|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **学 期** | 2020-2021学年第1学期 | | | | **实验日期** | | 2020.12.23 | | |
| **学 院** | 计算机学院 | | | | **专 业** | | 计算机类 | | |
| **班 级** | 190741 | | **学 号** | | 19030419 | | **姓 名** | | 王昕 |
| **组 号** | 24 | | **学 号** | |  | | **姓 名** | |  |
| **评 阅 内 容** | | | | | | | | | |
| **实验预习** | | **实验原理** | | **详细设计** | | **实验结果** | | **成 绩** | |
|  | |  | |  | |  | |  | |
|  | | | | | | | | | |
| **题 目** | 实验12：实用电路——数字钟 | | | | | | | | |

1. **实验目的**

综合运用所学知识以及之前学习的模块，完成数字钟电路的设计实验。

**二．实验任务及要求**

基本要求：

1．设计一个具有时、分、秒计时的数字钟电路，按24小时制计时。

2．准确计时，通过动态数码管上显示时、分和秒，如14:30:10。

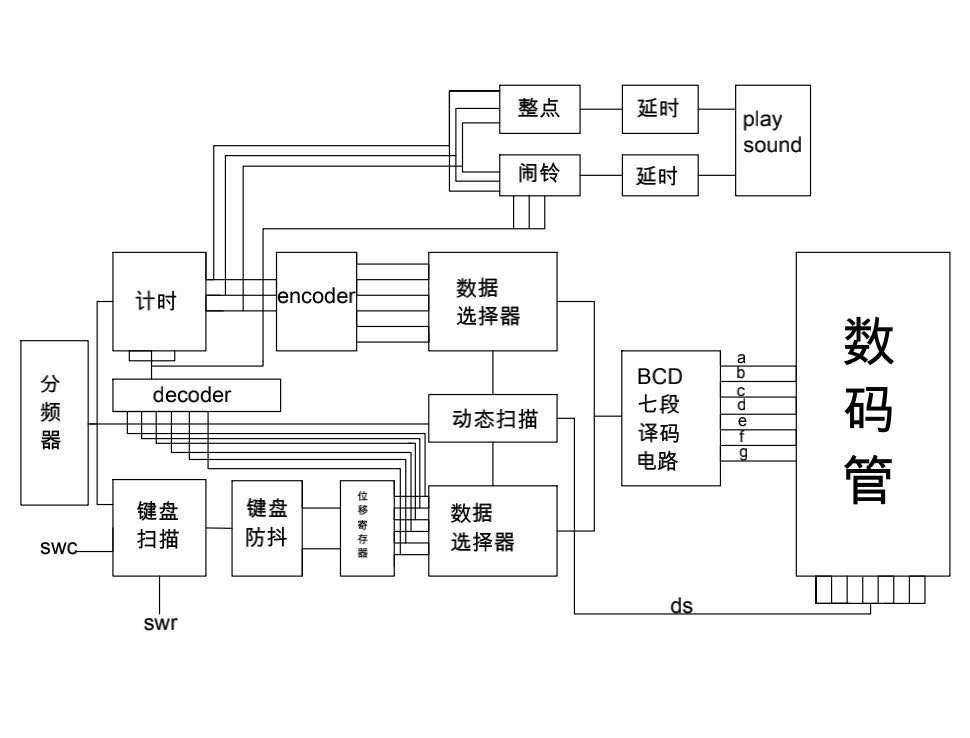
3．校时功能：对小时、分钟和秒能手动调整，以校准时间。

扩展要求：

1．具有整点报时的功能：每逢整点产生“嘀嘀嘀嘀嘟”四短一长的报时音。

2．具有闹钟功能：能在设定的时间发出闹铃音

**三．实验原理**



**动态数码管的实现主要是依靠动态扫描显示，也可以称之为扫描显示。其主要特点是将所有位数码管的段选线并联在一起（称为段选信号），并为每一位数码管的公共端增加选通控制信号（称为位选信号）。当输出字形码时，所有数码管都会接收到相同的字形码（段选信号），但究竟是哪个数码管显示出字形，取决于位选信号，所以只需要将显示的数码管的选通控制打开，该数码管就显示出字形，没有选通的数码管就不会亮。虽然在某一时刻只有一位LED数码管是点亮的，但只要扫描频率足够高（至少大于25 Hz），就可以利用发光管的余晖和人眼视觉暂留作用，从直观上给我们的感觉这些数码管是同时点亮的，这就是动态扫描显示的原理。**

**实验中所谓的时钟，实际上就是计数器的应用。以开发板上的晶振时钟为时间基准，给出一个50mhz的频率，通过设计的分频器模块进行分频，得到1hz，50hz以及1khz的信号脉冲。得到1hz的脉冲信号作为秒的脉冲，用模60的计数器构成秒的计数单元，记到60即进行清零并且进位到分，其中分计数器也是模60的计数器，都是由模6和模10组成的。每记满60分，则会进位到小时。小时的计数器就是模24的计数器，这整个计时电路的脉冲信号均为1hz。然后将这些数通过编码器通过对10取余以及除10操作分别得到对应的个位数以及十位数的数据，通过数据选择器传到BCD七段译码电路中，分频器的1khz的信号脉冲给到动态扫描，实现动态数码管显示时间，从左到右依次显示秒，分，以及小时，均为两位数。数字钟的校准以及闹铃的输入都是通过键盘输入，先使用键盘扫描识别到所输入的数字，（定义4位输入变量SWR和SWC作为行扫描和列扫描的变量，16位的keys作为键盘扫描的输出变量。初始化SWR为1110，电路从最低一行开始扫描，定义4位寄存器型变量keys3、keys2、keys1和keys0分别对应SWR3 – SWR0，并将这四个变量连接成16位的keys输出变量作为扫描的结果（每个按键是否按下）。在always块描述每当时钟信号clk为上升沿时，SWR左移循环计数，从第一行到第四行再回到第一行循环扫描，在case块中将每行扫描时所检测到的列SWC对应位取反，并将第一行到第四行的取反结果依次赋值给keys0 ~ keys3，最终再将keys0 ~ keys3组合成输出变量keys，从而扫描出哪些按键被按下。）为了避免键盘抖动，这里加入了键盘防抖的模块，更好的实现数据输入的准确性。输入的数据将给到位移寄存器中，键盘连续输入的数为小时——分钟——秒钟，这六个数据将会给到计时模块，在这之间连有一个译码器，这些输入的数将会进行判断，判断是否为闹铃，或者是校准时间；若为校准时间，则这些输入的数则给到计时模块，使时间紧接着跑；若给到闹铃模块，将会进行时间的判断，当时间与输入时间，即闹铃时间一致时，则判断出闹铃到点，电路的蜂鸣器模块将会起到作用，开始响音乐。整点的时候电路也会报时，这两个报时的声音是不同的，通过不同的音调实现不同的音乐。以上就是本次实验的原理以及设计思路。**

**四．详细设计**

1、电路实现及仿真（只提供新加入以及改动的模块）

1.分频器

module frequency\_divider\_074124 (clk\_50mhz,reset,clk\_1hz,clk\_2hz,clk\_50hz,clk\_1khz);

input reset,clk\_50mhz;

output clk\_1hz,clk\_2hz,clk\_50hz,clk\_1khz;

reg clk\_1hz,clk\_2hz,clk\_50hz,clk\_1khz;

reg[31:0]cnt\_1hz,cnt\_2hz,cnt\_50hz,cnt\_1khz;

parameter N\_1hz=50\_000\_000;

parameter N\_2hz=25\_000\_000;

parameter N\_50hz=1\_000\_000;

parameter N\_1khz=50\_000;

//parameter N\_1hz=50;

//parameter N\_2hz=24;

//parameter N\_50hz=10;

//parameter N\_1khz=6;

always@(posedge clk\_50mhz)

begin

if(!reset)

begin

cnt\_1hz<=1'b0;

cnt\_2hz<=1'b0;

cnt\_50hz<=1'b0;

cnt\_1khz<=1'b0;

clk\_1hz<=1'b0;

clk\_2hz<=1'b0;

clk\_50hz<=1'b0;

clk\_1khz<=1'b0;

end

//1hz

if(cnt\_1hz<N\_1hz/2-1)

cnt\_1hz<=cnt\_1hz+1'b1;

else

begin

cnt\_1hz<=1'b0;

clk\_1hz<=~clk\_1hz;

end

//2hz

if(cnt\_2hz<N\_2hz/2-1)

cnt\_2hz<=cnt\_2hz+1'b1;

else

begin

cnt\_2hz<=1'b0;

clk\_2hz<=~clk\_2hz;

end

//50hz

if(cnt\_50hz<N\_50hz/2-1)

cnt\_50hz<=cnt\_50hz+1'b1;

else

begin

cnt\_50hz<=1'b0;

clk\_50hz<=~clk\_50hz;

end

//1khz

if(cnt\_1khz<N\_1khz/2-1)

cnt\_1khz<=cnt\_1khz+1'b1;

else

begin

cnt\_1khz<=1'b0;

clk\_1khz<=~clk\_1khz;

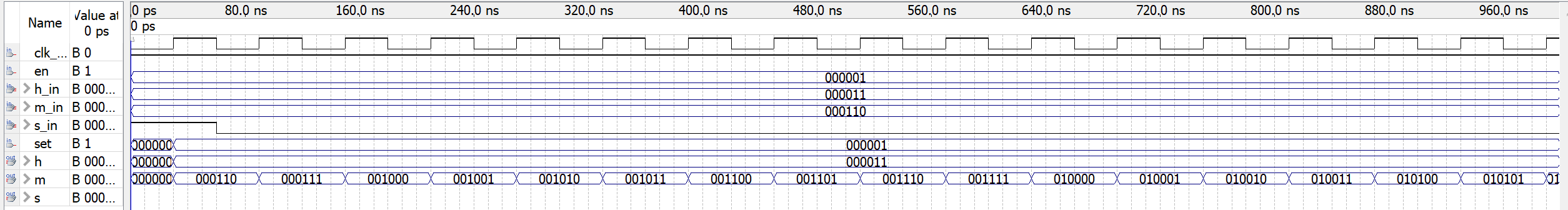
end

end

Endmodule

1. 计时模块

计时部分的功能需要精确到秒的两位数、分的两位数以及小时的两位数，等输入变量与其一一对应，秒表主要的运行原理是用计数器使每个输入变量随着clk信号的变化逐步加1，记到60即进行清零并且进位到分，其中分计数器也是模60的计数器，都是由模6和模10组成的。每记满60分，则会进位到小时。



module timecounter\_074124 (en,set,s,m,h,clk\_1hz,s\_in,m\_in,h\_in);

input clk\_1hz,set,en;

input [5:0]s\_in,m\_in,h\_in;

output [5:0]s,m,h;

reg [5:0]s,m,h;

always @ (posedge clk\_1hz)

begin

if(en)

begin

if(set)

begin

s=s\_in;

m=m\_in;

h=h\_in;

end

else

begin

if(s>=6'd59)

begin

m=m+1;

s=6'd0;

end

else

s=s+1;

if(m>=6'd60)

begin

h=h+1;

m=6'd0;

end

if(h>=6'd24)

begin

h=6'd0;

end

end

end

end

endmodule

1. 编码器

为了实现将六位二进制数变为十位和个位。

module encoder\_074124 (in,out0,out1);

input [5:0]in;

output [3:0]out0,out1;

reg [3:0]out0,out1;

always @ (in)

begin

out0=in%10;

out1=in/10;

end

endmodule

1. 译码器

module decoder\_074124 (in0,in1,out);

input [3:0]in0,in1;

output [5:0]out;

wire [6:0]temp;

assign temp=in0+in1\*10;

assign out=temp[5:0];

endmodule

1. 数据选择器

module multiplexers\_074124(out,en,sel,in1,in2,in3,in4,in5,in6,in7,in8);

input en;

input [2:0]sel;

input [3:0]in1,in2,in3,in4,in5,in6,in7,in8;

output [3:0]out;

reg [3:0]out;

always @(sel or in8 or in7 or in6 or in5 or in4 or in3 or in2 or in1 or en)

begin

if(~en)

out=4'b1111;

else

case(sel)

3'b000:out=in1;

3'b001:out=in2;

3'b010:out=in3;

3'b011:out=in4;

3'b100:out=in5;

3'b101:out=in6;

3'b110:out=in7;

3'b111:out=in8;

default:out=4'bx;

endcase

end

endmodule

1. 动态扫描电路

module dtsm\_074124(clk,ds,select);

input clk;

output [7:0]ds;

output [2:0]select;

reg [7:0]ds;

reg [2:0]select;

reg [2:0]state;

parameter s0=0,s1=1,s2=2,s3=3,s4=4,s5=5,s6=6,s7=7;

always @ (state)

begin

case(state)

s0: {select[2:0],ds[7:0]}=11'b000\_1111\_1110;

s1: {select[2:0],ds[7:0]}=11'b001\_1111\_1101;

s2: {select[2:0],ds[7:0]}=11'b010\_1111\_1011;

s3: {select[2:0],ds[7:0]}=11'b011\_1111\_0111;

s4: {select[2:0],ds[7:0]}=11'b100\_1110\_1111;

s5: {select[2:0],ds[7:0]}=11'b101\_1101\_1111;

s6: {select[2:0],ds[7:0]}=11'b110\_1011\_1111;

s7: {select[2:0],ds[7:0]}=11'b111\_0111\_1111;

default: {select[2:0],ds[7:0]}=11'b000\_1111\_1111;

endcase

end

always @ (posedge clk)

begin

case(state)

s0:state<=s1;

s1:state<=s2;

s2:state<=s3;

s3:state<=s4;

s4:state<=s5;

s5:state<=s6;

s6:state<=s7;

s7:state<=s0;

default:state<=s0;

endcase

end

endmodule

1. 位移寄存器

module shift\_register\_074124 (clk,clrn,dsl,dsr,s0,s1,a,b,c,d,qa,qb,qc,qd);

input clk,clrn,s0,s1;

input [3:0]a,b,c,d,dsl,dsr;

output [3:0]qa,qb,qc,qd;

reg [3:0]qa,qb,qc,qd;

always @ (posedge clk or negedge clrn)

begin

if(clrn==0) {qd,qc,qb,qa}=4'b0000\_0000\_0000\_0000;

else

case({s1,s0})

2'b00:{qd,qc,qb,qa}<={qd,qc,qb,qa};

2'b11:{qd,qc,qb,qa}<={d,c,b,a};

2'b01:{qd,qc,qb,qa}<={dsr,qd,qc,qb};

2'b10:{qd,qc,qb,qa}<={qc,qb,qa,dsl};

endcase

end

endmodule

1. 键盘防抖

module jpfd\_074124 (clk\_50hz,en,key\_in,key\_out,flag);

input clk\_50hz,en;

input [3:0]key\_in;

output [3:0]key\_out;

output flag;

reg flag;

reg [3:0]key\_out,key\_last;

always @ (posedge clk\_50hz)

begin

if(en)

begin

key\_out=key\_in;

if(key\_out==key\_last) flag=1;

else flag=0;

key\_last=key\_out;

end

else

;

end

endmodule

1. 键盘扫描

module jpsm\_074124 (clk\_1khz,en,swc,key,swr,flag);

input clk\_1khz,en;

input [3:0]swc;

output [3:0]key,swr;

output flag;

reg [3:0]key,swr;

reg flag;

always @(posedge clk\_1khz)

begin

if(~en) swr=4'b1110;

else

begin

case({swr,swc})

8'b1110\_1110:{key,flag}=5'b00001;

8'b1110\_1101:{key,flag}=5'b00011;

8'b1110\_1011:{key,flag}=5'b00101;

8'b1110\_0111:{key,flag}=5'b00111;

8'b1101\_1110:{key,flag}=5'b01001;

8'b1101\_1101:{key,flag}=5'b01011;

8'b1101\_1011:{key,flag}=5'b01101;

8'b1101\_0111:{key,flag}=5'b01111;

8'b1011\_1110:{key,flag}=5'b10001;

8'b1011\_1101:{key,flag}=5'b10011;

8'b1011\_1011:{key,flag}=5'b10101;

8'b1011\_0111:{key,flag}=5'b10111;

8'b0111\_1110:{key,flag}=5'b11001;

8'b0111\_1101:{key,flag}=5'b11011;

8'b0111\_1011:{key,flag}=5'b11101;

8'b0111\_0111:{key,flag}=5'b11111;

default:{key,flag}={key,1'b0};

endcase

case(swr)

4'b1110:swr=4'b1101;

4'b1101:swr=4'b1011;

4'b1011:swr=4'b0111;

4'b0111:swr=4'b1110;

default:swr=4'b1110;

endcase

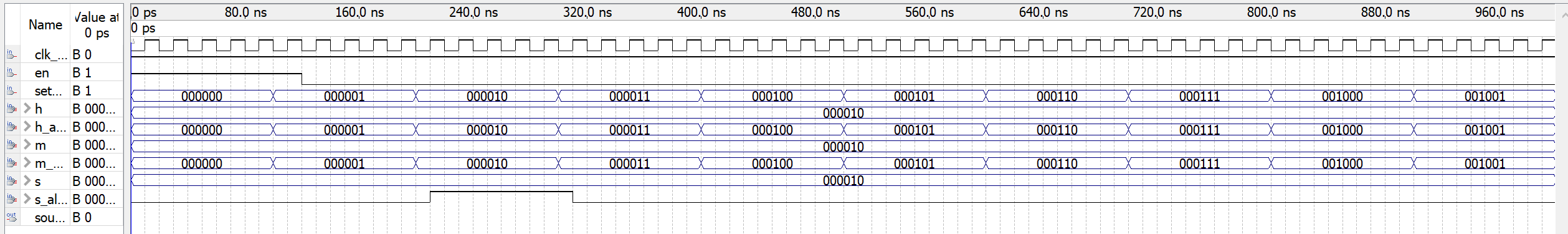
end

end

endmodule

1. 闹铃模块

当设定闹铃的电平为高电平时，既可以手动输入设定的时间，之后计时器继续计时，当计时的时间和设定的闹钟的时间一致的时候，将会输出一位sound判断是否到该时间。



module alarm\_074124 (clk\_1hz,en,set\_alarm,s,m,h,s\_alarm\_set,m\_alarm\_set,h\_alarm\_set,sound\_alarm);

input clk\_1hz,en,set\_alarm;

input [5:0]s,m,h,s\_alarm\_set,m\_alarm\_set,h\_alarm\_set;

output sound\_alarm;

reg sound\_alarm;

reg [5:0]s\_alarm,m\_alarm,h\_alarm;

always @ (posedge clk\_1hz)

begin

if(en)

begin

if(~set\_alarm)

begin

if(s\_alarm==s&&m\_alarm==m&&h\_alarm==h)

sound\_alarm=1;

else

sound\_alarm=0;

end

else

begin

{s\_alarm,m\_alarm,h\_alarm}={s\_alarm\_set,m\_alarm\_set,h\_alarm\_set};

end

end

else

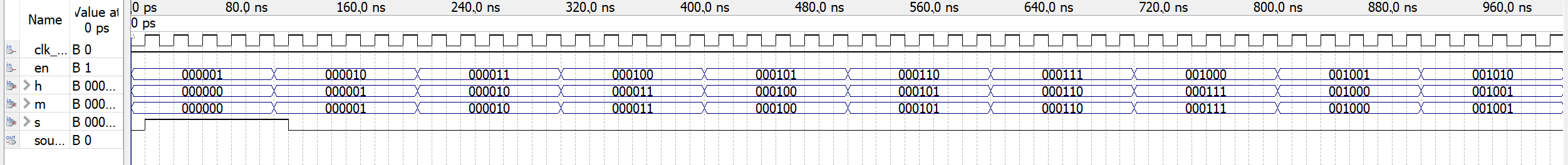
sound\_alarm=0;

end

endmodule

1. 整点报时模块

当计时器的时间到达整点的时候，例如13:00:00，即分钟和秒数均为00的时候，将会使sound变为高电平。



module zhengdian\_074124 (clk\_1hz,en,s,m,h,sound\_zhengdian);

input clk\_1hz,en;

input [5:0]s,m,h;

output sound\_zhengdian;

reg sound\_zhengdian;

always @ (posedge clk\_1hz)

begin

if(en)

begin

if(s==0&&m==0&&h!=0)//为了使刚开机时不响，将小时为0调为不响。

sound\_zhengdian=1;

else

sound\_zhengdian=0;

end

else

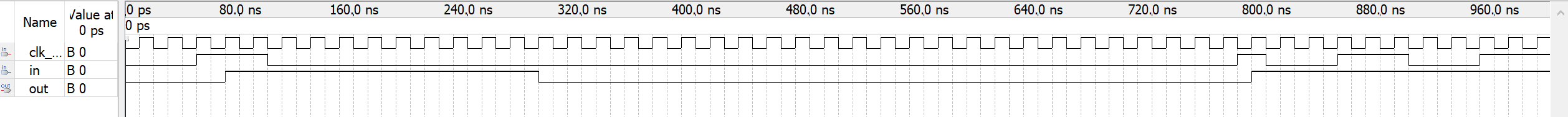
sound\_zhengdian=0;

end

endmodule

1. 延时模块（为了让声音延长）

下载到实验箱上的时候，若时间过短，则无法听见声音，所以需要建立一个模块，使发声时间变长，即可以听见声音的发出。



module yanshi\_074124 (clk\_1hz,in,out);

input in,clk\_1hz;

output out;

reg out;

reg [4:0]cnt;

parameter N=10;

always @(posedge clk\_1hz)

begin

if(in==1)

begin

cnt<=1;

out<=in;

end

else

begin

if(cnt>=N)

out<=0;

else

cnt<=cnt+1;

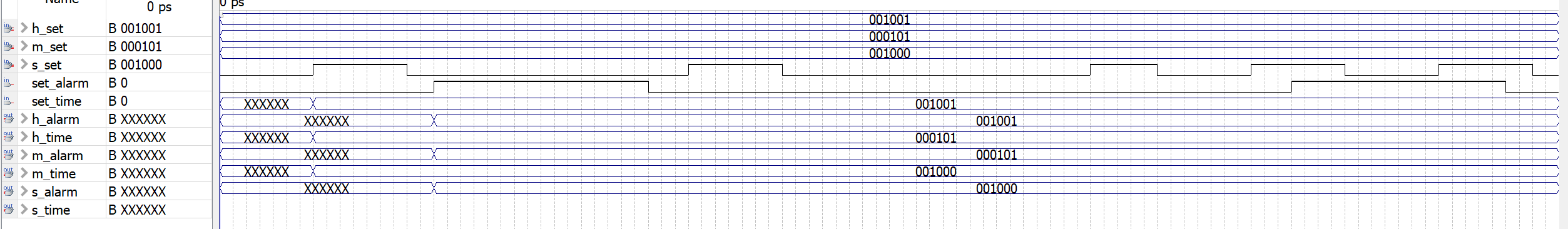
end

end

endmodule

1. 判断输入是校准还是闹铃模块

该模块判断所输入的数据是给到闹铃模块还是校准模块，当校准时间的使能端电平为高的时候，则输入的数据给到校准时间的模块，若闹铃的使能端为高电平时，则输入的数据给到设定的闹铃的时间。



module alarmortime(s\_set,m\_set,h\_set,h\_time,m\_time,s\_time,s\_alarm,m\_alarm,h\_alarm,set\_time,set\_alarm);

input set\_time,set\_alarm;

input [5:0]s\_set,m\_set,h\_set;

output [5:0]h\_time,m\_time,s\_time,s\_alarm,m\_alarm,h\_alarm;

reg [5:0]h\_time,m\_time,s\_time,s\_alarm,m\_alarm,h\_alarm;

always @(s\_set,m\_set,h\_set,set\_time,set\_alarm)

begin

if(set\_time==1&&set\_alarm==0)

begin

s\_time=s\_set;

m\_time=m\_set;

h\_time=h\_set;

end

else if(set\_time==0&&set\_alarm==1)

begin

s\_alarm=s\_set;

h\_alarm=h\_set;

m\_alarm=m\_set;

end

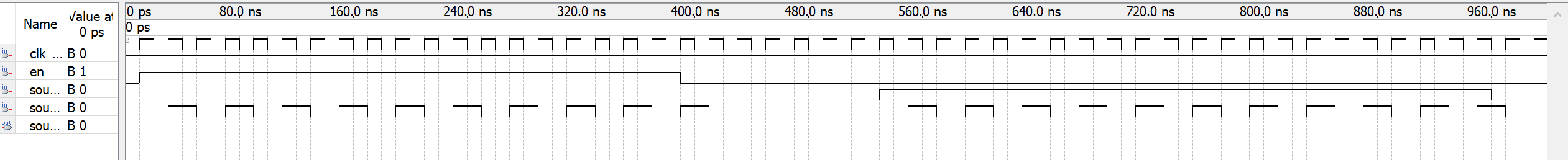
else;

end

endmodule

1. 发声模块

当sound——alarm为高电平时，蜂鸣器发出一段音调，不同的音对应不同的音高；当sound——zhengdian为高电平时，则蜂鸣器发出滴滴滴嘟的四声，两者被区分开来，便于识别。



module playsound\_074124 (clk\_50mhz,en,sound\_alarm,sound\_zhengdian,sound);

input clk\_50mhz,en,sound\_alarm,sound\_zhengdian;

output sound;

reg sound;

reg[10:0] state;

reg[25:0] count,count\_end;

reg[25:0] count\_countinue;

parameter M\_1=25'd47774,

M\_2=25'd42568,

M\_3=25'd37919,

M\_4=25'd35791,

M\_5=25'd31888,

M\_6=25'd28409,

M\_7=25'd25309,

kong=25'd50\_000\_000;

parameter TIME=50\_000\_000; //控制每一个音的长短

always @(posedge clk\_50mhz)

begin

if(en&(sound\_alarm|sound\_zhengdian))

begin

if(count>=count\_end)

begin

count<=25'h0;

sound<=!sound;

end

else count<=count+1'b1;

if(count\_countinue<TIME)

count\_countinue=count\_countinue+1'b1;

else

begin

count\_countinue= 4'd0;

if({sound\_alarm,sound\_zhengdian}==2'b01)//zheng dian bao shi

begin

if(state>=8'd10)

state=8'd0;

else

state=state+1'b1;

case(state)

8'd0:count\_end=M\_3;

8'd1:count\_end=kong;

8'd2:count\_end=M\_3;

8'd3:count\_end=kong;

8'd4:count\_end=M\_3;

8'd5:count\_end=kong;

8'd6:count\_end=M\_3;

8'd7:count\_end=kong;

8'd8:count\_end=M\_1;

8'd9:count\_end=kong;

default:count\_end=16'd0;

endcase

end

else if({sound\_alarm,sound\_zhengdian}==2'b10) //naozhong

begin

if(state>=8'd10)

state=8'd0;

else

state=state+1'b1;

case(state)

8'd0:count\_end=M\_1;

8'd1:count\_end=M\_1;

8'd2:count\_end=M\_5;

8'd3:count\_end=M\_5;

8'd4:count\_end=M\_6;

8'd5:count\_end=M\_6;

8'd6:count\_end=M\_5;

8'd7:count\_end=kong;

8'd8:count\_end=M\_4;

8'd9:count\_end=M\_4;

default:count\_end=16'd0;

endcase

end

else ;

end

end

else

begin

sound=0;

state=0;

count=0;

count\_end=0;

count\_countinue=0;

end

end

endmodule

1. BCD七段译码电路

module led\_g074124 (data,a,b,c,d,e,f,g);

input [3:0]data;

output a,b,c,d,e,f,g;

reg a,b,c,d,e,f,g;

always @(data)

begin

case(data)

4'b0000:{a,b,c,d,e,f,g}=7'b1111110;

4'b0001:{a,b,c,d,e,f,g}=7'b0110000;

4'b0010:{a,b,c,d,e,f,g}=7'b1101101;

4'b0011:{a,b,c,d,e,f,g}=7'b1111001;

4'b0100:{a,b,c,d,e,f,g}=7'b0110011;

4'b0101:{a,b,c,d,e,f,g}=7'b1011011;

4'b0110:{a,b,c,d,e,f,g}=7'b1011111;

4'b0111:{a,b,c,d,e,f,g}=7'b1110000;

4'b1000:{a,b,c,d,e,f,g}=7'b1111111;

4'b1001:{a,b,c,d,e,f,g}=7'b1111011;

4'b1010:{a,b,c,d,e,f,g}=7'b1110111;

4'b1011:{a,b,c,d,e,f,g}=7'b0011111;

4'b1100:{a,b,c,d,e,f,g}=7'b1001110;

4'b1101:{a,b,c,d,e,f,g}=7'b0111101;

4'b1110:{a,b,c,d,e,f,g}=7'b1001111;

4'b1111:{a,b,c,d,e,f,g}=7'b1000111;

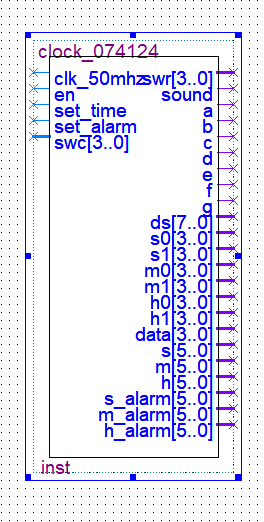
default:{a,b,c,d,e,f,g}=7'b0000000;

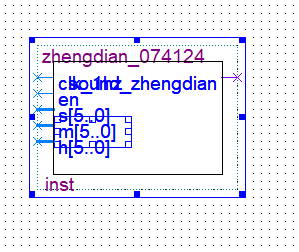
endcase

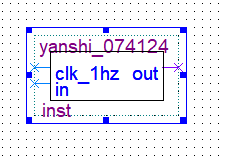
end

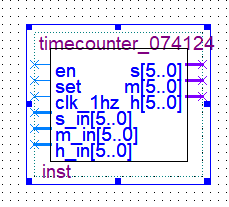
endmodule

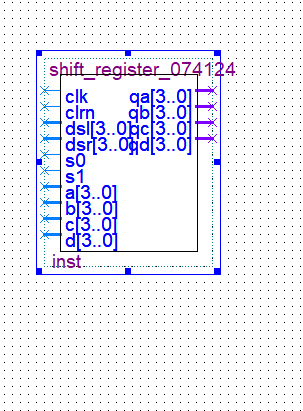
1. 封装图

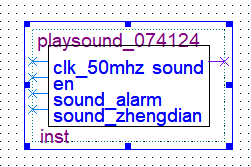
此封装图为顶层模块封装图，输入为50mhz时钟信号，en为使能端，set\_time和set\_alarm分别接开关，swc为按键。输出中很多是为了调试方便，真正有用的是swr,a,b,c,d,e,f,g,ds.

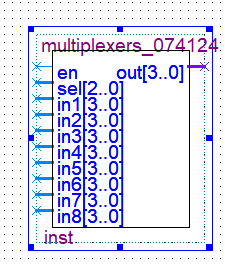
此封装图为整点报时模块，输入端为en和时钟信号以及小时分钟秒。输出为整点报时的有效与否。为新增模块。

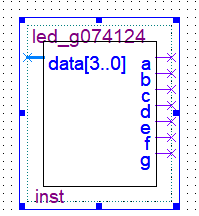
为了加长声音的时间，为新增模块。

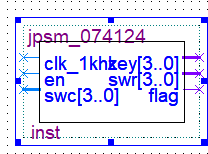
此封装图为计时器模块，与秒表的一样。

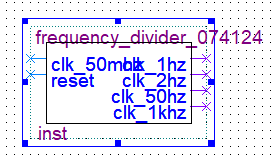
此为位移寄存器模块，与之前一样。

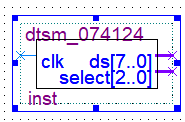
此为发出声音模块，输入为50mhz时钟信号，使能端，以及整点或者闹铃的有效与否。为新增模块。

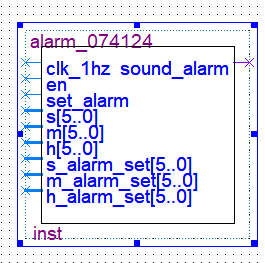
数据选择器，为原先模块。

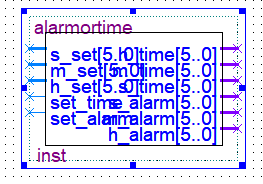
为原先模块，数码管模块

键盘扫描模块，之前模块。

分频器模块，为之前的模块

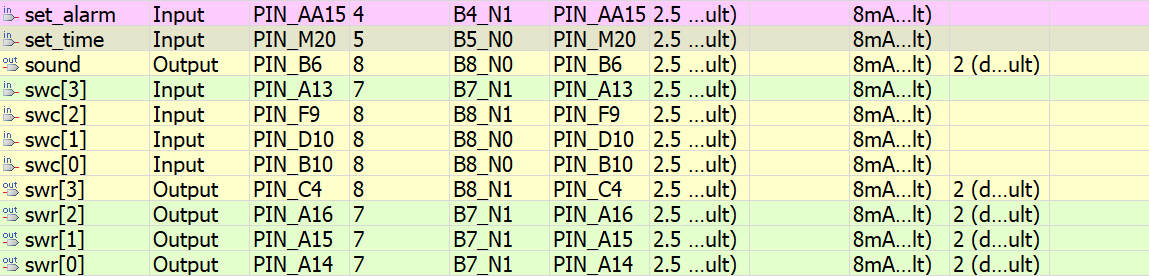
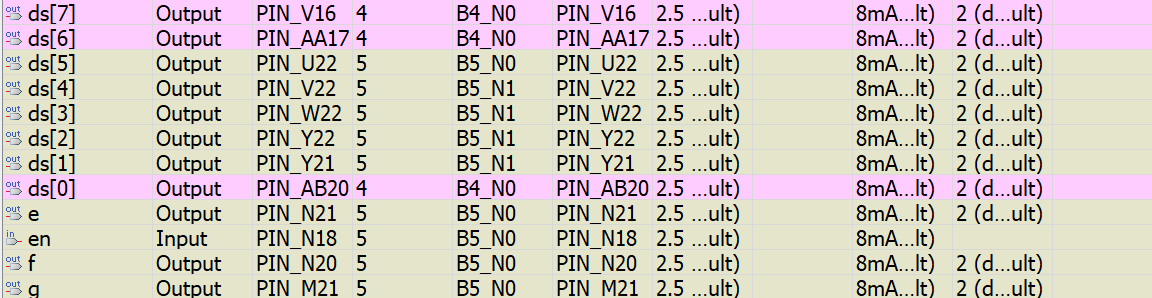
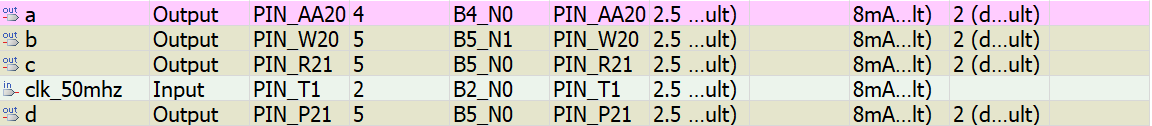
动态扫描模块，为之前模块。

闹领模块，新加入模块。

判断模块，新加入模块。

**五．实验结果**

（1）引脚分配



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **平台端口** | SWR3 | SWR2 | SWR1 | SWR0 | 蜂鸣器 |
| **引脚编号** | PIN\_C4 | PIN\_A16 | PIN\_A15 | PIN\_A14 | PIN\_B6 |
| **节点名称** | SWR[3] | SWR[2] | SWR[1] | SWR[0] | sound |
| **平台端口** | DS8 | DS7 | DS6 | DS5 | DS4 | DS3 | DS2 | SW3 |
| **引脚编号** | V16 | AA17 | U22 | V22 | W22 | Y22 | Y21 | PIN\_AA15 |
| **节点名称** | ds[7] | ds[6] | ds[5] | ds[4] | ds[3] | ds[2] | ds[1] | Set\_alarm |
| **平台端口** | LA | LB | LC | LD | LE | LF | LG | SW2 |
| **引脚编号** | PIN\_AA20 | PIN\_W20 | PIN\_R21 | PIN\_P21 | PIN\_N21 | PIN\_N20 | PIN\_M21 | PIN\_M20 |
| **节点名称** | a | b | c | d | e | f | g | Set\_time |
| **平台端口** | SWC3 | SWC2 | SWC1 | SWC0 | DS1 | LH | T1 | SW1 |
| **引脚编号** | PIN\_A13 | PIN\_F9 | PIN\_D10 | PIN\_B10 | AB20 | M19 | PIN\_T1 | PIN\_N18 |
| **节点名称** | SWC[3] | SWC[2] | SWC[1] | SWC[0] | ds[0] | h | Clk\_50mhz | en |

（2）实验现象

打开实验箱，将程序下载到实验箱中去，将第一个开关给高电平，数码管开始计时，从00-00-00开始跑，实现了基本的数字钟。此时为了校准时间，将第二个开关拨至高电平，即可连续输入现在的准确时间，例如我们输入125803，则实现时间的校准为12:58:03；输入之后我们把开关拨回低电平，则数码管上的数字将会从125803开始接着跑，秒数满60进1给分钟，分钟满60进1给小时，即12:59:59的下一秒将会显示13:00:00.与此同时，13:00:00为整点时间，所以蜂鸣器将会起到作用，发出滴——滴——滴——嘟的声音。接下来我们调试闹铃的模块，我们将第三个开关拨置高电平，连续输入时间，假设我们输入12:43:09，再将开关拉下来，则时钟回到原先的时间，当跑到的时间到达12:43:09秒的时候，蜂鸣器发出声音，响起一段音乐。至此，基本功能以及拓展功能全部实现。

1. 任务分配（19030419 王昕）

1.分频器

2.计时模块

3.编码器

4，译码器

5.数据选择器

6.动态扫描电路

7位移寄存器

8键盘防抖

9键盘扫描

10.闹铃模块

11.整点报时模块

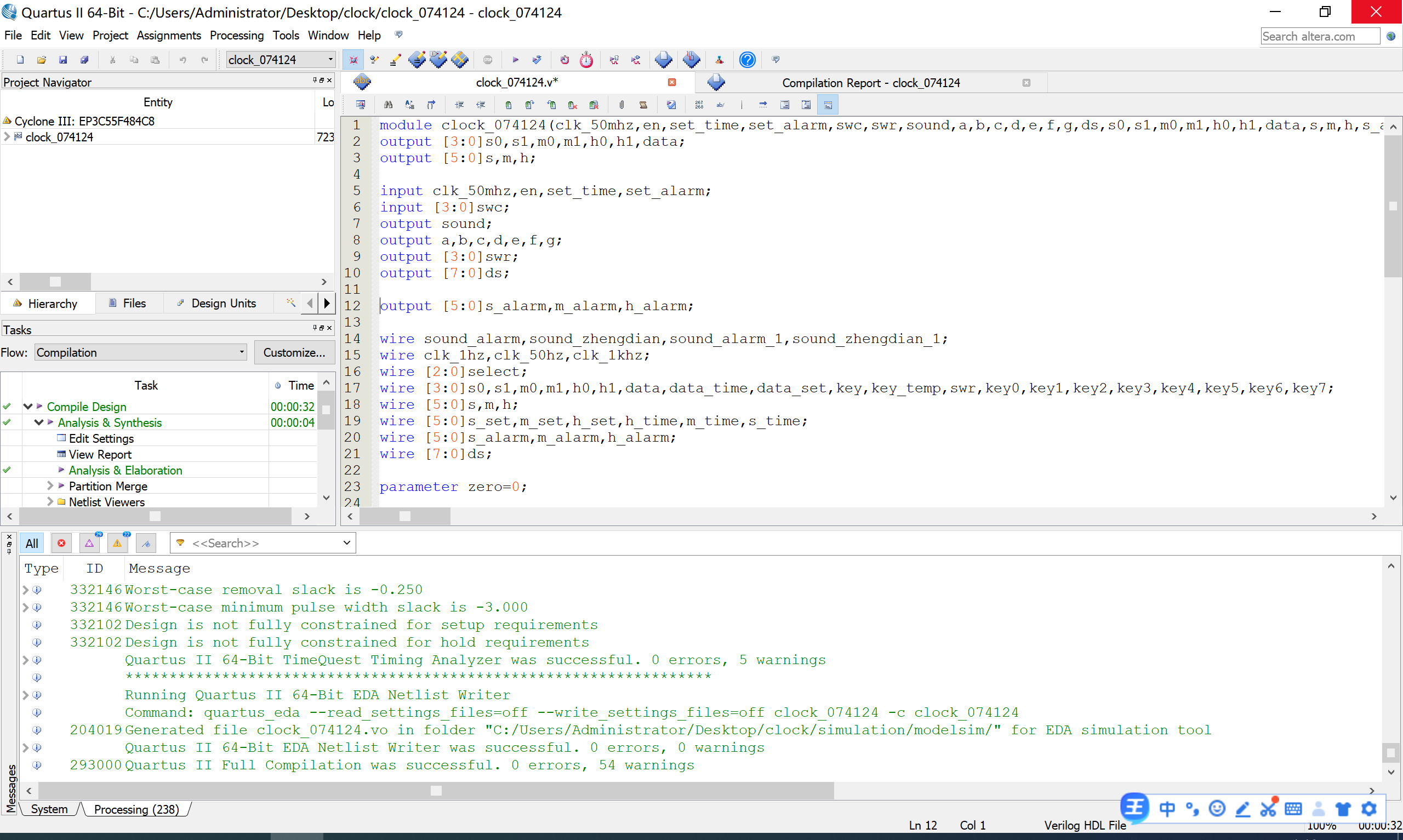
12.延时模块（为了让声音延长）

13.判断输入是校准还是闹铃模块

14.发声模块

15.BCD七段译码电路

七．顶层电路附图



附图：顶层模块