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。

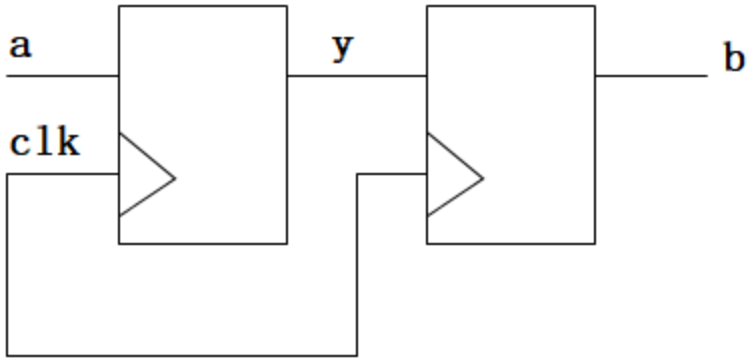
always @(posedge clk) 时钟上升沿触发

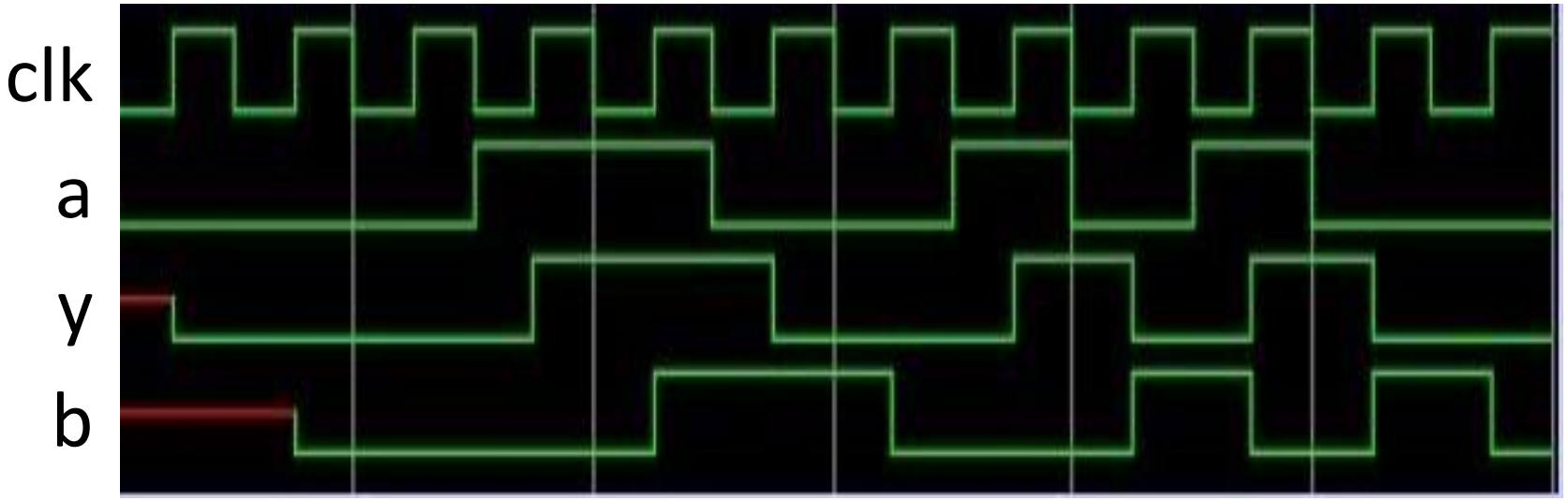
always @(negedge clk) 时钟下降沿触发

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

非阻塞赋值

```
always @ ( posedge clk )
 begin
      c<= b;
      b<= a;
      a<= d;
 End
```

结果：
a=2, b=5
c=3, d=2

结果与书写的顺序无关（原因：同步更新）

时序电路特点：输出不会随输入变化而立即变化

初值：
a=5, b=3
c=10, d=2

非阻塞赋值

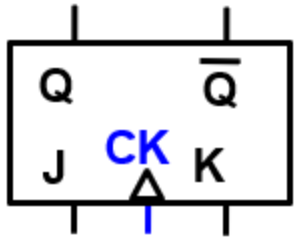
```
always @ ( posedge clk )
 begin
      a<= d;
      b<= a;
      c<= b;
 End
```

结果：
a=2, b=5
c=3, d=2

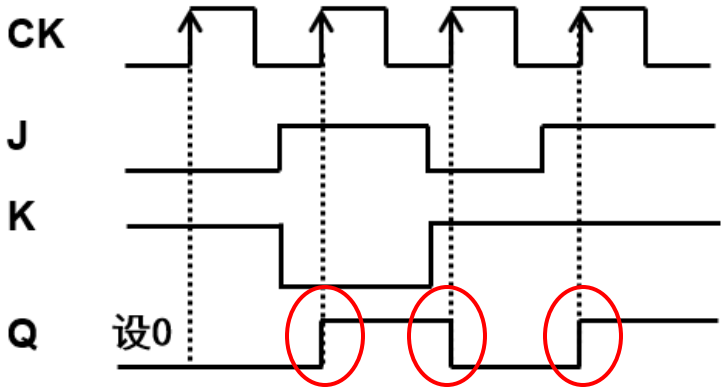
本质上，在一个时钟沿触发里，a得到d 的值，但b得到的永远是a的旧值，c得到的永远是b的旧值（原因：同步更新）。

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。



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

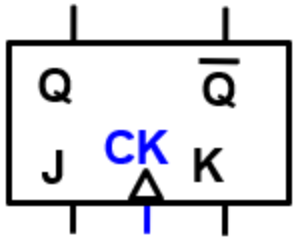


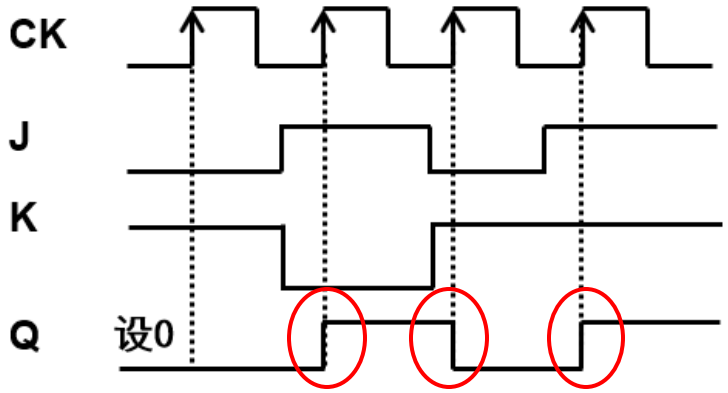

```
module J_K (
    input    wire clk,
    input    wire j,
    input    wire k,
    input    wire rst,
    output   reg  Q
);

    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
```

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

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。

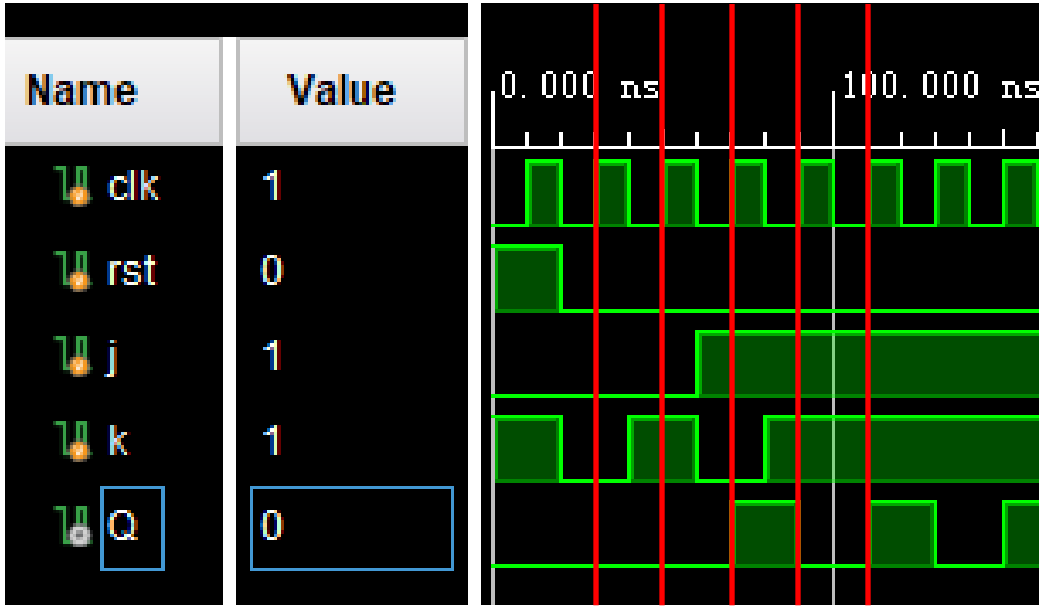




输入端 J	输入端 K	次态 Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q}_n$

```
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;          //如果{j,k}=00，则触发器处于保持状态
                2'b01: Q <= 1'b0;       //如果{j,k}=01，则触发器置0
                2'b10: Q <= 1'b1;       //同理10，则触发器置1
                2'b11: Q <= ~Q;         //11,翻转
                default: Q <= Q;
            endcase
    end
endmodule
```

Testbench-时序逻辑



输入端 J	输入端 K	次态 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
```