Part Ⅳ Verilog HDL 及其编程

Lecture 17 Verilog HDL 时序逻辑设计

一、触发器的设计

1、*R-S* 触发器

（1）基本 *R-S* 触发器

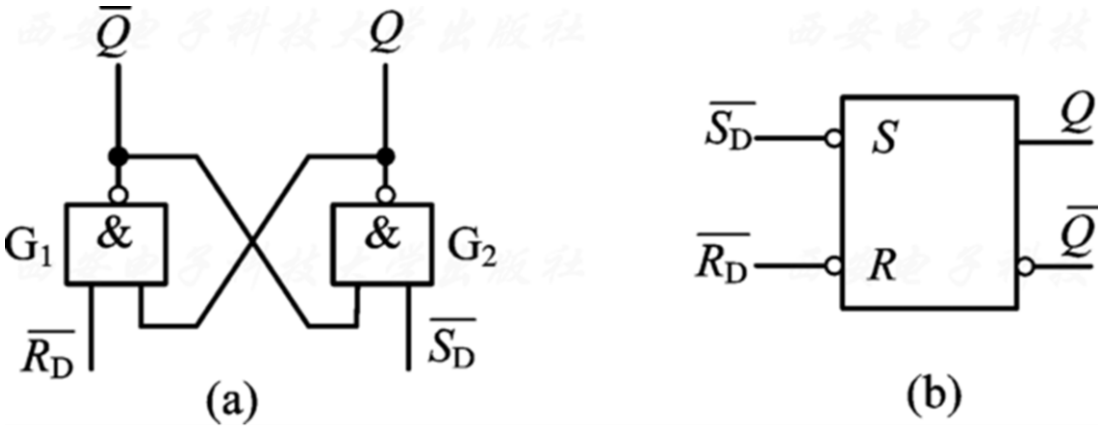

图5.2.1 与非门构成的基本R-S触发器

```
module BASICRS(R,S,Q,Qn);

    input R,S;
```

```
output Q,Qn;

wire Q1,Qn1;

wire Rd,Sd;

assign Q=Q1;

assign Qn=Qn1;

nand u1(Qn1,Q1,Rd);

nand u2(Q1,Qn1,Sd);
```

endmodule

（2）钟控 *R-S* 触发器

```
module BASIC_RS_CP(R, S,CP,Q,Qn);

    input R,S,CP;

    output Q,Qn;

    wire Q1,Qn1;
```

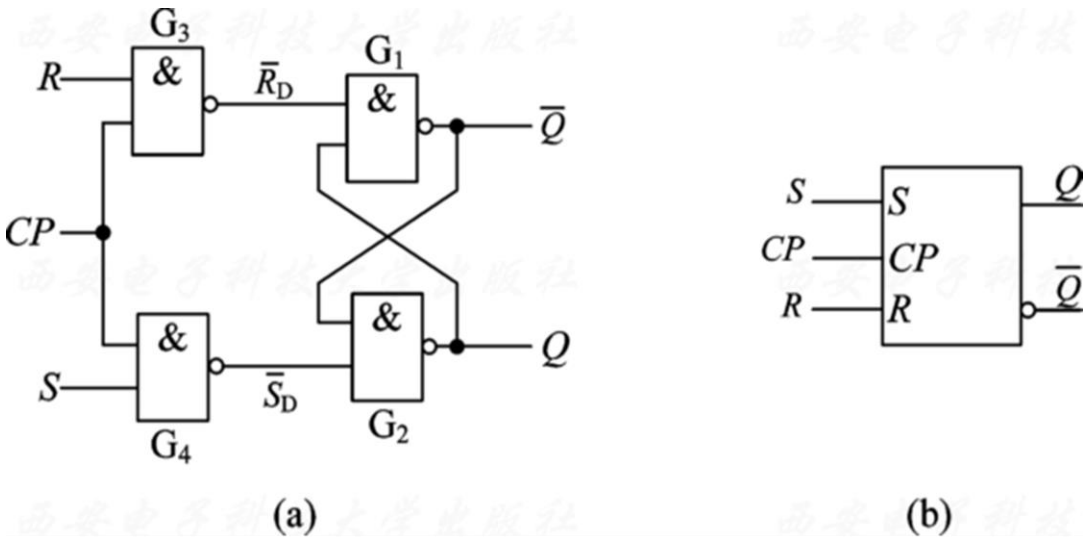

图5.2.7 钟控*R-S*触发器

(a) 电路图； (b) 逻辑符号

```
wire Rd,Sd;

assign Q=Q1;

assign Qn=Qn1;

nand u3(S,CP,Sd);

nand u4(R,CP,Rd);

nand u1(Sd,Qn1,Q1);

nand u2(Rd,Q1,Qn1);
```

```
endmodule
```

2、D 触发器

（1）基本功能 *D* 触发器

```
module BASIC_DFF_DN(D,CP,Q,QN);
    input D,CP;
    output Q,QN;
    reg Q;
```

```
    assign QN=~Q;
    always @(negedge CP)
        Q=D;
endmodule
```

（2）异步置位复位 *D* 触发器

```
module ASYNC_RS_D_FF (D,CP,R,S,Q,QN);
    input D,CP,R,S;
```

```
output Q,QN;

reg Q;

assign QN=~Q;

always @(posedge CP,posedge R,posedge S)

    if (R) Q<=1'b0;

    else if (S) Q<=1'b1;

    else Q<=D;
```

```
endmodule
```

（3）同步置位复位 *D* 触发器

```
module SYNC_RS_D_FF(D,CP,R,S,Q,QN);
    input D,CP,R,S;
    output Q,QN;
    reg Q;
    assign QN=~Q;
```

```
    always @(posedge CP)
        if (R) Q<=1'b0;
        else if (S) Q<=1'b1;
        else Q<=D;
endmodule
```

3、异步复位置位 J-K 触发器

```
module ASYNC_RS_JK_FF(J,K,CP,R,S,Q,QN);

    input J,K,CP,R,S;

    output Q,QN;

    reg Q;

    assign QN=~Q;

    always @(posedge CP,posedge R,posedge S)
```

```
if (R) Q<=1'b0

else if (S) Q<=1'b1;

else begin

        if (J==1&&K==1) Q<=~Q;

        else if (J==0&&K==1) Q<=1'b0;

        else if (J==1&&K==0) Q<=1'b1;

    end
```

```
endmodule
```

4、钟控 *T* 触发器

```
module T_FF(T,CP,Q,QN);
    input T,CP;
    output Q,QN;
    reg Q;
    assign QN=Q;
```

```
    always @(posedge CP)

        if (T) Q<=~Q

endmodule
```

二、时序逻辑电路设计的基本思路

从问题出发，画出正确的原始状态图，直接根据原始状态图进行行为描述。

或者对原始状态图进行化简，然后根据简化后的状态图进行行为描述。

当然，各状态需要先进行编码！

也可以进行结构描述、数据流描述等。

显然，行为描述效率高！

【例 1】设计一个“111”序列检测器，用来检测串行二进制序列，要求当连续输入三个或三个以上 1 时，检测器输出为 1，否则输出为 0。

【解答】原始状态图及状态表如下：

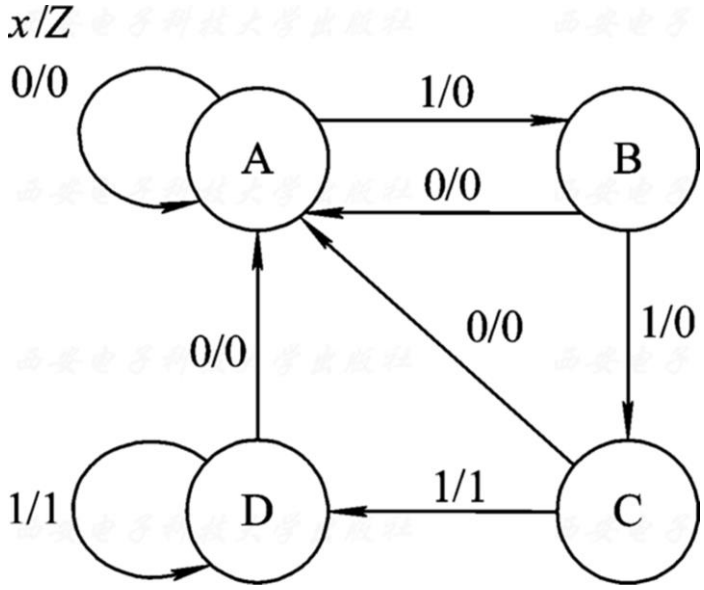

图5.4.29 例5.4.9的原始状态转移图

表5.4.15 例5.4.9的原始状态转移真值表

现态	次态/输出 Z	
	$x=0$	$x=1$
A	A/0	B/0
B	A/0	C/0
C	A/0	D/1
D	A/0	D/1

```
module Mealy_samp3(clock,resetn,x,z);

    input clock,resetn,x;

    output reg z;
```

```
reg [1:0] y;

parameter [1:0] A=2'b00,B=2'b01;

parameter [1:0] C=2'b10,D=2'b11;

always @(negedge resetn,posedge clock)

    if (~resetn) y<=A;

    else case (y)

                A:if (x) begin z<=0;y<=B;end
```

else begin z<=0;y<=A;end

B:if (x) begin z<=0;y<=C;end

else begin z<=0;y<=A;end

C:if (x) begin z<=1;y<=D;end

else begin z<=0;y<=A;end

D:if (x) begin z<=1;y<=D;end

else begin z<=0;y<=A;end

endcase

endmodule

简化状态图如下：

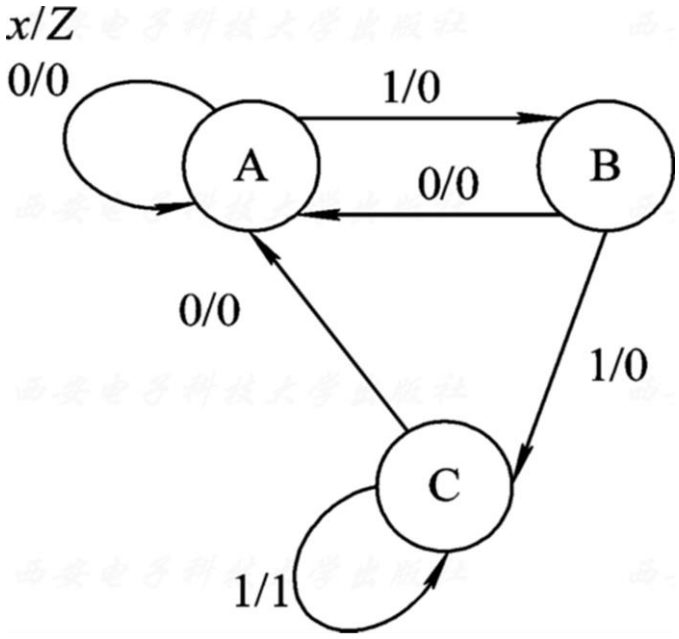

图5.4.30 例5.4.9化简后的状态转移图

第一种代码：

module

```
Mealy_samp1(clock,resetn,x,z);

    input clock,resetn,x;

    output reg z;

    reg [1:0] y;

    parameter [1:0] A=2'b00,B=2'b01,C=2'b10;

    always @(negedge resetn,posedge clock)

        if (~resetn) y<=A;
```

```
else case (y)

        A:if (x) begin y<=B;z<=0;end

            else begin y<=A;z<=0;end

        B:if (x) begin y<=C;z<=0;end

            else begin y<=A;z<=0;end

        C:if (x) begin y<=C;z<=1;end

            else begin y<=A;z<=0;end
```

```
                default:begin y<=A;z<=0;end

            endcase

endmodule
```

第二种代码：

```
module Mealy_samp2(clock,resetn,x,z);

    input clock,resetn,x;
```

```
output reg z;

reg [1:0] y;

parameter [1:0] A=2'b00,B=2'b01,C=2'b10;

always @(negedge resetn,posedge clock)

    if (~resetn) y<=A;

    else begin

                case (y)
```

```
        A:if (x) y<=B; else y<=A;

        B:if (x) y<=C; else y<=A;

        C:if (x) y<=C; else y<=A;

        default: y<=A;

    endcase

    z=(y==C&&x);

end
```

```
endmodule
```

第三种代码：最符合 Mealy 型的特点

```
module Mealy_samp1(clock,resetn,x,z);
    input clock,resetn,x;
    output reg z;
    reg [1:0] y,Y;
```

```
parameter [1:0] A=2'b00,B=2'b01,C=2'b10;

always @(y,x)    //与时钟无关

    case (y)

        A:if (x) begin y<=B;z<=0;end

          else begin y<=A;z<=0;end

        B:if (x) begin y<=C;z<=0;end

          else begin y<=A;z<=0;end
```

```
            C:if (x) begin y<=C;z<=1;end

              else begin y<=A;z<=0;end

            default:begin y<=A;z<=0;end

        endcase

    always @(negedge resetn,posedge clock)

        if (~resetn) Y<=A; else Y<=y;

endmodule
```

第四种代码：最符合 Mealy 型的特点，更简洁

```
module Mealy_samp1(clock,resetn,x,z);
    input clock,resetn,x;
    output z;
    reg [1:0] y;
    parameter [1:0] A=2’b00,B=2’b01,C=2’b10;
```

```
always @(negedge resetn,posedge clock)
    if (~resetn) Y<=A;
    else case (y)
            A:if (x) y<=B; else y<=A;
            B:if (x) y<=C; else y<=A;
            C:if (x) y<=C; else y<=A;
            default: y<=A;
```

```
        endcase

    assign z=(y==C&&x);

endmodule
```

【例 2】用 Verilog HDL 描述下图所示电路：

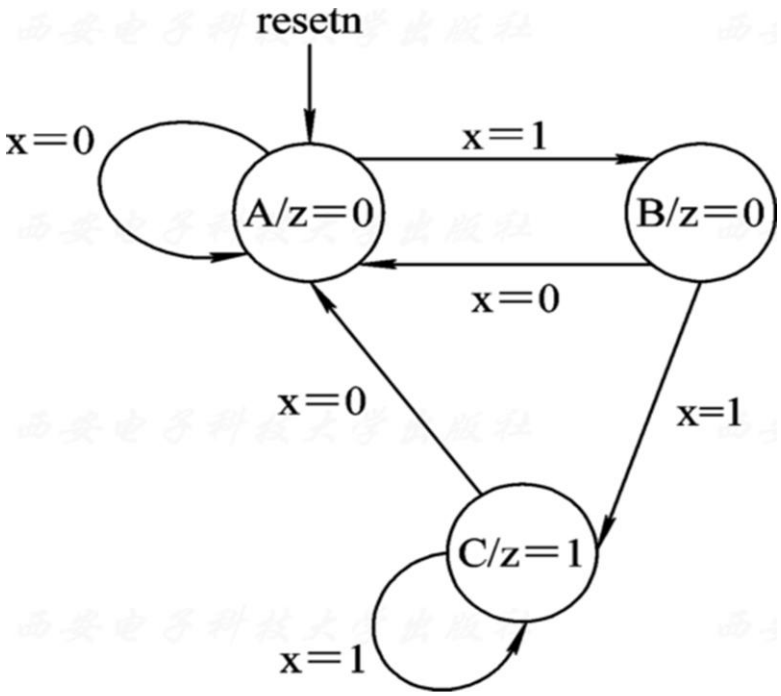

图5.4.33 例5.4.10的状态转移图

图中：A、B、C 表示三个

有效状态；clk 是同步时钟；x 是输入信号并控制状态的转移；resetn 是复位控制信号；z 是输出信号。

状态的转移只能在同步时钟信号 clk 的上升沿发生。

```
module Moore_samp(clock,resetn,x,z);

    input clock,resetn,x;
```

```
output z;

reg [2:0] y;

parameter [2:0] A=3'b001,B=3'b010;

parameter [2:0] C=3'b100;

always @(negedge resetn,posedge clock)

if (~resetn) y<=A;

else case (y)
```

```
            A:if (x) y<=B; else y<=A;

            B:if (x) y<=C; else y<=A;

            C:if (x) y<=C; else y<=A;

            default:y<=A;

        endcase

    assign z=(y==C);

endmodule
```

三、常用时序逻辑电路的设计

1、计数器

（1）基本同步计数器

```
Module counter_basic(cp,q,co);

    Parameter msb=3;
```

```
Input cp;

Output reg[msb:0] q;

Output reg co;

Always @(posedge cp)

    Begin

        If (q==4'b1110) co<=1;

        Else co<=0;
```

```
        Q<=q+1'b1;

    End

endmodule
```

（2）同步复位计数器

```
Module counter_basic(cp,r,q,co);

    Parameter msb=3;

    Input cp,r;
```

```
Output reg[msb:0] q;

Output reg co;

Always @(posedge cp)

    If (r) Begin

            q<=0;

            co<=0;

        end
```

```
        else Begin
                If (q==4'b1110) co<=1;
                Else co<=0;
                Q<=q+1'b1;
            End
endmodule
```

（3）异步复位计数器

```
Module counter_basic(cp,r,q,co);

    Parameter msb=3;

    Input cp,r;

    Output reg[msb:0] q;

    Output reg co;

    Always @(posedge cp,posedge r)

        If (r) Begin
```

```
        q<=0;

        co<=0;

    end

else Begin

        If (q==4'b1110) co<=1;

        Else co<=0;

        Q<=q+1'b1;
```

End

endmodule

（4）同步置数计数器

```
Module counter_basic(cp,r,s,d,q,co);
    Parameter msb=3;
    Input cp,r,s;
    Input[msb:0] d;
```

```
Output reg[msb:0] q;

Output reg co;

Always @(posedge cp)

    If (r) Begin

            q<=0;

            co<=0;

        end
```

```
else if (s) begin
        q<=d;
    end
else Begin
        If (q==4'b1110) co<=1;
        Else co<=0;
        Q<=q+1'b1;
```

End

endmodule

（5）计数控制计数器

（6）加减同步计数器

2、寄存器

（1）锁存寄存器

```
Module latch(clk,lock,in,out);
```

```
Input clk;

Input lock;

Input[7:0] in;

Output reg[7:0] out;

always@(posedge clk)

    if (lock) out<=in;

endmodule
```

（2）缓冲寄存器

```
Module latch(clk,OE,in,out);

Input clk;

Input OE;

Input[7:0] in;

Output[7:0] out;

reg[7:0] R;
```

```
assign out=OE?R:8'bzzzzzzzz;

always@(posedge clk) R<=in;

endmodule
```

（3）锁存、缓冲寄存器

```
Module latch(clk,lock,OE,in,out);

Input clk;

Input lock;
```

```
Input OE;

Input[7:0] in;

Output[7:0] out;

reg[7:0] R;

always@(posedge clk)

    if (lock) R<=in;

assign out=OE?R:8'bzzzzzzzz;
```

```
endmodule
```

3、移位寄存器

（1）右移串入、串出移位寄存器

```
Module shift(clk,in,out);

Input clk;

Input in;

Output out;
```

```
Reg[7:0] R;

Assign out=R[7];

always@(posedge clk) R<={R[6:0],in};

endmodule
```

（2）右移串入、并出移位寄存器

```
Module shift(clk,in,out);
```

```
Input clk;

Input in;

Output[7:0] out;

Reg[7:0] R;

Assign out=R;

always@(posedge clk) R<={R[6:0],in};

endmodule
```

（3）并入、右移串出移位寄存器