方便的。一下我会谈及对以上两个认识在本次数电课设中的体现：

首先，在设计的过程中我们先认真分析了课程设计的基本和进阶要求，也根据已经存在的电子钟进行学习调研，发现电子钟（电子表）已经有了一套完整的交互模式。在电子钟这个项目上我们并不需要去优化太多已有的设计，而只需要实现类似的功能即可。我们就将功能分为三个模式：时钟、设定、闹钟（传统电子手表如卡西欧为四个：时钟、闹钟、秒表、设定）。再自顶向下分析，我们需要三个计数器来计时，分频器来处理时钟信号，两个计数器来设定闹钟。三种现实模式，分别输出时钟，时钟设定，闹钟设定。最后，闹钟和整点的蜂鸣器。这些功能和他们的交互，以及有限的资源让我们不断打磨我们的设计，最终找到在试验箱上最优的交互方法，固定设计，修改bug。自定向下的方式，让我们很快明确了设计的目标，分配了设计的任务，最后完成我们的设计。而挖掘到的需求，比如在调试时没有调整好的蜂鸣器太吵想要关掉，保留到了最终的设计中，因为我们觉得实际使用时也有必要关闭蜂鸣器，比如一个慵懒的星期六的早上。

其次，在设计的过程中我们发现老师建议的使用原理图连接模块的设计方式非常有用。但认识到这个事情的重要性已经到验收之前了。我们原本使用了一个VHDL portmap来设计，这是非常容易出错且枯燥的工作，使用原理图对于实现这个模块的连接时容易且轻松的。

团队协作

团队协作也是非常重要的一环。在这个课程设计当中，所有的同学都深入地参与到了这个课程设计。包括我们电子中团队的实践和认识在之后也帮助了药瓶计数器的设计。我们团队的沟通，分享，分工和相互促进才使我们的课程设计得以按时保证质量地完成。

举个例子，在设计电子钟的过程中，我们获得了很多经验，如编译、测试、写入、运行、分配接口等。以及一些设计好的资产如计时器，分频器，闪烁的实现，现实切换的实践等等。以上这些知识和设计好的模块极大地减小了设计药片计数器的设计难度；对药片计数器完全推导重来的决定也极大地节约了设计资源，从而能够让我们添加更多的功能。

可以做得更好的地方

在实现上，我们使用了VHDL语言进行电路设计。我们在设计过程中发现使用原理图连接模块的方式更容易，也更轻松。在未来的设计中，我们可以更深入地学习并应用这种方式。此外，在团队协作方面，我们可以更加注重沟通和协作，以提高效率和质量。

郭栩源：

数字逻辑课程设计在上半学期的数字逻辑理论部分的基础上进行了延伸。通过将理论知识点串联起来，帮助我整体了解了硬件程序设计的大致流程，同时也让我深刻感受到了硬件实验与软件实验之间的差异。这次课程设计让我学习到了很多。

首先，在课程设计中，我学会了进行模块设计。通过将整个程序划分为各个模块，我能够更好地组织代码，使其更易于理解和维护。模块设计为我提供了一个清晰的思路，使得我能够系统地思考问题，将复杂的任务分解为简单的子任务，从而更高效地完成设计。通过这种模块化的方法，我能够将复杂的硬件程序分解成多个独立的部分，并对每个部分进行单独设计和测试。这种分而治之的策略不仅提高了代码的可读性，还便于后续的调试和维护工作。

其次，仿真实验对于我理解程序行为和验证逻辑的正确性起到了至关重要的作用。通过使用仿真工具，我能够在虚拟环境中模拟和观察电路的运行情况。仿真实验使我能够在实际上机之前发现和修复潜在的错误，节省了时间和资源。在仿真过程中，我可以通过输入不同的测试数据，观察输出结果是否符合预期。如果出现问题，我可以对设计进行调整，直到逻辑正确。这种迭代的过程帮助我深入理解程序的行为，同时也增强了我的问题解决能力。

最后，上机实验给了我一个真实的环境来测试和验证我的设计。通过将代码加载到实际的硬件电路中，我可以亲身体验程序的运行情况，了解硬件和软件之间的互动。这种实际操作的经验对于我理解和掌握硬件编程技术至关重要。在上机实验中，我能够面对真实的硬件限制和挑战，例如时序要求、资源限制等。通过与实际硬件的交互，我更好地理解了程序在硬件平台上的运行特点，并学会了优化设计以满足特定的硬件要求。

在我的理解中，模块设计、仿真实验和上机实验是密不可分的环节。缺少了高层模块设计，程序设计就会失去了一个明确的框架；若没有进行仿真实验而盲目进行上机实验，效率将大大降低，可能需要多次上机来解决程序的逻辑错误；而如果没有进行最后的上机实操，就无法确定程序是否真正能在实际电路上运行。因此，这三个环节相互依存，相辅相成，共同构成了一个完整的硬件程序设计流程。

总而言之，数字逻辑课程设计为我提供了一个全面的学习平台，使我能够将理论知识与实际应用相结合。通过模块设计、仿真实验和上机实验的有机结合，我深入了解了硬件编程的过程和技术要点，这些内容在理论课程的部分中常常被忽略的。这次课程设计的收获对我未来的学习和职业发展具有重要意义，使我在硬件编程领域迈出了坚实的一步。我相信通过不断的实践和学习，我将能够在硬件编程领域取得更大的收获。

张棋睿：

数字逻辑与数字电路是计算机科学与技术专业的一门重要课程，也是我们计算机学生必须掌握的技能之一。在这门数电课程设计中，我把大量时间投入了设计实践中，最终完成了一个电子钟和药品装片系统。在这个过程中，我收获了很多，并从中学到了很多宝贵的经验。

首先，在这个课程设计中，我的VHDL语言水平得到了全面提升。我熟练掌握了VHDL语言的使用及使用VHDL语言实现电路设计的基本方法。在本次课程设计中的电子钟模块，我尝试采用了自顶向下的方法进行VHDL语言的书写，先写出顶层设计模块，然后采用了对VHDL的元件例化、实体调用和实体实现，来完成整个代码。这次的VHDL语言的顶层设计实现，使我对整个电子钟设计有了一个宏观认识，我第一次体会到了顶层设计在工程实践中的重要性，它是整个工程实践的基石，是整个项目实现的基础。这是这个课程设计我首要的心得。有了本次数电课设的经验，在以后的工程设计中，不管是使用VHDL语言还是其它语言，我都会注重对顶层的把握，至少也需要一个基本的宏观认识。

在具体的VHDL语言代码上，我反复运用if语句、switch-case语句、各种逻辑表达式，使用各种脉冲信号和电平信号等，在一次次的编译报错和调试错误的情况下锻炼出熟练的VHDL代码能力。在不同实体之间的信号传递、对时钟信号和电平信号的判断、对分频器和计数器间的运用、对各种未知的报错信息的处理和对VHDL语言的逻辑错误导致功能无法实现的问题处理，都使我的VHDL语言的能力更上一层楼。

除了VHDL语言的能力，本次实验使我印象更深的是硬件的运用与调试。与纯语言类的课程设计不同，数电课程设计的最大特色就是还要对硬件TEC-8进行操作。硬件的操作是本次课程设计的最大特色，也是我在完成课程设计时遇到问题最多的方面。硬件产生的问题，大多是未知的、难以预见的、难以调试的。它不像代码的错误，大多数是可见的语法错误和逻辑错误。一段VHDL代码，如果以纯逻辑的角度来审视没有问题，也不代表它能在硬件上正常运行。在实现电子钟的过程中，我先后遇到了时钟不跳、时钟乱跳、时钟无法设置、音乐颤音、硬件资源不足等问题，调试解决这一系列问题，是本次课设钟最难能可贵的部分。通过研究和思考、充分的分析与交流，我们最终都找到了解决问题的方法。这些问题的解决方式不仅仅是语言的使用和知识的累积，更是对实际问题进行深入分析和了解，对问题进行系统的探讨和提出各种方案，真实有效地提高了我解决实际问题的能力。这种能力将陪伴我，成为我一生中难能可贵的财富。

此外，这个课程设计也让我更加深入了解了电子钟和药品装片系统的实现原理。通过本次课程设计，我了解到这些系统常见的构造原理、各种元器件的配合和使用，以及各种各样的设计和调试技巧。我认为这是一段宝贵的经验，让我为今后接触各种不同类型的问题和任务打下了坚实的基础。

在实际设计过程中，我感受到了自己对数字逻辑和电路知识的掌握能力大大提高，可以更快地理解、设计、调试数字电路。除此之外，这个课程设计也提升了我的团队协作能力。当我们一开始拿到这个任务的时候，我们分工清晰，紧密配合，分别负责系统需求、电路设计、程序编写等方面的工作。在完成这个任务的过程中，我们互相支持，帮助和鼓励，克服了很多困难。我们的合作精神和紧密配合，在这过程中体现得淋漓尽致，为今后接触实际项目中的团队协作打下了基础。

姜杨皓：

本次数字逻辑课程设计是在上学期数字逻辑课程的基础上进行的，依托于上学期所学习的基础知识，是对上学期所学习知识的一次加强巩固与检验，提高了我对数字逻辑基本理论知识的掌握。课程设计依托于VHDL语言进行，在实践过程中锻炼了我的VHDL编码能力，也学到了一些新的语法知识。

在本次实验中我第一次尝试利用VHDL进行模块化编码，多个模块之间相互关联，共同由顶层模块调用，对我来说既是一次全新的体验，也是一次新的挑战。如何合理地在不同模块之间传递信号也是不小的问题。在实验中我们又一次使用了TEC-8机器，但第一次实践使用EMP7128芯片，需要统筹安排管脚等信息，也是一次全新的经历。

在本次实验中，我们出现了各种各样的bug，有些是vhdl的语法问题，有些则是由系统硬件造成。我们多次出现实际上机情况与仿真波形不同，这种bug尤为头疼，只能通过修改代码进行各种尝试来修正，在尝试的过程中逐渐熟悉了TEC-8系统，学到了一些课本上没有的知识，也锻炼了我的debug能力。

在前期，我们在还未完成所有功能的情况下就已经占用了所有资源，无法继续增加功能。为了解决这一问题，我们进行思考，修改了一些语句，也优化了一些模块，最终成功节省资源，完成了所有功能。

除此之外，我还认识到了团队分工合作的重要性。我们在前期分工不够明确，导致只有两人在推进工作，其余两人在等待进展。后来我们及时认识到了问题，分为两组同步推进工作，提高了时间利用率。团队分工明确以及良好的团队协作能力尤为重要。