

# 时序逻辑的Verilog描述

高翠芸

School of Computer Science

gaocuiyun@hit.edu.cn

# 时序逻辑的Verilog描述

---

- 时序逻辑电路在逻辑功能上的特点是任意时刻的输出不仅取决于当时的输入信号，而且还取决于电路原来的状态。
- 时序逻辑电路的变化通过时钟沿触发，需要使用时钟沿触发的always块描述。
- 用always块描述时序逻辑电路时，用非阻塞赋值。

# 时钟沿触发的always块描述

---

```
always @ (<敏感信号列表>)  
begin  
    //过程赋值  
    //if-else、case选择语句  
    //for、while等循环块  
end
```

**边沿触发：**即当时钟处在上升沿或下降沿时，语句被执行。

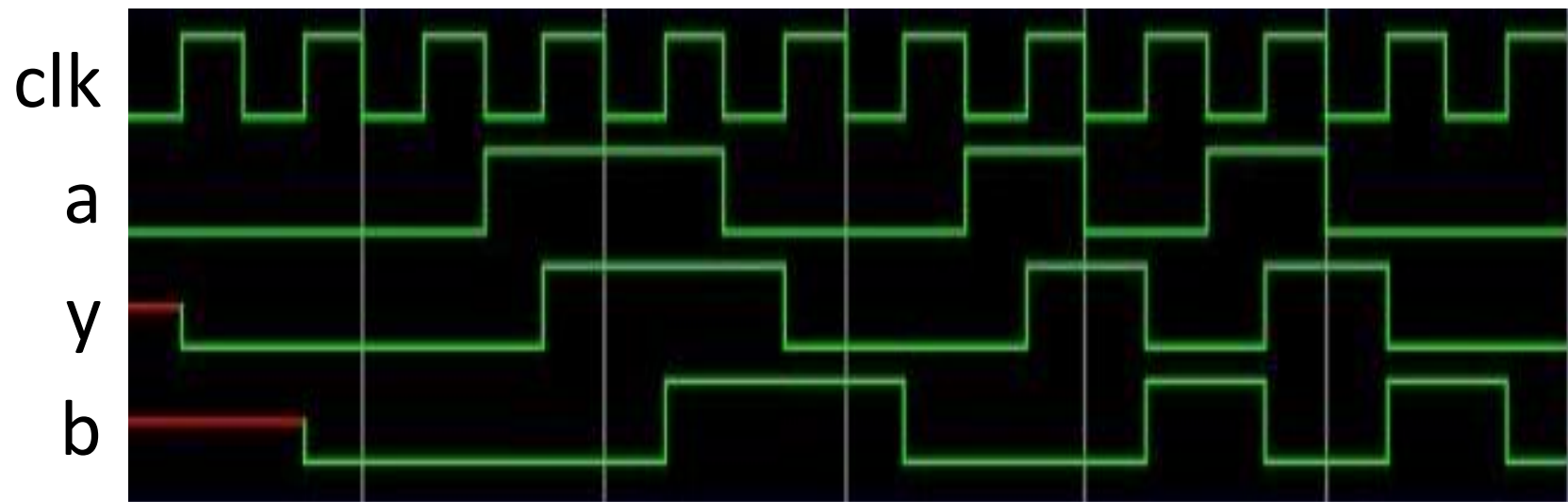
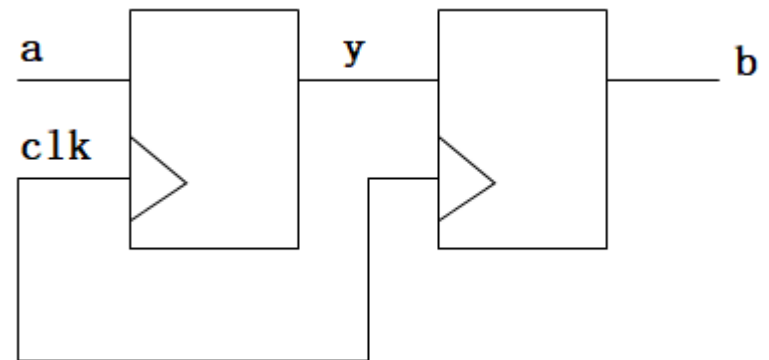
always @( **posedge** clk ) 时钟上升沿触发

always @( **negedge** clk ) 时钟下降沿触发

always @( **posedge** clk or **negedge** rst\_n ) 带异步复位的时钟上升沿触发

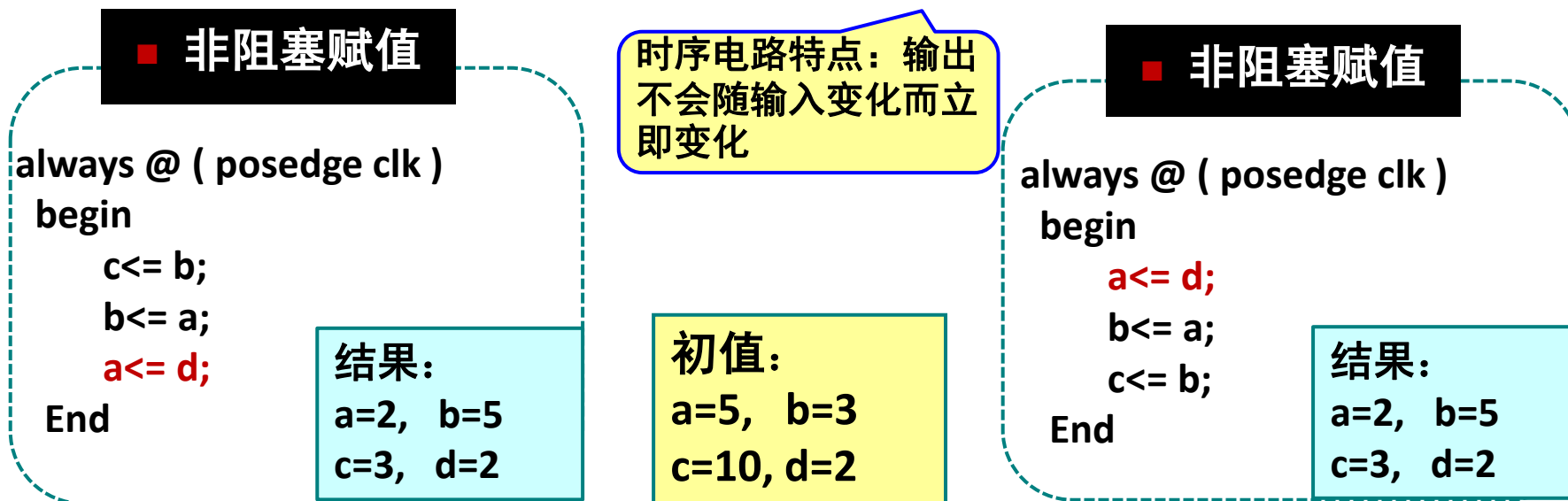
# 非阻塞赋值示例

```
module nonbloc (clk, a, b);  
  input clk, a;  
  output b; reg b;  
  reg y;  
  always @(posedge clk)  
  begin  
    y<=a;  
    b<=y;  
  end  
endmodule
```



# 非阻塞赋值

块内的赋值语句同时进行：先同时采样，最后一起更新

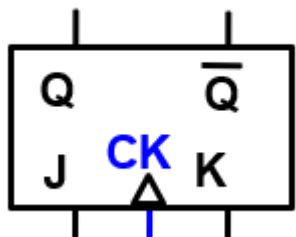


结果与书写的顺序无关（原因：同步更新）

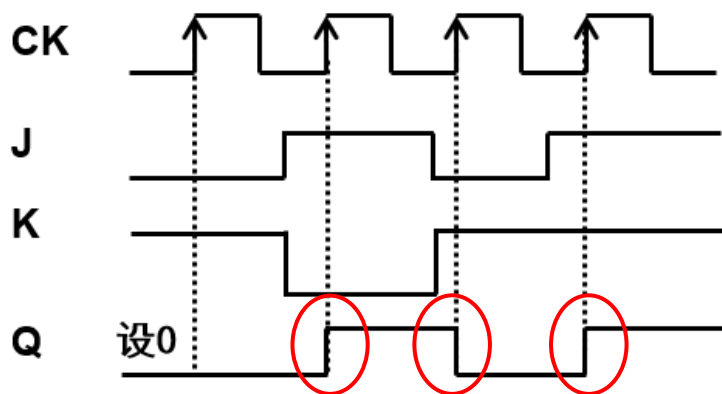
本质上，在一个时钟沿触发里，a得到d的值，但b得到的永远是a的旧值，c得到的永远是b的旧值（原因：同步更新）。

# JK触发器的Verilog描述

- 时序逻辑电路的变化通过时钟沿触发，需要使用**时钟沿触发**的always块描述。
- 用always块描述时序逻辑电路时，用**非阻塞赋值**。



$$Q_{n+1} = J \bar{Q}_n + \bar{K} Q_n$$

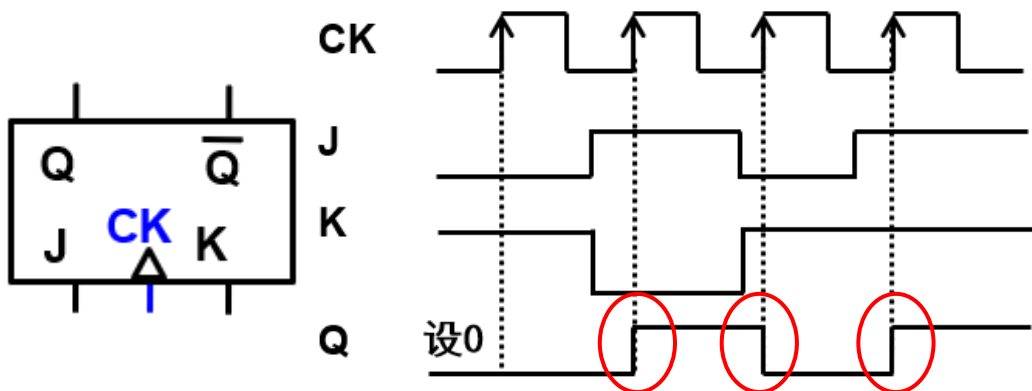


```
module J_K (  
    input    wire clk,  
    input    wire j,  
    input    wire k,  
    input    wire rst,  
    output   reg  Q  
);  
  
    wire rst_n;  
  
    assign rst_n=~rst;  
  
    always@ (posedge clk or negedge rst_n )begin  
        if (~rst_n)  
            Q <= 1'b0;  
        else  
            Q <= j && (~Q) || (~k) && Q;  
        end  
  
endmodule
```

$$Q_{n+1} = J \bar{Q}_n + \bar{K} Q_n$$

# JK触发器的Verilog描述

- 时序逻辑电路的变化通过时钟沿触发，需要使用**时钟沿触发**的always块描述。
- 用always块描述时序逻辑电路时，用**非阻塞赋值**。

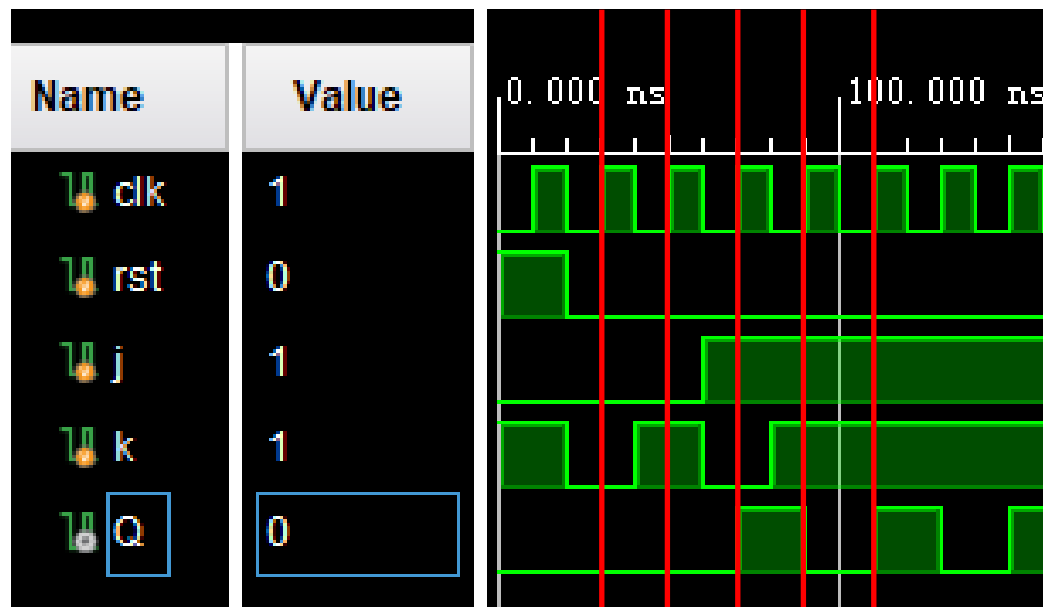


| 输入端 |   | 次态          |
|-----|---|-------------|
| J   | K | $Q_{n+1}$   |
| 0   | 0 | $Q_n$       |
| 0   | 1 | 0           |
| 1   | 0 | 1           |
| 1   | 1 | $\bar{Q}_n$ |

```
module J_K (  
    input wire clk,  
    input wire j,  
    input wire k,  
    input wire rst,  
    output reg Q  
);  
wire rst_n;  
assign rst_n=~rst;  
  
always@( posedge clk or negedge rst_n )begin  
    if (~rst_n)  
        Q <= 1'b0;  
    else  
        case({j,k})  
            2'b00: Q <= Q;  
            2'b01: Q <= 1'b0;  
            2'b10: Q <= 1'b1;  
            2'b11: Q <= ~Q;  
            default: Q <= Q;  
        endcase  
    end  
endmodule
```

//如果{j,k}=00，则触发器处于保持状态  
//如果{j,k}=01，则触发器置0  
//同理10，则触发器置1  
//11，翻转

# Testbench-时序逻辑



| 输入端 |   | 次态               |
|-----|---|------------------|
| J   | K | $Q_{n+1}$        |
| 0   | 0 | $Q_n$            |
| 0   | 1 | 0                |
| 1   | 0 | 1                |
| 1   | 1 | $\overline{Q_n}$ |

```
`timescale 1ns/1ps

module J_K_sim();

    reg    clk;
    reg    rst;
    reg    j,k;
    wire   Q;

    J_K u_J_K(.clk(clk),.rst(rst),.j(j),.k(k),.Q(Q));

    initial begin
        clk=1'b0;j=1'b0;k=1'b1;rst=1'b1;
        #20 rst=1'b0;j=1'b0;k=1'b0;
        #20 j=1'b0;k=1'b1;
        #20 j=1'b1;k=1'b0;
        #20 j=1'b1;k=1'b1;
    end

    always #10 clk=~clk;

endmodule
```



# 用Verilog实现D触发器

---

```
module VrDff(input CLK, input D, output reg Q);  
    always @ (posedge CLK)  
        Q <= D;  
endmodule
```

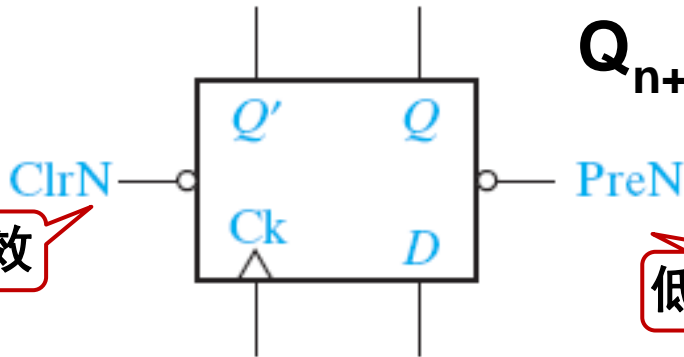
# 带附加输入端的边沿触发器

## 带异步清零端和异步置1端

异步：独立于时钟信号

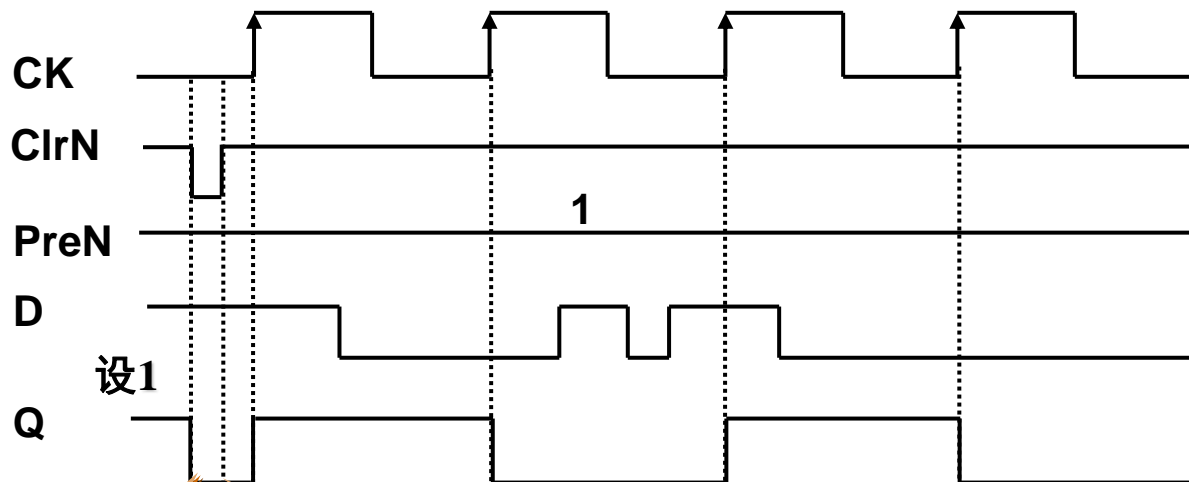
用途：为触发器设置指定状态

$$Q_{n+1} = D$$



低电平有效

低电平有效



清0

保持

| 时钟端<br>CK | 输入端<br>D | 异步置1端<br>PreN | 异步清零端<br>ClrN | 次态<br>$Q_{n+1}$ |
|-----------|----------|---------------|---------------|-----------------|
| X         | X        | 0             | 0             | 不允许             |
| X         | X        | 0             | 1             | 1               |
| X         | X        | 1             | 0             | 0               |
| ↑         | 0        | 1             | 1             | 0               |
| ↑         | 1        | 1             | 1             | 1               |
| 0,1, ↓    | X        | 1             | 1             | $Q_n$           |

- 被用于将触发器设置成**初始态**，**而不依赖于时钟信号**。

# 用Verilog实现带异步清零端的D触发器

---

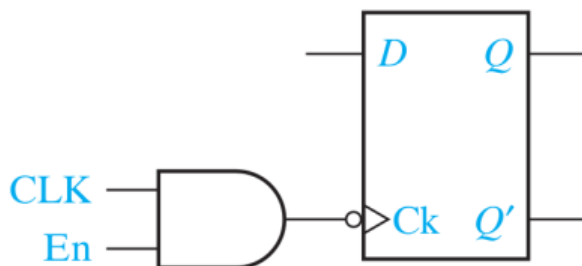
```
module VrDffC(input CLK, input CLR, input D,  
    output reg Q);  
    always @ (posedge CLK or posedge CLR)  
        if (CLR==1) Q <= 0;  
        else Q <= D;  
endmodule
```

# 带附加输入端的边沿触发器—续

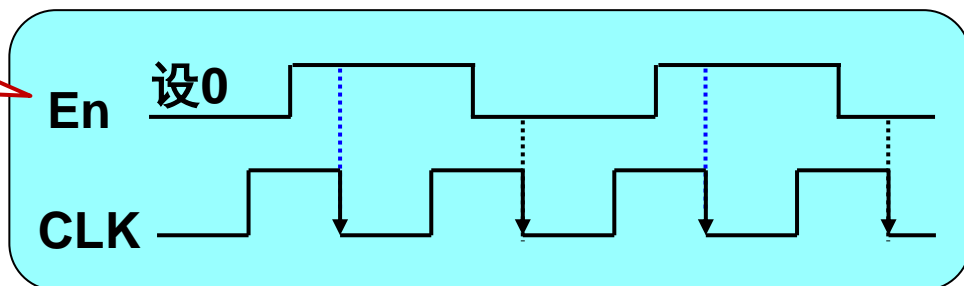
## ■ 带时钟使能端

解决方案:

使能端CE不与时钟端CLK捆绑使用

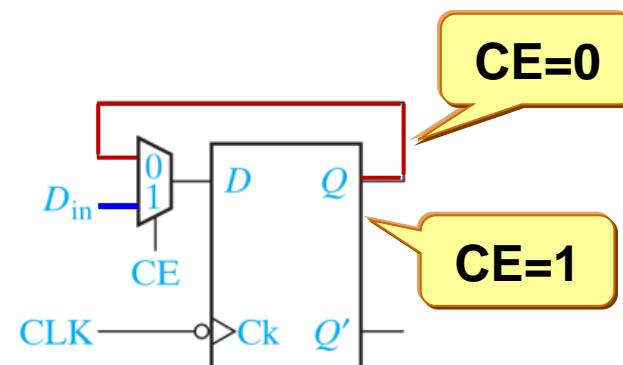
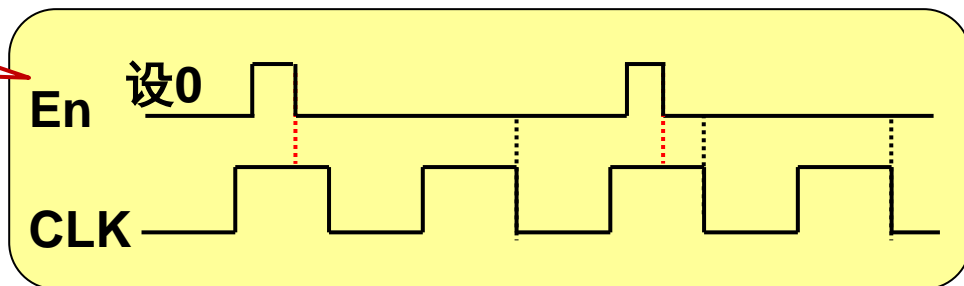


信号给定正确



后果: 失去同步性!

信号给定错误



CE=0

CE=1

$$Q^+ = Q \cdot CE' + D \cdot CE$$

# 带时钟使能端和同步置位D触发器

```
module VrDffSE(input CLK, input S, input CE,  
               input D, output reg Q);
```

```
always @ (posedge CLK)
```

```
    if (S==1) Q <= 1;
```

```
    else if (CE==1) Q <= D;
```

```
    else Q <= Q;
```

```
endmodule
```

组合逻辑（阻塞赋值描述）和时序逻辑（非阻塞赋值描述）的差别就是在这里，组合逻辑左右相等，形成的是环路，时序逻辑实际上是等于前一个状态的值，就不存在环路