

**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 11 月 28日 |

**实验 8——****加法器、加减法器和 ALU 基本原理与设计**

课程名称： 数字逻辑设计 实验类型： 综合

实验项目名称： 加法器、加减法器和 ALU 基本原理与设计

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 11 月 10 日

1. **操作方法与实验步骤**

#### 数据选择器设计

新建工程,工程名称用 MyALU。Top Level Source Type 用 HDL。

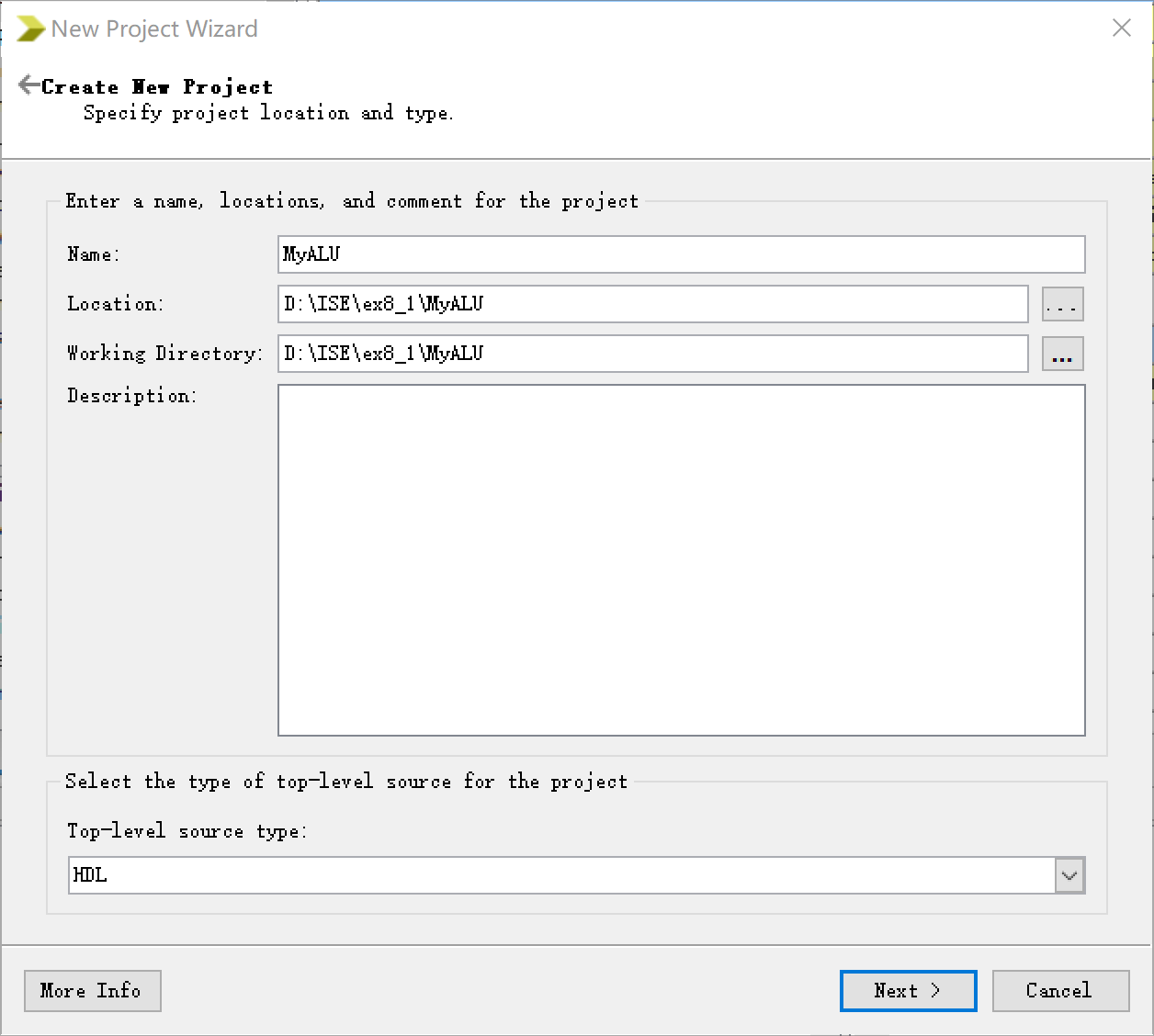


图1：创建文件

##### 绘图并编译运行

新建源文件,类型是 Schematic,文件名称用 AddSub1b。按照原理图方式进行设计。

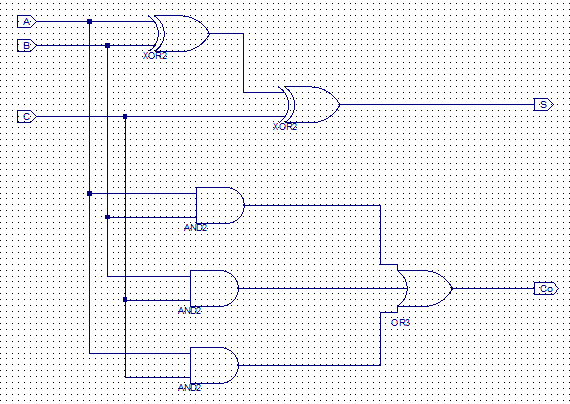


图2：原理图绘制

新建源文件类型是 Schematic。文件名称用 AddSub4b。原理图方式进行设计，调用前面设计的 AddSub1b。

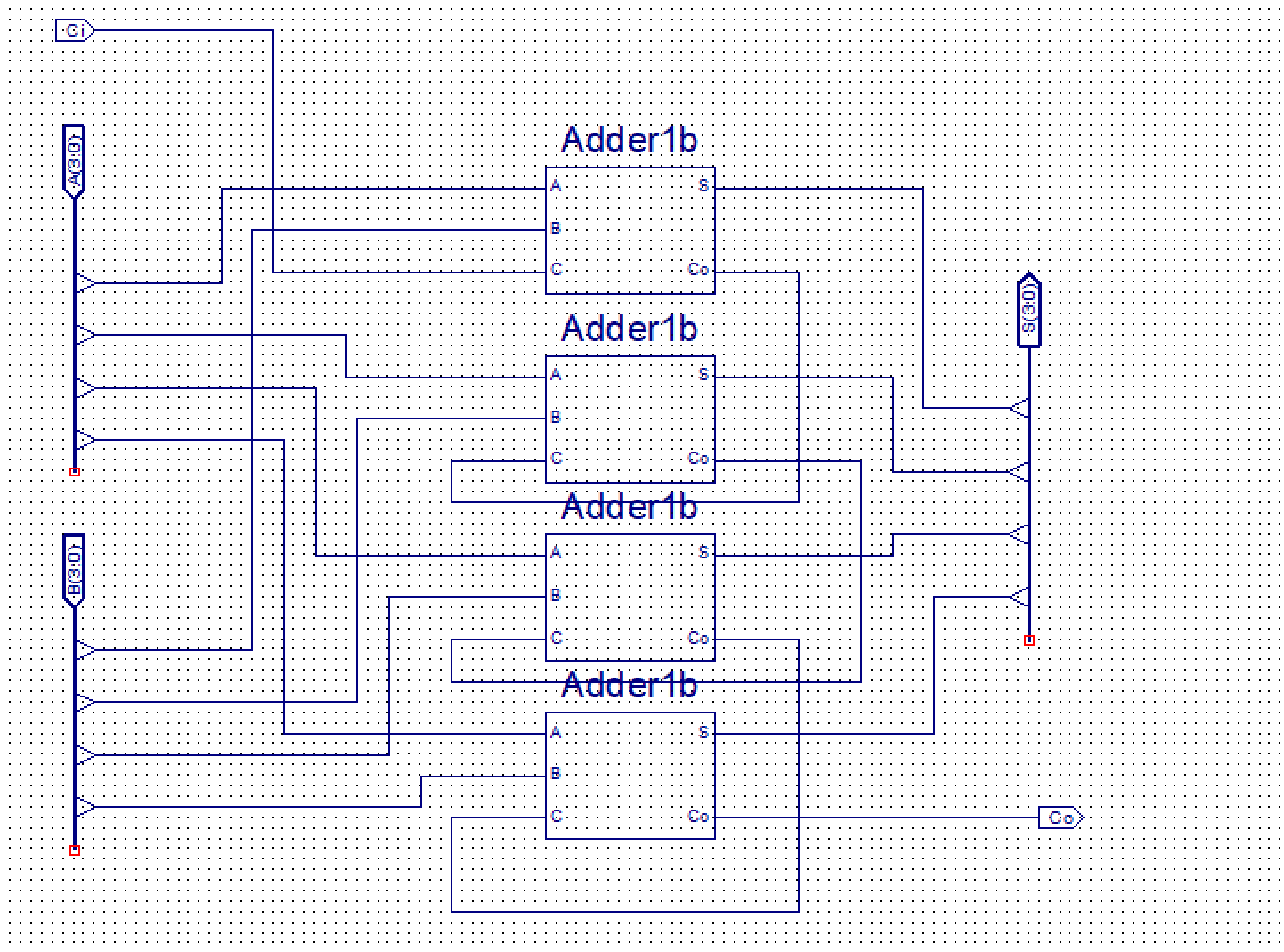


图3：原理图绘制

**1.3 进行波形仿真**

激励输入至少 4 组。

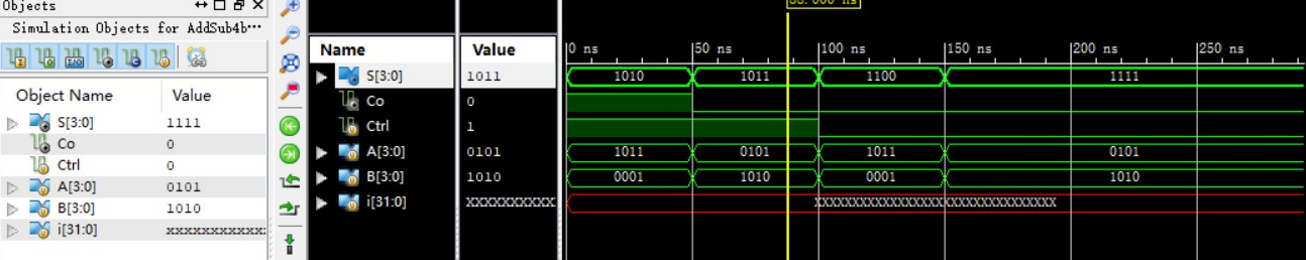


图4：激励仿真1

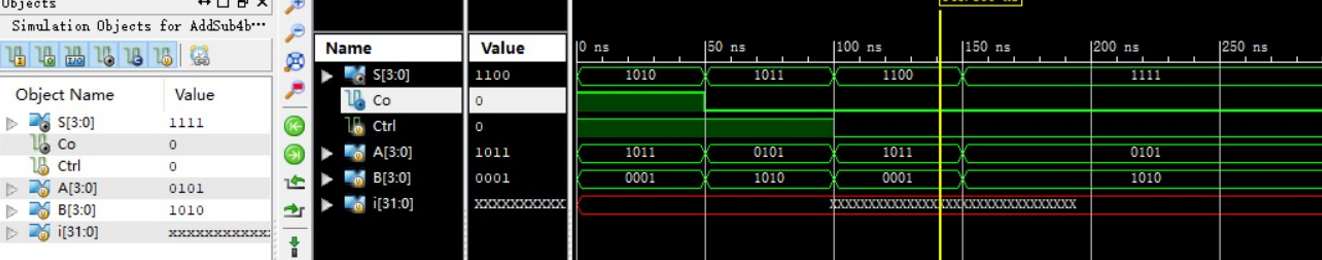


图5：激励仿真2

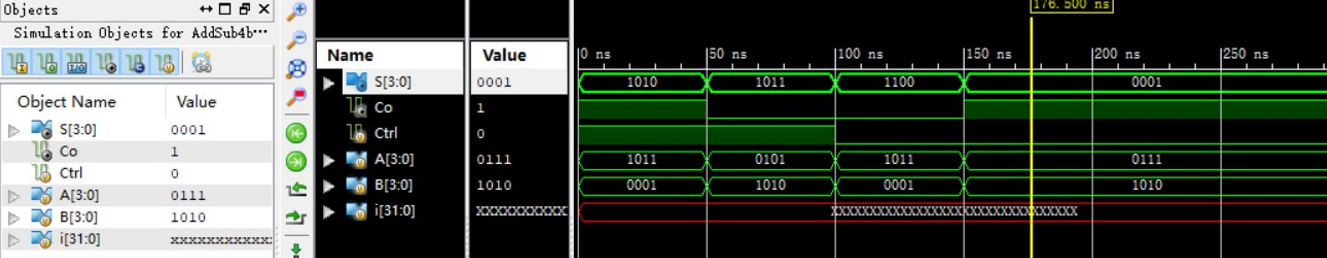


图6：激励仿真3

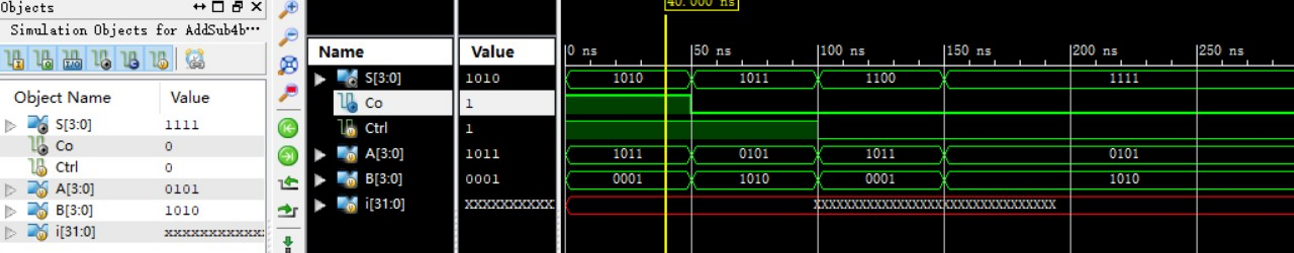


图7：激励仿真4

#### 1.4 ALU 设计

新建源文件，类型是 Verilog 或 Schematic。文件名称用 ALU。按照原理图方式进行设计，分别要绘制两个新的子图与一个大图。

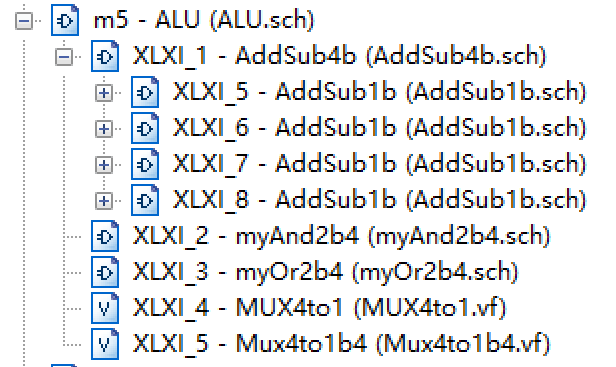


图8：文件创立



图9：原理图绘制

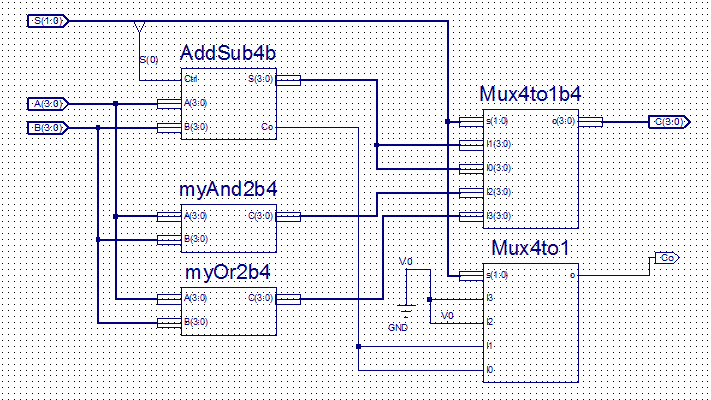


图10：原理图绘制

**1.5 进行波形仿真**

激励输入至少 4 组。覆盖 4 种操作。

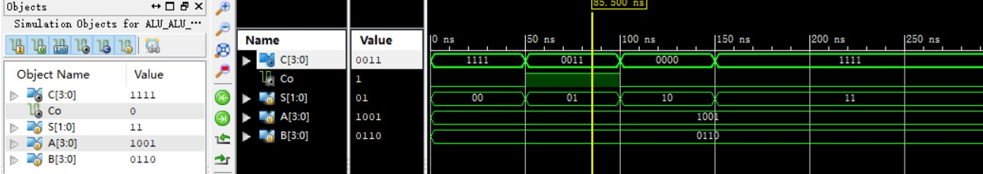


图11：激励仿真1

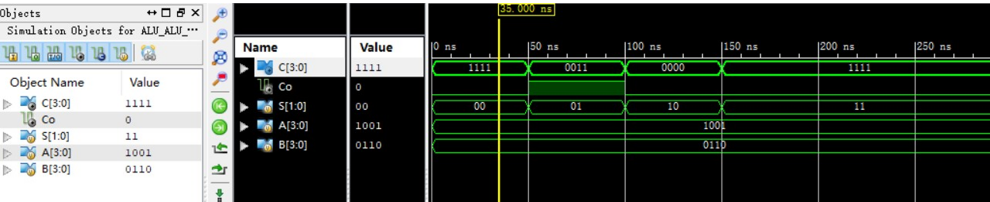


图12：激励仿真2

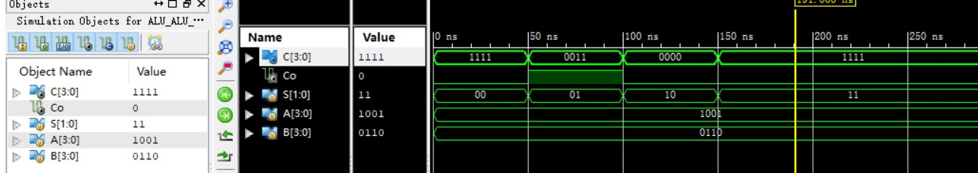


图13：激励仿真3

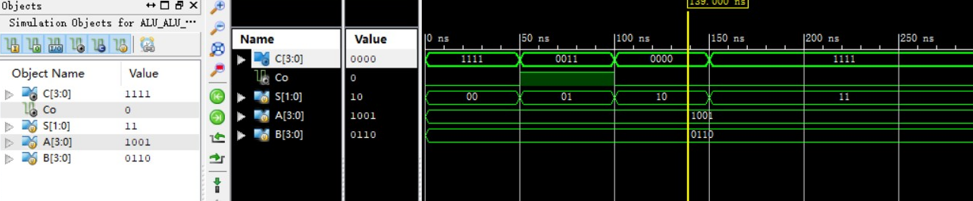
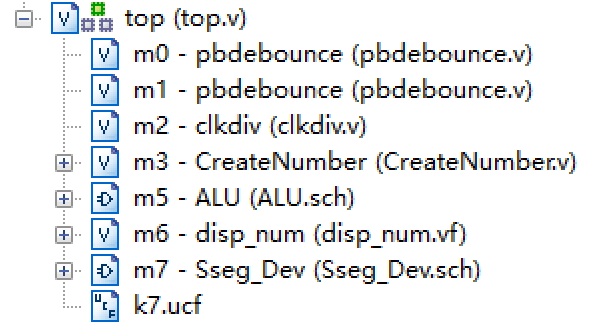


图14：激励仿真4

**1.6 Top设计**

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



调用 pbdebounce、AddSub4b、pbdebounce、clkdiv、disp\_num、CreateNumber 模块。

图15：调用模块

##### 1.7 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”。

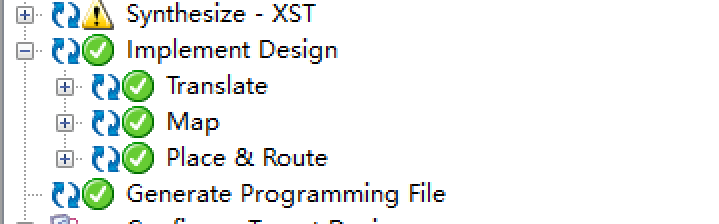


图16：Process调试

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。



图17：Scan调试

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。

##### 1.8 代码块

1. pbdebounce

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

2.clkdiv

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. CreateNumber

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'b1010\_1011\_1100\_1101;

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

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

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

AddSub4b 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

4.k7引脚

NET"clk"LOC=AC18 | IOSTANDARD=LVCMOS15;

NET"SW1[0]"LOC=AF13 | IOSTANDARD=LVCMOS15;

NET"SW1[1]"LOC=AF10 | IOSTANDARD=LVCMOS15;

NET"SW2[0]"LOC=AA10 | IOSTANDARD=LVCMOS15;

NET"SW2[1]"LOC=AB10 | IOSTANDARD=LVCMOS15;

NET"BTN[0]"LOC=W14 | IOSTANDARD=LVCMOS15;

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

NET"BTN[1]"LOC=V14 | IOSTANDARD=LVCMOS15;

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

NET"BTNX4"LOC=W16 | IOSTANDARD=LVCMOS15;

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;

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;

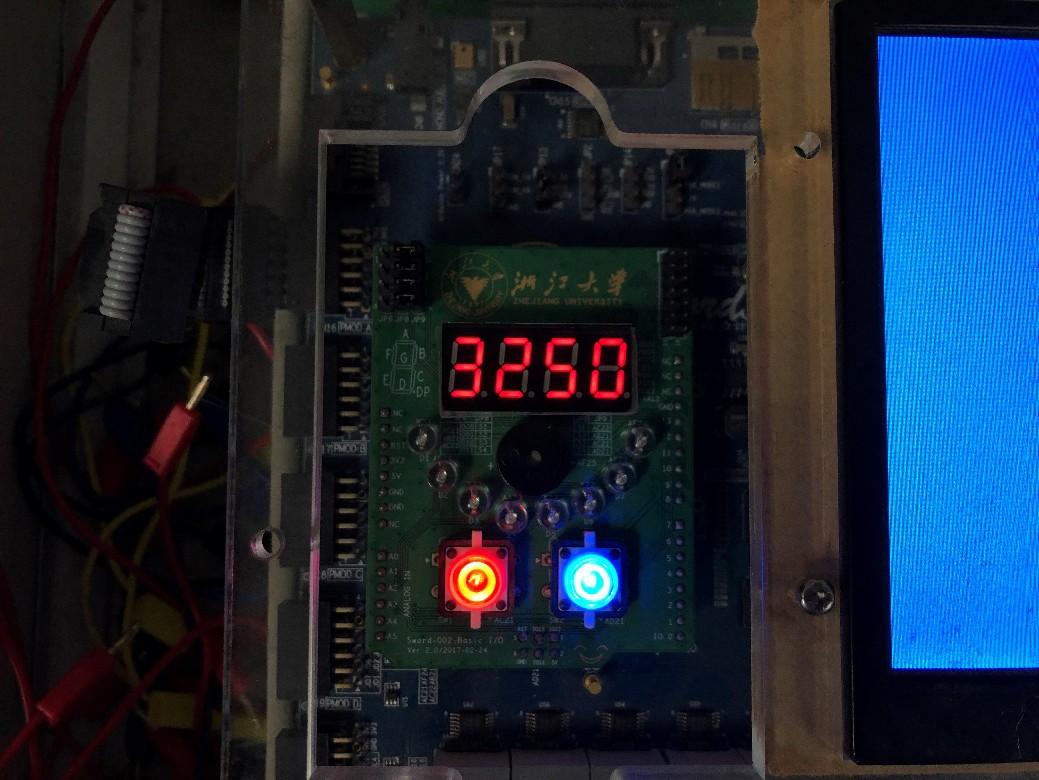
NET"AN[1]"LOC=AC21 | IOSTANDARD=LVCMOS33;

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

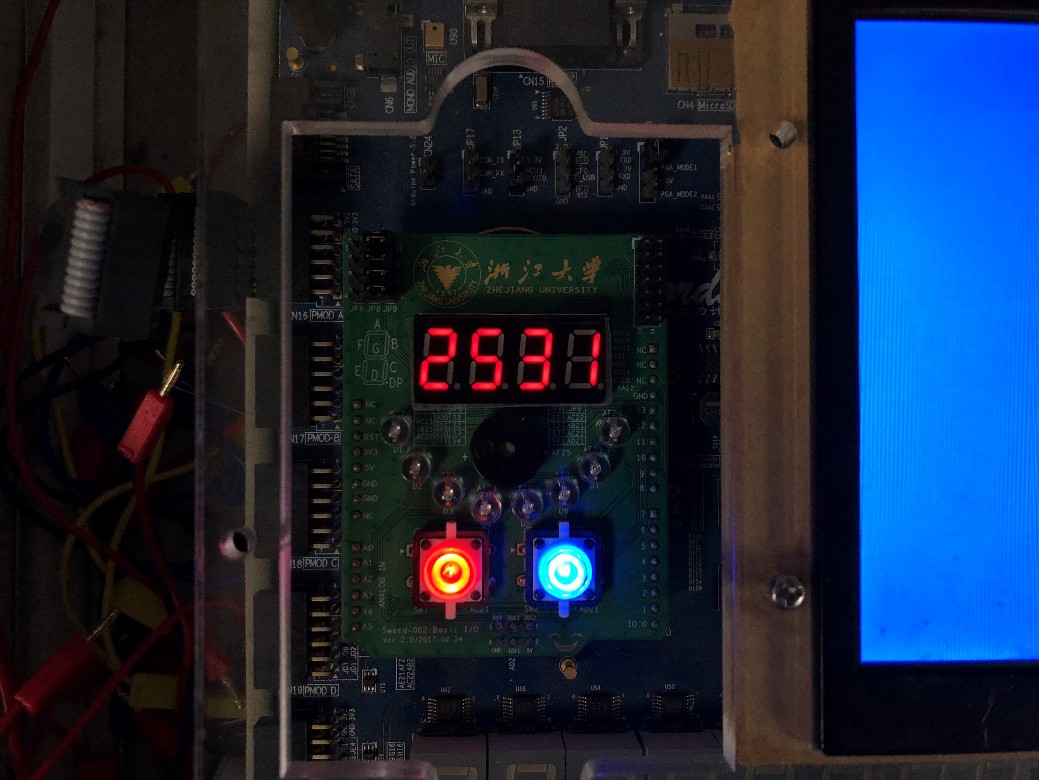
NET"AN[3]"LOC=AC22 | IOSTANDARD=LVCMOS33;

1. **实验结果与分析**

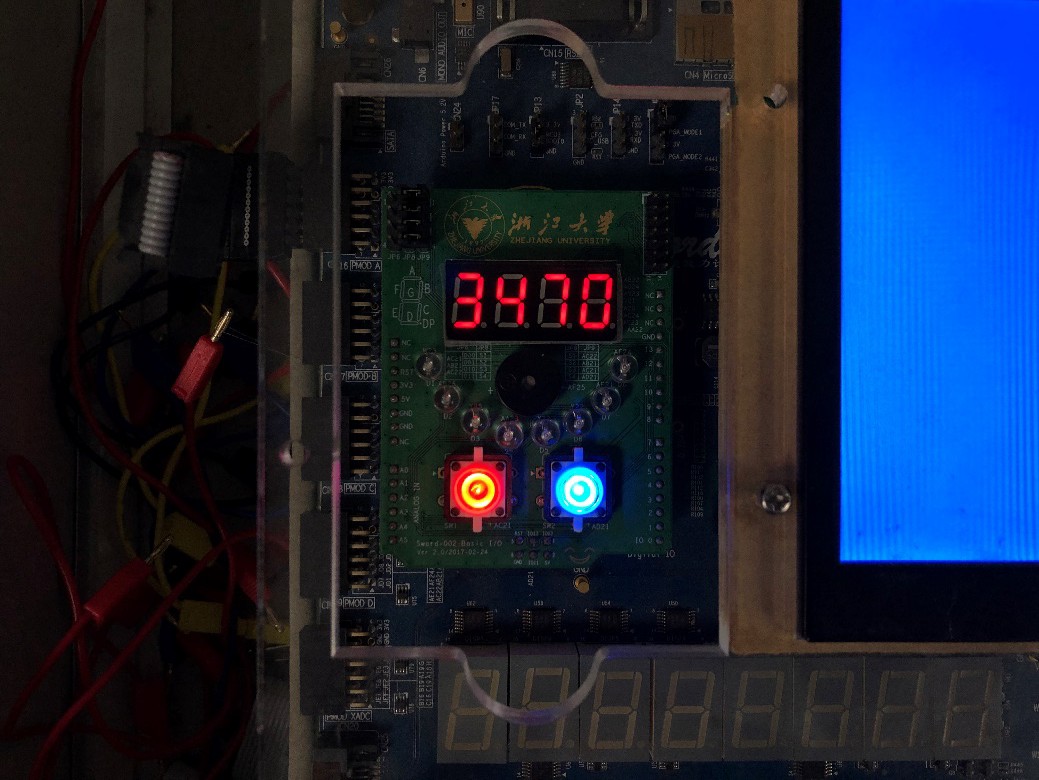
加法显示：3+2=5



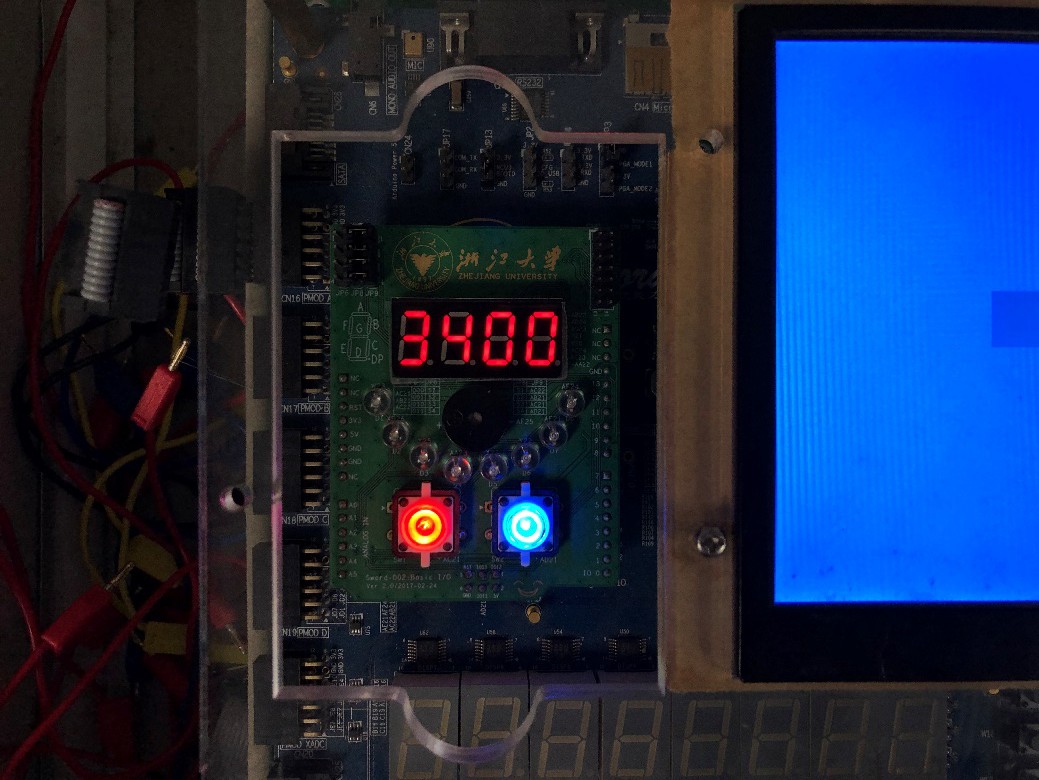
减法显示：5-2=3



或运算



与运算



1. **讨论与心得**

这次实验的过程中，主要的问题是要进行多次引入，其中操作的问题就是要进行代码的输入。而从逻辑功能方面，这次进行的ALU可以通过改变一个控制信号的功能键，从而实现加减法和或与法的功能，尤其是，其中有时候会出现的错误主要是因为引脚没有把自己希望实现的功能放在对应键所造成的，所以一定要注重引脚的编写。其他方面，我发现每次都要及时进行一个仿真模拟，这样可以帮助我在整个工程的过程中找到问题。

**实验 9——锁存器与触发器基本原理**

课程名称： 数字逻辑设计 实验类型： 综合

实验项目名称： 锁存器与触发器基本原理

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 11 月 17 日

1. **操作方法与实验步骤**

**1.1 基本 SR 锁存器**

新建工程 MyLATCHS。并新建源文件 SR\_LATCH.sch。用原理图方式设计。

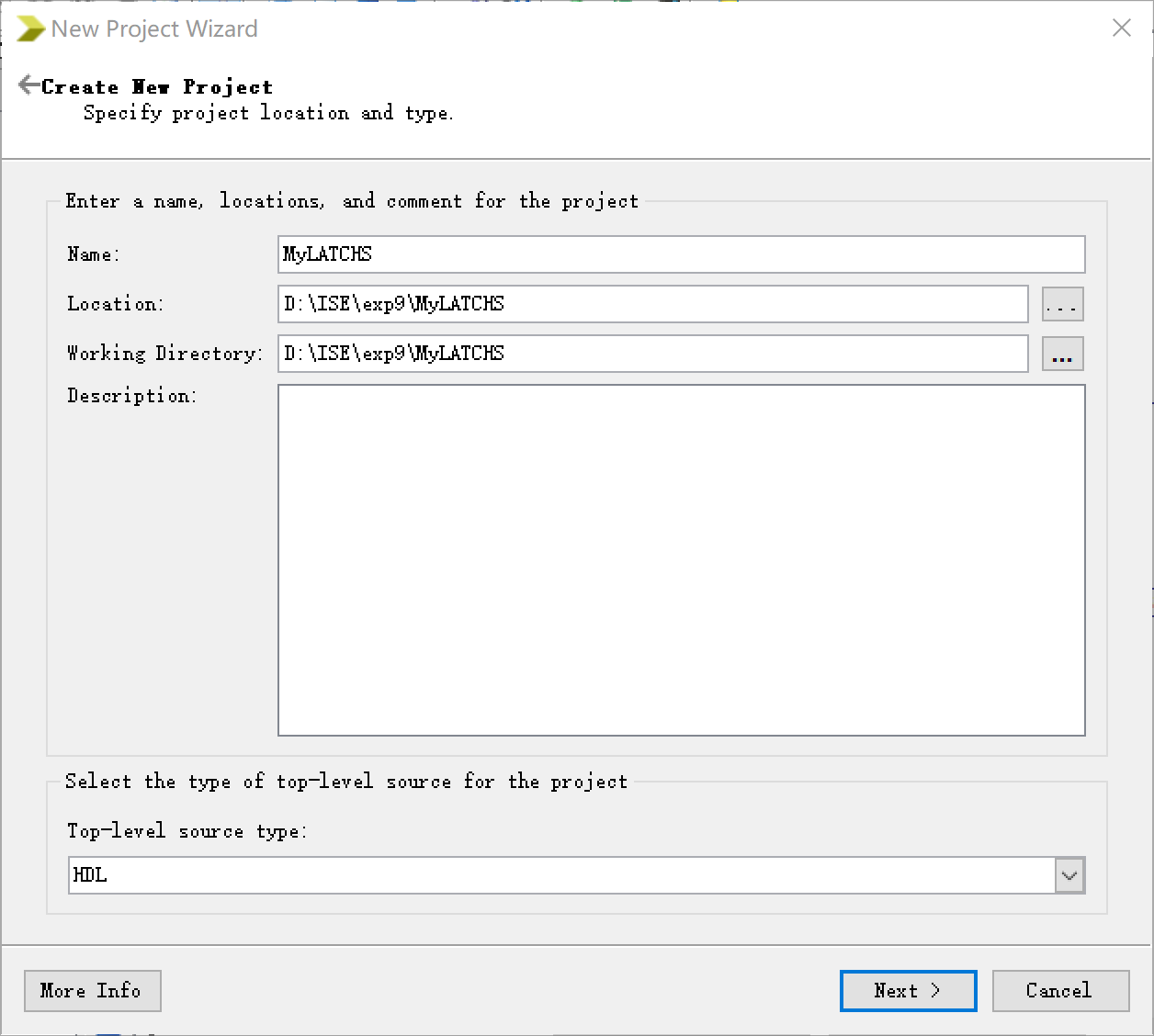


图1：工程创建

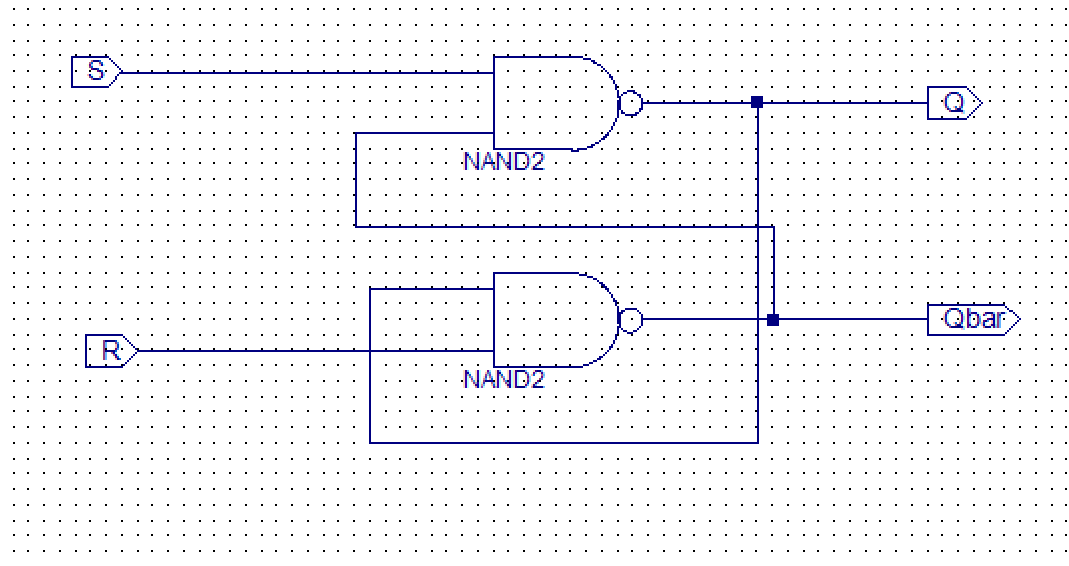


图2：原理图绘制

**1.2 仿真**

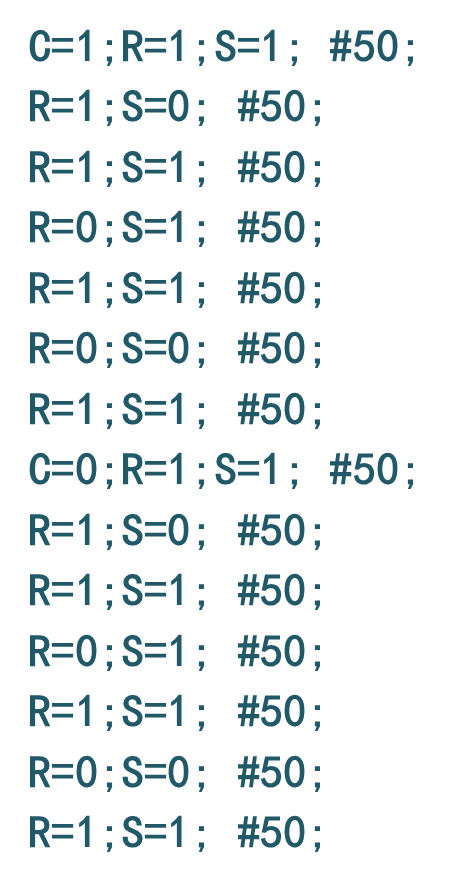
激励代码：



图3：仿真

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 时间 | S | R | 状态 | Q | Qbar |
| 0-50ns | 1 | 1 | 未知 | - | - |
| 50-100ns | 0 | 1 | 置1 | 1 | 0 |
| 100-150ns | 1 | 1 | 保持 | 1 | 0 |
| 150-200ns | 1 | 0 | 置0 | 0 | 1 |
| 200-250ns | 1 | 1 | 保持 | 0 | 1 |
| 250-300ns | 0 | 0 | 未定义 | 1 | 1 |
| 300ns后 | 1 | 1 | 不确定 |  |  |

**1.3 门控 SR 锁存器**

新建源文件 CSR\_LATCH.sch，用原理图方式设计。

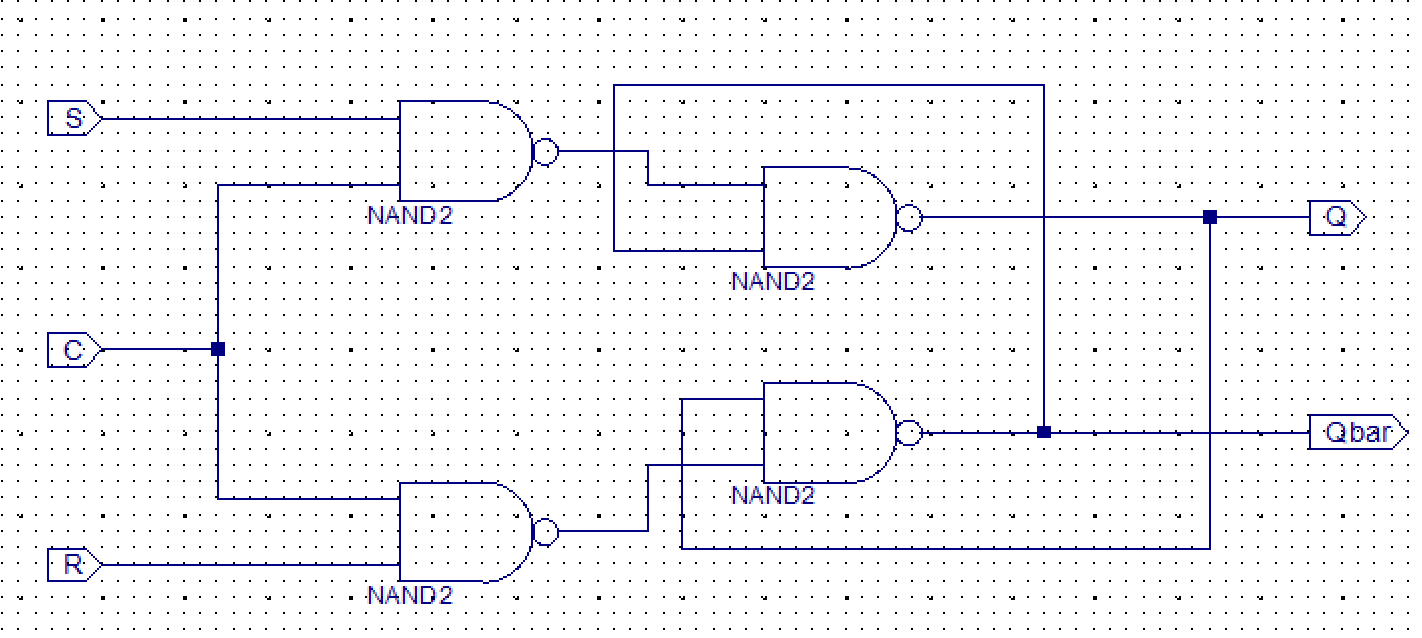
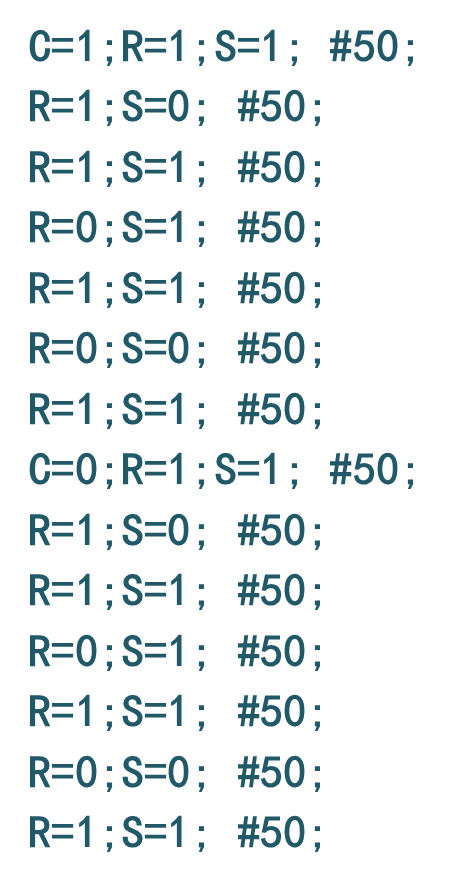


图4：原理图绘制

**1.4 仿真（包含空翻）**



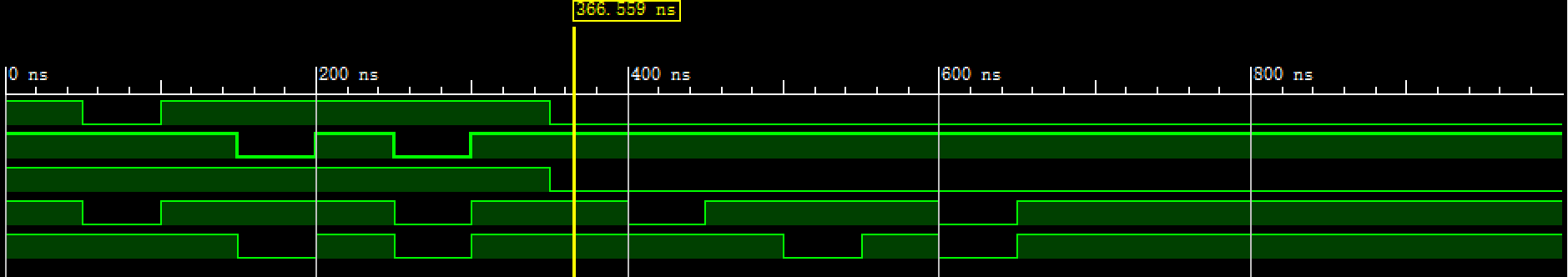


图5：仿真

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 时间 | C | S | R | 状态 | Q | Qbar |
| 0-50ns | 1 | 1 | 1 | 未定义 | 1 | 1 |
| 50-100ns | 1 | 0 | 1 | 置0 | 0 | 1 |
| 100-150ns | 1 | 1 | 1 | 未定义 | 1 | 1 |
| 150-200ns | 1 | 1 | 0 | 置1 | 1 | 0 |
| 200-250ns | 1 | 1 | 1 | 未定义 | 1 | 1 |
| 250-300ns | 1 | 0 | 0 | 不确定 | 1 | 0 |
| 300-350ns | 1 | 1 | 1 | 未定义 | 1 | 1 |
| 350ns后 | 0 | 1 | 1 | 不确定 | 0 | 1 |

**1.5 D 锁存器**

新建源文件 D\_LATCH.sch。用原理图方式设计。

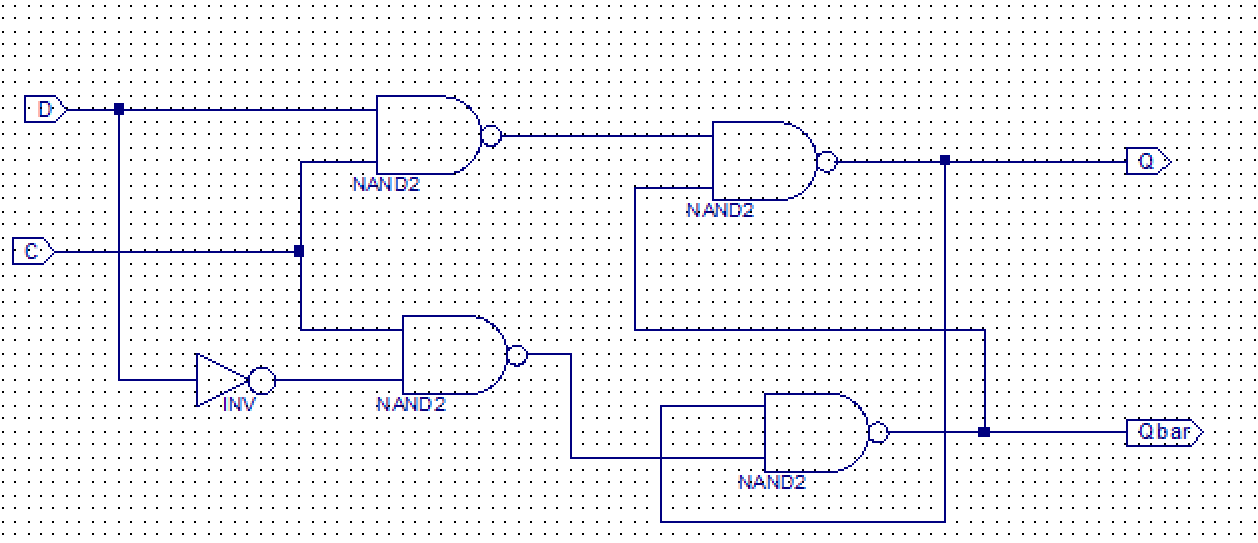
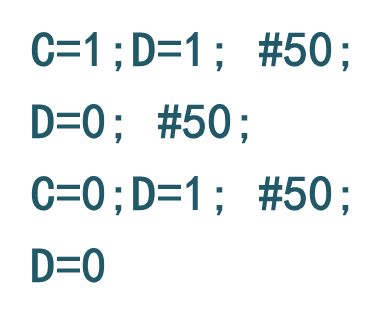


图6：原理图

**1.6 仿真（包含空翻）**



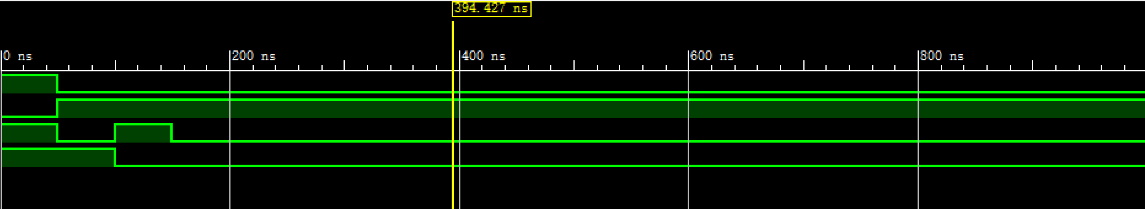


图7：仿真

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 时间 | C | D | 状态 | Q | Qbar |
| 0-50ns | 1 | 1 | 置1 | 1 | 0 |
| 50-100ns | 1 | 0 | 置0 | 0 | 1 |
| 100-150ns | 0 | 1 | 保持 | 0 | 1 |

* 1. **SR 主从触发器**

新建源文件 MS\_FLIPFLOP.sch。用原理图方式设计。调用 CSR\_LATCH 实现。

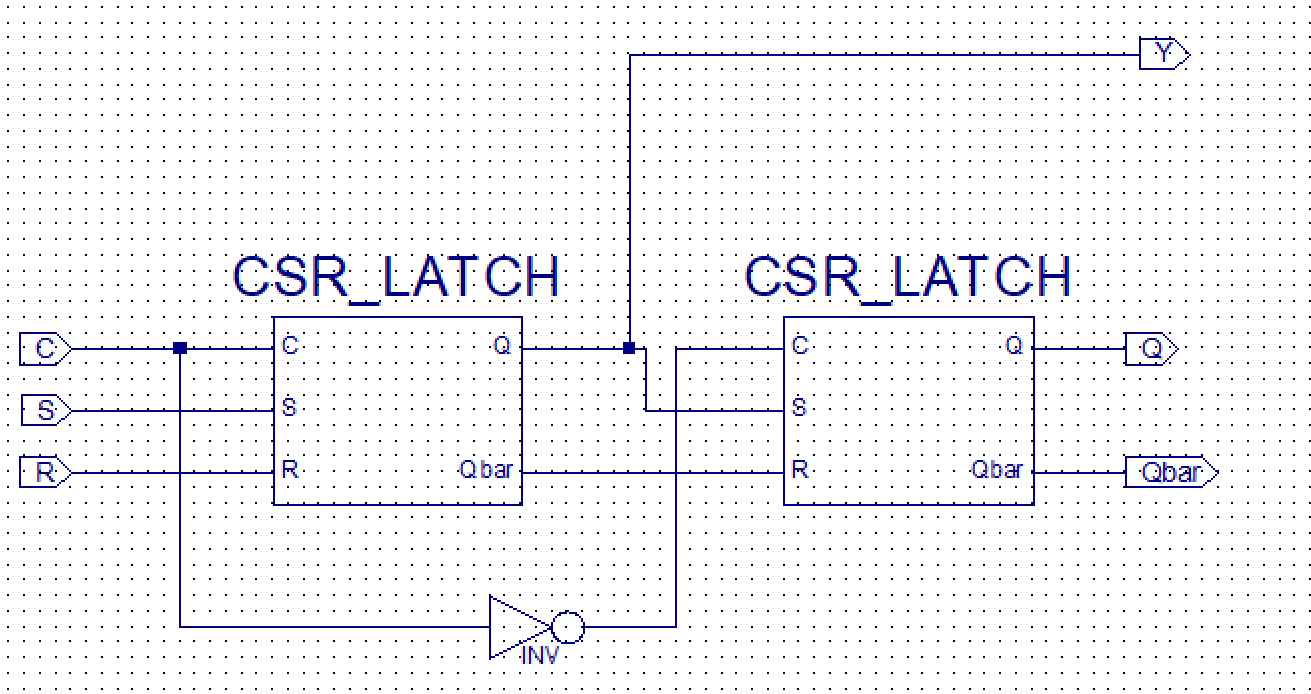
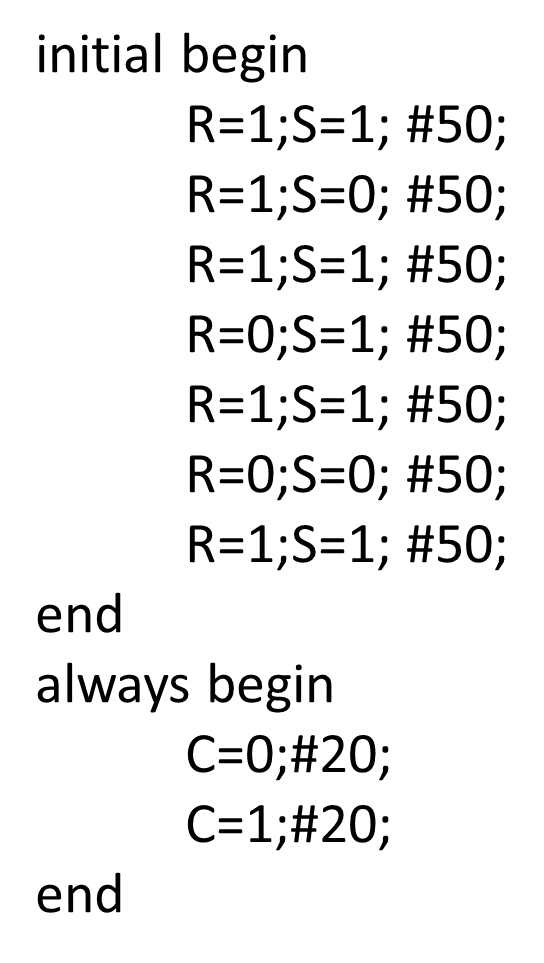


图8：原理图绘制

**1.8 仿真（包含一次性采样）**



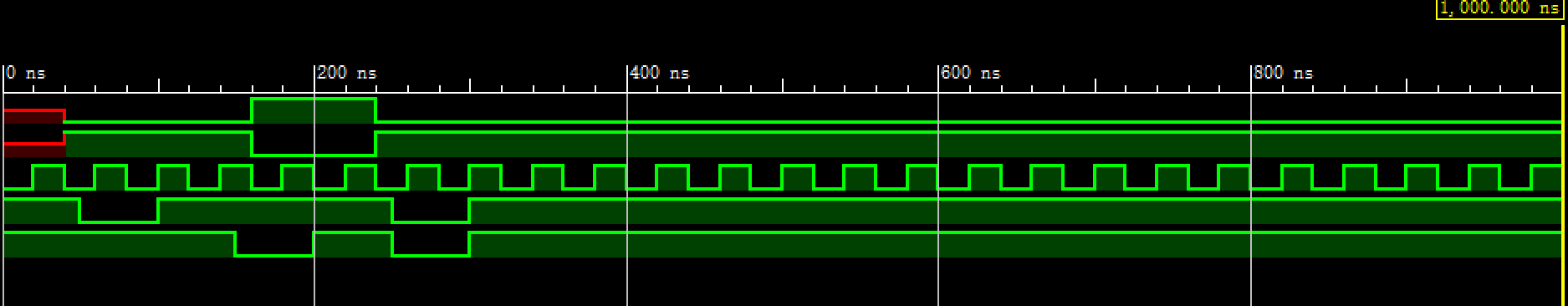


图9：仿真

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 时间 | C | R | S | Q | Qbar |
| 0-100ns | 1 | 变化 | 变化 | 1 | 0 |
| 100-150ns | 0 | 1 | 0 | 0 | 1 |
| 150-200ns | 1 | 变化 | 变化 | 0 | 1 |
| 200-250ns | 0 | 0 | 1 | 1 | 0 |
| 250-300ns | 1 | 变化 | 变化 | 1 | 0 |
| 300-350ns | 0 | 0 | 0 | 1 | 0 |
| 350-400ns | 1 | 变化 | 变化 | 1 | 0 |

**1.9 D触发器**

新建文件D\_FLIPFLOP.sch，按照原理图绘图。

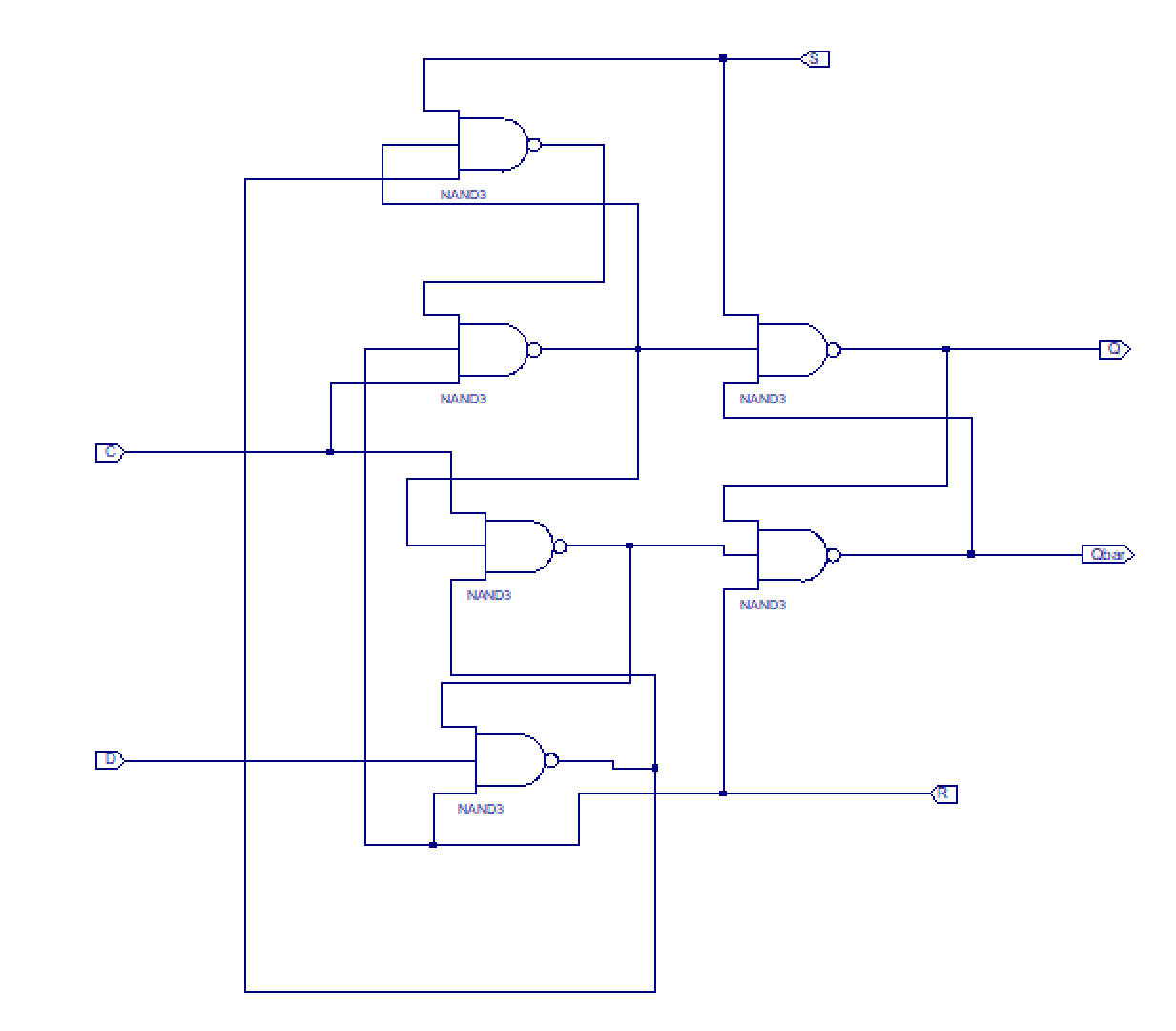
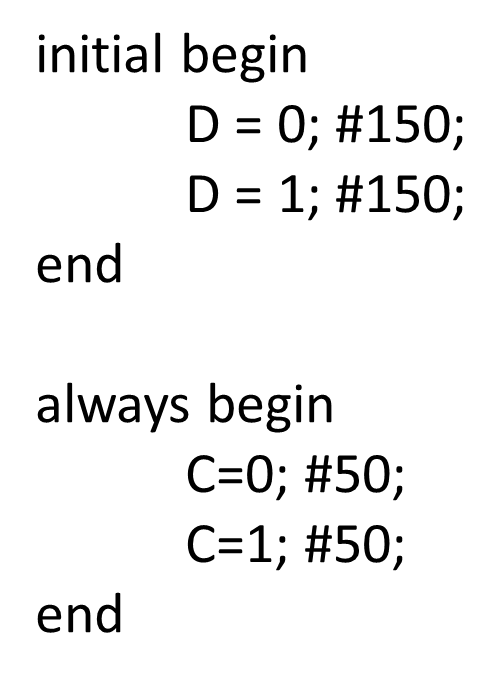


图10：原理图绘制

**1.10 仿真**

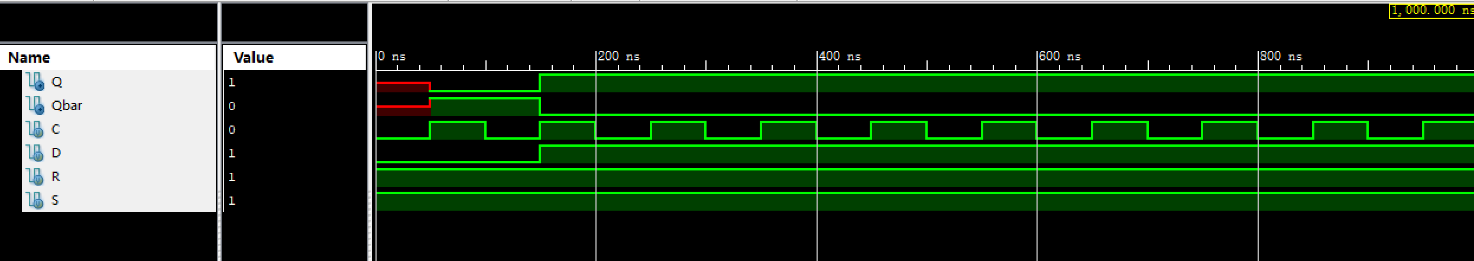


图11：仿真

**1.11 顶层模块设计**

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

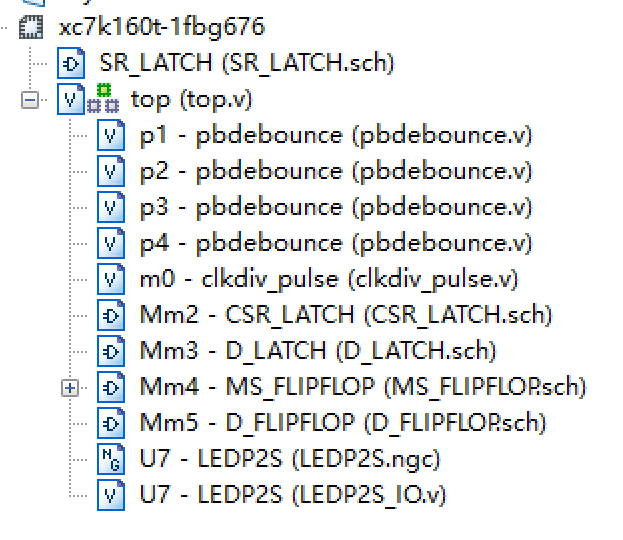


图12：引入设计

##### 1.12 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”。

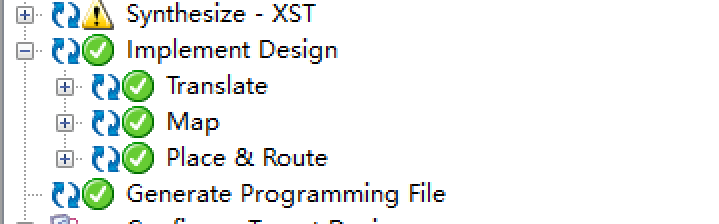


图13：调试

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。



图14：调试

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。

**1.13 代码块**

1. pbdebounce

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

2. clkdiv

module clkdiv\_pulse(input clk,

input rst,

input Sel\_CLK,

input pulse,

output CK,

output reg [31:0] clkdiv

);

always @ (posedge clk or posedge rst) begin

if (rst) clkdiv<=0;

else clkdiv <= clkdiv + 1'b1;

end

assign CK = (Sel\_CLK) ? ~pulse : clkdiv[26];

endmodule

3.引脚代码

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

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

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

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

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

NET "clk" TNM\_NET = TM\_CLK;

TIMESPEC TS\_CLK\_100M = PERIOD "TM\_CLK" 10ns HIGH 50%;

NET "BTN[0]" LOC = V18 | IOSTANDARD = LVCMOS18;

NET "BTN[0]" CLOCK\_DEDICATED\_ROUTE = FALSE;

NET "BTN[1]" LOC = V19 | IOSTANDARD = LVCMOS18;

NET "BTN[1]" CLOCK\_DEDICATED\_ROUTE = FALSE;

NET "BTN[2]" LOC = V14 | IOSTANDARD = LVCMOS18;

NET "BTN[2]" CLOCK\_DEDICATED\_ROUTE = FALSE;

NET "BTN[3]" LOC = W14 | IOSTANDARD = LVCMOS18;

NET "BTN[3]" CLOCK\_DEDICATED\_ROUTE = FALSE;

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 "SW[5]"LOC =Y12 | IOSTANDARD = LVCMOS15;

NET "SW[6]"LOC =AD11 | IOSTANDARD = LVCMOS15;

NET "SW[7]"LOC =AD10 | IOSTANDARD = LVCMOS15;

NET "SW[8]"LOC =AE10 | IOSTANDARD = LVCMOS15;

NET "SW[9]"LOC =AE12 | IOSTANDARD = LVCMOS15;

NET "SW[10]"LOC =AF12 | IOSTANDARD = LVCMOS15;

NET "SW[11]"LOC =AE8 | IOSTANDARD = LVCMOS15;

NET "SW[12]"LOC =AF8 | IOSTANDARD = LVCMOS15;

NET "SW[13]"LOC =AE13 | IOSTANDARD = LVCMOS15;

NET "SW[14]"LOC =AF13 | IOSTANDARD = LVCMOS15;

NET "SW[15]"LOC =AF10 | IOSTANDARD = LVCMOS15;

NET"LED[0]"LOC=W23 | IOSTANDARD=LVCMOS33;

NET"LED[1]"LOC=AB26 | IOSTANDARD=LVCMOS33;

NET"LED[2]"LOC=Y25 | IOSTANDARD=LVCMOS33;

NET"LED[3]"LOC=AA23 | IOSTANDARD=LVCMOS33;

NET"LED[4]"LOC=Y23 | IOSTANDARD=LVCMOS33;

NET"LED[5]"LOC=Y22 | IOSTANDARD=LVCMOS33;

NET"LED[6]"LOC=AE21 | IOSTANDARD=LVCMOS33;

NET"LED[7]"LOC=AF24 | IOSTANDARD=LVCMOS33;

NET "BTNX3" LOC = W15 | IOSTANDARD = LVCMOS18;

#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;

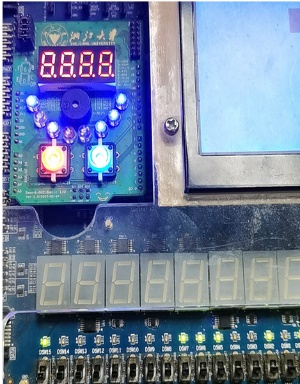
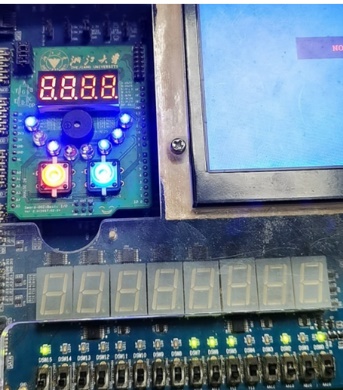
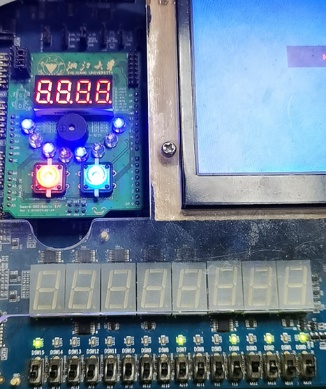
#NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;

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

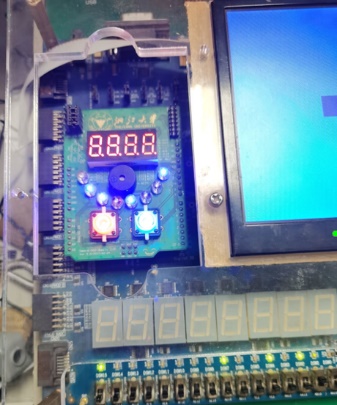
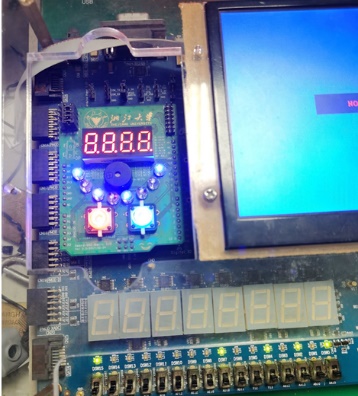
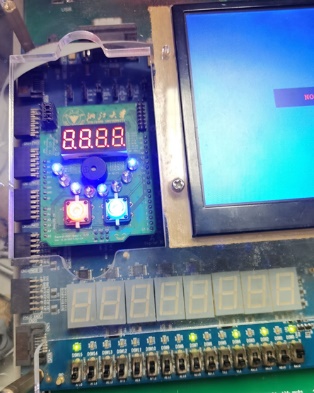
#NET "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33;

**二. 实验结果分析**

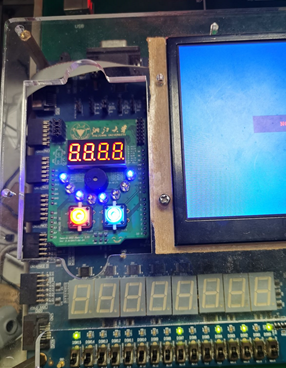
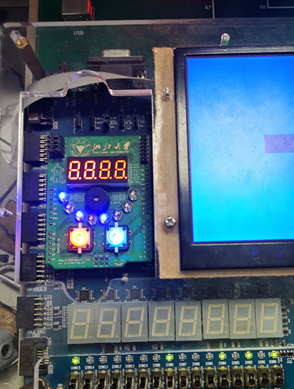
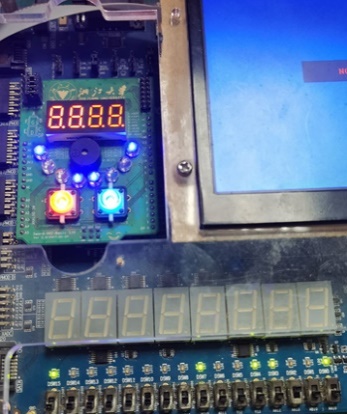
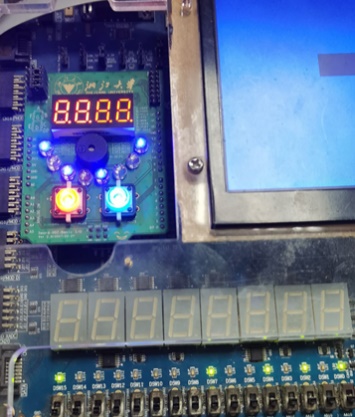
1. SR主从触发器（01，00，10，11）

![C:\Users\lenovo\AppData\Roaming\Tencent\Users\2811668688\QQ\WinTemp\RichOle\2YU2@`A$4(M2P](IVGNTLDJ.png](data:image/jpeg;base64,)  

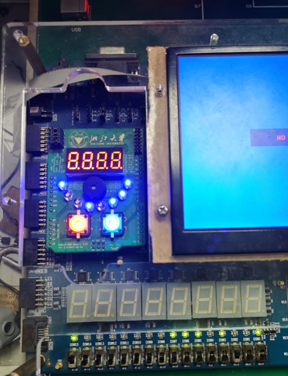
2. SR门控锁存器（01，00，10，11）



3. D锁存器



4. 时钟选择改为按钮单步触发



**三．心得**

本次实验要做出的图像其实并不复杂，各种操作也都很简单，但是要注意的问题是理解其中的含义，本次实验设计到了锁存器和触发器，是和课堂上面的内容息息相关的，因此一定要把数逻课上的内容理解清楚，才可以理解本次实验进行的sim仿真操作。

特别的，我认为触发器其实是后续时序电路的重要基础，只有熟悉了每种触发器如何使用，次态方程是什么，才能在后面要自行设计一个电路的时候得心应手。

**实验 10——同步时序电路设计**

课程名称： 数字逻辑设计 实验类型： 综合

实验项目名称： 同步时序电路设计

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 11 月 24 日

1. **操作方法与实验步骤**

**1.1 4 位同步二进制计数器**

新建工程，工程名称用 MyCounter，Top Level Source Type 用 HDL。并且新建源文件，类型是 Schematic。文件名称用 Counter4b。按照原理图方式进行设计.

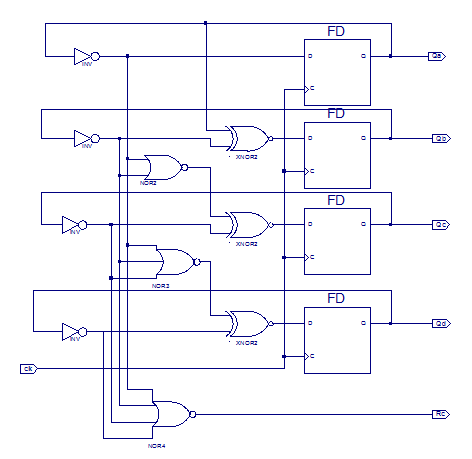


图1：原理图绘制

**1.2 进行波形仿真**



图2：仿真

**1.3 clk\_1s 设计**

新建源文件，用作时钟类型是 Verilog。文件名称用 clk\_1s，采用Verilog 行为描述

也可以用原语描述进行设计：

module counter\_1s(clk, clk\_1s);

input wire clk;

output reg clk\_1s;

reg [31:0] cnt;

always @ (posedge clk) begin

if (cnt < 50\_000\_000) begin

cnt <= cnt + 1;

end else begin

cnt <= 0;

clk\_1s <= ~clk\_1s;

end

end

endmodule

图3：代码

**1.4 顶层设计**

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

**1.5 16 位可逆二进制同步计数器**

新建工程，工程名称用 myRevCounter。Top Level Source Type 用 HDL。新建源文件

类型是 Verilog。文件名称用 RevCounter。进行结构化描述方式进行设计。

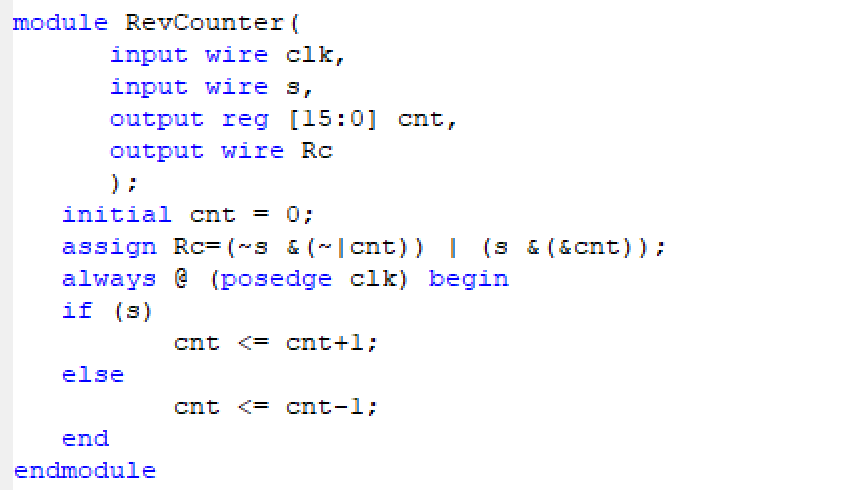


图4：代码

**1.6 波形仿真（包含正向计数和反向计数）**

正向计数



图5：正向计数

逆向计数

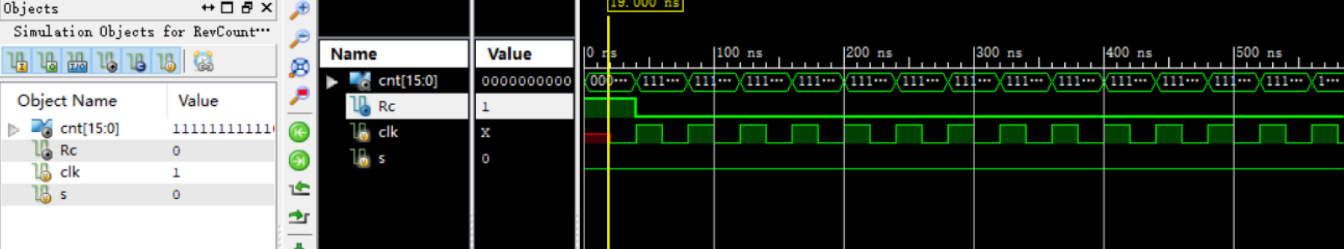


图6：逆向计数

**1.7 clk\_100ms设计**

新建源文件，设计 100ms 时钟类型是 Verilog。文件名称用 clk\_100ms。采用Verilog 行为描述



图6：代码

**1.8 顶层设计**

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

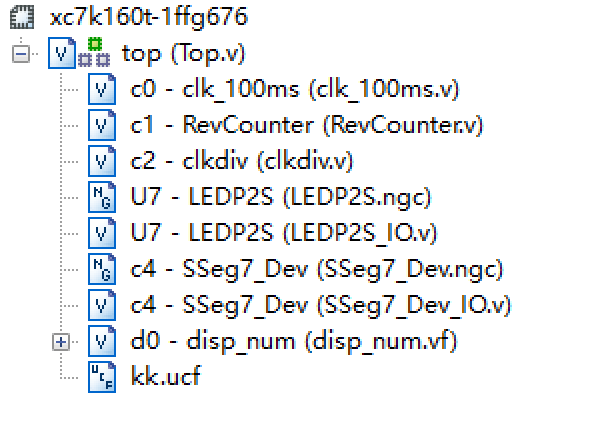


图7：设计

##### 1.9 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”。

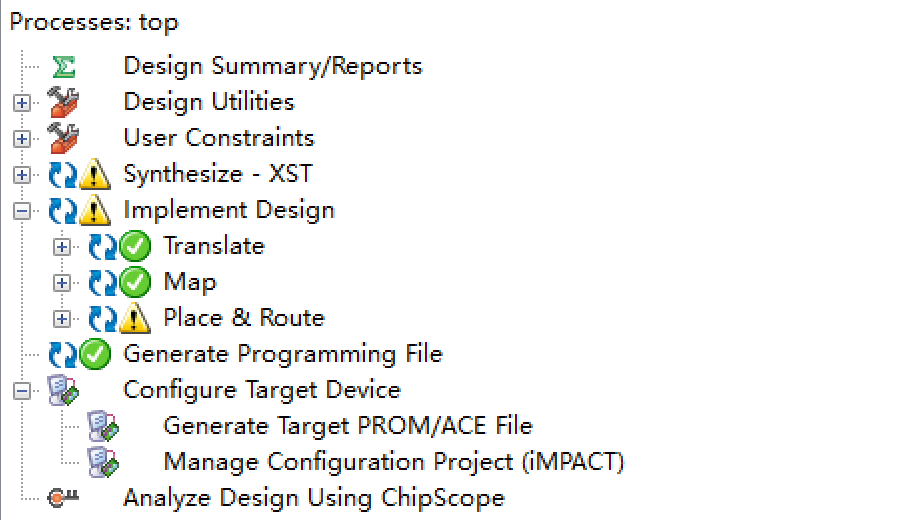


图8：调试

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。



图9：调试

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。

**1.10 代码块**

1. top块

module top(

input wire clk,

input wire [7:0] SW,

output wire [7:0] SEGMENT,

output wire [3:0] AN,

output wire Rc,

output wire [7:0] LED,

output wire ledclk,

output wire ledsout,

output wire ledclrn,

output wire LEDEN,

output wire seg\_clk,

output wire seg\_clrn,

output wire seg\_sout,

output wire SEG\_PEN

);

wire clk\_100ms;

wire [15:0] num;

wire [31:0] div;

wire NLED0;

assign LED[0] = ~NLED0;

assign LED[6:2] = SW[6:2];

clk\_100ms c0(.clk(clk),.clk\_100ms(clk\_100ms));

RevCounter c1(.clk(clk\_100ms),.s(SW[0]),.cnt(num),.Rc(LED[7]));

clkdiv c2(.clk(clk),.rst(1'b0),.clkdiv(div[31:0]));

LEDP2S #(.DATA\_BITS(16),.DATA\_COUNT\_BITS(4),.DIR(0))

U7(.clk(clk), .rst(1'b0), .Start(div[20]), .PData({8'hFF,LED}), .sclk(ledclk), .sclrn(ledclrn), .sout(ledsout), .EN(LEDEN));

SSeg7\_Dev c4(.clk(clk),.rst(1'b0),.Start(div[20]),.SW0(SW[1]),.flash(1'b0),

.Hexs({16'hFF\_FF,num}),.point(8'b00001111),.LES(8'b11110000),

.seg\_clk(seg\_clk),.seg\_clrn(seg\_clrn),.seg\_sout(seg\_sout),.SEG\_PEN(SEG\_PEN));

disp\_num d0(.clk(clk),.HEXS(num),.LES(4'b0000),.points(4'b0000),.RST(1'b0),.AN(AN),.Segment(SEGMENT));

endmodule

1. clk\_100ms

module clk\_100ms(

input wire clk,

output reg clk\_100ms

);

reg [31:0] cnt;

initial cnt<=32'b0;

always@(posedge clk) begin

cnt=cnt+1;

if (cnt == 32'd5\_000\_000) begin

clk\_100ms=~clk\_100ms;

cnt=0;

end

end

endmodule

3. RevCounter

module RevCounter(

input wire clk,

input wire s,

output reg [15:0] cnt,

output wire Rc

);

initial cnt = 0;

assign Rc=(~s &(~|cnt)) | (s &(&cnt));

always @ (posedge clk) begin

if (s)

cnt <= cnt+1;

else

cnt <= cnt-1;

end

endmodule

4.clkdiv

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

5.引脚

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

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 "Rc" LOC = AF21 | IOSTANDARD = LVCMOS33;

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 "SW[5]"LOC =Y12 | IOSTANDARD = LVCMOS15;

NET "SW[6]"LOC =AD11 | IOSTANDARD = LVCMOS15;

NET "SW[7]"LOC =AD10 | IOSTANDARD = LVCMOS15;

NET"LED[0]"LOC=W23 | IOSTANDARD=LVCMOS33;

NET"LED[1]"LOC=AB26 | IOSTANDARD=LVCMOS33;

NET"LED[2]"LOC=Y25 | IOSTANDARD=LVCMOS33;

NET"LED[3]"LOC=AA23 | IOSTANDARD=LVCMOS33;

NET"LED[4]"LOC=Y23 | IOSTANDARD=LVCMOS33;

NET"LED[5]"LOC=Y22 | IOSTANDARD=LVCMOS33;

NET"LED[6]"LOC=AE21 | IOSTANDARD=LVCMOS33;

NET"LED[7]"LOC=AF24 | IOSTANDARD=LVCMOS33;

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

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

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

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

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

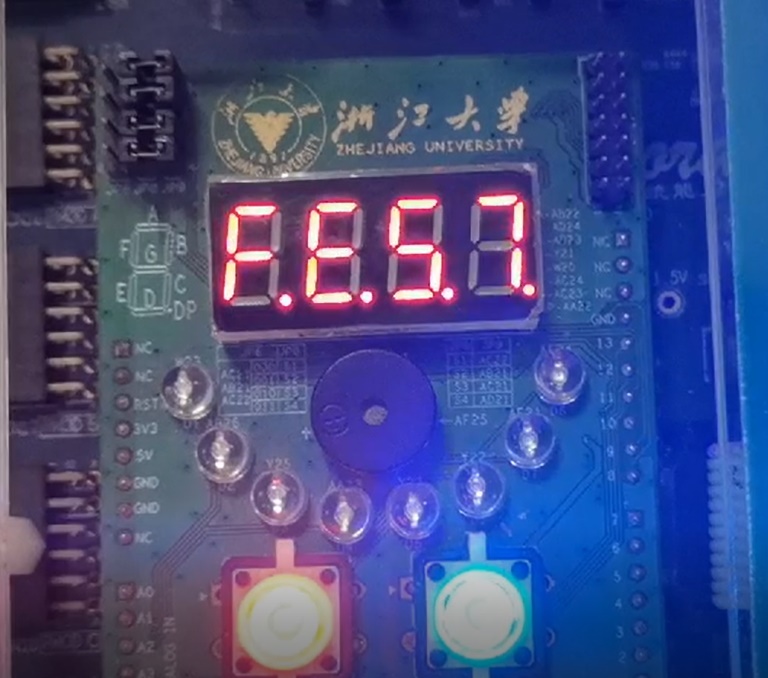
NET"seg\_clrn"LOC=M20 | IOSTANDARD=LVCMOS33;

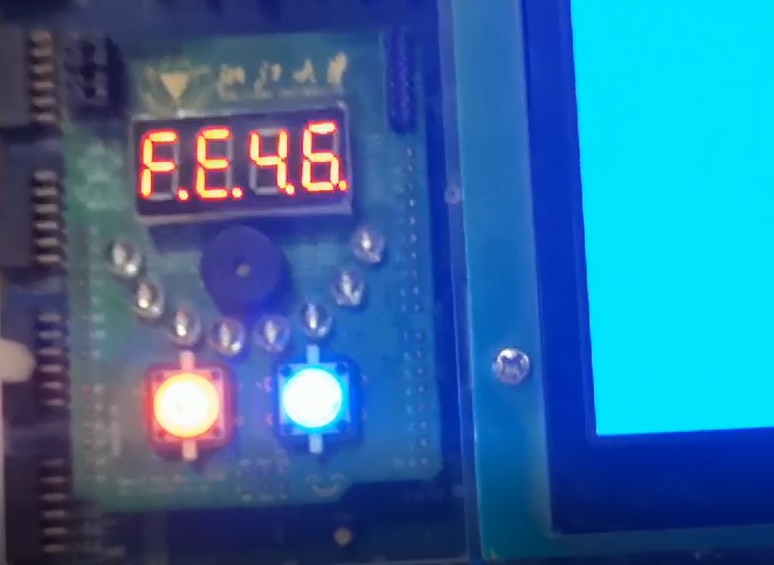
NET"seg\_sout"LOC=L24 | IOSTANDARD=LVCMOS33;

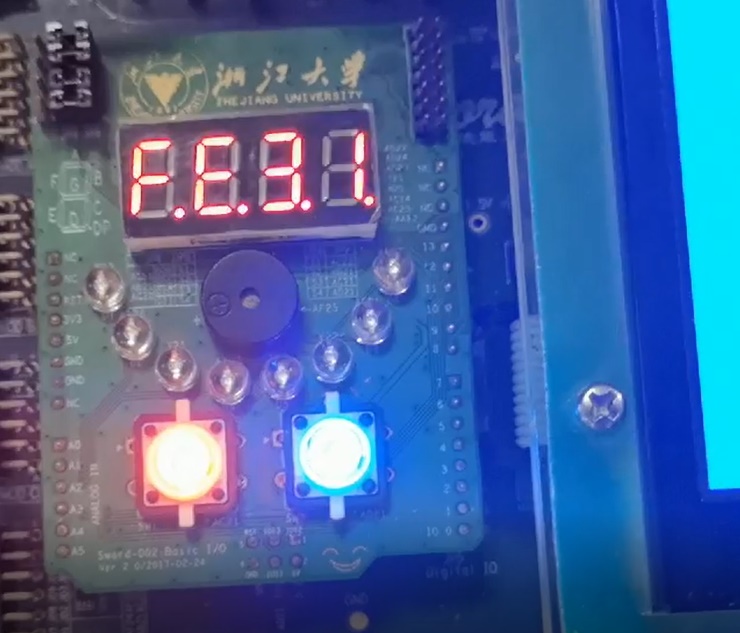
NET"SEG\_PEN"LOC=R18 | IOSTANDARD=LVCMOS33;

**二．实验结果**

反向计时器上板结果：

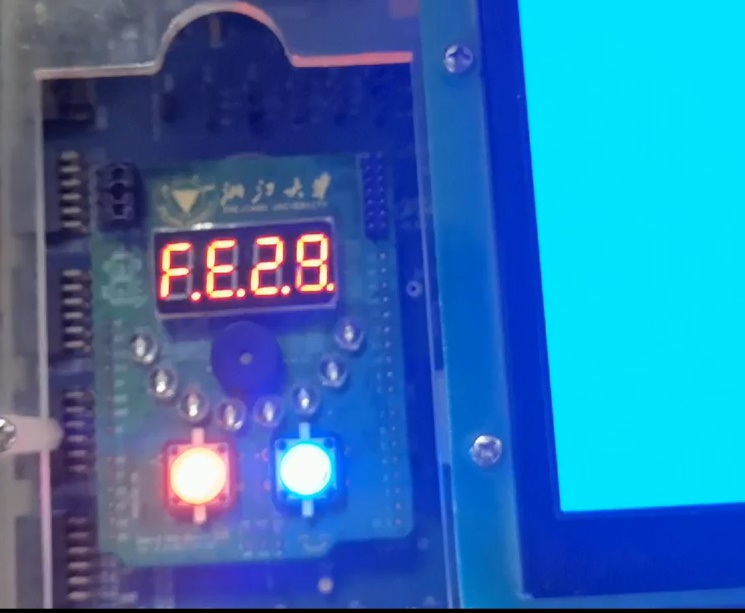






可以看到，在逆向计数时，我们的计时器数字逐渐减小。





当拨动下方按钮后，计时器数字逐渐增大，符合实验要求。

**三．实验心得**

本次实验中主要设计的是正向和逆向的计数器。对于这样的器件，一定要按照逻辑课上的知识，明白它的状态转换原理，这样才能在设计的时候知道自己每一个步骤的目的是什么。我觉得到了这个实验开始，ppt给予我们的内容越来越少，所以既需要我们自己知道逻辑图的原理，又需要掌握编程语言。

而在这次实验之前，我刚好复习了相关的课堂内容，因此感觉在编写这个实验的进程里，不像之前那样感觉吃力，反而有一种融会贯通的解题感。

**个人照片**

