一、实验内容

用 Verilog 实现同步复位的 D 触发器，该触发器只有一个数据输入端 D,，在始终上升沿的时候触发器置位， 当数据输入 D =0　　且在时钟上升沿，触发器复位。

二、硬件逻辑图

该实验不需要 logicsim 验证。

三、模块建模

该触发器具有三个输入端，分别为时钟信号 CLK，输入信号 D，复位信号 RST_n,两位的输出信号，Q1 代表 Q， Q2 代表 Q 非。依据行为描述进行 Verilog 书写
D 触发器的真值表：

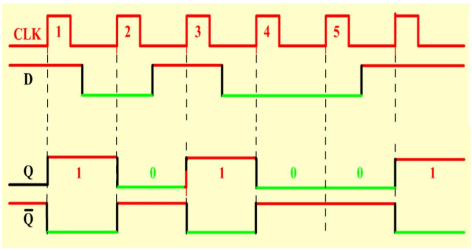
输入		输出		说明
D	CLK	Q	$\overline{Q}$	
1	↑	1	0	置位(存1)
0	↑	0	1	复位(存0)

```
module Synchronous_D_FF(
    input CLK,
    input D,
    input RST_n,
    output reg Q1,
    output reg Q2
    );

    always @(posedge CLK)
    begin

        if (!RST_n)
        begin
            Q1 = 0;
            Q2 = 1;
        end
        else
        begin
            Q1 = D;
            Q2 = !Q1;
        end

    end
```

endmodule

四、测试模块建模

同步复位测试的 RST_n 需要等待时钟上升沿进行变化，
通过模拟下图进行 testbench 的设计



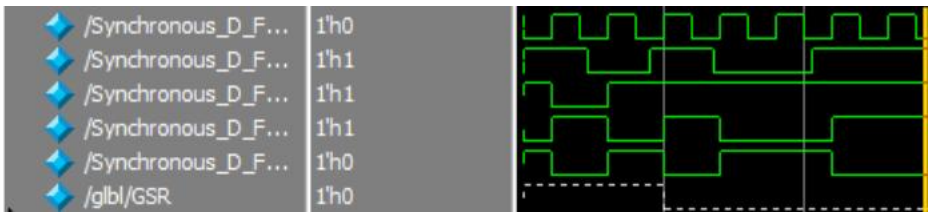
其中时钟脉冲产生方式是使用 always #20 CLK 产生间隔为 20s 的脉冲，D 的赋值同上图

```
module Synchronous_D_FF_tb();
    reg CLK;
    reg D;
    reg RST_n;
    wire Q1;
    wire Q2;

    Synchronous_D_FF sdff(.CLK(CLK), .D(D), .RST_n(RST_n), .Q1(Q1), .Q2(Q2));

    initial
    begin
    CLK = 1;
    #1 CLK = 0;
    end
    always #20 CLK = ~CLK;

    initial
    begin
        RST_n = 1;
        @(posedge CLK);
        RST_n = 0;
        #30
        @(posedge CLK);
        RST_n = 1;

    end
```

```
    initial
    begin
        D = 0;
        #1 D = 1;
        #45 D = 0;
        #45 D = 1;
        #45 D = 0;
        #70 D = 1;

    end

endmodule
```

五、实验结果

1. Modelsim 仿真波形图

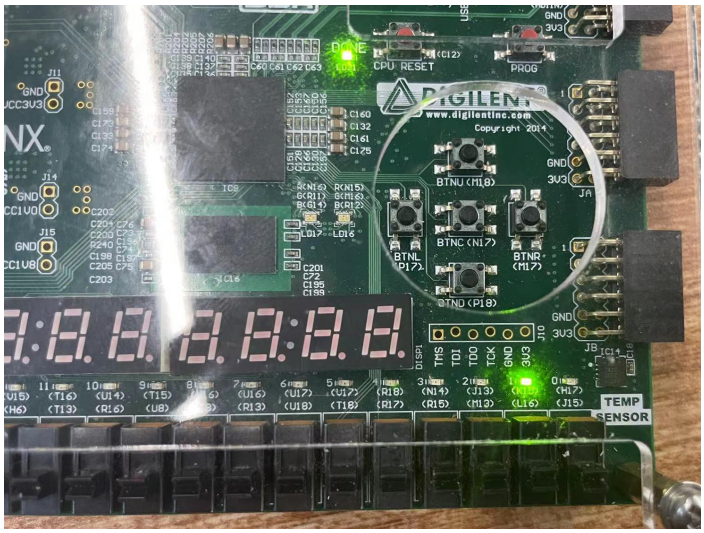


可以看到所输出的波形图大致符合预期设想，波形仿真达到预期

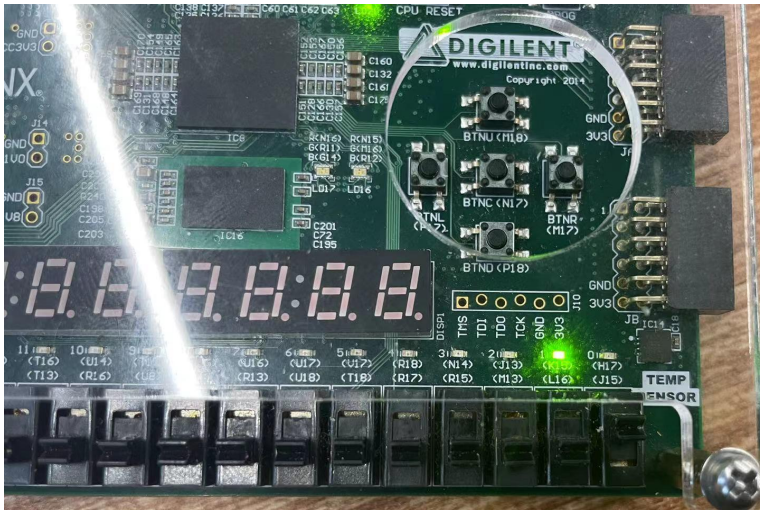
2. 下板验证

CLK 输入 M17，RES_n 输入 M18，D 输入 J15，输出 Q1:M17, Q2：K15

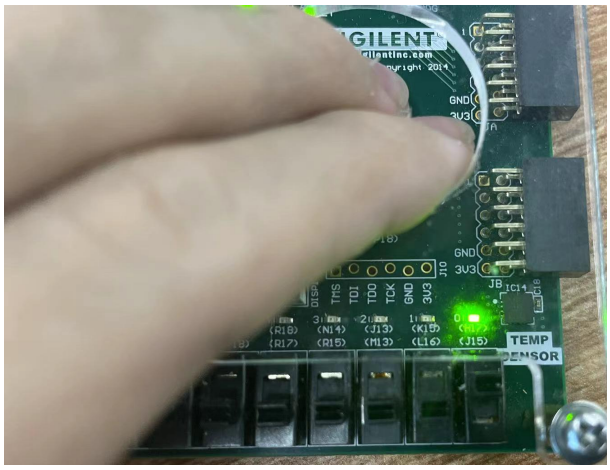
初始状态为 0 态 ，按下清零键，保证清零输入维持在高电平



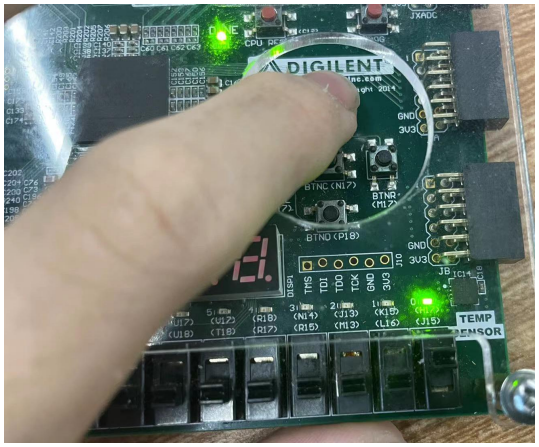
当无时钟上升沿时，D 输入为高电平，输出维持在低电平



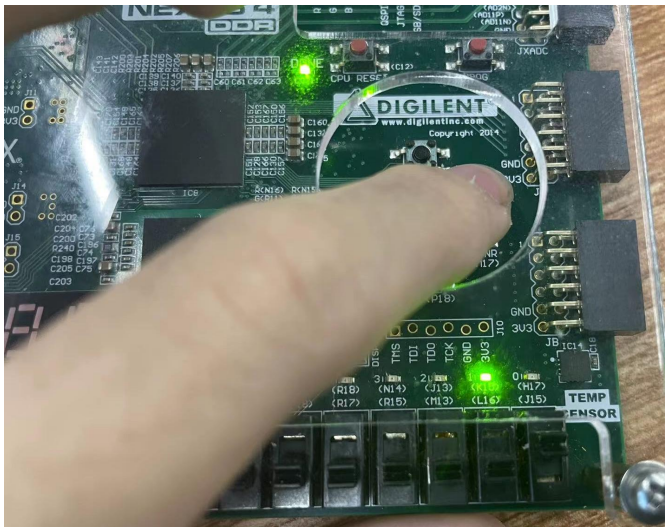
当时钟上升沿到来，D 输入为高电平，输出为高电平



同步清零，单独清零无作用



同步清零，时钟上升沿清零



下板验证逻辑正确

一、 实验内容

用 Verilog 实现异步复位的 D 触发器，该触发器只有一个数据输入端 D，，在始终上升沿的时候触发器置位，当重置信号为低电平时进行复位清零

二、 硬件逻辑图

该实验未要求

三、 模块建模

通过在 always 检测复位信号的下降沿来来实现不依赖于时钟的复位，当复位信号变为 0 时，输出为 0 态

```
module Asynchronous_D_FF(
    input CLK,
    input D,
    input RST_n,
    output reg Q1,
    output reg Q2
    );

    always @(posedge CLK or negedge RST_n)
    begin
        if (!RST_n)
        begin
            Q1 = 0;
            Q2 = 1;

        end
        else
        begin
            Q1 = D;
            Q2 = !Q1;

        end
    end
endmodule
```

四、 测试模块建模

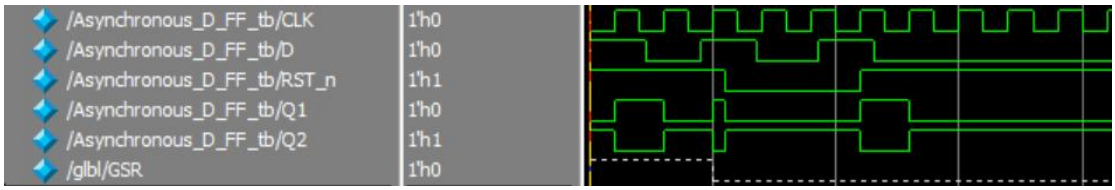
利用 always #20 CLK = ~CLK 产生间隔为 20 的周期时钟信号
初始 D 输入为高电平，间隔 45sD 输入由 0 转为 1，清零信号在 110s 后高电平转为低电平，

再次间隔 110s 后转为高电平

```
module Asynchronous_D_FF_tb();
    reg CLK;
    reg D;
    reg RST_n;
    wire Q1;
    wire Q2;

    Asynchronous_D_FF adff(.CLK(CLK), .D(D), .RST_n(RST_n), .Q1(Q1), .Q2(Q2));

    initial
        begin
        CLK = 1;
        #1 CLK = 0;
        end
        always #20 CLK = ~CLK;


        initial
        begin
            RST_n = 1;
            #110
            RST_n = 0;
            #110
            RST_n = 1;
        end

        initial
        begin
            D = 0;
            #1 D = 1;
            #45 D = 0;
            #45 D = 1;
            #45 D = 0;
            #50 D = 1;
            #45 D = 0;

        end

endmodule
```

五、 实验结果

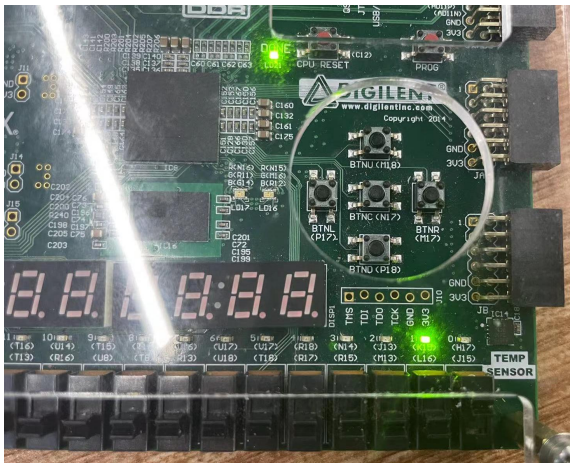
1. ModelSim 波形仿真图



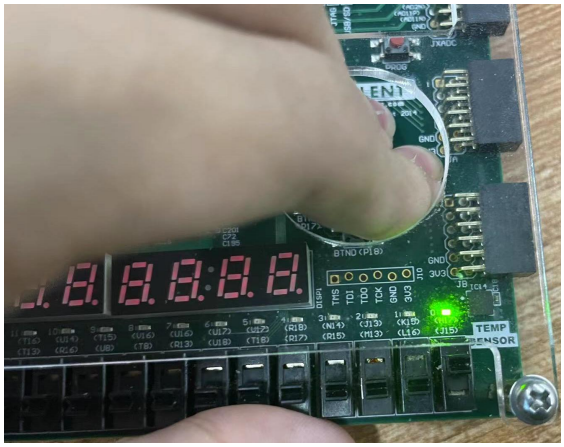
可以观察到
在第一个时钟上升沿，清零信号高电平无作用，D 信号高电平，输出信号 Q 转换为高电平
在第二个时钟上升沿，清零信号高电平无作用，D 信号低电平，输出信号 Q 转为低电平
在 110 左右，清零信号低电平，输出信号置零
在 220 左右，清零信号转换为高电平，输入信号高电平，且此时为时钟上升沿，则输出信号转换为高电平
则波形仿真符合预期

2. 下板验证
CLK 输入 M17，RES_n 输入 M18，D 输入 J15，输出 Q1:M17, Q2：K15

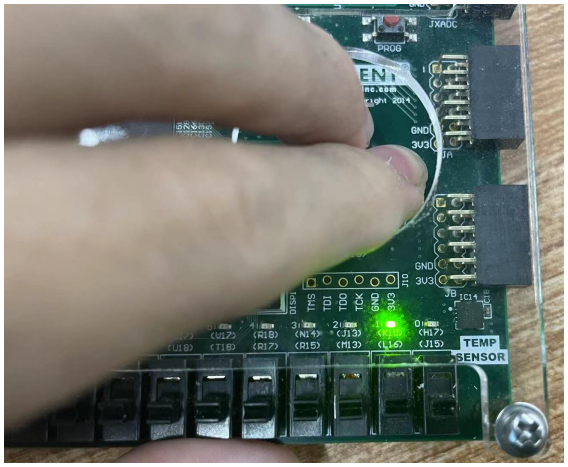
初始状态



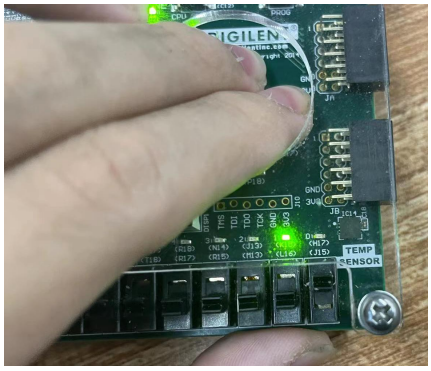
清零信号高电平无效，时钟上升沿到来，D 输入高电平，输出为高电平



清零信号高电平无效，时钟上升沿到来，D 输入低电平，输出为低电平



时钟没有到上升沿，清零信号低电平有效，输出端清零，实现异步



下板验证逻辑正确

一、 实验内容

用 Verilog 实现 JK 触发器，在置位、复位方面与 SR 触发器功能相同，但在 SR 触发器的不稳定状态，JK 触发器的处理是进行输出信号的反转交替，其中复位信号是异步清零的

二、 硬件逻辑图

该实验未要求

三、 模块建模

进行行为描述，首先特殊判断清零信号是否为低电平，若为高电平则对相应的 JK 信号不同情况进行判断，决定输出是置相应位还是进行翻转。注意在反转时需要用临时变量记录过程值。

```
module JK_FF(
    input CLK,
    input J,
    input K,
    input RST_n,
    output reg Q1,
    output reg Q2
    );
    reg temp;

    always @(posedge CLK or negedge RST_n)
    begin
        //reset
        if (!RST_n)
        begin
            Q1 = 0;
            Q2 = 1;
        end

        //flip
        else if (J == 1 && K == 1)
        begin
            temp = Q1;
            Q1 = Q2;
            Q2 = temp;
        end

        else if (J == 0 && K == 1)
        begin
```

```
            Q1 = 0;
            Q2 = 1;
        end

        else if (J == 1 && K == 0)
        begin
            Q1 = 1;
            Q2 = 0;
        end
    end
endmodule
```

四、 测试模块建模

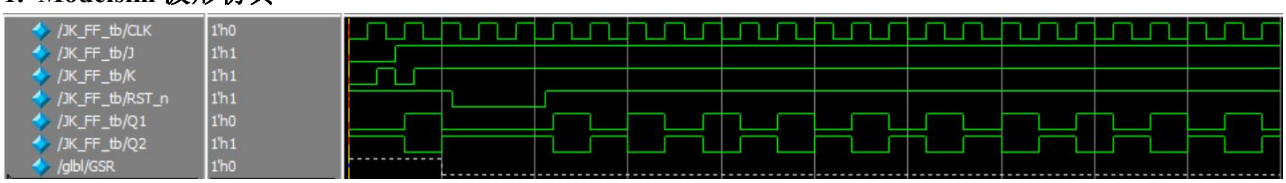
分别依次测试清零信号无效时 JK 取值为 00，01，10，11 的情况，在 110 左右开始测试清零信号的异步置零情况，再待 100 之后清零信号恢复高电平

```
module JK_FF_tb();
    reg CLK;
    reg J;
    reg K;
    reg RST_n;
    wire Q1;
    wire Q2;

    JK_FF jf(
        .CLK(CLK),
        .J(J),
        .K(K),
        .RST_n(RST_n),
        .Q1(Q1),
        .Q2(Q2)
    );

    //CLK
    initial
    begin
    CLK = 1;
    #1 CLK = 0;

    end
    always #20 CLK = ~CLK;
```

```
    //J, K
    initial
    begin

        J = 0;
        K = 0;
        #30
        J = 0;
        K = 1;
        #20
        J = 1;
        K = 0;
        #20
        J = 1;
        K = 1;

    end

    //RST_n
    initial
    begin
    RST_n = 0;
    #1
    RST_n = 1;
    #110
    RST_n = 0;
    #100
    RST_n = 1;

    end
endmodule
```

五、 实验结果

1. Modelsim 波形仿真



在 110 以前，置零信号无效，当 JK 为 00 时，可以看到输出保持不变
当 JK 为 01 时，可以看到此时输出 Q 为低电平，
当 JK 为 10 时，此时输出 Q 为高电平，
在 110 左右，清零信号有效，输出信号置 0
100 过后，因 JK 输入一直为 11，则每次到时钟上升沿，输出信号反转
波形仿真符合预期

2. 下板验证

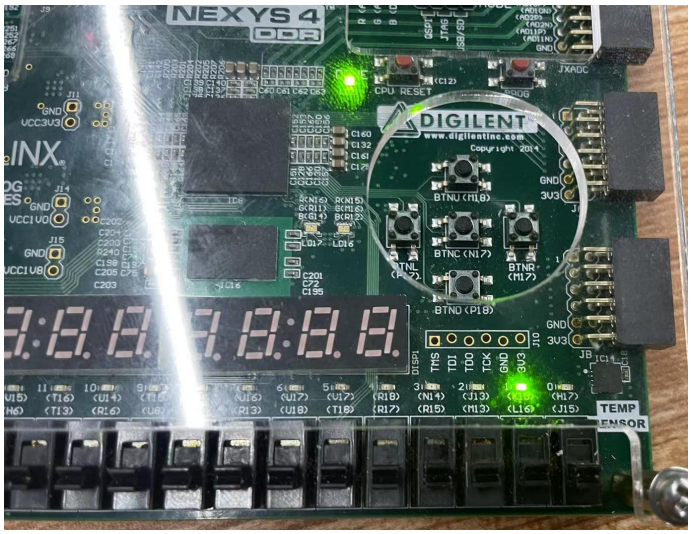
JK 触发器功能表

输入			输出		说明
J	K	CLK	Q	$\overline{Q}$	
0	0	↑	Q^n	$\overline{Q^n}$	保持
0	1	↑	0	1	置0
1	0	↑	1	0	置1
1	1	↑	$\overline{Q^n}$	Q^n	交替

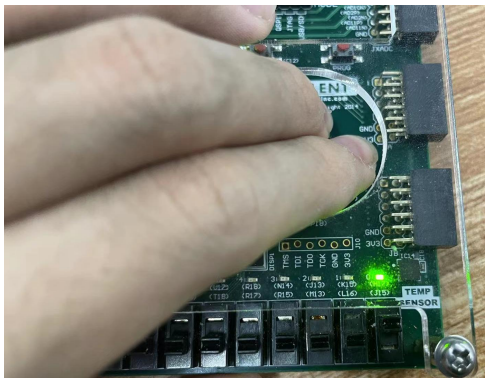
管脚分配

变量	CLK	J	K	RST_n	$Q1$	$Q2$
N4 板上的管脚	BTNR(M17)	SW0(J15)	SW1(L16)	BTNU(M18)	LD0(H17)	LD1(K15)

初始状态

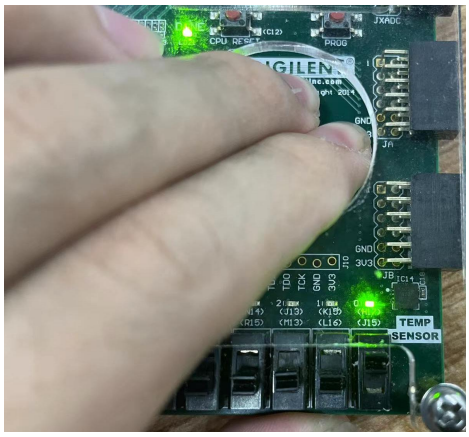


时钟上升沿，JK:10



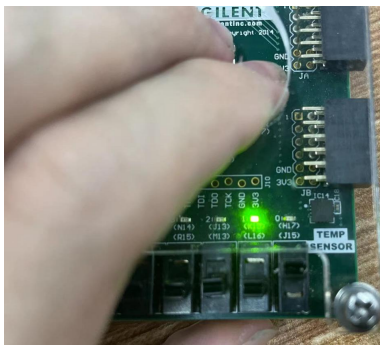
输出 10

时钟上升沿，JK:00



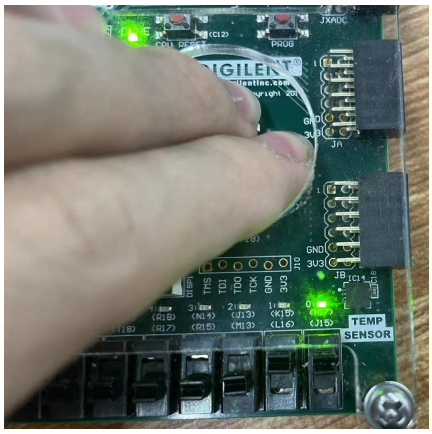
输出保持

时钟上升沿，JK:01



输出 01

时钟上升沿，JK:11



反转

可以看到与预期结果一致，下板验证逻辑正确

一、 实验内容

首先实现带有使能端的异步复位 D 触发器，将 32 个 D 触发器进行组合形成一个 32 位的 PC 寄存器。利用 Verilog 实例化建模，基于 D 触发器实现 32 位的 PC 寄存器。其中清零信号不受使能端控制，使能信号有效时，在时钟的上升沿，寄存器会把存放在寄存器中的信号量输出到输出端

二、 硬件逻辑图

该试验未要求 Logicsim 原理图

三、 模块建模

首先实现带有使能端的异步复位的 D 触发器，其次在主模块中实例化 32 个此类型的触发器，并连接好输入输出端口、时钟信号、清零信号等。同时注意此时清零信号未高电平时未有效，进行清零

```
module ADFF(
    input clk,
    input d,
    input rst,
    input ena,
    output reg Q1
);
    always @(posedge clk or posedge rst)
    begin
        if (rst)
        begin
            Q1 = 0;
        end

        else
        begin
            if (ena)
            begin
                Q1 = d;
            end
        end
    end
endmodule

module pcreg(
    input clk,
```

```
    input rst,
    input ena,
    input [31:0] data_in,
    output [31:0] data_out
    );


        ADFF p0(.clk(clk), .d(data_in[0]), .rst(rst), .ena(ena), .Q1(data_out[0]));
        ADFF p1(.clk(clk), .d(data_in[1]), .rst(rst), .ena(ena), .Q1(data_out[1]));
        ADFF p2(.clk(clk), .d(data_in[2]), .rst(rst), .ena(ena), .Q1(data_out[2]));
        ADFF p3(.clk(clk), .d(data_in[3]), .rst(rst), .ena(ena), .Q1(data_out[3]));
        ADFF p4(.clk(clk), .d(data_in[4]), .rst(rst), .ena(ena), .Q1(data_out[4]));
        ADFF p5(.clk(clk), .d(data_in[5]), .rst(rst), .ena(ena), .Q1(data_out[5]));
        ADFF p6(.clk(clk), .d(data_in[6]), .rst(rst), .ena(ena), .Q1(data_out[6]));
        ADFF p7(.clk(clk), .d(data_in[7]), .rst(rst), .ena(ena), .Q1(data_out[7]));
        ADFF p8(.clk(clk), .d(data_in[8]), .rst(rst), .ena(ena), .Q1(data_out[8]));
        ADFF p9(.clk(clk), .d(data_in[9]), .rst(rst), .ena(ena), .Q1(data_out[9]));
        ADFF p10(.clk(clk), .d(data_in[10]), .rst(rst), .ena(ena), .Q1(data_out[10]));
        ADFF p11(.clk(clk), .d(data_in[11]), .rst(rst), .ena(ena), .Q1(data_out[11]));
        ADFF p12(.clk(clk), .d(data_in[12]), .rst(rst), .ena(ena), .Q1(data_out[12]));
        ADFF p13(.clk(clk), .d(data_in[13]), .rst(rst), .ena(ena), .Q1(data_out[13]));
        ADFF p14(.clk(clk), .d(data_in[14]), .rst(rst), .ena(ena), .Q1(data_out[14]));
        ADFF p15(.clk(clk), .d(data_in[15]), .rst(rst), .ena(ena), .Q1(data_out[15]));
        ADFF p16(.clk(clk), .d(data_in[16]), .rst(rst), .ena(ena), .Q1(data_out[16]));
        ADFF p17(.clk(clk), .d(data_in[17]), .rst(rst), .ena(ena), .Q1(data_out[17]));
        ADFF p18(.clk(clk), .d(data_in[18]), .rst(rst), .ena(ena), .Q1(data_out[18]));
        ADFF p19(.clk(clk), .d(data_in[19]), .rst(rst), .ena(ena), .Q1(data_out[19]));
        ADFF p20(.clk(clk), .d(data_in[20]), .rst(rst), .ena(ena), .Q1(data_out[20]));
        ADFF p21(.clk(clk), .d(data_in[21]), .rst(rst), .ena(ena), .Q1(data_out[21]));
        ADFF p22(.clk(clk), .d(data_in[22]), .rst(rst), .ena(ena), .Q1(data_out[22]));
        ADFF p23(.clk(clk), .d(data_in[23]), .rst(rst), .ena(ena), .Q1(data_out[23]));
        ADFF p24(.clk(clk), .d(data_in[24]), .rst(rst), .ena(ena), .Q1(data_out[24]));
        ADFF p25(.clk(clk), .d(data_in[25]), .rst(rst), .ena(ena), .Q1(data_out[25]));
        ADFF p26(.clk(clk), .d(data_in[26]), .rst(rst), .ena(ena), .Q1(data_out[26]));
        ADFF p27(.clk(clk), .d(data_in[27]), .rst(rst), .ena(ena), .Q1(data_out[27]));
        ADFF p28(.clk(clk), .d(data_in[28]), .rst(rst), .ena(ena), .Q1(data_out[28]));
        ADFF p29(.clk(clk), .d(data_in[29]), .rst(rst), .ena(ena), .Q1(data_out[29]));
        ADFF p30(.clk(clk), .d(data_in[30]), .rst(rst), .ena(ena), .Q1(data_out[30]));
        ADFF p31(.clk(clk), .d(data_in[31]), .rst(rst), .ena(ena), .Q1(data_out[31]));

endmodule
```

四、 测试模块建模

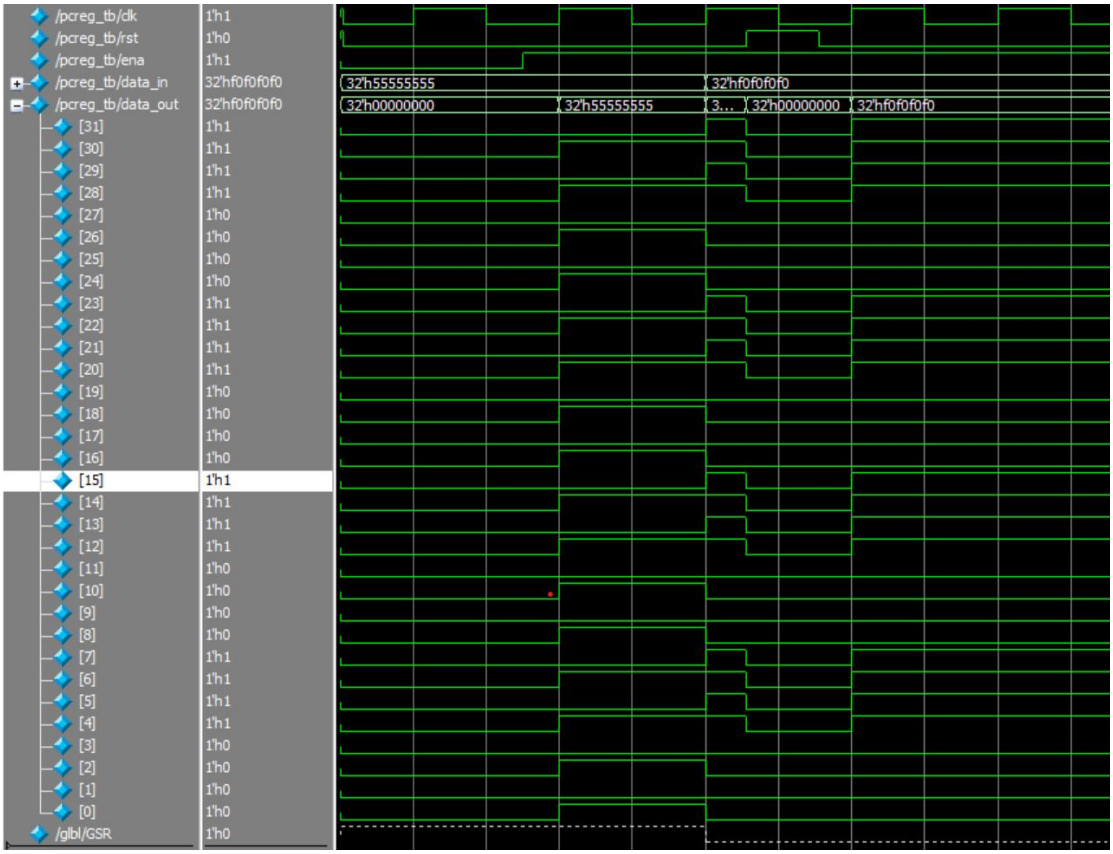
初始时刻时钟置为 0，利用 always #20 CLK = ~CLK 产生时钟信号
前 50，ena 使能信号置为 0，仅清零信号有效，50 过后，ena 信号置为 1，
初始将 data 输入置为 32'b01010101010101010101010101010101
100 过后将信号置为 32'b11110000111100001111000011110000，
此时再 110 时清零信号 rst 置为 1，输出清零持续 20，清零信号恢复低电平

```
module pcreg_tb();
    reg clk;
    reg rst;
    reg ena;
    reg [31:0] data_in;
    wire [31:0] data_out;


    pcreg pc(.clk(clk), .rst(rst), .ena(ena), .data_in(data_in), .data_out(data_out));


    //clk
    initial
    begin
        clk = 1;
        #1 clk = 0;
    end
    always #20 clk = ~clk;


    //rst
    initial
    begin
        rst = 1;
        #1 rst = 0;
        #110 rst = 1;
        #20 rst = 0;

    end

    //ena
    initial
    begin
        ena = 0;
        #50 ena = 1;
```

```
    end


    //data_in
    initial
    begin
        data_in = 32'b01010101010101010101010101010101;
        #100
        data_in = 32'b11110000111100001111000011110000;
    end
endmodule
```

五、 实验结果

1. Modelsim 波形仿真



在 50 以前，使能端 ena 无作用，无输出
50 - 100，此时输出信号为输入信号 32'b01010101010101010101010101010101
100 - 110， 此时输出信号为输入信号 32'b11110000111100001111000011110000
110 - 130， 此时置零信号高电平，输出置零
130 以后置零信号变为低电平，输出信号为输入信号 32'b11110000111100001111000011110000

可以看到输出波形图符合预期，波形验证逻辑正确