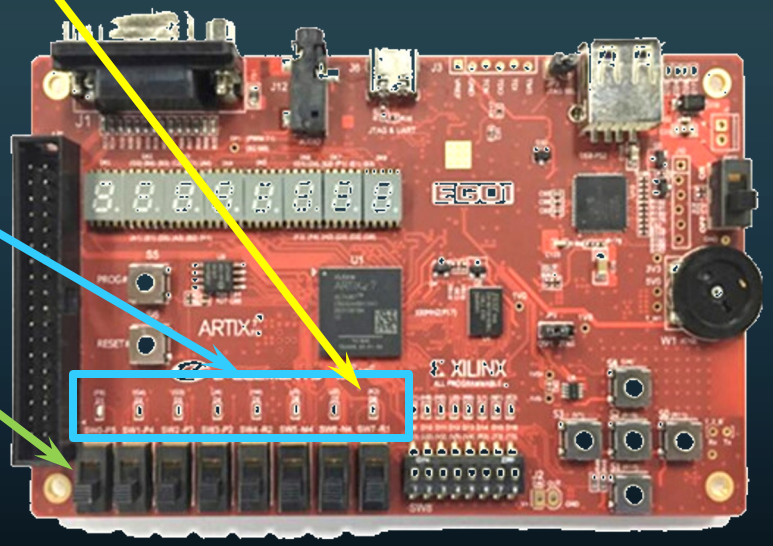
流水灯实验

纽带：Verilog编程

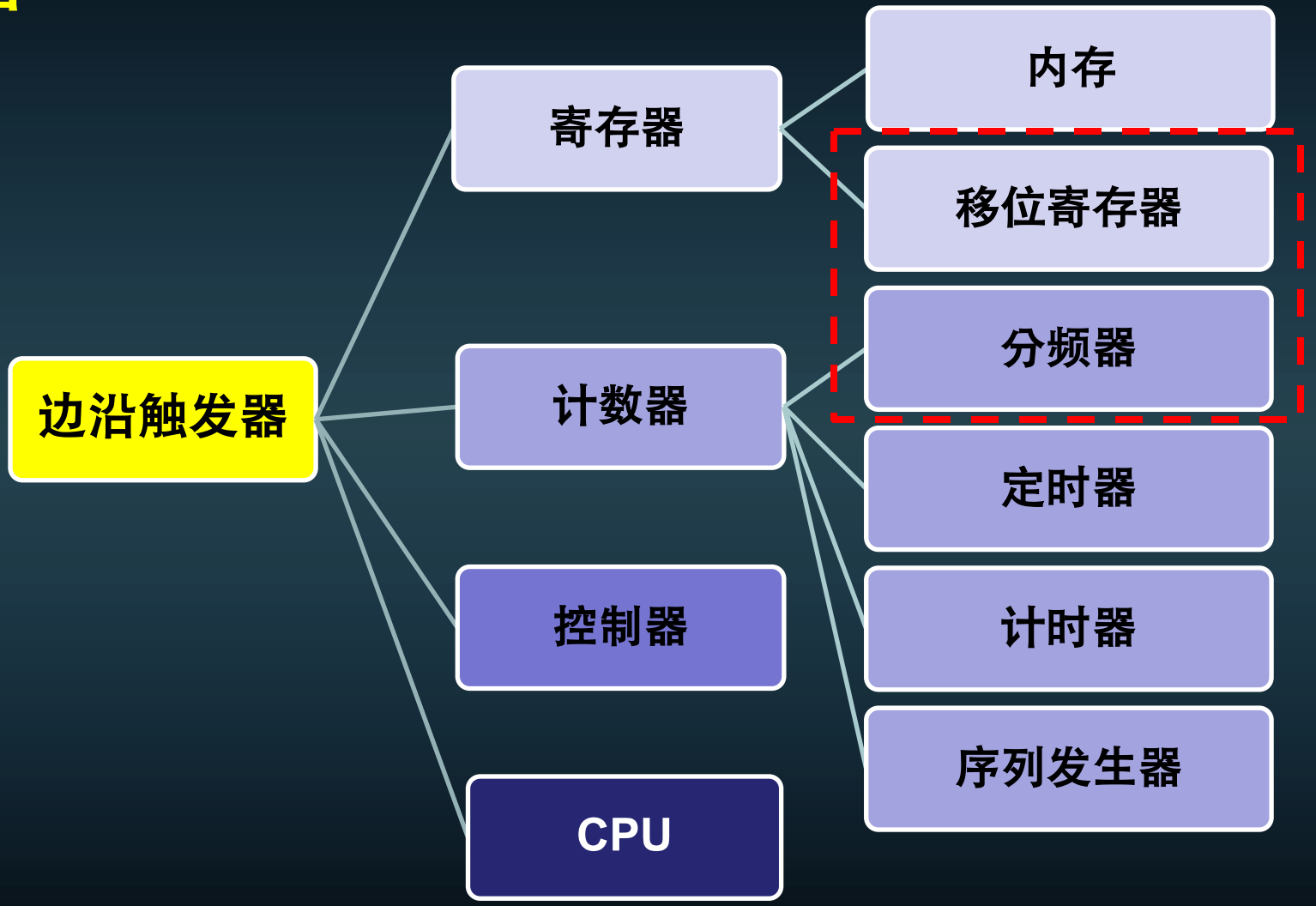
实 验 要 求

- 设计并实现一个流水灯数字控制器，要求：
  - 8个发光二极管间隔0.5s轮流点亮
  - 设置一个reset按钮，复位时，最右边灯亮，其余灯灭
  - 流水方向自右向左
  - 主频100Hz
  - 流水灯闪烁频率2Hz

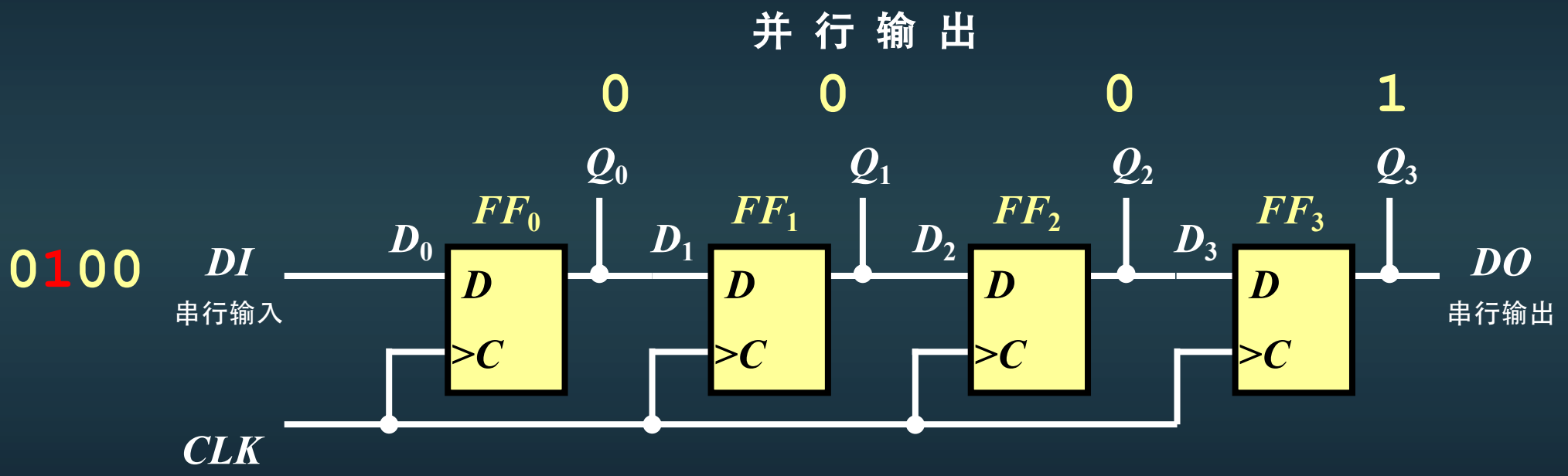


知识图谱

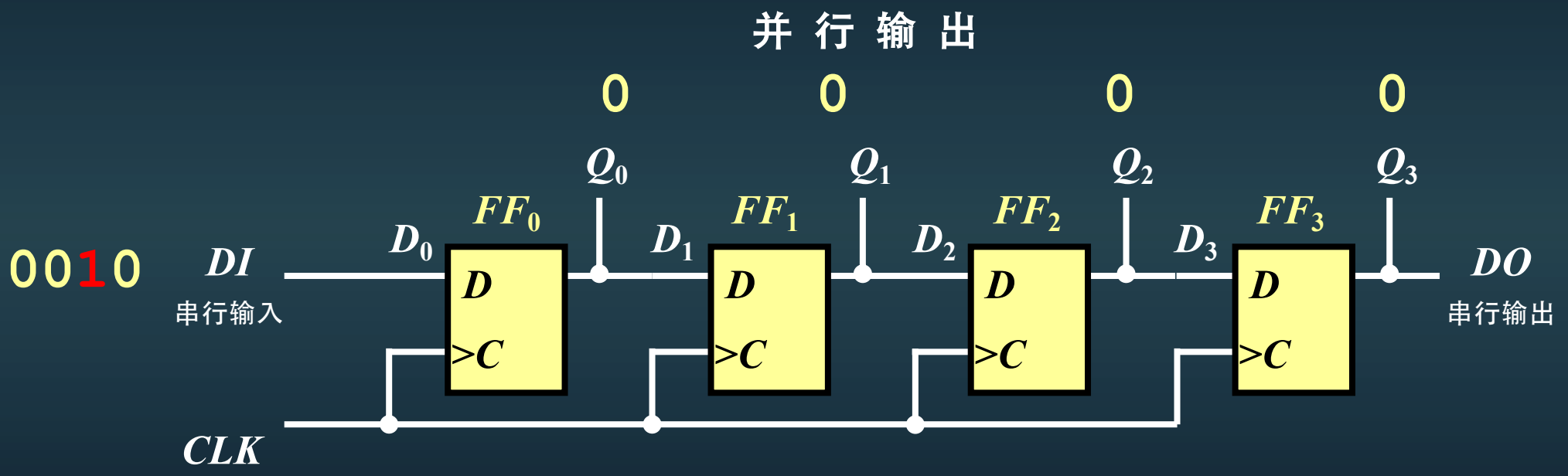
边沿触发器的应用



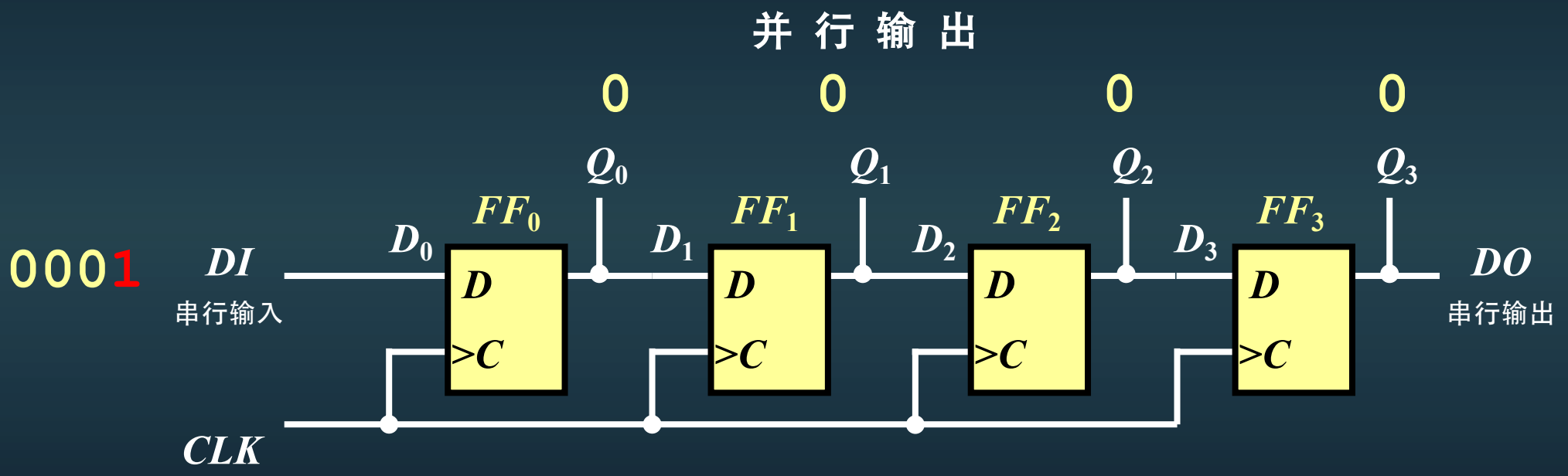
移位寄存器$Q_3Q_2Q_1Q_0$



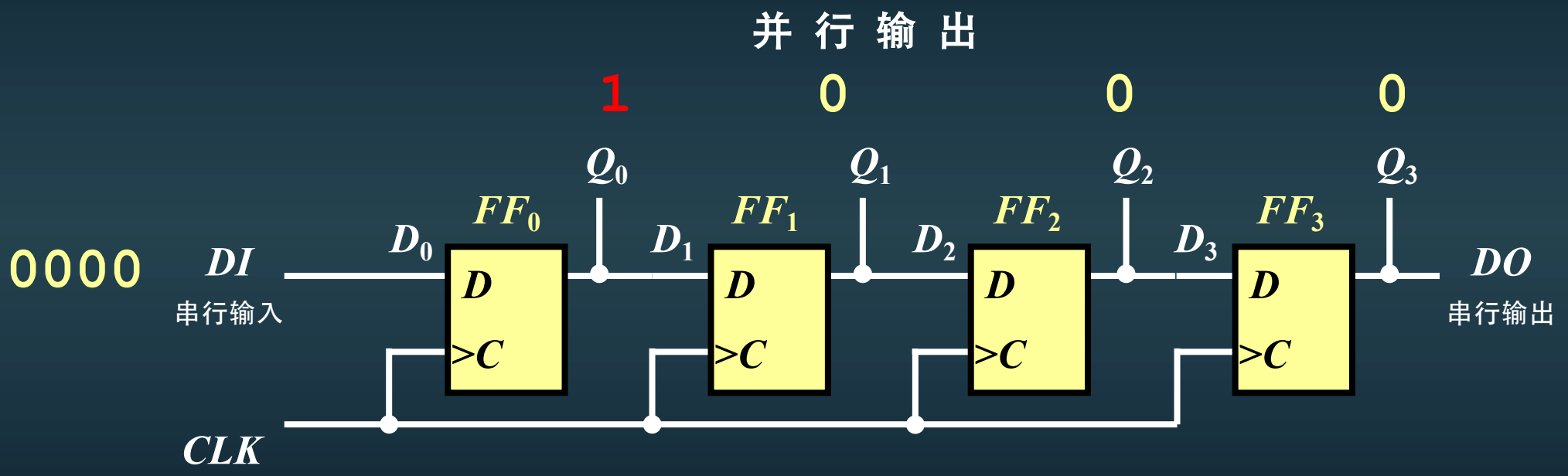
移位寄存器$Q_3Q_2Q_1Q_0$



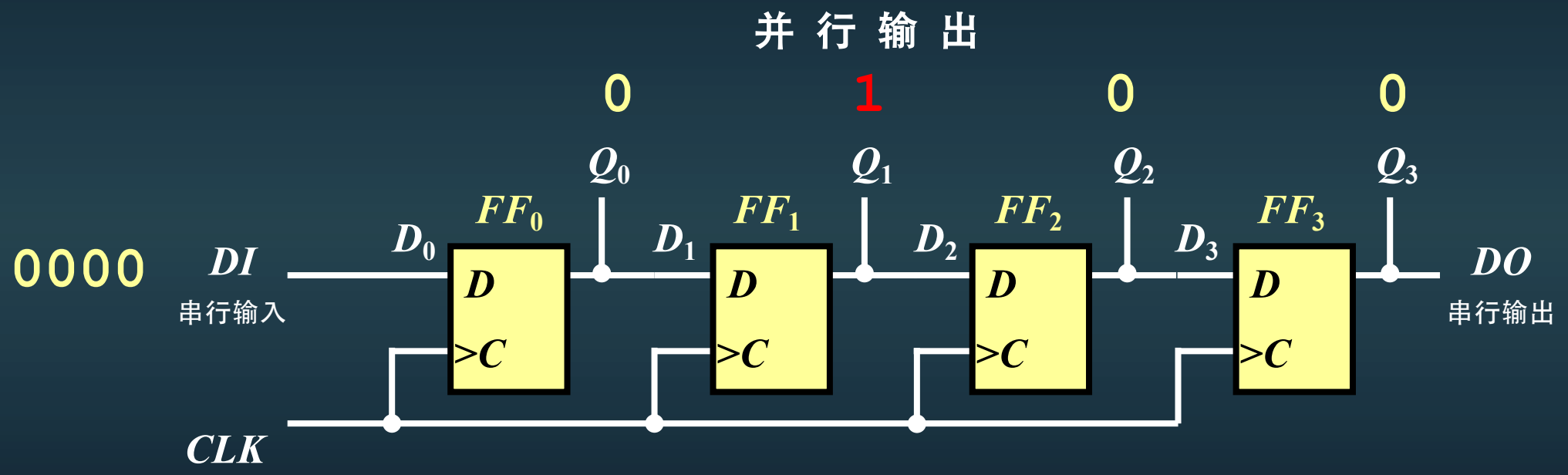
移位寄存器$Q_3Q_2Q_1Q_0$



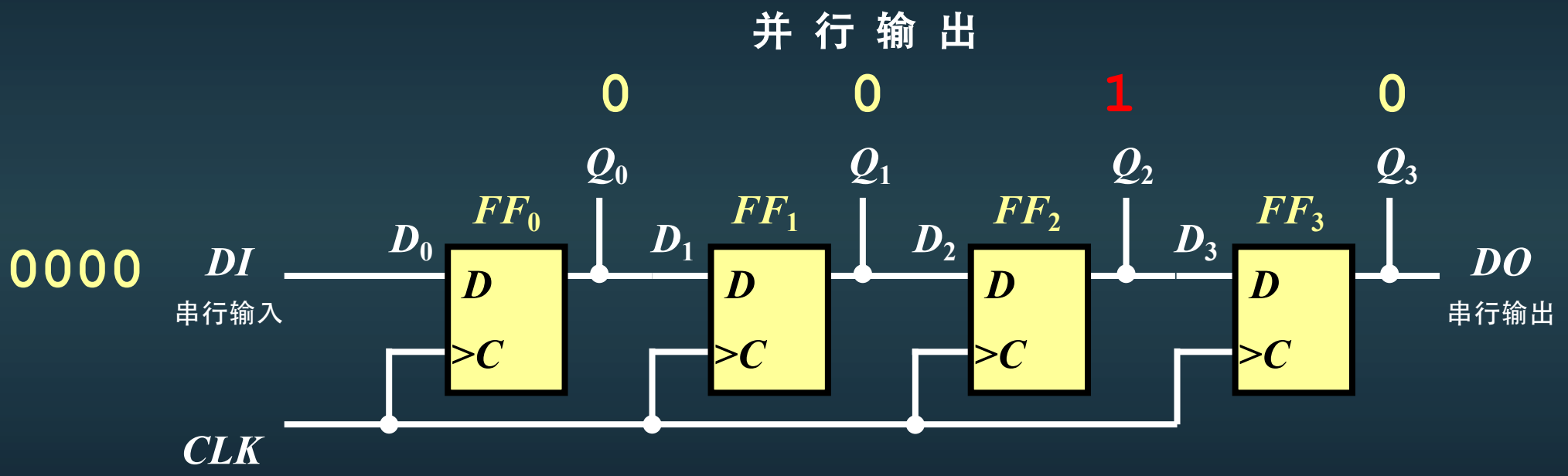
移位寄存器$Q_3Q_2Q_1Q_0$



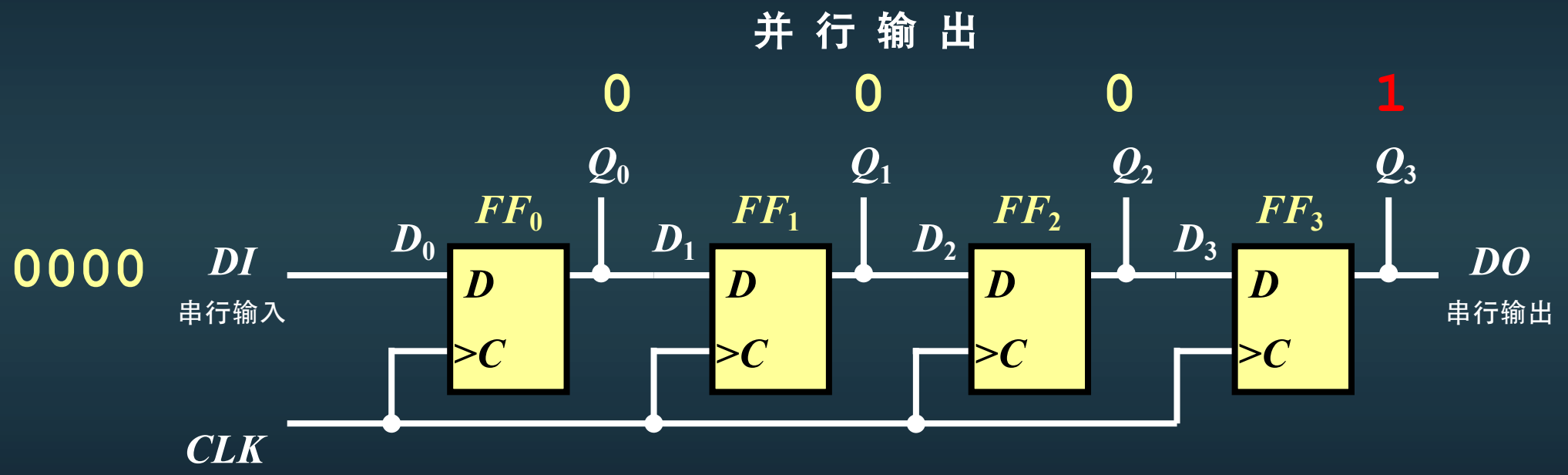
移位寄存器$Q_3Q_2Q_1Q_0$



移位寄存器$Q_3Q_2Q_1Q_0$



移位寄存器$Q_3Q_2Q_1Q_0$



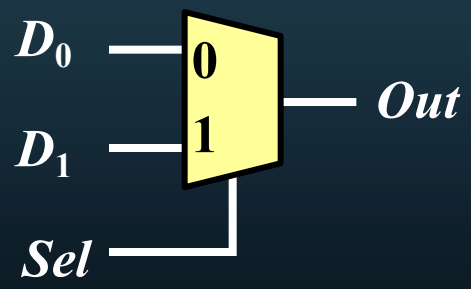
触发条件考虑

```
module D_latch_with_enable (
        input D,
        input Enable,
        output reg Q
        );
        always @(D or Enable) begin
                if (Enable == 1'b1) begin
                        Q <= D; //assign value from D to Q
                end
        end
endmodule
```

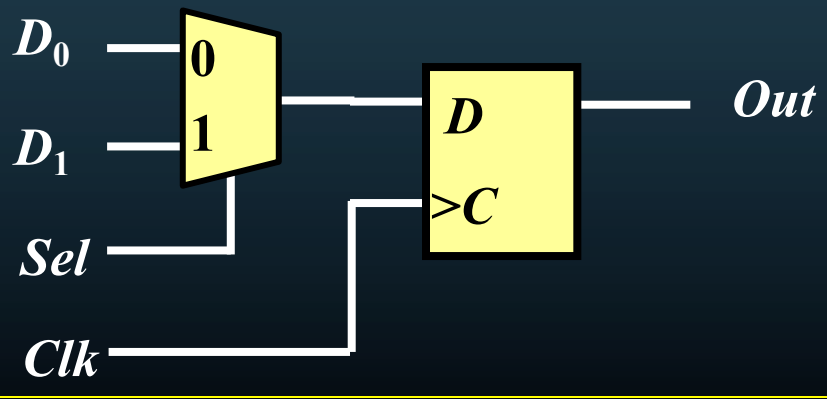
```
module D_flipflop (
        input D,
        input Clock,
        input Enable,
        output reg Q
        );
        always @(posedge Clock) begin
                if (Enable == 1'b1) begin
                        Q <= D;  //assign value from D to Q only enable
                end
        end
endmodule
```

触发条件考虑

```
Module comb(
       input D0,D1,Sel,
       output reg Out
       );
       always @(*) begin
         if (Sel) Out = b;
         else Out = a;
       end
endmodule
```




```
Module sequence(
       input D0,D1,Sel,clk,
       output reg Out
       );
 always @(posedge clk) begin
       if (Sel) Out <= b;
         else Out <= a;
       end
endmodule
```



Verilog中阻塞赋值与非阻塞赋值

1．在描述组合逻辑的always块中用阻塞赋值；

2．在描述时序逻辑的always块中用非阻塞赋值。

Verilog中阻塞赋值与非阻塞赋值

1．在描述组合逻辑的always块中用**阻塞赋值**；

2．在描述时序逻辑的always块中用非阻塞赋值。

Verilog中阻塞赋值与非阻塞赋值

1．在描述组合逻辑的always块中用**阻塞赋值**；

2．在描述时序逻辑的always块中用**非阻塞赋值**。

阻塞赋值

- 阻塞赋值操作符用等号(即 =)表示。
- “阻塞”是指在进程语句（initial和always）中，当前的赋值语句阻断了其后的语句
- 后面的语句必须等到当前的赋值语句执行完毕才能执行。
- 阻塞赋值可以看成是一步完成的
- 即：计算等号右边的值并同时赋给左边变量

阻塞赋值

- 阻塞赋值操作符用等号(即 =)表示。
- `G1 = next_Q;`
- `G2 = G1;`
- 结果是 `G2` 等于 `next_Q`

非阻塞赋值

- 非阻塞赋值操作符用小于等于号 （即 <= ）表示。
- “非阻塞”是指在进程语句（initial和always）中，当前的赋值语句不会阻断其后的语句。
- 非阻塞语句可以认为是分为两个步骤进行的：
- ①计算等号右边的表达式的值
- ②在本条赋值语句结束时，将等号右边的值赋给等号左边的变量。

非阻塞赋值

- 非阻塞赋值操作符用等号（即 <= ）表示。
- G1 <= next_Q;
- G2 <= G1;
- 结果是 G2 等于 G1 以及 G1 等于 next_Q

阻塞与非阻塞的区别

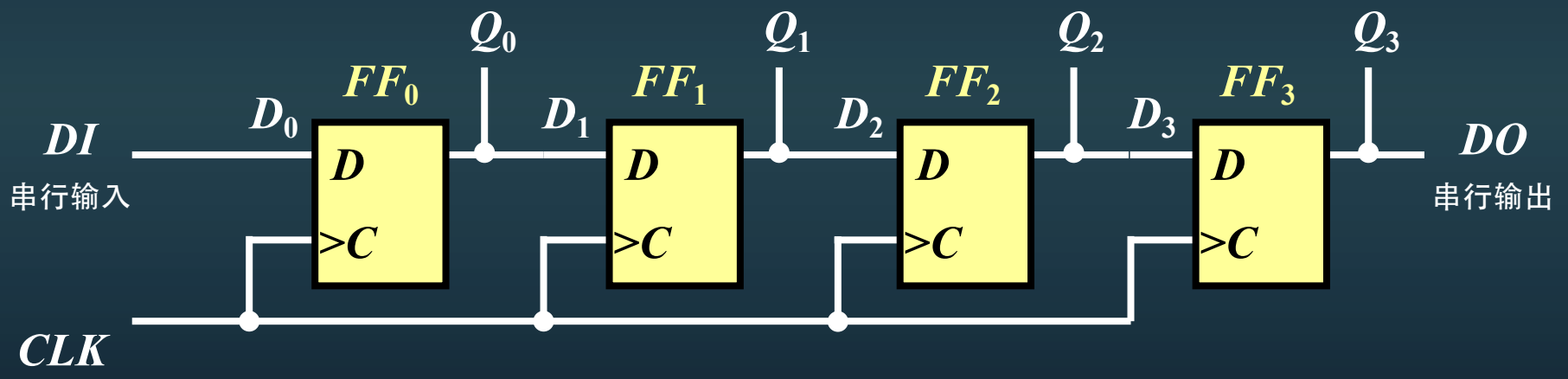
阻塞赋值是按**顺序**执行

非阻塞赋值是**并行**执行

提问：这里使用阻塞赋值还是非阻塞赋值？

```
always @(posedge clk)
    begin
        if (reset) begin
            reg_value <= 8'b00000001;
        end
        else if (en) begin
            reg_value[1] = reg_value[0];
            reg_value[2] = reg_value[1];
            reg_value[3] = reg_value[2];
            reg_value[4] = reg_value[3];
            reg_value[5] = reg_value[4];
            reg_value[6] = reg_value[5];
            reg_value[7] = reg_value[6];
            reg_value[0] = reg_value[7];
        end
    end
```

移位寄存器



问题：这里使用阻塞赋值还是非阻塞赋值？

```
always @(posedge clk)     //循环移位寄存器
    begin
        if (reset) begin
            reg_value <= 8'b00000001;  //按下复位键（reset）时，循环重置寄存器的值
        end
        else if (en) begin  //使能状态下，每个时钟上升沿循环左移一位
            reg_value[1] <= reg_value[0];
            reg_value[2] <= reg_value[1];
            reg_value[3] <= reg_value[2];
            reg_value[4] <= reg_value[3];
            reg_value[5] <= reg_value[4];
            reg_value[6] <= reg_value[5];
            reg_value[7] <= reg_value[6];
            reg_value[0] <= reg_value[7];
        end
    end
    assign out = reg_value;  //将寄存器连接到输出端
```

约定一个习惯

- 高位到低位
- 从左到右
- 比如地址端口的定义
  - $A_3A_2A_1A_0$
- 比如数据端口的定义
  - $D_3D_2D_1D_0$

课后作业

1．实现流水灯，你还有哪些方法？

- 至少给出一种

2．分频器设计

- 2分频
- 十分频
- 奇数分频