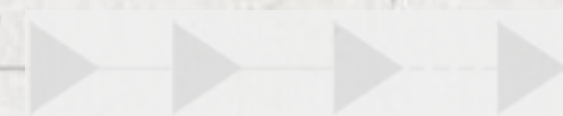


# 芯动力——硬件加速设计方法

## 第三章 同步电路与跨时钟域电路设计(6)

邸志雄@西南交通大学

zxdi@home.swjtu.edu.cn





逻辑设计的重要内容



状态机



工程师的逻辑功底



本章主要内容如下:

- 状态机的基本概念
- 如何写好状态机





# 概述

## 状态机的本质

- 对具有逻辑顺序或时序规律事件的一种描述方法。

逻辑顺序

时序规律

核心

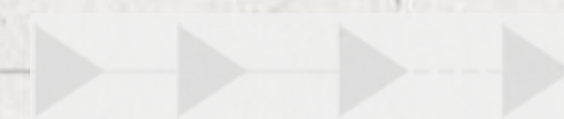
强项

所有具有逻辑顺序和时序规律的事情  
都适合用状态机描述

地点：食堂  
功能：吃饭

地点：宿舍  
功能：睡觉

地点：教室  
功能：学习





## 状态机的应用思路：

1

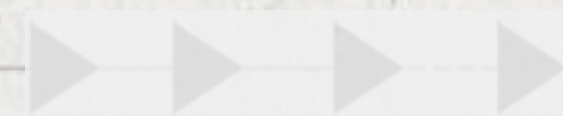
### 从状态变量入手

- 如果一个电路具有时序规律或者逻辑顺序，我们就可以自然而然地规划出状态，从这些状态入手，分析每个状态的输入，状态转移和输出，从而完成电路功能；

2

- 明确电路的输出的关系，这些输出相当于状态的输出
- 回溯规划每个状态
- 和状态转移条件与状态输入

- 无论那种思路，使用状态机的目的都是要控制某部分电路，完成某种具有逻辑顺序或时序规律的电路设计。







逻辑电路

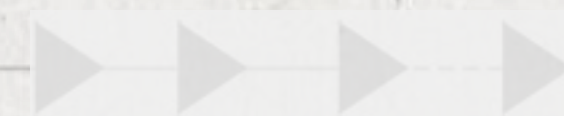
时序逻辑

微处理器



- 都适合用状态机方法进行描述

- 大家要打开思路，不要仅仅局限于时序逻辑
  - 发现电路的内在规律
  - 确认电路的“状态变量”
  - 大胆使用状态机描述电路模型
- 由于状态机不仅仅是一种电路描述工具，它更是一种思想方法，而且状态机的 HDL 语言表达方式比较规范，**有章可循**，所以很多有经验的设计者习惯用状态机思想进行逻辑设计，对各种复杂设计都套用状态机的设计理念，从而提高设计的效率和稳定性。





# 状态机的基本描述方式

状态机的基本要素：

状态

输出

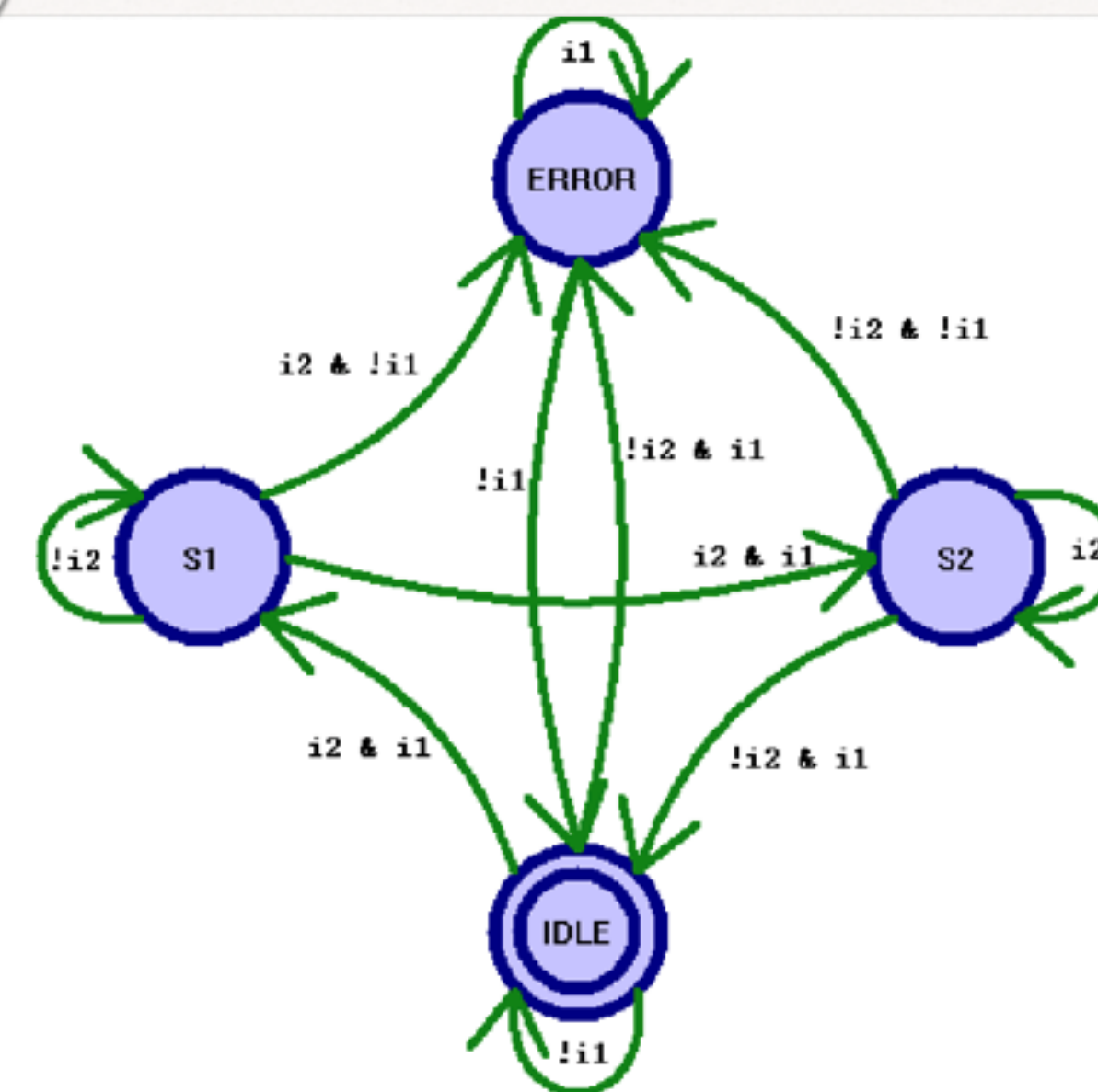
输入

逻辑设计中，状态机的基本描述方式：

状态转移图

状态转移列表

HDL 语言描述





# 如何写好状态机

## HDL 语言描述状态机

重点

- 使用 HDL 语言描述状态机有一定的灵活性，但是决不是天马行空，而是有章可循的。通过一些规范的描述方法，可以使 HDL 语言描述的状态机：

安全

稳定

高效

易于维护



# 如何写好状态机

## 什么是 RTL 级好的 FSM 描述？

- FSM 要安全，稳定性高。
- FSM 速度快，满足设计的频率要求。
- FSM 面积小，满足设计的面积要求。
- FSM 设计要清晰易懂、易维护。

如果要求 FSM 安全，则很多时候需要使用 “full case” 的编码方式，即将状态转移变量的所有向量组合情况都在 FSM 中有相应的处理，这经常势必意味着要多花更多的设计资源，有时也会影响 FSM 的频率。



## 什么是 RTL 级好的 FSM 描述?

最重要

- FSM 要安全，稳定性高。

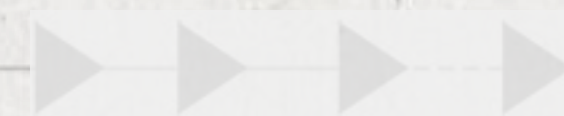
优先级最高

- FSM 速度快，满足设计的频率要求。
- FSM 面积小，满足设计的面积要求。

相对次要

- FSM 设计要清晰易懂、易维护。

优先级最低





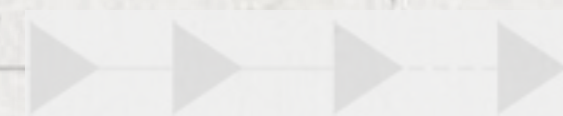
- 状态机描述时关键是要描述清楚前面提到的几个状态机的要素：
  - 如何进行状态转移；
  - 每个状态的输出是什么；
  - 状态转移是否和输入条件相关等。

状态机通常的写法:

一段式

两段式

三段式





## 一个非常典型的米勒型状态机:

共有 4 个状态

IDLE

S1

S2

ERROR

输入信号为时钟

clk

低电平异步复位信号

nrst

输入信号

i1

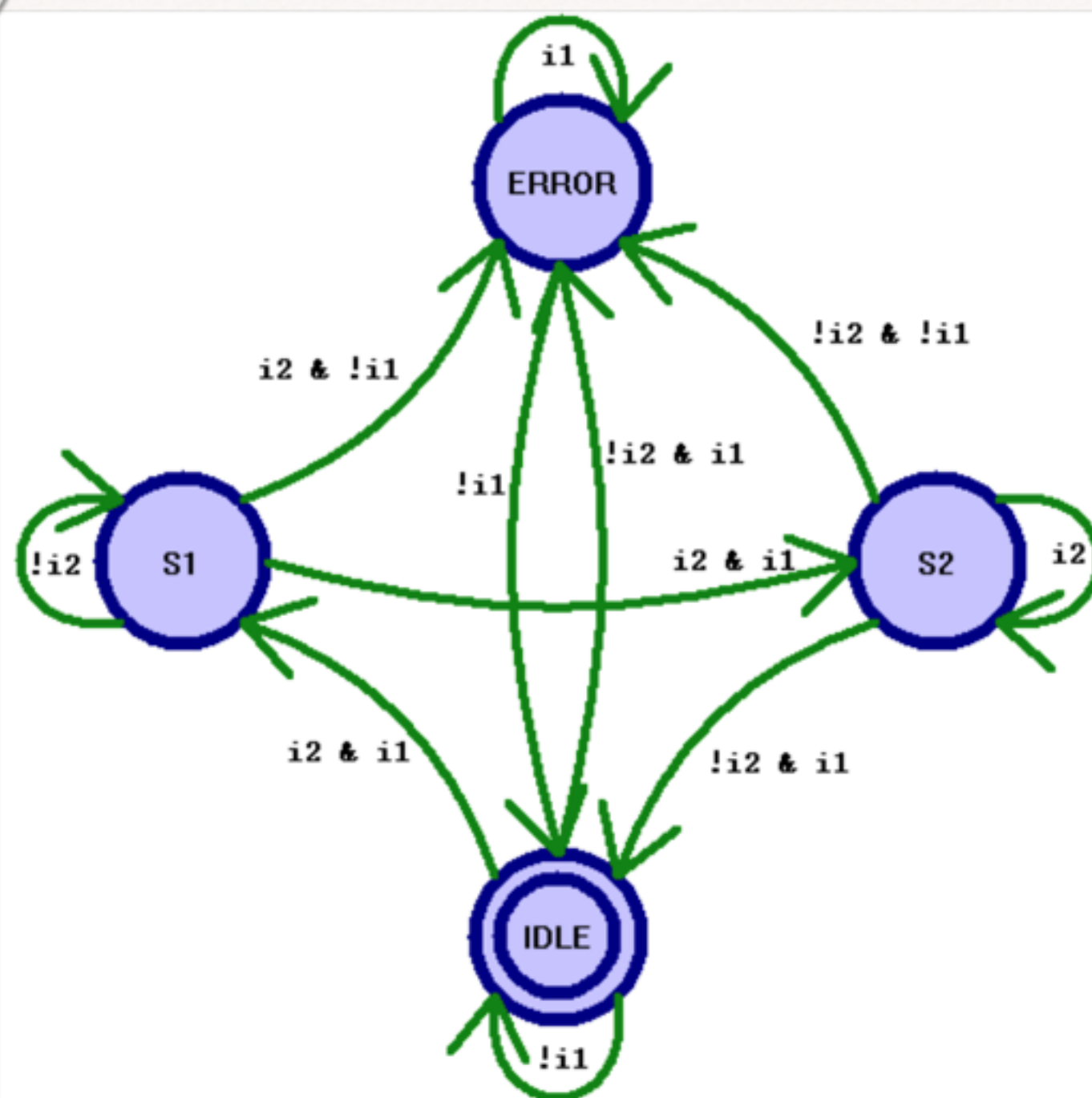
i2

输出信号

o1

o2

err





IDLE 状态的输出

$\{o1,o2,err\} = 3'b000$

S1 状态的输出

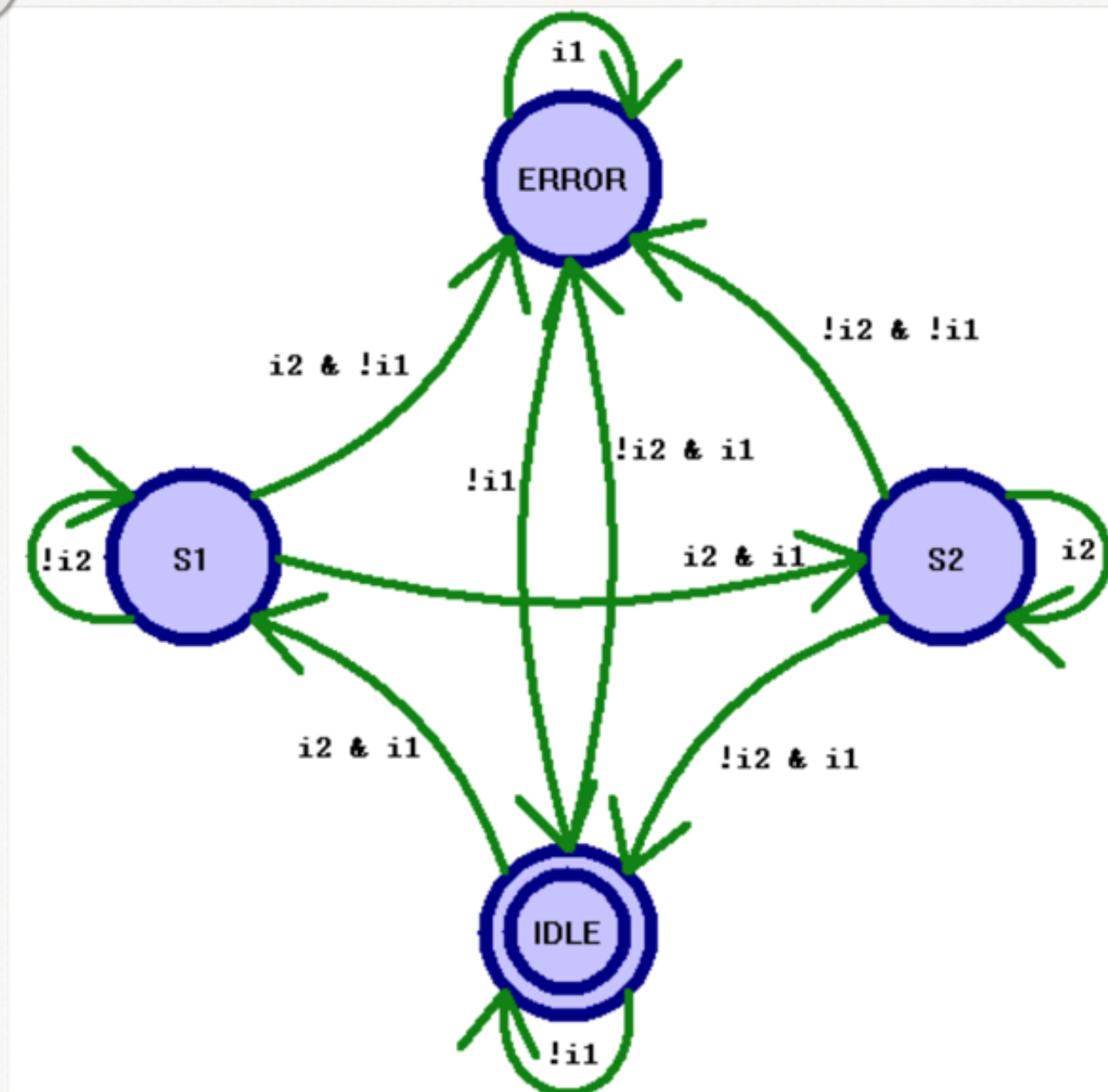
$\{o1,o2,err\} = 3'b100$

S2 状态的输出

$\{o1,o2,err\} = 3'b010$

ERROR 状态的输出

$\{o1,o2,err\} = 3'b111$

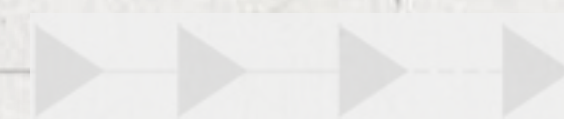





## 一段式状态机描述方法（应该避免的写法）



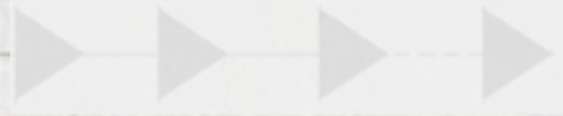
这种方式应该避免写，有很多缺点。







```
module state (nrst,clk,i1,i2,o1,o2,err);  
input        nrst,clk ;  
input        i1,i2 ;  
output       o1,o2,err;  
reg          o1,o2,err;  
reg [2:0]    NS; //NextState  
parameter [2:0] //one hot with zero idle  
    IDLE    = 3' b000,  
    S1      = 3' b001,  
    S2      = 3' b010,  
    ERROR   = 3' b100,
```







```
always @ (posedge clk or negedge nrst)
```

```
if (!nrst)
```

```
begin
```

```
    NS      <=IDLE;
```

```
    {o1,o2,err} <=3'  b000;
```

```
end
```

```
else
```

```
begin
```

```
    NS      <=3'  bx;
```

```
    {o1,o2,err} <=3'  b000;
```

```
    case (NS)
```

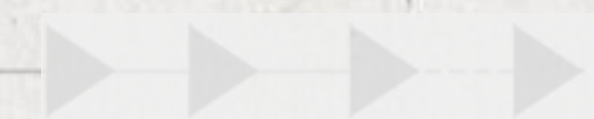
```
        IDLE: begin
```

```
            if (~i1)      begin {o1,o2,err} <=3'  b000; NS <=IDLE;  end
```


```
            if (i1 && i2)   begin {o1,o2,err} <=3'  b100; NS <=s1;    end
```

```
            if (i1 && ~i2)  begin {o1,o2,err} <=3'  b111 ; NS <=ERROR; end
```

```
        end
```





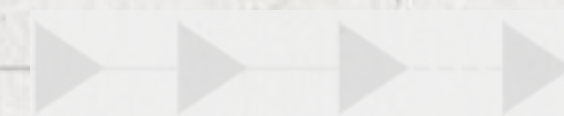
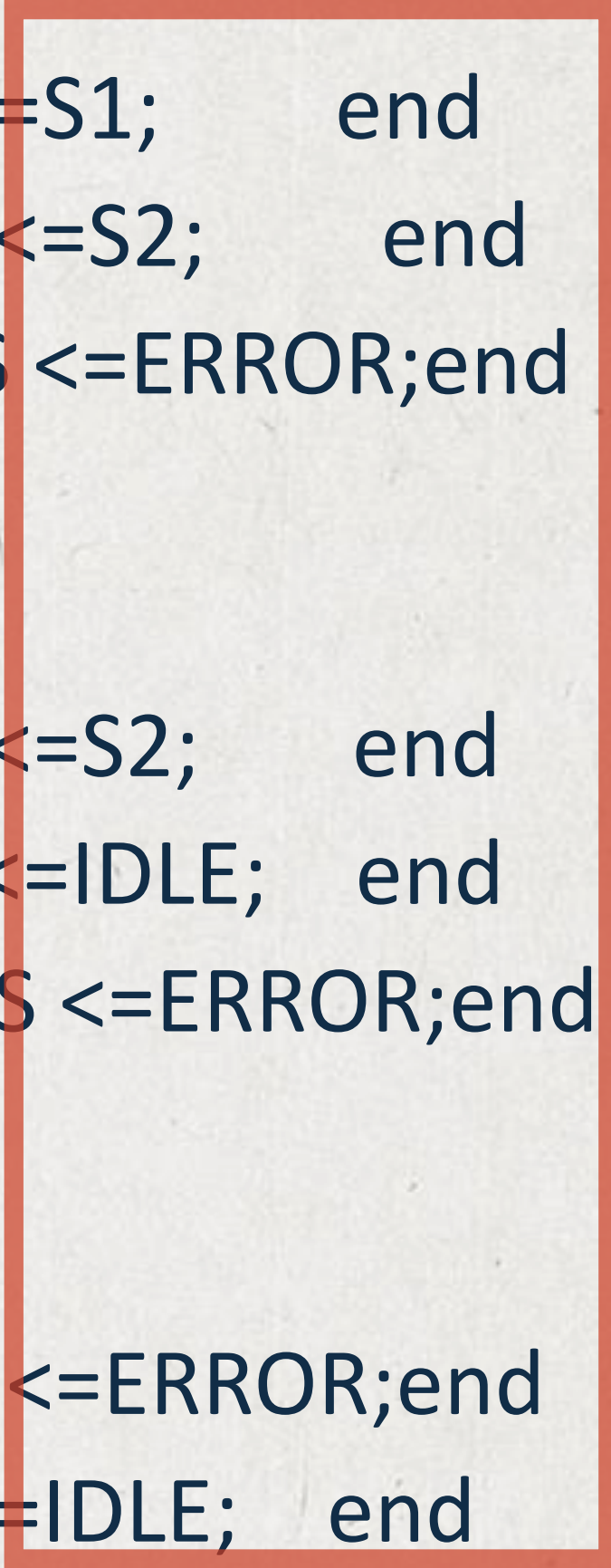
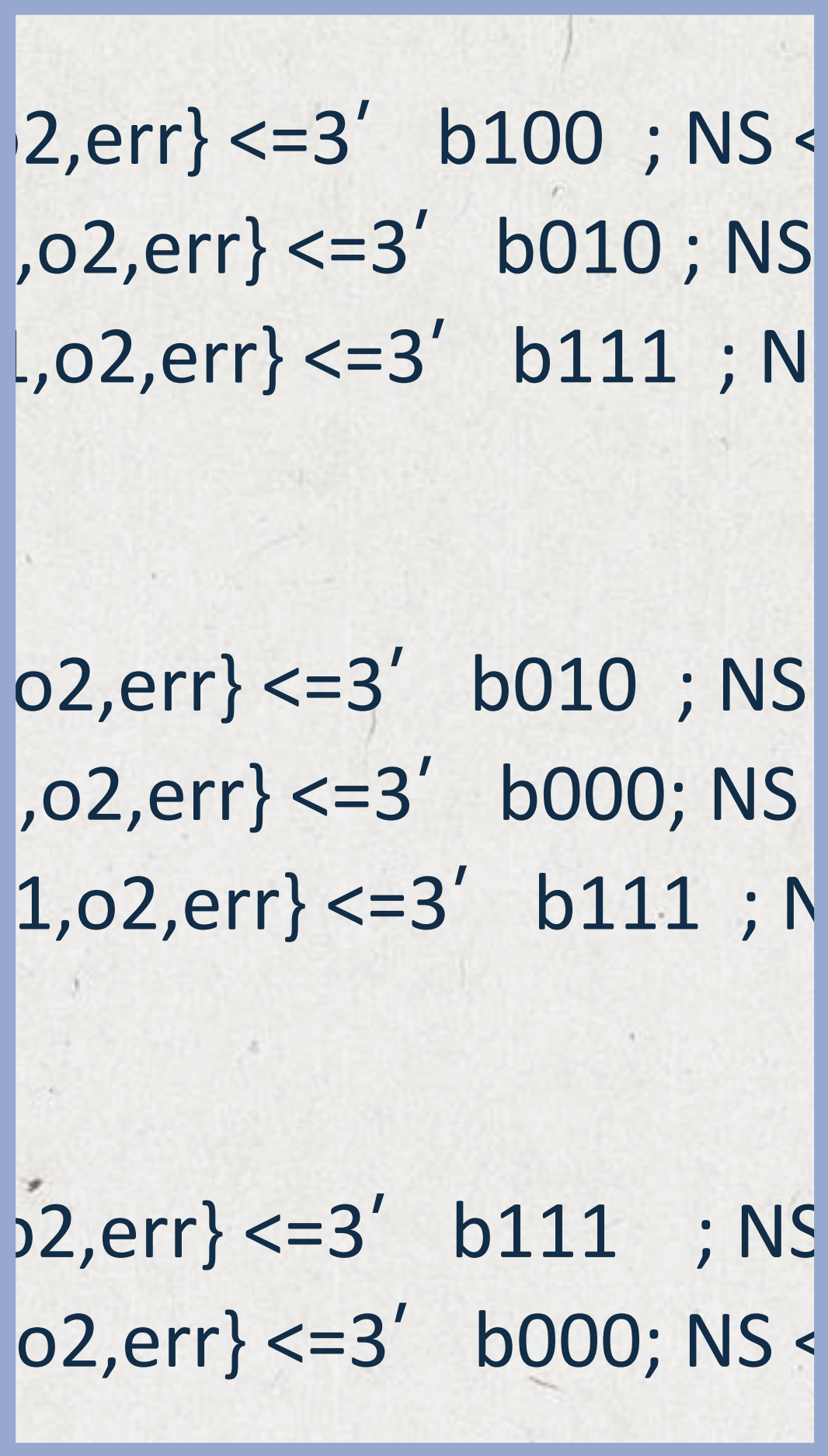


```
S1:  begin
      if (~i2)      begin {o1,o2,err} <=3'  b100 ; NS <=S1;    end
      if (i2 && i1)  begin {o1,o2,err} <=3'  b010 ; NS <=S2;    end
      if (i2 && (~i1)) begin {o1,o2,err} <=3'  b111 ; NS <=ERROR;end
    end

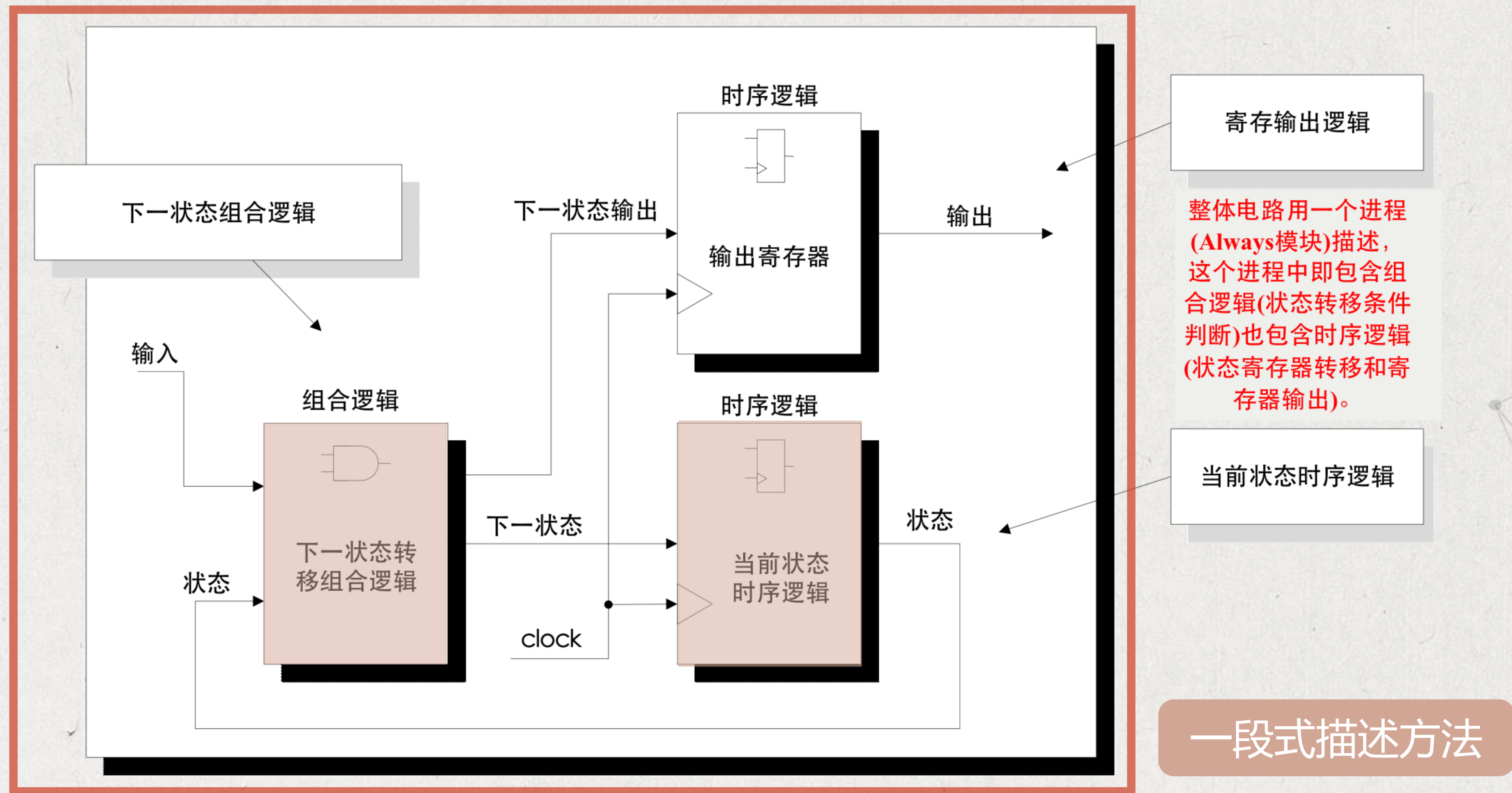
S2:  begin
      if (i2)      begin {o1,o2,err} <=3'  b010 ; NS <=S2;    end
      if (~i2 && i1) begin {o1,o2,err} <=3'  b000; NS <=IDLE;  end
      if (~i2 && (~i1)) begin {o1,o2,err} <=3'  b111 ; NS <=ERROR;end
    end

ERROR: begin
      if (i1)      begin {o1,o2,err} <=3'  b111  ; NS <=ERROR;end
      if (~i1)     begin {o1,o2,err} <=3'  b000; NS <=IDLE;  end
    end

endcase
end
```

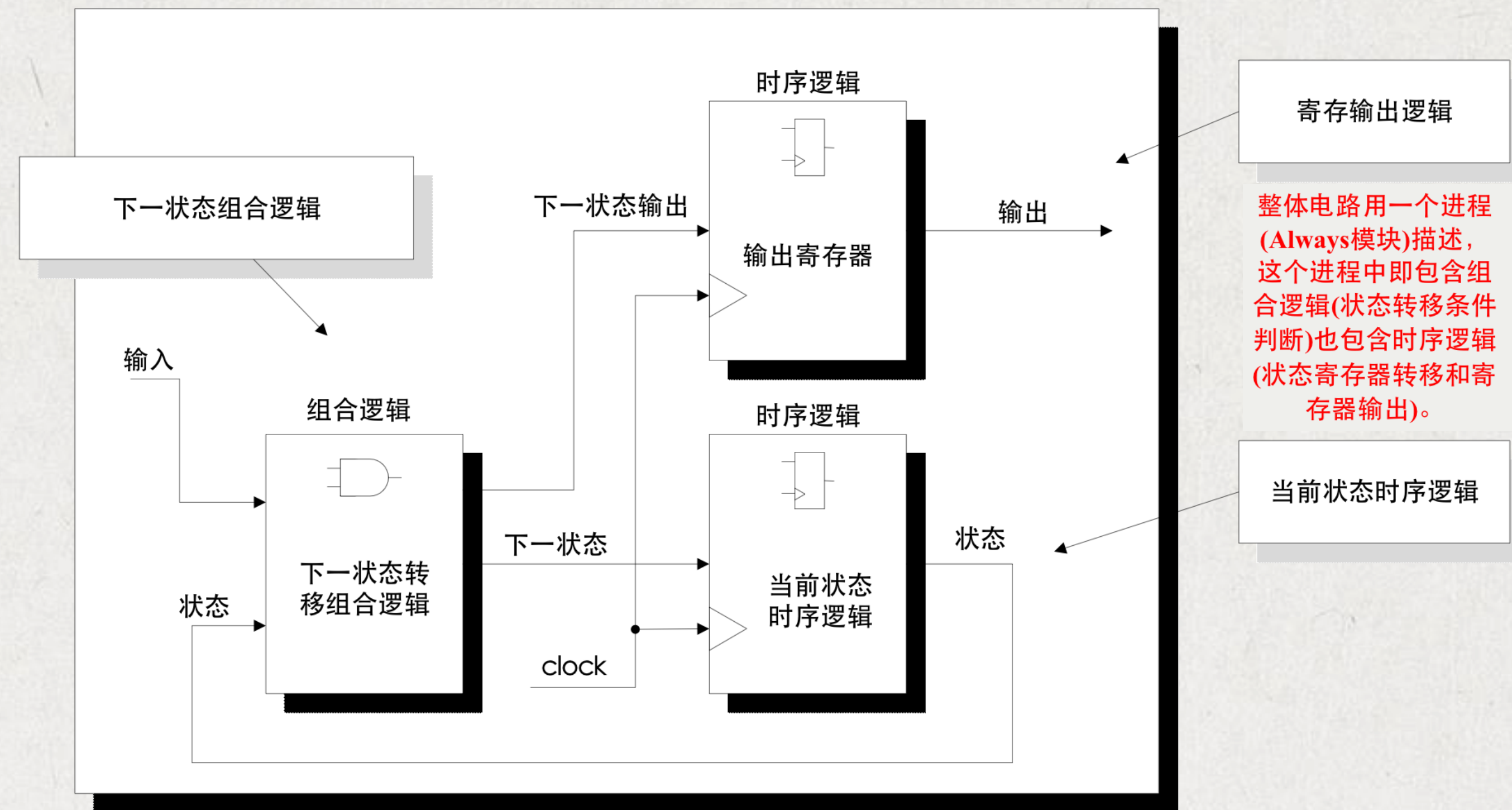






不符合将时序和组合逻辑分开描述的 Coding Style (代码风格)





- 在描述当前状态时要考虑下个状态的输出，整个代码不清晰
  - 不利于维护修改
  - 不利于附加约束
  - 不利于综合器和布局布线器对设计的优化



- 另外，这种描述相对于两段式描述比较冗长。

本例

非常简单的米勒型状态机

一段式比较冗长的缺点

状态机相对复杂

一段式

冗长大约  
80%到 150%左右

两段式

一段式 FSM 描述是不推荐的 FSM 描述方式

一定要避免

