中山大学数据科学与计算机学院本科生实验报告

（**2017**学年春季学期）

课程名称：**数字电路与逻辑设计实验**　任课教师：**保延翔**　助教：**岳锐**

年级**&**班级	**16**级计科九班	专业**(**方向**)**	计算机类
学号	**15323032**	姓名	李新锐
电话	**18142820920**	Email	**lixr26@mail2.sysu.edu.cn**
开始日期	**2017/6/12**	完成日期	**2017/6/24**

一、实验题目

分别在Proteus和Basys3开发板上完成一个数字时钟设计

二、实验目的

综合演练数字系统的设计方法，提高设计数字系统的能力。

三、实验内容

第一部分：在**Proteus**上完成数字时钟设计

1.思路和模块划分

设计这个较为复杂的数字系统时，我的思路是：采用模块化的设计，分开各个功能首先要考虑顶层设计，然后自顶而下完成各项功能。

如图，是最后总体的电路图，已经添加了注释（是矢量图，可以放大查看文字）。

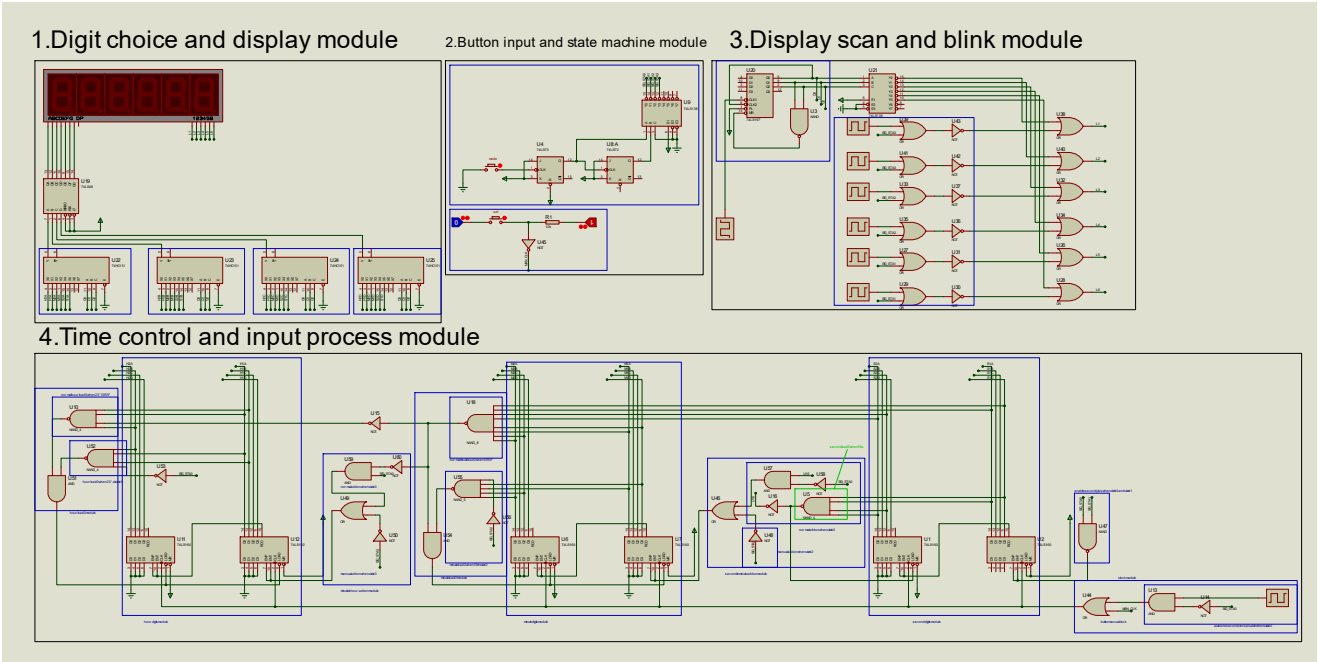
共分为4个模块，用粗体字说明了它们的在模块间的输入输出（通过Proteus的标签功能进行连接，而不是直接连线）

1：数字选择和显示模块，接收的输入有**时分秒数字输入**和**数码管位选择信号**和**对应的时分秒数字选择信号**，通过数码管**输出时间**

2：处理按钮输入和状态机模块，接收的输入有**按钮输入**，输出**状态机状态值**和**手动调整时间脉冲**

3：扫描显示六位数字，产生闪烁效果模块，接收**状态机状态值**输入，输出不同的**数码管位选择信号**和**对应的时分秒数字选择信号**

4：计时和根据输入调整时间模块，接收输入有**状态机状态值**和**手动调整时间脉冲**，输出**时分秒数字**



下面讲依次解释各个模块的具体实现

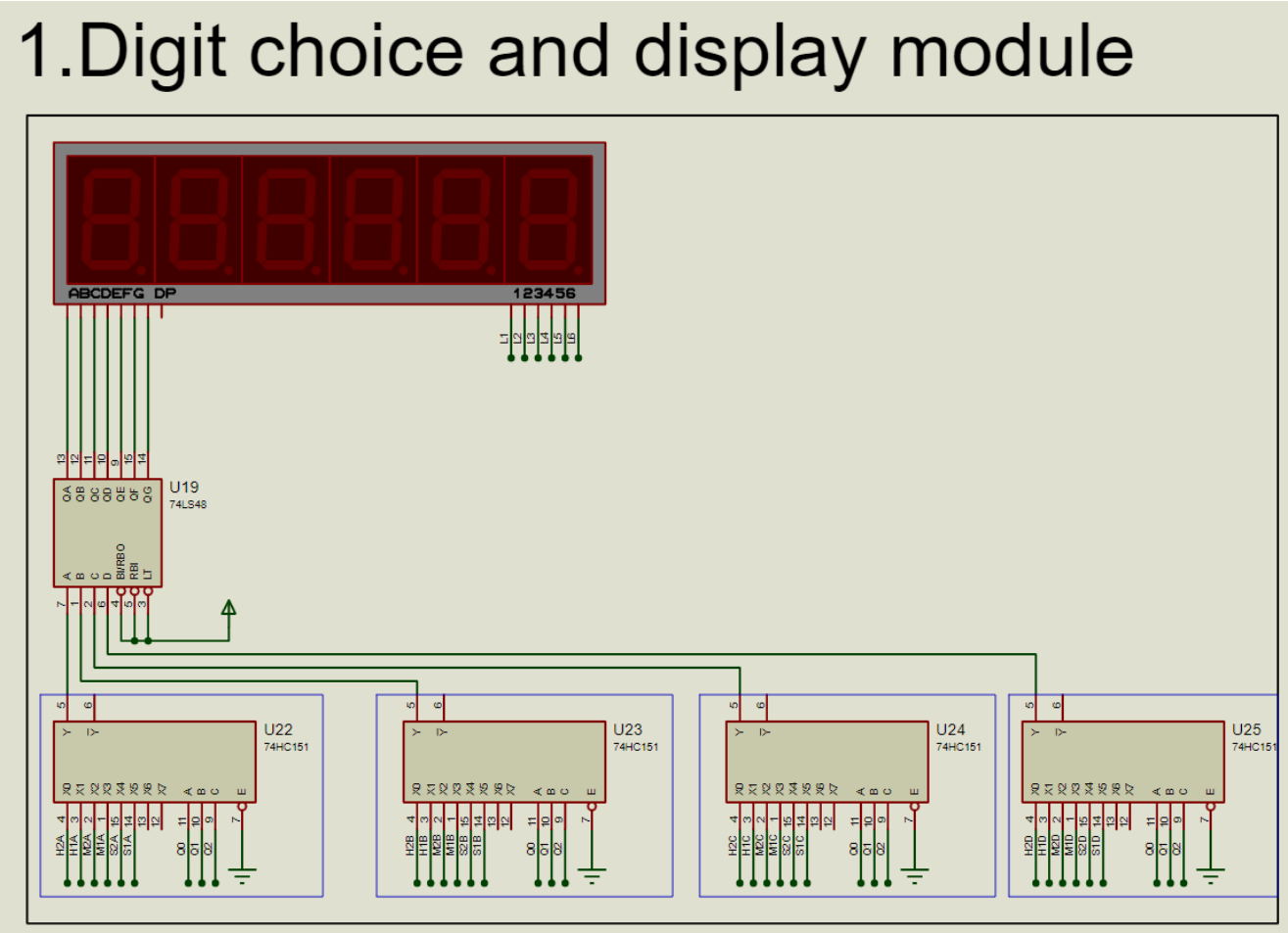
2.模块1

使用元件：

	7SEG-MPX6-CC	7段数码显示管
	74LS48	BCD转7段数码管段选信号
	74HC151	8选1数据选择器

蓝框中的4个74HC151的X0-X5输入分别连接时分秒的十位/个位数字的最高位到最低位。ABC接模块3的Q0/Q1/Q2输出，依次选择这六个数字。4个74HC151的Y输出，代表着这一时刻选择的数字的最高位到最低位，分别连接到74LS48的A-D输入。以上操作做到了数字选择。

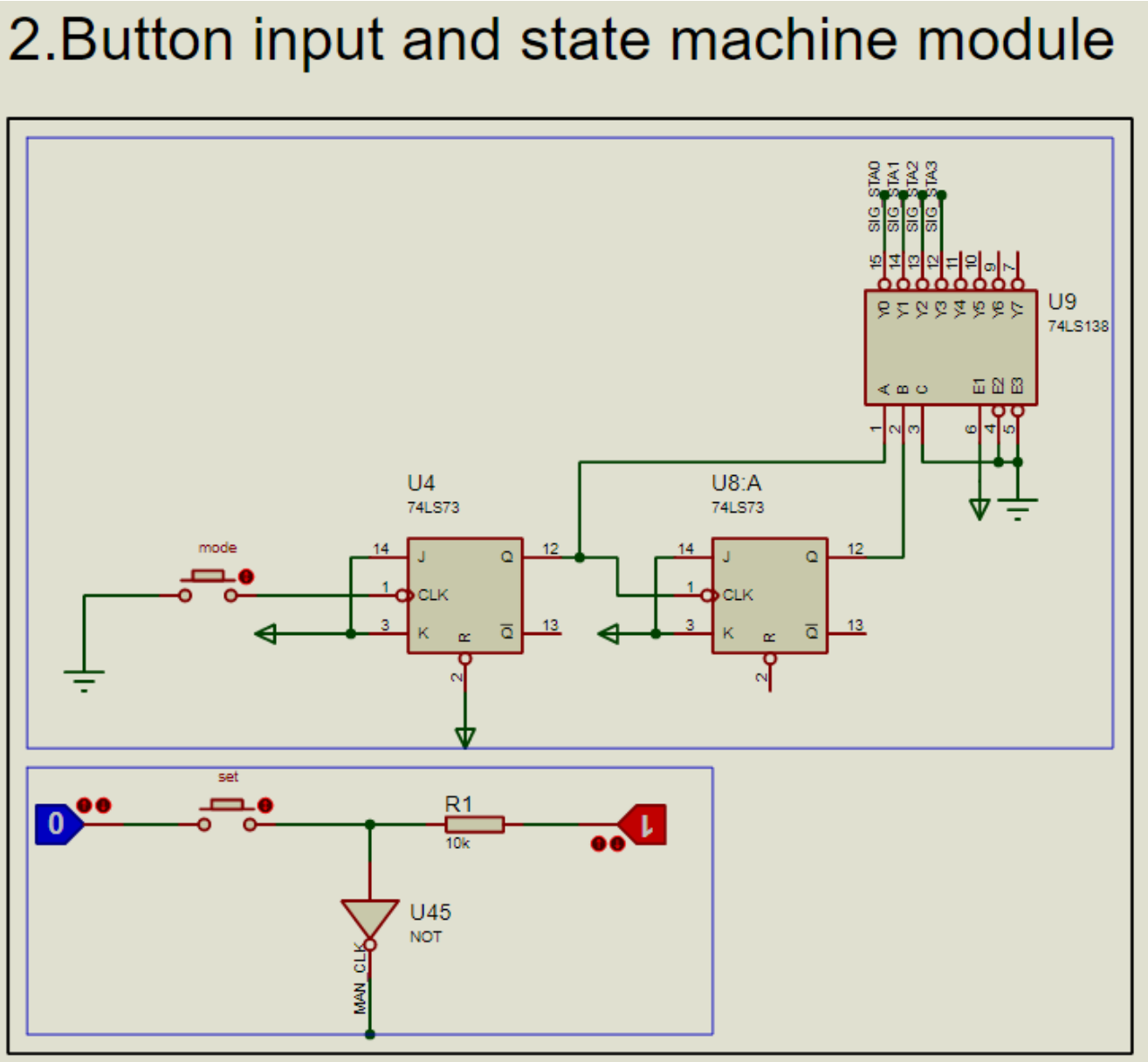
7段数码显示管的位选端口连接到模块3的最右端的6个与门的输出。做到了位选。



3.模块2

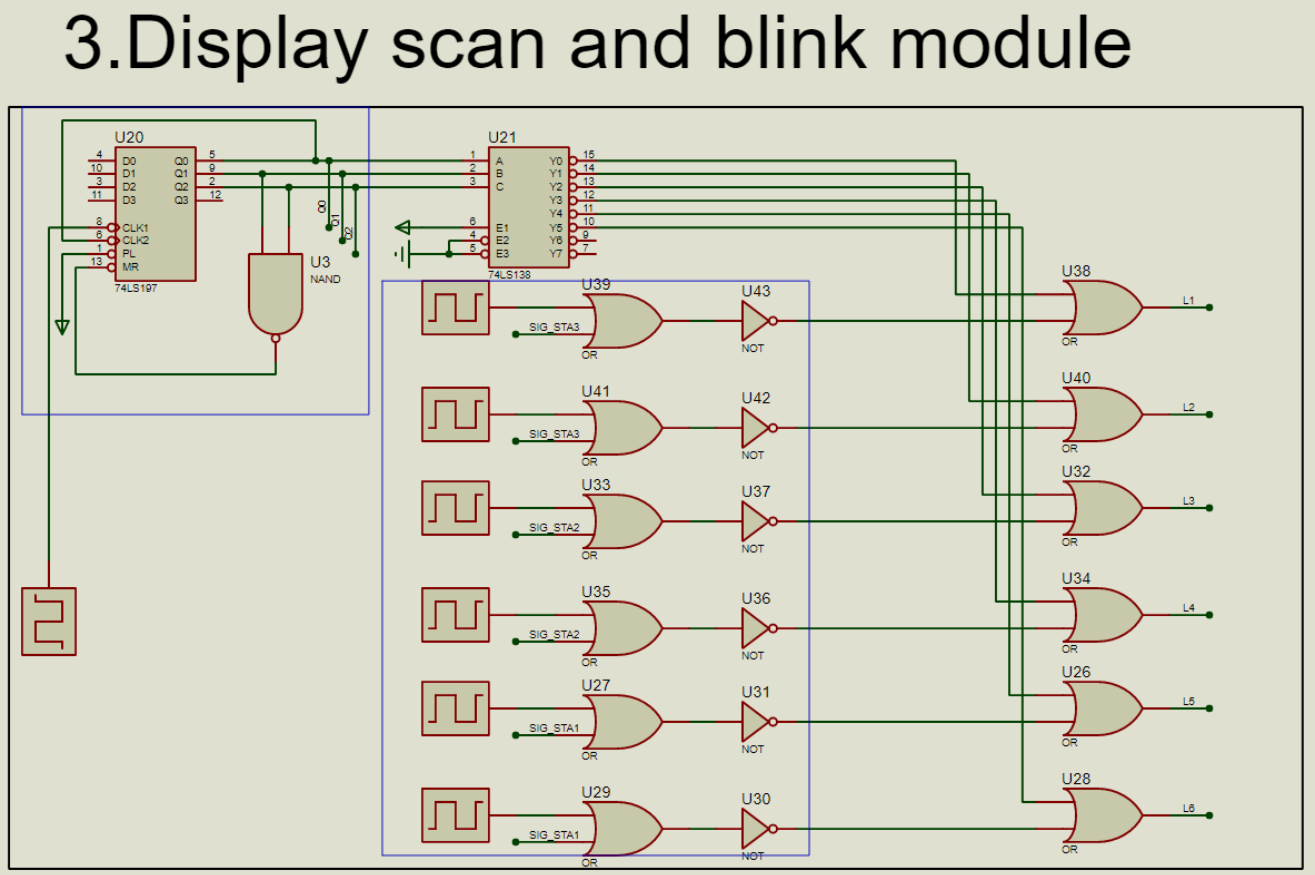
上面的蓝框mode按钮连接了两个JK触发器组成的状态储存电路，再连接到74LS138，输出了4个状态

下面的蓝框中的set按钮一端接地，一段接上拉电阻接高电平，引出一条线MAN_CLK连接到模块4，作为调节时间的信号



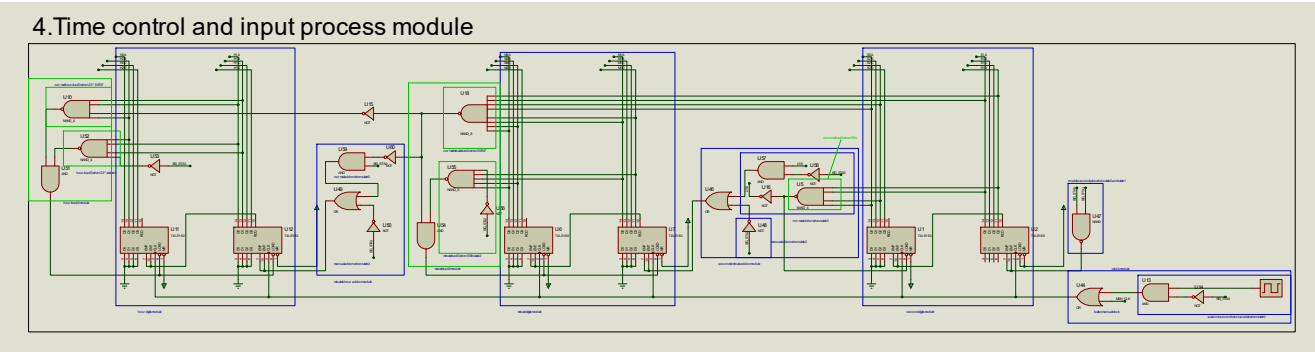
4.模块3

左上的蓝框中的74LS197输入的是1000hz的时钟信号，产生0-5的选择信号，输出到74LS138中。中间的蓝框是产生闪烁效果的模块，时钟是2hz的，与SIG_STA(x)连接到与门，当处于状态x为1-3时，SIG_STA(x)变为低电平，此时2hz的时钟信号就会发挥作用，使得L(2x), L(2x-1)两个门每秒有一半时间处于高电平，通过与门与138的输出连接在一起，就使显示管对应的时分秒每1s有0.5s处于灭灯状态，产生了闪烁效果

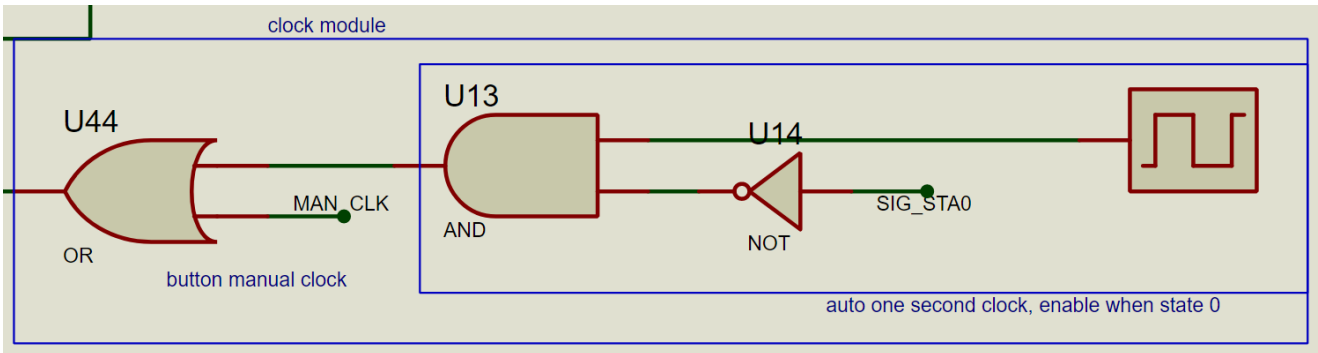


5.模块4

该模块较大，但可以分为时/分/秒/时钟4个小模块进行分析

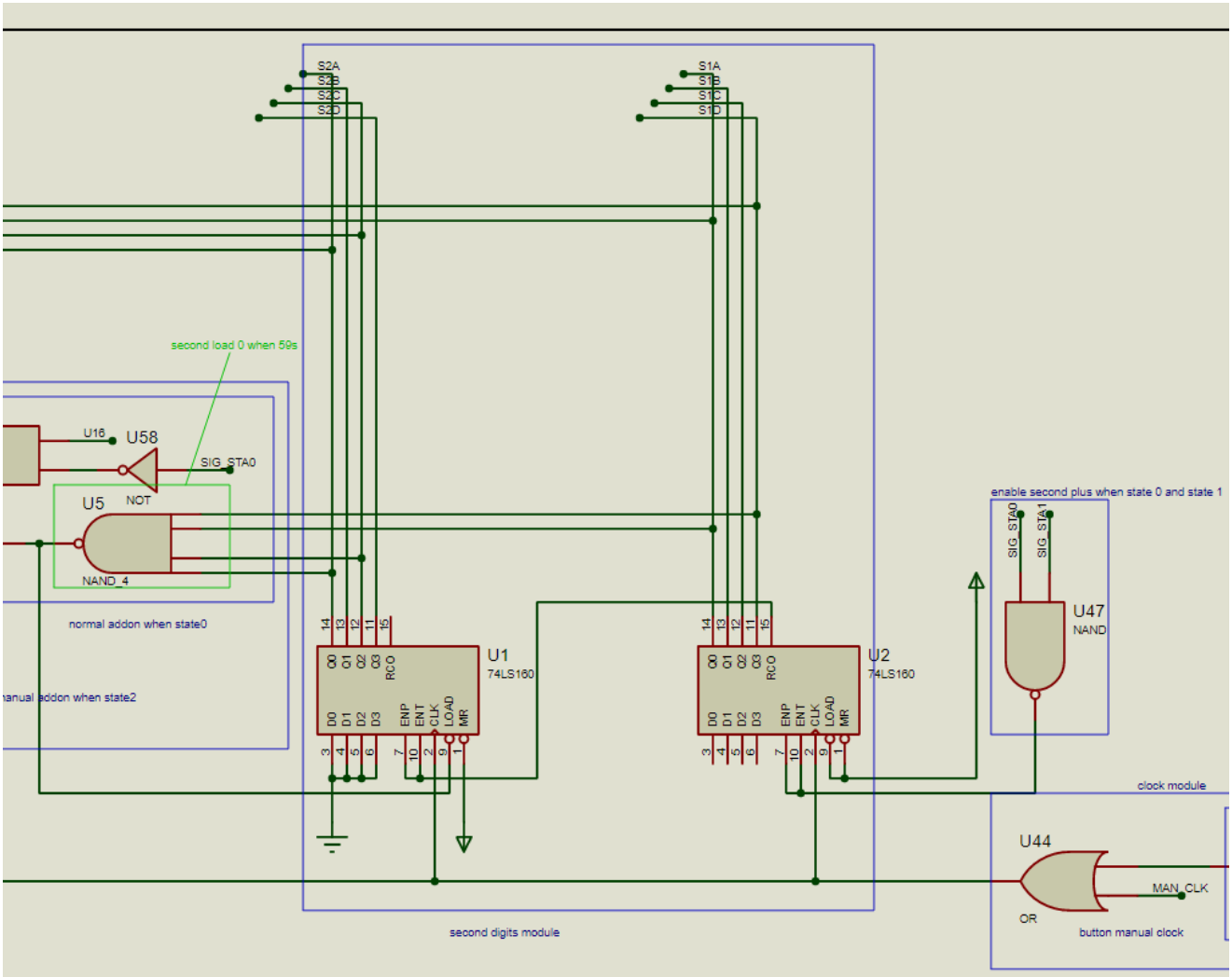


右下角蓝框是时钟模块，状态0时由时钟输入1s的时钟信号。其他状态下时钟被禁用，当按下调整时间按钮时会手动产生相当于时钟脉冲的信号，时对应调整的数字加一



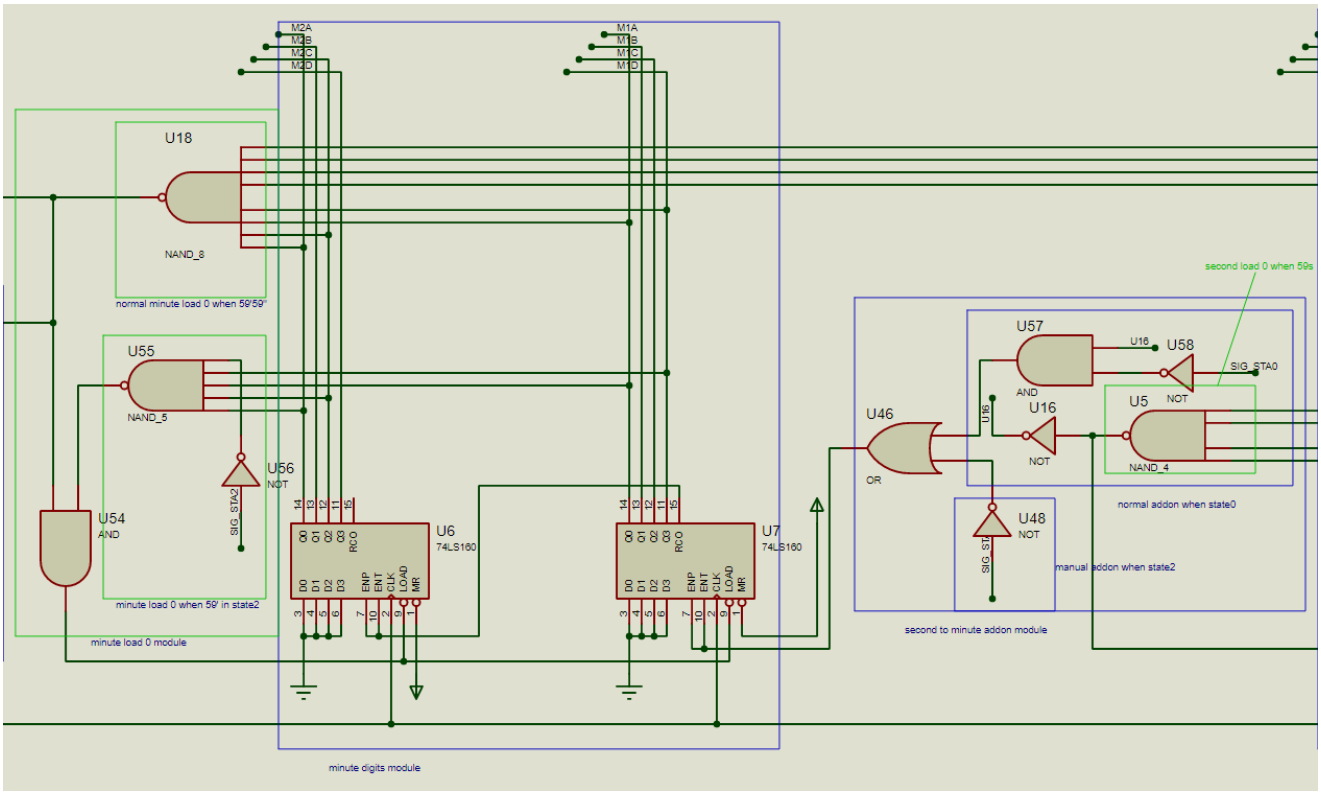
时分秒模块都要处理何时模块有效（有效即在接收到时候信号时增加时间），何时清零两个问题

下图是秒钟数字模块，右侧蓝框中的与非门实现了在状态0和1时有效，左侧绿框中的4入与非门实现了在59秒时产生置位信号，因为74LS160是同步置位，所以会在下个时钟到来时置位。而D0-D3接地，因而会清零。

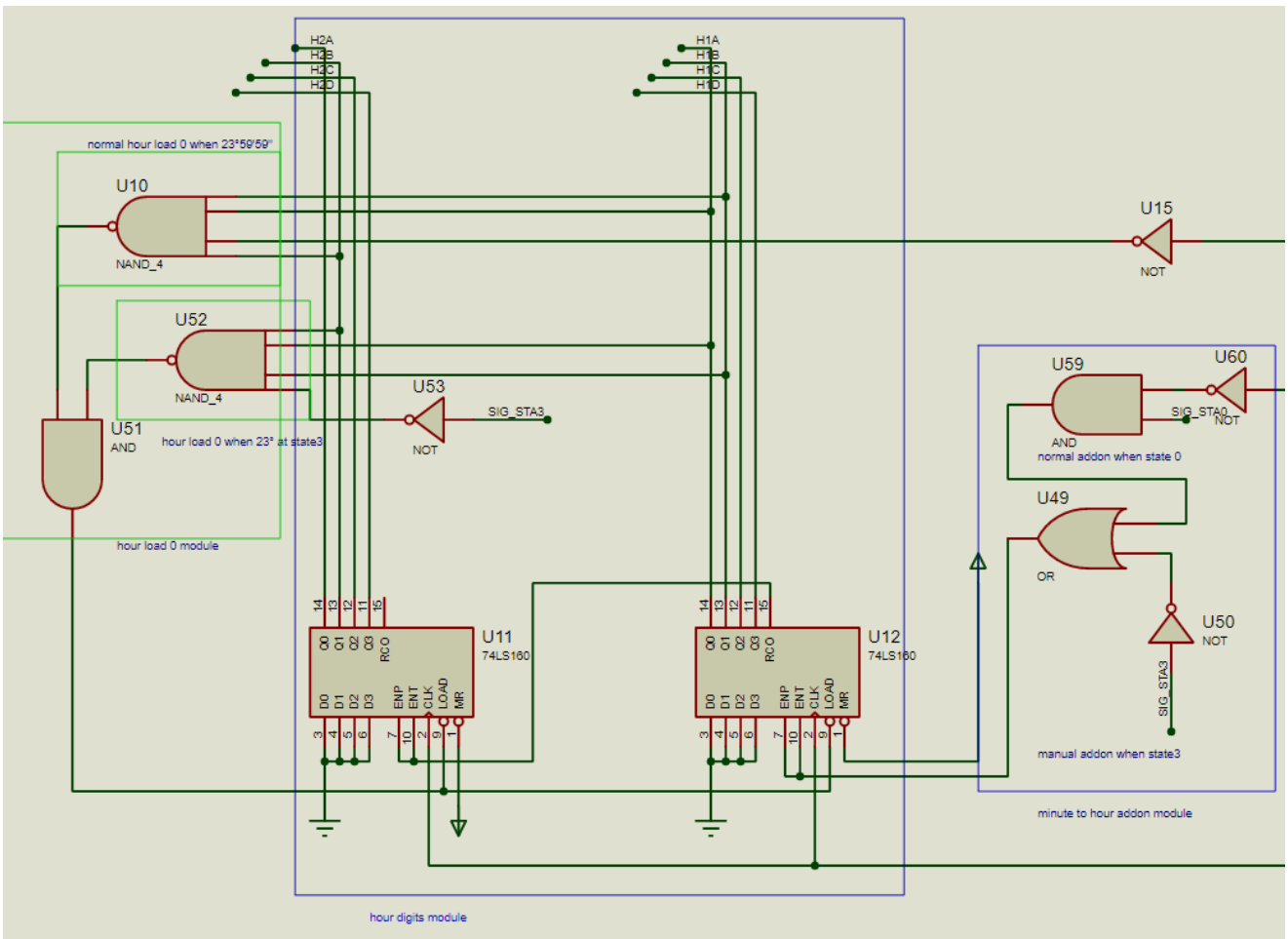


下图是分钟处理模块，右侧蓝框中实现了在状态0时，会接收在秒钟为59秒时的信号，使分钟模块有效，进位加一；或在状态1时始终有效（按下按钮即会加一）。

与秒钟模块类似。左侧绿框中实现了与门连接着一个8入与非门和一个4入与非门。实现了在59分59秒后清零和在状态2时在59分后清零。

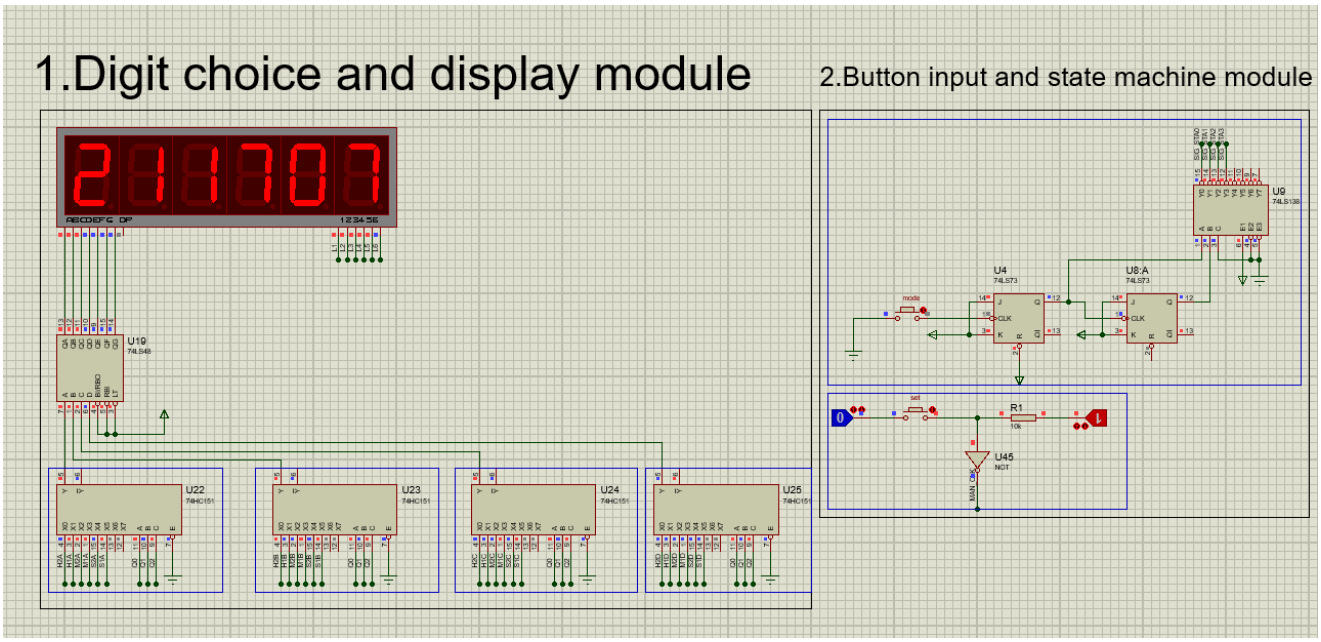


下图是时钟处理模块，与上面两个模块也类似。在状态0时，每59分59秒加一，或在状态3时每次按下按键加一。在23时59分59秒后清零或在状态3时在23时后清零。



6.实验结果

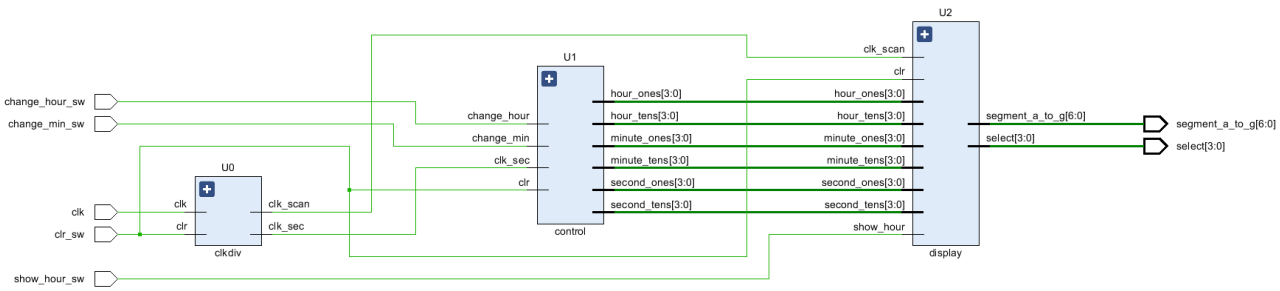
经测试各种功能正常。如图，时钟正在显示现在的时间（21点17分07秒）。



第二部分： 使用Verilog语言在Basys3开发板上实现时钟

1.顶层模块

如图，顶层模块的输入是100Mhz的时钟clk，清零开关clr_sw，调整时钟开关change_hour_sw，调整分钟开关change_min_sw，切换显示时-分和显示分-秒状态的开关show_hour_sw。输出是7段数码管的位选信号select[3:0]和段选segment_a_to_g[7:0]信号。



源代码如下，在注释中说明了其原理功能(由于使用vim写代码，如果输入中文注释得来回切换中英文，十分麻烦，所以写的是英文注释)

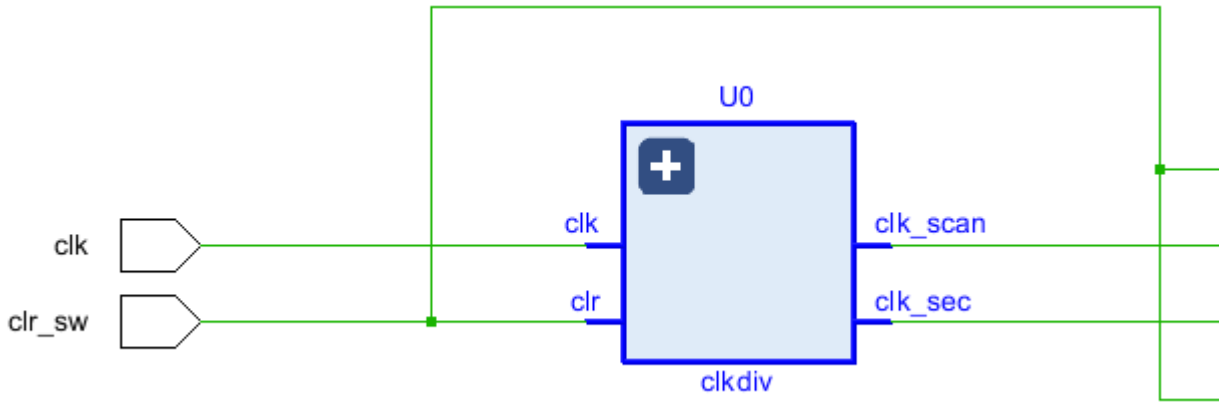
```
`timescale 1ns / 1ps


module top(
    input wire clk,                     //100Mhz clock
    input wire clr_sw,                  //clear switch
    input wire change_hour_sw,          //change hour switch(add one in hour)
```

```
input wire change_min_sw,          //change minunte switch(add one in minute)
input wire show_hour_sw,           //show hour-minute/minute-second switch
output wire [3:0] select,          //4x7seg display bit select
output wire [6:0] segment_a_to_g);//7seg display segment select
wire clr;                  //wire connected to clear switch
wire clk_sec;               //wire for connecting of one second clock
wire clk_scan;          //wire for connecting of 7seg display scan clock
wire show_hour;      //wire connected to show hour switch
wire change_min;    //wire connected to change minute switch
wire change_hour;   //wire connected to change hour switch
wire [3:0] second_ones;  //wire for connecting of one's digit of second
wire [3:0] second_tens;   //wire for connecting of ten's digit of second
wire [3:0] minute_ones;  //wire for connecting of one's digit of minute
wire [3:0] minute_tens;   //wire for connecting of ten's digit of minute
wire [3:0] hour_ones;       //wire for connecting of one's digit of hour
wire [3:0] hour_tens;       //wire for connecting of ten's digit of hour
assign show_hour = show_hour_sw;          //do wire connecting
assign clr = clr_sw;                      //do wire connecting
assign change_min = change_min_sw;        //do wire connecting
assign change_hour = change_hour_sw;     //do wire connecting
//MODULE 1
//clock divide module
//producting 1s clock and scan clock
clkdiv U0(.clk(clk),
        .clr(clr),
        .clk_scan(clk_scan),
        .clk_sec(clk_sec));

//MODULE 2
//time control module
//calculate current time and adjusting time
control U1(.clk_sec(clk_sec),
```

```
            .clr(clr),
            .change_hour(change_hour),
            .change_min(change_min),
            .second_ones(second_ones),
            .second_tens(second_tens),
            .minute_ones(minute_ones),
            .minute_tens(minute_tens),
            .hour_ones(hour_ones),
            .hour_tens(hour_tens));

    //MODULE 3
    //4x7seg display control module
    //display the time
    display  U2(.second_ones(second_ones),
             .second_tens(second_tens),
             .minute_ones(minute_ones),
             .minute_tens(minute_tens),
             .hour_ones(hour_ones),
             .hour_tens(hour_tens),
             .clk_scan(clk_scan),
             .clr(clr),
             .show_hour(show_hour),
             .segment_ato_g(segment_ato_g),
             .select(select));
endmodule
```

2.clkdiv

输入是100Mhz的时钟clk，清零开关clr_sw，产生周期为1ms的扫描数码管信号和1s的时钟驱动信号clk_sec



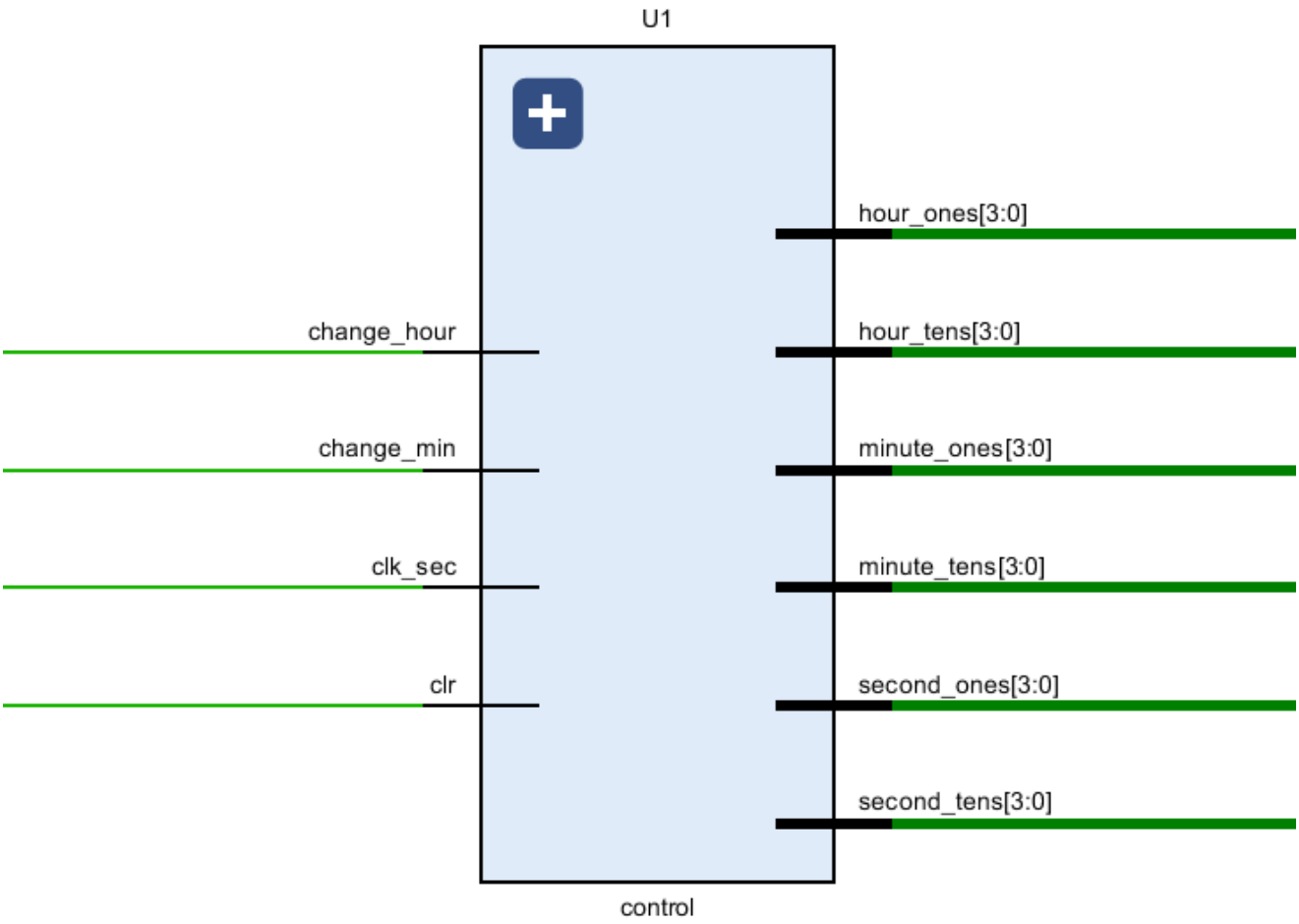
源代码如下，在注释中说明了其原理功能

```
module clkdiv (
    input clk,
    input clr,
    output reg clk_scan,
    output reg clk_sec);
    reg [27:0] counter_1s;
    reg [16:0] counter_1ms;
    always @ (posedge clk or posedge clr) //At 100Mhz clock posedge and asynchronous clear
    begin
        if (clr == 1)                 //clear counter
            counter_1s = 0;
        else if (counter_1s == 49999999) //every 0.5s clk_sec overturns, producting 1s clock
        begin
            counter_1s = 0;
            clk_sec = ~clk_sec;
        end
        else if (counter_1ms == 49999)  //every 0.5ms clk_scan overturns, producting 1ms clock
        begin
            counter_1ms = 0;
            clk_scan = ~clk_scan;
        end
        else
        begin                         //counter auto plus
```

```
            counter1s <= counter1s + 1;
            counter1ms <= counter1ms + 1;
        end
    end
endmodule
```

3.control

输入周期1s的时钟信号clk_sec，清零信号clr，调整时钟的信号change_hour，调整分钟的信号change_min，切换显示时-分和显示分-秒状态的信号show_hour。输出时分秒个位/十位 6个数字。



源代码如下，在注释中说明了其原理功能

```
`timescale 1ns / 1ps
module control(
    input wire clk_sec,
    input wire clr,
    input wire change_hour,
```

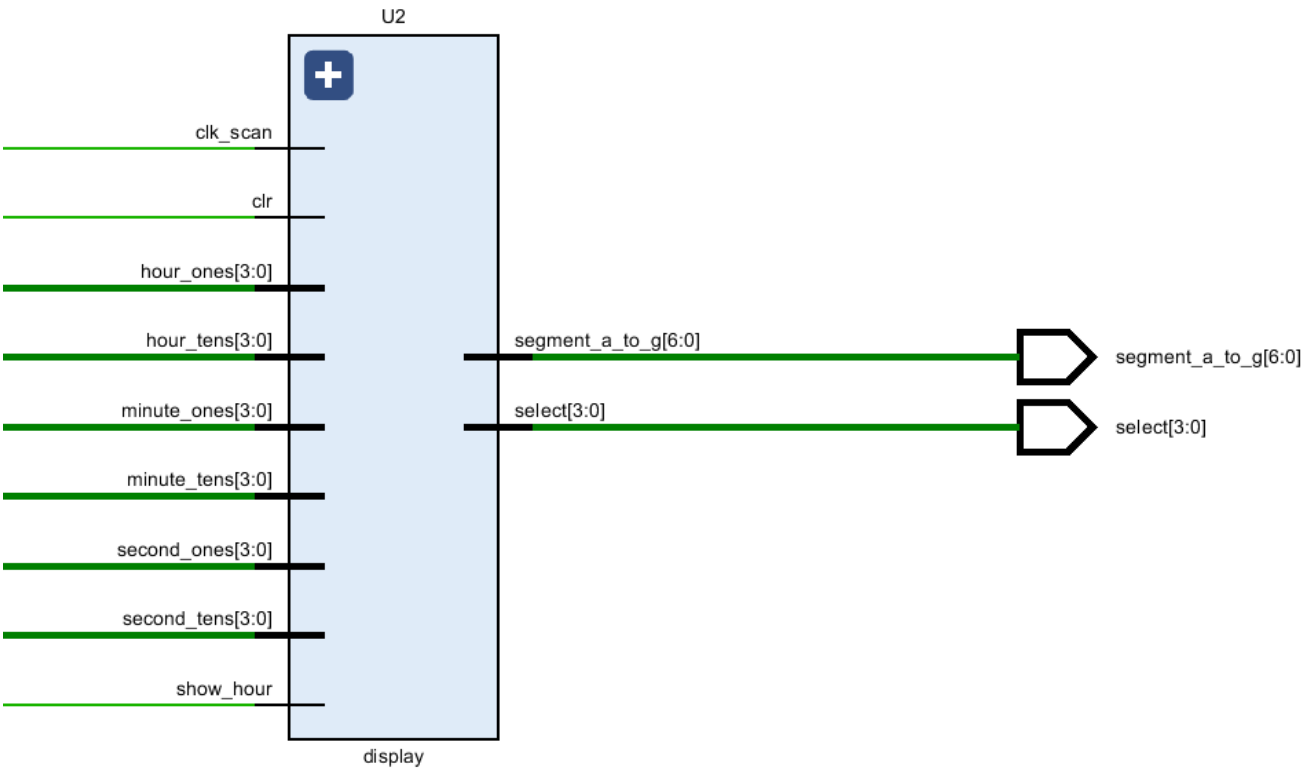
```
    input wire change_min,
    output reg [3:0] second_ones,
    output reg [3:0] second_tens,
    output reg [3:0] minute_ones,
    output reg [3:0] minute_tens,
    output reg [3:0] hour_ones,
    output reg [3:0] hour_tens
    );
always @(posedge clk_sec or posedge clr)  //At 1s clock and asynchronous clear
    begin
    if(clr==1)                          //clear
    begin
       second_ones<=4'b0000;
       second_tens<=4'b0000;
       minute_ones<=4'b0000;
       minute_tens<=4'b0000;
       hour_ones<=4'b0000;
       hour_tens<=4'b0000;
    end
    else
    begin
       if(!change_hour && !change_min) //Normal mode
       begin
          second_ones=second_ones+1;  //Adds second
          if(second_ones==4'd10)      //Second one's digit addon
          begin
             second_tens=second_tens+1;
             second_ones=4'b0000;
          end
          if(second_tens==4'd6)      //Second ten's digit addon
          begin
             minute_ones=minute_ones+1;
```

```
        second_tens=4'b0000;
    end
    if(minute_ones==4'd10)    //Minute one's digit addon
    begin
        minute_tens=minute_tens+1;
        minute_ones=4'b0000;
    end
    if(minute_tens==4'd6)    //Minute ten's digit addon
    begin
        hour_ones=hour_ones+1;
        minute_tens=4'b0000;
    end
    if(hour_ones==4'd10)    //Hour one's digit addon
    begin
        hour_tens=hour_tens+1;
        hour_ones=4'b0000;
    end
    if(hour_tens==4'd2&&hour_ones==4'd4) //24 hour clear
    begin
        hour_ones=4'b0000;
        hour_tens=4'b0000;
    end
end
else // Change time mode
begin
    if(change_hour)                //Change hour mode, add hour by one
        hour_ones=hour_ones+1;
    if(change_min)
        minute_ones=minute_ones+1;  //Change minute mode, add minute by one
    if(minute_ones==4'd10)
    begin
      minute_tens=minute_tens+1;   //Minute one's digit addon
```

```
            minute_ones=4'b0000;
        end
        if(minute_tens==4'd6)          //Now minute ten's digit addon is disabled
        begin
            // hour_ones=hour_ones+1;
            minute_tens=4'b0000;
        end
        if(hour_ones==4'd10)          //Hour one's digit addon
        begin
             hour_tens=hour_tens+1;
             hour_ones=4'b0000;
        end
        if(hour_tens==4'd2&&hour_ones==4'd4) //24 hour clear
        begin
            hour_ones=4'b0000;
            hour_tens=4'b0000;
        end
      end
   end
end
endmodule
```

4.display

输入周期1ms的时钟信号clk_scan，清零信号clr，时分秒个位/十位 6个数字，切换显示时-分和显示分-秒状态的信号show_hour。输出7段数码管的位选信号select[3:0]和段选segment_a_to_g[7:0]信号。



源代码如下，在注释中说明了其原理功能

```
`timescale 1ns / 1ps
module display  (input wire [3:0] second_ones,
                 input wire [3:0] second_tens,
                 input wire [3:0] minute_ones,
                 input wire [3:0] minute_tens,
                 input wire [3:0] hour_ones,
                 input wire [3:0] hour_tens,
                 input wire clk_scan,
                 input wire clr,
                 input wire show_hour,
                 output reg [6:0] segment_a_to_g,          //7-seg display segment select output
                 output reg [3:0] select                   //7-seg display bit select output
    );
    reg [1:0] select_bit;//select which bit to display
    reg [3:0] digit;     //the hex digit to display
```

```
    always @(*)         //At select bit change
    case (select_bit)
        0 : digit = show_hour ? minute_ones : second_ones; //determine what to display in each bit, according to show hour switch
        1 : digit = show_hour ? minute_tens : second_tens; //determine what to display in each bit, according to show hour switch
        2 : digit = show_hour ? hour_ones : minute_ones;   //determine what to display in each bit, according to show hour switch
        3 : digit = show_hour ? hour_tens : minute_tens;   //determine what to display in each bit, according to show hour switch
    endcase
    always @(*)         //At the hex digit to display change
    case (digit)
        0 : segment_a_to_g = 7'b0000001;                //convert hex digit to 7-seg display segment select signal
        1 : segment_a_to_g = 7'b1001111;                //convert hex digit to 7-seg display segment select signal
        2 : segment_a_to_g = 7'b0010010;                //convert hex digit to 7-seg display segment select signal
        3 : segment_a_to_g = 7'b0000110;                //convert hex digit to 7-seg display segment select signal
        4 : segment_a_to_g = 7'b1001100;                //convert hex digit to 7-seg display segment select signal
        5 : segment_a_to_g = 7'b0100100;                //convert hex digit to 7-seg display segment select signal
        6 : segment_a_to_g = 7'b0100000;                //convert hex digit to 7-seg display segment select signal
        7 : segment_a_to_g = 7'b0001111;                //convert hex digit to 7-seg display segment select signal
        8 : segment_a_to_g = 7'b0000000;                //convert hex digit to 7-seg display segment select signal
        9 : segment_a_to_g = 7'b0000100;                //convert hex digit to 7-seg display segment select signal
    default:
        segment_a_to_g = 7'b0000001;                    //default: display off
    endcase
    always @(posedge clk_scan or posedge clr)           //At scan clock or asynchronous clear
    begin
        if (clr == 1)                                   //clear
        begin
            select_bit <= 0;
        end
        else                                            //scan display, each time light one display
```

```
        select_bit <= select_bit + 1;
    end
    always @(*)                         //do scan dispaly
    begin
        select = 4'b1111;
        if (!clr)
            select[select_bit] = 0;
    end
endmodule
```

五、实验感想

1.我将此前在程序设计课程中学到的软件设计的模块化思想应用到硬件设计中，自顶而下考虑问题，首先设计接口，再考虑实现，这种思路对我本次数字时钟的设计起到了很大帮助。说明了模块化思想在任何系统设计中的重要性。

2.在时序电路设计中，分清异步和同步信号是很重要的。例如本实验使用到了74LS160的置数功能设计60进制计数器，此时要注意到它是同步置数，因此要在输出为59时译出置数信号，在下个时钟信号到来时才将0置入计数器中。反之如果要使用它的异步清零功能实现这个计数器，就要在输出为60时译出清零信号，60的状态将极短的出现，然后清零。

又如在使用Verilog设计时钟时，要实现异步清零功能，就要把它放在always语句的括号中。否则，实验表明清零将会无效。

3.在使用Verilog设计时钟时，要时刻练习硬件实际，想像所写语句对应的电路是否可以实现，例如在设计用来连接各个模块的变量时，在实际电路中应该使用普通的电线，因此应该设为wire类型。RTL Analysis功能能很好地帮助检查连线是否正确。在启动漫长的Implementation操作之前最好使用这一工具进行检查，避免浪费时间。