module PAOBIAO(a,b,c,d);  
input a,b,c;  
output d;  
reg d;  
      
       always@(posedge a or negedge b)  
  
if(b) d<=0;  
  
else if(c)d<=1;  
endmodule  
先看这么一段简单的程序,编译:提示错误。错在哪里？经过一晚上才终于琢磨透彻。这是错误提示：Error (10200): Verilog HDL Conditional Statement error at PAOBIAO.v(75): cannot match operand(s) in the condition to the corresponding edges in the enclosing event control of the always construct  
大概意思就是不能匹配操作条件中的边沿事件和always语句中的控制事件。错误指示在红色一行。  
仔细分析语法吧，看起来真的没有错啊。不断地修改再编译，无意中去掉了negedge b  
这句，即酱红色部分改为always@(posedge clk),编译后没错了。网上搜也有人遇见过这种情况可是几乎没一人解释到正点上。最后答案还是在自己觉得编写的很垃圾的课本上找到了（因为刚上来我总觉得这本书一大堆屁话，看不懂把浅显的问题说的那么复杂，呵呵，看来该改变下自己的态度了）。  
加上我的理解分析一下吧：首先这很明确要描述的是一个时序模块电路（这里仅讨论带posedge or negedge 的时序模块），敏感信号列表里有a和b两个信号，但是边沿敏感时钟信号只能有一个，如何区分哪个是时钟信号？规定是这样的，如果把a定义为时钟信号，那么在always过程结构中不能再出现a信号了，如上面例子就可以看出a是作为时钟信号的。这也满足要求似乎不应该编译出错啊，还有一点就是剩余的b了，这个b在这里是一个异步控制信号（或者是异步输入信号），这个b除了在敏感信号表中给出对应的表述外（即always后的negedge b）在always过程语句中必须给予说明（经实践是always结构中带if语句的，不带if语句也可不说明），简单的说就是在always语句中加入有if语句则必须把b放在if的（）中作为判定条件。看上面例子，b也组为了判定条件可是为什么还会报错？还有一点经过实践，假如在敏感信号里表中描述的是negedge b，则必须是if（！b），如果是posedge b则必须是if（b）。所以这里想要改正错误有两种方法：  
一、如上所述，去掉红色部分的negedge b，但是这样改已经改变了程序的功能；  
二、将if（b）改为if（！b），可以想想把if(b)作为判断条件没有实际意义的。  
  
最后还有一点就是把else if(c)中的c换为！b即else if(!b),由此看来假如把b作为敏感控制信号，并且always语句中出现if语句的话，则b必须在第一条if语句中判断，若放在后面的else if语句中就会编译出错。