1）在描述组合逻辑的 always 块中使用阻塞赋值，则综合组合逻辑的电路结构；
2）在描述时序逻辑的 always 块中使用非阻塞赋值，则综合时序逻辑的电路结构。

语法要点

1) always里面赋值左边必须申明成reg

2) assign表达式左边必须申明成wire

3) 阻塞赋值用<=

4) 非阻塞赋值用 =

1) 边沿触发生成寄存器的时序逻辑

2) 电平触发条件完整，生成组合逻辑

3) 电平触发，条件不完成，生成锁存器的时序逻辑

结论：

声明成reg，不一定得到寄存器。

声明成reg，也可能得到锁存器

Coding要点

- 如果是边沿触发的逻辑

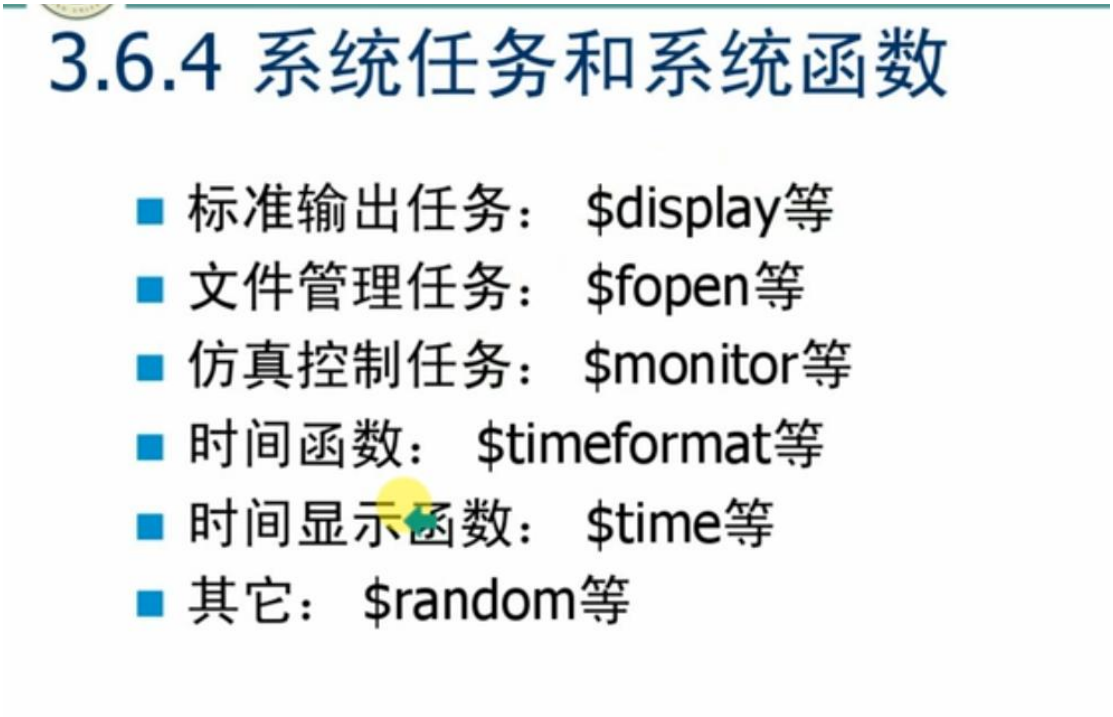
比如 always@(posedge clk), 里面一律使用 <= 赋值

- 如果是电平触发的逻辑，一律使用 = 赋值

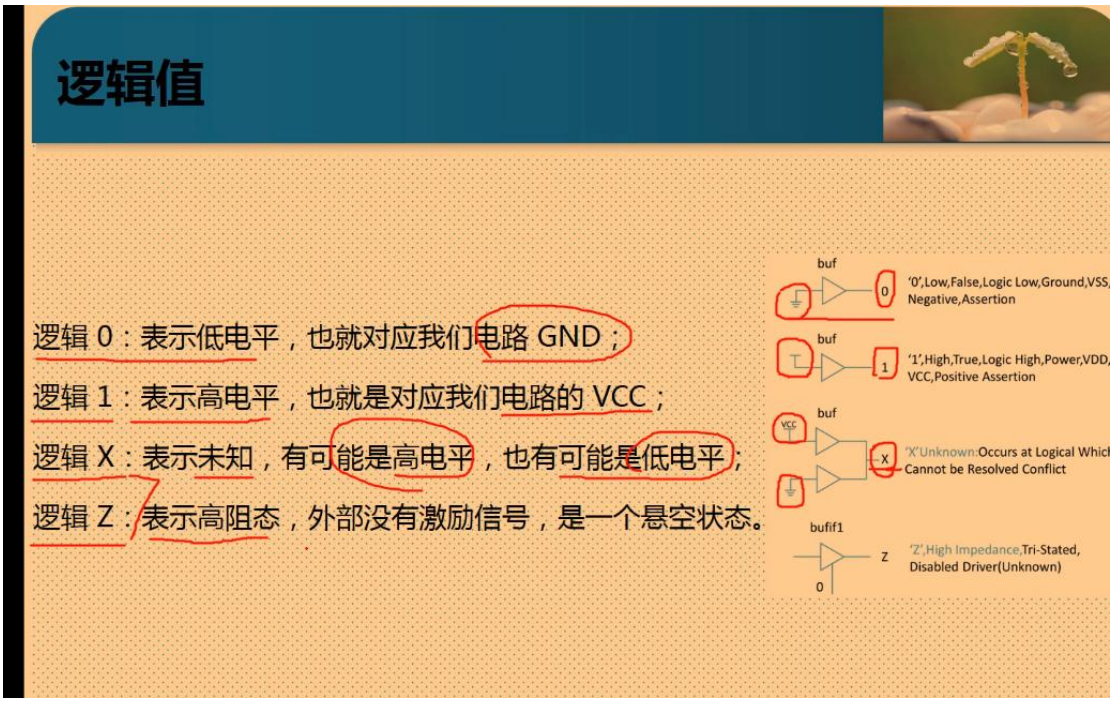
逻辑简单用 assign语句；逻辑复杂，用always 语句

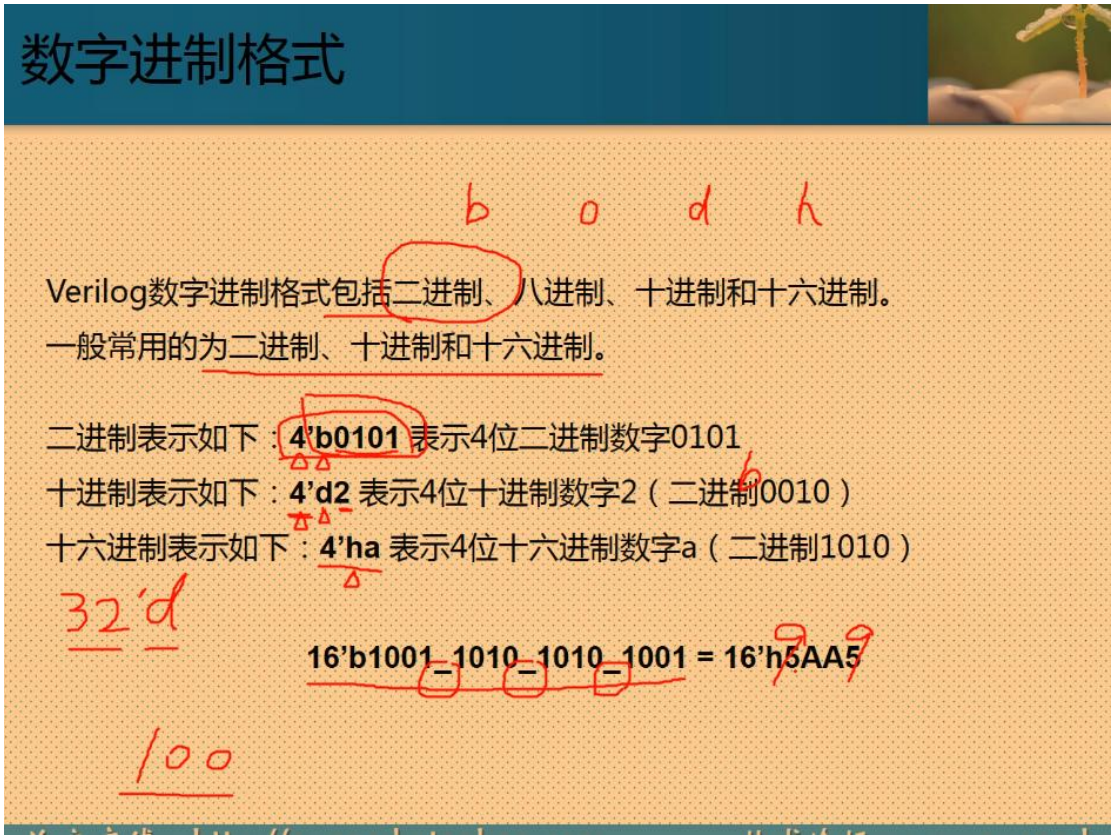
分支条件写完整，防止出现锁存器

以下语法只用于仿真

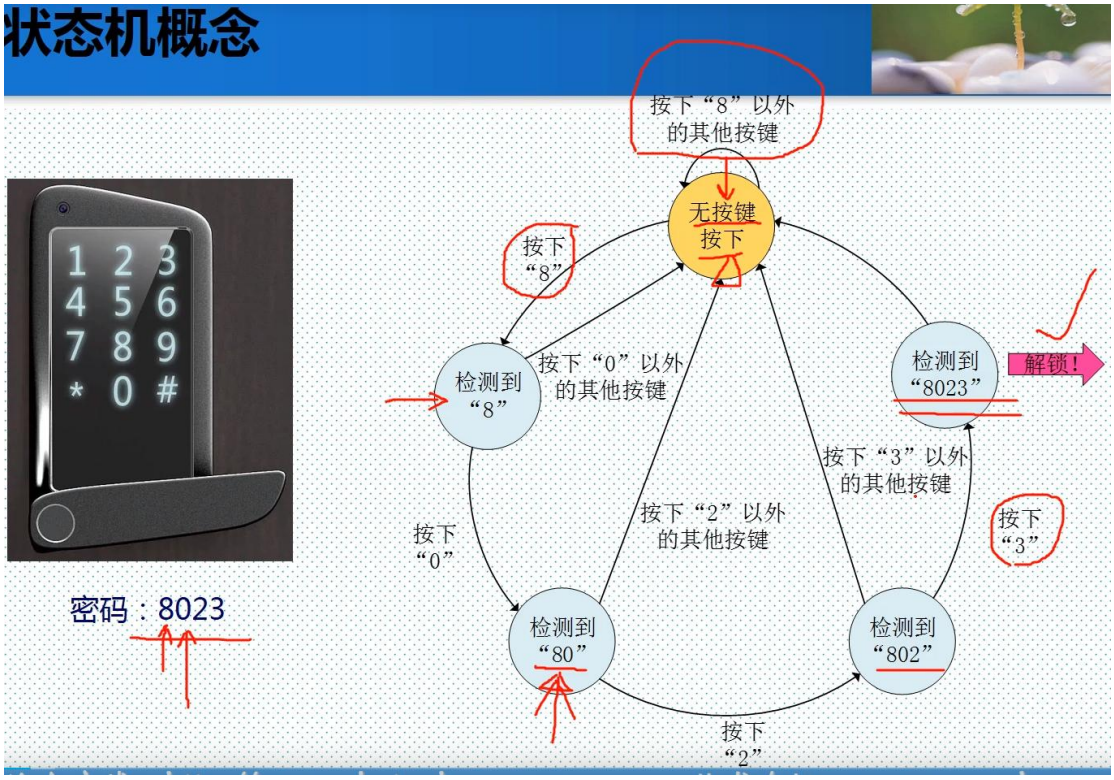


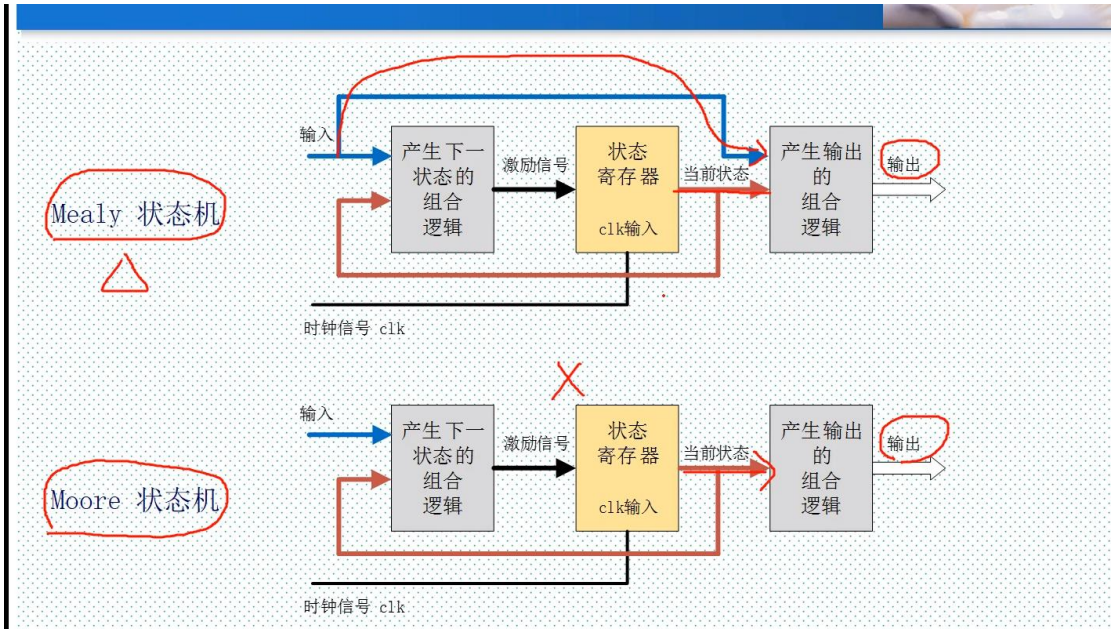
always 语句块里面：电平触发的是组合逻辑，边缘触发的是时序逻辑.




数字进制格式
Verilog数字进制格式包括二进制、八进制、十进制和十六进制。
一般常用的为二进制、十进制和十六进制。
二进制表示如下：4'b0101 表示4位二进制数字0101
十进制表示如下：4'd2 表示4位十进制数字2（二进制0010）
十六进制表示如下：4'ha 表示4位十六进制数字a（二进制1010）
16'b1001_1010_1010_1001 = 16'h5AA5

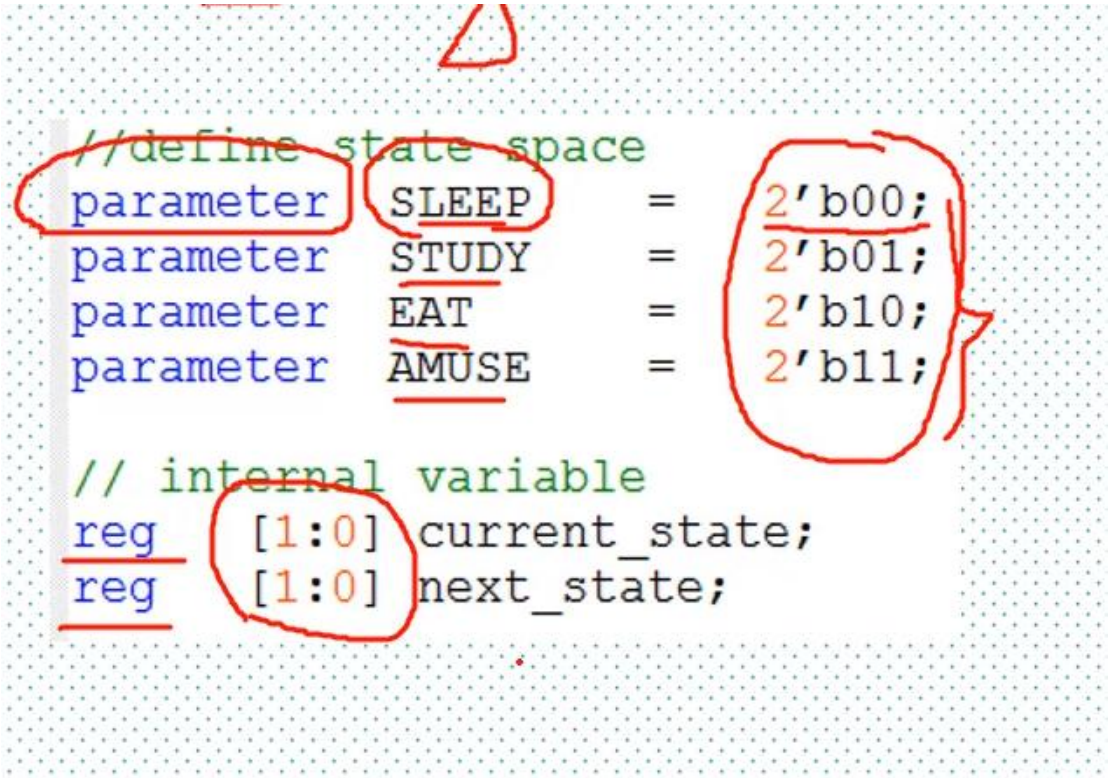
状态机


状态机概念
密码：8023
无按键按下
检测到“8”
检测到“80”
检测到“802”
检测到“8023”
解锁！
按下“8”以外的其他按键
按下“8”
按下“0”以外的其他按键
按下“0”
按下“2”以外的其他按键
按下“2”
按下“3”以外的其他按键
按下“3”



状态机设计＂四段论＂

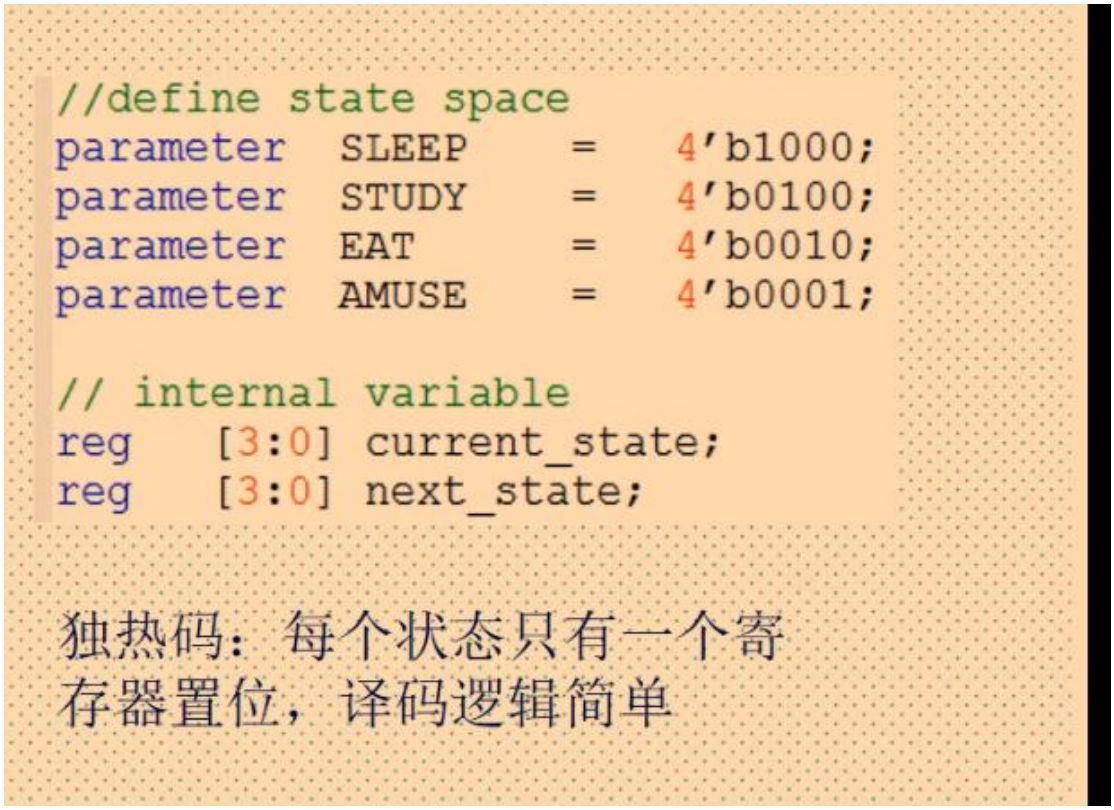
1．定义状态空间：列举所有可能出现的状态，定义当前状态和下一个状态的寄存器



```
//define state space
parameter   SLEEP    =   2'b00;
parameter   STUDY    =   2'b01;
parameter   EAT      =   2'b10;
parameter   AMUSE    =   2'b11;

// internal variable
reg   [1:0] current_state;
reg   [1:0] next_state;
```

状态空间可以有多种编码方式：二进制编码，one-hot 编码等



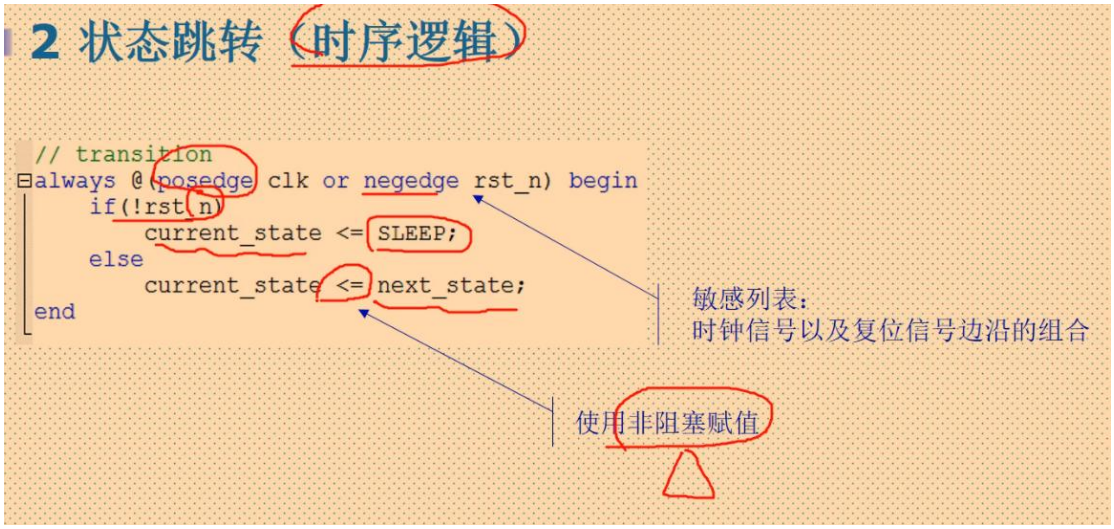
时序逻辑要用非阻塞赋值，组合逻辑要用阻塞赋值.

阻塞赋值：这一行语句没执行完，不会执行下一行，串行

非阻塞复制：这一行语句是否执行完不影响下一行语句的执行，并行

2．状态跳转

在时钟上升沿或者下降沿到达时进行状态转换

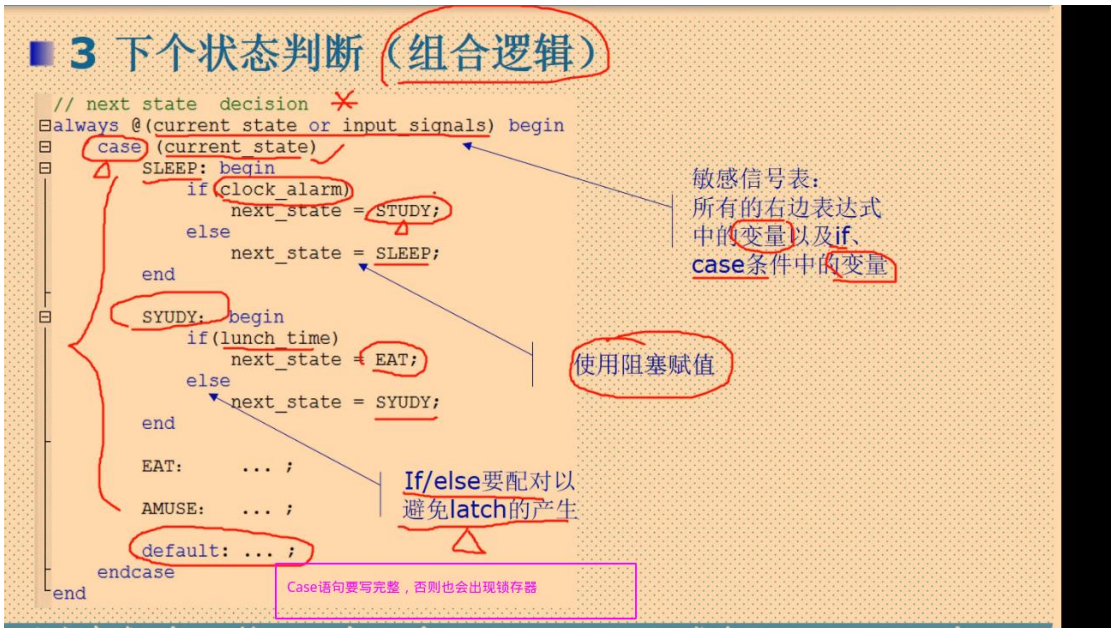


锁存器：电平触发的存储器

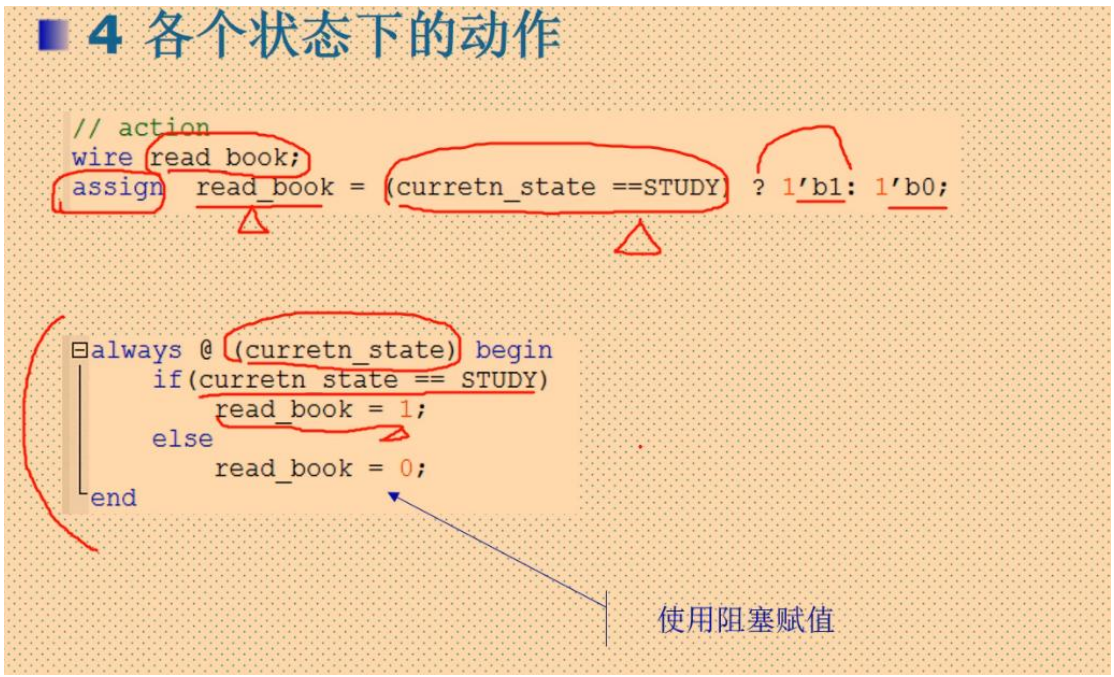
触发器：边缘触发的存储器

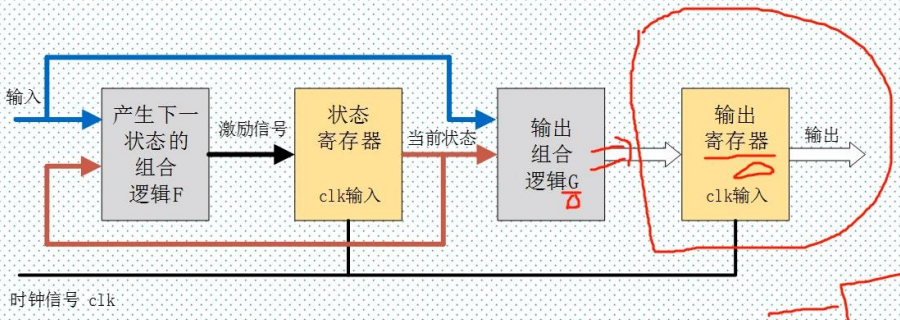
3．下一个状态的判断

根据当前状态和输入来判断下一个状态应该是什么



4．各个状态下的动作





三段式可以在组合逻辑后再增加一级寄存器来实现时序逻辑输出：

1、可以有效地滤去组合逻辑输出的毛刺；

2、可以有效地进行时序计算与约束；

3、另外对于总线形式的输出信号来说，容易使总线数据对齐，从而减小总线数据间的偏移，减小接收端数据采样出错的频率。