实验二 基本功能模块程序设计

**一、实验目的**

1. 掌握Verilog语言框架，编程及调试的方法。
2. 熟悉Verilog的基本语法。
3. 掌握Verilog中时序模块电路的设计方法。
4. 熟悉Verilog中层次结构的设计方法。
5. 掌握Logisim的使用。

**二、实验内容（同时用Logisim和Vivado实现）**

1. 完成编码器、译码器等功能。
2. 完成一个触发器电路模块（使能端、复位）的设计。
3. 完成寄存器和移位寄存器（循环移位）电路模块的设计。
4. 完成各种数字（个人学号末两位）进制的计数器。
5. 掌握以上电路的程序结构和风格。
6. 观察和分析仿真波形，注重输入输出之间的时序关系。

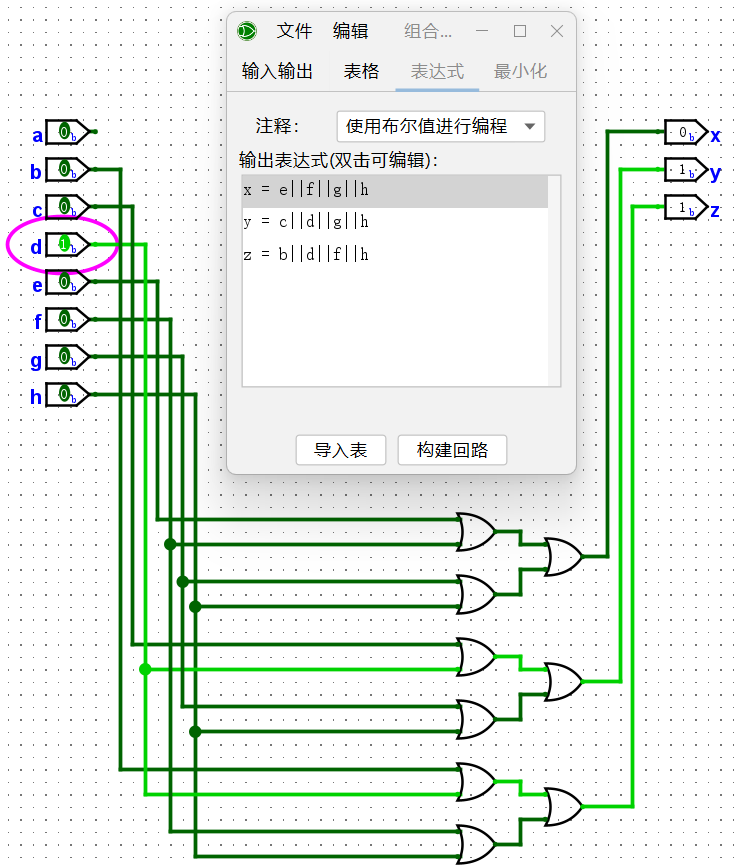
**三、实验要求**

1. 画出模块的电路图。
2. 分析电路的仿真波形。
3. 记录设计和调试过程。

**四、实验代码及结果**

1）8-3编码器

Logisim实现



Vivado实现

* 1. 8-3编码器（assign语句实现）

代码：

module encoder83\_assign(I, Y);

input I;

output Y;

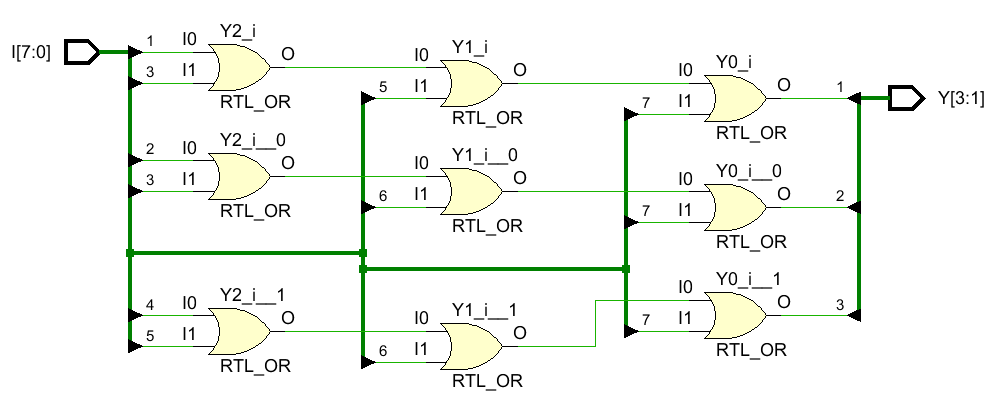
wire[7:0] I;

wire[3:1] Y;

assign Y = {I[4]|I[5]|I[6]|I[7], I[2]|I[3]|I[6]|I[7], I[1]|I[3]|I[5]|I[7]};

endmodule

电路图：



* 1. 8-3编码器（case语句实现）

代码：

module encoder83\_case(I, Y);

input I;

output Y;

wire[7:0] I;

reg[3:1] Y;

always @(I) begin

case(I)

8'b0000\_0001: Y = 3'b000;

8'b0000\_0010: Y = 3'b001;

8'b0000\_0100: Y = 3'b010;

8'b0000\_1000: Y = 3'b011;

8'b0001\_0000: Y = 3'b100;

8'b0010\_0000: Y = 3'b101;

8'b0100\_0000: Y = 3'b110;

8'b1000\_0000: Y = 3'b111;

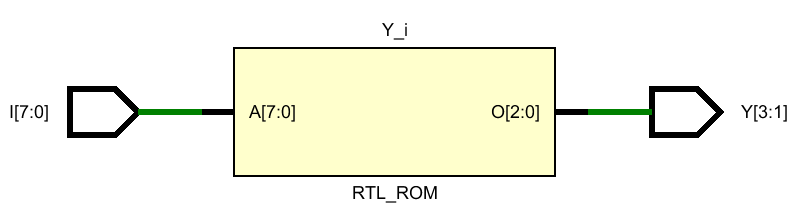
default: Y = 3'b000;

endcase

end

endmodule

电路图：



* 1. 8-3优先编码器（if语句实现）

代码：

module pre\_encoder83\_if(I, Y);

input I;

output Y;

wire[7:0] I;

reg[3:1] Y;

always @(\*) begin

if(I[7] == 1) Y = 3'b111;

else if(I[6] == 1) Y = 3'b110;

else if(I[5] == 1) Y = 3'b101;

else if(I[4] == 1) Y = 3'b100;

else if(I[3] == 1) Y = 3'b011;

else if(I[2] == 1) Y = 3'b010;

else if(I[1] == 1) Y = 3'b001;

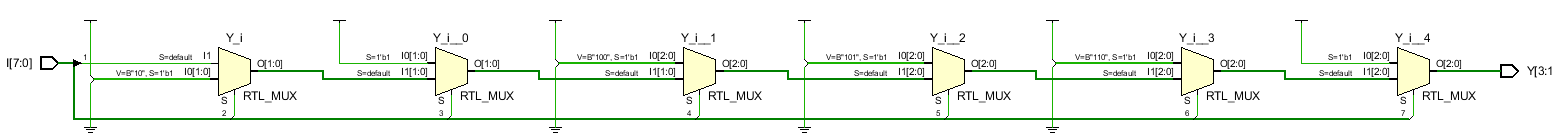
else if(I[0] == 1) Y = 3'b000;

else Y = 3'b000;

end

endmodule

电路图：



* 1. 8-3优先编码器（casex语句实现）

代码：

module pre\_encoder83\_case(I, Y);

input I;

output Y;

wire[7:0] I;

reg[3:1] Y;

always @(I) begin

casex(I)

8'b0000\_0001: Y = 3'b000;

8'b0000\_001X: Y = 3'b001;

8'b0000\_01XX: Y = 3'b010;

8'b0000\_1XXX: Y = 3'b011;

8'b0001\_XXXX: Y = 3'b100;

8'b001X\_XXXX: Y = 3'b101;

8'b01XX\_XXXX: Y = 3'b110;

8'b1XXX\_XXXX: Y = 3'b111;

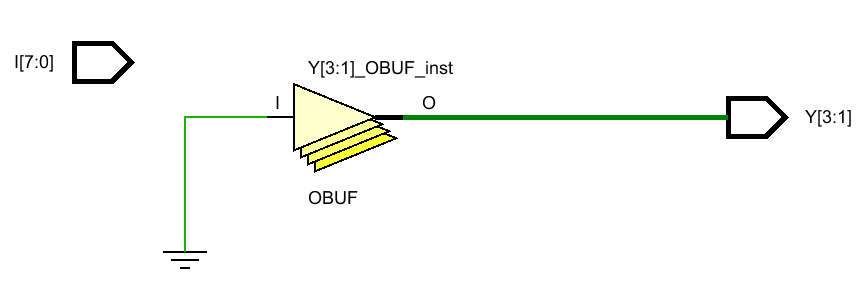
default: Y = 3'b000;

endcase

end

endmodule

电路图：



仿真：

module sim\_encoder83();

reg[7:0] x;

wire[2:0] y\_assign, y\_case, y\_pre\_case, y\_pre\_if;

integer i;

initial begin

x = 1;

for(i = 0; i < 7; i = i + 1) #10 x = x \* 2;

#10 x = 128;

while(x > 0) #5 x = x - 1;

end

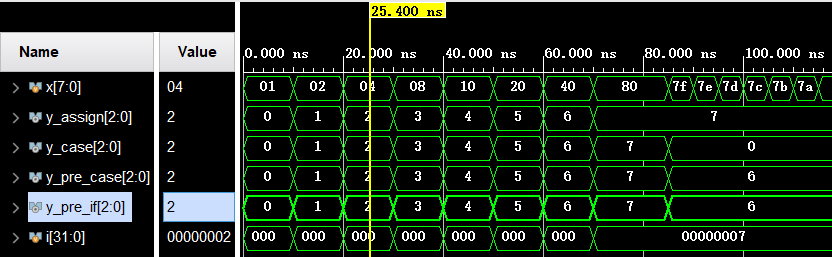
encoder83\_assign encoder\_assign\_1(x, y\_assign);

encoder83\_case encoder\_case\_1(x, y\_case);

pre\_encoder83\_case pre\_encoder83\_case\_1(x, y\_pre\_case);

pre\_encoder83\_if pre\_encoder83\_if\_1(x, y\_pre\_if);

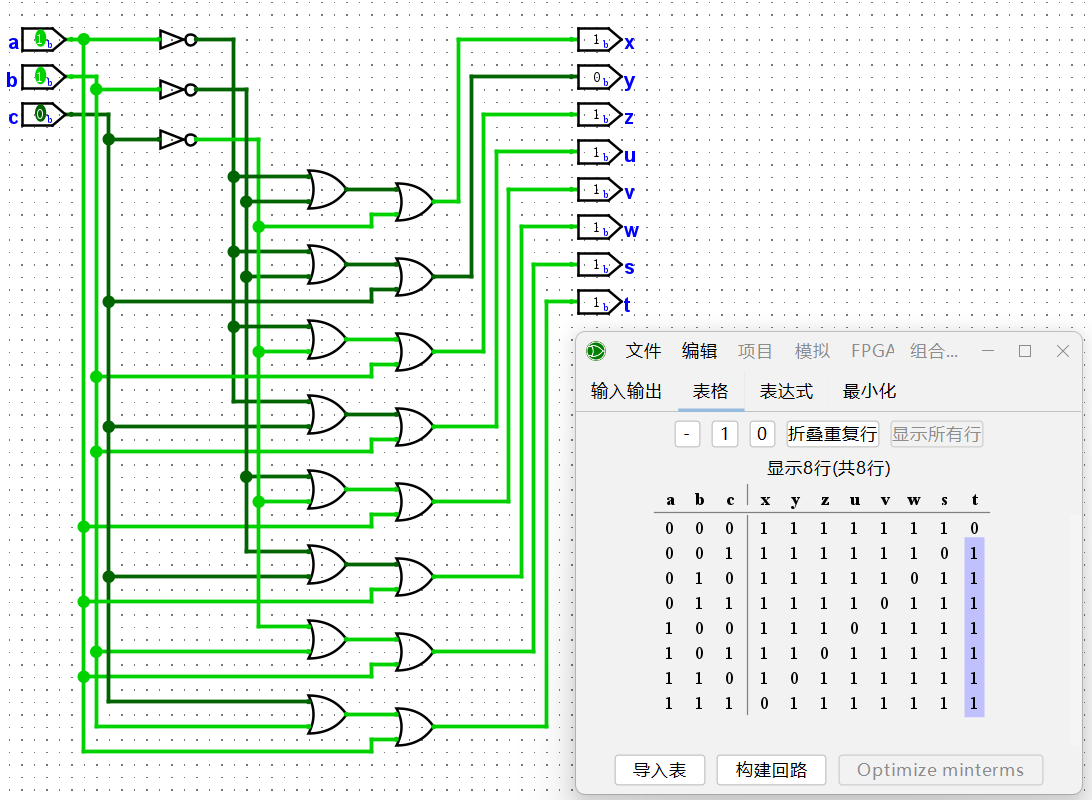
endmodule



当输入正确（即x为2的幂）时，输出y符合8-3编码器的特点。而当输入中有多个1时，由于具体实现的不同，输出y也不完全相同，如：assign方法按照逻辑门的连线输出，case方法则默认输出0；而优先编码器的casex与if方法则根据从左往右的第一个1的位数输出。

2）3-8译码器

Logisim实现



Vivado实现

代码：

module Decoder38(A, Y);

input [2:0] A;

output reg [7:0] Y;

integer i;

always @(A) begin

Y = 8'b1111\_1111;

for(i = 0; i <= 7; i = i + 1)

if(A == i)

Y[i] = 0;

else

Y[i] = 1;

end

endmodule

module sim\_Decoder38();

reg [2:0] x;

wire [7:0] y;

initial begin

x = 3'b000;

repeat(7) #20 x = x + 1;

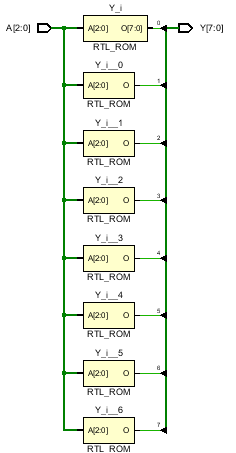
repeat(7) #20 x = x - 1;

end

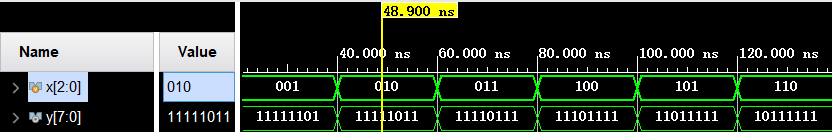
Decoder38 decoder38(x, y);

endmodule

电路图：



仿真波形：



输出y的第x位为0，其余各位均为1。

3）D锁存器

Vivado实现

代码：

module D\_latch(Q, QN, D, EN, RST);

output reg Q, QN;

input D;

input EN, RST;

always @(EN, RST, D) begin

if(RST) begin

Q = 0;

QN = 1;

end

else if(EN) begin

Q <= D;

QN <= ~D;

end

end

endmodule

module sim\_D\_latch\_1();

reg D, EN, RST;

wire Q, QN;

initial begin

D = 1; EN = 0; RST = 0;

fork

#10 RST = 1;

#50 RST = 0;

forever #20 D = ~D;

forever #55 EN = ~EN;

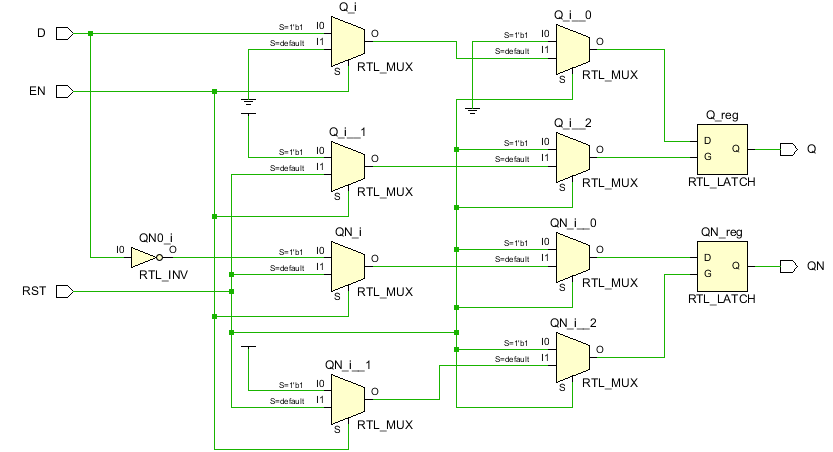
join

end

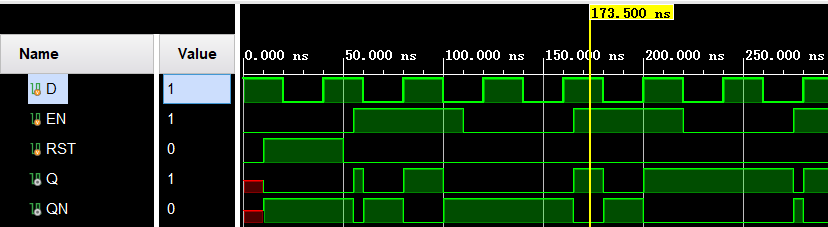
D\_latch D\_latch\_1(Q, QN, D, EN, RST);

endmodule

电路图：



仿真波形：



QN始终是Q的取反。当RST为1时，Q为0；当RST为0且使能端EN为1时，Q与D相同；其余情况下，Q维持不变。

4）D触发器

Vivado实现

代码：

module D\_ff(Q, QN, D, EN, RST, CLK);

output reg Q, QN;

input D;

input EN, RST, CLK;

always @(posedge CLK) begin

if(RST) begin

Q <= 1'b0;

QN <= 1'b1;

end

else if(EN) begin

Q <= D;

QN <= ~D;

end

end

endmodule

module sim\_D\_ff();

reg D, CLK, EN, RST;

wire Q, QN;

initial begin

D = 1; CLK = 0; EN = 0; RST = 0;

fork

forever #25 CLK = ~CLK;

forever #20 D = ~D;

forever #55 EN = ~EN;

#10 RST = 1;

#50 RST = 0;

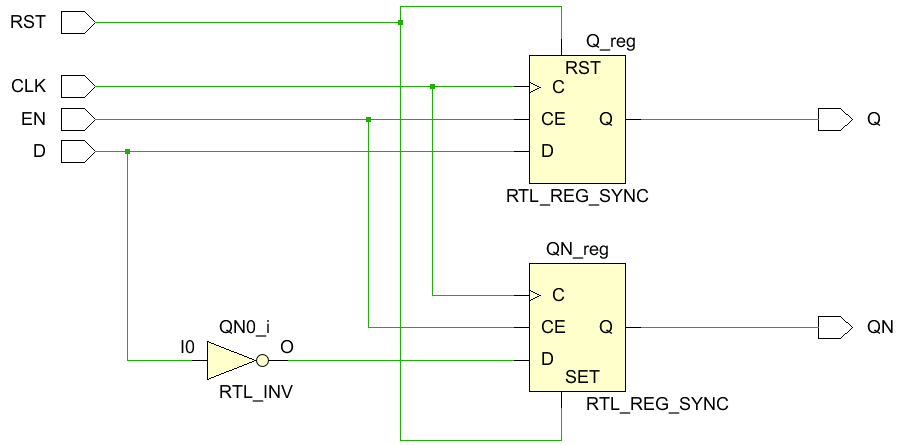
join

end

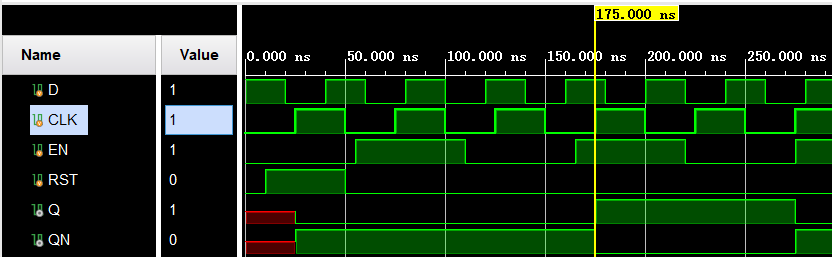
D\_ff D\_ff\_1(Q, QN, D, EN, RST, CLK);

endmodule

电路图：



仿真波形：



QN始终是Q的取反。当时钟信号CLK的上升沿到来时，Q才会相应改变。此时，当RST为1时，Q为0；当RST为0且使能端EN为1时，Q与D相同；其余情况下，Q维持不变。

5）寄存器

Vivado实现

代码：

module register(Q, D, OE, CLK);

parameter N = 8;

output reg [N-1:0] Q;

input [N:1] D;

input OE, CLK;

always @(posedge CLK or posedge OE)

if(OE) Q <= 8'bzzzz\_zzzz;

else Q <= D;

endmodule

module sim\_reg();

reg OE, CLK;

reg [8:1] D;

wire [8:1] Q;

initial begin

OE = 1; CLK = 0; D =0;

fork

#20 OE = 0;

forever #10 CLK = ~CLK;

forever #50 D = D + 1;

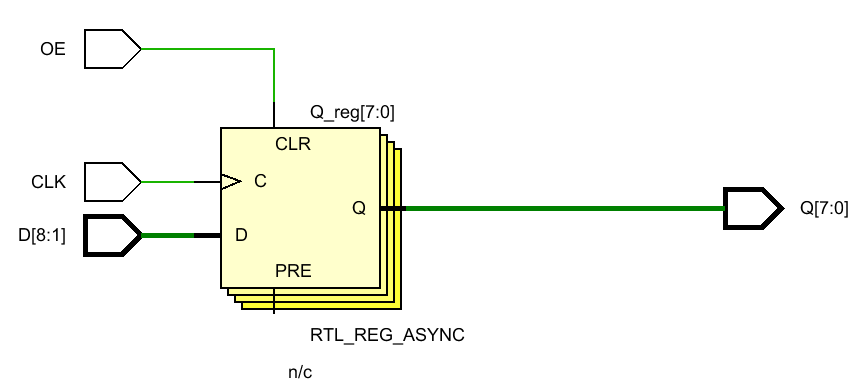
join

end

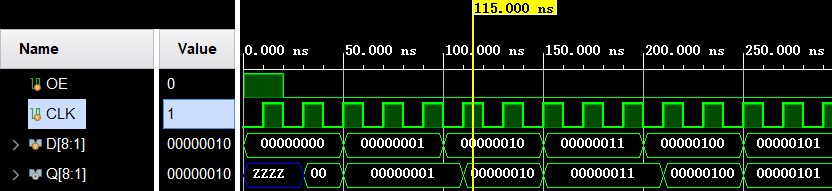
register reg\_1(Q, D, OE, CLK);

endmodule

电路图：



仿真波形：



当时钟信号CLK或者OE信号的上升沿到来时，若OE信号为1，则Q为高阻态；否则Q与D相同。

6）移位寄存器（循环移位）

Vivado实现

代码：

module cyclic\_shift(S, D, Q, CLK, CR);

parameter N = 4;

input [1:0] S;

input CLK, CR;

input [N-1:0] D;

output [N-1:0] Q;

reg [N-1:0] Q;

always @(posedge CLK or posedge CR)

if(CR)

Q <= 0;

else

case (S)

2'b00: Q <= Q;

2'b01: Q <= {Q[0], Q[N-1:1]};

2'b10: Q <= {Q[N-2:0], Q[N-1]};

2'b11: Q <= D;

endcase

endmodule

module sim\_cyclic\_shift();

reg CR, CLK;

reg [1:0] S;

reg [3:0] D;

wire [3:0] Q;

initial begin

CLK = 0; CR = 1; D = 4'b1011; S = 2'b11;

fork

forever #20 CLK = ~CLK;

#30 CR <= 0;

#100 S = 2'b01;

#400 S = 2'b10;

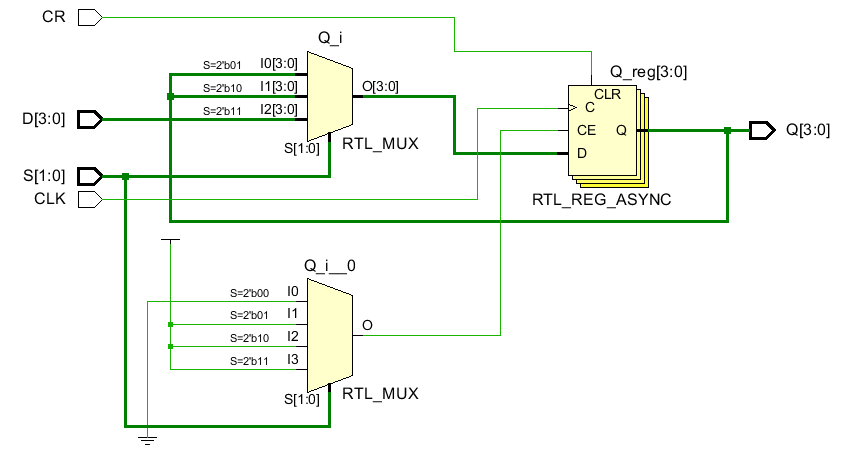
join

end

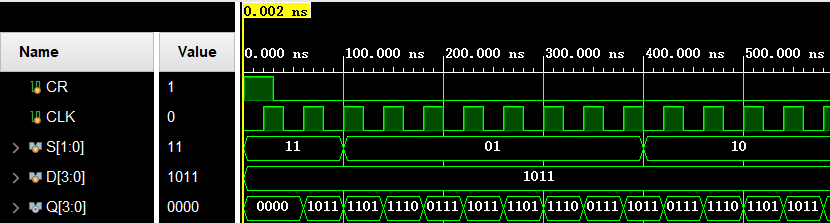
cyclic\_shift cyclic(S, D, Q, CLK, CR);

endmodule

电路图：



仿真波形：



当时钟信号CLK或者CR信号的上升沿到来时，若CR为1，Q清零。否则，若S为00，Q维持；若S为11，Q与D相同；若S为01，Q循环右移；若S为10，Q循环左移。

7）16进制计数器

Vivado实现

代码：

module counter74x161(CEP, CET, PE, CLK, CR, D, TC, Q);

parameter N = 8;

parameter M = 16;

input CEP, CET, PE, CLK, CR;

input [N-1:0] D;

output reg TC;

output reg [N-1:0] Q;

wire CE;

assign CE = CEP & CET;

always @(posedge CLK, negedge CR)

if(~CR) begin Q <= 0; TC <= 0; end

else if(~PE) Q <= D;

else if(CE) begin

if(Q == M-1) begin

TC <= 1;

Q <= 0;

end

else Q <= Q + 1;

end

else Q <= Q;

endmodule

module sim\_counter();

reg [7:0] D;

reg CLK, CEP, CET, PE, CR;

wire [7:0] Q;

wire TC;

initial begin

D = 4; CLK = 0; CEP = 0; CET = 0; PE = 1; CR = 1;

fork

forever #10 CLK = ~CLK;

#30 CEP = 1;

#30 CET = 1;

#20 CR = 0;

#30 CR = 1;

#30 PE = 0;

#40 PE = 1;

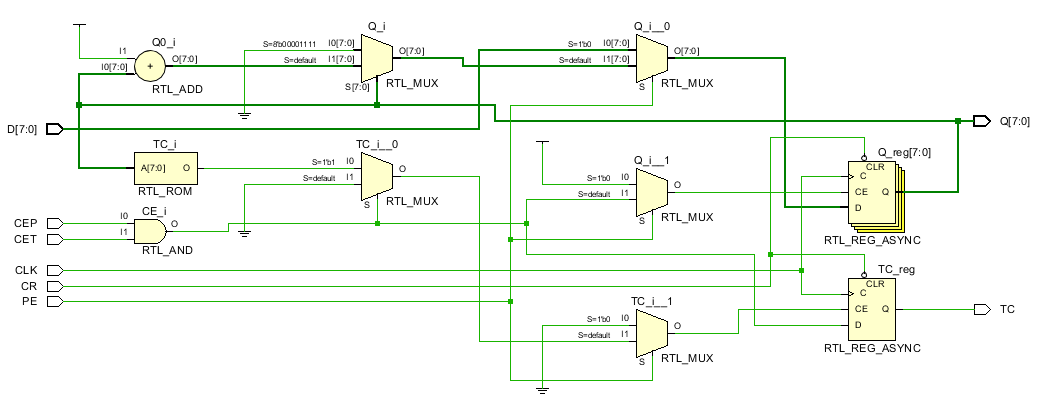
join

end

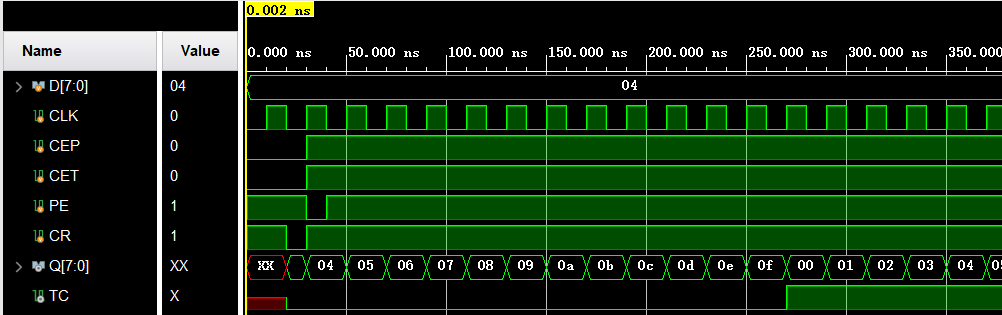
counter74x161 counter(CEP, CET, PE, CLK, CR, D, TC, Q);

endmodule

电路图：



仿真波形：



当CR为0时，Q异步清零；当PE为0时，Q同步装入输入数据D；当CEP与CET均为1时，开始计数，Q满16则进位TC为1，计数Q归零。

**五、调试和心得体会**

通过本次实验，我进一步掌握了两种开发工具Vivado与Logisim的使用，并熟悉了一些基本功能模块的程序设计，如：编码器、译码器、触发器、寄存器和计数器等。在实验过程中，我遇到的困难主要是仿真文件的编写，特别是控制时钟信号与使能信号的同步关系，经过这次实验，我对于输入输出之间的时序关系有了更深刻的认识与理解。