**大连理工大学本科实验报告**

题目：数字钟

课程名称： 数字电路课程设计

学院（系）： 电子信息与电气工程学部

专 业： 电子信息工程

班 级： 电信1801

学生姓名： 杨题鸣

学 号： 201883016

完成日期： 20201010

成 绩：

2020 年 10 月 11 日

**数字电路课程设计得分表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **考勤** | **线上实验**  **5分** | **课程设计**  **35分** | **考试**  **55分** | **其它**  **5分** | **合计** | **最后得分** |
|  |  |  |  |  |  |  |

1. **数字钟课程设计要求：**

**在一个顶层实体里选做如下内容。第1题20分，其他每题3分。可选做。**

1. 设计一个具有‘时’、‘分’、‘秒’的十进制数字显示（小时从00～23）计时器。

2、整点报时。仿中央人民广播电台的整点报时信号，即从第59分50秒算起，每隔2秒钟发出一次信号，连续5次，最后一次信号结束即达到整点。不同步扣2分，可通过LED闪烁实现。

3、实现手动校时、校分、校秒功能。（缺一项扣一分，如果利用系统时钟校时，扣1分）

4、定时与闹钟功能，只需要设置分钟和小时。手动设置能在设定的时间发出闹铃声,声音用LED实现。

5、设计一个10个数的倒计时，闪烁显示，闪烁频率自定（无闪烁扣2分）。

6、用LCD液晶屏来显示当前时间。

**二、课程设计考试（55分）：**

**考试题目：**

**分项得分：**

**1、**

**2、**

**3、**

**4、**

**5、**

**6、**

**7、**

**合计：**

**数字钟**

杨题鸣

1. 大连理工大学电子信息与电气工程学部，辽宁大连，116023；

2．大连理工大学电工电子实验中心，辽宁大连，116023）

**摘要**：在公共场合当中，我们常需要看时间，这就需要一个精准的数字时钟来为我们服务，数字钟是一种用数字电路技术实现时、分、秒计时的装置，与机械式时钟相比具有更高的准确度和直观性，且无机械装置，具有更长的使用寿命，因此得到了广泛的使用。本数字钟由verilog语言来完成。有基本的时钟功能，还有闹钟，闪烁10秒倒计时，整点播报功能，lcd显示，等等

**关键词**：组合逻辑电路；时序电路；verilog

**Digital clock**

YANG TIMING

(1. Department of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116023, China;2. Laboratory Center of Electrical and Electronics, Dalian University of Technology, Dalian 116023, China)

**Abstract**（小5号黑体）**: In public, we often need to look at time, which requires a precise digital clock to serve us, digital clock is a kind of when using a digital circuit technology, minutes and seconds timing device, has higher accuracy compared with the mechanical clock and intuitive, and no mechanical device, has a longer service life, therefore has been widely used. This digital clock is made by Verilog language. There's the basic clock function, there's the alarm clock, the flashing 10 second countdown, the hourly broadcast, THE LCD display, etc**

**Key words** **: Verilog, clock function, alarm, hourly broadcast, countdown.**

1 设计要求

1. 设计一个具有‘时’、‘分’、‘秒’的十进制数字显示（小时从00～23）计时器。

2、整点报时。仿中央人民广播电台的整点报时信号，即从第59分50秒算起，每隔2秒钟发出一次信号，连续5次，最后一次信号结束即达到整点。通过LED闪烁实现。

3、实现手动校时、校分、校秒功能。

4、定时与闹钟功能，只需要设置分钟和小时。手动设置能在设定的时间发出闹铃声,声音用LED实现。

5、设计一个10个数的倒计时，闪烁显示，闪烁频率2HZ

6、用LCD液晶屏来显示当前时间。

2 设计分析及系统方案设计

系统总体框图：

图示

描述已自动生成

2.1分频模块

简介：将50MHZ的时钟信号变成1HZ的CLK信号。通过加法，div\_cnt从0加到4999999.每当完成一次加的过程，clk\_div由0变成1。从而实现分频功能。

输入：

clk --总的clk，接50mhz的石英振荡器 ------->PIN\_N2

输出：

clk\_div –分频变成1HZ

敏感条件：

always @(posedge clk )

2.2自动计时模块

简介：本模块用于自动计时功能，从而实现从00:00:00—>23：59：59的计时功能。用if先判断是否处于23:59:59分，若不处于，然后在进行计数功能，首先判断最低位secL是否为9，如果是则让其归零，否则再判断secH是否为9，如果是则让其归零，一直判断到hourL是否为9，如果是则让其归零。如果每一项都不为最大进位数字，那么分别由最高位到最低位嵌套加1.

输入：

set\_mod ；set\_alarm

自动计时：set\_mod==1 set\_alarm==x --------->sw0--PIN\_N25

输出：

reg [3:0]secL\_1; ---秒低位

reg [3:0]secH\_1; ---秒高位

reg [3:0]minL\_1; ---分钟低位

reg [3:0]minH\_1; ---分钟高位

reg [3:0]hourL\_1; ---小时低位

reg [3:0]hourH\_1; ---小时高位

敏感条件：

always @(posedge clk\_div)

2.3手动调整模块

2.3.1手动调节时分秒功能模块

简介：本模块用于选择对哪一项进行时分秒调节，KEY按键每次按动一次就option\_1加一次1，一直循环进行，就可以进行选择哪一项调节。其设计思路类似于计数器。

输入：

option\_pattern，option\_1

pattern\_set==0表示手动调秒; pattern\_set==1表示手动调分; pattern\_set==2表示调时

输出：

option\_1

敏感条件：

always@(posedge option\_pattern)

2.3.2 手动调节时间以及闹钟设置模块

简介：本模块用于调整现在的时间以及设置闹钟的功能。通过两个输入set\_mod，set\_alarm 分别控制是 手动计时 .手动设置闹钟。用if先判断是否处于23:59:59分，若不处于，然后在进行手动调节时间。pattern\_set==0表示手动调秒。pattern\_set==1表示手动调分，pattern\_set==2表示调时。原理与自动计时模块的进位功能类似。

输入：

set\_mod ；set\_alarm

手动计时：set\_mod==0 且 set\_alarm==0

手动设置闹钟：set\_mod==0 且 set\_alarm==1

输出：

reg [3:0]secL\_2; ---秒低位

reg [3:0]secH\_2; ---秒高位

reg [3:0]minL\_2; ---分低位

reg [3:0]minH\_2; ---分高位

reg [3:0]hourL\_2; ---时低位

reg [3:0]hourH\_2; ---时高位

敏感条件：

always@(posedge up\_switch)

2.4计时输出模块。

简介：该模块主要用于将前面的自动计时模块或者是手动调节时间以及闹钟设置模块的输出显示赋值功能。将自动计时模块或者是手动调节时间以及闹钟设置模块的输出赋值给计时输出模块用于在lcd1602液晶屏，以及数码管上的显示。

输入：

set\_mod ；set\_alarm

手动计时的输入：set\_mod==0 且 set\_alarm==0

手动设置闹钟的输入：set\_mod==0 且 set\_alarm==1

输出：

reg [3:0]secL; ---秒低位

reg [3:0]secH; ---秒高位

reg [3:0]minL; ---分低位

reg [3:0]minH; ---分高位

reg [3:0]hourL; ---时低位

reg [3:0]hourH; ---时高位

敏感条件：

always@(secL\_1 or secH\_1 or minL\_1 or minH\_1 or hourL\_1 or hourH\_1 or secL\_2 or secH\_2 or minL\_2 or minH\_2 or hourL\_2 or hourH\_2)

2.5 整点报时模块

简介：从59：50到00：00，产生以1为起始的0-1序列，对应指示灯间隔为2s的闪烁。通过以下的输入放到if语句里面，当满足minH==5 && minL== 9 && secH== 5并且secL==0 || secL==2 || secL==4 || secL==6 || secL==8时，blink为1否则为0。从而可以用blink控制led 的闪烁。

输入：

minH==5 && minL== 9 && secH== 5

secL==0 || secL==2 || secL==4 || secL==6 || secL==8

输出：

blink

敏感条件：

always @(secL)

2.6 倒计时模块

简介：本模块有一个新的分频模块，模块的分频参数为parameter m=24999999用于闪烁频率为2hz。倒计时模块当中，基本思路是通过夹杂：“0，灭，1，灭，2，灭，3，灭，4，灭，5，灭，6，灭，7，灭，8，灭，9，灭，0”来进行循环显示最终显示在de2上面的HEX0数码管上。

输入：

clk

输出：

[6:0]qout\_recnt

敏感条件：

always@(posedge clk)

2.7 闹钟设定模块和闹钟触发模块

简介：当闹钟所设定的时间（分.时）与当前的时间一致时minL==alarm\_minL && minH==alarm\_minH && hourL==alarm\_hourL && hourH==alarm\_hourH，那么alarm\_sound由0变成1，那么由ledr1就会保持长亮。直到时间到下一秒为止，alarm\_sound由1变成0；

输入：

alarm\_minL

alarm\_minH

alarm\_hourL

alarm\_hourH

输出：

alarm\_sound

敏感条件：

always@(secL)

2.8 数码管可视化模块

简介：用于数码管可视化显示

输入：

set\_alarm

secL

输出：

alarm\_sound

qout\_i

敏感条件：

always@(secL or alarm\_secL)

2.9 lcd液晶屏显示

模块9.1：LCD分频模块

输入：

Clk

输出：

clk\_2ms

模块的敏感条件：

posedge clk。

模块9.2：译码模块

输入：

[3:0]secL

[3:0]secH

[3:0]minL

[3:0]minH

[3:0]hourL

[3:0]hourH

输出：

用作模块9.3循环显示模块的输入数据

[7:0]secL\_lcd

[7:0]secH\_lcd

[7:0]minL\_lcd

[7:0]minH\_lcd

[7:0]hourL\_lcd

[7:0]hourH\_lcd

模块的敏感条件：

secL

secH

minL

minH

hourL

hourH

模块9.3：循环输入模块

输入：

[7:0]secL\_lcd

[7:0]secH\_lcd

[7:0]minL\_lcd

[7:0]minH\_lcd

[7:0]hourL\_lcd

[7:0]hourH\_lcd

输出：

LCD的显示

模块的敏感条件：

posedge clk\_2ms or posedge rst

3系统以及模块硬件电路设计

系统硬件电路设计

说明：

图中分别包含了输入输出引脚。输入引脚主要是一些控制引脚，以及时钟引脚。输出引脚主要包括了数码管的输出引脚，以及lcd液晶屏的显示输出模块。

3.2各引脚说明：

图片包含 窗户, 游戏机, 建筑, 窗帘

描述已自动生成

表格

描述已自动生成

4 系统的VHDL设计

module clockmyself(

input clk, // --总的clk，接50mhz的石英振荡器 ------->PIN\_N2

input option\_pattern, // 用于手动调节时分秒的按钮 --------->KEY3--PIN\_W2

input up\_switch, // 用于选择手动时间后的a=a+1工作 -------->KEY2--PIN\_p23

input set\_mod, // --------->sw1--PIN\_N26

input set\_alarm, //\* 自动计时：set\_mod==1 --------->sw0--PIN\_N25

//\* 手动计时：set\_mod==0 且 set\_alarm==0//

//\* 手动设置闹钟：set\_mod==0 且 set\_alarm==1//

output reg blink, // 整点报时模块output->blinking// --------->LEDR0--PIN\_AE23

output reg [6:0]qout\_recnt, // 用于在数码管上显示倒计时// ==>HEX0（完成）

input recout, // 控制是否倒计时// --------->SW17--PIN\_V2

output reg alarm\_sound, // 显示闹钟是否工作状态.led show// --------->LEDR1--PIN\_AF23

//时分秒的数码管输出

output reg [6:0]qout\_1, //==>HEX2（完成）

output reg [6:0]qout\_2, //==>HEX3（完成）

output reg [6:0]qout\_3, //==>HEX4（完成）

output reg [6:0]qout\_4, //==>HEX5（完成）

output reg [6:0]qout\_5, //==>HEX6（完成）

output reg [6:0]qout\_6, //==>HEX7（完成）

output reg clk\_div, //fenping

output reg [1:0]option\_1, //chice led

//lcd的input和output

input rst, //rst为全局复位信号（高电平有效）

output LCD\_EN,LCD\_ON, //LCD\_EN为LCD模块的使能信号（下降沿触发）

output reg RS, //RS=0时为写指令；RS=1时为写数据

output RW, //RW=0时对LCD模块执行写操作；RW=1时对LCD模块执行读操作

output reg [7:0] DB8 //8位指令或数据总线

);

//---分频模块1--//

parameter n= 49999999;//--选用50Mhz的石英振荡器--//

reg [30:0]div\_cnt;

//reg clk\_div;

always @(posedge clk )

begin

if(div\_cnt==n)

begin

div\_cnt<=0;

clk\_div<=1;//--将50mHZ分成50m份---//

end

else

begin

div\_cnt<=div\_cnt+1;

clk\_div<=0;

end

end

//--模块2：自动计时模块--//

//--实现 00：00：00 ==> 23：59：59 的循环的时间计时--//

//设置一个flag，来控制此模块//

reg [3:0]secL\_1;

reg [3:0]secH\_1;

reg [3:0]minL\_1;

reg [3:0]minH\_1;

reg [3:0]hourL\_1;

reg [3:0]hourH\_1;

always @(posedge clk\_div)

begin

if(set\_mod==1)

begin

//23：59：59

if(hourH\_1==4'b0010 && hourL\_1==4'b0011 && minH\_1==4'b0101 && minL\_1==4'b1001 && secH\_1==4'b0101 && secL\_1==4'b1001)

begin

secL\_1<=4'b0000;

secH\_1<=4'b0000;

minL\_1<=4'b0000;

minH\_1<=4'b0000;

hourL\_1<=4'b0000;

hourH\_1<=4'b0000;

end

else

//--no-23：59：59->计数//

if(secL\_1==9)

begin

secL\_1<=4'b0000;

if(secH\_1==5)

begin

secH\_1<=4'b0000;

if(minL\_1==9)

begin

minL\_1<=4'b0000;

if(minH\_1==5)

begin

minH\_1<=4'b0000;

if(hourL\_1==9)

begin

hourL\_1<=4'b0000;

hourH\_1<=hourH\_1+1;

end

else

hourL\_1<=hourL\_1+1;

end

else

minH\_1<=minH\_1+1;

end

else

minL\_1<=minL\_1+1;

end

else

secH\_1<=secH\_1+1;

end

else

secL\_1<=secL\_1+1;

end

else if(set\_mod==0 && set\_alarm==0)

begin

//--让手动调整后的时间进行自动计时--//

secL\_1<=secL\_2;

secH\_1<=secH\_2;

minL\_1<=minL\_2;

minH\_1<=minH\_2;

hourL\_1<=hourL\_2;

hourH\_1<=hourH\_2;

end

end

//--模块3:手动记时模块--//

//--模块3\_1:手动调节时分秒选择--//

//--pattern\_set==0表示手动调秒; pattern\_set==1表示手动调分; pattern\_set==2表示调时-//

always@(posedge option\_pattern)

begin

if(option\_1==3)

begin option\_1<=0; end

else

begin option\_1<=option\_1+1; end

end

//--模块3\_2:手动调节时模块--//

reg [3:0]secL\_2;

reg [3:0]secH\_2;

reg [3:0]minL\_2;

reg [3:0]minH\_2;

reg [3:0]hourL\_2;

reg [3:0]hourH\_2;

always@(posedge up\_switch)

begin

//手动校时模式

if(set\_mod==0 && set\_alarm==0) //--手动计时：set\_mod==0 且 set\_alarm==0--//

begin

//23：59：59//

if(hourH\_2==4'b0010 && hourL\_2==4'b0011 && minH\_2==4'b0101 && minL\_2==4'b1001 && secH\_2==4'b0101 && secL\_2==4'b1001)

begin

secL\_2<=4'b0000;

secH\_2<=4'b0000;

minL\_2<=4'b0000;

minH\_2<=4'b0000;

hourL\_2<=4'b0000;

hourH\_2<=4'b0000;

end

else

begin

//--no-23：59：59->计数--//

//option\_1==0-->调秒//

if(option\_1==0)

begin

if(secL\_2==9)

begin

secL\_2<=4'b0000;

if(secH\_2==5)

secH\_2<=4'b0000;

else

secH\_2<=secH\_2+1;

end

else

secL\_2<=secL\_2+1;

end

//option\_1==1-->调分//

if(option\_1==1)

begin

if(minL\_2==9)

begin

minL\_2<=4'b0000;

if(minH\_2==5)

minH\_2<=4'b0000;

else

minH\_2<=minH\_2+1;

end

else

minL\_2<=minL\_2+1;

end

//--option\_1==1-->调时--//

if(option\_1==2)

begin

if(hourL\_2==3 && hourH\_2==2)

begin

hourL\_2<=0;

hourH\_2<=0;

end

else

begin

if(hourL\_2==9 )

begin

hourL\_2<=4'b0000;

if(hourH\_2==2)

hourH\_2<=4'b0000;

else

hourH\_2<=hourH\_2+1;

end

else

hourL\_2<=hourL\_2+1;

end

end

end

end

if(set\_mod==0 && set\_alarm==1) //手动设置闹钟：set\_mod==0 且 set\_alarm==1//

begin

//23：59：59//

if(alarm\_hourH==4'b0010 && alarm\_hourL==4'b0011 && alarm\_minH==4'b0101 && alarm\_minL==4'b1001 && alarm\_secH==4'b0101 && alarm\_secL==4'b1001)

begin

alarm\_secL<=4'b0000;

alarm\_secH<=4'b0000;

alarm\_minL<=4'b0000;

alarm\_minH<=4'b0000;

alarm\_hourL<=4'b0000;

alarm\_hourH<=4'b0000;

end

else

begin

//--no-23：59：59->计数--//

//调秒

if(option\_1==0)

begin

if(alarm\_secL==9)

begin

alarm\_secL<=4'b0000;

if(alarm\_secH==5)

alarm\_secH<=4'b0000;

else

alarm\_secH<=alarm\_secH+1;

end

else

alarm\_secL<=alarm\_secL+1;

end

//调分

if(option\_1==1)

begin

if(alarm\_minL==9)

begin

alarm\_minL<=4'b0000;

if(alarm\_minH==5)

alarm\_minH<=4'b0000;

else

alarm\_minH<=alarm\_minH+1;

end

else

alarm\_minL<=alarm\_minL+1;

end

//调时

if(option\_1==2)

begin

if(alarm\_hourL==3 && alarm\_hourH==2)

begin

alarm\_hourL<=0;

alarm\_hourH<=0;

end

else

begin

if(alarm\_hourL==9 )

begin

alarm\_hourL<=4'b0000;

if(alarm\_hourH==2)

alarm\_hourH<=4'b0000;

else

alarm\_hourH<=alarm\_hourH+1;

end

else

alarm\_hourL<=alarm\_hourL+1;

end

end

end

end

end

//模块4：计时输出模块//

//作为数码管的输入

reg [3:0]secL;

reg [3:0]secH;

reg [3:0]minL;

reg [3:0]minH;

reg [3:0]hourL;

reg [3:0]hourH;

// 该模块的输出至少1s更新一次，数码管上的示数也是至少1s更新一次 //

always@(secL\_1 or secH\_1 or minL\_1 or minH\_1 or hourL\_1 or hourH\_1 or secL\_2 or secH\_2 or minL\_2 or minH\_2 or hourL\_2 or hourH\_2)

begin

//自动计数模式

if(set\_mod==1)

begin

secL<=secL\_1;

secH<=secH\_1;

minL<=minL\_1;

minH<=minH\_1;

hourL<=hourL\_1;

hourH<=hourH\_1;

end

else

//手动计数模式

if(set\_mod==0 && set\_alarm==0) //手动计时：set\_mod==0 且 set\_alarm==0//

begin

secL<=secL\_2;

secH<=secH\_2;

minL<=minL\_2;

minH<=minH\_2;

hourL<=hourL\_2;

hourH<=hourH\_2;

end

end

//模块5：整点报时模块//

//--从59：50到00：00，产生以1为起始的0-1序列，对应指示灯间隔为2s的闪烁--//

always @(secL)

begin

if(minH==5 && minL== 9 && secH== 5) //--异步闪烁--//

begin

if(secL==0 || secL==2 || secL==4 || secL==6 || secL==8)

begin

blink<=~blink;

end

else

begin

blink<=0;

end

end

end

//模块6：倒计时模块//

//倒计时闪烁模块//

parameter m=24999999;

reg [50:0]q;

reg clk\_22;

always@(posedge clk)

begin

if(q==m)

begin

clk\_22<=1;

q<=0;

end

else

begin

q<=q+1;

clk\_22<=0;

end

end

reg [4:0]count;

reg [3:0]recnt;

always @(posedge clk\_22 or negedge recout)

begin

if(!recout)

begin

count<=4'b0000;

end

else

begin

if(count==19)

begin count<=0; end

else

begin

count <=count+1;

end

end

case(count)

5'b00000:recnt<=4'b1001;

5'b00001:recnt<=4'b1111;

5'b00010:recnt<=4'b1000;

5'b00011:recnt<=4'b1111;

5'b00100:recnt<=4'b0111;

5'b00101:recnt<=4'b1111;

5'b00110:recnt<=4'b0110;

5'b00111:recnt<=4'b1111;

5'b01000:recnt<=4'b0101;

5'b01001:recnt<=4'b1111;

5'b01010:recnt<=4'b0100;

5'b01011:recnt<=4'b1111;

5'b01100:recnt<=4'b0011;

5'b01101:recnt<=4'b1111;

5'b01110:recnt<=4'b0010;

5'b01111:recnt<=4'b1111;

5'b10000:recnt<=4'b0001;

5'b10001:recnt<=4'b1111;

5'b10010:recnt<=4'b0000;

5'b10011:recnt<=4'b1111;

default: recnt<=4'b0000;

endcase

end

always @(recnt)//用于在数码管上显示倒计时//

begin

case(recnt)

4'b0000:qout\_recnt<=7'b1000000;

4'b0001:qout\_recnt<=7'b1111001;

4'b0010:qout\_recnt<=7'b0100100;

4'b0011:qout\_recnt<=7'b0110000;

4'b0100:qout\_recnt<=7'b0011001;

4'b0101:qout\_recnt<=7'b0010010;

4'b0110:qout\_recnt<=7'b0000010;

4'b0111:qout\_recnt<=7'b1111000;

4'b1000:qout\_recnt<=7'b0000000;

4'b1001:qout\_recnt<=7'b0010000;

default: qout\_recnt<=7'b1111111;

endcase

end

//模块7：闹钟设定模块和闹钟触发模块//

//闹钟设定时间的可视化

reg [3:0]alarm\_secL;

reg [3:0]alarm\_secH;

reg [3:0]alarm\_minL;

reg [3:0]alarm\_minH;

reg [3:0]alarm\_hourL;

reg [3:0]alarm\_hourH;

//闹钟触发开始：判断闹钟设定时间 是否等于 现在的时间，如果是则灯亮//

reg [3:0]alarm\_cnt;

always@(secL)

begin

alarm\_sound<=1'b0;

if( minL==alarm\_minL && minH==alarm\_minH && hourL==alarm\_hourL && hourH==alarm\_hourH)

begin

alarm\_sound<=1'b1;

end

end

//模块8：可视化数码管模块//

always@(secL or alarm\_secL)

begin

if(set\_alarm==0)

begin

case(secL)

4'b0000:qout\_1<=7'b1000000;

4'b0001:qout\_1<=7'b1111001;

4'b0010:qout\_1<=7'b0100100;

4'b0011:qout\_1<=7'b0110000;

4'b0100:qout\_1<=7'b0011001;

4'b0101:qout\_1<=7'b0010010;

4'b0110:qout\_1<=7'b0000010;

4'b0111:qout\_1<=7'b1111000;

4'b1000:qout\_1<=7'b0000000;

4'b1001:qout\_1<=7'b0010000;

default:qout\_1<=7'b1111111;

endcase

end

else if(set\_alarm==1)

begin

case(alarm\_secL)

4'b0000:qout\_1<=7'b1000000;

4'b0001:qout\_1<=7'b1111001;

4'b0010:qout\_1<=7'b0100100;

4'b0011:qout\_1<=7'b0110000;

4'b0100:qout\_1<=7'b0011001;

4'b0101:qout\_1<=7'b0010010;

4'b0110:qout\_1<=7'b0000010;

4'b0111:qout\_1<=7'b1111000;

4'b1000:qout\_1<=7'b0000000;

4'b1001:qout\_1<=7'b0010000;

default:qout\_1<=7'b1111111;

endcase

end

end

always@(secH or alarm\_secH)

begin

if(set\_alarm==0)

begin

case(secH)

4'b0000:qout\_2<=7'b1000000;

4'b0001:qout\_2<=7'b1111001;

4'b0010:qout\_2<=7'b0100100;

4'b0011:qout\_2<=7'b0110000;

4'b0100:qout\_2<=7'b0011001;

4'b0101:qout\_2<=7'b0010010;

4'b0110:qout\_2<=7'b0000010;

4'b0111:qout\_2<=7'b1111000;

4'b1000:qout\_2<=7'b0000000;

4'b1001:qout\_2<=7'b0010000;

default:qout\_2<=7'b1111111;

endcase

end

else if(set\_alarm==1)

begin

case(alarm\_secH)

4'b0000:qout\_2<=7'b1000000;

4'b0001:qout\_2<=7'b1111001;

4'b0010:qout\_2<=7'b0100100;

4'b0011:qout\_2<=7'b0110000;

4'b0100:qout\_2<=7'b0011001;

4'b0101:qout\_2<=7'b0010010;

4'b0110:qout\_2<=7'b0000010;

4'b0111:qout\_2<=7'b1111000;

4'b1000:qout\_2<=7'b0000000;

4'b1001:qout\_2<=7'b0010000;

default:qout\_2<=7'b1111111;

endcase

end

end

always@(minL)

begin

if(set\_alarm==0)

begin

case(minL)

4'b0000:qout\_3<=7'b1000000;

4'b0001:qout\_3<=7'b1111001;

4'b0010:qout\_3<=7'b0100100;

4'b0011:qout\_3<=7'b0110000;

4'b0100:qout\_3<=7'b0011001;

4'b0101:qout\_3<=7'b0010010;

4'b0110:qout\_3<=7'b0000010;

4'b0111:qout\_3<=7'b1111000;

4'b1000:qout\_3<=7'b0000000;

4'b1001:qout\_3<=7'b0010000;

default:qout\_3<=7'b1111111;

endcase

end

else if(set\_alarm==1)

begin

case(alarm\_minL)

4'b0000:qout\_3<=7'b1000000;

4'b0001:qout\_3<=7'b1111001;

4'b0010:qout\_3<=7'b0100100;

4'b0011:qout\_3<=7'b0110000;

4'b0100:qout\_3<=7'b0011001;

4'b0101:qout\_3<=7'b0010010;

4'b0110:qout\_3<=7'b0000010;

4'b0111:qout\_3<=7'b1111000;

4'b1000:qout\_3<=7'b0000000;

4'b1001:qout\_3<=7'b0010000;

default:qout\_3<=7'b1111111;

endcase

end

end

always@(minH)

begin

if(set\_alarm==0)

begin

case(minH)

4'b0000:qout\_4<=7'b1000000;

4'b0001:qout\_4<=7'b1111001;

4'b0010:qout\_4<=7'b0100100;

4'b0011:qout\_4<=7'b0110000;

4'b0100:qout\_4<=7'b0011001;

4'b0101:qout\_4<=7'b0010010;

4'b0110:qout\_4<=7'b0000010;

4'b0111:qout\_4<=7'b1111000;

4'b1000:qout\_4<=7'b0000000;

4'b1001:qout\_4<=7'b0010000;

default:qout\_4<=7'b1111111;

endcase

end

else if(set\_alarm==1)

begin

case(alarm\_minH)

4'b0000:qout\_4<=7'b1000000;

4'b0001:qout\_4<=7'b1111001;

4'b0010:qout\_4<=7'b0100100;

4'b0011:qout\_4<=7'b0110000;

4'b0100:qout\_4<=7'b0011001;

4'b0101:qout\_4<=7'b0010010;

4'b0110:qout\_4<=7'b0000010;

4'b0111:qout\_4<=7'b1111000;

4'b1000:qout\_4<=7'b0000000;

4'b1001:qout\_4<=7'b0010000;

default:qout\_4<=7'b1111111;

endcase

end

end

always@(hourL)

begin

if(set\_alarm==0)

begin

case(hourL)

4'b0000:qout\_5<=7'b1000000;

4'b0001:qout\_5<=7'b1111001;

4'b0010:qout\_5<=7'b0100100;

4'b0011:qout\_5<=7'b0110000;

4'b0100:qout\_5<=7'b0011001;

4'b0101:qout\_5<=7'b0010010;

4'b0110:qout\_5<=7'b0000010;

4'b0111:qout\_5<=7'b1111000;

4'b1000:qout\_5<=7'b0000000;

4'b1001:qout\_5<=7'b0010000;

default:qout\_5<=7'b1111111;

endcase

end

else if(set\_alarm==1)

begin

case(alarm\_hourL)

4'b0000:qout\_5<=7'b1000000;

4'b0001:qout\_5<=7'b1111001;

4'b0010:qout\_5<=7'b0100100;

4'b0011:qout\_5<=7'b0110000;

4'b0100:qout\_5<=7'b0011001;

4'b0101:qout\_5<=7'b0010010;

4'b0110:qout\_5<=7'b0000010;

4'b0111:qout\_5<=7'b1111000;

4'b1000:qout\_5<=7'b0000000;

4'b1001:qout\_5<=7'b0010000;

default:qout\_5<=7'b1111111;

endcase

end

end

always@(hourH)

begin

if(set\_alarm==0)

begin

case(hourH)

4'b0000:qout\_6<=7'b1000000;

4'b0001:qout\_6<=7'b1111001;

4'b0010:qout\_6<=7'b0100100;

4'b0011:qout\_6<=7'b0110000;

4'b0100:qout\_6<=7'b0011001;

4'b0101:qout\_6<=7'b0010010;

4'b0110:qout\_6<=7'b0000010;

4'b0111:qout\_6<=7'b1111000;

4'b1000:qout\_6<=7'b0000000;

4'b1001:qout\_6<=7'b0010000;

default:qout\_6<=7'b1111111;

endcase

end

else if(set\_alarm==1)

begin

case(alarm\_hourH)

4'b0000:qout\_6<=7'b1000000;

4'b0001:qout\_6<=7'b1111001;

4'b0010:qout\_6<=7'b0100100;

4'b0011:qout\_6<=7'b0110000;

4'b0100:qout\_6<=7'b0011001;

4'b0101:qout\_6<=7'b0010010;

4'b0110:qout\_6<=7'b0000010;

4'b0111:qout\_6<=7'b1111000;

4'b1000:qout\_6<=7'b0000000;

4'b1001:qout\_6<=7'b0010000;

default:qout\_6<=7'b1111111;

endcase

end

end

//模块9:lcd液晶屏显示//

//-----------------lcd显示开始---------------------

//译码开始

reg [7:0]secL\_lcd;

reg [7:0]secH\_lcd;

reg [7:0]minL\_lcd;

reg [7:0]minH\_lcd;

reg [7:0]hourL\_lcd;

reg [7:0]hourH\_lcd;

always@(secL)

begin

begin

case(secL)

4'b0000:secL\_lcd<=8'b00110000;

4'b0001:secL\_lcd<=8'b00110001;

4'b0010:secL\_lcd<=8'b00110010;

4'b0011:secL\_lcd<=8'b00110011;

4'b0100:secL\_lcd<=8'b00110100;

4'b0101:secL\_lcd<=8'b00110101;

4'b0110:secL\_lcd<=8'b00110110;

4'b0111:secL\_lcd<=8'b00110111;

4'b1000:secL\_lcd<=8'b00111000;

4'b1001:secL\_lcd<=8'b00111001;

default:secL\_lcd<=8'b00100000;

endcase

end

end

always@(secH)

begin

begin

case(secH)

4'b0000:secH\_lcd<=8'b00110000;

4'b0001:secH\_lcd<=8'b00110001;

4'b0010:secH\_lcd<=8'b00110010;

4'b0011:secH\_lcd<=8'b00110011;

4'b0100:secH\_lcd<=8'b00110100;

4'b0101:secH\_lcd<=8'b00110101;

4'b0110:secH\_lcd<=8'b00110110;

4'b0111:secH\_lcd<=8'b00110111;

4'b1000:secH\_lcd<=8'b00111000;

4'b1001:secH\_lcd<=8'b00111001;

default:secH\_lcd<=8'b00100000;

endcase

end

end

always@(minL)

begin

begin

case(minL)

4'b0000:minL\_lcd<=8'b00110000;

4'b0001:minL\_lcd<=8'b00110001;

4'b0010:minL\_lcd<=8'b00110010;

4'b0011:minL\_lcd<=8'b00110011;

4'b0100:minL\_lcd<=8'b00110100;

4'b0101:minL\_lcd<=8'b00110101;

4'b0110:minL\_lcd<=8'b00110110;

4'b0111:minL\_lcd<=8'b00110111;

4'b1000:minL\_lcd<=8'b00111000;

4'b1001:minL\_lcd<=8'b00111001;

default:minL\_lcd<=8'b00100000;

endcase

end

end

always@(minH)

begin

begin

case(minH)

4'b0000:minH\_lcd<=8'b00110000;

4'b0001:minH\_lcd<=8'b00110001;

4'b0010:minH\_lcd<=8'b00110010;

4'b0011:minH\_lcd<=8'b00110011;

4'b0100:minH\_lcd<=8'b00110100;

4'b0101:minH\_lcd<=8'b00110101;

4'b0110:minH\_lcd<=8'b00110110;

4'b0111:minH\_lcd<=8'b00110111;

4'b1000:minH\_lcd<=8'b00111000;

4'b1001:minH\_lcd<=8'b00111001;

default:minH\_lcd<=8'b00100000;

endcase

end

end

always@(hourL)

begin

begin

case(hourL)

4'b0000:hourL\_lcd<=8'b00110000;

4'b0001:hourL\_lcd<=8'b00110001;

4'b0010:hourL\_lcd<=8'b00110010;

4'b0011:hourL\_lcd<=8'b00110011;

4'b0100:hourL\_lcd<=8'b00110100;

4'b0101:hourL\_lcd<=8'b00110101;

4'b0110:hourL\_lcd<=8'b00110110;

4'b0111:hourL\_lcd<=8'b00110111;

4'b1000:hourL\_lcd<=8'b00111000;

4'b1001:hourL\_lcd<=8'b00111001;

default:hourL\_lcd<=8'b00100000;

endcase

end

end

always@(hourH)

begin

begin

case(hourH)

4'b0000:hourH\_lcd<=8'b00110000;

4'b0001:hourH\_lcd<=8'b00110001;

4'b0010:hourH\_lcd<=8'b00110010;

4'b0011:hourH\_lcd<=8'b00110011;

4'b0100:hourH\_lcd<=8'b00110100;

4'b0101:hourH\_lcd<=8'b00110101;

4'b0110:hourH\_lcd<=8'b00110110;

4'b0111:hourH\_lcd<=8'b00110111;

4'b1000:hourH\_lcd<=8'b00111000;

4'b1001:hourH\_lcd<=8'b00111001;

default:hourH\_lcd<=8'b00100000;

endcase

end

end

//译码结束

reg LCD\_EN\_Sel;

wire[127:0] data\_row1,data\_row2;

assign LCD\_ON = 1'b1;

//------------------------------------//

//输入时钟50MHz 输出周期2ms

//分频模块

reg [15:0]lcd\_count;

reg clk\_2ms;//2ms输出时钟

always @ (posedge clk)

begin

if(lcd\_count <16'd50\_000)

lcd\_count <= lcd\_count + 1'b1;

else

begin

lcd\_count <= 16'd1;

clk\_2ms <= ~clk\_2ms;

end

end

//---------------------------------------//

reg [127:0] Data\_Buf; //液晶显示的数据缓存

reg [4:0] disp\_count;

reg [3:0] state;

parameter Clear\_Lcd = 4'b0000, //清屏并光标复位

Set\_Disp\_Mode = 4'b0001, //设置显示模式：8位2行5x7点阵

Disp\_On = 4'b0010, //显示器开、光标不显示、光标不允许闪烁

Shift\_Down = 4'b0011, //文字不动，光标自动右移

Write\_Addr = 4'b0100, //写入显示起始地址

Write\_Data\_First = 4'b0101, //写入第一行显示的数据

Write\_Data\_Second = 4'b0110; //写入第二行显示的数据

assign RW = 1'b0; //RW=0时对LCD模块执行写操作(一直保持写状态）

assign LCD\_EN = LCD\_EN\_Sel ? clk\_2ms : 1'b0;//通过LCD\_EN\_Sel信号来控制LCD\_EN的开启与关闭

assign data\_row1="YANGTIMING-TIME:";

assign data\_row2 = {{4{8'b00100000}},hourH\_lcd,hourL\_lcd,{8'b00111010},minH\_lcd,minL\_lcd,{8'b00111010},secH\_lcd,secL\_lcd,{4{8'b00100000}}};//"####HH:MM:SS####"

always @(posedge clk\_2ms or posedge rst)

begin

if(rst)

begin

state <= Clear\_Lcd; //复位：清屏并光标复位

RS <= 1'b1; //复位：RS=1时为读指令；

DB8 <= 8'b0; //复位：使DB8总线输出全0

LCD\_EN\_Sel <= 1'b0; //复位：关液晶使能信号

disp\_count <= 5'b0;

//---------下面是测试数据------------------------//

end

else

begin

case(state) //初始化LCD模块

Clear\_Lcd:

begin

LCD\_EN\_Sel <= 1'b1; //开使能

RS <= 1'b0; //写指令

DB8 <= 8'b0000\_0001; //清屏并光标复位

state <= Set\_Disp\_Mode;

end

Set\_Disp\_Mode:

begin

DB8 <= 8'b0011\_1000; //设置显示模式：8位2行5x8点阵

state <= Disp\_On;

end

Disp\_On:

begin

DB8 <= 8'b0000\_1100; //显示器开、光标不显示、光标不允许闪烁

state <= Shift\_Down;

end

Shift\_Down:

begin

DB8 <= 8'b0000\_0110; //文字不动，光标自动右移

state <= Write\_Addr;

end

//---------------------------------显示循环------------------------------------//

Write\_Addr:

begin

RS <= 1'b0;//写指令

DB8 <= 8'b1000\_0000; //写入第一行显示起始地址：第一行第1个位置

Data\_Buf <= data\_row1; //将第一行显示的数据赋给Data\_First\_Buf

state <= Write\_Data\_First;

end

Write\_Data\_First: //写第一行数据

begin

if(disp\_count == 5'd16) //disp\_count等于15时表示第一行数据已写完

begin

RS <= 1'b0;//写指令

DB8 <= 8'b1100\_0000; //送入写第二行的指令,第2行第1个位置

disp\_count <= 5'b00000; //计数清0

Data\_Buf <= data\_row2;//将第2行显示的数据赋给Data\_First\_Buf

state <= Write\_Data\_Second; //写完第一行进入写第二行状态

end

else//没写够16字节

begin

RS <= 1'b1; //RS=1表示写数据

DB8 <= Data\_Buf[127:120];

Data\_Buf <= (Data\_Buf << 8);

disp\_count <= disp\_count + 1'b1;

state <= Write\_Data\_First;

end

end

Write\_Data\_Second: //写第二行数据

begin

if(disp\_count == 5'd16)//数据写完了

begin

RS <= 1'b0;//写指令

DB8 <= 8'b1000\_0000; //写入第一行显示起始地址：第一行第1个位置

disp\_count <= 5'b00000;

state <= Write\_Addr; //重新循环

end

else//

begin

RS <= 1'b1;

DB8 <= Data\_Buf[127:120];

Data\_Buf <= (Data\_Buf << 8);

disp\_count <= disp\_count + 1'b1;

state <= Write\_Data\_Second;

end

end

//--------------------------------------------------------------------------//

default: state <= Clear\_Lcd; //若state为其他值，则将state置为Clear\_Lcd

endcase

end

end

endmodule

5 结论以及结果说明

系统运行的软硬件环境：quartus II6.0

计算机平台： wind xp

仿真图：

1.自动计时模块

图片包含 日程表

描述已自动生成

分析：

如图所示，由于周期短，无法呈现00-24时所有的内容，但是由图可知：secH\_1和secL\_1满足要求。

secL\_1在00-09之间逐渐每个一个clock 就变化一次，最大为1001为9.

当secL\_1重置为0时，进位一次，secH\_1由0000变成0001。其他也同样满足时钟规律，故满足要求。

2.整点报时模块

图片包含 图示

描述已自动生成

分析：

如图很容易知道，blink 随着sec的改变，每隔两秒就改变一次。

3.倒计时模块

图片包含 图示

描述已自动生成

分析：

由图可知，count随着clk\_div的改变由9逐渐变成了0.然后依次往复循环。

总结：数字钟采用verilog程序设计语言，并能满足所要求的所有基本内容。能在DE2开发版上跑起来，完成了本次数电课设的实验。