浙江大学

本科实验报告

课程名称：	数字逻辑电路设计
姓　　名：	
学　　院：	计算机科学与技术学院
专　　业：	计算机科学与技术
邮　　箱：	
QQ　号：	
电　　话：	
指导教师：	洪奇军
报告日期：	2020 年 11 月 24 日

浙江大学实验报告

课程名称：数字逻辑设计 实验类型：综合

实验项目名称：实验九——锁存器与触发器

学生姓名： 学号： 同组学生姓名：

实验地点：紫金港东四 509 室 实验日期：2020 年 11 月 24 日

一、实验目的

1.1 掌握锁存器与触发器构成的条件和工作原理
1.2 掌握锁存器与触发器的区别
1.3 了解静态存储器 SRAM 存储单元结构
1.4 掌握基本 RS 锁存器的基本功能与使用要点
1.5 掌握 RS、D 触发器的基本功能及基本应用
1.6 掌握集成触发器的使用和异步清零的作用
1.7 了解用 D 触发器实现分频电路
1.8 了解用 D 触发器构成单稳态电路(开关去抖动)

二、实验内容和原理

2.1 实验内容

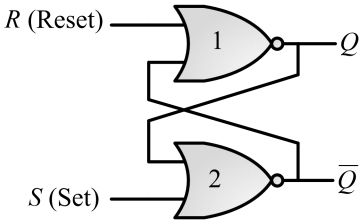
- 用原理图实现 RS 锁存器并仿真验证
- 实现门控 RS 锁存器、D 锁存器并仿真验证
- 用 RS 锁存器实现 RS 主从触发器并仿真验证
- 用 D 锁存器和用 RS 锁存器实现主从 D 触发器并仿真验证
- 用原理图实现维持阻塞型 D 触发器
- 触发器物理测试

2.2 实验原理

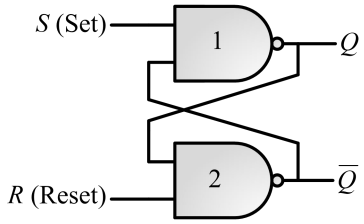
2.2.1 锁存器

锁存器必须满足三个基本条件：能长期保持给定的某个稳定状态；有两个稳定状态，“0”和“1”；在一定条件下能随时改变状态。而最基本的锁存器有 R-S 锁存器和 D 锁存器。

将两个具有 2 输入端的反向逻辑器件的输出与输入端交叉连起来，另一个输入端作为外部信息输出端，就构成最简单的 SR 锁存器。

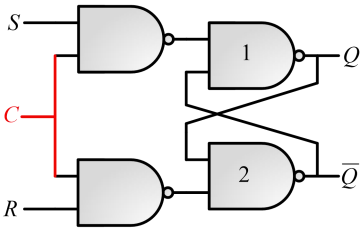


图表 2.2.1-1 NOR 结构锁存器示意图

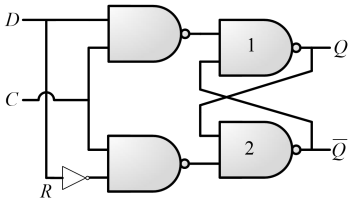


图表 2.2.1-2 NAND 结构锁存器示意图

其中，加入一个控制信号 C 我们就可以得到门控锁存器。为了消除 R-S 锁存器的不确定状态，我们可以通过一个反相器来确保两个输入信号不会同时 1，如此得到如图表 2.2.1-4 所示的门控 D 锁存器。



图表 2.2.1-3 门控 R-S 锁存器示意图

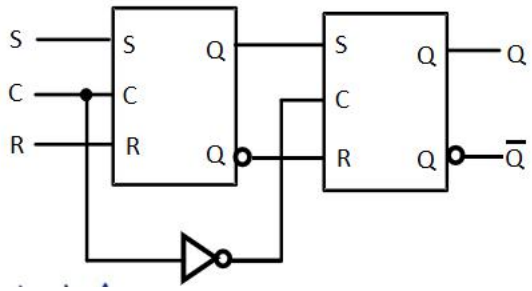


图表 2.2.1-4 门控 D 锁存器示意图

2.2.2 主从 RS 触发器与主从 D 触发器

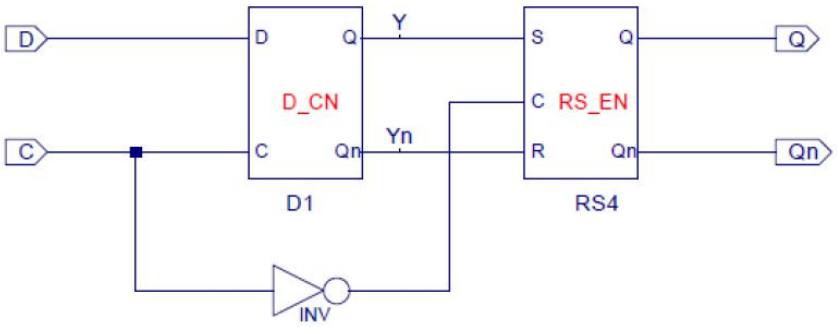
锁存器在实践中使用较少，原因是其存在的“空翻”或“毛刺”问题：锁存器的输出端可以直接反应出输入端的数据，导致在时序电路存储状态时出现“空翻”无法控制。

要想解决这个问题，需要切断直接通路。使用两个锁存器并增加一个使能信号，使能信号变化时，输出端同步变化一次。原理图如图表 2.2.2-1 所示。



图表 2.2.2-1 主从 RS 触发器示意图

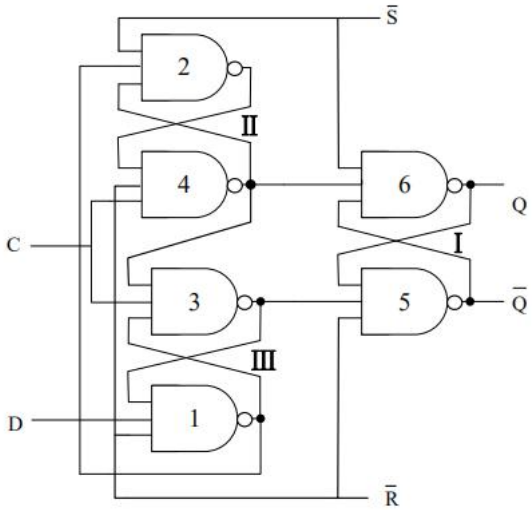
考虑到主从 RS 触发器存在的一次性采样问题与不确定态，将第一个 R-S 锁存器替换为 D 锁存器可得主从 D 触发器。原理图如图表 2.2.2-2 所示。



图表 2.2.2-2 主从 D 触发器示意图

2.2.3 边沿触发器

为了避免前述触发器存在的问题，可以设计维持阻塞型边沿触发器，其只有在使能信号的正边沿改变输出信号。原理图如图表 2.2.3-1 所示。



图表 2.2.3-1 维持阻塞型边沿触发器示意图

三、操作方法与实验步骤

3.1 实验设备与材料

1. 装有 ISE 14.7 的计算机　　1 台
2. SWORD 开发板　　1 套

3.2 实验步骤

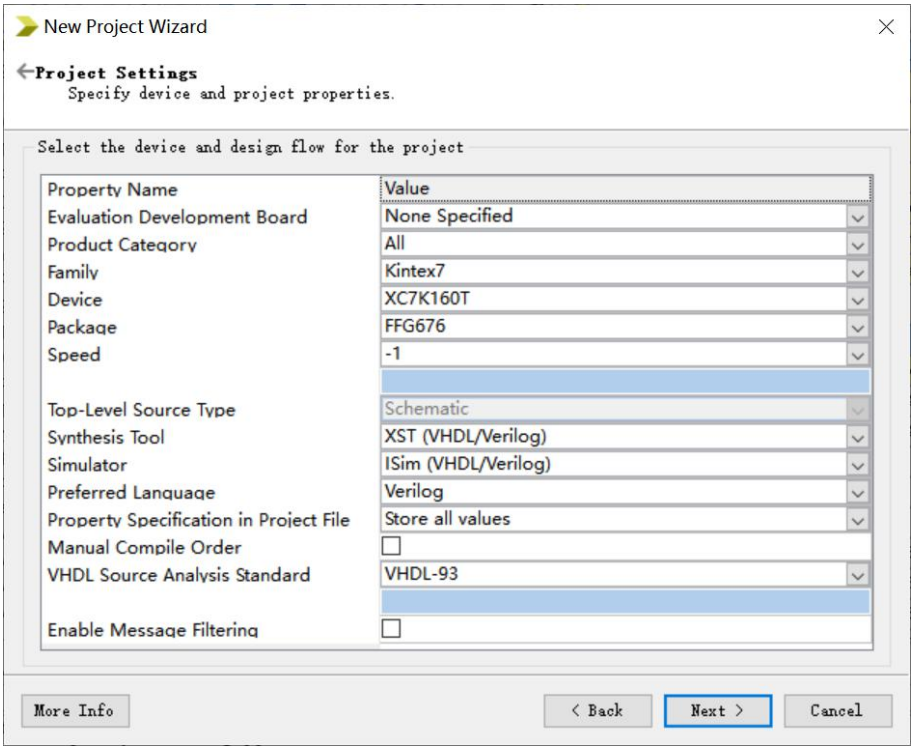
3.2.1 用原理图实现 RS 锁存器并仿真验证

1. 建立工程

打开 ISE Design Suite 14.7，在左上角点击 File，再点击 New Project 命名为 Locker。

将工程命名，选择 Top-level source type 为 Schematic 并选择项目保存位置。

点击 Next 后，出现如图表 3.2.1-1 所示的对话框，如图表所示选择对应的 Family、Device、Package、Speed 等属性，确认无误后一直点击 Next 直至创建工程结束。



图表 3.2.1-1 项目配置

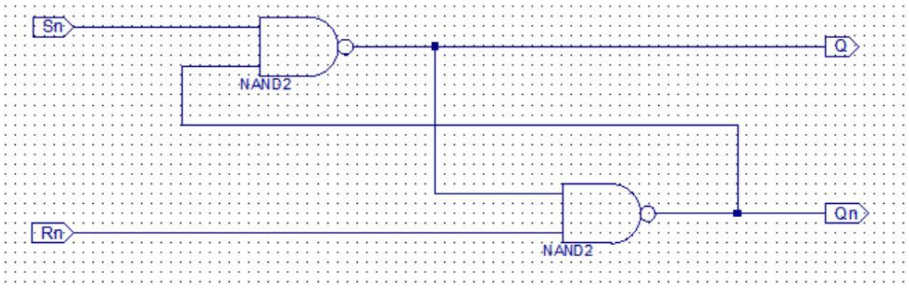
2. 创建 Schematic 源文件

在左侧 Sources 窗口空白处右键菜单中选择 New Source。

在对话框中，选择源类型为 Schematic，输入文件名 RS_NAND，并且勾选下方 Add to Project。此后，一直点击 Next 直至结束。

此后，双击打开左侧 Sources 窗口中的 add.sch 文件，在文件内画出原理图，如图表 3.2.1-2

所示。



图表 3.2.1-2 RS_NAND

绘画完成后，在 Source 窗口单击选中 RS_NAND.sch 文件，然后双击下方 Process 窗口的 Design Utilities 下的 Check Design Rules 检查连线规则。连线无误后，双击 View HDL Functional Model 查看原理图的 HDL 描述，生成.vf 文件 。

3. 建立逻辑符号

在 Sources 窗口中单击选中 add.sch 文件，然后在 Process 窗口双击运行 Design Utilities 下的 Create Schematic Symbol，生成后缀为.sym 的逻辑符号图文件，该文件保存在工程目录下。

4. 仿真运行

在 Source 窗口右键新建源类型为 Verilog Test Fixture 的文件，勾选 Add to Project，命名为 RS_NAND_test。在点击 Next 之后的关联资源中选择 RS_NAND.sch，一直点击 Next 直至结束。在文件中输入仿真激励代码，并确认仿真结果无误。仿真激励代码如下：

```
`timescale 1ns / 1ps

module RS_NAND_RS_NAND_sch_tb();

// Inputs
   reg Sn;
   reg Rn;

// Output
   wire Q;
   wire Qn;

// Bidirs

// Instantiate the UUT
   RS_NAND UUT (
        .Sn(Sn),
        .Rn(Rn),
        .Q(Q),
        .Qn(Qn)
   );
// Initialize Inputs
   initial begin
    Rn = 1;
    Sn = 0;
    #50;
    Sn = 0;
    Rn = 1;
    #50;
    Sn = 1;
    Rn = 1;
```

```
    #50;
    Sn = 1;
    Rn = 0;
    #50;
    Sn = 1;
    Rn = 1;
    #50;
    Sn = 0;
    Rn = 0;
    #50;
    Sn = 1;
    Rn = 1;
    #50;
    Sn = 0;
    Rn = 0;
    #50;
    Sn = 0;
    Rn = 1;
    #50;
    Sn = 0;
    Rn = 0;
    #50;
    Sn = 1;
    Rn = 0;
    #50;
    Sn = 1;
    Rn = 1;
    end

endmodule
```

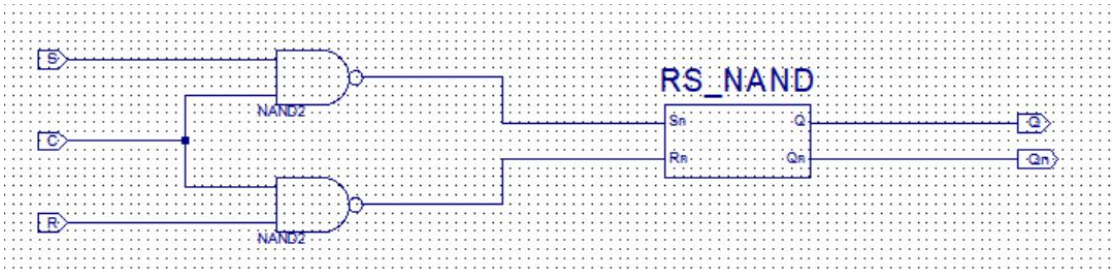
3.2.2 实现门控 RS 锁存器、D 锁存器并仿真验证

1. 门控 RS 锁存器

在 Locker 工程下，左侧 Sources 窗口空白处右键菜单中选择 New Source。

在对话框中，选择源类型为 Schematic，输入文件名 RS_EN，并且勾选下方 Add to Project。此后，一直点击 Next 直至结束。

此后，双击打开左侧 Sources 窗口中的 add.sch 文件，在文件内画出原理图，调用 RS_NAND 逻辑符号，如图表 3.2.2-1 所示。



图表 3.2.2-1 RS_EN

绘画完成后，除仿真代码外，其余步骤与 3.2.1 相同。仿真代码如下：

```
`timescale 1ns / 1ps

module RS_EN_RS_EN_sch_tb();

// Inputs
   reg C;
```

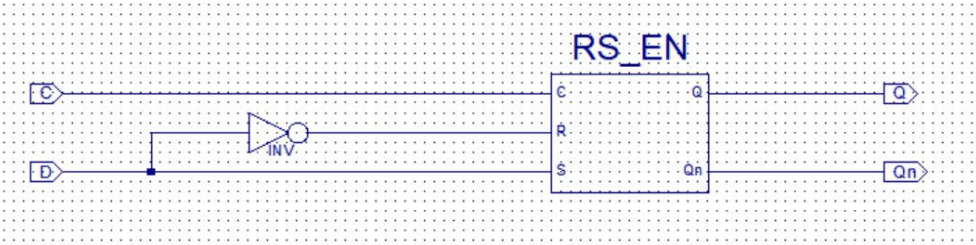
```
   reg R;
   reg S;

// Output
   wire Q;
   wire Qn;

// Bidirs

// Instantiate the UUT
   RS_EN UUT (
        .Q(Q),
        .Qn(Qn),
        .C(C),
        .R(R),
        .S(S)
   );
// Initialize Inputs
        integer i;
      initial begin
        C = 0;
        R = 0;
        S = 0;
        #40;

        S=1;
        R=0;
        #100;

        S=0;
        R=1;
        #100;

        S=1;
        R=1;
        #100;

        S=0;
        R=0;
        #100;

        S=1;
        R=0;
        #100;
        end

        always @*
            for(i=0; i<20; i=i+1)begin
            #50;
            C<=~C;
        end
endmodule
```

2. D 锁存器

在 Locker 工程下，左侧 Sources 窗口空白处右键菜单中选择 New Source。

在对话框中，选择源类型为 Schematic，输入文件名 D_EN，并且勾选下方 Add to Project。此后，一直点击 Next 直至结束。

此后，双击打开左侧 Sources 窗口中的 add.sch 文件，在文件内画出原理图，调用 RS_EN 逻辑符号，如图表 3.2.2-2 所示。



图表 3.2.2-2 D_EN

绘画完成后，除仿真代码外，其余步骤与 3.2.1 相同。仿真代码如下：

```
`timescale 1ns / 1ps

module D_EN_D_EN_sch_tb();

// Inputs
   reg D;
   reg C;

// Output
   wire Q;
   wire Qn;

// Bidirs

// Instantiate the UUT
   D_EN UUT (
        .Q(Q),
        .Qn(Qn),
        .D(D),
        .C(C)
   );
// Initialize Inputs
   integer i, j;
   initial begin
    D = 0;
    C = 0;
    #40;
    for(i=0; i<5; i=i+1)begin
        D=~D;
        #80;
    end
   end

    always@*
    for(j=0; j<10; j=j+1)begin
        #50;
        C<=~C;
    end
endmodule
```

3.2.3 用 RS 锁存器实现 RS 主从触发器并仿真验证

1. 建立工程

打开 ISE Design Suite 14.7，在左上角点击 File，再点击 New Project 命名为 Trigger。

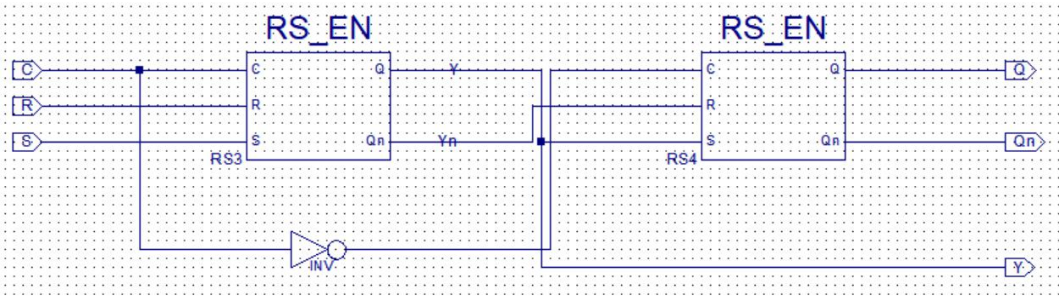
将工程命名，选择 Top-level source type 为 Schematic 并选择项目保存位置。

点击 Next 后，选择对应的 Family、Device、Package、Speed 等属性，确认无误后一直点击 Next 直至创建工程结束，同 3.2.1。

2. 创建 Schematic 源文件

在左侧 Sources 窗口空白处右键菜单中选择 New Source。

在对话框中，选择源类型为 Schematic，输入文件名 RS_Tri，并且勾选下方 Add to Project。此后，一直点击 Next 直至结束。调用 RS_EN，根据原理图画图，如图表 3.2.3-1 所示。



图表 3.2.3-1 RS_Tri

绘画完成后，除仿真代码外，其余步骤与 3.2.1 相同。仿真代码如下：

```
`timescale 1ns / 1ps

module RS_Tri_RS_Tri_sch_tb();

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

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

// Bidirs

// Instantiate the UUT
   RS_Tri UUT (
        .R(R),
        .S(S),
        .Y(Y),
        .Qn(Qn),
        .Q(Q),
        .C(C)
   );
// Initialize Inputs
   integer i;
    initial begin
    C = 0;
    S = 0;
    R = 0;
    #55;
    S = 1;
    #80;
    S = 0;
    #100;
    R = 1;
    #100;
    R = 0;
    #100;
```

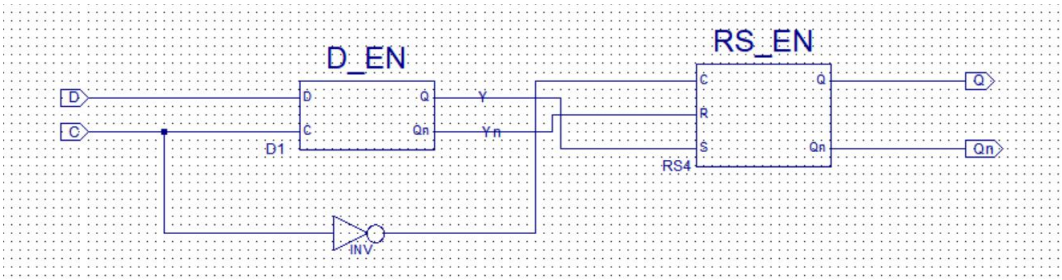
```
    S = 1;
    #20;
    S = 0;
    #5;
    R = 1;
    #20;
    R = 0;
    #55;
    S = 1;
    #20;
    S = 0;
    #120;
    R = 1;
    S = 1;
    #100;
    R = 0;
    S = 0;
    end

    always @*
        for (i=0;i<20;i=i+1) begin
            #50;
            C <= ~C;
    end
endmodule
```

3.2.4 用 D 锁存器和用 RS 锁存器实现主从 D 触发器并仿真验证

1. 创建 Schematic 源文件

在 Trigger 工程下，左侧 Sources 窗口空白处右键菜单中选择 New Source。

在对话框中，选择源类型为 Schematic，输入文件名 D_Tri，并且勾选下方 Add to Project。此后，一直点击 Next 直至结束。调用 RS_EN 与 D_EN，根据原理图画图，如图表 3.2.4-1 所示。



图表 3.2.4-1 D_Tri

绘画完成后，除仿真代码外，其余步骤与 3.2.1 相同。仿真代码如下：

```
`timescale 1ns / 1ps

module D_Tri_D_Tri_sch_tb();

// Inputs
   reg C;
   reg D;

// Output
   wire Q;
   wire Qn;

// Bidirs
```

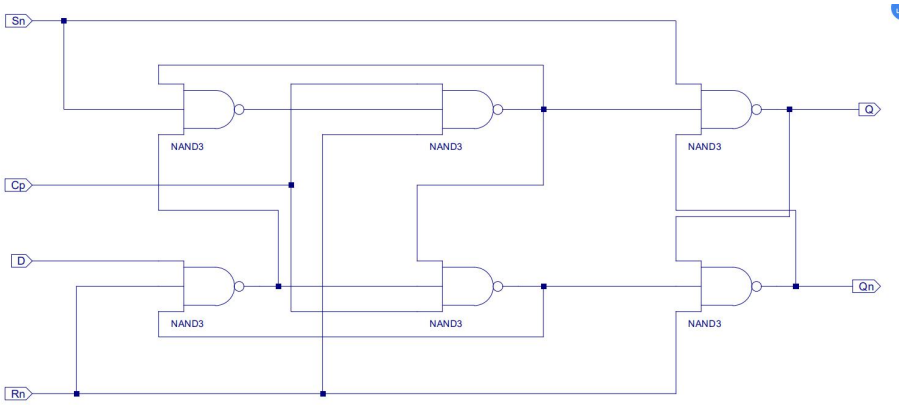
```
// Instantiate the UUT
   D_Tri UUT (
        .C(C),
        .D(D),
        .Q(Q),
        .Qn(Qn)
   );
// Initialize Inputs
   integer i;
    initial begin
    C = 0;
    D = 0;
    #115;
    D = 1;
    #20;
    D = 0;
    #220;
    D = 1;
    #20;
    D = 0;
    #70;
    D = 1;
    #20;
    D = 0;
    #130;
    D = 1;
    #20;
    D = 0;
    #80;
    end

    always@*
        for(i=0; i<20; i=i+1)begin
            #50;
            C<=~C;
        end
endmodule
```

3.2.5 用原理图实现维持阻塞型 D 触发器

1. 创建 Schematic 源文件

在 Trigger 工程下，左侧 Sources 窗口空白处右键菜单中选择 New Source。

在对话框中，选择源类型为 Schematic，输入文件名 MB_DFF，并且勾选下方 Add to Project。此后，一直点击 Next 直至结束。根据原理图画图，如图表 3.2.5-1 所示。



图表 3.2.5-1 MB_DFF

绘画完成后，除仿真代码外，其余步骤与 3.2.1 相同。仿真代码如下：

```
`timescale 1ns / 1ps

module MB_DFF_MB_DFF_sch_tb();

// Inputs
   reg D;
   reg Rn;
   reg Sn;
   reg Cp;

// Output
   wire Qn;
   wire Q;

// Bidirs

// Instantiate the UUT
   MB_DFF UUT (
        .D(D),
        .Rn(Rn),
        .Qn(Qn),
        .Q(Q),
        .Sn(Sn),
        .Cp(Cp)
   );
// Initialize Inputs
    integer i;
   initial begin
    D = 0;
    Rn = 1;
    Sn = 1;
    Cp = 0;
    #115;
    D=1;
    #20;
    D=0;
    #220;

    D=1;
    #20;
    D=0;
    #70;

    D=1;
    #20;
    D=0;
    #130;

    D=1;
    #20;
    D=0;
    #80;

    Rn=1;
    Sn=0;
    #50;
    Rn=0;
    Sn=1;
    #50;
    Rn=0;
```

```
    Sn=0;
    #50;
    Rn=1;
    Sn=1;
    #50;
   end

    always@*
        for(i=0; i<40; i=i+1)begin
            #50;
            Cp<=~Cp;
        end
endmodule
```

3.2.6 触发器物理测试

1. 建立工程

打开 ISE Design Suite 14.7，在左上角点击 File，再点击 New Project 命名为 Top_Tri。

将工程命名，选择 Top-level source type 为 HDL 并选择项目保存位置。

点击 Next 后，选择对应的 Family、Device、Package、Speed 等属性，确认无误后一直点击 Next 直至创建工程结束，同 3.2.1。

2. 设计 clkdiv

为了满足连续信号与单步信号的要求，我们需要重新设计 clkdiv 模块。代码如下：

```
`timescale 1ns / 1ps

module clkdiv(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. 设计顶层结构

添加调用前述步骤完成的三个触发器，同时添加调用老师下发的 SAnti_jitter.v，SAnti_jitter.ngc，SPLIO.v 和 SPLIO.ngc 文件。新建名为 Top_Trigger 的顶层模块 Verilog Module 文件，代码如下：

```
`timescale 1ns / 1ps

module Top_Trigger(input clk_100mhz,
                    input wire RSTN,
                    input wire [3:0]K_COL,
                    output wire [4:0]K_ROW,
                    input wire [15:0]SW,
                    output wire LEDCLK,
                    output wire LEDDT,
                    output wire LEDCLR,
```

```
                    output wire LEDEN,
                    output [7:0]LED
);

wire [31:0]clkdiv, PD;
wire [15:0]SW_OK;
wire [3:0]BTN_OK, pulse_out;
wire rst, CR, CK;

assign clk=clk_100mhz;

RS_Tri M1(.C(CK),
          .R(SW_OK[1]),
          .S(SW_OK[0]),
          .Q(PD[0]),
          .Qn(PD[1]),
          .Y(PD[2]));

D_Tri M2(.C(CK),
         .D(SW_OK[3]),
         .Q(PD[3]),
         .Qn(PD[4]));

MB_DFF M3(.Cp(CK),
          .D(SW_OK[4]),
          .Rn(SW_OK[6]),
          .Sn(SW_OK[5]),
          .Q(PD[5]),
          .Qn(PD[6]));

SAnti_jitter U8(.clk(clk),
                .RSTN(RSTN),
                .readn(),
                .Key_y(K_COL),
                .Key_x(K_ROW),
                .SW(SW),
                .Key_out(),
                .Key_ready(),
                .pulse_out(),
                .BTN_OK(BTN_OK),
                .SW_OK(SW_OK),
                .CR(),
                .rst(rst));

clkdiv U9(.clk(clk),
          .rst(rst),
          .Sel_CLK(SW[2]),
          .pulse(BTN_OK[0]),
          .clkdiv(clkdiv),
          .CK(CK));

SPLIO U7(.clk(clk),
         .rst(rst),
         .Start(clkdiv[20]),
         .EN(1'b1),
         .P_Data(PD),
         .LED(LED),
         .led_clk(LEDCLK),
         .led_sout(LEDDT),
         .led_clrn(LEDCLR),
         .LED_PEN(LEDEN),
```

```
                .GPIOf0());

endmodule
```

4. 分配引脚

引脚约束文件代码如下：

```
NET"clk_100mhz"LOC=AC18 | IOSTANDARD=LVCMOS18;
NET"clk_100mhz" TNM_NET=TM_CLK;
TIMESPEC TS_CLK_100M=PERIOD"TM_CLK" 10 ns HIGH 50%;

NET"RSTN"LOC=W13 | IOSTANDARD=LVCMOS18;

NET "K_ROW[0]"LOC=V17 | IOSTANDARD=LVCMOS18;
NET "K_ROW[1]"LOC=W18 | IOSTANDARD=LVCMOS18;
NET "K_ROW[2]"LOC=W19 | IOSTANDARD=LVCMOS18;
NET "K_ROW[3]"LOC=W15 | IOSTANDARD=LVCMOS18;
NET "K_ROW[4]"LOC=W16 | IOSTANDARD=LVCMOS18;

NET "K_COL[0]"LOC=V18 | IOSTANDARD=LVCMOS18;
NET "K_COL[1]"LOC=V19 | IOSTANDARD=LVCMOS18;
NET "K_COL[2]"LOC=V14 | IOSTANDARD=LVCMOS18;
NET "K_COL[3]"LOC=W14 | IOSTANDARD=LVCMOS18;

#NET"readn"LOC=U21 | IOSTANDARD=LVCMOS33;
#NET"RDY"LOC=U22 | IOSTANDARD=LVCMOS33;
#NET"CR"LOC=V22 | IOSTANDARD=LVCMOS33;
#NET"SEGCLK"LOC=M24 | IOSTANDARD=LVCMOS33;
#NET"SEGCLR"LOC=M20 | IOSTANDARD=LVCMOS33;
#NET"SEGDT"LOC=L24 | IOSTANDARD=LVCMOS33;
#NET"SEGEN"LOC=R18 | IOSTANDARD=LVCMOS33;
NET"LEDCLK"LOC=N26 | IOSTANDARD=LVCMOS33;
NET"LEDCLR"LOC=N24 | IOSTANDARD=LVCMOS33;
NET"LEDDT"LOC=M26 | IOSTANDARD=LVCMOS33;
NET"LEDEN"LOC=P18 | 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"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"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 "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;
```

后续步骤则为检查文件，生成比特流文件，并将文件下载到实验板，在实验板物理运行，不再赘述。

四、实验结果与分析

4.1 用原理图实现 RS 锁存器并仿真验证

如图表 4.1-1 可见，根据代码，0-100ns 中 Sn 信号为 0，Rn 信号为 1，锁存器为置 1 状态。相应地此时 Q 信号为 1，Qn 信号为 0。

100-150ns 中，Rn 和 Qn 信号都为 1，锁存器为保持状态，可见此时 Q 信号保持 1，Qn 信号保持 0。

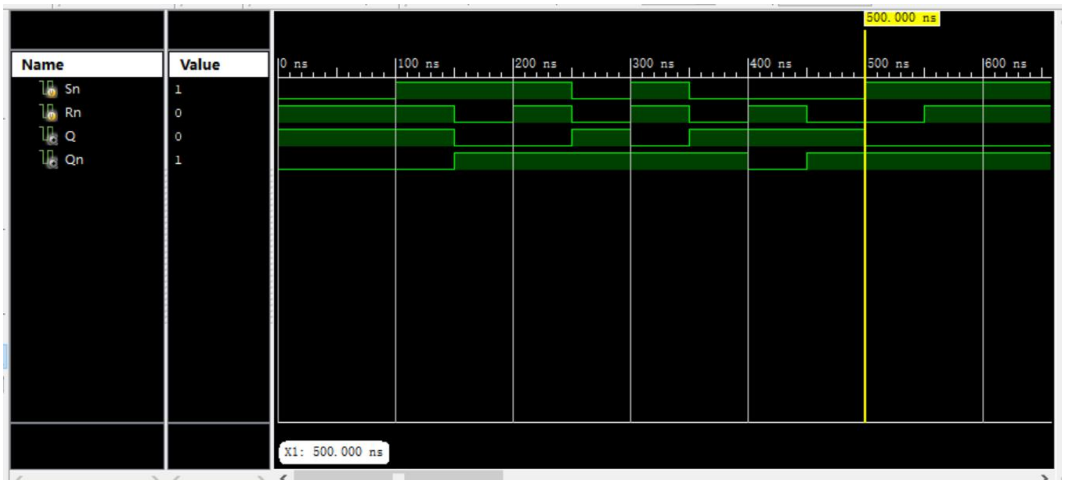
150-200ns 中，Sn 信号为 1，Rn 信号为 0，锁存器为置 0 状态。相应地此时 Q 信号为 0，Qn 信号为 1。

200-250ns 中，Rn 和 Qn 信号都为 1，锁存器为保持状态，可见此时 Q 信号保持 0，Qn 信号保持 1。

250-300ns 中，Rn 和 Qn 信号都为 0，锁存器为不确定状态。此时 Q 信号为 1，Qn 信号为 1。结合后续的结果可以看出，每次不确定状态的输出都是 Q、Qn 都为 1，这个输出是由仿真软件所确定的。现实中，由于实际的电路情况并不总是理想的，Q 和 Qn 的值都是确定的，但是具体是什么值是不能预知的。

300-350ns 中，Rn 和 Qn 信号都为 1，锁存器为保持状态。由于前一个状态为不确定状态，此时的保持状态的输出也变得确定但不能预知。图中的结果是由仿真软件确定的，现实中可能不同。

后续分析类似，不再赘述，可知设计正确。



图表 4.1-1 RS 锁存器仿真模拟实验结果

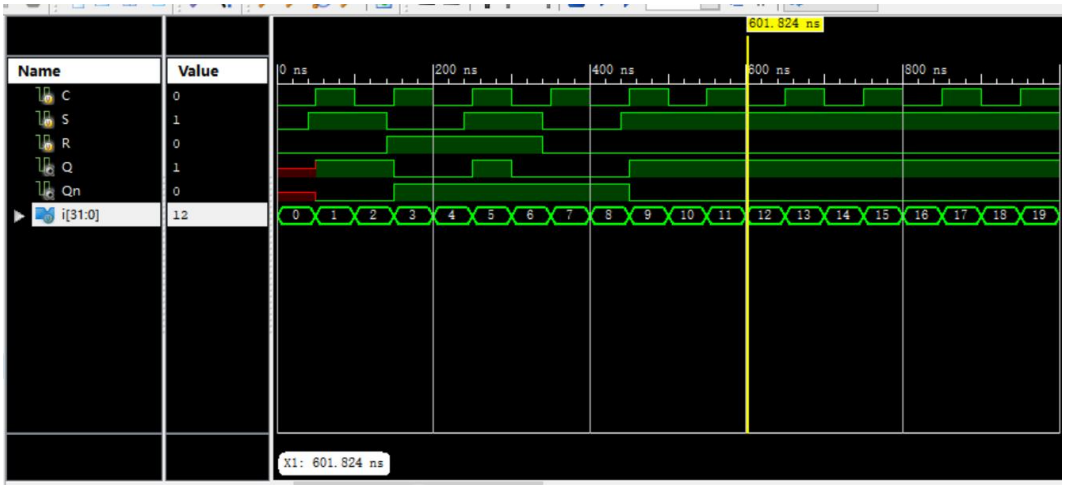
4.2 实现门控 RS 锁存器、D 锁存器并仿真验证

如图表 4.2-1 可见，根据代码，0-50ns 中，使能信号 C 为 0，此时输出信号不确定。注意到 40-50ns，S 信号为 1，但此时输出信号仍为不确定，可见使能信号起到了控制的作用。

50-100ns，使能信号 C 为 1，S 信号为 1，R 信号为 0。此时锁存器开放，为置 1 状态，可见此时 Q 信号为 1，Qn 信号为 0。

100-150ns，可见使能信号 C 为 0。其中 140-150ns 中，S 信号和 R 信号都发生了变化但输出信号没有变化。使能信号起到了控制作用。

此后，在 250-300ns，S 和 R 信号同时为 1，此时锁存器为不确定状态。此时的输出是由仿真软件所确定的。现实中，由于实际的电路情况并不总是理想的，Q 和 Qn 的值都是确定的，但是具体是什么值是不能预知的。最后由于使能信号为 0，相当于 R=1，S=1 的保持信号输入到了 RS_NAND 锁存器中；由于前一个状态是不确定的，保持状态的输出也是由仿真软件所确定的，现实中可能不同。



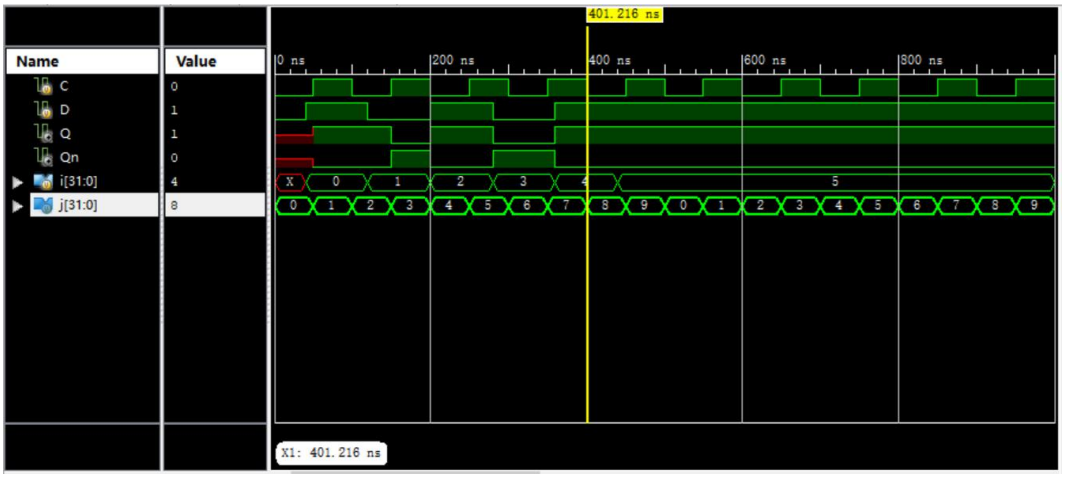
图表 4.2-1 门控 RS 锁存器仿真模拟实验结果

如图表 4.2-2 所示，根据代码，0-50ns 中，使能信号 C 为 0，此时输出信号不确定。注意到 40-50ns，D 信号为 1，但此时输出信号仍为不确定，可见使能信号起到了控制的作用。

50-100ns，使能信号 C 为 1，D 信号为 1。此时锁存器开放，为置 1 状态，可见此时 Q 信号为 1，Qn 信号为 0。

150-200ns，使能信号 C 为 1，D 信号为 0。此时锁存器开放，为置 0 状态，可见此时 Q 信号为 0，Qn 信号为 1。可见，门控 D 锁存器不存在不确定态。后续分析类似，不再赘述。

由结果可知设计正确。



图表 4.2-2 门控 D 锁存器仿真模拟实验结果

4.3 用 RS 锁存器实现 RS 主从触发器并仿真验证

如图表 4.3-1 可见，根据代码，0-55ns 中，没有有效的输入信号，因此 Y 信号不确定。而 0-100ns 中，使能信号 C 未出现下降沿，因此输出 Q 和 Qn 不确定。在存在有效输入且出现 C 的下降沿的第 100ns，可见 Q 和 Qn 信号有了确定值。

第 55ns，可见在使能信号 C 为 1 的情况下，Y 信号立刻随着 S 信号的置 1 而置 1。但此时 C 未出现下降沿，第二个 RS 锁存器依然不透明，Y 信号无法产生输出 Q 和 Qn。

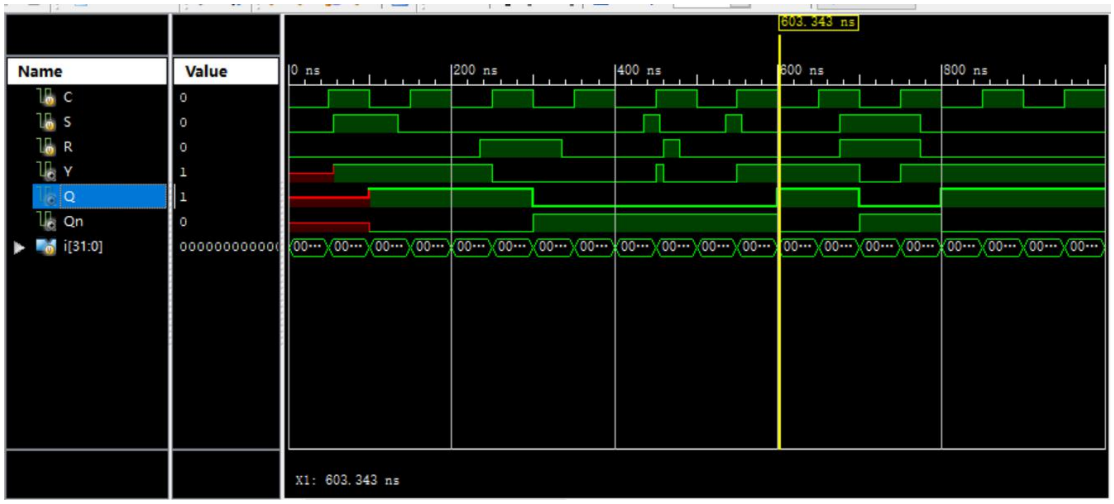
而在 235-250ns，可见在 C 的上升沿出现前，为 1 的 R 信号（置 0 信号）不能改变 Y 信号。C 的上升沿出现后，Y 信号被置 0。相应地，第 300ns，随后的第一个 C 的下降沿出现时，Q 信号被置 0，Qn 信号置 1。

在 450-500ns 的采样窗口，可见由于置 1 信号（S=1，R=0）出现后又立刻出现了置 0 信号（S=0，R=1），一次性采样问题并未出现。

在 550-600ns 的采样窗口，短暂地出现了置 1 信号，此时 Y 信号随之置 1。直到采样窗口结束，由于没有置 0 信号，Y 信号保持为 1。此后第 600ns，出现 C 的下降沿时，Q 信号随 Y 信号置 1，Qn 信号置 0，出现了一次性采样。

而几次不确定状态（S=0，R=0）和保持状态（S=1，R=1）的分析与锁存器的分析类似，不再赘述。

其余结果分析类似，不再赘述。由该结果可知设计正确。



图表 4.3-1 RS 主从触发器仿真模拟实验结果

4.4 用 D 锁存器和用 RS 锁存器实现主从 D 触发器并仿真验证

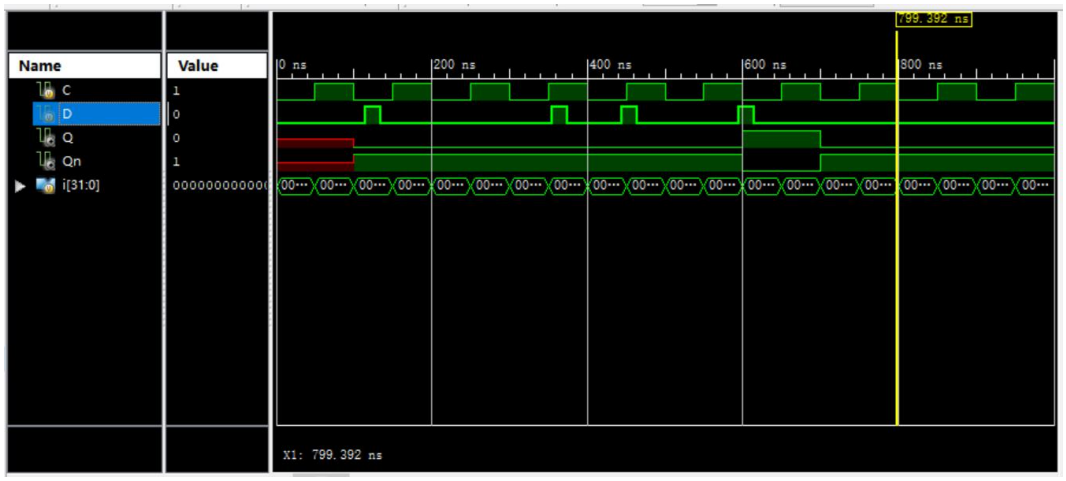
如图表 4.4-1 可见，根据代码，0-100ns 中，由于没有 C 的下降沿出现，因此输出 Q 和 Qn 信号不确定。

115-135ns 中，由于 D 的脉冲并不是出现在采样窗口，D 的变化没有引起触发器的变化。

355-375ns 为一次性采样情形。主从 D 触发器的内容是由采样窗口最后时刻的输入信号决定。可见由于该采样窗口的最后 D 为 0，因此触发器输出没有变化。

而后续的两次 D 信号变化则说明了，主从 D 触发器是由下降沿触发变化而不是由上升沿触发变化。

由结果可知设计正确。



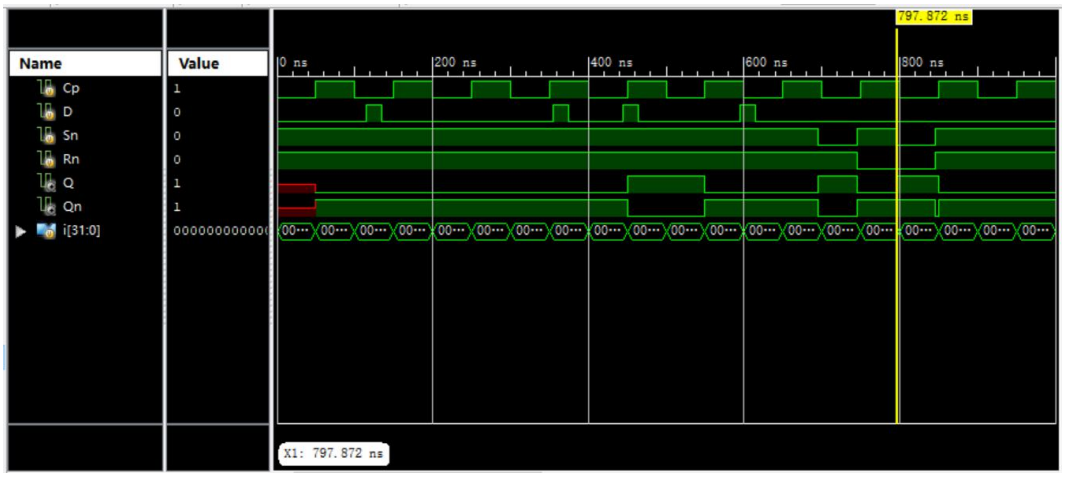
图表 4.4-1 主从 D 触发器仿真模拟实验结果

4.5 用原理图实现维持阻塞型 D 触发器

如图表 4.5-1 可见，根据代码，与主从 D 触发器的分析类似，0-615ns 的四次 D 的脉冲说明维持阻塞型 D 触发器由上升沿触发。

而 695ns 之后的变化可以看出，无论使能信号 Cp 为何值，当输入置 1 信号（Sn=0，Rn=1），输出 Q 为 1，Qn 为 0；当输入置 0 信号（Sn=1，Rn=0），输出 Q 为 0，Qn 为 1。而 Sn=Rn=0 时为不确定状态，输出由仿真软件给出。

由结果可知，设计正确。



图表 4.5-1 维持阻塞型 D 触发器仿真模拟实验结果

4.6 触发器物理测试

实验结果如图表所示。

图表 4.6-1 RS 主从触发器测试结果

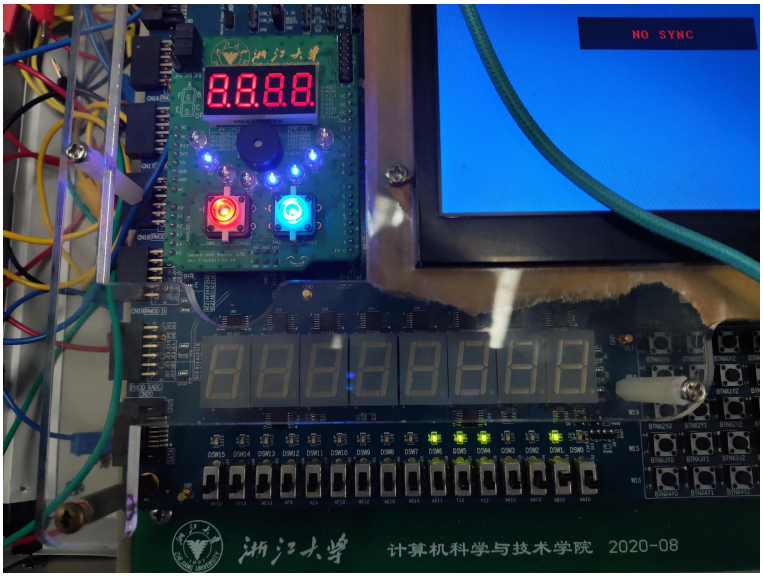
功能	R	S	C 单步	Y	Q	Qn	备注
置 1	0	1	0	1	X	X	Y 初始值为 1
			1→0	1	X→1	X→0	Q 置 1
保持	0	0	0	1	1	0	触发器阻塞
			1→0	1	1	0	保持
置 0	1	0	0	1	1	0	触发器阻塞
			1→0	0	1→0	0→1	Q 置 0
保持	0	0	0	0	0	1	触发器阻塞
			1→0	0	0	1	保持
无定义	1	1	0	0	0	1	触发器阻塞
			1→0	X	X	X	不确定
保持	0	0	0	X	X	X	不确定
			1→0	X	X	X	不确定

图表 4.6-2 一次性采样测试结果

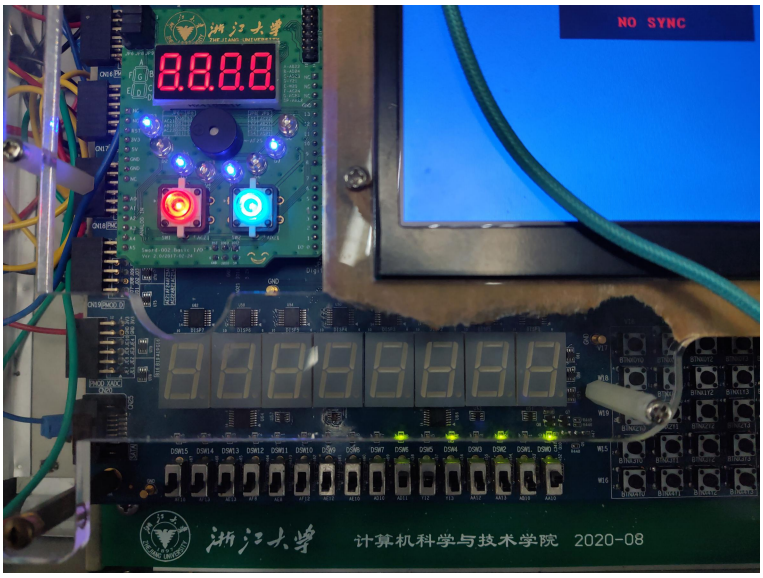
功能		R	S	C 单步	Y	Q	Qn	备注
一次性采样	置 1	0	1	1→0	1	1	0	
	采样 RS	0	0	1	1	1	0	按住 BTN0，开启采样窗口
	R=脉冲	1	0	1	0	1	0	按住 BTN0，Y 信号随置 0 信号而置 0，Q 不变
	保持	0	0	1→0	0	0	1	输出 Q 变为 0，产生了一次性采样
一次性采样	置 1	0	1	1→0	1	1	0	
	采样 RS	0	0	1	1	1	0	按住 BTN0，开启采样窗口
	S=脉冲	0	1	1	1	1	0	按住 BTN0，Y 信号随置 1 信号而置 1，Q 不变
	保持	0	0	1→0	1	1	0	输出 Q 为 1，此时为一个隐性的一次性采样

图表 4.6-3 D 触发器测试结果

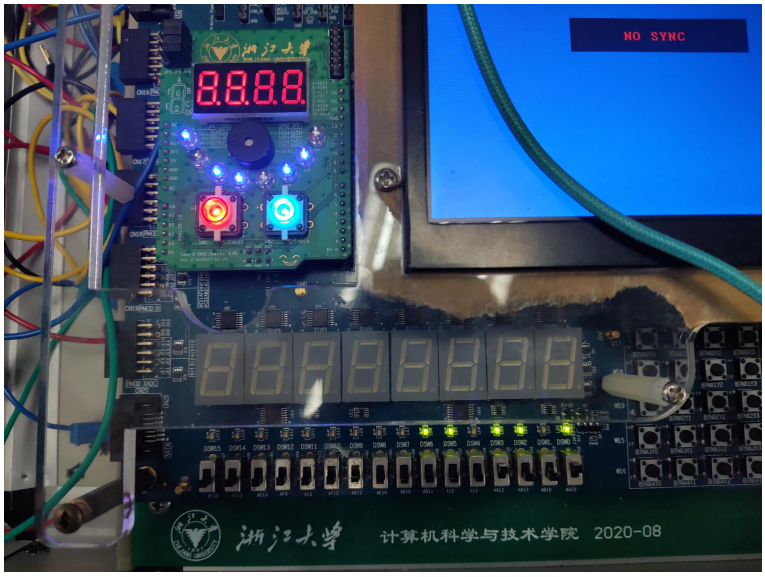
主从 D 触发器				边沿 D 触发器					
D	C	Q	Qn	Rn	Sn	D	Cp	Q	Qn
0	1→0	X→0	X→1	1	1	1	0→1	X→1	X→0
1	0	0	1	1	1	1	1→0	1	0
1	0→1	0	1	0	1	0	X	0	1
1	1→0	0→1	1→0	1	0	0	X	1	0



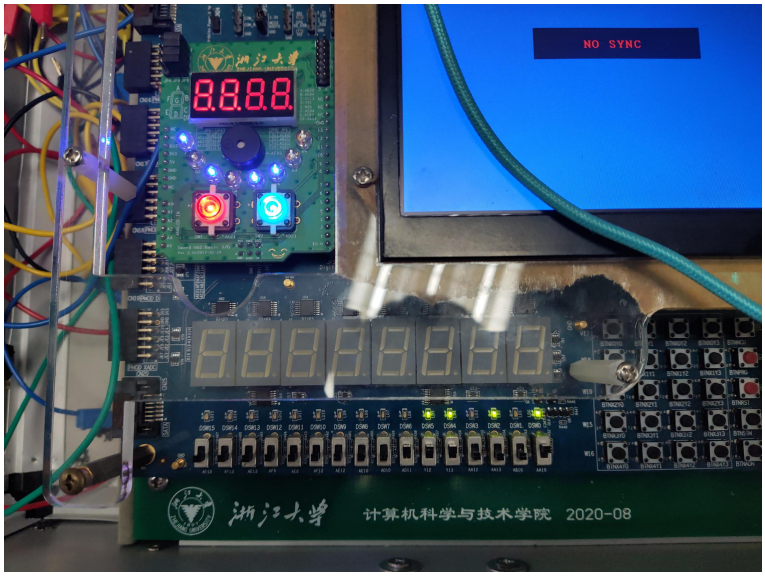
图表 4.6-4 RS 主从触发器 R=1，仅 Qn 灯亮



图表 4.6-5 RS 主从触发器 S=1，仅 Qn 灯灭



图表 4.6-6 主从 D 触发器 D=1，Q 灯亮



图表 4.6-7 边沿 D 触发器 Sn=Rn=D=1，Q 灯亮

五、讨论、心得

本次需要有许多需要自己设计的仿真激励代码，乍一看好像很难，但只要仔细分析电路结构，想清楚自己的仿真的目的（即需要检验的重点），最后分析仿真，就可以完成一套自己设计、自己分析的仿真激励流程。

本次实验由浅入深，让我一步步了解了毛刺、空翻等现象，明白了为何不同的锁存器、触发器有着不同的应用范围。