

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 数字逻辑电路设计 |
| 姓 名： | 胡若凡 |
| 学 院： | 计算机科学与技术学院 |
| 专 业： | 计算机科学与技术 |
| 邮 箱： | 2811668688@qq.com |
| QQ 号： | 2811668688 |
| 电 话： | 13913421107 |
| 指 导 教 师： | 洪奇军 |
| 报 告 日 期： | 2021年 12 月 25 日 |

**实验十一：寄存器和寄存器传输设计**

课程名称： 数字逻辑设计 实验类型： 综合

实验项目名称： 寄存器和寄存器传输设计

学生姓名： 胡若凡 学号： 3200102312 同组学生姓名： 叶之凡

实验地点： 紫金港东四 509 室 实验日期： 2021 年 12 月 11 日

1. **操作方法与实验步骤**

**1.1 新建工程文件**

新建工程，工程名称用MyALUTrans。添加四个模块，ALU模块，4位4选1模块，防抖动模块，显示模块。

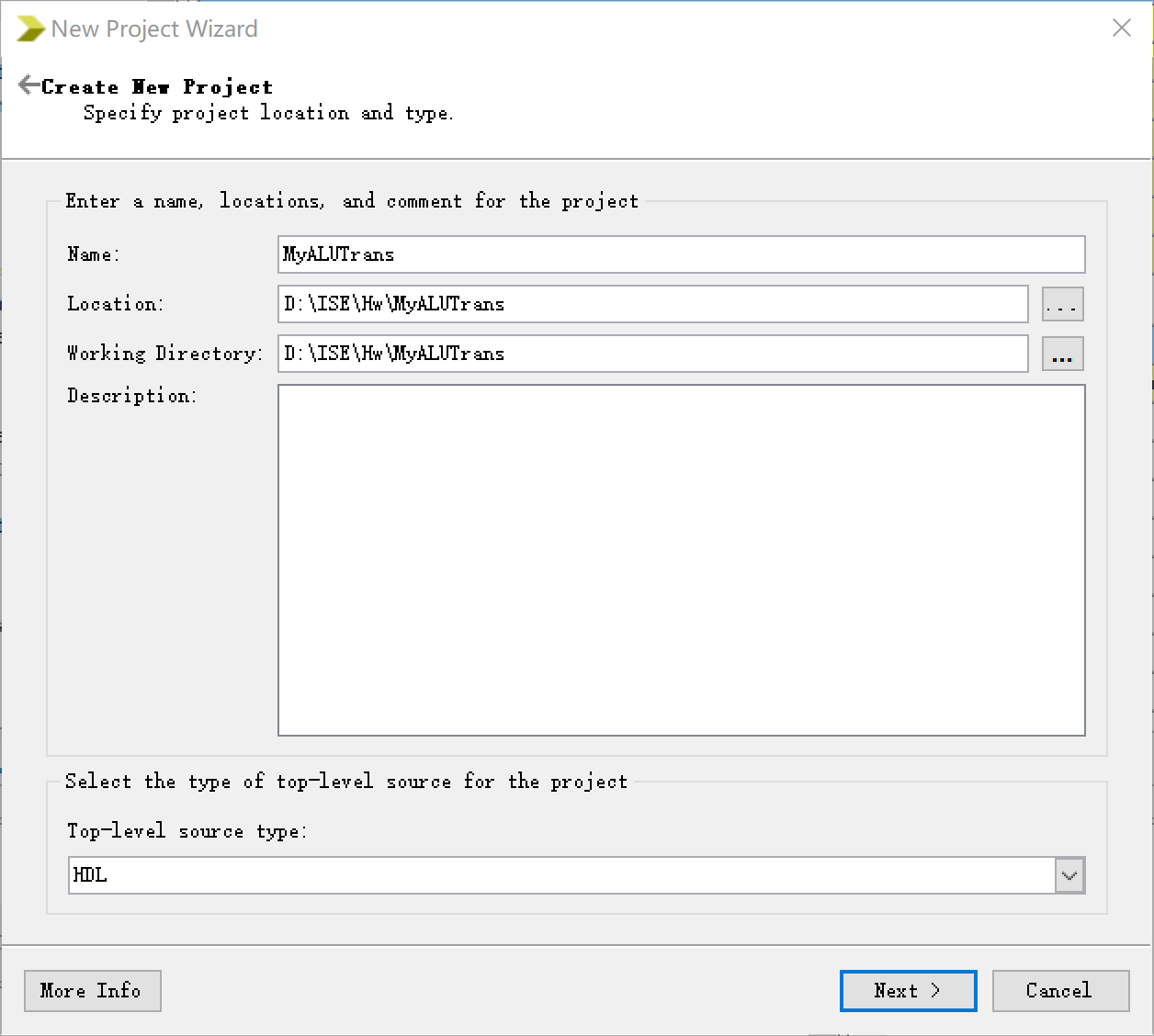


图 1.1 新建工程文件（一）

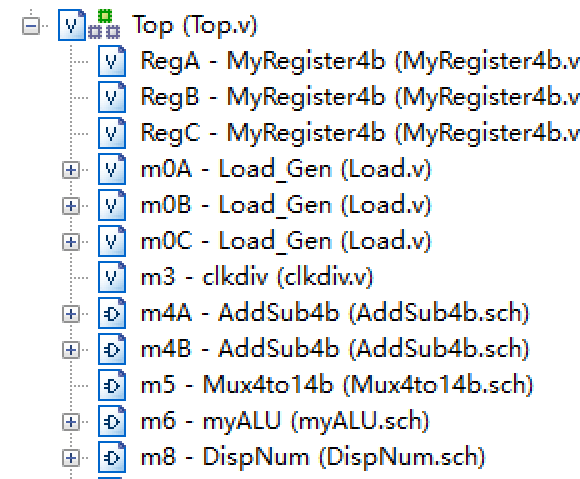
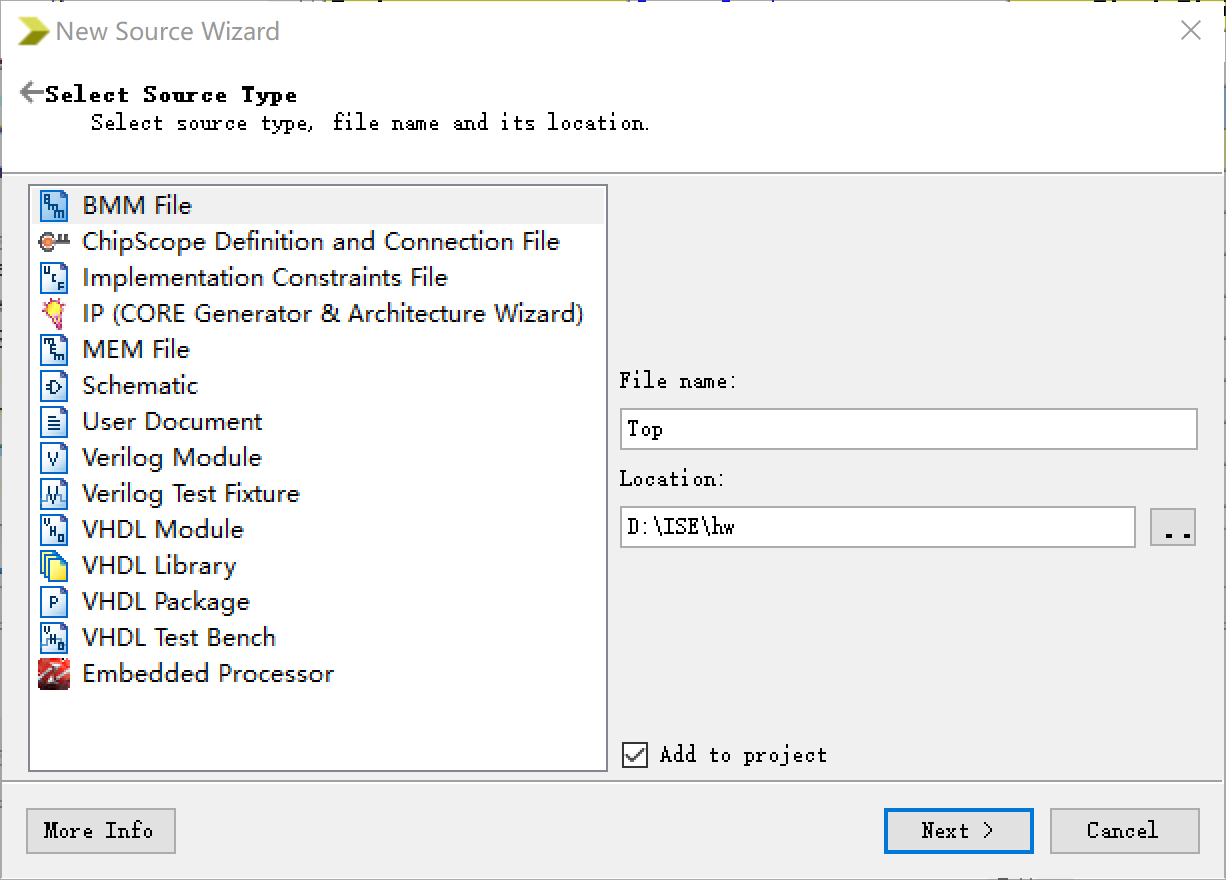
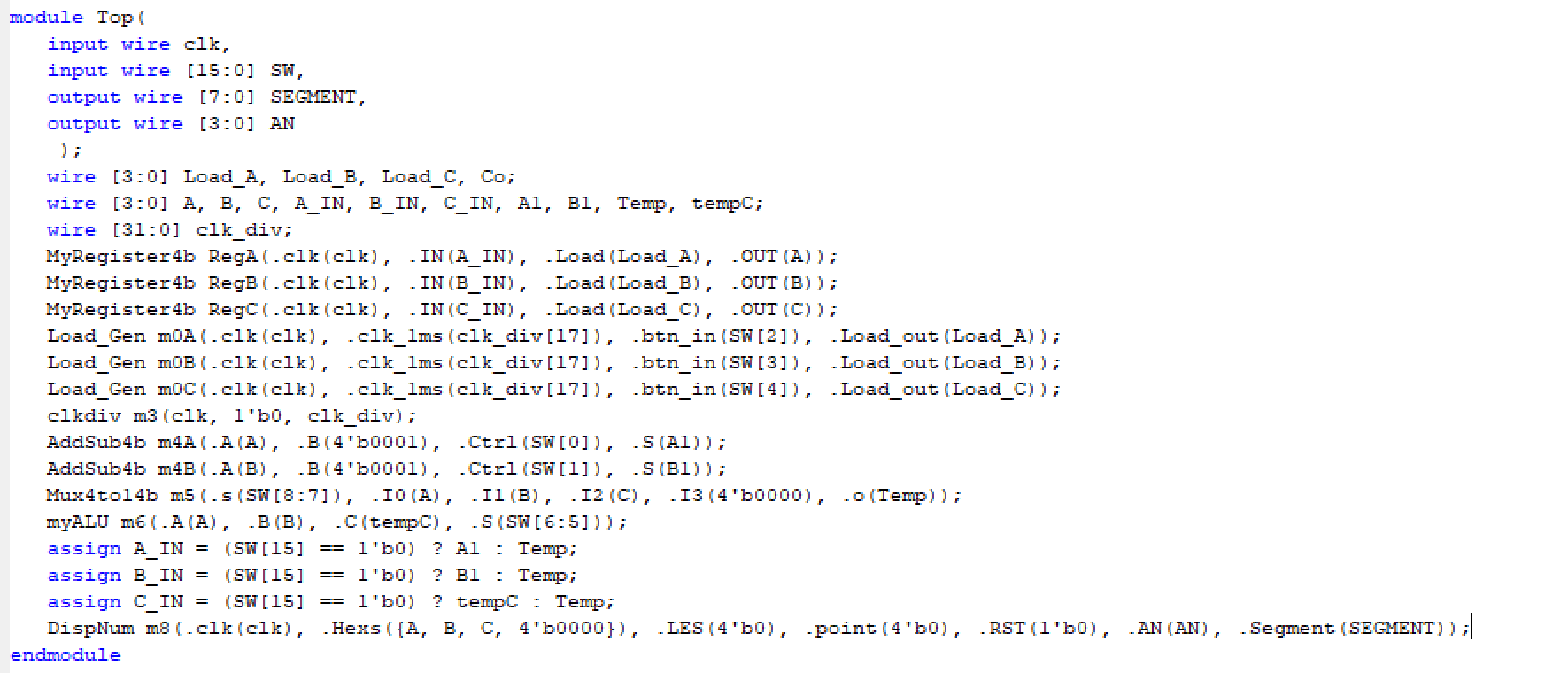


图 1.2 模块添加

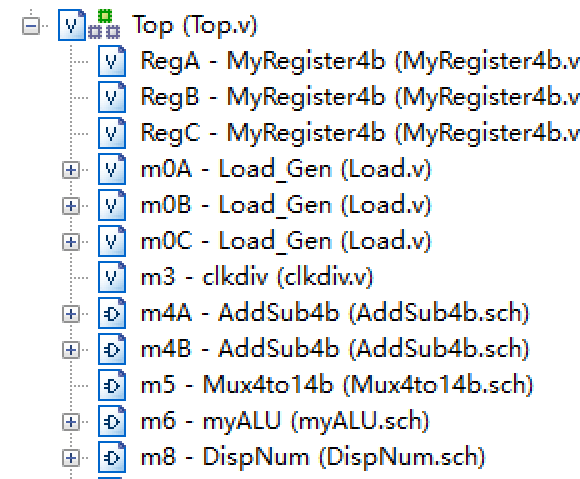
**1.2 Top设计**

新建源文件类型是 Verilog，文件名 Top。右键设为“Set as Top Module”.





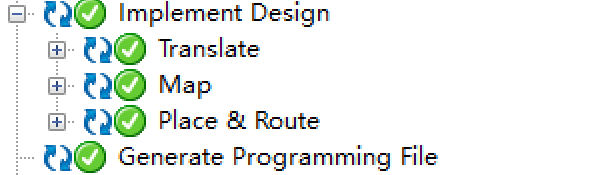
将top内部分设置好。



##### 1.3 建立用户时序约束并为模块的端口指定引脚分配

1.新建引脚分配文件。在新建源文件向导中勾选“Implementation Constraints File”命名为K7 点击 “Finish”进入“K7.ucf”编辑窗口。并在其中输入代码。

2.在“Processes”窗口进行 “Synthesize – XST”,“Implementation Design”， “Generate Programming File”。



3. “Processes”窗口中点开“Config Targe Device”，双击“Manage Configuration Project(iMPACT)”选项，出现如下 IMPACT 窗口。



4. 双击“Boundary Scan”弹出下载编辑窗口（边界扫描），鼠标右键选择 “Initialize Chain”，系统自动查找已连接在电脑上的开发平台JTAG下载链。出 现“XCK160t”容器，右击，选择“Assign New Configuration File”窗口，找到工程目录，选择“.bit”文件，在弹出的“Attach SPI or BPI PROM”窗口单击“No”，“Device Programming Properties”窗口单击“Yes”。右击容器，单击“Program”下载到SWORD板上。窗口下方出现“SUCCESS”后，即可以拨动开关，进行实验。

##### 1.4 代码块

1.Top

Top(

input wire clk,

input wire [15:0] SW,

output wire [7:0] SEGMENT,

output wire [3:0] AN

);

wire [3:0] Load\_A, Load\_B, Load\_C, Co;

wire [3:0] A, B, C, A\_IN, B\_IN, C\_IN, A1, B1, Temp, tempC;

wire [31:0] clk\_div;

MyRegister4b RegA(.clk(clk), .IN(A\_IN), .Load(Load\_A), .OUT(A));

MyRegister4b RegB(.clk(clk), .IN(B\_IN), .Load(Load\_B), .OUT(B));

MyRegister4b RegC(.clk(clk), .IN(C\_IN), .Load(Load\_C), .OUT(C));

Load\_Gen m0A(.clk(clk), .clk\_1ms(clk\_div[17]), .btn\_in(SW[2]), .Load\_out(Load\_A));

Load\_Gen m0B(.clk(clk), .clk\_1ms(clk\_div[17]), .btn\_in(SW[3]), .Load\_out(Load\_B));

Load\_Gen m0C(.clk(clk), .clk\_1ms(clk\_div[17]), .btn\_in(SW[4]), .Load\_out(Load\_C));

clkdiv m3(clk, 1'b0, clk\_div);

AddSub4b m4A(.A(A), .B(4'b0001), .Ctrl(SW[0]), .S(A1));

AddSub4b m4B(.A(B), .B(4'b0001), .Ctrl(SW[1]), .S(B1));

Mux4to14b m5(.s(SW[8:7]), .I0(A), .I1(B), .I2(C), .I3(4'b0000), .o(Temp));

myALU m6(.A(A), .B(B), .C(tempC), .S(SW[6:5]));

assign A\_IN = (SW[15] == 1'b0) ? A1 : Temp;

assign B\_IN = (SW[15] == 1'b0) ? B1 : Temp;

assign C\_IN = (SW[15] == 1'b0) ? tempC : Temp;

DispNum m8(.clk(clk), .Hexs({A, B, C, 4'b0000}), .LES(4'b0), .point(4'b0), .RST(1'b0), .AN(AN), .Segment(SEGMENT));

Endmodule

2. Load\_Gen

module Load\_Gen(

input wire clk,

input wire clk\_1ms,

input wire btn\_in,

output reg Load\_out

);

initial Load\_out = 0;

wire btn\_out;

reg old\_btn;

pbdebounce p0(clk\_1ms, btn\_in, btn\_out);

always@(posedge clk) begin

if ((old\_btn == 1'b0) && (btn\_out == 1'b1)) //btn出现上升沿

Load\_out <= 1'b1;

else

Load\_out <= 1'b0;

end

always@(posedge clk) begin //保存上一个周期btn的状态

old\_btn <= btn\_out;

end

endmodule

3. MyRegister4b

module MyRegister4b(

input wire clk,

input wire [3:0] IN,

input wire Load,

output reg [3:0] OUT

);

always @ (posedge clk) begin

if (Load) OUT <= IN;

end

endmodule

4. clkdiv

module clkdiv(input clk,

input rst,

output reg[31:0] clkdiv

);

// Clock divider-时钟分频器

always @ (posedge clk or posedge rst) begin

if (rst) clkdiv <= 0;

else clkdiv <= clkdiv + 1'b1;

end

endmodule

5.

NET "SEGMENT[0]" LOC = AB22 | IOSTANDARD = LVCMOS33;

NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;

NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;

NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;

NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;

NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;

NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;

NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;

NET "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33;

NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;

NET "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33;

NET "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33;

NET "SW[0]" LOC = AF10 | IOSTANDARD = LVCMOS15;

NET "SW[1]" LOC = AF13 | IOSTANDARD = LVCMOS15;

NET "SW[2]" LOC = AE13 | IOSTANDARD = LVCMOS15;

NET "SW[3]" LOC = AF8 | IOSTANDARD = LVCMOS15;

NET "SW[4]" LOC = AE8 | IOSTANDARD = LVCMOS15;

NET "SW[5]" LOC = AF12 | IOSTANDARD = LVCMOS15;

NET "SW[6]" LOC = AE12 | IOSTANDARD = LVCMOS15;

NET "SW[7]" LOC = AE10 | IOSTANDARD = LVCMOS15;

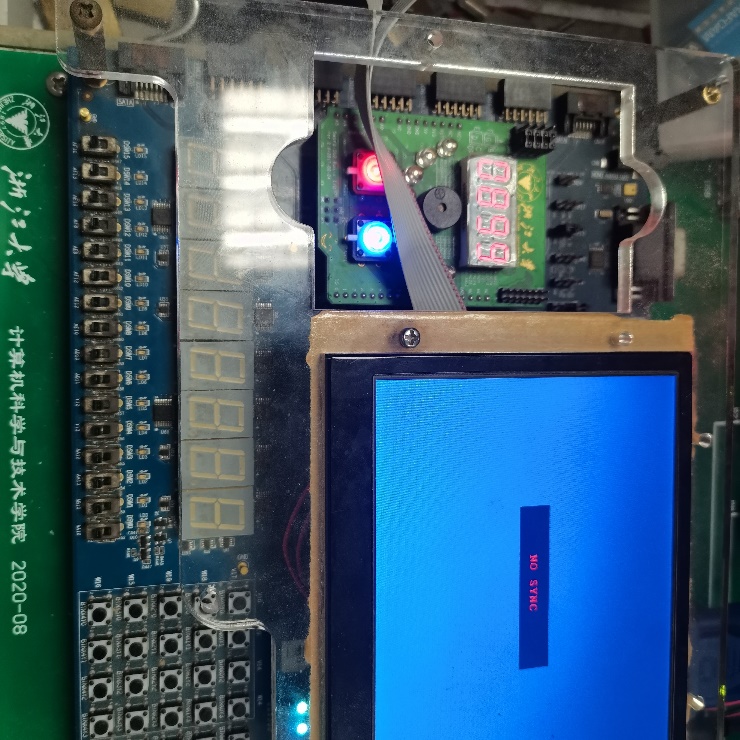
NET "SW[8]" LOC = AD10 | IOSTANDARD = LVCMOS15;

NET "SW[15]" LOC = AA10 | IOSTANDARD = LVCMOS15;

NET "clk" LOC = AC18 | IOSTANDARD = LVCMOS18;

1. **实验结果与分析**

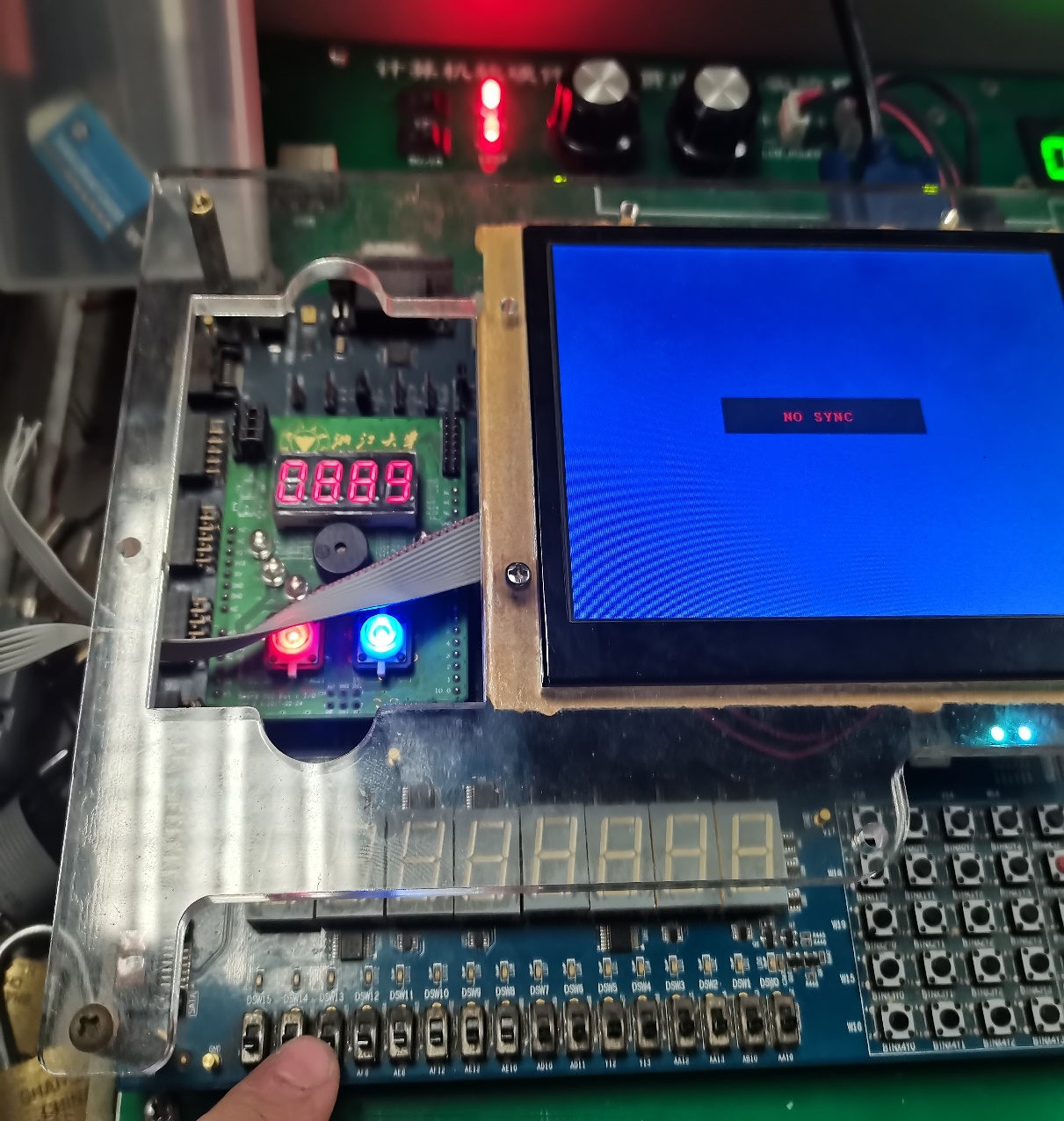
原图



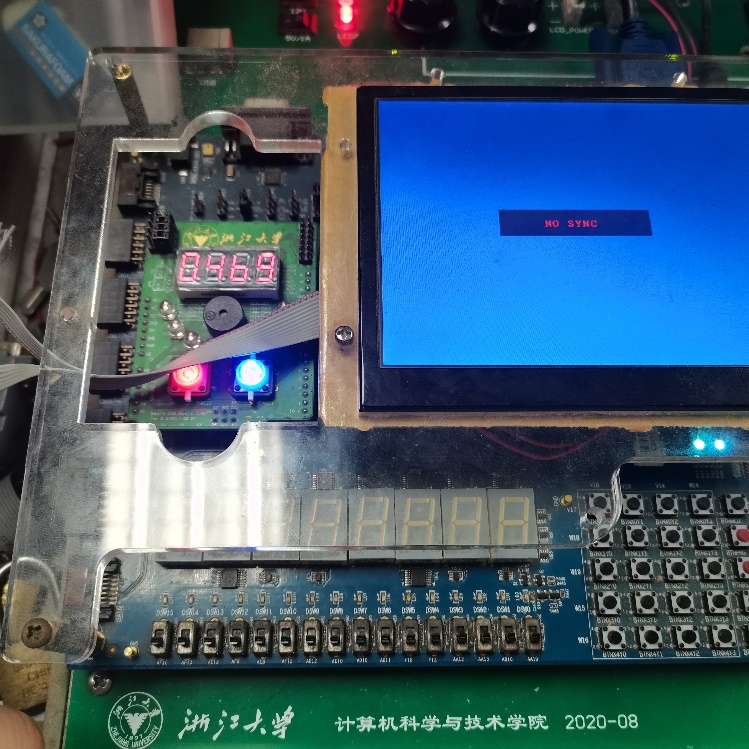
进行各个位上的加法



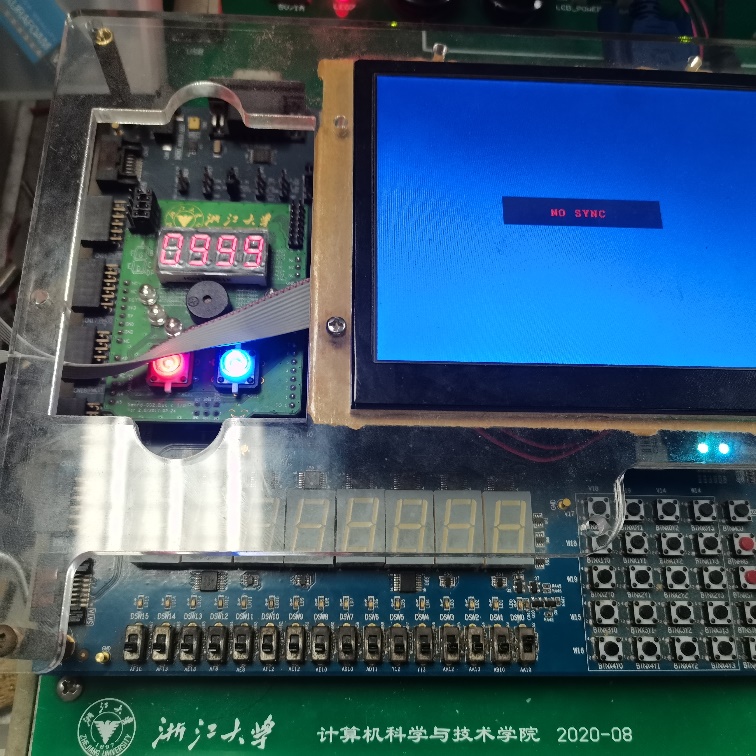
进行各个位上的减法



位操作



移位操作



1. **讨论与心得**

这是第一次出现寄存器的实验，其实也是第一次开始接受越来越简易的一份ppt的实验。在一开始的时候其实我有一些措手不及，只能按照ppt的指示一点点的进行操作，先把模块进行引入，然后对于top模块一点点进行编写，其中一定要十分清楚寄存器的运行机制，知道它的功能，才能进行更好的编辑。

而引脚则是和之前的实验中所相仿类似，只要整合前面的引脚就可以完成。我觉得其实比较困难的是上板的检验，仔细对照了很久才完成了验收方面的各种操作。

**实验十二：计数器、定时器设计与应用**

课程名称： 数字逻辑设计 实验类型： 综合

实验项目名称： 计数器、定时器设计与应用

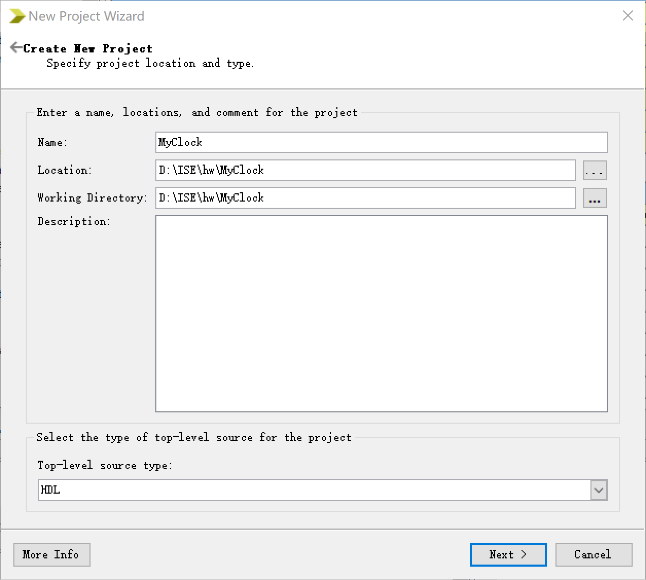
学生姓名： 胡若凡 学号： 3200102312 同组学生姓名： 叶之凡

实验地点： 紫金港东四 509 室 实验日期： 2021 年 12 月 18 日

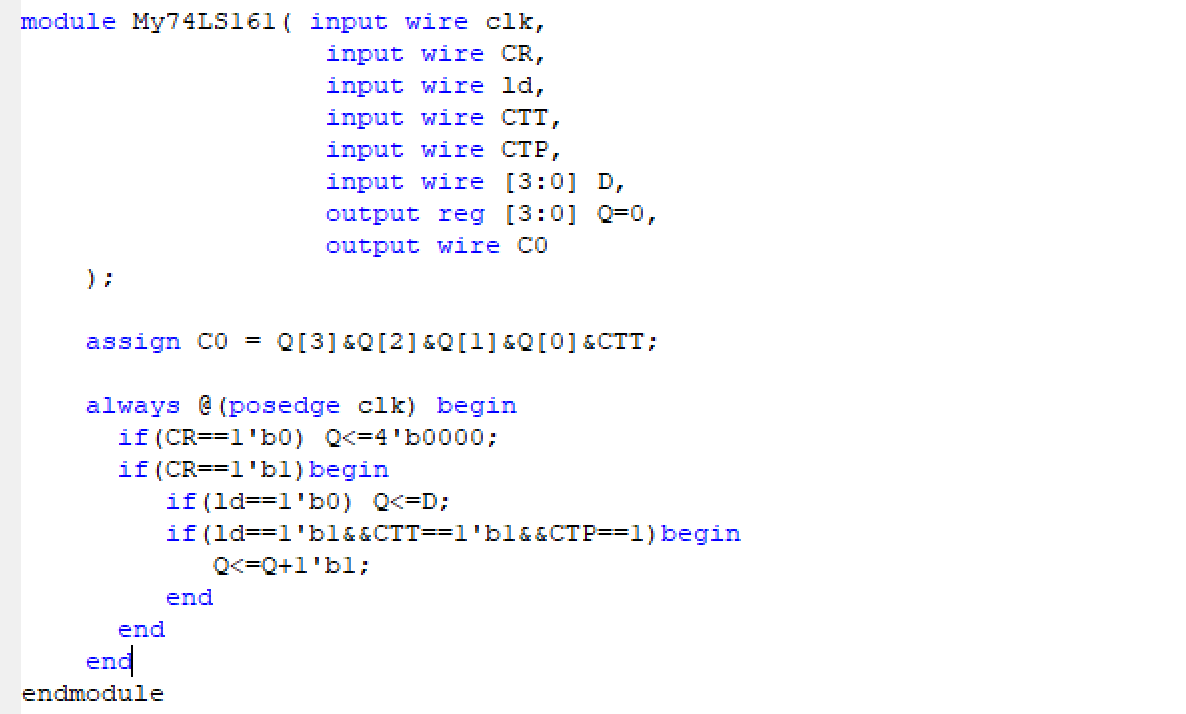
**一．操作方法与实验步骤**

**1.1 新建工程文件**

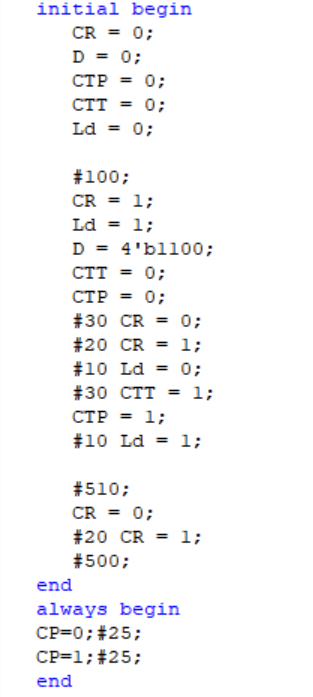
新建工程，工程名称用 MyClock。



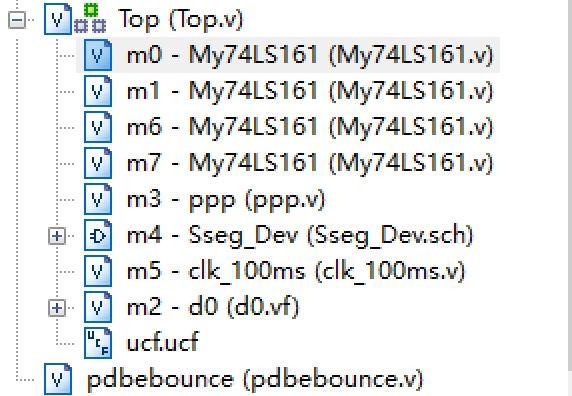
设计My74LS161，调用分频模块，用100ms作为分的驱动时钟，调用显示模块，设计一个数字钟，使用60进制和24进制计数器，实现24小时内时间的实时显示。



设计My74LS161的激励代码，并进行激励仿真对比。

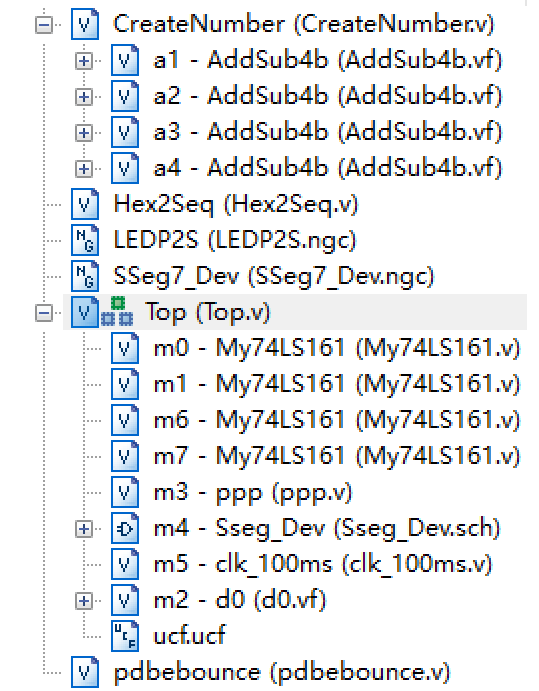


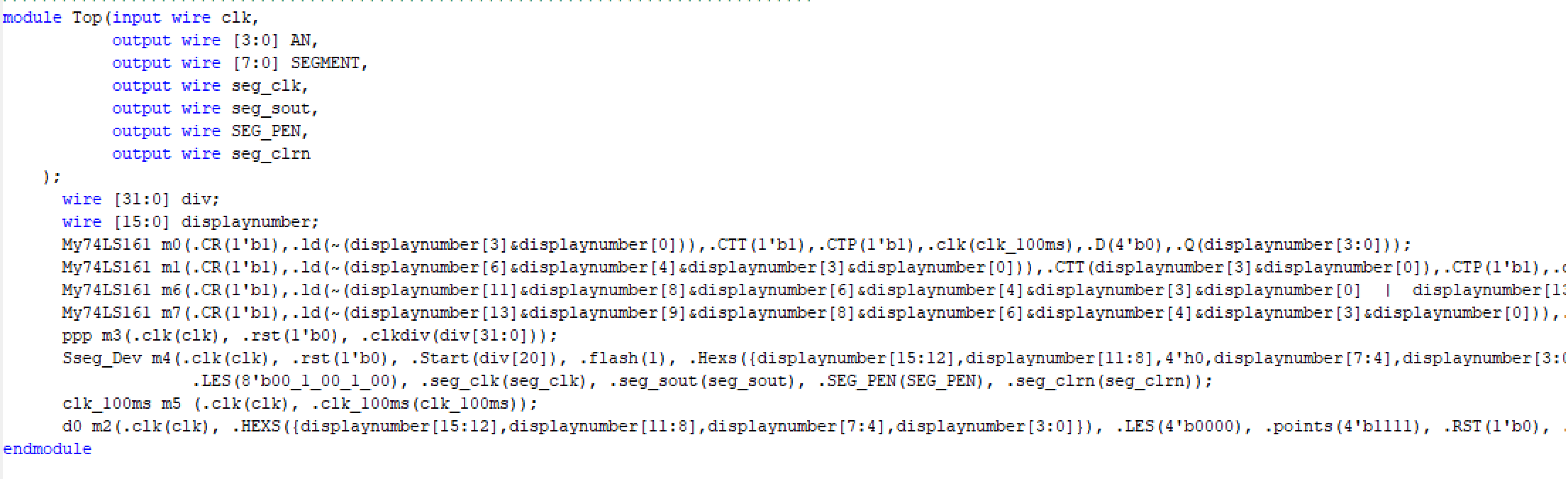
发现My74LS161模块在CR=1，Ld=1且CPP=CTT=1时进行计数，CR=0时置为0，其他情况保持不变，My74LS161模块正确。



**1.2 Top设计**

新建源文件类型是 Verilog，文件名 Top。右键设为“Set as Top Module”.

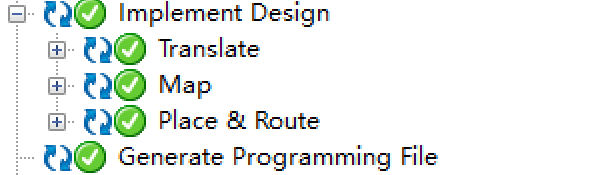




##### 1.3 建立用户时序约束并为模块的端口指定引脚分配

1.新建引脚分配文件。在新建源文件向导中勾选“Implementation Constraints File”命名为K7 点击 “Finish”进入“K7.ucf”编辑窗口。并在其中输入代码。

2.在“Processes”窗口进行 “Synthesize – XST”,“Implementation Design”， “Generate Programming File”。



3. “Processes”窗口中点开“Config Targe Device”，双击“Manage Configuration Project(iMPACT)”选项，出现如下 IMPACT 窗口。



4. 双击“Boundary Scan”弹出下载编辑窗口（边界扫描），鼠标右键选择 “Initialize Chain”，系统自动查找已连接在电脑上的开发平台JTAG下载链。出 现“XCK160t”容器，右击，选择“Assign New Configuration File”窗口，找到工程目录，选择“.bit”文件，在弹出的“Attach SPI or BPI PROM”窗口单击“No”，“Device Programming Properties”窗口单击“Yes”。右击容器，单击“Program”下载到SWORD板上。窗口下方出现“SUCCESS”后，即可以拨动开关，进行实验。

**1.4 代码块**

1. Hex2Seq

module Hex2Seq(

input [3:0] Hex,

input LE,

input point,

input flash,

output [7:0]Segment

);

wire en= LE&flash;

MyMc14495 MSEG

Endmodule

2. My74LS161

module My74LS161( input wire clk,

input wire CR,

input wire ld,

input wire CTT,

input wire CTP,

input wire [3:0] D,

output reg [3:0] Q=0,

output wire C0

);

assign C0 = Q[3]&Q[2]&Q[1]&Q[0]&CTT;

always @(posedge clk) begin

if(CR==1'b0) Q<=4'b0000;

if(CR==1'b1)begin

if(ld==1'b0) Q<=D;

if(ld==1'b1&&CTT==1'b1&&CTP==1)begin

Q<=Q+1'b1;

end

end

end

endmodule

3. clk\_100ms

module clk\_100ms(

input wire clk,

output reg clk\_100ms

);

reg[31:0] cnt;

always@(posedge clk)begin

if(cnt<50\_000\_00) begin

cnt<=cnt+1;

end

else begin

cnt<=0;

clk\_100ms<=~clk\_100ms;

end

end

endmodule

4. pdbebounce

module pdbebounce(

input wire clk\_1ms,

input wire button,

output reg pbreg

);

reg [7:0] pbshift;

always@(posedge button) begin

pbshift=pbshift<<1;

pbshift[0]=clk\_1ms;

if(pbshift==8'b0)

pbreg =0;

if(pbshift==8'hFF)

pbreg=1;

end

endmodule

5.

NET"clk"LOC = AC18 | IOSTANDARD=LVCMOS18;

NET "SEGMENT[0]" LOC = AB22 | IOSTANDARD = LVCMOS33;#a

NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;#b

NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;

NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;

NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;

NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;

NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;#g

NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;#point

NET "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33;

NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;

NET "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33;

NET "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33;

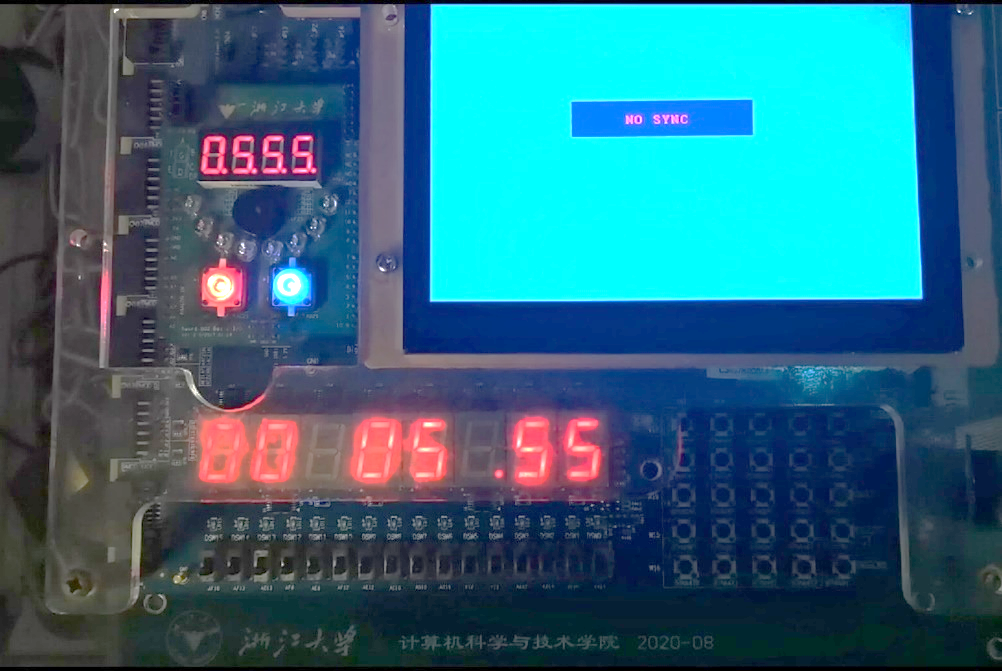
NET "seg\_clk" LOC = M24 | IOSTANDARD = LVCMOS33;

NET "seg\_clrn" LOC = M20 | IOSTANDARD = LVCMOS33;

NET "seg\_sout" LOC = L24 | IOSTANDARD = LVCMOS33;

NET "SEG\_PEN" LOC = R18 | IOSTANDARD = LVCMOS33;

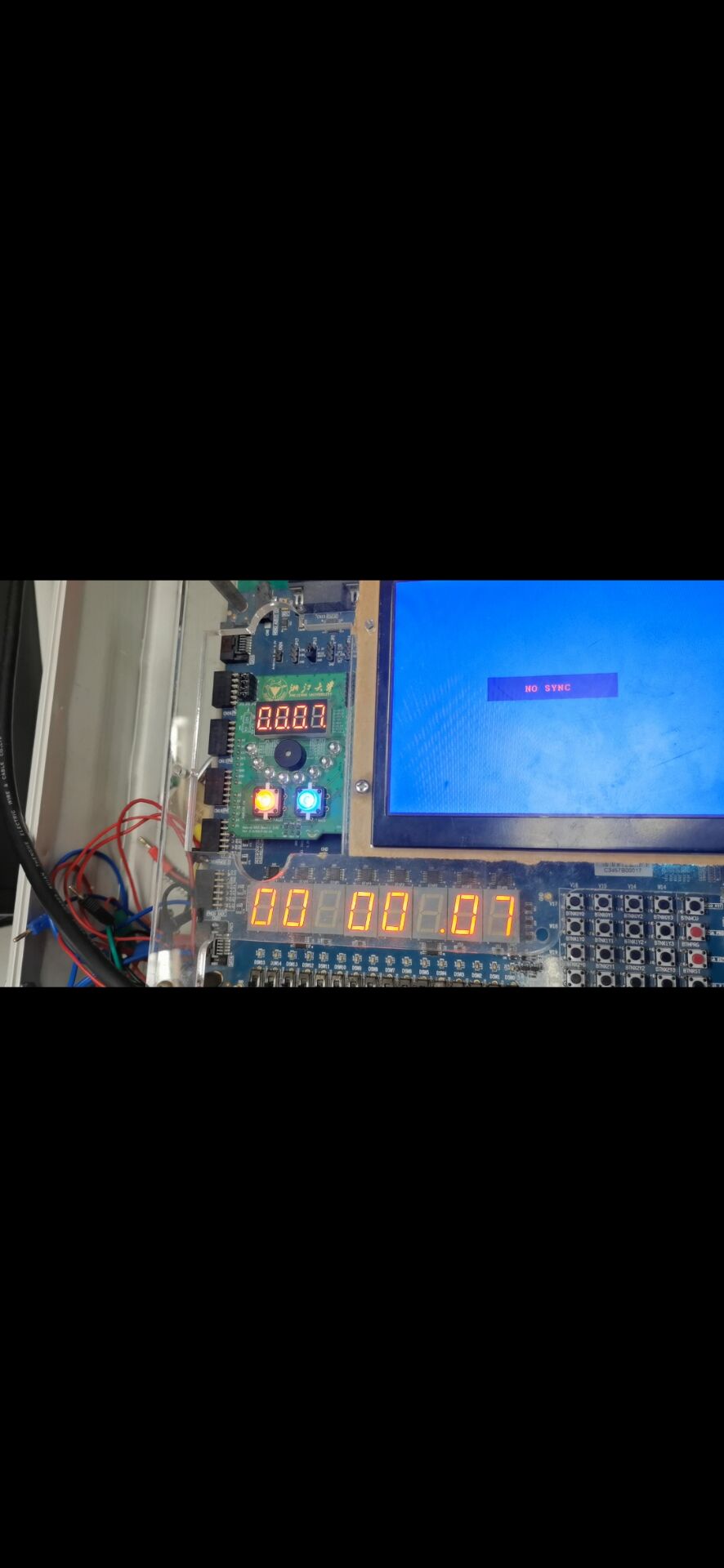
**二． 实验结果与分析**



随着时间变化，时钟也逐渐变化。



随着达到2359后，时钟清0。



**三． 实验心得体会**

本次实验对于74LS161的设计让我对于 verilog 语言的结构化描述掌握更加熟练，同时时钟的制作也让我对各个函数的关系有了更加深刻的了解。

时钟的操作是一个典型的时序逻辑电路的设计，其中涉及四个位数的变化，并且一个数字还需要用特定模块才能实现，因此这个并行的工程，十分的重要，也和课堂上这一块的内容联系紧密，在结束这个实验之后，我对于时钟数据的变化更加熟悉了。

**实验十三：移位寄存器设计与应用**

课程名称： 数字逻辑设计 实验类型： 综合

实验项目名称： 移位寄存器设计与应用

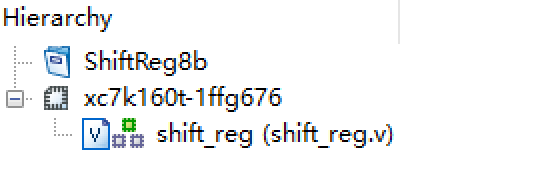
学生姓名： 胡若凡 学号： 3200102312 同组学生姓名： 叶之凡

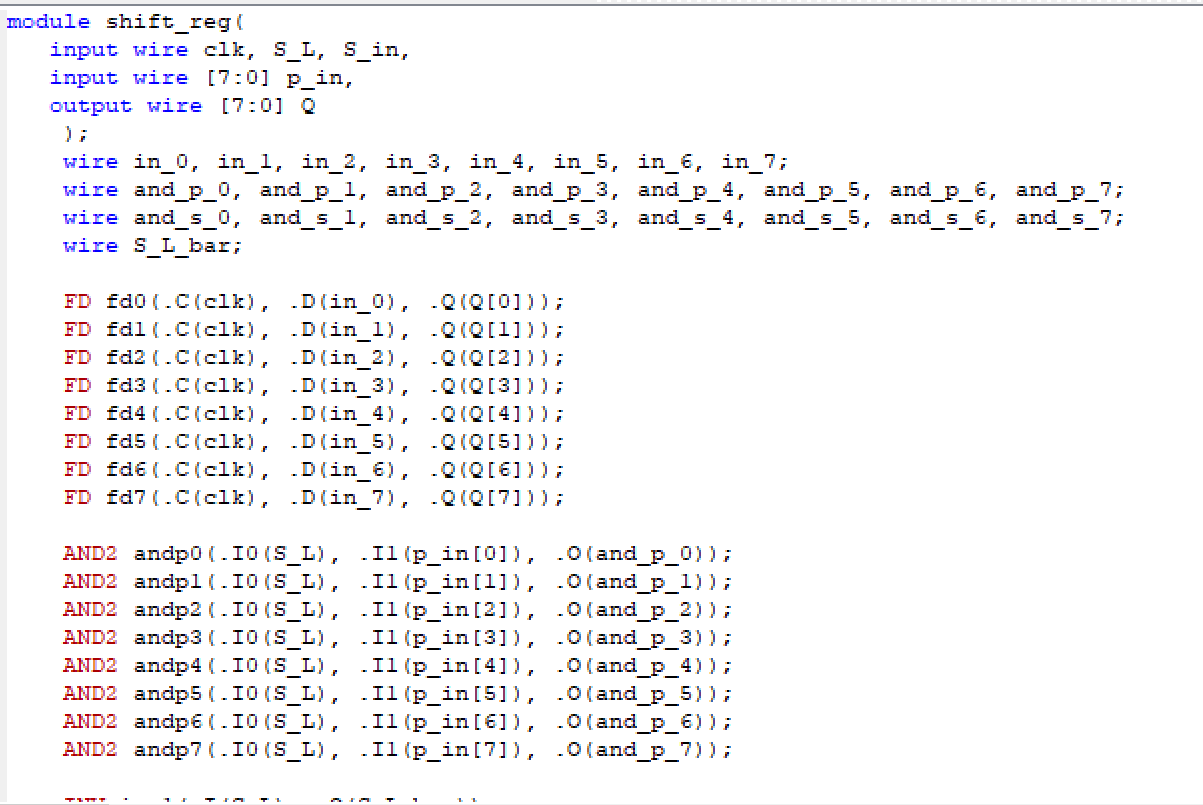
实验地点： 紫金港东四 509 室 实验日期： 2021 年 12 月 25 日

**一．操作方法与实验步骤**

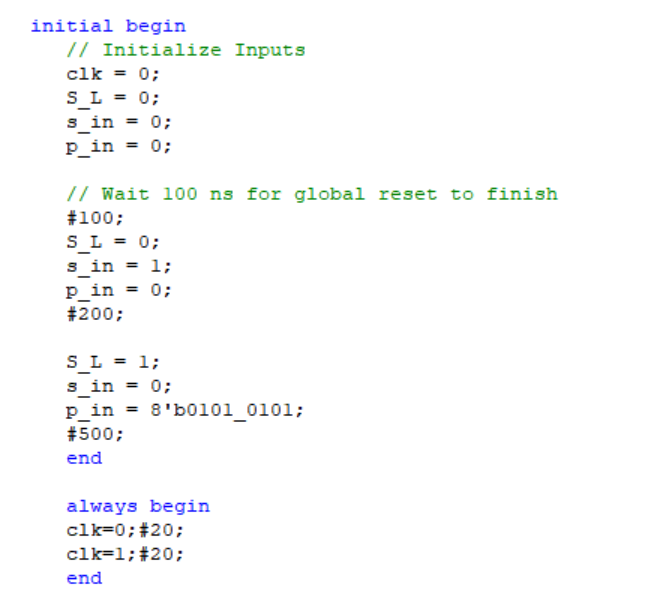
**1．设计8位带并行输入的右移移位寄存器**

新建工程，工程名称用ShfitReg8b，Top Level Source Type用HDL，用结构化描述设计。

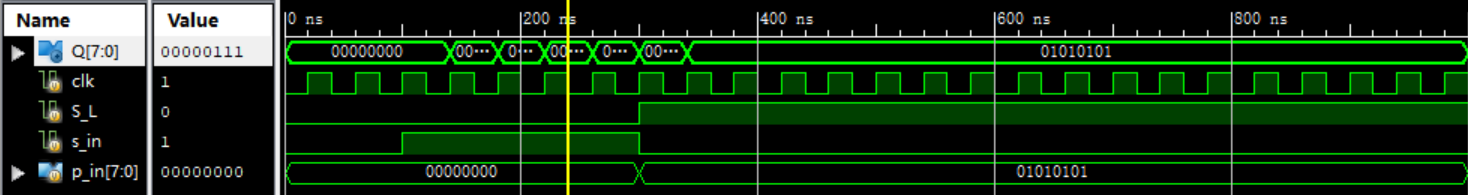




输入激励代码，进行仿真对比。



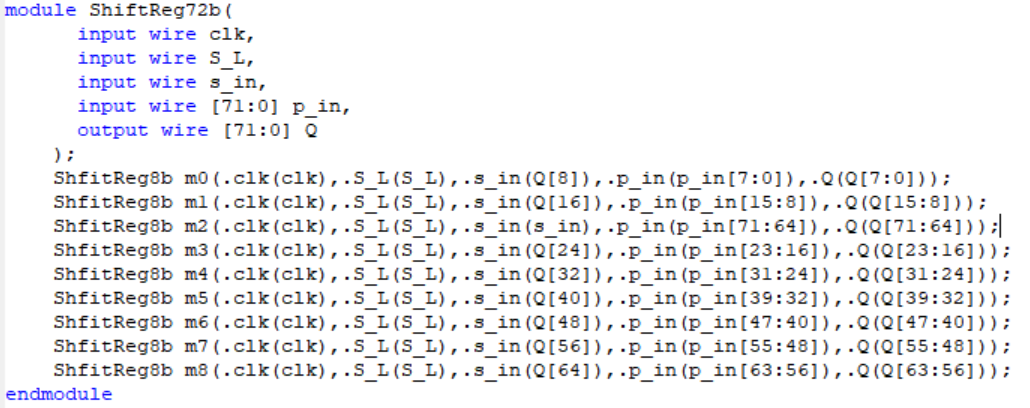
发现串行输入1之后在100ns后当s\_in=1时每个时钟的上升沿寄存器都会右移一位，当s\_in=0时，进行了并行输入，01010101被赋给了寄存器，ShiftReg8b功能完好。



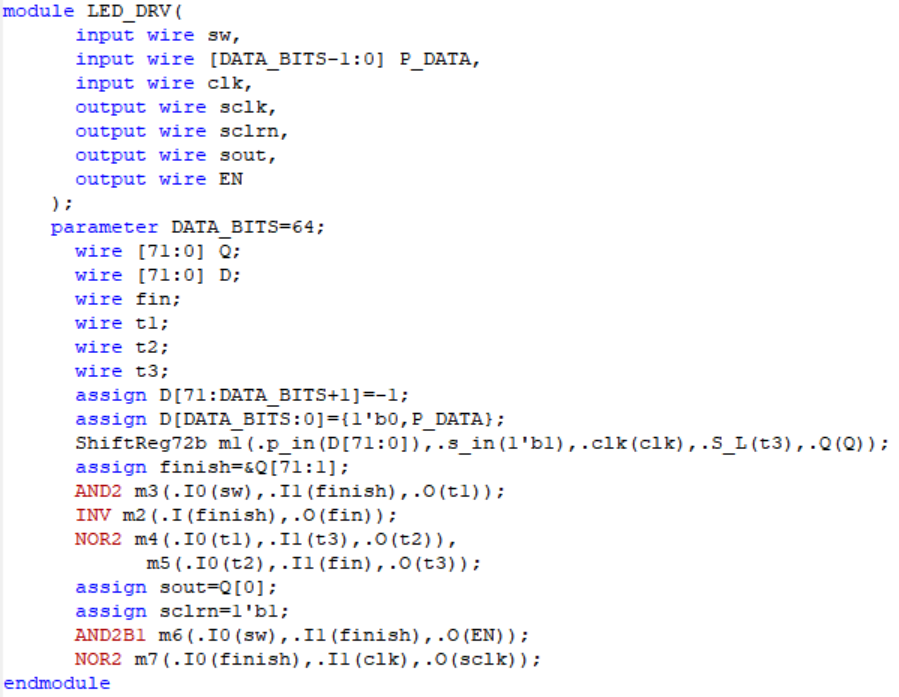
**2．设计主板LED灯驱动模块。**

新建工程，工程名称用LEDP2S，Top Level Source Type用HDL，用结构化描述设计，调用CreatNumber模块，用4位七段数码管设置16位LED灯的初值，调用显示模块。

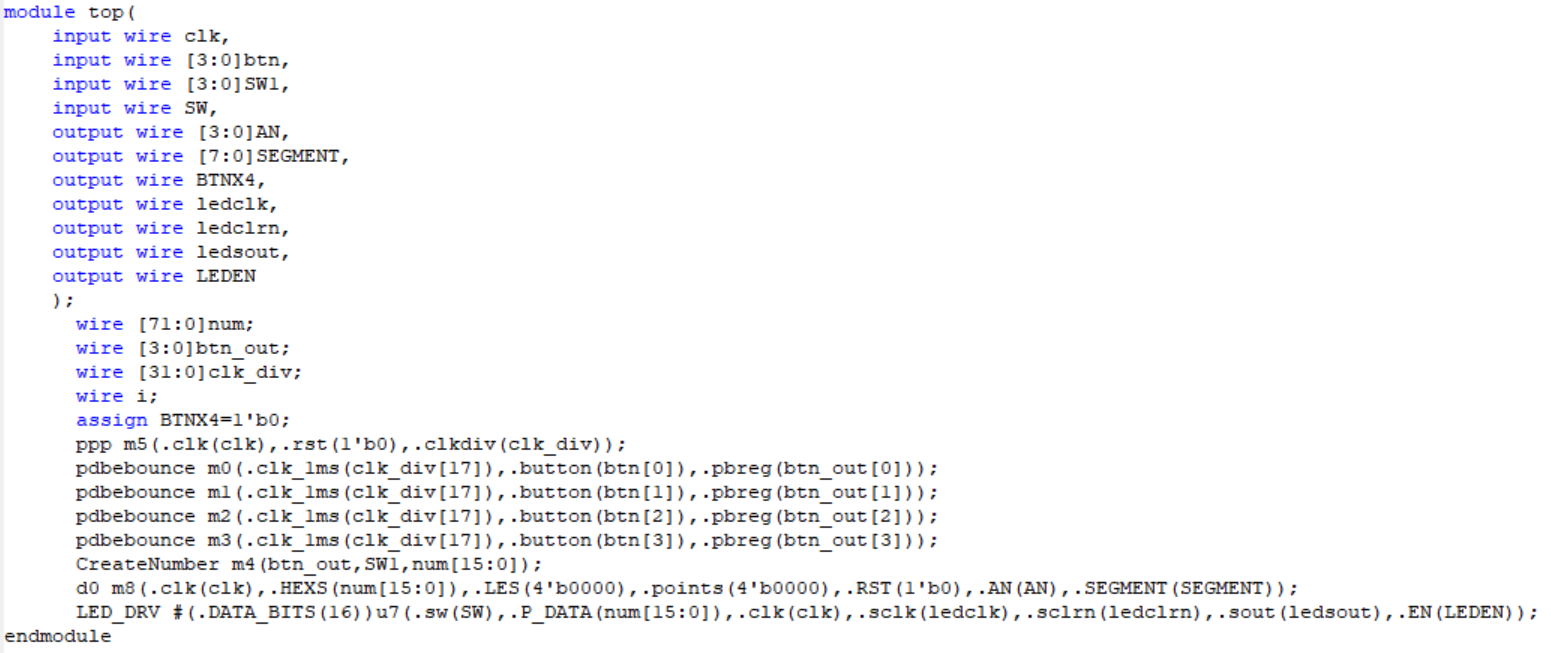
改造ShfitReg8b模块，设计72位带并行输入的右移移位寄存器。

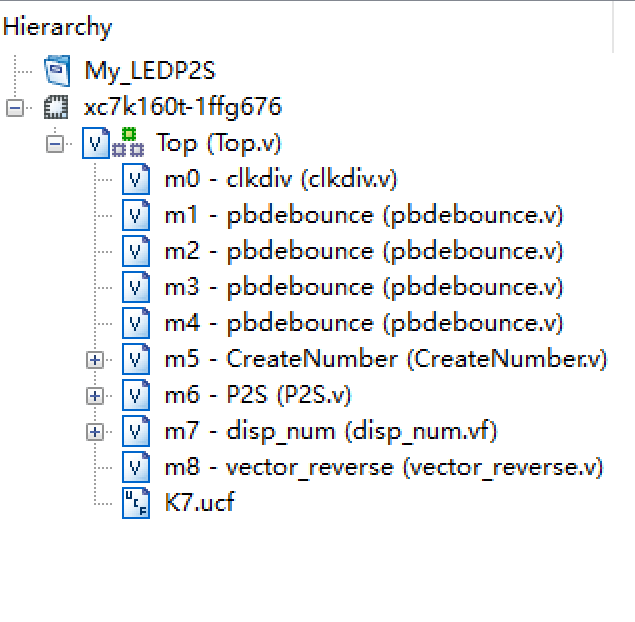


利用ShiftReg72设计LED灯驱动模块LED\_DRV。



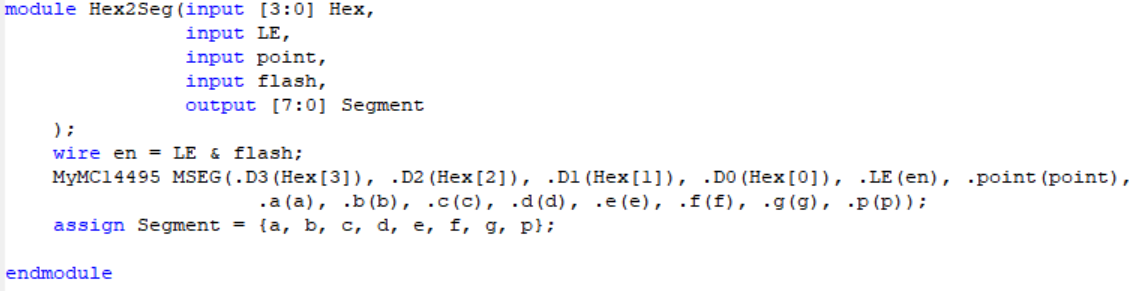
设计top模块实现功能。



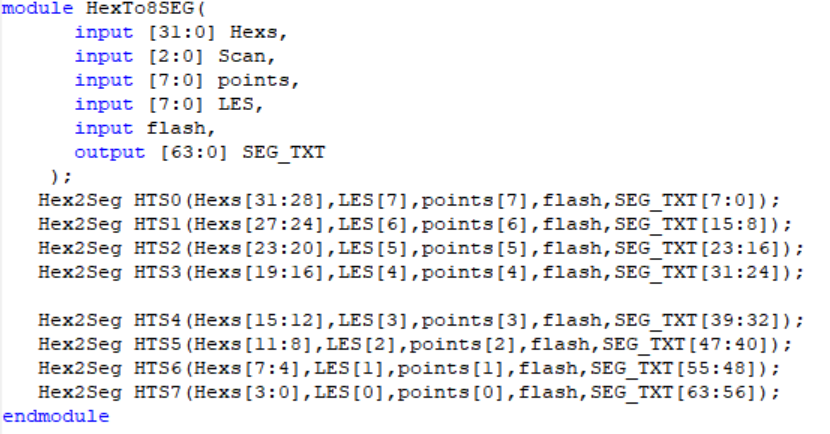


**3．设计主板七段数码管驱动模块**

设计Hex2Seg模块：



设计Hex8Seg模块：



设计top模块



**4.代码块**

1.My\_Segp2s

(1)module Top( clk,

sclk, sclrn, sout, SEG\_PEN

);

input wire clk;

output wire sout;

output wire sclrn;

output wire sclk;

output wire SEG\_PEN;

wire[31:0] num;

wire[31:0] num\_rev;

assign num = 32'h00100556;

wire[31:0] clkd;

clkdiv c0(.clk(clk), .clkdiv(clkd), .rst(1'b0));

SEG\_DRV m0(.flash(1'b1), .Hexs(num), .point(8'b0000\_0000), .LES(8'b0000\_0000), .clk(clk), .rst(1'b0), .Start(clkd[20]), .sout(sout),

.sclrn(sclrn), .sclk(sclk), .SEG\_PEN(SEG\_PEN));

vector\_reverse #(.NUM\_BITS(32)) m8(.in(num), .out(num\_rev));

endmodule

(2)module SEG\_DRV( flash, Hexs, point, LES, clk, rst, Start,

sclk, sclrn, sout, SEG\_PEN

);

input wire flash;

input wire[31:0] Hexs;

input wire[7:0] point;

input wire[7:0] LES;

input wire clk;

input wire rst;

input wire Start;

output wire sout;

output wire sclrn;

output wire sclk;

output wire SEG\_PEN;

wire[63:0] SEG\_TXT;

HexTo8SEG m0(.flash(flash), .Hexs(Hexs[31:0]), .points(point[7:0]), .LES(LES[7:0]), .SEG\_TXT(SEG\_TXT));

P2S #(.DATA\_BITS(64)) m1(.Start(Start), .PData(SEG\_TXT), .clk(clk), .sclk(sclk), .sclrn(sclrn), .sout(sout), .EN(SEG\_PEN));

endmodule

(3)module vector\_reverse( in,

out

);

parameter NUM\_BITS = 64;

input wire [NUM\_BITS-1:0] in;

output wire [NUM\_BITS-1:0] out;

genvar i;

generate for(i = 0; i < NUM\_BITS; i = i+1) begin

assign out[i] = in[NUM\_BITS-1-i];

end endgenerate

endmodule

(4)NET"clk"LOC = AC18 | IOSTANDARD = LVCMOS18;

NET"sclk" LOC = M24 | IOSTANDARD = LVCMOS33;

NET"sclrn" LOC = M20 | IOSTANDARD = LVCMOS33;

NET"sout" LOC = L24 | IOSTANDARD = LVCMOS33;

NET"SEG\_PEN" LOC = R18 | IOSTANDARD = LVCMOS33;

(5) module shift\_reg72b\_sim;

// Inputs

reg clk;

reg S\_L;

reg s\_in;

reg [71:0] p\_in;

// Outputs

wire [71:0] Q;

// Instantiate the Unit Under Test (UUT)

shift\_reg72b uut (

.clk(clk),

.S\_L(S\_L),

.s\_in(s\_in),

.p\_in(p\_in),

.Q(Q)

);

initial begin

// Initialize Inputs

clk = 0;

S\_L = 0;

s\_in = 0;

p\_in = 0;

// Wait 100 ns for global reset to finish

#100;

// Add stimulus here

S\_L = 0;

s\_in = 1;

p\_in = 0;

#3000;

S\_L = 1;

s\_in = 0;

p\_in = 72'h5555\_5555\_5555\_5555\_55;

#500;

end

always begin

#20; clk = ~clk;

end

endmodule

2. ShiftReg8b

`timescale 1ns / 1ps

module shift\_reg(

input wire clk, S\_L, S\_in,

input wire [7:0] p\_in,

output wire [7:0] Q

);

wire in\_0, in\_1, in\_2, in\_3, in\_4, in\_5, in\_6, in\_7;

wire and\_p\_0, and\_p\_1, and\_p\_2, and\_p\_3, and\_p\_4, and\_p\_5, and\_p\_6, and\_p\_7;

wire and\_s\_0, and\_s\_1, and\_s\_2, and\_s\_3, and\_s\_4, and\_s\_5, and\_s\_6, and\_s\_7;

wire S\_L\_bar;

FD fd0(.C(clk), .D(in\_0), .Q(Q[0]));

FD fd1(.C(clk), .D(in\_1), .Q(Q[1]));

FD fd2(.C(clk), .D(in\_2), .Q(Q[2]));

FD fd3(.C(clk), .D(in\_3), .Q(Q[3]));

FD fd4(.C(clk), .D(in\_4), .Q(Q[4]));

FD fd5(.C(clk), .D(in\_5), .Q(Q[5]));

FD fd6(.C(clk), .D(in\_6), .Q(Q[6]));

FD fd7(.C(clk), .D(in\_7), .Q(Q[7]));

AND2 andp0(.I0(S\_L), .I1(p\_in[0]), .O(and\_p\_0));

AND2 andp1(.I0(S\_L), .I1(p\_in[1]), .O(and\_p\_1));

AND2 andp2(.I0(S\_L), .I1(p\_in[2]), .O(and\_p\_2));

AND2 andp3(.I0(S\_L), .I1(p\_in[3]), .O(and\_p\_3));

AND2 andp4(.I0(S\_L), .I1(p\_in[4]), .O(and\_p\_4));

AND2 andp5(.I0(S\_L), .I1(p\_in[5]), .O(and\_p\_5));

AND2 andp6(.I0(S\_L), .I1(p\_in[6]), .O(and\_p\_6));

AND2 andp7(.I0(S\_L), .I1(p\_in[7]), .O(and\_p\_7));

INV inv1(.I(S\_L), .O(S\_L\_bar));

AND2 ands0(.I0(S\_L\_bar), .I1(Q[1]), .O(and\_s\_0));

AND2 ands1(.I0(S\_L\_bar), .I1(Q[2]), .O(and\_s\_1));

AND2 ands2(.I0(S\_L\_bar), .I1(Q[3]), .O(and\_s\_2));

AND2 ands3(.I0(S\_L\_bar), .I1(Q[4]), .O(and\_s\_3));

AND2 ands4(.I0(S\_L\_bar), .I1(Q[5]), .O(and\_s\_4));

AND2 ands5(.I0(S\_L\_bar), .I1(Q[6]), .O(and\_s\_5));

AND2 ands6(.I0(S\_L\_bar), .I1(Q[7]), .O(and\_s\_6));

AND2 ands7(.I0(S\_L\_bar), .I1(S\_in), .O(and\_s\_7));

OR2 or0(.I0(and\_p\_0), .I1(and\_s\_0), .O(in\_0));

OR2 or1(.I0(and\_p\_1), .I1(and\_s\_1), .O(in\_1));

OR2 or2(.I0(and\_p\_2), .I1(and\_s\_2), .O(in\_2));

OR2 or3(.I0(and\_p\_3), .I1(and\_s\_3), .O(in\_3));

OR2 or4(.I0(and\_p\_4), .I1(and\_s\_4), .O(in\_4));

OR2 or5(.I0(and\_p\_5), .I1(and\_s\_5), .O(in\_5));

OR2 or6(.I0(and\_p\_6), .I1(and\_s\_6), .O(in\_6));

OR2 or7(.I0(and\_p\_7), .I1(and\_s\_7), .O(in\_7));

Endmodule

3. My\_LEDP2S

(1)module Top( btn, SW, clk,

sclk, sclrn, sout, LEDEN, AN, SEGMENT, BTNX4

);

input wire[3:0] btn;

input wire[4:0] SW; // [4:1] 控制四个数字的增减

// [0] 是否将七段数码管的数字传输到LED上

input wire clk;

output wire sclk;

output wire sclrn;

output wire sout;

output wire LEDEN;

output wire [3:0] AN;

output wire [7:0] SEGMENT;

output wire BTNX4;

wire[3:0] btn\_out;

wire[31:0] clk\_div;

wire[15:0] num;

wire[15:0] num\_rev;

assign BTNX4=0;

clkdiv m0(clk, 0, clk\_div);

// 防抖动模块

pbdebounce m1(clk\_div[17], btn[0], btn\_out[0]);

pbdebounce m2(clk\_div[17], btn[1], btn\_out[1]);

pbdebounce m3(clk\_div[17], btn[2], btn\_out[2]);

pbdebounce m4(clk\_div[17], btn[3], btn\_out[3]);

CreateNumber m5(btn\_out, SW[4:1], num);

P2S #(.DATA\_BITS(16))m6(.clk(clk), .Start(SW[0]), .PData(~num\_rev[15:0]), .sclk(sclk), .sclrn(sclrn), .sout(sout), .EN(LEDEN));

disp\_num m7(.clk(clk), .HEXS(num), .LES(4'b0000), .points(4'b1111), .RST(1'b0), .AN(AN), .Segment(SEGMENT));

vector\_reverse #(.NUM\_BITS(16)) m8(.in(num), .out(num\_rev));

endmodule

(2)

module clkdiv(input clk,

input rst,

output reg[31:0]clkdiv

);

always @ (posedge clk or posedge rst) begin

if(rst) clkdiv <= 0;

else clkdiv <= clkdiv + 1'b1;

end

endmodule

(3)

module pbdebounce(input wire clk\_1ms,

input wire button,

output reg pbreg

);

reg [7:0] pbshift;

always@(posedge clk\_1ms)begin

pbshift = pbshift << 1;

pbshift[0] = button;

if(pbshift == 8'b0) pbreg = 0;

if(pbshift == 8'hFF) pbreg = 1;

end

endmodule

(4)

module CreateNumber(input wire [3:0] btn,

input wire [3:0] sw,

output reg [15:0] num

);

wire [3:0] A1, B1, C1, D1;

initial num <= 16'habcd;

AddSub4b\_sch a1(.A(num[3:0]), .B(4'b0001), .Ctrl(sw[0]), .S(A1));

AddSub4b\_sch a2(.A(num[7:4]), .B(4'b0001), .Ctrl(sw[1]), .S(B1));

AddSub4b\_sch a3(.A(num[11:8]), .B(4'b0001), .Ctrl(sw[2]), .S(C1));

AddSub4b\_sch a4(.A(num[15:12]), .B(4'b0001), .Ctrl(sw[3]), .S(D1));

always @(posedge btn[0]) num[3:0] <= A1;

always @(posedge btn[1]) num[7:4] <= B1;

always @(posedge btn[2]) num[11:8] <= C1;

always @(posedge btn[3]) num[15:12] <= D1;

endmodule

(5)

module P2S( Start, PData, clk,

sclk, sclrn, sout, EN

);

parameter DATA\_BITS = 64;

input wire Start;

input wire clk;

input wire[DATA\_BITS-1:0] PData;

output wire sclk;

output wire sclrn;

output wire sout;

output wire EN;

wire[71:0] Q;

wire[71:0] D;

wire finish;

wire q;

wire nq;

wire set;

wire reset;

assign D[71:DATA\_BITS+1] = -1;

assign D[DATA\_BITS:0] = {1'b0, PData};

AND2 And\_Set(.I0(Start), .I1(finish), .O(set));

INV INV\_Reset(.I(finish), .O(reset));

NOR2 NOR\_nq(.I0(set), .I1(q), .O(nq));

NOR2 NOR\_q(.I0(reset), .I1(nq), .O(q));

shift\_reg72b m0 (.clk(clk), .S\_L(q), .s\_in(1'b1), .p\_in(D[71:0]), .Q(Q[71:0]));

assign finish = &Q[DATA\_BITS: 1];

assign sout = Q[0];

assign sclrn = 1'b1;

AND2B1 m2(.I0(Start), .I1(finish), .O(EN));

NOR2 m3(.I0(finish), .I1(clk), .O(sclk));

Endmodule

(6)

module vector\_reverse( in,

out

);

parameter NUM\_BITS = 64;

input wire [NUM\_BITS-1:0] in;

output wire [NUM\_BITS-1:0] out;

genvar i;

generate for(i = 0; i < NUM\_BITS; i = i+1) begin

assign out[i] = in[NUM\_BITS-1-i];

end endgenerate

endmodule

(7)

NET"clk"LOC = AC18 | IOSTANDARD = LVCMOS18;

NET"btn[0]"LOC = W14 | IOSTANDARD = LVCMOS18;

NET"btn[0]"clock\_dedicated\_route = false;

NET"btn[1]"LOC = V14 | IOSTANDARD = LVCMOS18;

NET"btn[1]"clock\_dedicated\_route = false;

NET"btn[2]"LOC = V19 | IOSTANDARD = LVCMOS18;

NET"btn[2]"clock\_dedicated\_route = false;

NET"btn[3]"LOC = V18 | IOSTANDARD = LVCMOS18;

NET"btn[3]"clock\_dedicated\_route = false;

NET"BTNX4"LOC = W16 | IOSTANDARD = LVCMOS18;

NET"SW[0]"LOC = AA10 | IOSTANDARD = LVCMOS15;

NET"SW[1]"LOC = AB10 | IOSTANDARD = LVCMOS15;

NET"SW[2]"LOC = AA13 | IOSTANDARD = LVCMOS15;

NET"SW[3]"LOC = AA12 | IOSTANDARD = LVCMOS15;

NET"SW[4]"LOC = Y13 | IOSTANDARD = LVCMOS15;

NET"SEGMENT[0]"LOC = AB22 | IOSTANDARD = LVCMOS33;

NET"SEGMENT[1]"LOC = AD24 | IOSTANDARD = LVCMOS33;

NET"SEGMENT[2]"LOC = AD23 | IOSTANDARD = LVCMOS33;

NET"SEGMENT[3]"LOC = Y21 | IOSTANDARD = LVCMOS33;

NET"SEGMENT[4]"LOC = W20 | IOSTANDARD = LVCMOS33;

NET"SEGMENT[5]"LOC = AC24 | IOSTANDARD = LVCMOS33;

NET"SEGMENT[6]"LOC = AC23 | IOSTANDARD = LVCMOS33;

NET"SEGMENT[7]"LOC = AA22 | IOSTANDARD = LVCMOS33;

NET"AN[0]"LOC = AD21 | IOSTANDARD = LVCMOS33;

NET"AN[1]"LOC = AC21 | IOSTANDARD = LVCMOS33;

NET"AN[2]"LOC = AB21 | IOSTANDARD = LVCMOS33;

NET"AN[3]"LOC = AC22 | IOSTANDARD = LVCMOS33;

NET "sclk"LOC = N26 | IOSTANDARD = LVCMOS33;

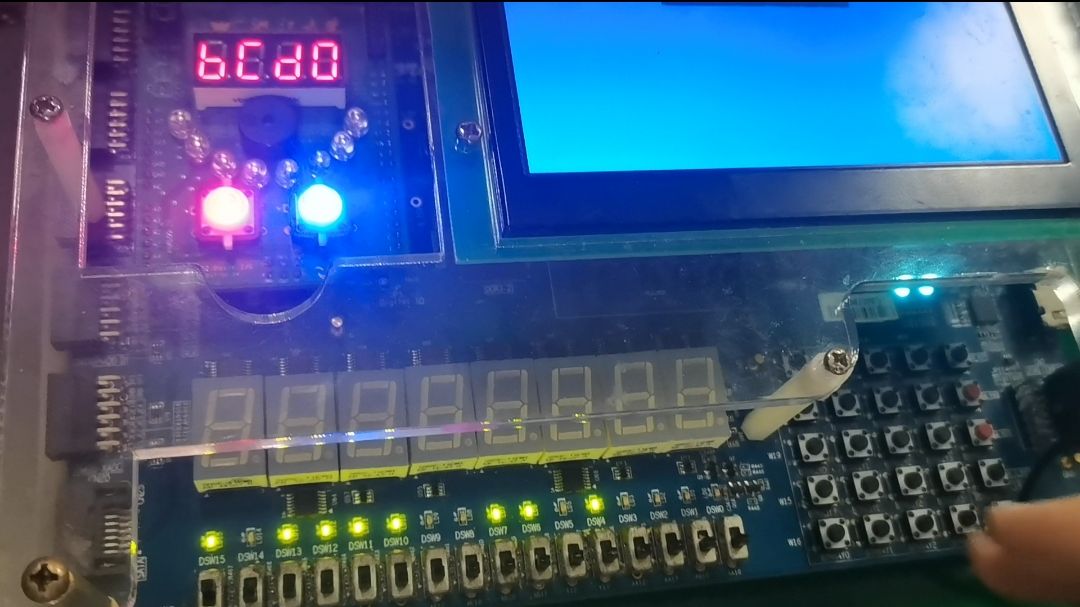
NET "sclrn"LOC = N24 | IOSTANDARD = LVCMOS33;

NET "sout"LOC = M26 | IOSTANDARD = LVCMOS33;

NET "LEDEN"LOC = P18 | IOSTANDARD = LVCMOS33;

**二． 实验结果与分析**

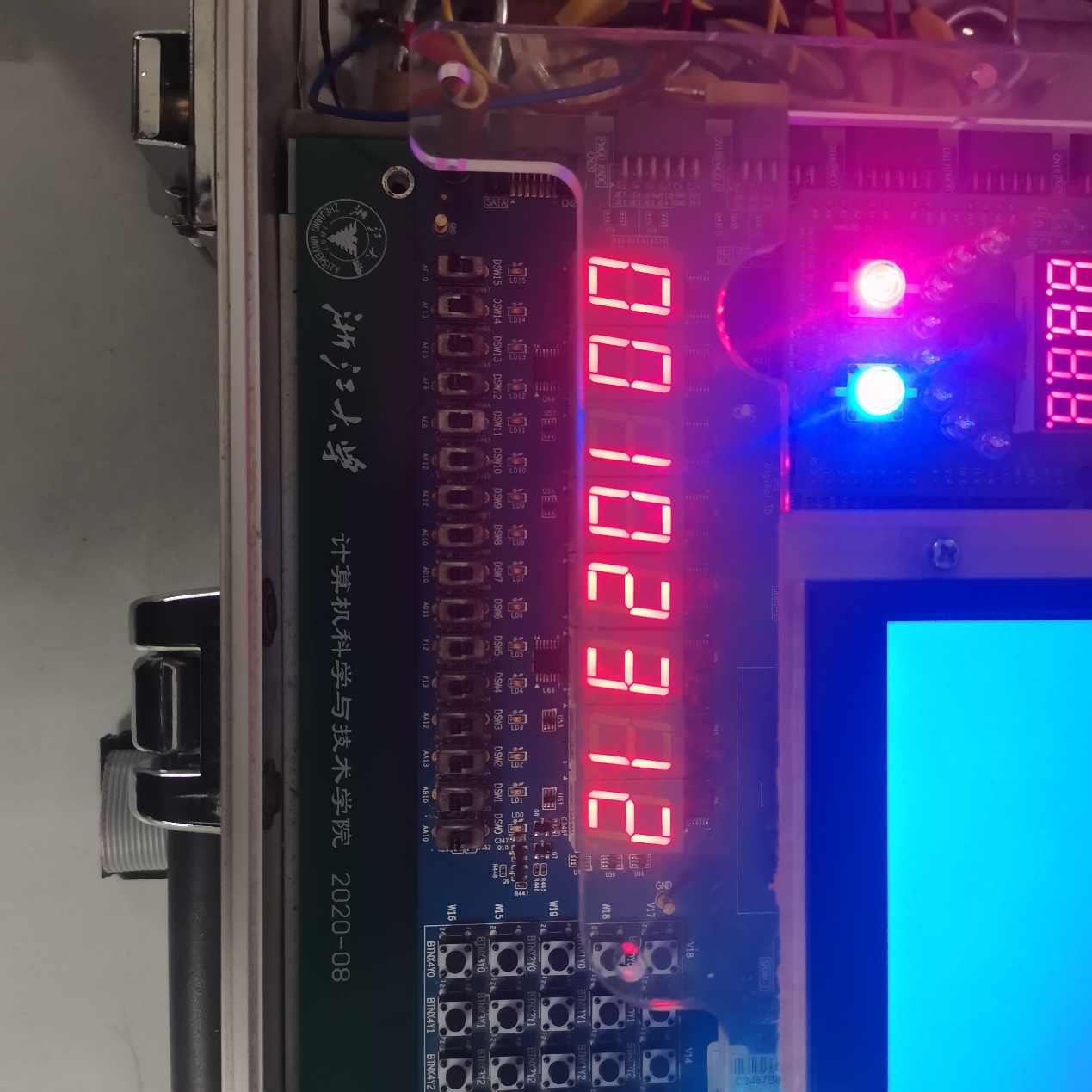
按下按钮，数字相应增大，按下开关，灯随之而变化







下面显示出自己的学号



**三．心得体会**

这次实验由三个任务组成，非常综合。对于第二个实验，我使用了老师上课的方法进行了编写，但是其中碰到了一个小问题就是，一开始不小心把调整灯的开关给定在了非常奇怪的位置上，导致我一开始按了许久灯都没用变化，后来在随便实验的时候突然发生了变化，才想可能是我在引脚中出现了问题，于是再去修改，才得到了一个十分正常的开关。而在学号实验和第一个实验中，则是没用碰到太大的困难。

总的来说，做完这个实验，这个学期的lab也就全部结束了，我感觉全部下来，自己进步了许多，了解了ise很多知识，很多实验也和课堂上的知识紧密相关，自己成长了许多，但是仍然有许多不足的地方，很多实验的细节我还是不知道怎么解释，一些地方还是依靠同伴的帮助才得到了解决，在寒假的时候，我打算好好重温一下其中的实验，能让自己对曾经不懂的地方有更多的了解。

**个人照片**

