浙江大学

本科实验报告

课程名称：	计算机逻辑设计基础
姓　　名：	王晓宇
学　　院：	竺可桢学院
专　　业：	计算机
邮　　箱：	1657946908@qq.com
QQ　　号：	1657946908
电　　话：	19550222634
指导教师：	洪奇军
报告日期：	2023 年 11 月 16 日

浙江大学实验报告

课程名称：计算机逻辑设计基础 实验类型：综合

实验项目名称：锁存器与触发器基本原理

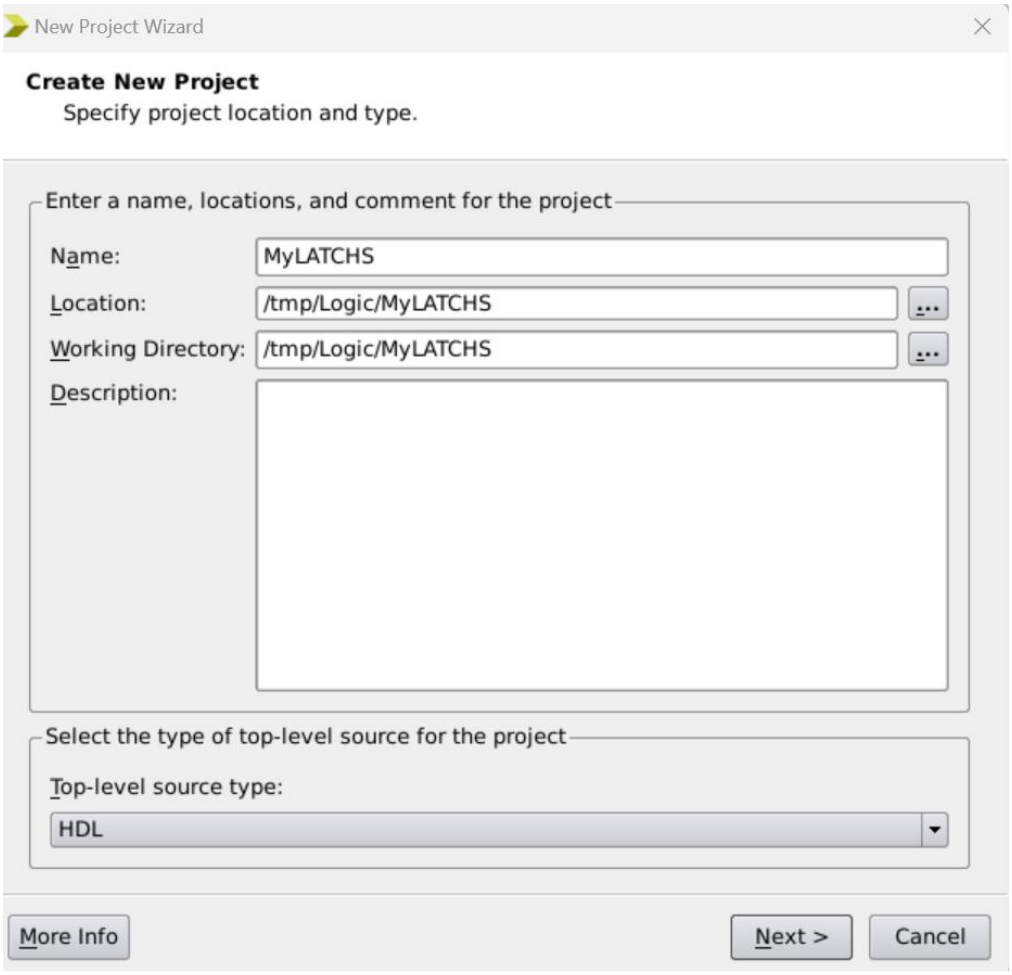
学生姓名：王晓宇 学号：3220104364 同组学生姓名：

实验地点：紫金港东四 509 室 实验日期：2023 年 11 月 16 日

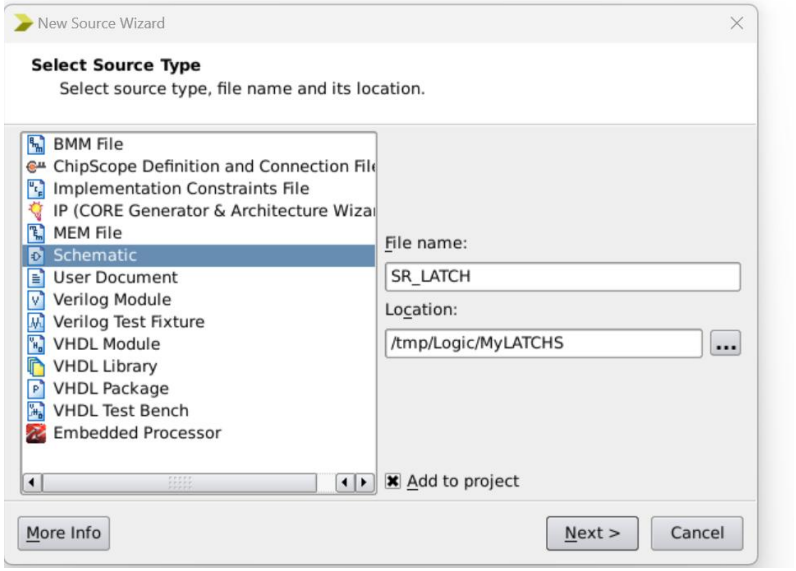
一、操作方法与实验步骤

1. 实现基本 SR 锁存器，验证功能和存在的时序问题

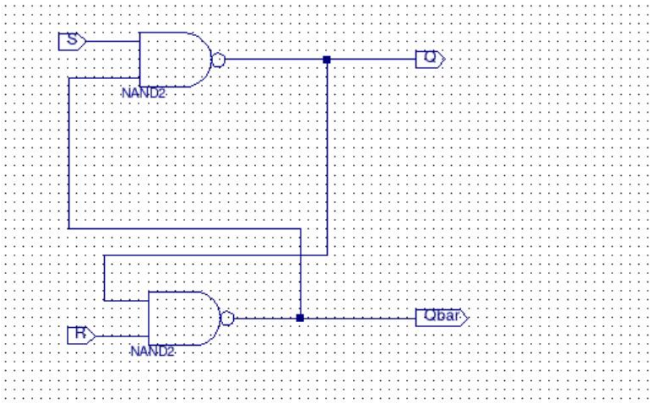
➢新建工程 MyLATCHS



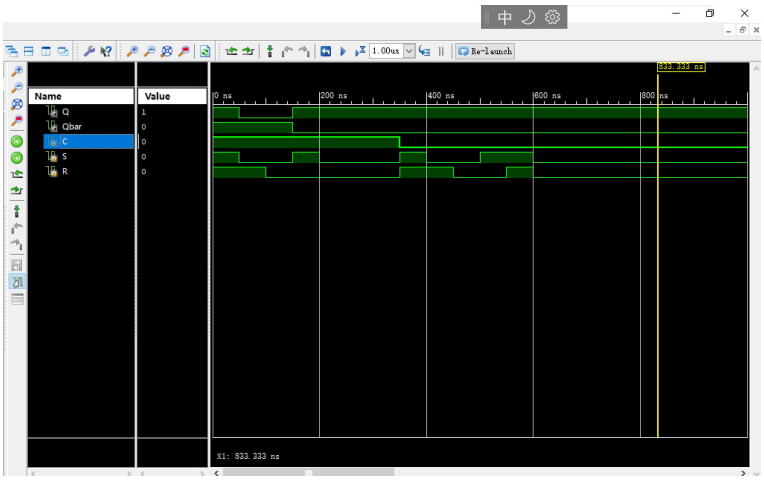
➢新建源文件 SR_LATCH.sch



➢用原理图方式设计



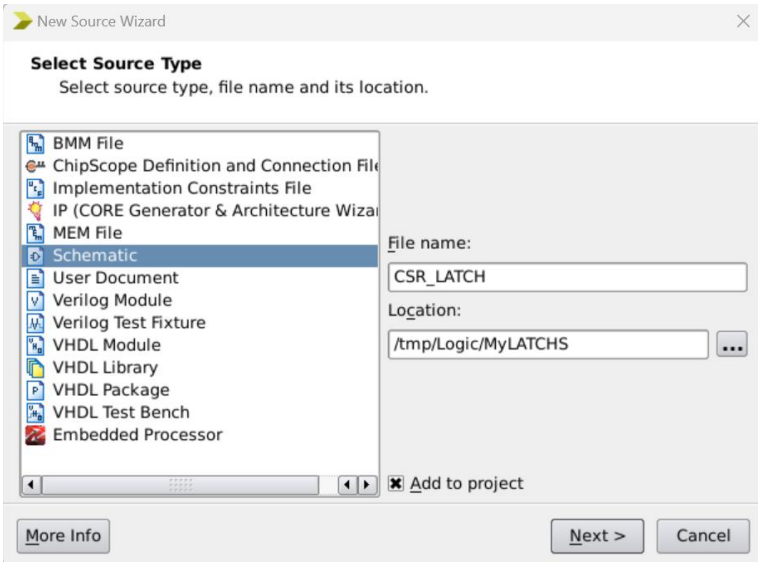
➢仿真



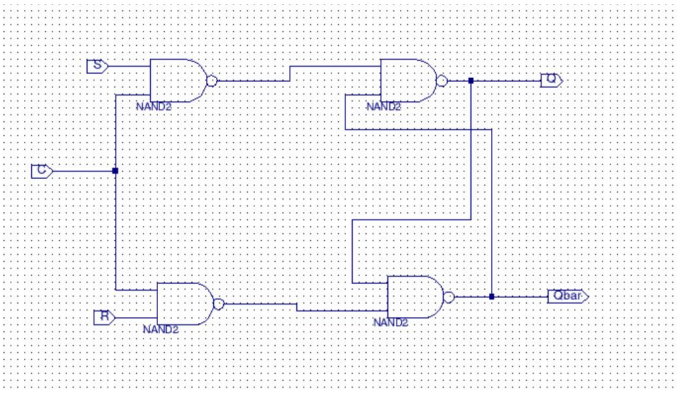
```
// Verilog test fixture created from sche

`timescale 1ns / 1ps

module SR_LATCH_SR_LATCH_sch_tb();

// Inputs
   reg S;
   reg R;

// Output
   wire Q;
   wire Qbar;

// Bidirs

// Instantiate the UUT
   SR_LATCH UUT (
      .S(S),
      .R(R),
      .Q(Q),
      .Qbar(Qbar)
   );
// Initialize Inputs
   //`ifdef auto_init
       initial begin
      R = 1;S = 1;#50;
      R = 1;S = 0;#50;
      R = 1;S = 1;#50;
      R = 0;S = 1;#50;
      R = 1;S = 1;#50;
      R = 0;S = 0;#50;
      R = 1;S = 1;#50;
      end
   //`endif
endmodule

```

2. 实现门控 SR 锁存器，并验证功能和存在的时序问题

➢新建源文件 CSR_LATCH.sch



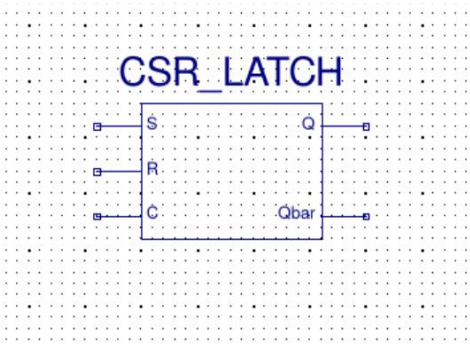
➢用原理图方式设计，用 NAND2 实现



➢仿真（包含空翻）

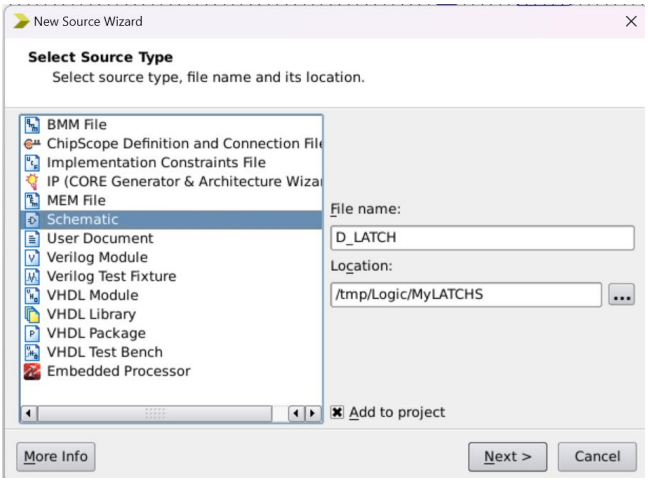
```
module CSR_LATCH_CSR_LATCH_sch_tb();

// Inputs
   reg S;
   reg R;
   reg C;

// Output
   wire Q;
   wire Qbar;

// Bidirs

// Instantiate the UUT
   CSR_LATCH UUT (
       .S(S),
       .R(R),
       .C(C),
       .Q(Q),
       .Qbar(Qbar)
   );
// Initialize Inputs
   //`ifdef auto_init
        initial begin
       S = 0;
       R = 0;
       C = 0;
       C=1;R=1;S=1;#50;
       R=1;S=0;#50;
       R=0;S=0;#50;
       R=0;S=1;#50;
       R=0;S=0;#50;
       C=0;R=1;S=1;#50;
       R=1;S=0;#50;
       R=0;S=0;#50;
       R=1;S=1;#50;
       R=0;S=1;#50;
       end
   //`endif
endmodule
```

➢生成自定义符号的 CSR_LATCH.sym

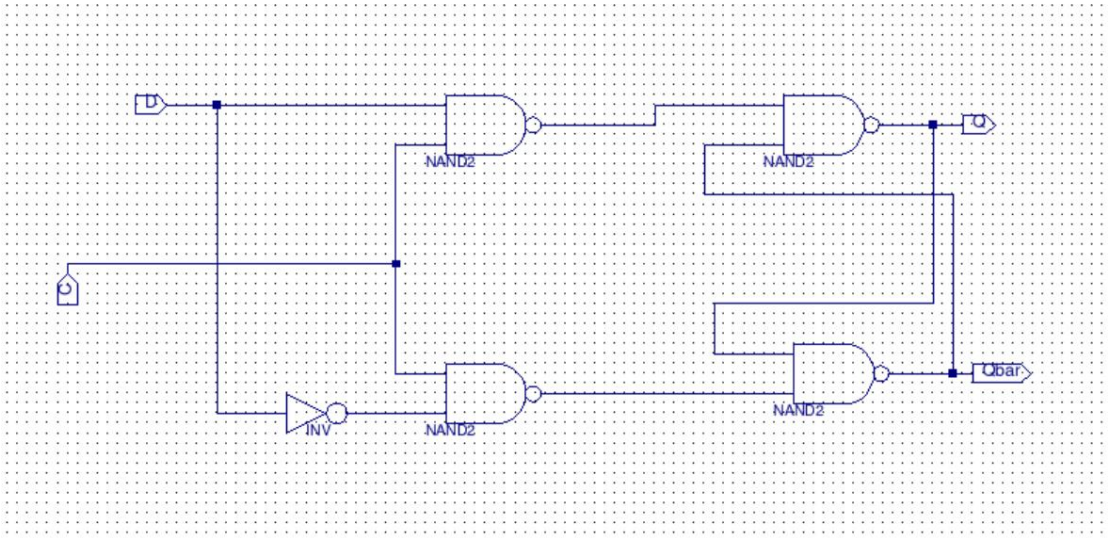


3.实现 D 锁存器，并验证功能和存在的时序问题

➢新建源文件 D_LATCH.sch

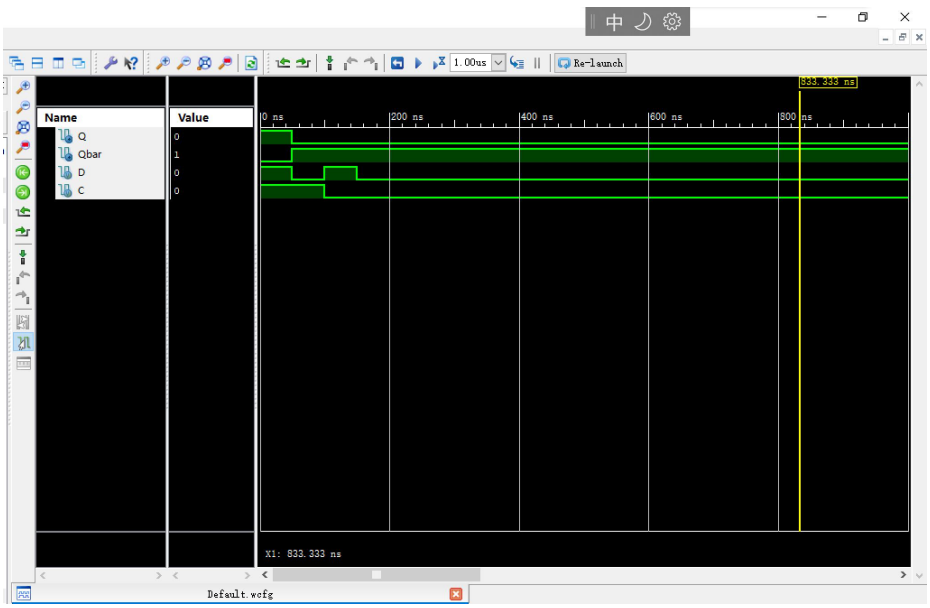


➢用原理图方式设计，用 NAND2 实现



➢仿真（包含空翻）

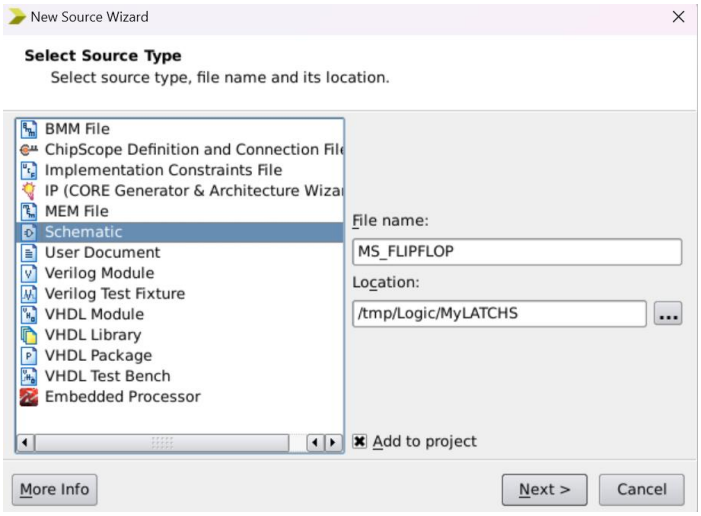
```
3   `timescale 1ns / 1ps
4
5   module D_LATCH_D_LATCH_sch_tb();
6
7   // Inputs
8      reg C;
9      reg D;
10
11  // Output
12     wire Q;
13     wire Qbar;
14
15  // Bidirs
16
17  // Instantiate the UUT
18     D_LATCH UUT (
19        .C(C),
20        .Q(Q),
21        .Qbar(Qbar),
22        .D(D)
23     );
24  // Initialize Inputs
25     //`ifdef auto_init
26         initial begin
27        D = 1;#20;
28        D = 0;#20;
29        D = 1;#20;
30        D = 0;#20;
31        D = 1;#20;
32        D = 0;#20;
33        D = 1;#20;
34        D = 0;#20;
35        D = 1;#20;
36        D = 0;#20;
37        end
38        always begin
39        C = 1;#100;
40        C = 0;#100;
41        end
42     //`endif
43  endmodule
```



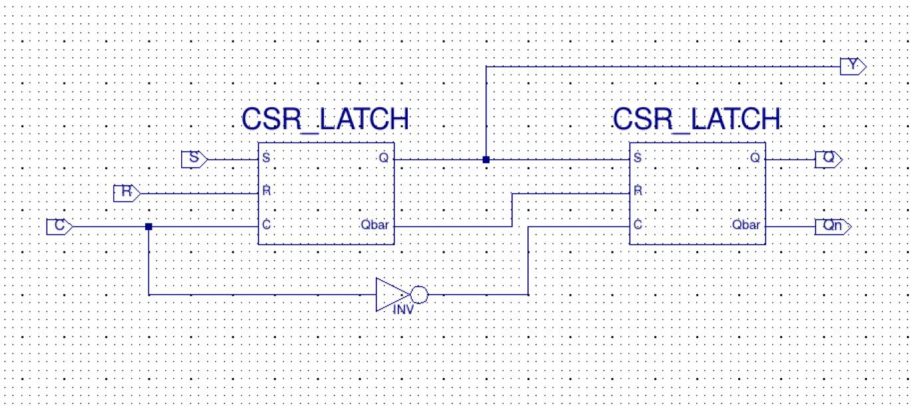
可以看到：C＝1 时，Q 随着 D 的变化而变化（Q 为 1 则 D 为 1，反之亦然， 此时不存在未定义的状态）；C = 0 时，Q 的值保持不变。因此仿真结果符合预期.

4. 实现 SR 主从触发器，并验证功能和存在的时序问题

➢新建源文件 MS_FLIPFLOP. sch

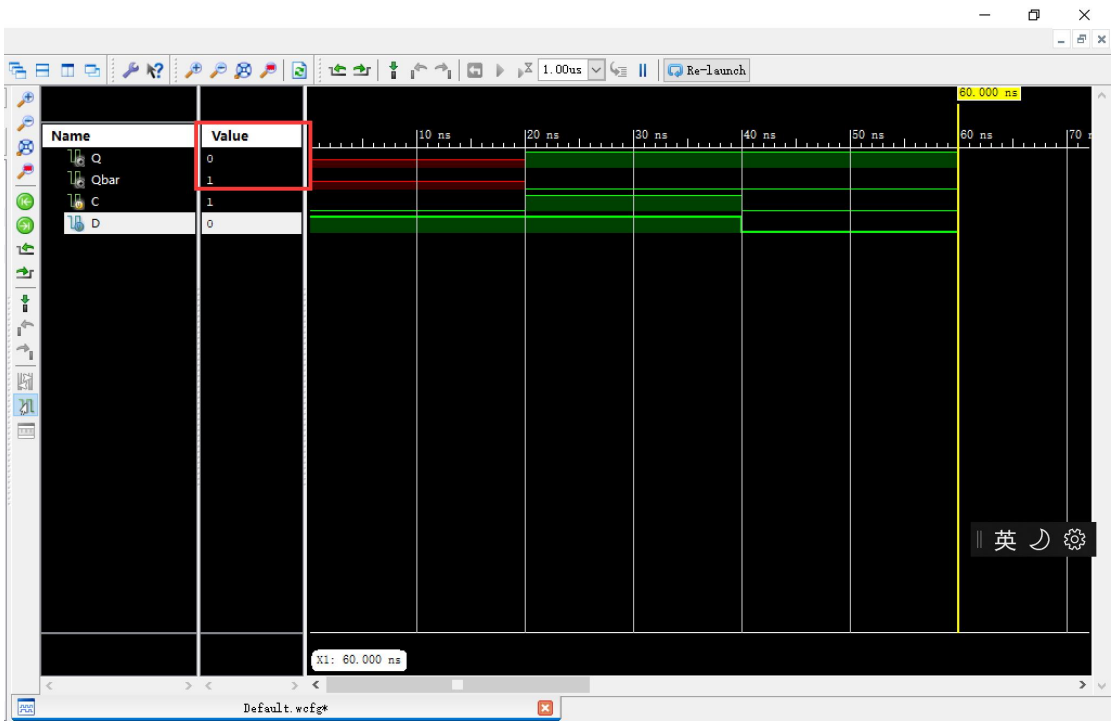


用原理图方式设计,调用 CSR_LATCH 实现



➢仿真（包含一次性采样）

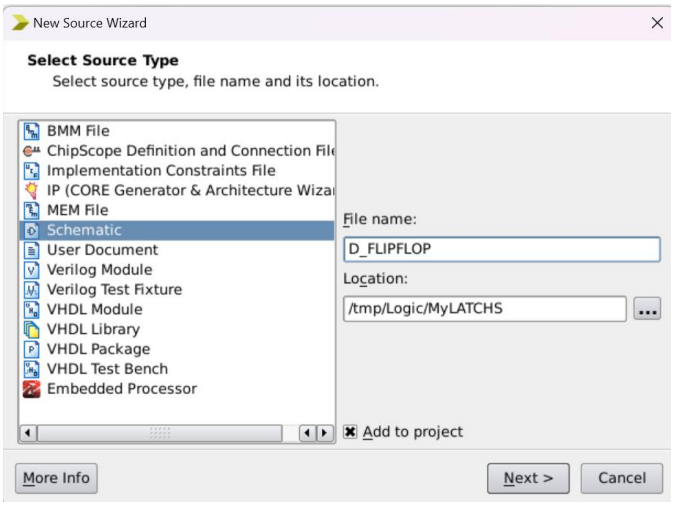
```
// Inputs
   reg R;
   reg C;
   reg S;

// Output
   wire Q;
   wire Qn;
   wire Y;

// Instantiate the UUT
   MS_FLIPFLOP UUT (
      .R(R),
      .C(C),
      .S(S),
      .Q(Q),
      .Qn(Qn),
      .Y(Y)
   );
// Initialize Inputs
   //`ifdef auto_init
       always begin
       C = 1;#50;
       C = 0;#50;
       end
       initial begin
      S = 0;R = 0;#25; S = 1;R = 0;#25;
      S = 0;R = 1;#100; S = 1;R = 0;#100;
      S = 0;R = 0;#100; S = 0;R = 1;#100;
      S = 1;R = 1;#100; S = 0;R = 0;#100;
      S = 1;R = 0;#10;
      S = 0;#5;  R=1;#10;
      S = 0;R = 0;#25;    R = 1;#5
      R = 0;#5;  S = 0;R = 0;#40;
      S = 1;#10;    S=0;#10;
      R = 1;#10;    R = 0;#10;
      end
   //`endif
endmodule
```



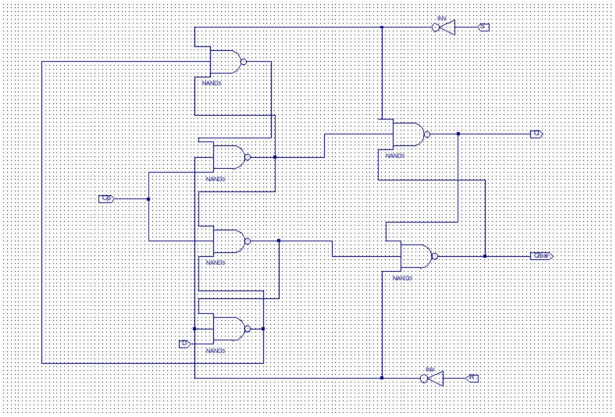
波形图在 20-60ns 电路就陷入了震荡状态

5. 实现 D 触发器，并验证功能

➢新建源文件 D_FLIPFLOP. sch

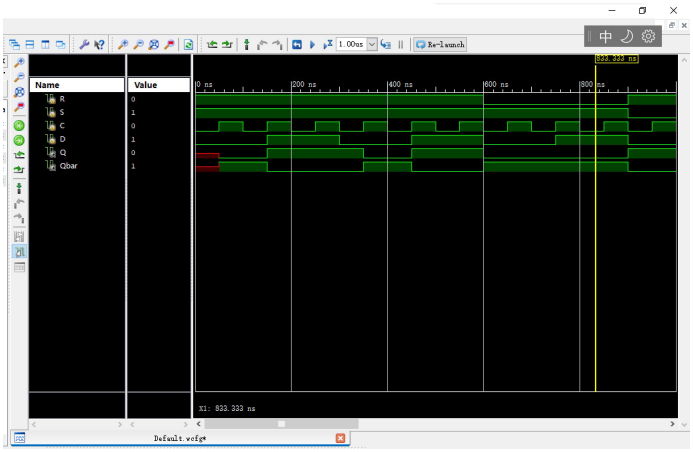


➢用原理图方式设计，调用 NAND3 实现



➢仿真

```
    ;;
// Initialize Inputs
    //`ifdef auto_init
        initial begin
        D = 0;#30;
        D = 1;#30;
        end
        always begin
        Cp = 1;#150;
        Cp = 0;#150;
        end
        always begin
        S = 1;R = 1;#50;
        S = 0;R = 0;#50;
        S = 1;R = 0;#50;
        S = 0;R = 0;#50;
        end
    //`endif
endmodule
```



6. 顶层模块设计（包含引脚）

➢Top 模块设计

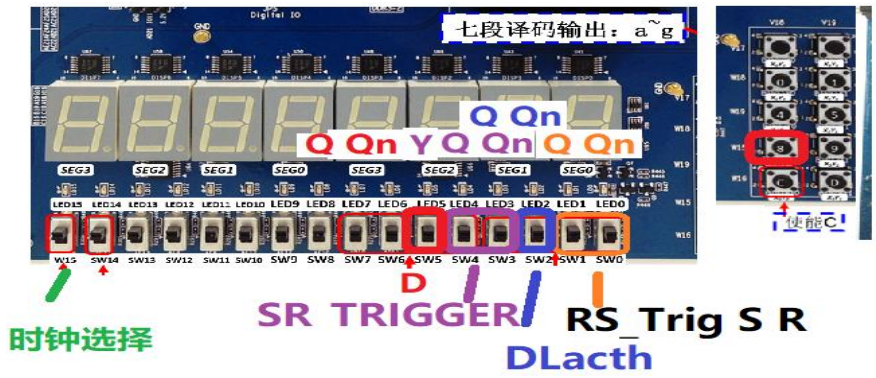
```
module TOP(
   input clk,
   input [15:0] SW,
   input [3:0] BTN,
   output [7:0] LED,
   output ledclk,
   output ledsout,
   output ledclrn,
   output LEDEN,
   output BTNX3
    );
   wire [31:0] div;
   wire [3:0] BTN_OUT;
   wire CK;
   wire [15:0] num;
   wire [8:0] NLED;
   wire LED8;
   assign BTNX3 = 0;
   assign NLED={~LED8,~LED};
   pbdebounce p1(div[17],BTN[0],BTN_OUT[0]);
   pbdebounce p2(div[17],BTN[1],BTN_OUT[1]);
   pbdebounce p3(div[17],BTN[2],BTN_OUT[2]);
   pbdebounce p4(div[17],BTN[3],BTN_OUT[3]);
   clkdiv_pulse m0(.clk(clk),.rst(1'b0),.Sel_CLK(SW[15]),.pulse(BTN_OUT[0]),.CK(CK),.clkdiv(div));
   CSR_LATCH Mm2(.C(CK),.R(SW[0]),.S(SW[1]),.Q(LED[1]),.Qbar(LED[0]));
   D_LATCH Mm3(.C(CK),.D(SW[2]),.Q(LED[3]),.Qbar(LED[2]));
   MS_FLIPFLOP Mm4(.C(CK),.R(SW[3]),.S(SW[4]),.Y(LED[6]),.Q(LED[5]),.Qn(LED[4]));
   D_FLIPFLOP Mm5(.Cp(CK),.D(SW[5]),.Q(LED8),.Qbar(LED[7]),.R(SW[6]),.S(SW[7]));
   LEDP2S #(.DATA_BITS(16),.DATA_COUNT_BITS(4),.DIR(0))
      U7(.clk(clk),.rst(1'b0),.Start(div[20]),.PData({7'h3F,NLED}),.sclk(ledclk),.sclrn(ledclrn),.sout(ledsout),.EN(LEDEN));

endmodule
```

➢Ucf 引脚设计

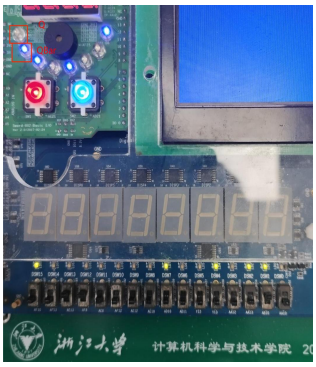
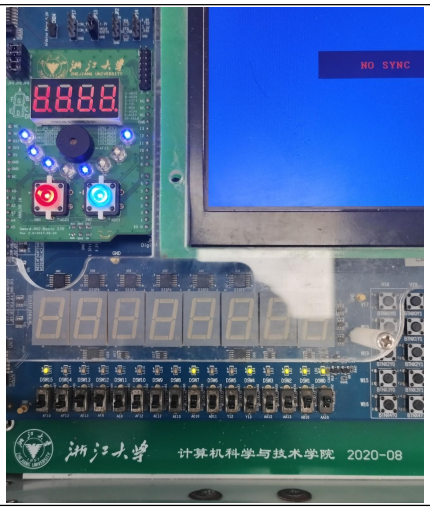
```
#clk, [15:0] SW, BTN[3:0], [7:0] LED,ledclk,ledsout,ledclrn
NET "ledclk"     LOC = N26 | IOSTANDARD = LVCMOS33;
NET "ledclrn"    LOC = N24 | IOSTANDARD = LVCMOS33;
NET "ledsout"    LOC = M26 | IOSTANDARD = LVCMOS33;
NET "LEDEN"     LOC = P18 | IOSTANDARD = LVCMOS33;

NET "clk"  LOC = AC18 | IOSTANDARD = LVCMOS18;
NET "clk"  TNM_NET = TM_CLK;
TIMESPEC TS_CLK_100M = PERIOD "TM_CLK" 10ns HIGH 50%;
#NET "RSTN" LOC =W13 | IOSTANDARD =  LVCMOS18;


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;
```

二、实验结果与分析

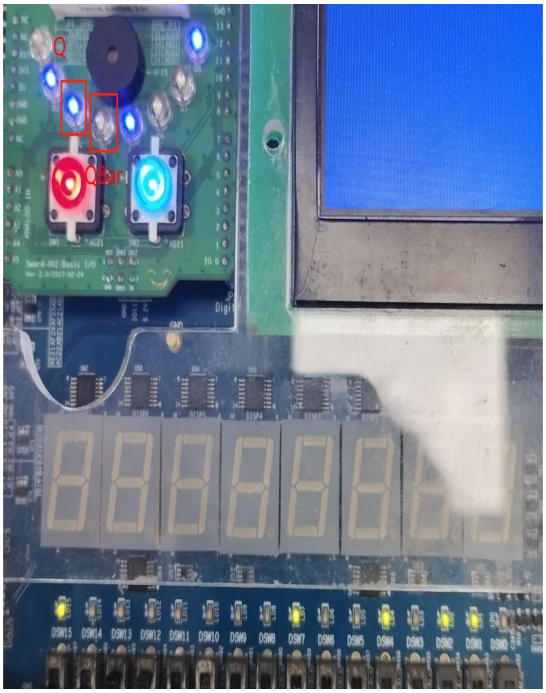
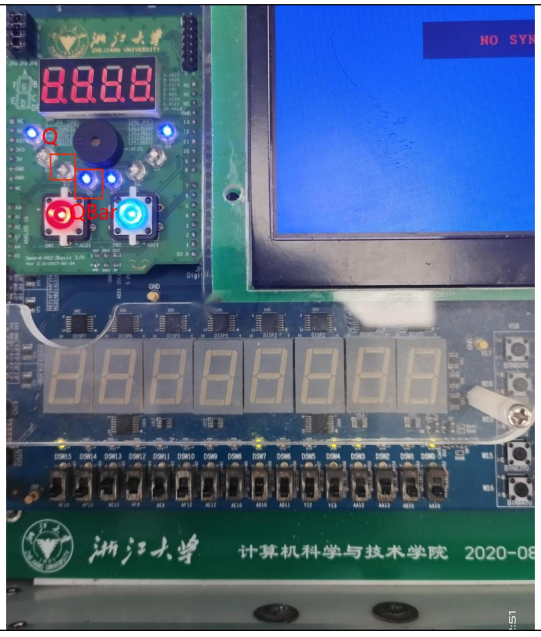
下图是各个信号输出的位置说明



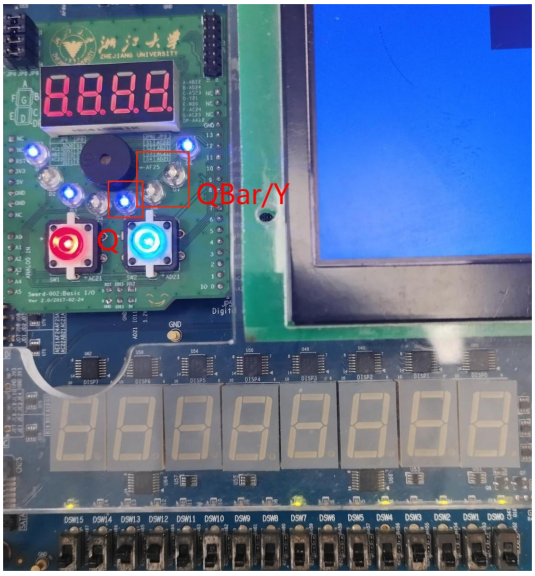
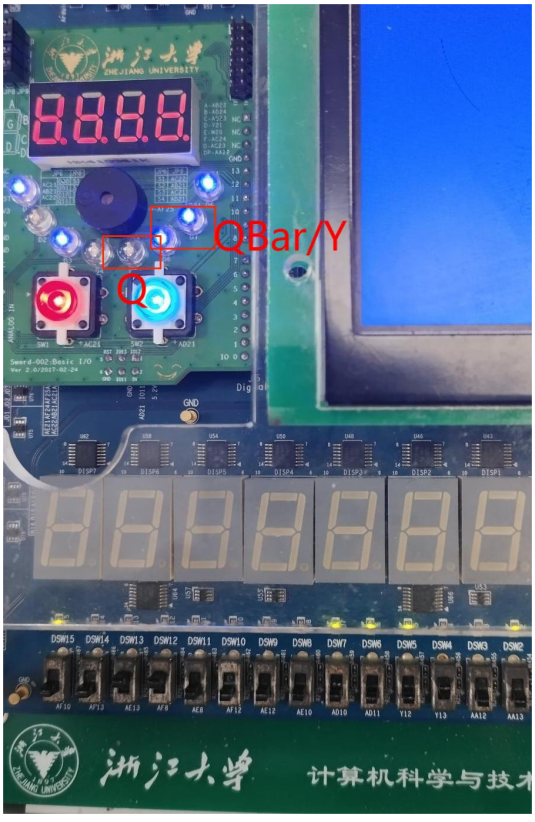
➢SR 锁存器

Q/Qbar	实验图片
1/0	
0/1	
空翻	

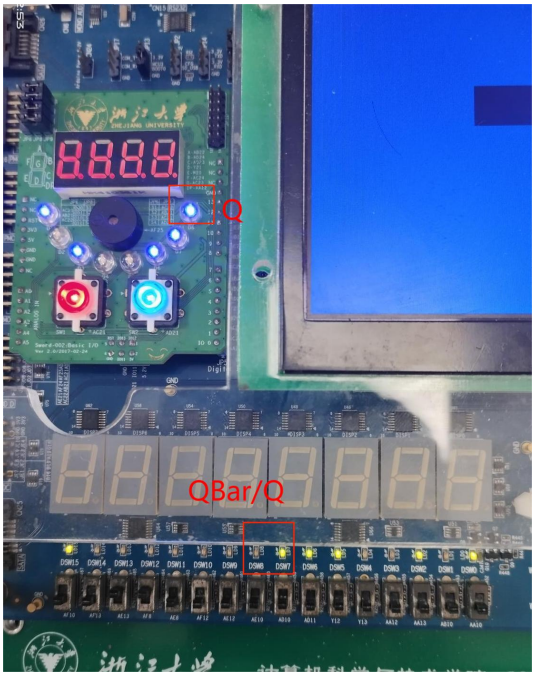
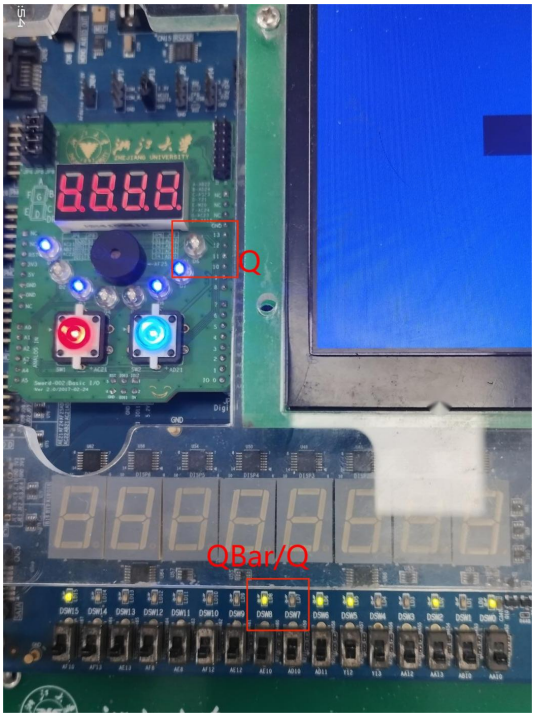
➢D 锁存器

Q/Qbar	实验图片
1/0	
0/1	

➢SR 主从触发器

Q/Qbar/Y	实验图片
1/0/0	
0/1/1	

➢D 触发器

Q/Qbar	实验图片
1/0	
0/1	

三、讨论、心得

这次的实验是研究锁存器和触发器的一次实验，SR 锁存器存在的 11 状态未定义可以通过 D 触发器解决，但是触发器存在的空翻现象仍存在。而触发器便能很好地处理空翻现象，由此改进出 SR 主从触发器、D 触发器，不可避免地 SR 触发器存在一次性采样的问题，At the end:D 触发器很好的性能使得它成为电路设计很重要的一环。

最开始设计完各类锁存器和触发器之后，发现输出信号有 9 个，而面板上的灯数只有 8 个，而老师提供了 P2S 的解决思路，也就是开关上面的那一栏小灯，默认为不亮，数量达到了 16 个，这个可以作为之后信号输出的一个实用的思路。另外设计了按钮来控制触发信号的手动产生，但是在按钮面板的触发规则是要求 xy 双信号同时输入才会产生信号，像是两个信号形成一个向量来输出给 sword 实验板，所以在 k7.ucf 引脚和 top.v 顶层模块中映入 BTNX3 变量来默认一排的 x 输出，所以在实际操作中只需要去敲击 y 轴上面的一个信号即可。