forever语句用法：在forever后紧接着表达式或者语句块。几乎总是用在initial块中，当仿真工具执行到$finish和$stop时结束forever循环。

与并发的始终产生器always@（Clock1）#10Clock<=~Clock1；不等效。这种方法不能用阻塞赋值的方法来产生时钟，只有当前仿真事件都完成了以后，Clock1的值才会发生改变，而对于forever来说则按照十个时间单位后翻转一次。always是一种并发结构，不能顺序执行。

repeat语句用法：repeat(number\_of\_times)，然后紧接着其他的语句或块。repeat是一种严格的循环计数结构。当仿真工发现tepeat语句时，把numerf times存下来，然后运行指定的次数。和for迭代不同，number\_of times的值不能在循环过程中改变。可以用前面提前终止forever的方法提前终止repeat循环（不需要控制循环的状态时才可以使用repeat）

while用法：它的用法是：while(表达式)，然后紧接着其他的语句或块。在工具第一次读到while的时候，工具会判断表达式的值，然后每一次执行完整段语句时，再重新判断表达式的值，直到表达式的值为假为止。只要while的表达式不为0，while就会执行后面的语句。如果表达式为0，则退出while循环接着执行下面的语句。

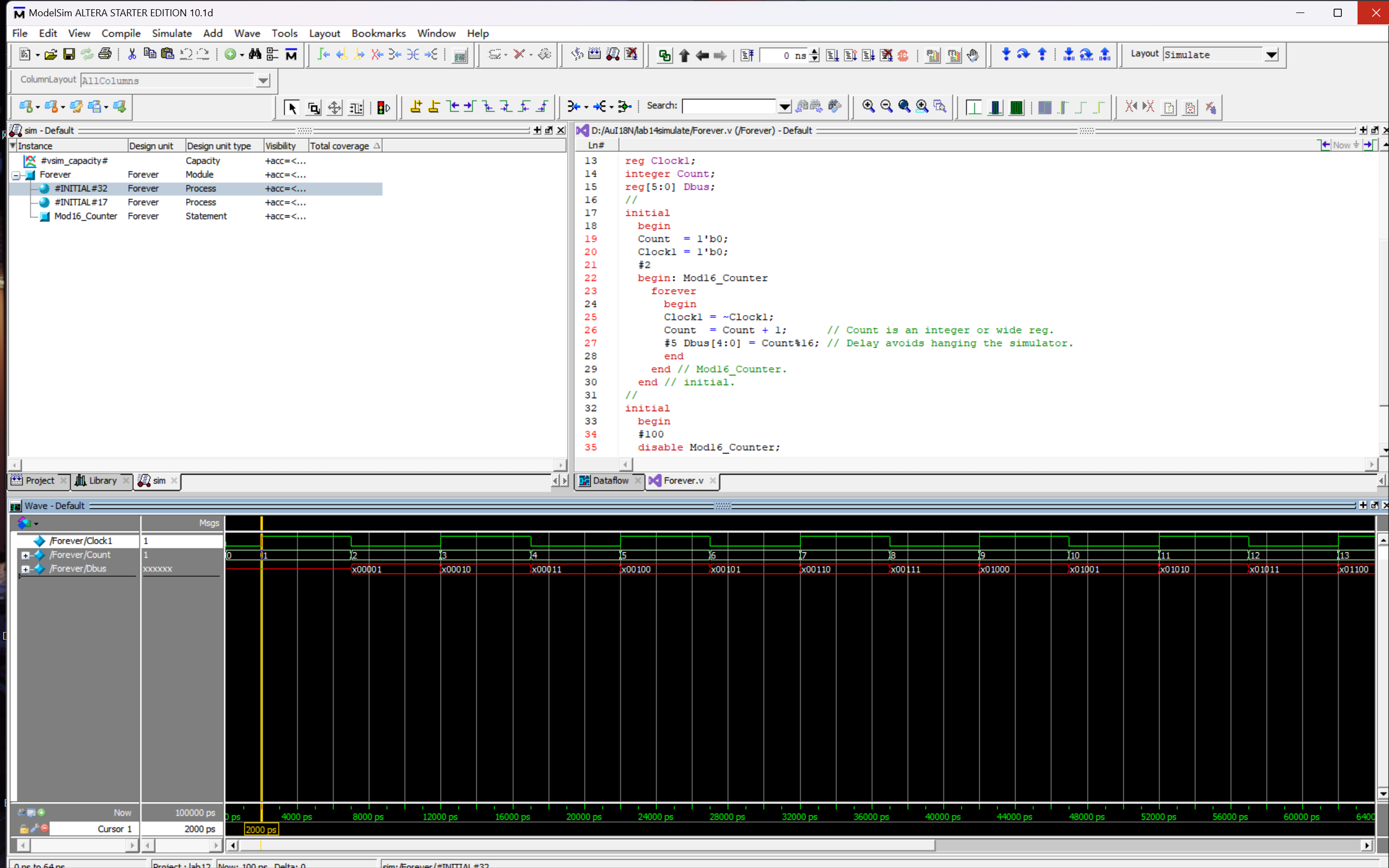
while的表达式使得while比repeat可以应用在更多的场合。而在while和for中，都可以加入时间延迟，因此，在很多情况下，while可以和for互换。但while在处理控制信号时可以添加延迟信息而for语句不行。

case语句用法：语句if主要优点是支持关系表达式，所有的条件都可以放到一个表达式里头去。与此对应的语句是case，一次只能指定一个确定的值。虽然多个条件之间可以用逗号隔开，但是仍然是需要明确指定的。不过，case条件列出来之后像一个表，可读性会比一连串的if…else链好一些。除此之外，if和case的主要区别在于它们处理“x”和“z”是不一样的。

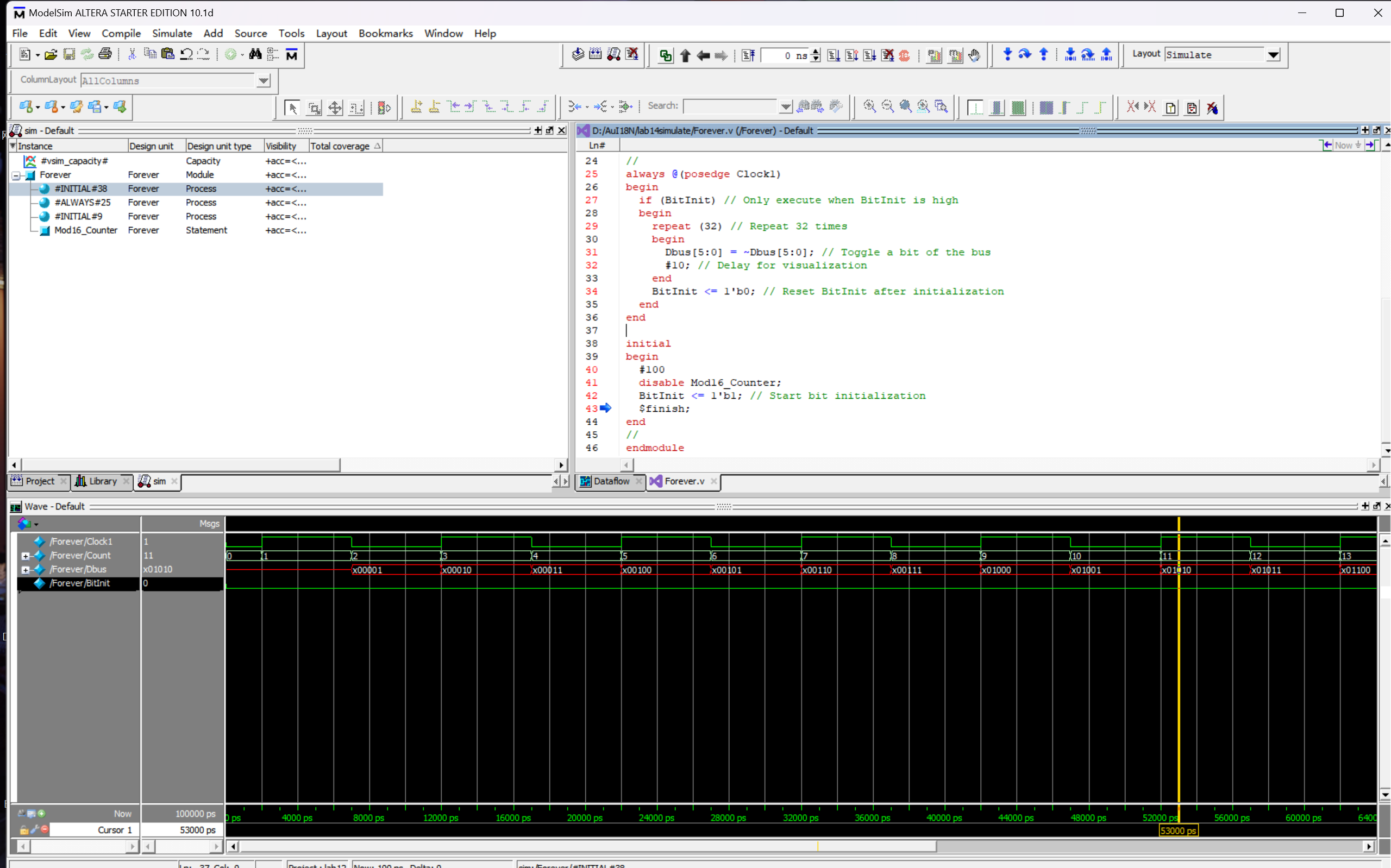
在if判断两个包含“x”和“z”的逻辑状态是否相等时，把“x”认为是1或0，并不是一个确定的值。因此，对于一个包含了“x”和“z”的变量X，即使对于“if(X=X)”这样的表达式也是认为不成立的，而会执行else分支。

而case却不是这么处理的，case对指定的比特类型进行匹配，无论其中是否含有“x”或“z”。如果case表达式中的向量含有“×”或“z”的某种逻辑状态，并且其中一个可选项包含相同类型，case语句认为找到匹配，并根据该匹配选项选择性地执行语句。

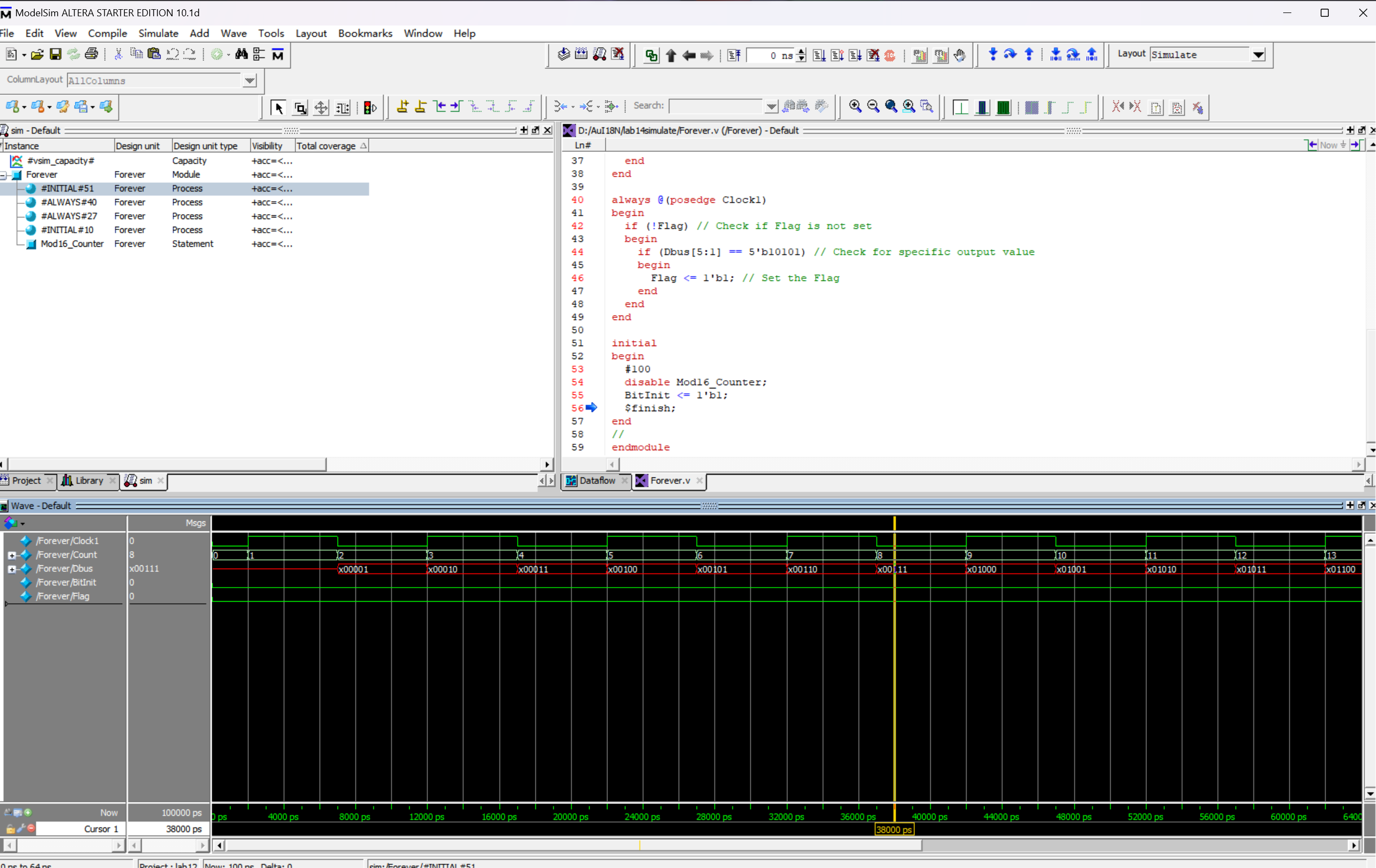
第一步：



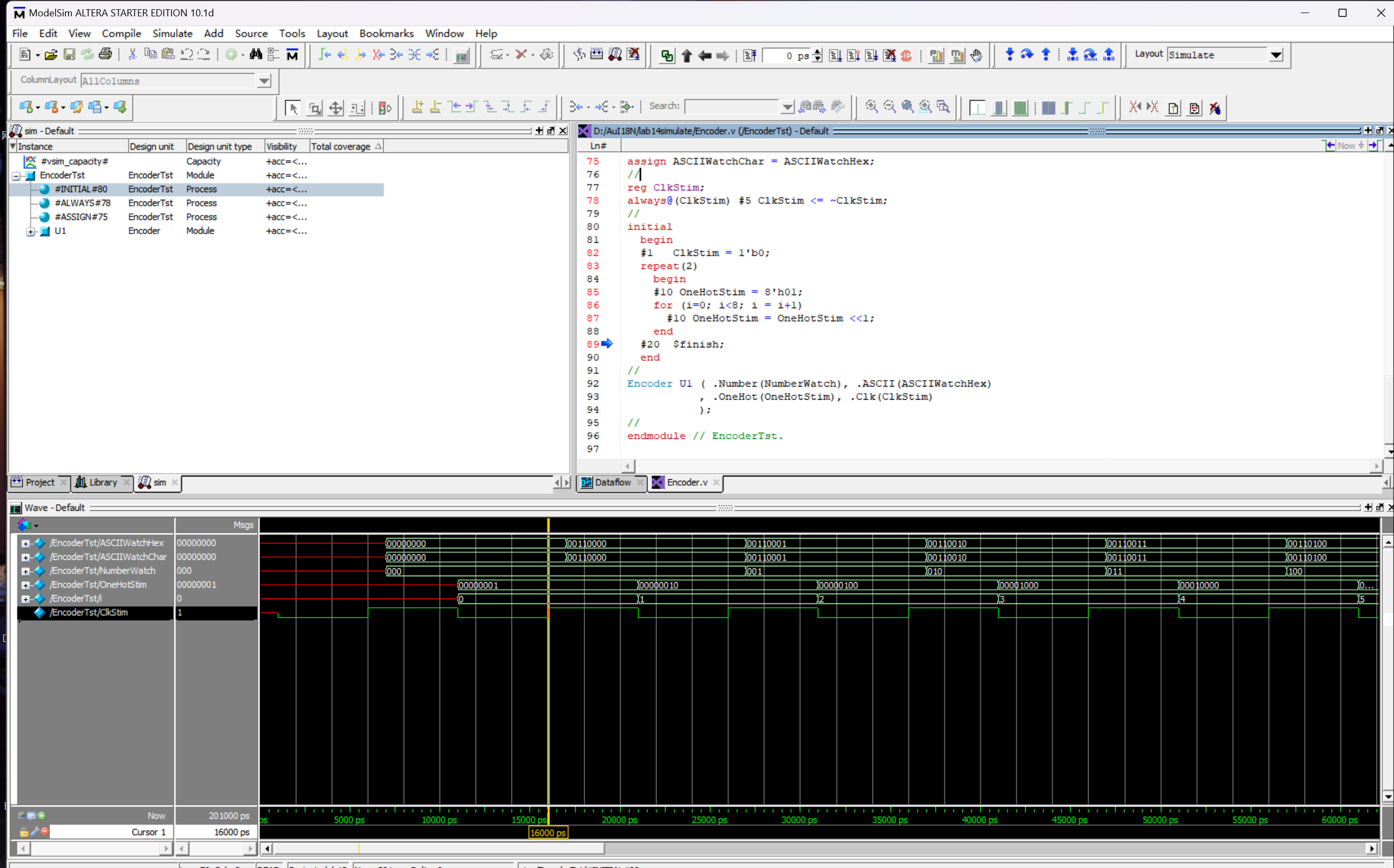
第二步：



第三步：

通过添加一个flag信号来显示是否检测到这个特定的组合。

第四步：



第五步：

