# 赋值语句

一、赋值语句

1.连续赋值语句——assign语句，用于对wire型变量赋值

[例] assign c=a&b; //a、b、c均为wire型变量

2.过程赋值语句——用于对reg型变量赋值，有两种方式：

非阻塞（non-blocking)赋值方式：

赋值符号为<=，如 b <= a ；

阻塞（blocking)赋值方式：

赋值符号为=，如 b = a ；

二 、非阻塞赋值与阻塞赋值方式的主要区别

非阻塞（non-blocking)赋值方式 ( b<= a)：

-b的值被赋成新值a的操作, 并不是立刻完成的，而是在块结束时才完成；

-块内的多条赋值语句在块结束时同时赋值；

-硬件有对应的电路。

阻塞（blocking)赋值方式 ( b = a)：

-b的值立刻被赋成新值a；

-完成该赋值语句后才能执行下一句的操作；

-硬件没有对应的电路，因而综合结果未知。

1. 非阻塞赋值方式

always 块中所有非阻塞赋值语句在求值时所用的值全部都是进入 always 时，各个变量已具有的值。

非阻塞的意思是每条赋值语句的结果直到 always块的结尾才能看到。

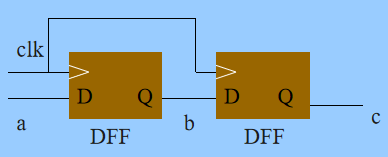
always @(posedge clk)

begin

b <= a ;

c <= b;

end



2. 阻塞赋值方式

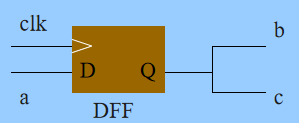
always @(posedge clk)

begin

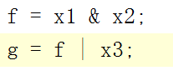
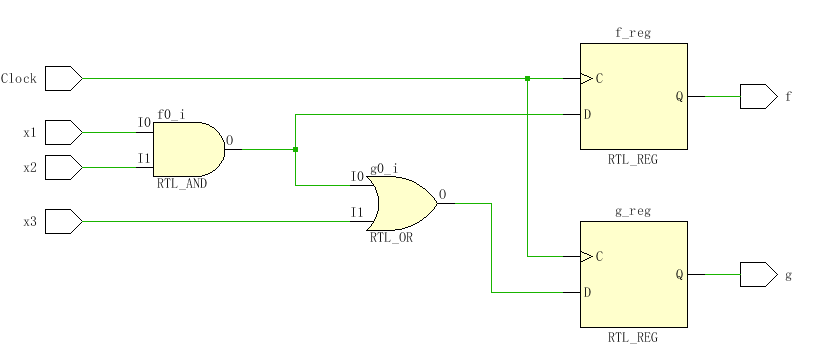
b = a ;

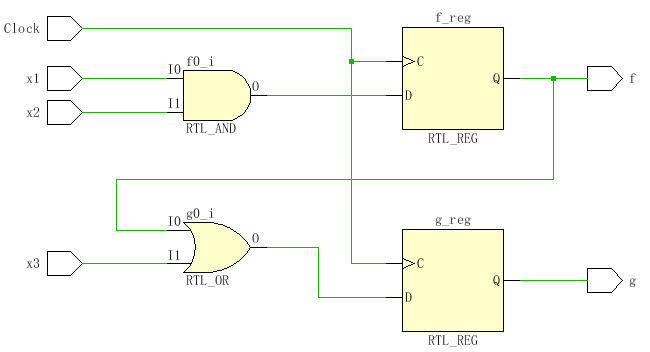
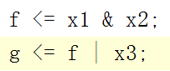
c = b;

end



例如：

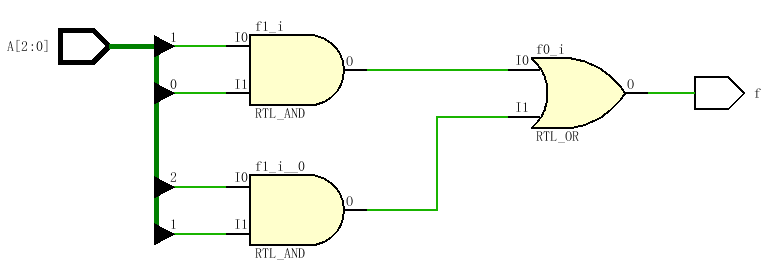


例2：

f =A[1]&A[0];

f = f|(A[2]&A[1]);

实现了f=a1a0 + a2a1



但是

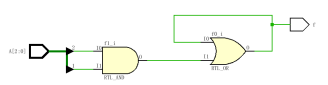
f <= A[1]&A[0];

f <= f |(A[2]&A[1]);

**1. 非阻塞赋值语句的结果仅在 always 块中所有语句求值结束后才可以看到。**

**2. 当always块中同一变量多次赋值后，只保留最后一次赋值的结果。**

所以产生逻辑函数 f = f + a2a1



# 组合电路和时序电路

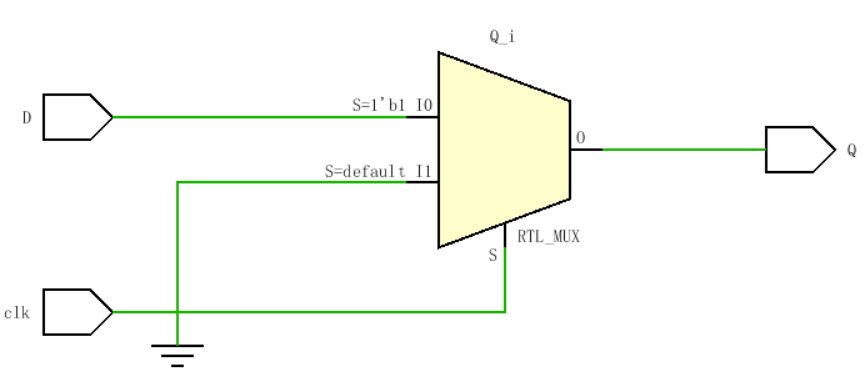
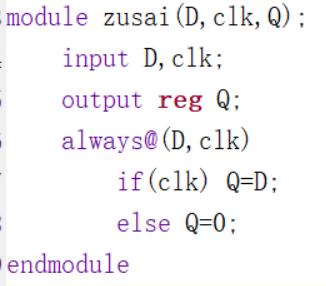
总结：

如果有**保持原值**的逻辑，则为时序逻辑；

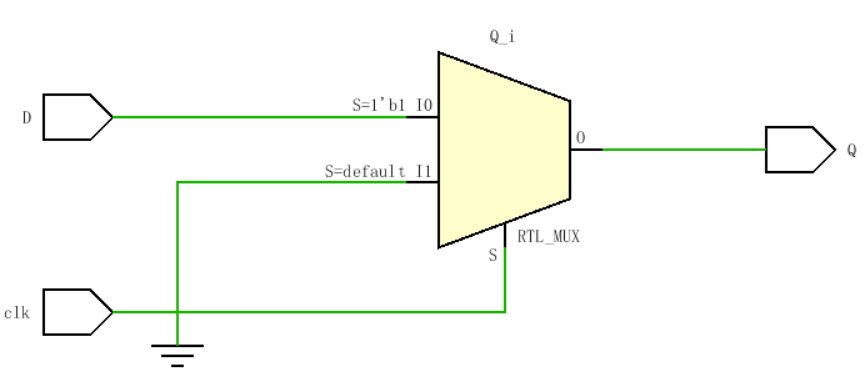
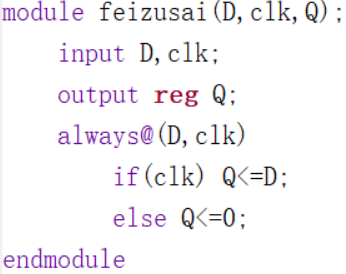
如果有**沿触发**的逻辑，则为时序逻辑；

1、非阻塞赋值不一定产生时许逻辑

阻塞赋值

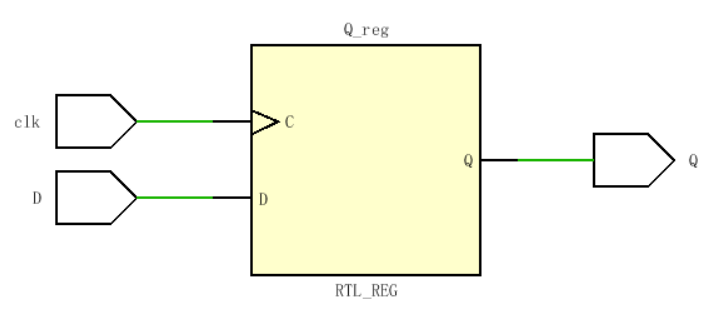
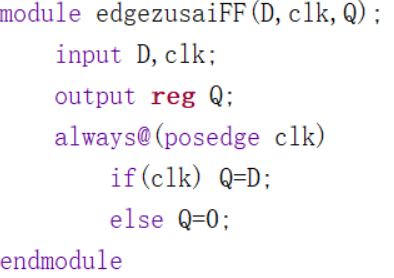


非阻塞赋值



2、沿触发一定是时许电路

沿触发阻塞赋值



沿触发非阻塞赋值

