華中科技大學

课程实验报告

VERILOG HDL 语言设计实现数字钟

院　　系	材料科学与工程学院
专业班级	电子封装 1801
姓　　名	肖玉奇
学　　号	U201810991

2020 年 11 月 10 日

目　　录

1 实验目的

1.掌握可编程逻辑器件的应用开发技术——设计输入、编译、仿真和器件编程；

2.熟悉 EDA 软件使用；

3.掌握 VERILOG HDL 设计方法；

4.分模块、分层次数字系统设计。

5.使用 FPGA 板实现多功能数字钟。

通过此次实验，我们将在实践中验证理论知识，不仅是为了巩固课堂上所学知识，更是为了加深我们对 EDA 技术和 VHDL 语言的理解;为了让我们自己动手完成从设计输入、逻辑综合、功能仿真、设计实现到实现编程、时序仿真，直到器件的下载测试的整个过程，真切感受利用 EDA 技术对 FPGA 进行设计开发的过程，锻炼和提高我们对器件的编程调试能力。

2 实验要求

2.1 基本功能

（1）能显示小时、分钟、秒钟（小时以 24 进制,时、分用显示器，秒用 LED）；

（2）能调整小时、分钟的时间；

（3）复位。

2.2 选做功能

（1）任意闹钟；

（2）小时为 12/24 进制可切换；

（3）报正点数（几点钟 LED 闪烁几下）；

（4）仿电台报时。

3 实验原理

3.1 设计方法

课程设计要求我们设计的数字电子钟具有显示时分秒的功能，且能根据set(设置)按钮进行相应的切换显示很容易想到数码管显示模块，即对 set 脉冲进行计数，然后根据计得的数值进行相应的操作。

按照设计内容和要求以及所有的设计思路，综合考虑后，采用元件例化和进程相结合的方法，设计模块化的结构:顶层设计实体为 clock (时钟)模块，其下又分为:秒脉冲、时分秒、闹钟、报时、数码管显示等模块。每个模块主要使用 VHDL 语言输入中常用的进程语句、元件例化语句、case 语句、if 语句以及赋值语句。

3.2 基础知识

1.层次化，模块化的设计方法

对于一个复杂的数字系统, 运用层次化设计方法, 使设计课题进一步细化, 分块设计, 条理清晰。另外, 在调试时可采用逆向调试方式, 即从模块调试向总体调试方向开展调试工作, 使设计中出现的问题在模块级就能发现, 及时处理, 这样就会使一个复杂的设计变得容易调试, 缩短了设计时间。

2. 自顶向下---从系统级开始把系统划分为基本单元，然后再把每个基本单元划分为下一层次的基本单元，一直这样做下去直到可以直接用 元件库中的元件来实现为止。

3. 自下而上---是一种传统的设计方法，从存在的基本单元出发，设计树最末枝上的单元要么是已经制造出的单元要么是其他项目已开发好的单元或者是可外购得到的单元，逐级叠加，逐模块叠加，直至实现功能。

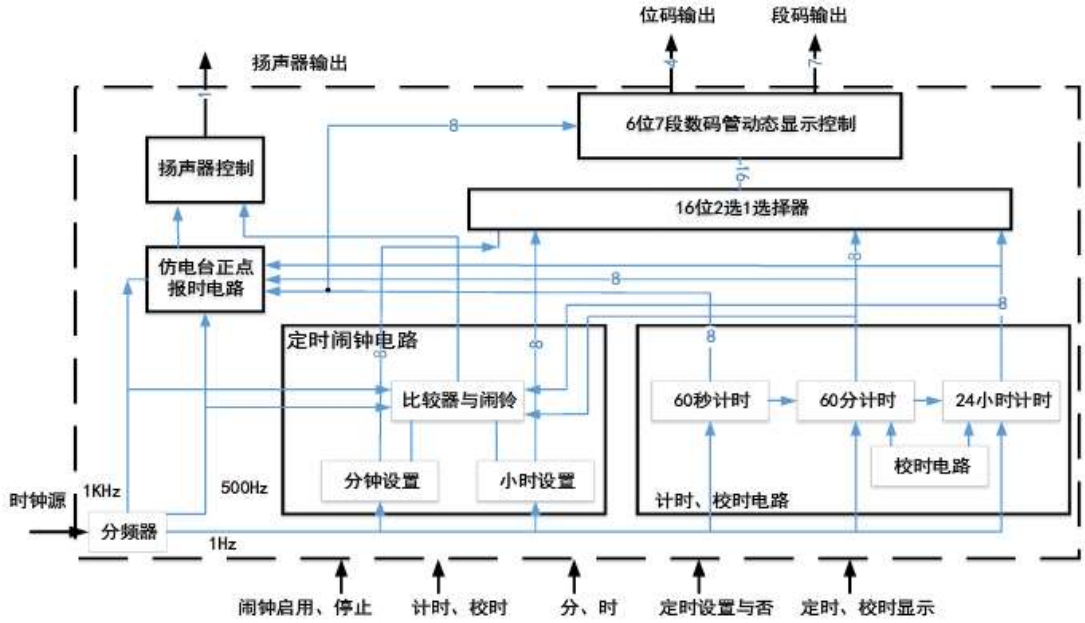
4. VERILOG 层次化设计中调用底层模块的方法：

基本方式： 模块名 调用名（端口名表项）

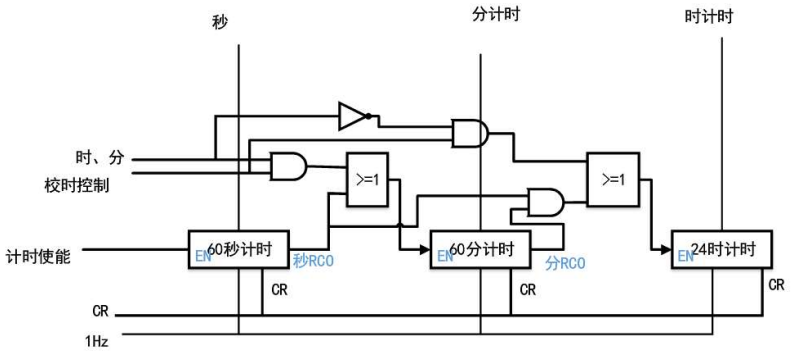
调用方式一：位置对应调用方式；注意位置要严格对应；

调用方式二：端口名对应调用方式；注意端口名要保持一致。

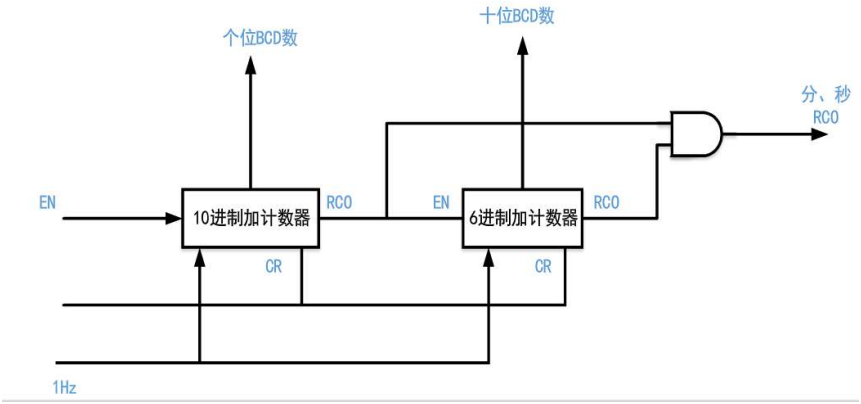
3.3 流程框图



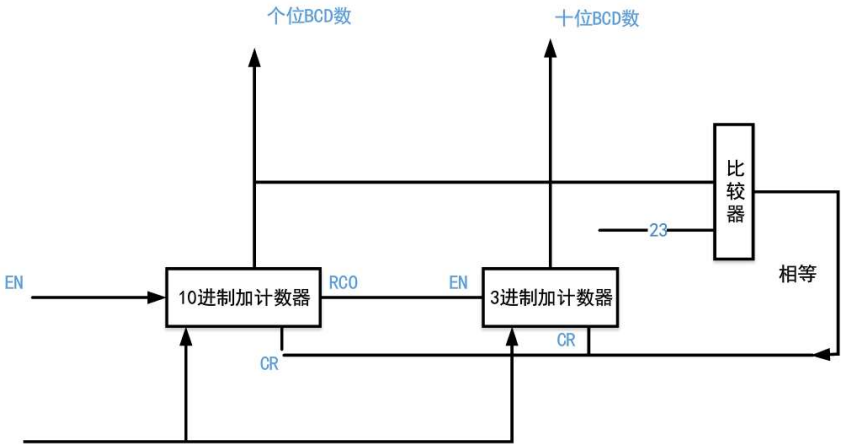
3.4 计时模块



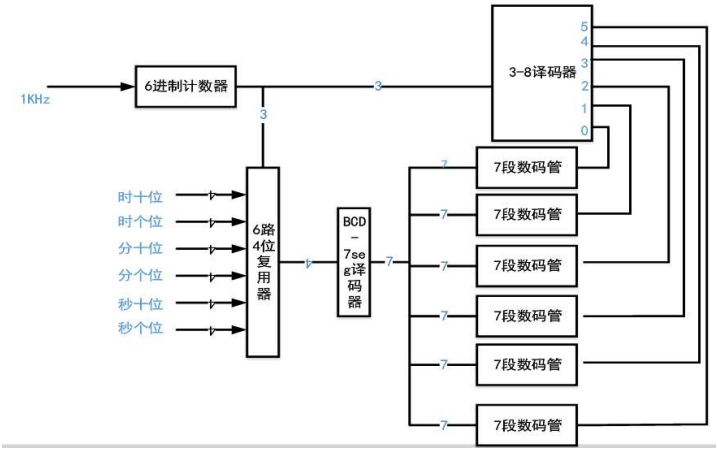
3.5 60 进制 BCD 计数器



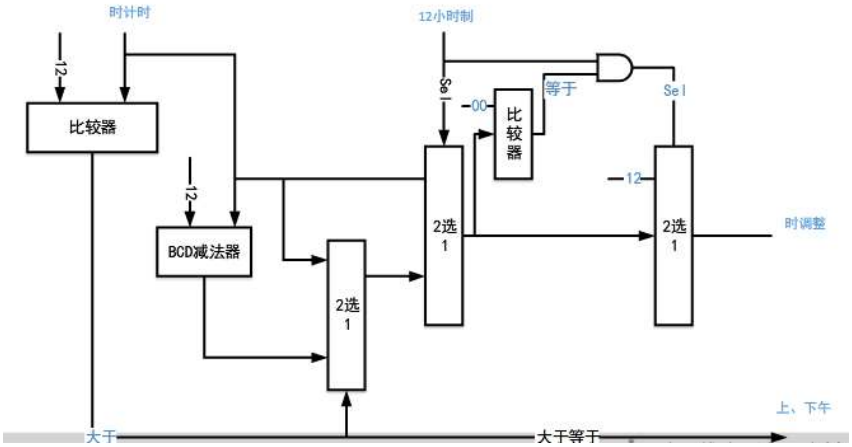
3.6 24 进制 BCD 计数器



3.7 6 位数码管显示



3.8 24/12 小时切换



4 实验内容

4.1 报时模块

```
module baoshi(clk,nCR,minuteh,minutel,hourh,hourl,rock        );
        input nCR;
        input clk;
        input [3:0] minuteh,minutel;
        input [3:0] hourh,hourl;
        output reg rock;
        integer i=0;
        reg en;
        always@(posedge clk or negedge nCR)
        begin
        if(~nCR)
        begin
        rock<=0;
        end
        else if((minuteh==4'd5)&&(minutel==4'd9)) //到达报时时间段
        begin
        i=0;                              //对 i 进行置零
        en<=1;                            //允许报时
        end
        else if((i<(hourh*20+hourl*2))&&(en))       //当 i<整点数字以及允许报时时
(en==1),开始报时
        begin
        rock<=~rock;
         rock<=~rock;
```

```
i=i+1;
end
else if(i==(hourh*20+hourl*2))     //当 i==整点数字时，允许信号置零
en<=0;
else
rock<=0;
end
endmodule
```

4.2 对时模块

```
module comparetime(CP, Set_Hour, Set_Minute, Hour, Minute, Time);
    input [7:0] Set_Hour, Set_Minute, Hour, Minute;
    input CP;
    output reg Time;
    always @(posedge CP)
        begin
            if((Set_Hour == Hour) && (Set_Minute == Minute))
                Time<= 1;
            else
                Time<= 0;
        end
endmodule
```

4.3 模 24

```
module count24_1(CntH,CntL,nCR,EN,CP,change);
        input nCR,CP,EN,change;
        output reg [3:0] CntH;
        output reg [3:0] CntL;
        always@(posedge CP,negedge nCR)
        begin
        if(nCR==0)
        begin {CntH,CntL}<=8'b00000000; end
        else if(change==0)          //*change=0 时，进入 24 小时制**********
        begin
                if(EN==0)
                    begin {CntH,CntL}<={CntH,CntL}; end
                else if((CntH>2)||(CntL>9)||((CntH==2)&&(CntL>=3)))
                    begin {CntH,CntL}<=8'b00000000; end
                else if((CntH==2)&&(CntL<3))
                    begin CntH<=CntH;CntL<=CntL+1'b1;end
                else if(CntL==9)
```

```
                    begin CntH<=CntH+1'b1;CntL<=4'b0000; end
                else
                    begin CntH<=CntH;CntL<=CntL+1'b1; end
            end
            /*else if(change==1)
            begin
                if((CntH>=1)&&(CntL>2))
                 begin tim<=1; CntH<=CntH-1'b1; CntL<=CntL-2'b10;end
    //将 24 小时制的下午时间转换为 12 小时制，并显示下午
                 else if(EN==0)
                  begin {CntH,CntL}<={CntH,CntL}; end
if((CntH>4'b0001)||(CntL>4'b1001)||((CntH==4'b0001)&&(CntL>=4'b0010)))
                 begin {CntH,CntL}<=8'b00000001; tim<=~tim; end
    //    led 灯上午下午进行转变
                 else if((CntH==1)&&(CntL<1))
                      begin CntH<=CntH;CntL<=CntL+1'b1;end
                 else if(CntL==9)
                      begin CntH<=CntH+1'b1;CntL<=4'b0000; end
                 else
                      begin CntH<=CntH;CntL<=CntL+1'b1; end
            end
            end*/
            else if(change==1)//*change=1 时，进入 12 小时制*********
          begin
         if(CntH>=4'b0001&&CntL>4'b0010) begin CntH<=CntH-1'b1; CntL<=CntL-
2'b10;   end
            else if(~EN) {CntH,CntL}<={CntH,CntL};
            else
            begin
if((CntH>4'b0001)||(CntL>4'b1001)||((CntH==4'b0001)&&(CntL>=4'b0010)))
             begin {CntH,CntL}<=8'h01;   end
            else if((CntH==4'b0001)&&(CntL<4'b0001))
             begin CntH<=CntH; CntL<=CntL+1'b1; end
            else if(CntL==4'b1001)
             begin CntH<=CntH+1'b1; CntL<=4'b0000; end
            else
             begin CntH<=CntH; CntL<=CntL+1'b1; end
          end
          end//12 进制小时计数完成
          end
          endmodule
```

4.4 模 6

```
module count6(Q,nCR,EN,CP);
        input EN,CP,nCR;
        output reg [3:0] Q;
        always@(posedge CP,negedge nCR)
        begin
            if(nCR==0)Q<=4'b0000;
            else if(EN==0)Q<=Q;
            else if(Q==4'b0101) Q<=4'b0000;
            else Q<=Q+1'b1;
        end
endmodule
```

4.5 模 10

```
module count10(Q,nCR,EN,CP);
        input EN,nCR,CP;
        output reg [3:0]Q;
        always@(posedge CP,negedge nCR)
        begin
        if(nCR==0)Q<=4'b0000;//清零
        else if(EN==0)Q<=Q;
        else if(Q==4'b1001)Q<=4'b0000;
        else Q<=Q+1'b1;
        end
endmodule
```

4.6 模 60

```
//模 60 计数器模块
module counter60(clk, rst_n, en, dout, co);
    input clk, rst_n, en;
    output co;
    output [7:0] dout;
    wire co10_1, co10, co6;
    wire [3:0] dout10, dout6;
    count10 U1(.clk(clk), .rst_n(rst_n), .en(en), .dout(dout10), .co(co10_1));
    count6 U2(.clk(clk), .rst_n(rst_n), .en(co10), .dout(dout6), .co(co6));
    and U3(co, co10, co6);
    and U4(co10, en, co10_1);
    assign dout = {dout6, dout10};
```

```
endmodule
```

4.7 驱动模块

```
module divider(CR,clk_in,clk_out        );
        parameter count=25;
          parameter infreq=50000000;
          parameter outfreq=1000;//修改为 1
          input clk_in,CR;
          output clk_out;
          reg clk_out;
          reg[count-1:0]counter;
          initial begin
               counter=0;
               clk_out=0;
          end
          always@(posedge clk_in or negedge CR)
          begin
               if(CR==0)
               begin
                    counter<=0;
                    clk_out<=0;
               end
               else begin
                    if(counter<infreq/(2*outfreq))
                         counter<=counter+1;
                    else begin
                         counter<=0;
                         clk_out<=~clk_out;
                    end
               end
          end
endmodule
```

4.8 1KHz 频率产生

```
module divider1Hz(CP,nCR,clk        );
       input CP,nCR;
       reg [29:0]count2;
       output reg clk;
      always @(posedge CP)//产生 1Hz 的频率
       begin
          if(nCR==0)    begin clk=0;count2<=30'd00;end
```

```
            else if(count2==30'd25000000) begin count2<=30'd00; clk=~clk; end
            else count2<=count2+1'b1;
        end
endmodule
```

4.9 50MHz 频率产生

```
module divider(clk1,clr,clk2        );
        parameter n=25;
        parameter infreq=50000000;
        parameter outfreq=1000;
        input clk_in,CR;
        output clk_out;
        reg clk_out;
        reg[count-1:0]counter;

        initial begin
            counter=0;
            clk_out=0;
        end
        always@(posedge clk_in or negedge CR)begin
            if(CR==0)begin
                counter<=0;
                clk_out<=0;
            end
            else begin
                if(counter<infreq/(2*outfreq))
                    counter<=counter+1;
                else begin
                    counter<=0;
                    clk_out<=~clk_out;
                end
            end
        end
endmodule
```

4.10 点亮数码管

```
module lut7(out,in        );
    input [3:0] in;
    output reg [6:0] out;
    always@(in)
    begin
```

```
        case(in)
                    4'h1:out=7'b111_1001;
                    4'h2:out=7'b010_0100;
                    4'h3:out=7'b011_0000;
                    4'h4:out=7'b001_1001;
                    4'h5:out=7'b001_0010;
                    4'h6:out=7'b000_0010;
                    4'h7:out=7'b111_1000;
                    4'h8:out=7'b000_0000;
                    4'h9:out=7'b001_0000;
                    4'h0:out=7'b100_0000;
                    default:out=7'b1000000;
                endcase
            end
    endmodule
```

4.11 闹钟模块

```
    module
naozhong(clk_2Hz,nCR,naozhong_swh,,baochi_swh,Adj_Hour,Adj_Min,hourl,hourh,m
inutel,minuteh,CntH,CntL,CntHm,CntLm,alarm);
    input clk_2Hz,nCR;
    input naozhong_swh,baochi_swh;      //闹钟开关
    input Adj_Hour,Adj_Min;
    input [3:0]hourl,hourh;          //时间比较
    input [3:0]minutel,minuteh;
    output reg[3:0]CntH,CntL;
    output reg[3:0]CntHm,CntLm;
    output reg alarm;
    always@(posedge clk_2Hz)
    begin
        if((naozhong_swh)&&(!baochi_swh))
        begin
        if(nCR==0)
                begin
alarm=0;{CntHm,CntLm}<=8'b00000000;{CntH,CntL}<=8'b00000000; end   //清零
        else if(Adj_Hour)              //设定小时
                begin
                    if((CntH>2)||(CntL>9)||((CntH==2)&&(CntL>=3)))
                       begin {CntH,CntL}<=8'b00000000; end
                    else if((CntH==2)&&(CntL<3))
                         begin CntH<=CntH;CntL<=CntL+1'b1;end
                    else if(CntL==9)
```

```
                    begin CntH<=CntH+1'b1;CntL<=4'b0000; end
                else
                    begin CntH<=CntH;CntL<=CntL+1'b1; end
                end
            else if(Adj_Min)        //设定分钟
              begin
                if((CntHm>5)||(CntLm>9)||((CntHm==5)&&(CntLm>=9)))
                   begin {CntHm,CntLm}<=8'b00000000; end
                else if((CntHm==5)&&(CntLm<9))
                    begin CntHm<=CntHm;CntLm<=CntLm+1'b1;end
                else if(CntLm==9)
                    begin CntHm<=CntHm+1'b1;CntLm<=4'b0000; end
                else
                    begin CntHm<=CntHm;CntLm<=CntLm+1'b1; end
              end
        end
    end
    //显示闹钟
    always@(posedge clk_2Hz)
        begin
            if((CntH == hourh)&&(CntL == hourl)&&(CntHm==minuteh)&&(CntLm==minutel))
                begin
                    alarm <= 'd1;
                end
            else if(baochi_swh)
                begin
                    alarm <= 'd0;
                end
            else
                begin
                   alarm <= 'd0;
                end
    end

    endmodule
```

4.12 扫描

```
    module scan(CP,nCR,clk1
        );
         input CP,nCR;
         output reg clk1;
```

```
        reg [29:0]count1;
        always @(posedge CP)
         begin
            if(~nCR)    count1<=30'd00;
            else if(count1==30'd100000) begin count1<=30'd00; clk1=~clk1; end
            else count1<=count1+1'b1;
         end
    endmodule
```

4.13 主程序代码

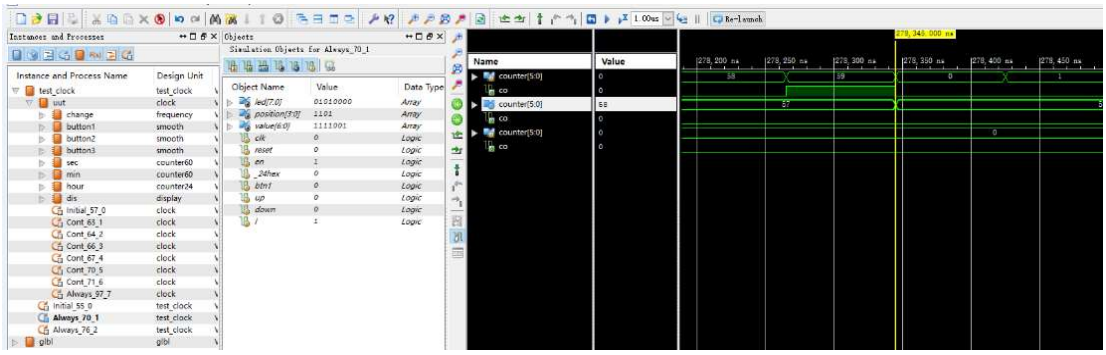
```
    module
top(Clock,Hour,Minute,Second,CP,nCR,EN,Adj_Min,Adj_Hour,out,Light,clk,in,change
,tim,rock,q,w,e,r,t,Set_Alarm,Set_Min,Set_Hour);

        input CP;                                   //输入的时钟信号
        input nCR;
        input EN,change;                           //change 为 12/24 进制转换开关
        input Adj_Min;                              //校分控制
        input Adj_Hour;                             //校时控制
    //      **
          input Set_Alarm;
         input Set_Min, Set_Hour;
         output    Clock;
        wire [7:0] C_Hour, C_Minute;
       wire C_MinL_EN, C_MinH_EN;
       wire C_Hour_EN;
    //**
        output [7:0] Hour,Minute,Second;
        output [3:0] Light,in;              //light 控制数码管，in 控制显示数字
        output [6:0] out;                    //控制 led 灯
       output clk,tim;                       //clk 为分频时钟信号，频率为 1Hz
        //
        output rock;
        //
        output reg q,w,e,r,t;
        wire [7:0] Hour,Minute,Second;
        reg clk;                                 //clk 为 1Hz 的频率,clk 为 2Hz 的频率
        reg [29:0] count,count1;                    //分频需要的计数器
        reg [1:0] scan;                     //扫描时需要的计数器
        reg [3:0] Light,in;
        reg [6:0] out;
        reg clk1;                            //为分频后的扫描频率
```

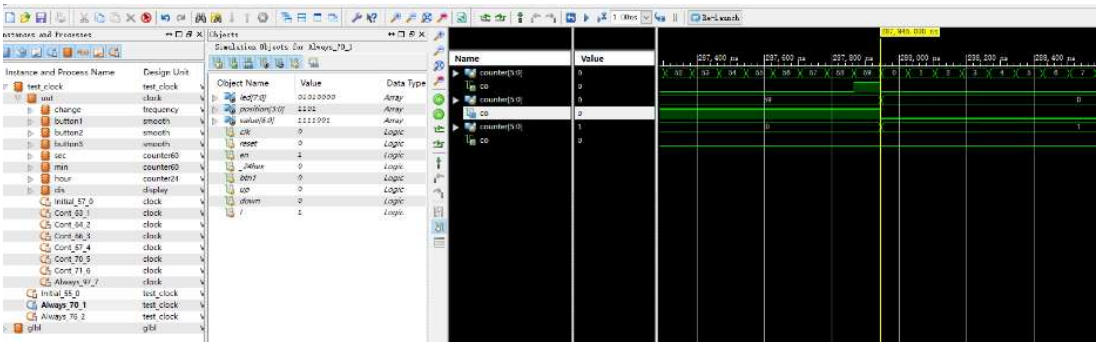
```
        supply1 Vdd;                        //vdd 恒等于 1
        wire MinL_EN,MinH_EN,Hour_EN;
        //
    //60 进制秒计数器
        count10 U1(Second[3:0],nCR,EN,clk);                    //秒个位
        count6 U2(Second[7:4],nCR,(Second[3:0]==4'h9),clk);    //秒十位
    //60 进制分钟计数器
        count10 U3(Minute[3:0],nCR,MinL_EN,clk);
        count6 U4(Minute[7:4],nCR,MinH_EN,clk);
    //产生分钟计数器使能信号。Adj_Min=1，校正分钟；Adj_Min=0，分钟正常计时
        assign MinL_EN=Adj_Min?Vdd:(Second==8'h59);
        assign MinH_EN=(Adj_Min&&(Minute[3:0]==4'h9))||((Minute[3:0]==4'h9)&&(Second==8'h59));
    //产生小时计数器使能信号。Adj_Hour=1，校正小时；Adj_Hour=0，小时正常计时
        assign Hour_EN=Adj_Hour?Vdd:((Minute==8'h59)&&(Second==8'h59));
    //小时计数器
        count24 U5(Hour[7:4],Hour[3:0],nCR,Hour_EN,clk,change,tim);
    //整点报时**************
        baoshi U9(clk,nCR,Minute[7:4],Minute[3:0],Hour[7:4],Hour[3:0],rock);
    //闹钟***************
        count10 M2 (C_Minute[3:0], nCR, C_MinL_EN, clk);
        count6   M3 (C_Minute[7:4], nCR, C_MinH_EN, clk);
        assign C_MinL_EN = (Set_Min && Set_Alarm);
        assign C_MinH_EN = (Set_Min && (C_Minute[3:0] == 4'h9)&&Set_Alarm);
        count24_1 H1(C_Hour[7:4],C_Hour[3:0],nCR,C_Hour_EN,clk,change);
        assign  C_Hour_EN=(Set_Alarm&&Set_Hour)?Vdd:(Set_Alarm&&C_Minute==8'h59&&Set_Min);
        comparetime C1 (clk,C_Hour,C_Minute,Hour,Minute, Time);
       assign Clock = Time;
    //分频
         always @(posedge CP)
         begin
           if(nCR==0) begin     t<=0; q<=0;w<=0;e<=0;r<=0;clk<=0; count<=30'd00; end
           else if(count==30'd25000000) begin count<=30'd00; clk<=~clk; end

           else count<=count+1'b1;
         end
    //扫描
       /*scan U7(CP,nCR,clk1);*/
        always @(posedge CP)
```

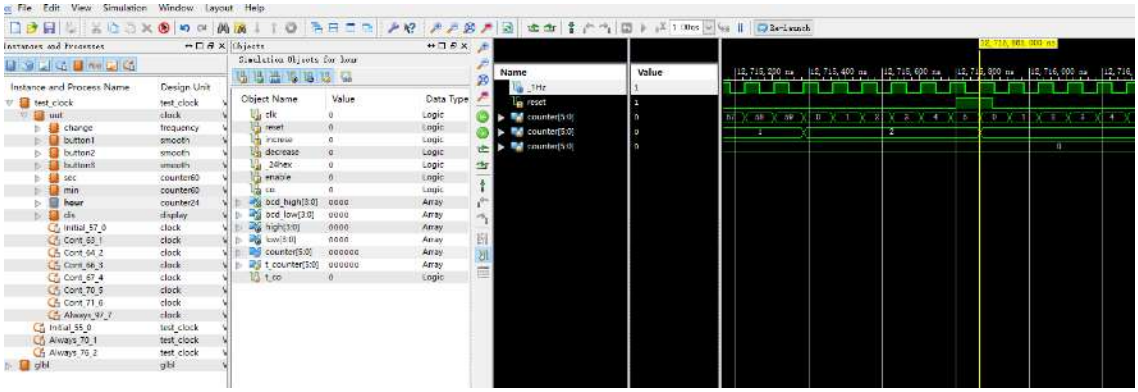
```
        begin
          if(~nCR) begin clk1<=0; count1<=30'd00;    end
          else if(count1==30'd100000) begin count1<=30'd00; clk1<=~clk1; end
          else count1<=count1+1'b1;
        end
//扫描计数器
        always @(posedge clk1)
        begin
          if(~nCR) scan<=2'b00;
          else if(scan==2'b11) scan<=2'b00;
          else scan<=scan+1'b1;
           end
      always @(scan[1:0])
        begin
        case(scan[1:0])
        2'b00: Light<=4'b0111;
        2'b01: Light<=4'b1011;
        2'b10: Light<=4'b1101;
        2'b11: Light<=4'b1110;
        endcase
        end
//显示数字
       always @ (scan[1:0] or Hour or Minute or C_Minute or C_Hour or Set_Alarm)
       begin
       if(Set_Alarm==0) //显示时钟
       begin
         case(scan[1:0])
         2'b00: in<=Hour[7:4];
         2'b01: in<=Hour[3:0];
         2'b10: in<=Minute[7:4];
         2'b11: in<=Minute[3:0];
         endcase
       end
      else              //设置时钟信号
       begin
         case(scan[1:0])
         2'b00: in<=C_Hour[7:4];
         2'b01: in<=C_Hour[3:0];
         2'b10: in<=C_Minute[7:4];
         2'b11: in<=C_Minute[3:0];
         endcase
       end
      end
/*//译码
```

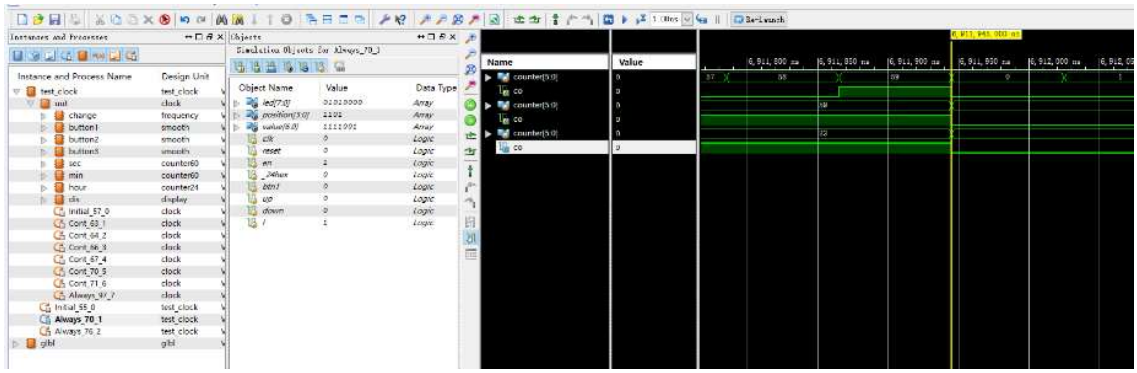
```
lut7 U8(out,in);*/
always @(in)
 begin
    case(in)
                4'd0:out<=7'b0000001;    //0
            4'd1:out<=7'b1001111;    //1
            4'd2:out<=7'b0010010;    //2
            4'd3:out<=7'b0000110;    //3
            4'd4:out<=7'b1001100;    //4
            4'd5:out<=7'b0100100;    //5
            4'd6:out<=7'b0100000;    //6
            4'd7:out<=7'b0001111;    //7
            4'd8:out<=7'b0000000;    //8
            4'd9:out<=7'b0000100;    //9
            default:out<=7'b0000001;
          endcase
      end
endmodule
```

4.14 \仿真时序图









5 总结与感想

在本实验中，我参考并仿照案例代码完成了基础功能，并且加入了 LED 灯计秒的功能，同时优化了按键布局。整个程序较之前的程序更加庞大，功能也更加复杂，但是拆分模块之后容易分部实现。主要的难点在整合部分，即主程序的编写。

通过此实验，我更加深入了解了 VHDL 语言的语法规则，并且练习了各种调用底层模块的方法，并且实现了简单的数字钟，整个实验还是较为成功的。