|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **学 期** | 2020-2021学年第1学期 | | | | **实验日期** | | 2020.12.1 | | |
| **学 院** | 信息学部 | | | | **专 业** | | 计算机类 | | |
| **班 级** | 190741 | | **学 号** | | 19030419 | | **姓 名** | | 王昕 |
| **组 号** | 24 | | **学 号** | | 19074104 | | **姓 名** | | 王佳玮 |
| **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10：动态数码管显示 | | | | | | | | |

**一．实验目的**

1. 理解动态数码管工作原理。
2. 掌握数码管动态扫描的控制方法。
3. **实验任务及要求**

在动态数码管上显示个人学号。

要求1.系统时钟为50MHz，使用之前设计的分频器模块。

1. 完成动态扫描模块的设计。

3.将个人学号显示在动态数码管上。

任务： 利用动态数码管实现一个简易秒表

要求：

1. 系统时钟为50 MHz，使用之前设计的分频器模块。
2. 完成秒表功能所需计数器的设计；完成动态扫描模块的设计。
3. 将秒表值显示在动态数码管上。

。

**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。

由于动态扫描方式只用到1个LED译码模块，这就需要增加1个位译码控制模块来控制8个数码管依次点亮。整个电路使用的输出引脚为15个（8个位选择信号和7个段控制信号），可见这种显示方式可以降低对器件资源的占用。动态扫描的频率有一定的要求。频率太低，LED会出现闪烁现象；频率太高，由于每个LED点亮的时间太短，造成LED的亮度低，所以一般取1 ms左右为宜（频率1 kHz即可）。

**四．详细设计**

1. 电路实现

基础：将本次实验主模块分别与分频器，数据选择器，七段数字显示译码器相连，其中分频器决定电路中数码管闪烁的频率，数据选择器控制分别输出8个数字到8个数码管上，七段数字显示译码器将数据转化到数码管上显示。

拓展：根据需求需采用实际电路对应的Verilog HDL代码来实现，本实验需要用到的动态扫描部分中定义输入变量8位的ds来对数码管进行位选（低电平有效），每当clk时钟信号产生新的上升沿时，ds向左移一位(如11101111 —> 11011111)，实现数码管位选的实时转换。代码中定义的dots变量是一个8位的输入信号，其作用是控制实验台上数码管右下角位置的小数点灯的亮灭，本实验后续用到的秒表功能需要用小数点来将分、秒、毫秒等单位进行划分。为此定义新的变量h，对应数码管中第八个输出端小数点，assign h = \_ds[sel] & dots[sel]; 这句代码的含义是当数码管的位选和控制小数点位置的灯亮灭的输入信号都有效时，说明此位置的小数点应该点亮，h变量为1点亮，对应位置的小数点。

秒表部分的功能需要精确到毫秒的三位数、秒的两位数以及分的两位数，定义m1, m0, s1, s0, ms2, ms1, ms0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，利用assign语句从毫秒的千分位进行判断，当毫秒的千分位变成9的时候将其清零，同理毫秒的百分位和千分位同时为9的时候将百分位清零，清零的效果可看成秒表的进位功能，当毫秒为999时下一个时钟信号到来变为000，秒的个位加1；同理assign语句也对秒和分进行判断，当秒的十位为5且个位为9时进行清零，在秒表功能中实现进位，整个电路在clk信号的节奏下进行不断地计数、进位，最终组合出秒秒表的实际功能。

动态数码管显示Verilog HDL代码：

基础：module dtsmg\_0741\_24(in1,in2,en,clk\_50mhz,ds,a,b,c,d,e,f,g,out);

input en,clk\_50mhz;

input [3:0]in1,in2;

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

output [7:0]ds;

output [3:0] out;

wire reset;

assign reset=1'b0;

wire [2:0]xuanze;

wire [3:0]out;

wire yj;

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

fpq (clk\_50mhz,reset,clk\_1hz,clk\_2hz,clk\_50hz,clk\_1khz);

dtsmdl (ds,clk\_50mhz,xuanze);

sjxzq(out,xuanze,in2,in1,en);

BCD (out,a,b,c,d,e,f,g,yj);

endmodule

module sjxzq(out,sel,in2,in1,en);

input [2:0]sel;

input en;

input [3:0]in1,in2;

output [3:0]out;

reg [3:0]out;

parameter in8=4'b0001;//1

parameter in7=4'b1001;//9

parameter in6=4'b0000;//0

parameter in5=4'b0011;//3

parameter in4=4'b0000;//0

parameter in3=4'b0100;//4

always @(sel or in2 or in1 or en)

if (!en)

case(sel)

3'b000:out=in8;

3'b001:out=in7;

3'b010:out=in6;

3'b011:out=in5;

3'b100:out=in4;

3'b101:out=in3;

3'b110:out=in2;

3'b111:out=in1;

default:out=4'bx;

endcase

else out=4'bx;

endmodule

module dtsmdl (ds,clk,xuanze);

output [7:0] ds;

input clk;

output [2:0] xuanze;

reg [2:0]state;

reg [7:0] ds;

reg [2:0] xuanze;

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

always@(state)

begin

case(state)

s0: {xuanze[2:0],ds[7:0]}=11'b000\_0111\_1111;

s1: {xuanze[2:0],ds[7:0]}=11'b001\_1011\_1111;

s2: {xuanze[2:0],ds[7:0]}=11'b010\_1101\_1111;

s3: {xuanze[2:0],ds[7:0]}=11'b011\_1110\_1111;

s4: {xuanze[2:0],ds[7:0]}=11'b100\_1111\_0111;

s5: {xuanze[2:0],ds[7:0]}=11'b101\_1111\_1011;

s6: {xuanze[2:0],ds[7:0]}=11'b110\_1111\_1101;

s7: {xuanze[2:0],ds[7:0]}=11'b111\_1111\_1110;

default: {xuanze[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

module BCD (data,a,b,c,d,e,f,g,yj);

input [3:0] data;

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

output yj;

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

assign yj=1'b0;

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;

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

endcase

end

endmodule

module fpq (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

拓展：module miaobiao190741\_24 (clk\_50mhz,en,a,b,c,d,e,f,g,ds,data);

input clk\_50mhz,en;

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

output [7:0]ds;

wire [3:0]min0,min1,s0,s1,ms0,ms1,ms2;

output [3:0]data;

wire [5:0]s,min;

wire [9:0]ms;

wire clk\_1hz,clk\_2hz,clk\_50hz,clk\_1khz,sel;

wire [2:0]select;

wire [3:0]zero;

assign zero=4'b0000;

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

dtsm\_074124(clk\_1khz,ds,select);

timecounter\_074124 (clk\_50mhz,ms,s,min,en);

decoder\_6bto4d\_074124 (min,min0,min1);

decoder\_6bto4d\_074124 (s,s0,s1);

decoder\_10bto4d\_074124 (ms,ms0,ms1,ms2);

multiplexers\_074124(data,en,select,ms0,ms1,ms2,s0,s1,min0,min1,zero);

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

endmodule

module decoder\_6bto4d\_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

module decoder\_10bto4d\_074124 (in,out0,out1,out2);

input [9:0]in;

output [3:0]out0,out1,out2;

reg [3:0]out0,out1,out2;

always @ (in)

begin

out0=in%10;

out1=(in/10)%10;

out2=in/100;

end

endmodule

module timecounter\_074124 (clk\_1khz,ms,s,min,clr);

input clk\_1khz,clr;

output [5:0]s,min;

output [9:0]ms;

reg [31:1]cnt;

reg [5:0]s,min;

reg [9:0]ms;

always @ (posedge clk\_1khz or negedge clr)

begin

if(~clr)

begin

ms=10'd0;

s=6'd0;

min=6'd0;

//cnt=32'd0;

end

else

begin

/\*

if(cnt>=32'd25\_000) //实验用

//if(cnt>=32'd20) //波形用

begin

ms=ms+1;

cnt=32'd0;

end

else cnt=cnt+1;

\*/

if(ms>=10'd1000)

begin

s=s+1;

ms=10'd0;

end

else

ms=ms+1;

if(s>=6'd60)

begin

min=min+1;

s=6'd0;

end

if(min>=6'd60) min=6'd0;

end

end

endmodule

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

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

input [3:0]data;

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

output sel;

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

assign sel=1'b0;

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;

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

endcase

end

endmodule

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

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. 仿真

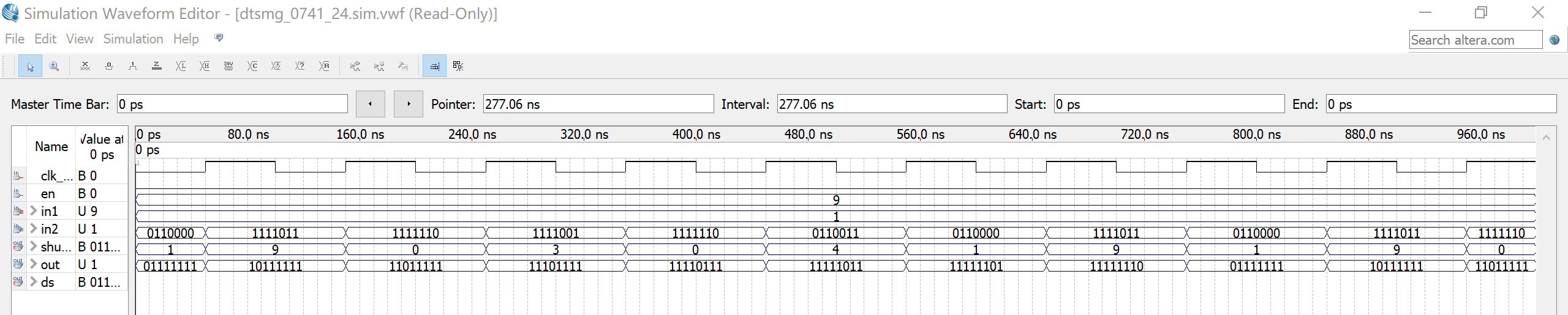


图1　动态数码管仿真波形

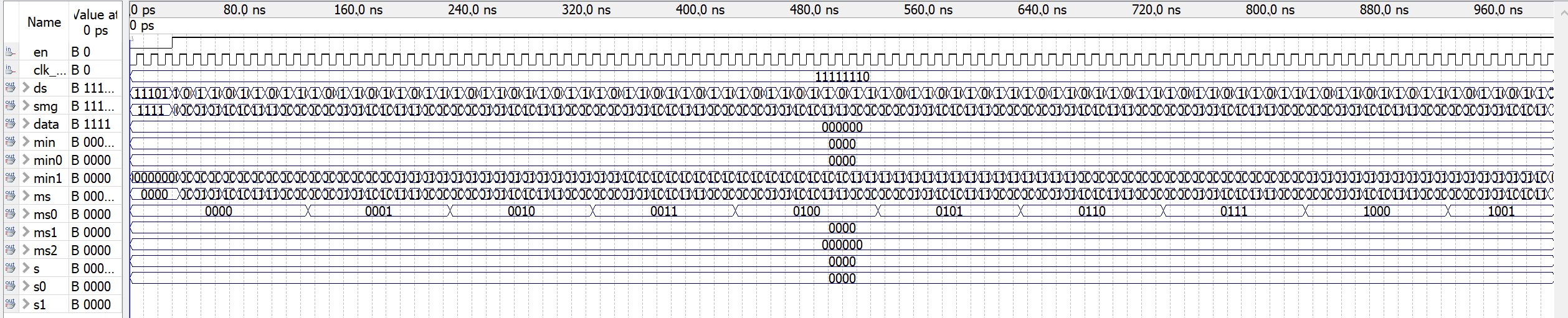
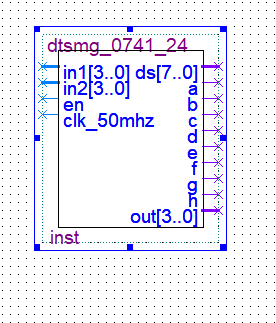


图 2　秒表基本功能仿真波形

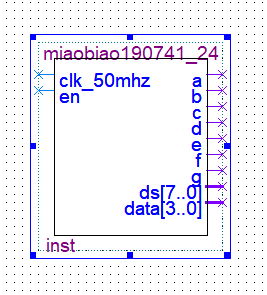
1. 封装图



输入端：en 使能端

clk时钟信号

In1in2输入学号后两位



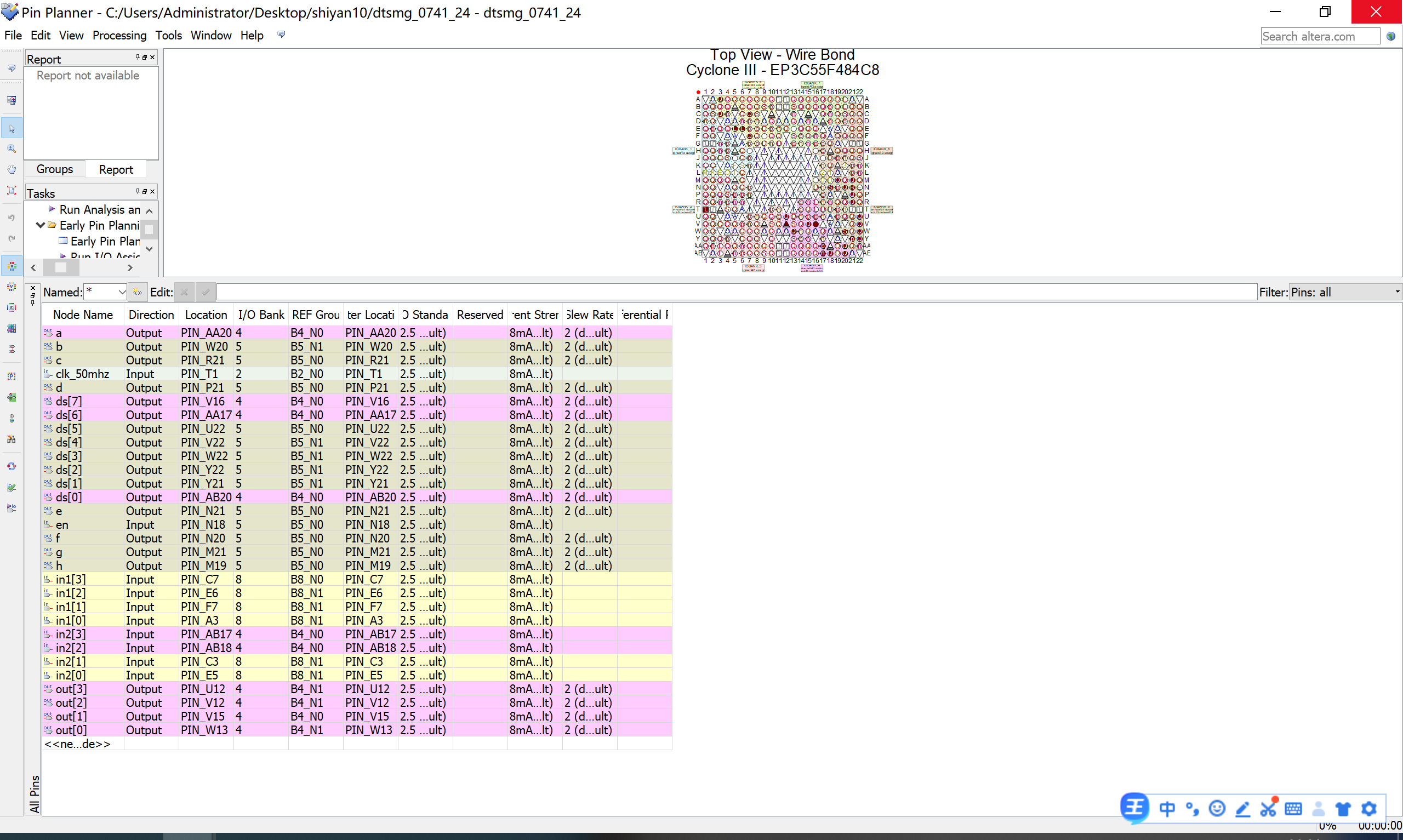
输入端：en 使能端  
clk 时钟信号

输出端：m1 – m0[3..0] 分的两位输出  
s1 – s0[3..0] 秒的两位输出  
ms2 – ms0[3..0] 秒的三位输出

**五．实验结果**

（1）引脚分配

本实验需要用到8个数码管，其位选端分配到实验台上的端口DS8 ~ DS1；数码管显示的使能开关分配到引脚E7；8个输出端分配到数码管段选信号指示灯LA ~ LH；时钟信号接试验台T1时钟信号。具体引脚分配及其编号见下：

基础

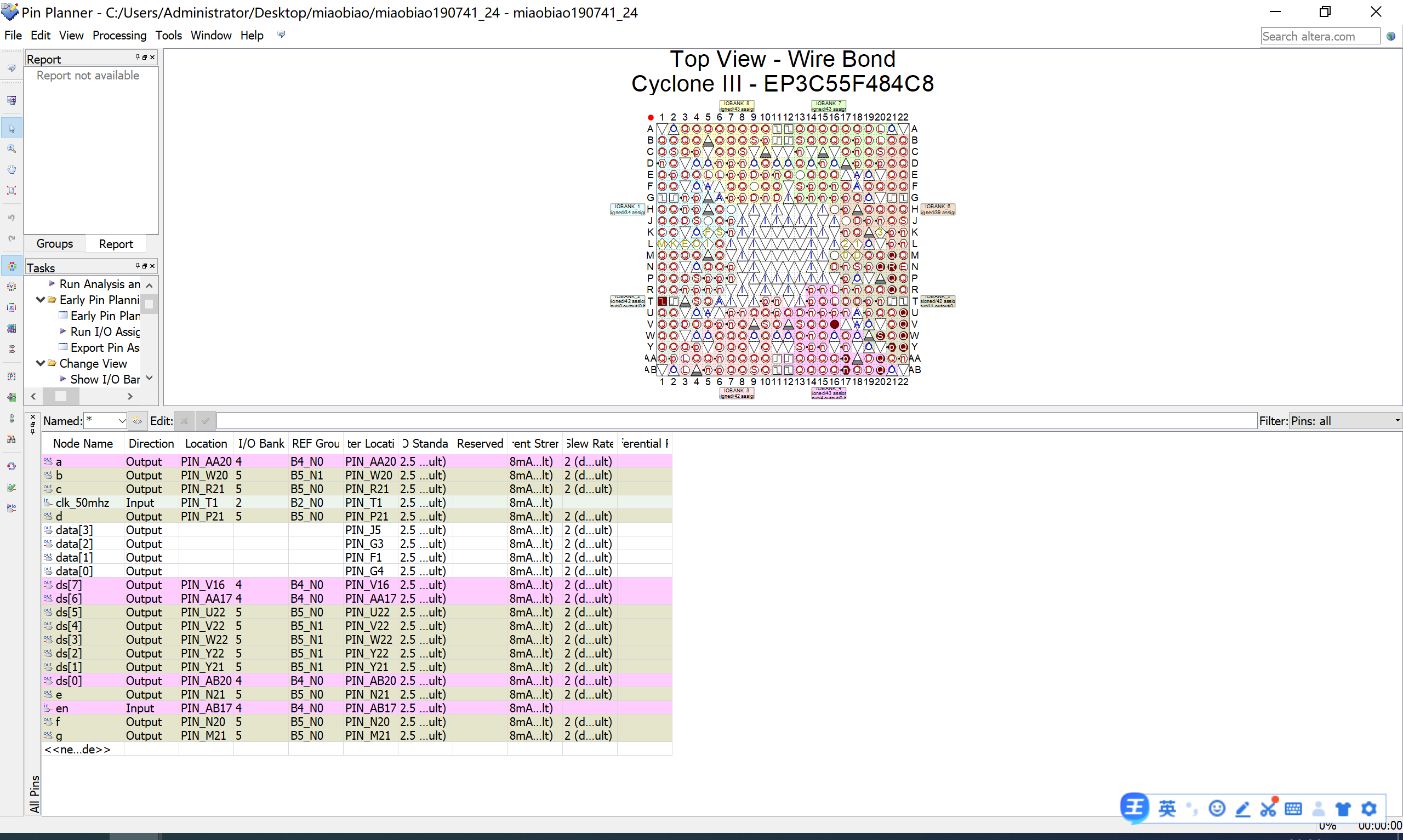
拓展

图4　引脚分配图

1. 实验现象

基础

打开实验箱，通电，下载后，将使能端只为有效，将in1in2调整成学号19，数码管上数字从左往右依次是19030419，则扫描成功。

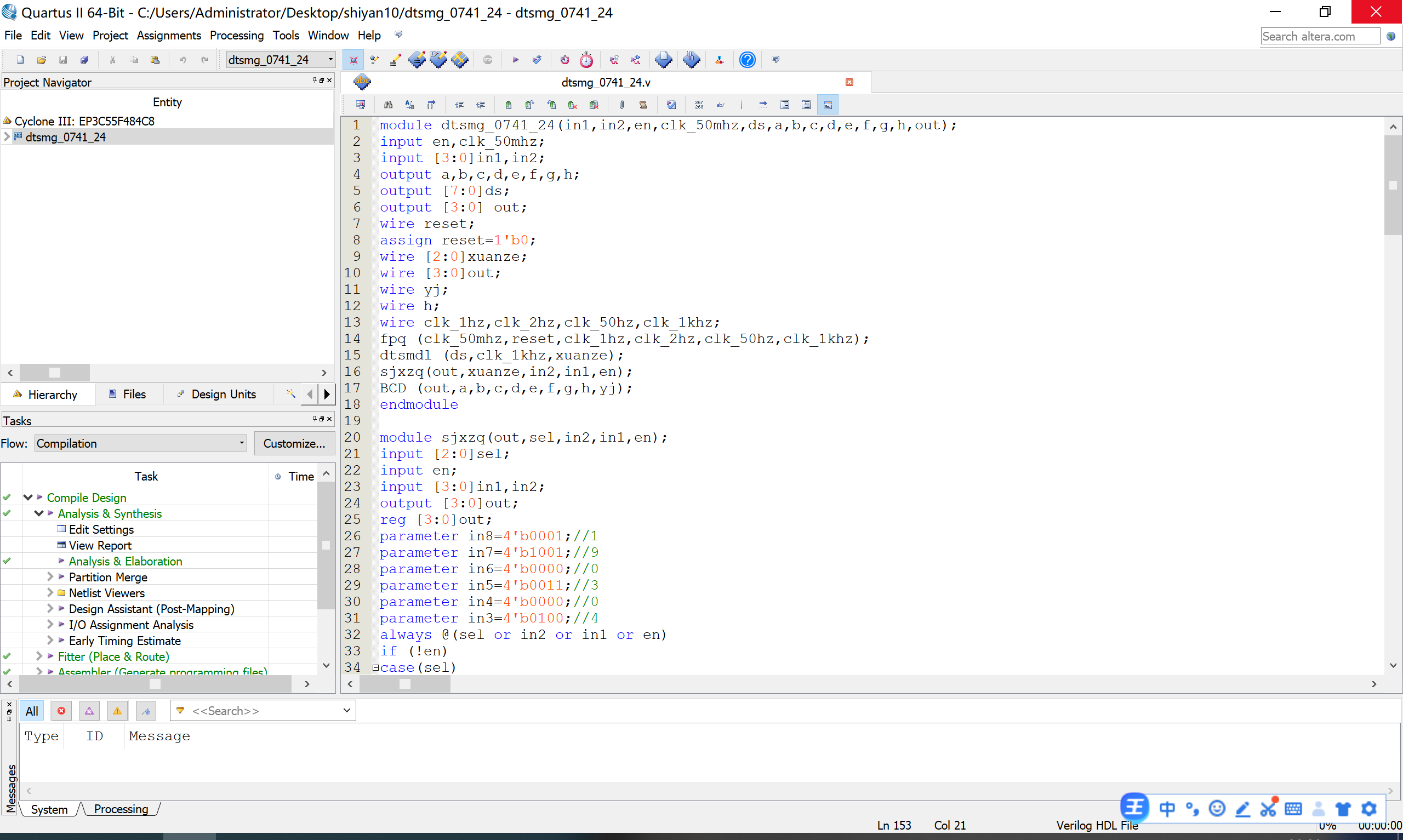
拓展

打开实验箱，通电，下载后，将使能开关拨至上，所有数码管点亮且显示数字0，进行如下操作：

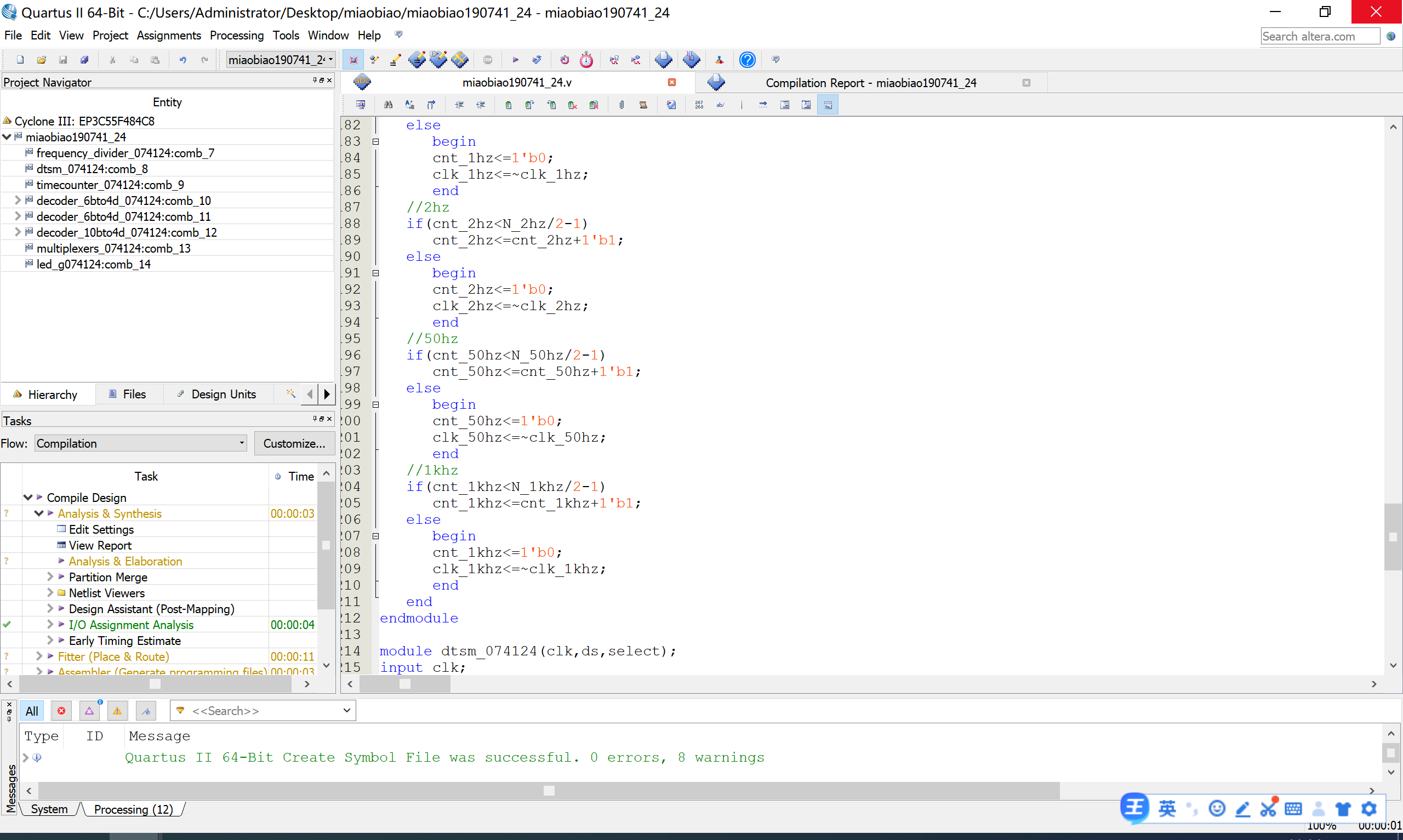
秒表开始工作，八个数码管的后七位从左到右分别是分钟的十位，个位、秒钟的十位、个位、毫秒的十分位、百分位、千分位，同时分别在分钟和秒钟、秒钟和毫秒之间点亮小数点作为单位的区分。前1分钟内秒表正常的运作，数码管应显示00.xx.xxx,其现象与正常的秒表是相同的。秒表的功能如预期运行，实验成功。实验箱断电，实验结束。

顶层附图：

基础



拓展

 附图：顶层模块