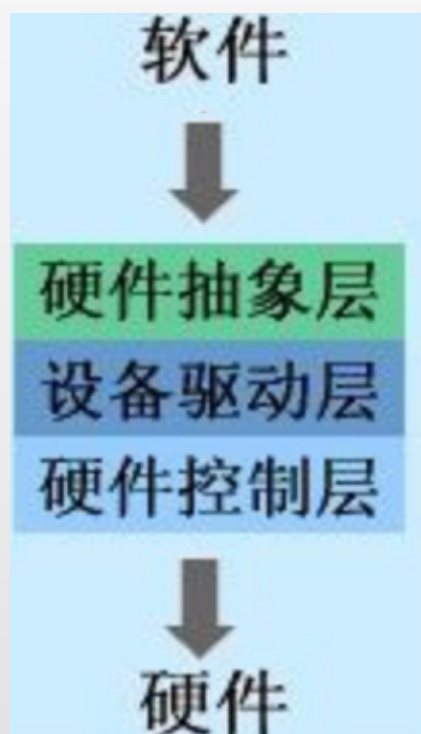


三、基本模拟程序的结构

- 根据模拟的方法不同，主要有两类基本的模拟程序：编译法和表格驱动法。早期的模拟程序为编译法，近代的程序大多为表格驱动法。因为驱动法更适合处理延迟，同时可以减少模拟时间。
- 软件是程序的编写，与硬件直接打交道的是硬件控制层。软件要实现对硬件的控制，必须经过硬件抽象层、设备驱动层、硬件控制层的桥接，这个桥接的实现就是编译。



硬件描述语言经过编译实现不同层级的硬件电路，可以是系统级电路，也可以是门级电路，独立于器件设计、与工艺无关。模拟仿真可以直接通过硬件控制层，调用不同的器件结构实现。设计者可以不懂硬件的结构，而进行独立的设计和模拟。



1. 编译法

例：一个三输入“与非”门（NAND3）可以编译成以下的程序。

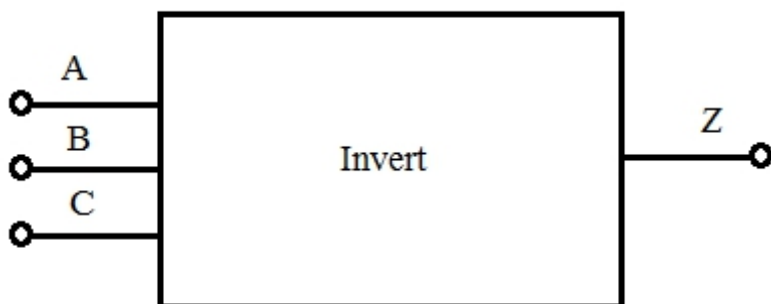
INPUT	A, B, C	
FETCH	A	取数A
COLLATE	B	A与B为AB
COLLATE	C	AB与C为ABC
INVERT		将ABC取反为 \overline{ABC}
STORE	Z	使 $Z = \overline{ABC}$

VHDL语言的实现

```
LIBRARY IEEE
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ex IS
    PORT(a, b, c: IN STD_LOGIC;
         z: OUT STD_LOGIC);
END ex;
ARCHITECTURE one OF IS
BEGIN
    z<=NOT(aANDbANDc);
END one;
```

Verilog语言实现

```
Module ex(a, b, c, z);
    input a, b, c;
    output z;
    assign z=~(a&&b&&c);
End module
```

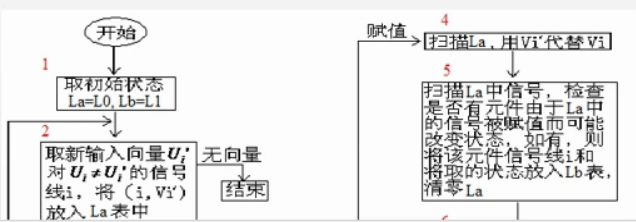


表格驱动面向事件单位延迟模拟 (鼠标滑过播放视频)

2 . 表格驱动面向事件单位延迟模拟

- 程序结构的思想：有两个表格La，Lb， $a \neq b$ ， $a, b \in \{0, 1\}$ ，La用来记录必须处理的活动元件以及它们的信号值Vi的输出在模拟时刻t应取的新值。Lb用来记录由于La中出现的事件（活动）而变成活动的元件j，及这些元件的输出在t+1时刻取的新值增加模拟事件单位（再加1），交换Lb和La，清空Lb内容，使用La处理活动元件，在Lb中又记录了下一部分活动电路元件的元件号j，这些元件在t+2时刻取新的输出，依次类推完成一个输入向量下的全部活动电路的模拟。输入新的向量，重复上述过程，直至完成全部输入时序（向量）。

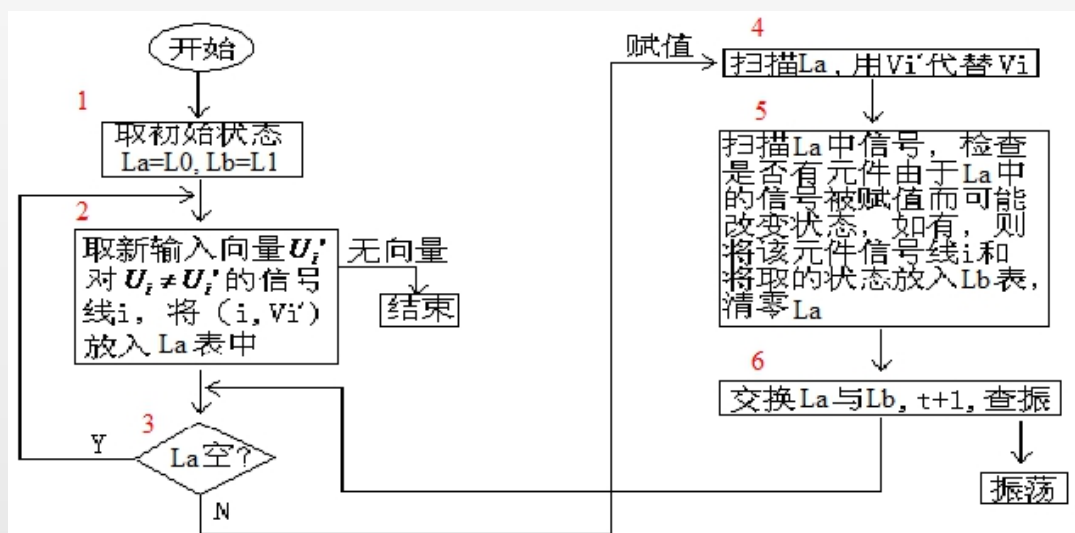
• 面向事件单位延迟、表格驱动算法流程



2. 表格驱动面向事件单位延迟模拟

- 程序结构的思想：有两个表格 L_a , L_b , $a \neq b$, $a, b \in \{0, 1\}$, L_a 用来记录必须处理的活动元件以及它们的信号值 V_i 的输出在模拟时刻 t 应取的新值。 L_b 用来记录由于 L_a 中出现的事件（活动）而变成活动的元件 j , 及这些元件的输出在 $t+1$ 时刻取的新值增加模拟事件单位（再加1），交换 L_b 和 L_a , 清空 L_b 内容，使用 L_a 处理活动元件，在 L_b 中又记录了下一部分活动电路元件的元件号 j , 这些元件在 $t+2$ 时刻取新的输出，依次类推完成一个输入向量下的全部活动电路的模拟。输入新的向量，重复上述过程，直至完成全部输入时序（向量）。

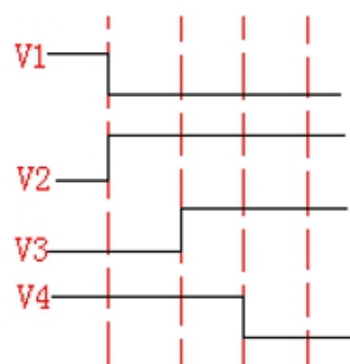
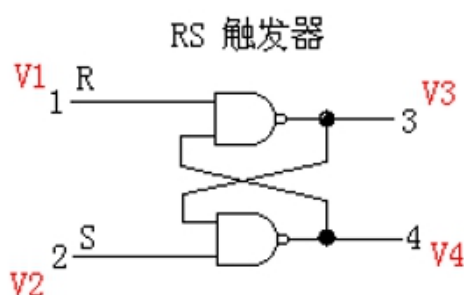
• 面向事件单位延迟、表格驱动算法流程



• RS触发器的具体算法实现

- 1. $L_a \leftarrow L_0, L_b \leftarrow L_1$
- 2. $V_1' \leftarrow 0, V_2' \leftarrow 1$
- 3.1 $L_a \neq \Phi$
- 4.1 $V_1 \leftarrow 0, V_2 \leftarrow 1$
- 5.1 $V_3' = 1 = V_2, V_4' = V_4 = \mu; L_b(3,1), L_a = \Phi$
- 6.1 交换 L_a, L_b
- 3.2 $L_a \neq \Phi$
- 4.2 $V_3 \leftarrow 1$
- 5.2 $V_4' = 0 \neq V_4; L_b(4,0), L_a = \Phi$
- 6.2 交换 L_a, L_b
- 3.3 $L_a \neq \Phi$
- 4.3 $V_4 \leftarrow 0$
- 5.3 $V_3' = 1 = V_3; L_b = \Phi, L_a = \Phi$
- 6.3 交换 L_a, L_b
- 3.1 $L_a = \Phi$
- 稳定，无向量变化，结束。

1. $L_a \leftarrow L_0, L_b \leftarrow L_1$	$V_1 \leftarrow V_1 \leftarrow V_1 \leftarrow V_1 \leftarrow \mu$	t=1
2. $V_1' \leftarrow 0, V_2' \leftarrow 1$	$L_a: (1,0), (2,1); 1号线网为0, 2号线网为1$	
3.1 $L_a \neq \Phi$		
4.1 $V_1 \leftarrow 0, V_2 \leftarrow 1$	$V_1、V_1$ 赋予新值	
5.1 $V_3' = 1 = V_2, V_4' = V_4 = \mu; L_b(3,1), L_a = \Phi$	产生将要变化的线网集合 $L_b(3,1)$	
6.1 交换 L_a, L_b		t=2
3.2 $L_a \neq \Phi$		
4.2 $V_3 \leftarrow 1$		
5.2 $V_4' = 0 \neq V_4; L_b(4,0), L_a = \Phi$		
6.2 交换 L_a, L_b		t=3
3.3 $L_a \neq \Phi$		
4.3 $V_4 \leftarrow 0$		
5.3 $V_3' = 1 = V_3; L_b = \Phi, L_a = \Phi$		
6.3 交换 L_a, L_b		t=4
3.1 $L_a = \Phi$	稳定，无向量变化，结束	



§3.测试码生成

- 逻辑模拟的两个主要作用：一是评价系统；二是分析故障。
- 分析故障必然需要将测试信号加在被测电路上，然后测量输出响应与电路应有的功能是否一致。这样的输入信号和应有的输出量称为测试码或测试序列（测试矢量）。
- 简单的说，用于测试电路的一组输入/输出信号就叫测试码或测试矢量。



一、测试方法

① 完全测试

有两种说法：(1)对应输入的所有可能的信号进行测试，共有 2^N 个测试输入， N 为输入端数，它可以测到包括冗余故障以外的所有节点，但会出现一个节点被几次测试的问题；(2)对除冗余故障以外的所有节点的故障测试，对应每个故障产生一组测试矢量。第二种方法的测试矢量小于等于第一种方法的测试矢量，即若用 N_1 表示第一种方法的测试矢量， N_2 表示第二种测试矢量数，有 $N_1 \geq N_2$ 。

② 功能测试

只就电路功能进行测试，只要电路满足功能的完成即可。显然会出现有些节点重复测试、有些节点没被测试的情况。通常对ASIC采用功能测试，而对CPU之类的芯片采用完全测试。

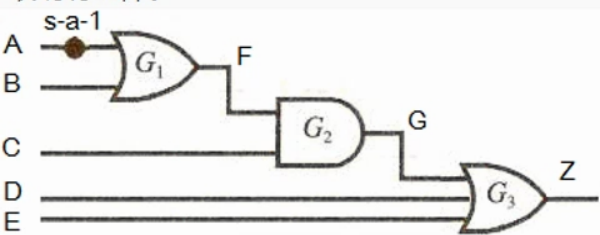


故障的检测

(鼠标滑过播放视频)

二、故障的检测

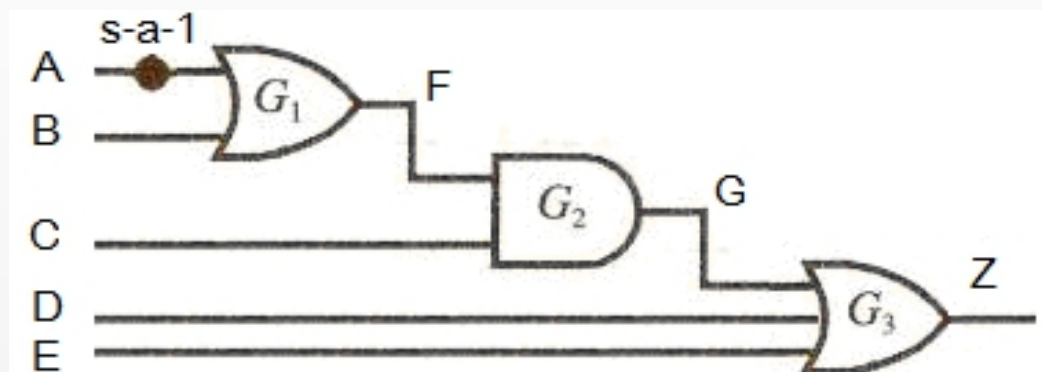
设节点A有故障As-a-1：
为使A的故障在Z点有反映，必须使所有通路全打开，让 $Z=A$ 。 $\because B=0, F=A, C=1, G=A, D=E=0$ 时 $Z=A$ ，所以要使 $Z=A$ ，那么为反映节点A的故障 $A=0$ ，必须 $B=D=E=0, C=1$ 。正常情况，若 $A=0$ ，则 $Z=0$ 。而有故障情况， $A=0, Z=1$ ，表明有故障。



若G有故障“1”，为反映G，让 $Z=G$ ，则 $D=E=0$ 。为反映 $G=1$ 故障，门2 输出 $FC=0$ ，则 $C=0$ 或 $F=0$ 。则 $A=0, B=0$ 即对应输入A, B, C, D, E为 00×00 或 $\times\times000$ 。共有5个输入序列能反映 $G=1$ 故障，即00000, 00100, 01000, 10000, 11000。

二、故障的检测

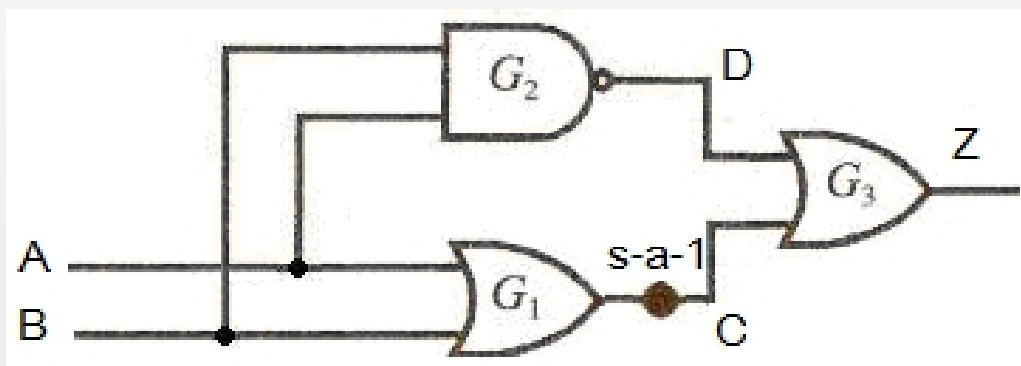
- 设节点A有故障As-a-1:
- 为使A的故障在Z点有反映, 必须使所有通路全打开, 让 $Z=A$ 。∵ $B=0$, $F=A$, $C=1$, $G=A$, $D=E=0$ 时 $Z=A$, 所以要使 $Z=A$, 那么为反映节点A的故障 $A=0$, 必须 $B=D=E=0$, $C=1$ 。正常情况, 若 $A=0$, 则 $Z=0$ 。而有故障情况, $A=0$, $Z=1$, 表明有故障。

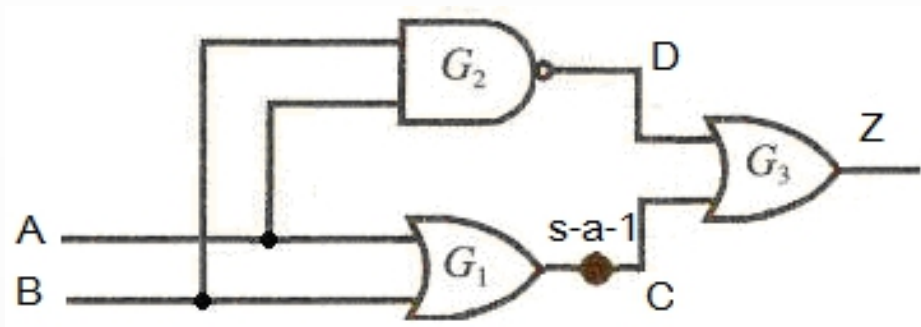


- 若G有故障“1”, 为反映G, 让 $Z=G$, 则 $D=E=0$ 。为反映 $G=1$ 故障, 门2 输出 $FC=0$, 则 $C=0$ 或 $F=0$ 。则 $A=0$, $B=0$ 即对应输入A, B, C, D, E为 00×00 或 $\times \times 000$ 。共有5个输入序列能反映 $G=1$ 故障, 即00000, 00100, 01000, 10000, 11000。



- 要检测C点的s-a-1故障，C点要为0，对应 $A+B=0$ ，必须 $A=B=0$ ，但 $A=B=0$ ，则 $D=1$ ，所以没有一组测试矢量能检测到这个故障，或者说该点故障不会影响电路功能。因为除输入为 $A=B=0$ 以外的输入使C点都应为1，对应 $Z=1$ ，而C点应为0时，D点又会为1，Z仍为1。
- 出现这种情况，电路对故障C是冗余的，称为故障冗余，这种电路叫冗余电路。该电路相当于 $Z=1$ ，在测试中不用再考虑。通常这种电路可以用一固定信号将其固化，所以叫冗余的。

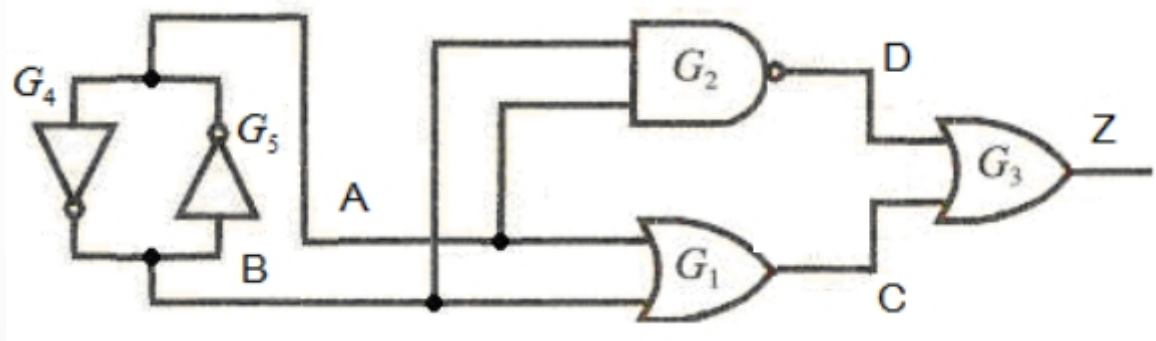




A	B	C	D	Z
0	0	0	1	1
0	μ	μ	1	1
0	1	1	1	1
μ	0	μ	1	1
μ	μ	μ	μ	μ
μ	1	1	μ	1
1	0	1	1	1
1	μ	1	μ	1
1	1	1	0	1

- 电路的三值真值表可以看出，这仍然是一个静态1冒险的电路，若需要固定输出1电平，就要避免输出 μ 状态，即A、B不允许同时改变状态。





A	B	C	D	Z
0	0	0	1	1
0	μ	μ	1	1
0	1	1	1	1
μ	0	μ	1	1
μ	μ	μ	μ	μ
μ	1	1	μ	1
1	0	1	1	1
1	μ	1	μ	1
1	1	1	0	1

- 若是在A、B端接一个双稳态电路，那么在电路上电的初始状态输出 μ 状态之后，A、B不允许同时为0或1状态，从电路的三值真值表可以看出，这个电路避免了成为一个静态1冒险的电路，可以实现固定输出1电平。



三、故障模型

- 物理故障
开路、短路、老化
- 逻辑故障
固定型故障、桥接故障

• 常用故障模型

故障模型	说 明
单固定型故障（SSAF）	一条线网总为0或1
多固定型故障（MFF）	两条或多条线网有固定值，但值不一定相同
桥接故障（BF）	两条或多条绝缘线网的连接
固定开路故障（SOP）	晶体管总处于开路状态
固定导通故障（SON）	晶体管总处于导通状态
时延故障（Delay Fault）	电路因路径时延引起的故障
间歇故障（Valve Fault）	电路中的信号值时常不正确，是由内部参数老化引起的故障，会持续到永久失效为止
瞬时故障（Transient Fault）	因耦合干扰引起错误的信号值。包括电容耦合或电感耦合，也有内部源和外部源及粒子辐射的原因。



故障模型

(鼠标滑过播放视频)

固定型故障 (SAF-Stuck At Fault)

- 当某个信号线的信号被固定在某个逻辑电平的电位上，即为固定型故障。如果该线（或该点）固定在逻辑高电平上，则称为固定1故障（stuck-at-1），记为（s-a-1）；若信号固定在逻辑低电平上，则称为固定0故障（stuck-at-0），记为（s-a-0）。
- 固定型故障模型在实际应用中用的最普遍，约占发生故障总数的90%。

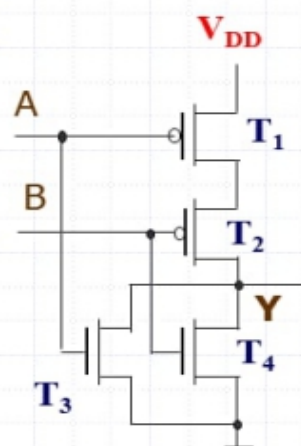
固定开路故障（**stuck-open**）

固定型故障 (SAF-Stuck At Fault)

- 当某个信号线的信号被固定在某个逻辑电平的电位上，即为固定型故障。如果该线（或该点）固定在逻辑高电平上，则称为固定1故障（stuck-at-1），记为（s-a-1）；若信号固定在逻辑低电平上，则称为固定0故障（stuck-at-0），记为（s-a-0）。
- 固定型故障模型在实际应用中用的最普遍，约占发生故障总数的90%。

固定开路故障（stuck-open）

A	B	Y	Y(A:s-op)	Y(B:s-op)	Y(V _{DD} :s-op)
0	0	1	1	1	Y _t
0	1	0	0	Y _t	0
1	0	0	Y _t	0	0
1	1	0	0	0	0



时滞故障（delay fault）模型

- ◆ 时滞故障是一种动态故障，这种故障在低频时工作正常，随着信号频率的升高，元件的延迟时间有可能超过规定的值，从而导致时序配合上的错误，使电路的功能出错，这种故障称为时滞故障。



桥接故障


(BF-Bridging Fault)


- 当两条以上信号线短路在一起并建立逻辑时，就会产生桥接故障。若故障涉及 n 个线网 ($n > 2$)，就称为 n 重桥接故障，否则称为简单桥接故障。一般在芯片原始的输入端口比较容易发生多重桥接故障。



①元件输入端之间的桥接故障

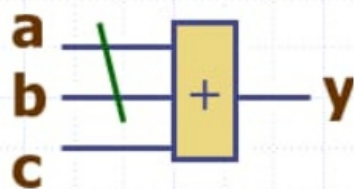
- 这种故障将导致“线与”、“线或”的效果，从而改变电路的逻辑关系。

正确或逻辑			<div>x_1 x_2 x_3</div> 	线与故障				
X_1	X_2	F_1		X_1	X_2	F_2		
0	0	0		\Rightarrow	0	0	0	
0	1	1			0	1	0	
1	0	1		\Rightarrow	1	0	0	
1	1	1			1	1	1	
真值表(a)				真值表(b)			真值表(c)	
$F_1 = X_1 + X_2$						$F_2 = X_1 X_2$		

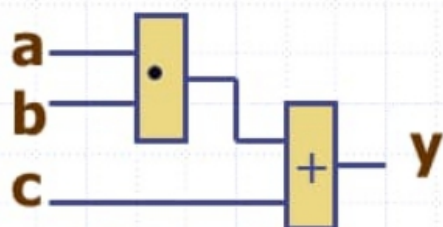
正确与逻辑			<div><div>X₁</div><div>X₂</div><div>X₃</div></div>	线或故障				
X ₁	X ₂	F ₁		X ₁	X ₂	F ₂		
0	0	0		0	0	0		
0	1	0			0	1		
1	0	0		1	0	1		
1	1	1			1	1		
真值表(a)			真值表(b)			真值表(c)		
$F_1 = X_1 X_2$						$F_2 = X_1 + X_2$		



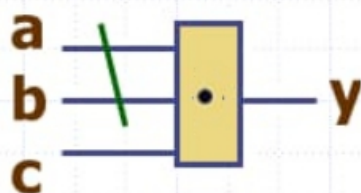
◆ 桥接故障



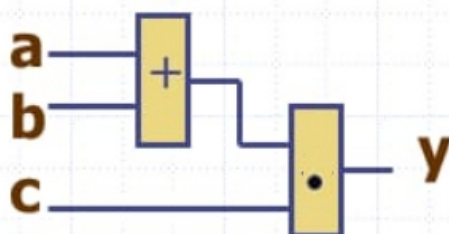
原电路



正逻辑等效
故障电路



原电路

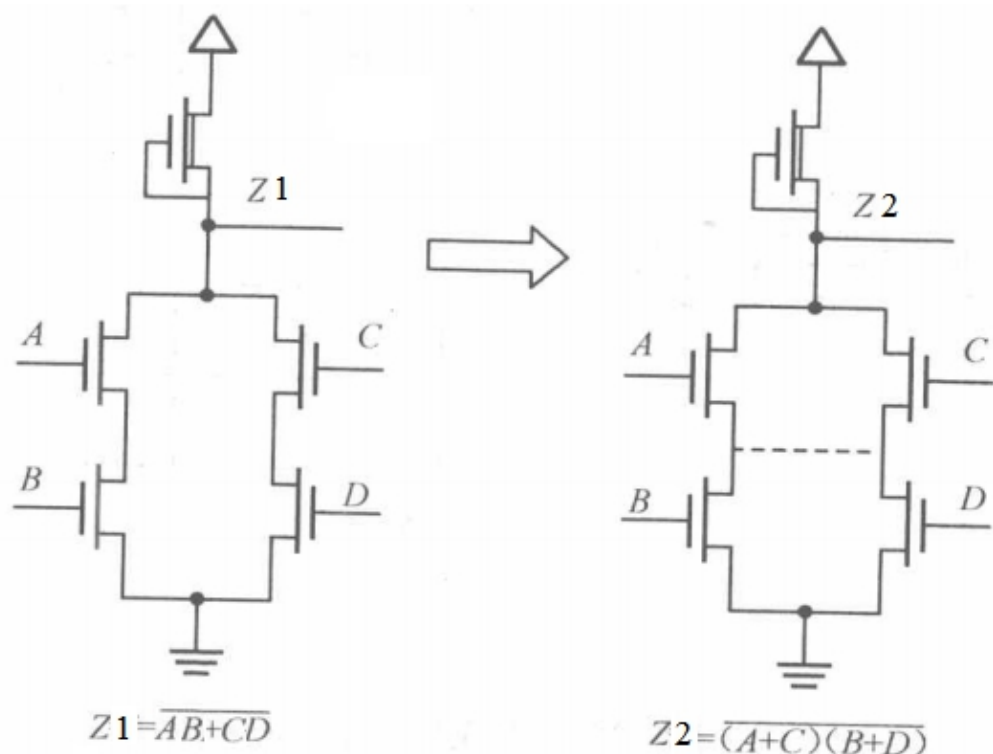


负逻辑等效
故障电路

固定故障使电路的逻辑值出错，但不会改变电路的结构。然而桥接故障不但改变电路的逻辑值，而且也可能改变电路的拓扑结构。



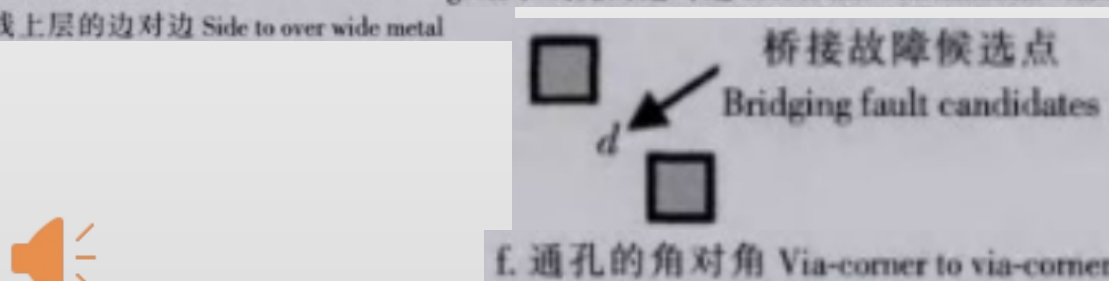
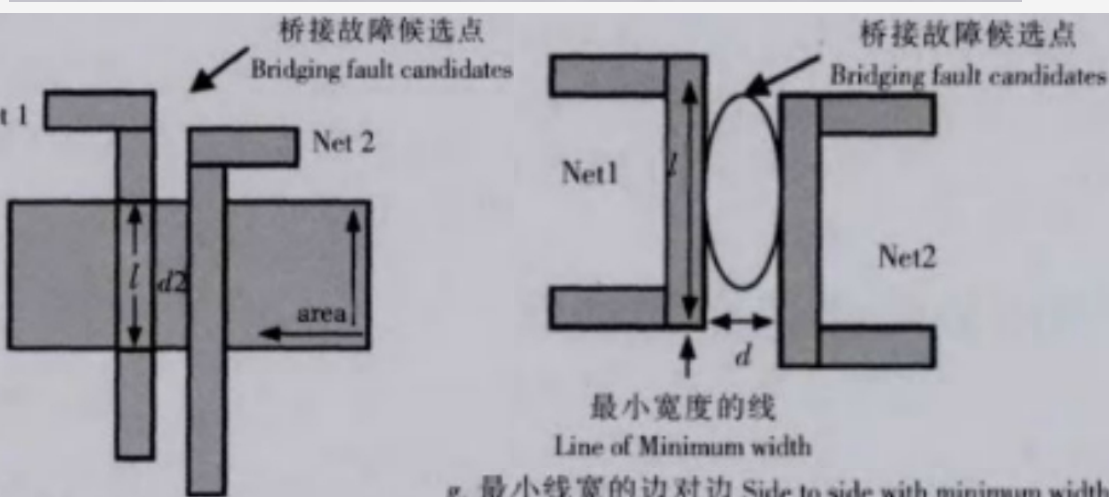
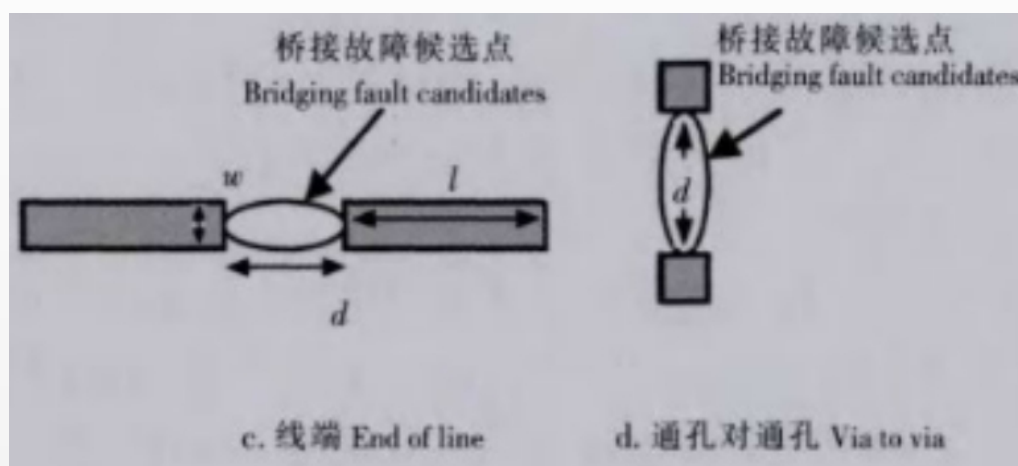
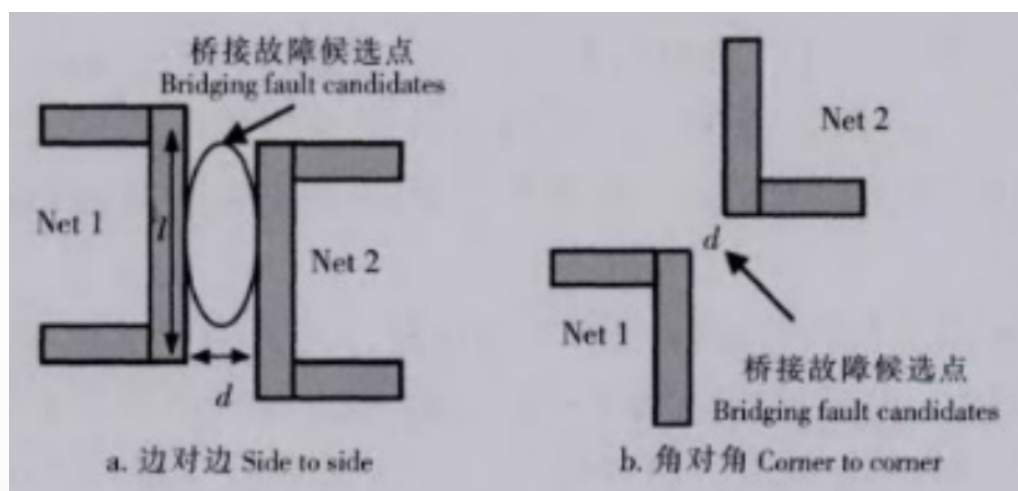
桥接引起的功能变化



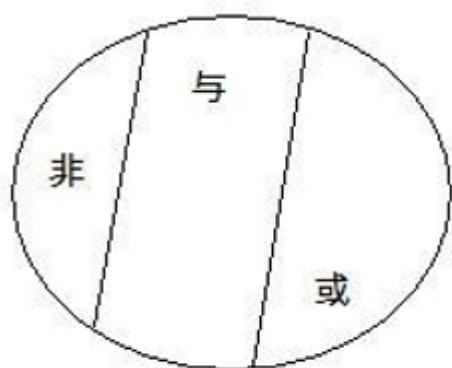
状态号	A	B	C	D	Z1	Z2	逻辑
1	0	0	0	0	1	1	T
2	0	0	0	1	1	1	T
3	0	0	1	0	1	1	T
4	0	0	1	1	0	0	T
5	0	1	0	0	1	1	T
6	0	1	0	1	1	1	T
7	0	1	1	0	1	0	F
8	0	1	1	1	0	0	T
9	1	0	0	0	1	1	T
10	1	0	0	1	1	0	F
11	1	0	1	0	1	1	T
12	1	0	1	1	0	0	T
13	1	1	0	0	0	0	T
14	1	1	0	1	0	0	T
15	1	1	1	0	0	0	T
16	1	1	1	1	0	0	T

错误发生率 = $(N-T) / N = (16-14) / 16 = 1/8$





- 通过以上的分析可以发现，逻辑本身是严格的封闭集合空间，线与和线或均是在正确的原逻辑上附加的一个故障逻辑，打破了原来与、或逻辑相对封闭的逻辑集合的区分，桥接了与、或两个逻辑区间，获得了额外的逻辑状态。



封闭的逻辑集合



桥接的逻辑集合



②反馈桥接

- 桥接故障的另一种重要的故障方式是桥接线跨接在输入和输出端，这种情况称为反馈桥接故障。这种故障将组合电路转变为时序电路，并增加了时序电路的状态，将使电路出现振荡，或者反馈。

