**华中科技大学**

**《电子线路设计、测试与实验》实验报告**

|  |  |
| --- | --- |
| 实验名称： | VERILOG HDL语言设计实现数字钟 |
| 院（系）： | 材料学院 |
| 专业班级： | 电封1701 |
| 姓名： | 徐灏阳 |
| 学号： | U201711106 |
| 实验成绩： |  |

**一、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

**二、实验要求**

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.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。

三、实验原理

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。

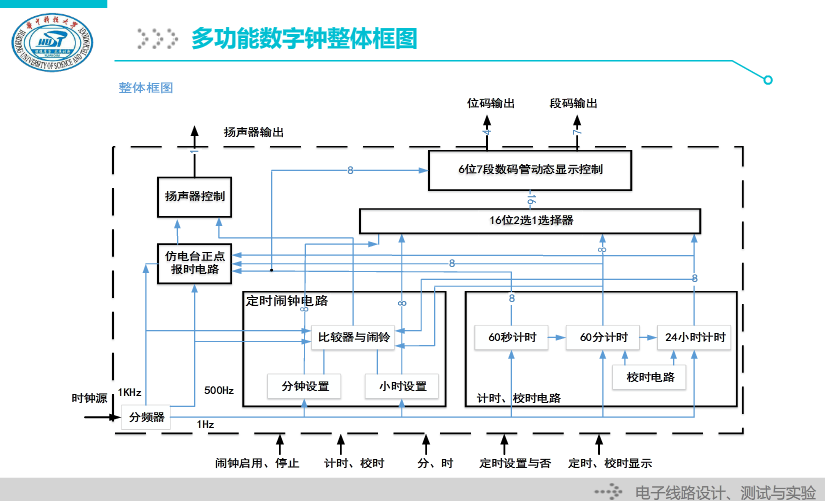
3. 自下而上---是一种传统的设计方法，从存在的基本单元出发，设计树最末枝上的单元要么是已经制造出的单元要么是其他项目已开发好的单元或者是可外购得到的单元，逐级叠加，逐模块叠加，直至实现功能。

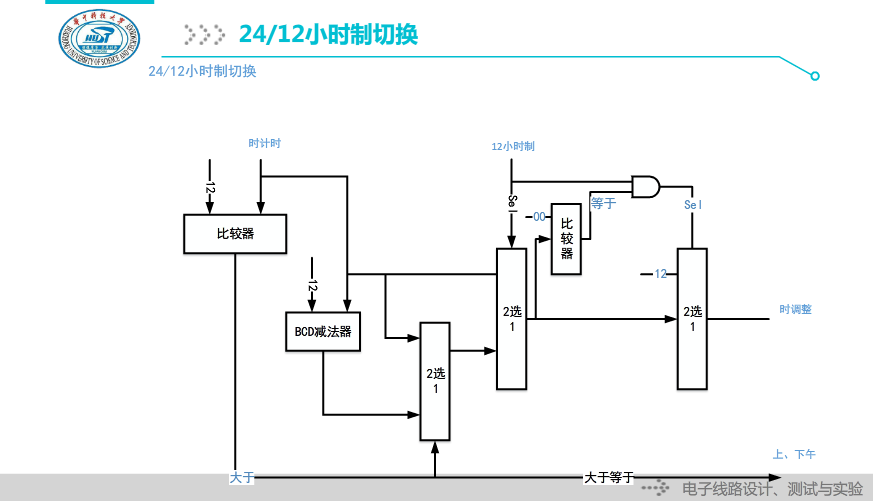
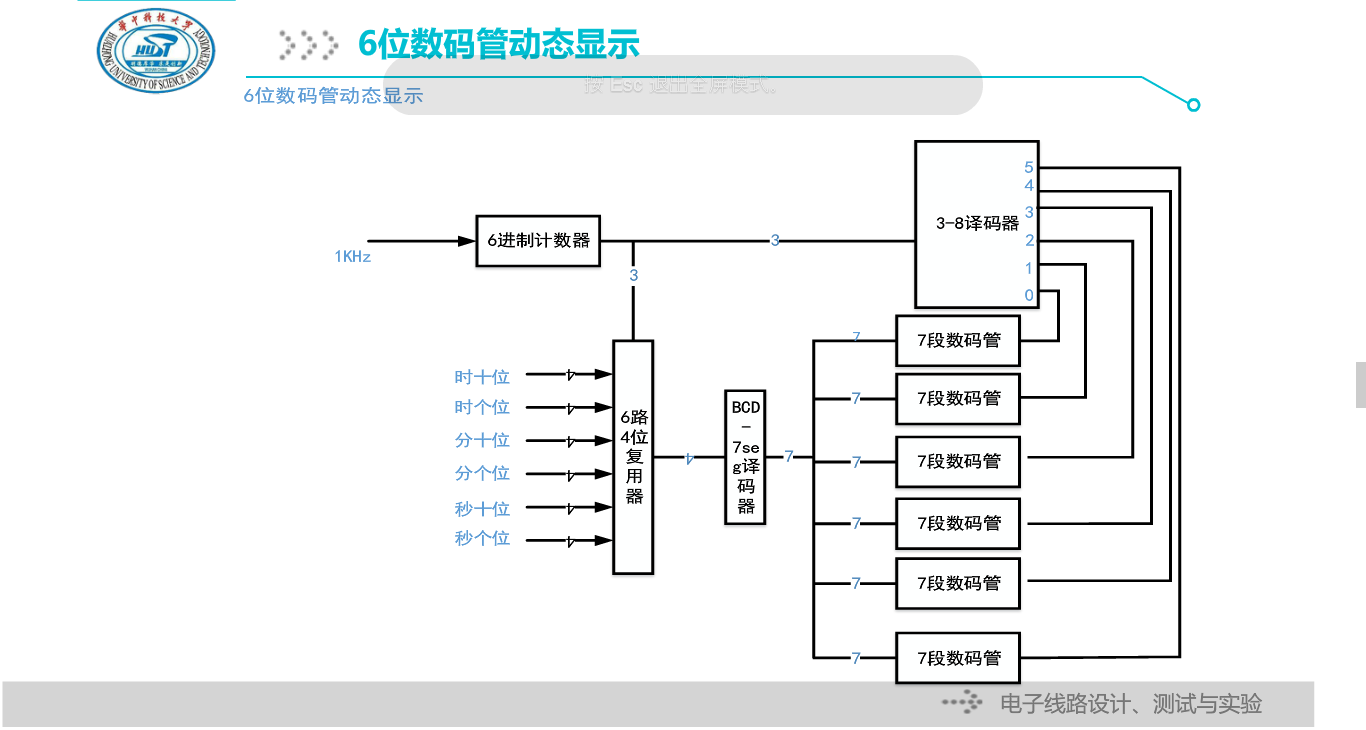
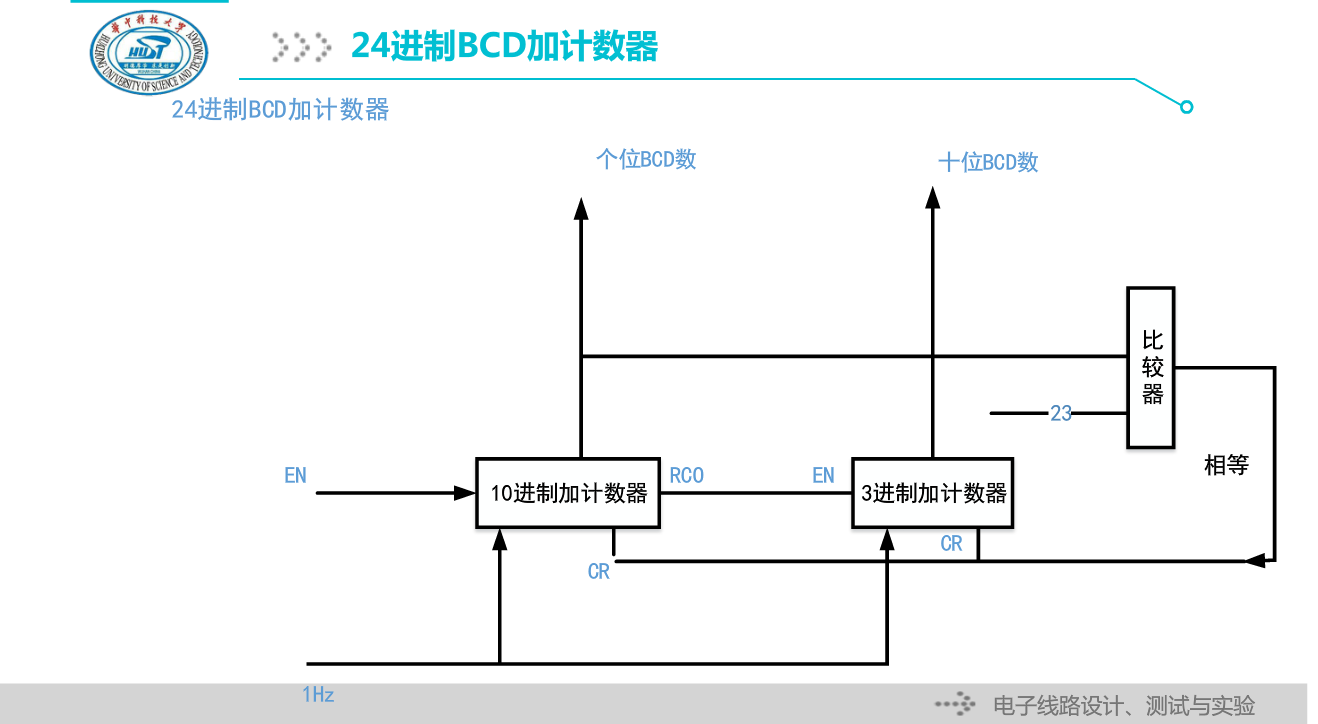
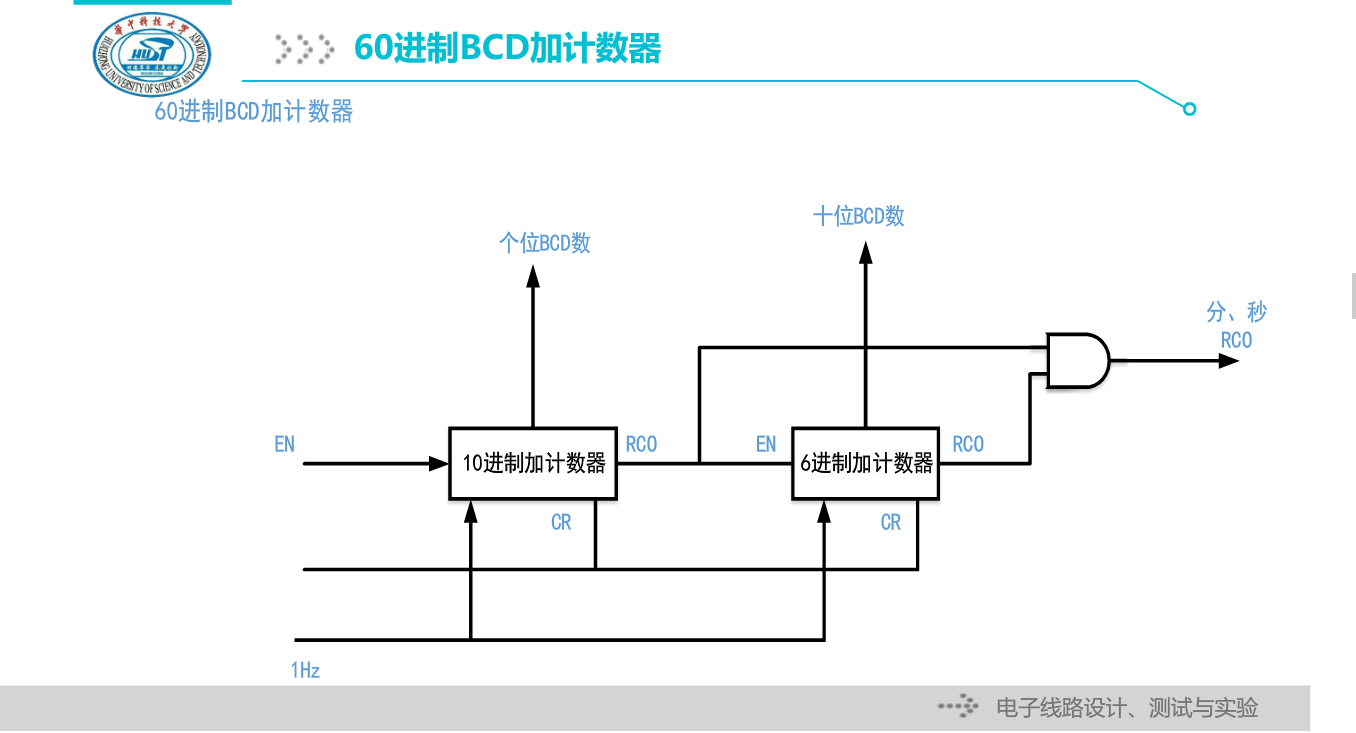
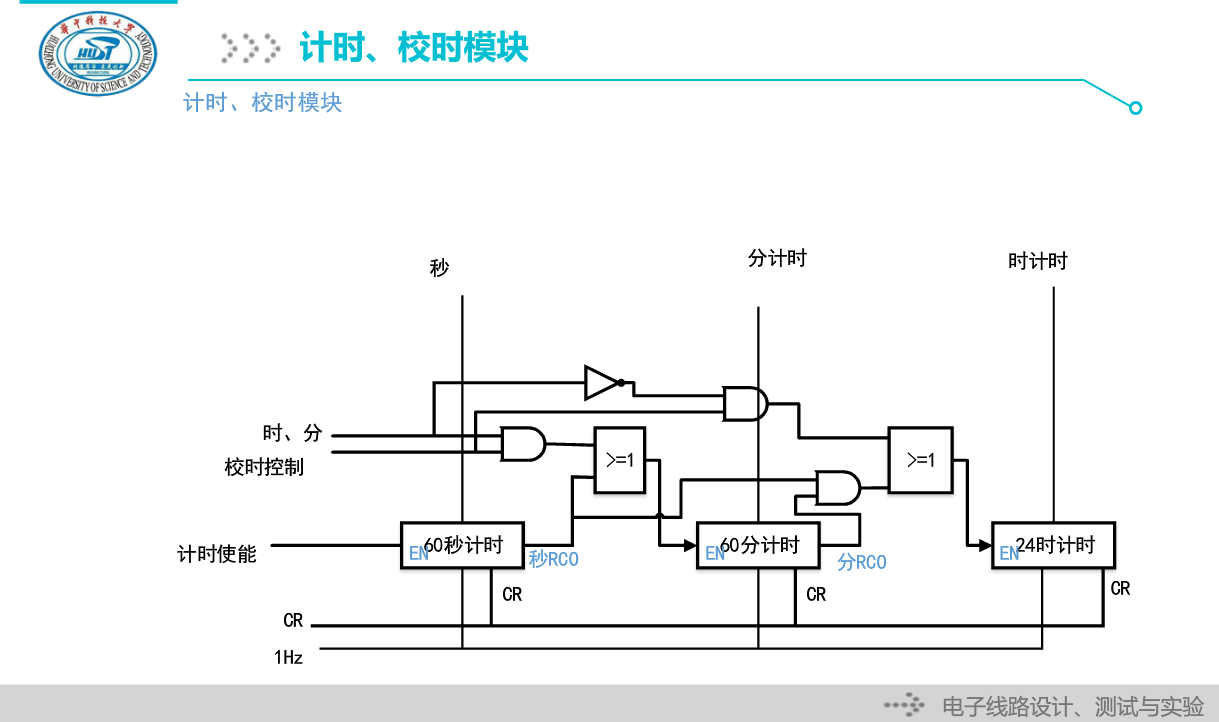
4. verilog 层次化设计中调用底层模块的方法：

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。





四、实验内容

module top\_clock(

input Hchange, //24小时，12小时切换信号

input Change, //用来进行时分和秒的显示切换

input CLK\_50, //50MHz时钟

input nCR,EN,

input Adj\_Min,Adj\_Hour, //使能信号，小时分钟调时允许信号

output [6:0] HEX0,

output reg [3:0] HEX //共阳极数码管对应端

);

wire ENM\_L,ENM\_H,ENH;

wire [7:0] Hour\_24,TMinute,TSecond;//中间变量声明,正常时钟变量

wire [7:0] Hour\_12,Display\_HourT,Display\_HourAdjust;

reg [3:0] bcd=4'b0000; //记载数码管所要显示的数据

reg [7:0] Display\_Hour,Minute;

supply1 Vdd;

wire CP\_1Hz;

//===========分频=============

Divider50MHz U0(.CLK\_50M(CLK\_50),.nCLR(nCR),.CLK\_1HzOut(CP\_1Hz));//用以时钟计数的CP

defparam U0.N = 25,

U0.CLK\_Freq = 50000000,

U0.OUT\_Freq = 1;

Divider50MHz U1(.CLK\_50M(CLK\_50),.nCLR(nCR),.CLK\_1HzOut(CP\_200Hz));//用以动态扫描的CP,供给数码管

defparam U1.N = 18,

U1.CLK\_Freq = 50000000,

U1.OUT\_Freq = 200;

//===========60进制秒计数器=========

Scounter10 S0(TSecond[3:0],nCR,EN,CP\_1Hz);//秒：个位

Scounter6 S1(TSecond[7:4],nCR,(TSecond[3:0]==4'h9),CP\_1Hz);//秒：十位

//===========60进制分计数器=========

Mcounter10 M0(TMinute[3:0],nCR,ENM\_L,EN,CP\_1Hz);//分：个位

Mcounter6 M1(TMinute[7:4],nCR,ENM\_H,EN,CP\_1Hz);//分：十位

assign ENM\_L=Adj\_Min?Vdd:(TSecond==8'h59);

assign ENM\_H=(Adj\_Min&&(TMinute[3:0]==4'h9))||(TMinute[3:0]==4'h9)&&(TSecond==8'h59); //24小时制

counter24 H0(Hour\_24[7:4],Hour\_24[3:0],nCR,ENH,EN,CP\_1Hz);

assign ENH = Adj\_Hour?Vdd:((TMinute==8'h59)&&(TSecond==8'h59));

//===========12小时与24小时进制切换控制==========

assign Display\_HourAdjust=((Hour\_24==8'h20)||(Hour\_24==8'h21))?(Hour\_24-24):(Hour\_24-18);

assign Hour\_12 = (Hour\_24<8'h13)?Hour\_24:Display\_HourAdjust;

assign Display\_HourT = Hchange?Hour\_12:Hour\_24;

//===========数码显示=========

always@(posedge CLK\_50)

begin

Display\_Hour <= Display\_HourT;Minute <= TMinute;

end

always@(posedge CP\_200Hz)

begin

if(Change==1) //数码管进行时分显示

begin

case(HEX)

4'b1110: begin HEX<=4'b0111; bcd<= Display\_Hour [7:4]; end //第一根数码管显示小时十位

4'b0111: begin HEX<=4'b1011; bcd<= Display\_Hour [3:0]; end //第二根显示小时个位

4'b1011: begin HEX<=4'b1101; bcd<= Minute [7:4]; end //第三根显示分钟十位

4'b1101: begin HEX<=4'b1110; bcd<= Minute [3:0]; end //第四根显示分钟个位

default: begin HEX<=0111; bcd<=Display\_Hour [7:4]; end

endcase

end

else //数码管进行秒显示,change为低电平时显示秒

begin

case(HEX)

4'b1110: begin HEX<=4'b1101; bcd<= TSecond [7:4]; end //第三根显示秒十位

4'b1101: begin HEX<=4'b1110; bcd<= TSecond [3:0]; end //第四根显示秒个位

default: begin HEX<=1101; bcd<= TSecond [7:4]; end

endcase

end

end

SEG7\_LUT L0(HEX0,bcd); //调用数码管子函数

endmodule

module Divider50MHz(CLK\_50M,nCLR,CLK\_1HzOut);

parameter N = 25; //位宽

parameter CLK\_Freq = 50000000; //50MHz时钟输入

parameter OUT\_Freq = 1; //1Hz时钟输出

input nCLR,CLK\_50M; //输入端口说明

output reg CLK\_1HzOut; //输出端口说明

reg [N-1:0] Count\_DIV; //内部节点，存放计数器的输出值

always@(posedge CLK\_50M or negedge nCLR)

begin

if(!nCLR)

begin

CLK\_1HzOut <= 0; Count\_DIV <= 0;

end

else

begin

if(Count\_DIV <(CLK\_Freq/(2\*OUT\_Freq)-1))//计数器模

Count\_DIV <= Count\_DIV + 1'b1; //分频器计数加1

else

begin

Count\_DIV <= 0; //分频器输出清零

CLK\_1HzOut <= ~CLK\_1HzOut; //输出信号取反

end

end

end

endmodule

module Scounter10(Q,nCR,EN,CP);

input CP,nCR,EN;

output Q;

reg [3:0] Q;

always @(posedge CP or negedge nCR)

begin

if(~nCR) Q <= 4'b0000;//异步清零

else if(~EN) Q <= Q; //暂停计数

else if(Q==4'b1001) Q <= 4'b0000;

else Q <= Q + 1'b1;

end

endmodule

module Scounter6(Q,nCR,EN,CP);

input CP,nCR,EN;

output Q;

reg [3:0] Q;

always @(posedge CP or negedge nCR)

begin

if(~nCR) Q <= 4'b0000;//异步清零

else if(~EN) Q <= Q; //暂停计数

else if(Q==4'b0101) Q <= 4'b0000;

else Q <= Q + 1'b1;

end

endmodule

module Mcounter10(Q,nCR,EN1,EN2,CP);

input CP,nCR,EN1,EN2;

output Q;

reg [3:0] Q;

always @(posedge CP or negedge nCR)

begin

if(~nCR) Q <= 4'b0000;//异步清零

else if(~EN1||!EN2) Q <= Q; //暂停计数

else if(Q==4'b1001) Q <= 4'b0000;

else Q <= Q + 1'b1;

end

endmodule

module Mcounter6(Q,nCR,EN1,EN2,CP);

input CP,nCR,EN1,EN2;

output Q;

reg [3:0] Q;

always @(posedge CP or negedge nCR)

begin

if(~nCR) Q <= 4'b0000;//异步清零

else if(~EN1||~EN2) Q <= Q; //暂停计数

else if(Q==4'b0101) Q <= 4'b0000;

else Q <= Q + 1'b1;

end

endmodule

module counter24(CntH,CntL,nCR,EN1,EN2,CP);

input CP,nCR,EN1,EN2;

output reg [3:0] CntH,CntL;//小时的十位和个位输出

always@(posedge CP or negedge nCR)

begin

if(~nCR)

{CntH,CntL} <= 8'h00; //异步清零

else if(~EN1||~EN2)

{CntH,CntL} <= {CntH,CntL};//暂停计数

else if((CntH)>2||(CntL>9)||(CntH)==2&&(CntL)>=3)

{CntH,CntL} <= 8'h00; //对小时计数器出错时的处理

else if((CntH)==2&&(CntL)<3) //进行20~23计数

begin

CntH <=CntH; CntL <= CntL + 1'b1;

end

else if(CntL==9) //小时十位的计数

begin

CntH <=CntH + 1'b1;

CntL <= 4'b0000;

end

else

begin

CntH <= CntH; CntL <= CntL + 1'b1;

end

end

endmodule

module counter60(nCLR,Clk,EN,CntH,CntL);

input nCLR,Clk,EN;

output reg [3:0] CntH,CntL;

always@(posedge Clk or negedge nCLR)

begin

if(~nCLR)

{CntH,CntL} <= 0; //异步清零

else if(~EN)

{CntH,CntL} <= {CntH,CntL}; //暂停信号

else if(((CntH > 5)||(CntL > 9))||((CntH == 5)&&(CntL == 9)))

{CntH,CntL} <= 8'h00; //异常处理

else if(CntL == 9)

begin CntH <= CntH + 1'b1;CntL <= 0;end //十位计数

else

begin CntH <= CntH;CntL <= CntL + 1'b1;end //个位计数

end

endmodule

module SEG7\_LUT(oSEG,iDIG);

input [3:0] iDIG; //二进制输入

output reg [6:0] oSEG; //7段码输出

always@(iDIG)

begin

case(iDIG)

4'h0: oSEG = 7'b000\_0001;

4'h1: oSEG = 7'b100\_1111;

4'h2: oSEG = 7'b001\_0010;

4'h3: oSEG = 7'b000\_0110;

4'h4: oSEG = 7'b100\_1100;

4'h5: oSEG = 7'b010\_0100;

4'h6: oSEG = 7'b010\_0000;

4'h7: oSEG = 7'b000\_1111;

4'h8: oSEG = 7'b000\_0000;

4'h9: oSEG = 7'b000\_0100;

default: oSEG=7'b1111111;

endcase

end

endmodule

NET "CLK\_50" TNM\_NET = CLK\_50;

TIMESPEC TS\_CLK\_50 = PERIOD "CLK\_50" 20 ns HIGH 50%;

NET "CLK\_50" LOC = B8;

NET "nCR" LOC = P11;//拨动清零开关P11，可以实现清零功能

NET "EN" LOC = L3;//打开使能开关L3，数码管可以正常显示分和时

NET "Adj\_Min" LOC = N3;

NET "Adj\_Hour" LOC = B4;//拨动K3和B4，可以实现校时功能

NET "Change" LOC = G3;//拨动时分和秒的切换显示开关G3，数码管可以切换显示秒

NET "HEX0[6]" LOC = L14;

NET "HEX0[5]" LOC = H12;

NET "HEX0[4]" LOC = N14;

NET "HEX0[3]" LOC = N11;

NET "HEX0[2]" LOC = P12;

NET "HEX0[1]" LOC = L13;

NET "HEX0[0]" LOC = M12;

NET "HEX[0]" LOC = F12;

NET "HEX[1]" LOC = J12;

NET "HEX[2]" LOC = M13;

NET "HEX[3]" LOC = K14;

NET "Hchange" LOC = F3;//拨动F3，可实现24小时与12小时的切换

NET "CLK\_50" SLEW = FAST;

9.仿真代码

`timescale 1ns / 1ps

module test\_clock;

// Inputs

reg clk;

reg reset;

reg en;

reg \_24hex;

reg btn1;

reg up;

reg down;

// Outputs

wire [7:0] led;

wire [3:0] position;

wire [6:0] value;

// Instantiate the Unit Under Test (UUT)

clock uut (

.clk(clk),

.reset(reset),

.en(en),

.\_24hex(\_24hex),

.btn1(btn1),

.up(up),

.down(down),

.led(led),

.position(position),

.value(value)

);

initial begin

// Initialize Inputs

clk = 0;

reset = 0;

en = 1;

\_24hex = 0;

btn1 = 0;

up = 0;

down = 0;

// Wait 100 ns for global reset to finish

// Add stimulus here

end

always begin

#5;

clk = ~clk;

end

reg l = 1;

always begin

#10000;

/\* btn1 = 1;

#100;

btn1 = 0;

#300;

btn1 = 1;

#100;

btn1 = 0;

#20;

down = 1;

#500;

down = 0;\*/

reset = 1;

#100;

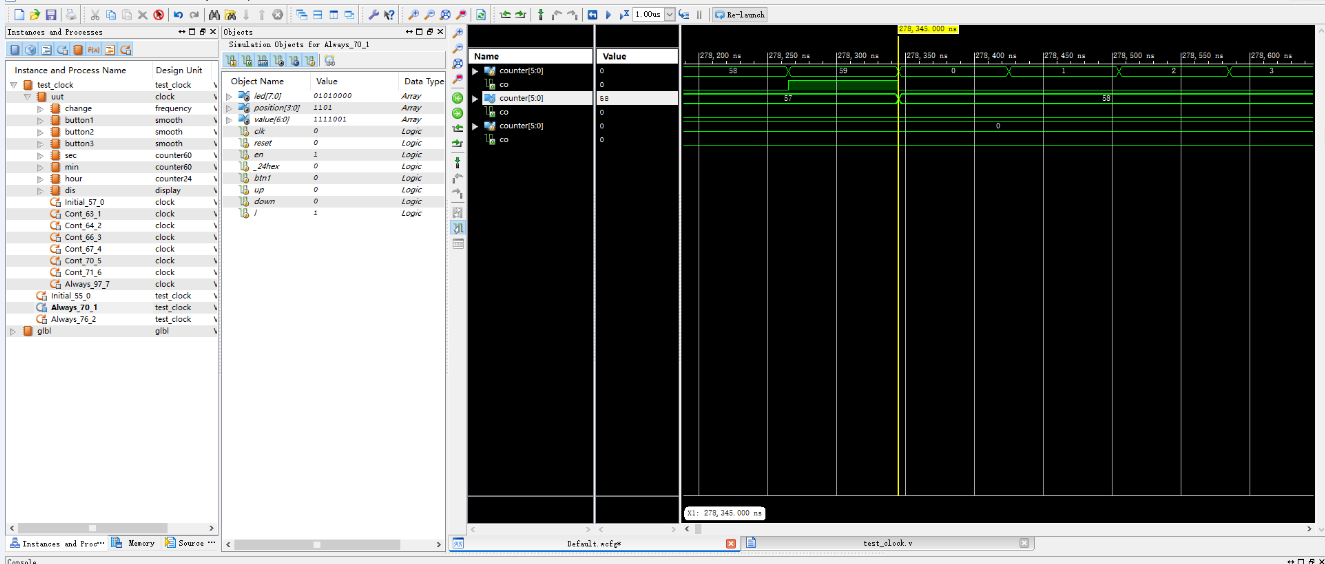
reset = 0;

end

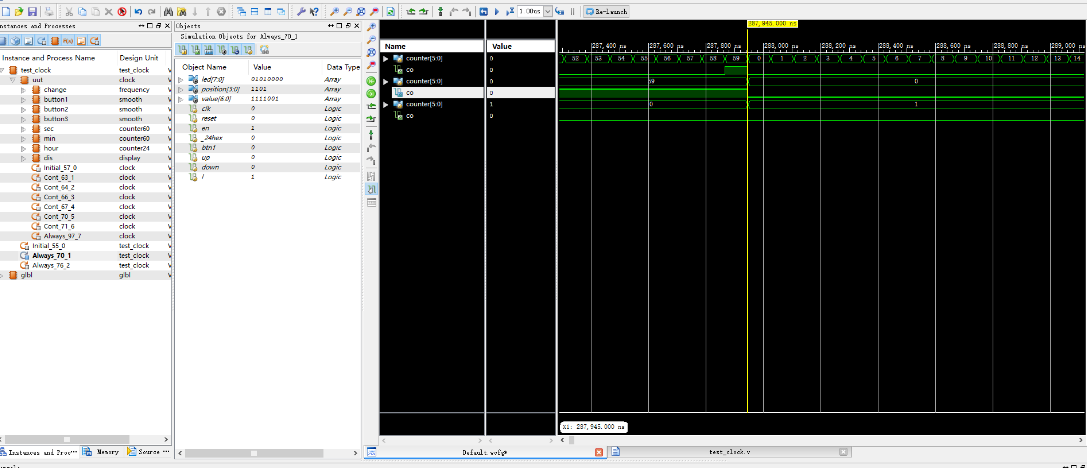
endmodule

仿真结果截图：

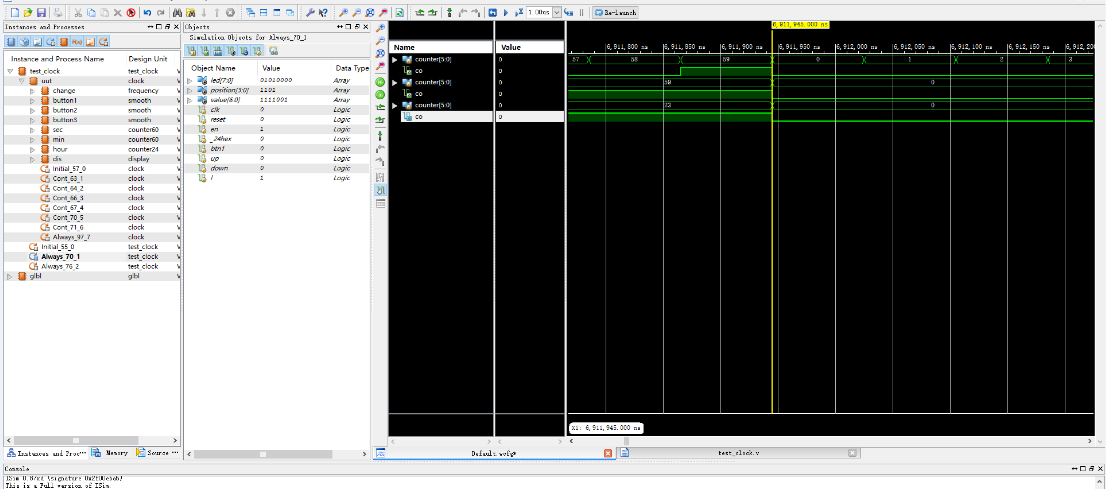
秒针进位分针：



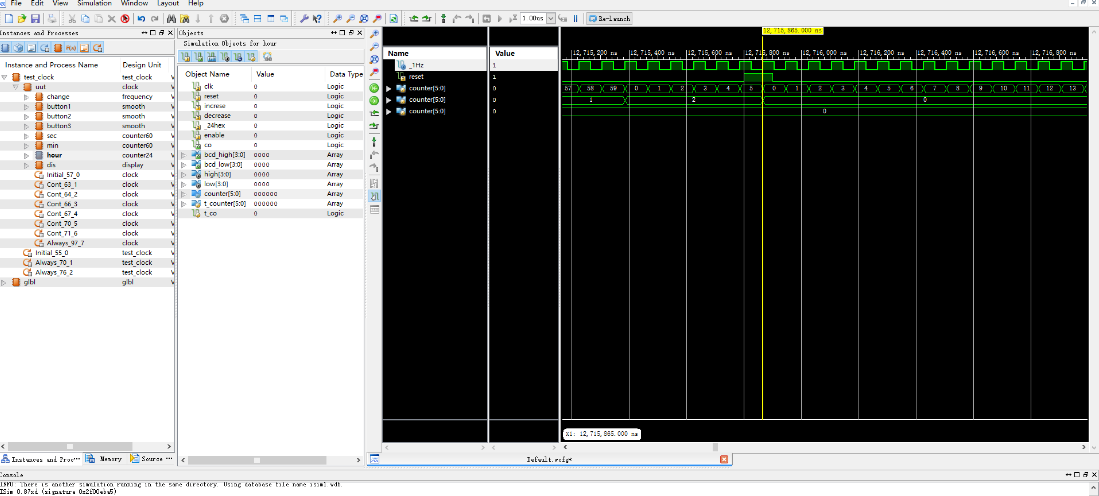
分针进位时针：



秒针分针时针总进位：



置零：



**六、实验总结**

这次实验学习了Verilog语言实现多功能数字钟的电路，因为模块较多，功能很复杂，完成起来非常吃力，但同时对复杂逻辑电路的设计又有了新的认识。