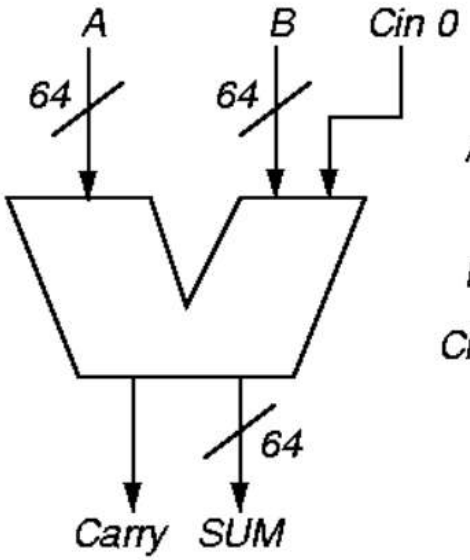
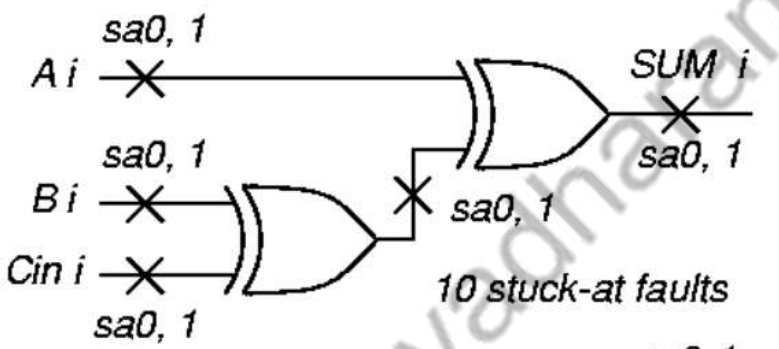
Testability of VLSI

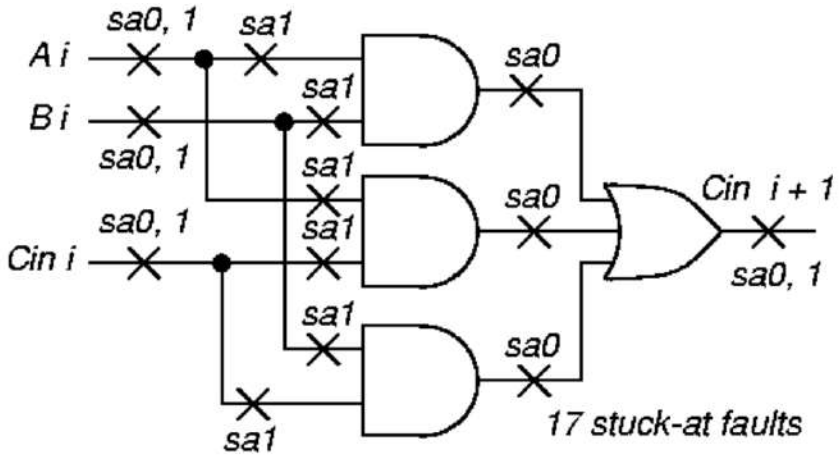
Lecture 3: Fault Collapsing

By Dr. Sanjay Vidhyadharan

Functional Versus Structural Testing







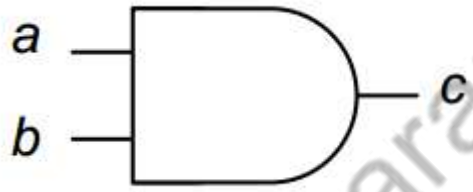
Functional Test

Test Vectots $2^{129} = 6.8 * 10^{38}$

Required time with Clock of $1\ GHz \approx 22\ years$

Structural Test vectors required $64 * (10 + 17)$

Single Stuck-at faults



How many Fault Sites ? 3

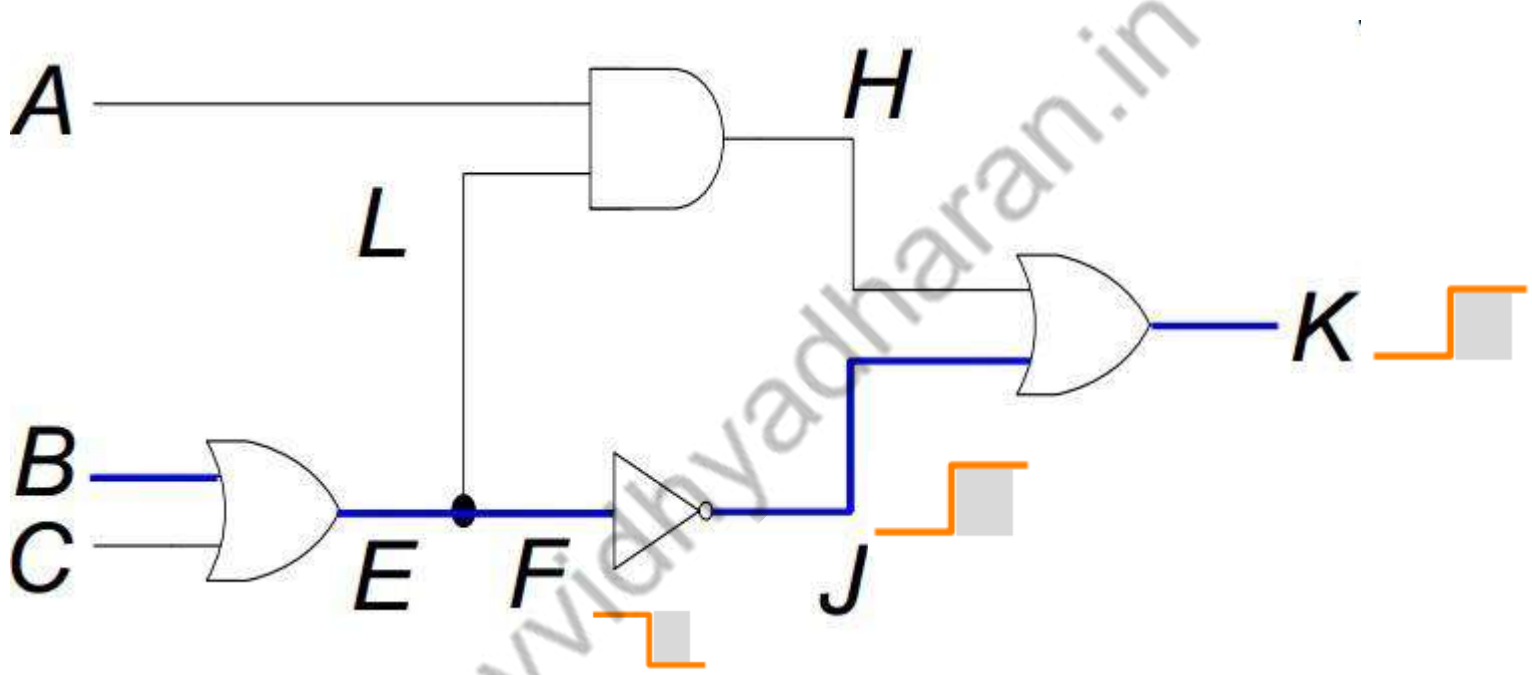
How many Fault ? **Total six faults: a/0, a/1, b/0, b/1, c/0, c/1**

Minimum test length for 100% SSF fault coverage ? **3**

Input	Fault-free	Faulty Output Value with SSF					
a b	Output	*a*/0	*a*/1	*b*/0	*b*/1	*c*/0	*c*/1
0 0	0	0	0	0	0	0	**1**
0 1	0	0	**1**	0	0	0	**1**
1 1	1	**0**	1	**0**	1	**0**	1
1 0	0	0	0	0	**1**	0	**1**

No requirement to exactly identify which fault. Entire gate is to be discarded

Delay faults

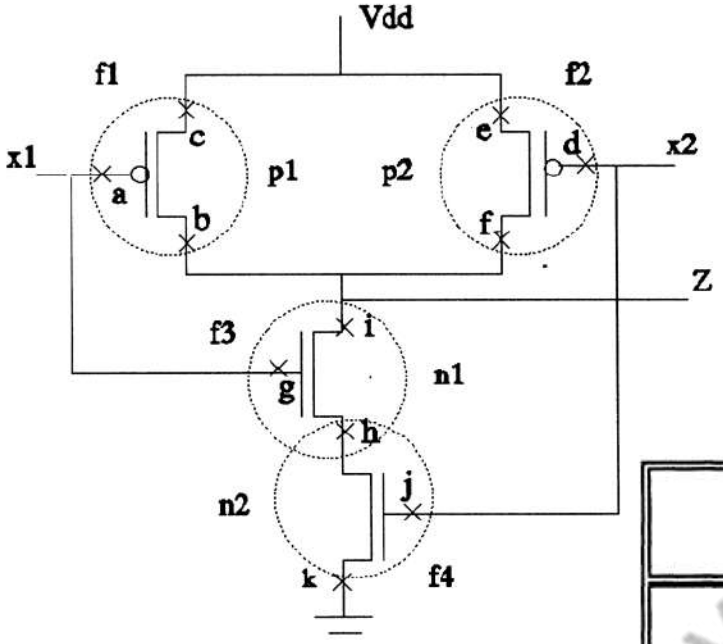


How Many Paths ? 5 paths: {AHK, BELHK, BEFJK, CELHK, CEFJK}

How Many set of test vectors ? 10 sets

Test vector to detect delay fault at F ? 001-000

Transistor faults



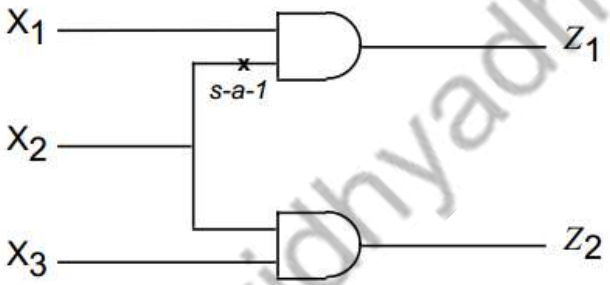
Two-pattern Tests for Stuck-open Faults

Faults	f1 (a,b,c)	f2 (d,e,f)	f3, f4 (g,h,i,j,k)
Initialization Vector (T1)	x1, x2 1 1	x1, x2 1 1	x1, x2 0 0 0 1 1 0
Test Vector (T2)	0 1	1 0	1 1

Automation available to optimize Stuck-at and Stuk-open Fault

Fault Detection

- **A test (vector) *t* detects a fault *f* iff** $z(t) \oplus z_f(t) = 1$
  - ***t* detects *f* <=>** $z_f(t) \neq z(t)$
- **Example**



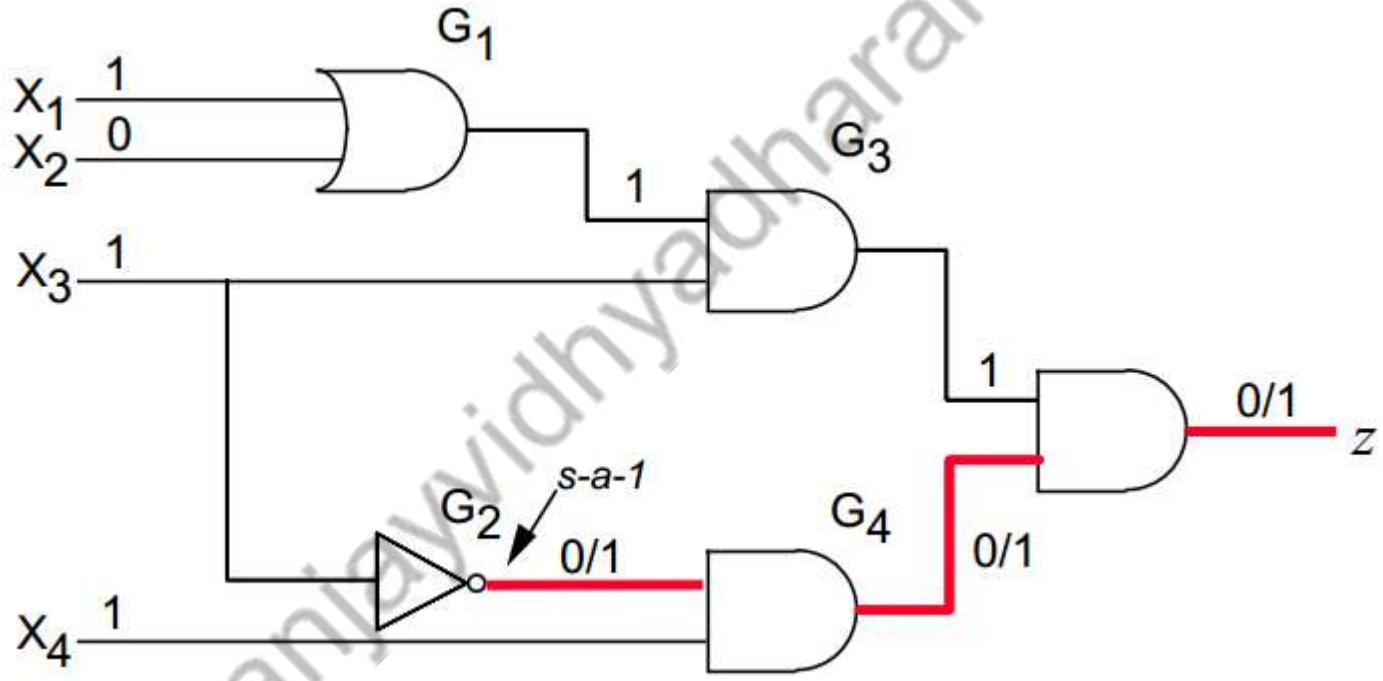
$Z_1 = X_1X_2$ $\quad Z_2 = X_2X_3$

$Z_{1f} = X_1$ $\quad Z_{2f} = X_2X_3$

The test 001 detects *f* because $z_1(001)=0$ while $z_{1f}(001)=1$

Fault Sensitization

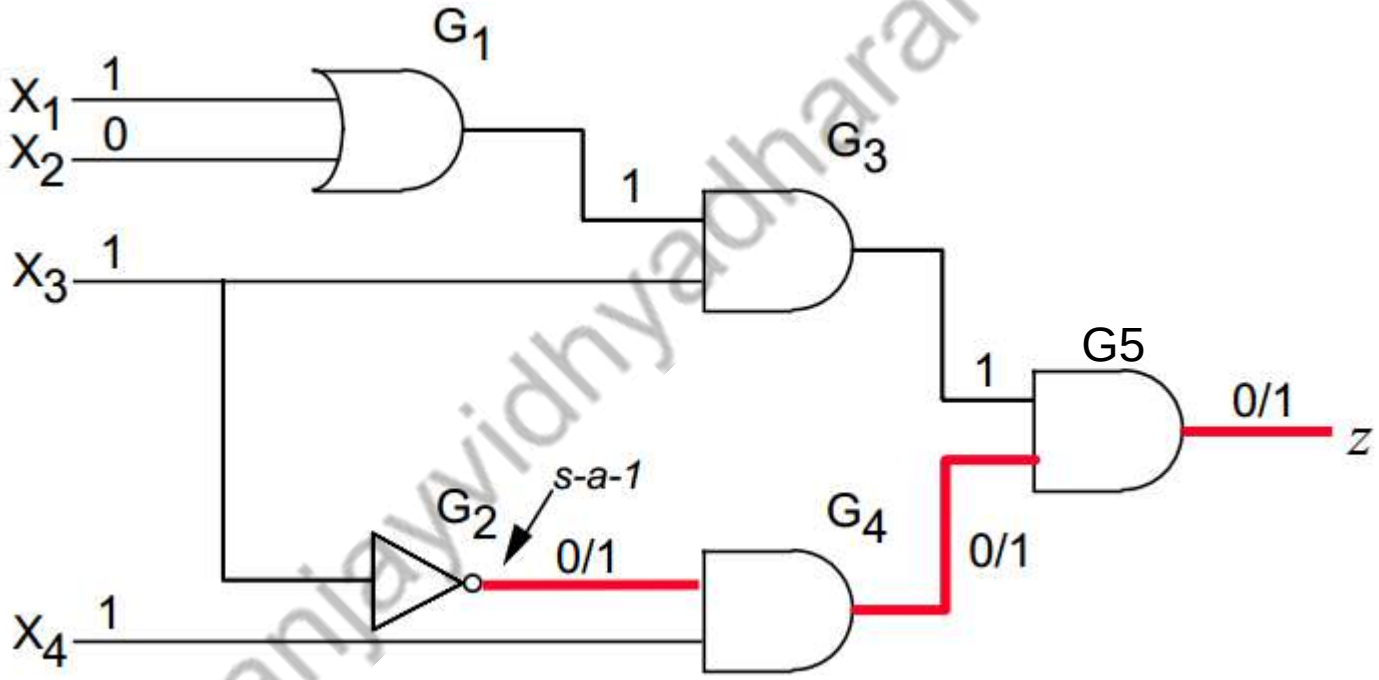
1. Fault Sensitization: We need to choose a test vector that activates the fault site with complementary signal



$X_3 = 1$

Fault Propagation

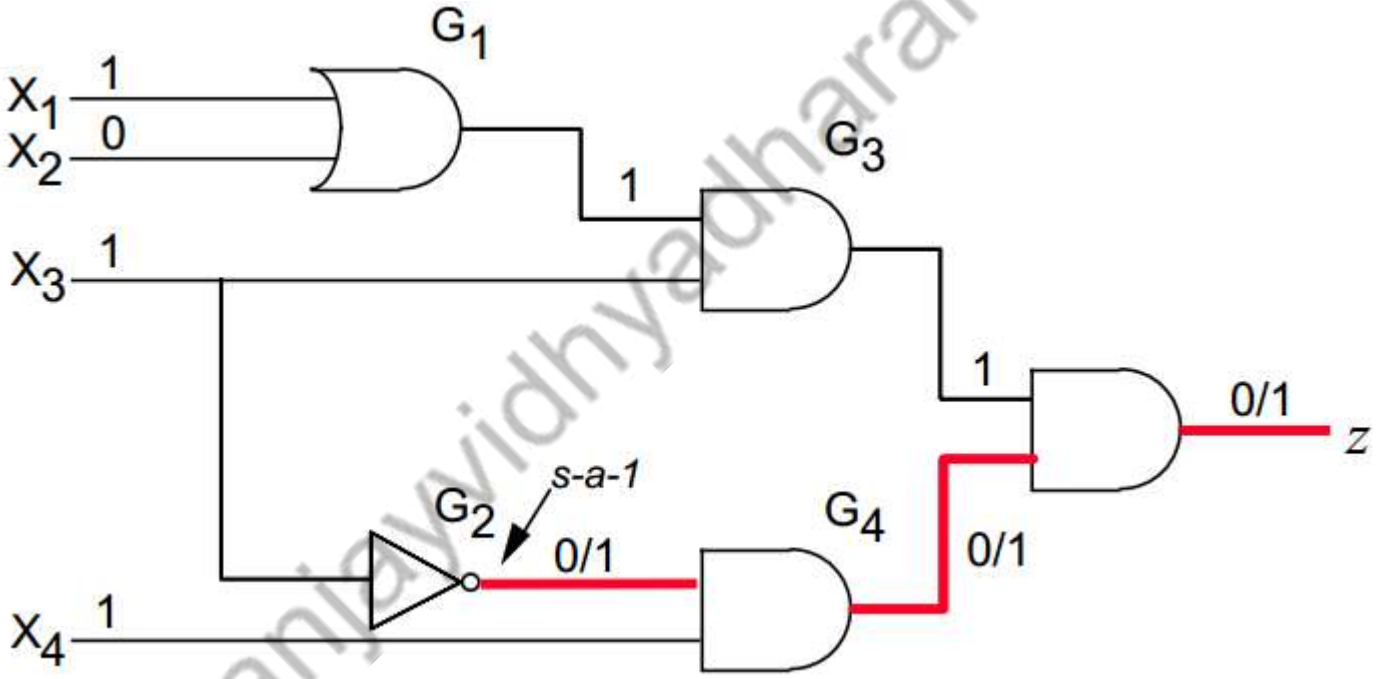
2. Fault Propagation: We need to chose a suitable path for propagate the fault to a primary output.



G4 >> G5>> z

Fault Justification

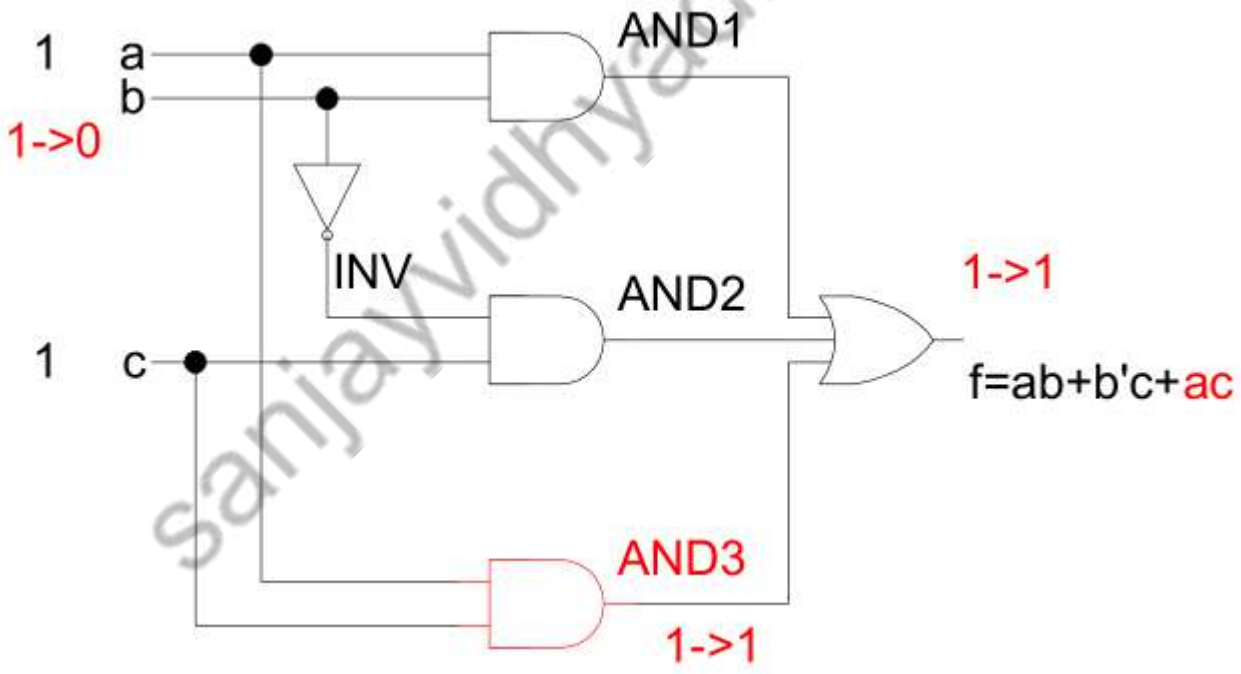
3. Fault Justification: We need to work from output to input to assign test vectors to primary inputs.



Fault Detectability

A fault *f* is said to be detectable if there exists a test *t* that detects *f* ; otherwise, *f* is an undetectable fault

E.g. And 3 S-a-0 is not detectable



Fault Coverage

Complete detection test set: A set of tests that detect any detectable faults in a class of faults

The quality of a test set is measured by fault coverage

Fault coverage: Fraction of faults that are detected by a test set

>95% - 99.9% is typically required

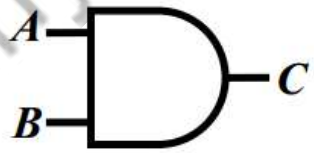
Fault Equivalence

Fault equivalence.

1. *Two faults of a Boolean circuit are called* equi*valent iff they transform the circuit such that the two faulty circuits have identical output functions.*

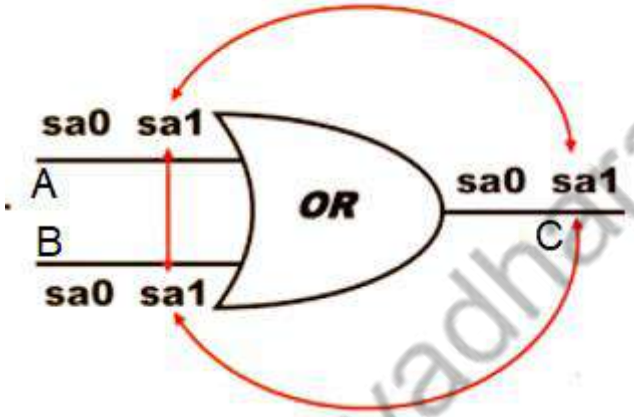
2. *Equivalent faults are also called indistinguishable and have exactly the same set of tests.*

Faults *f* and *g* are *functionally equivalent* (or simply *equivalent*) if faulty outputs of them are identical for *all* test patterns



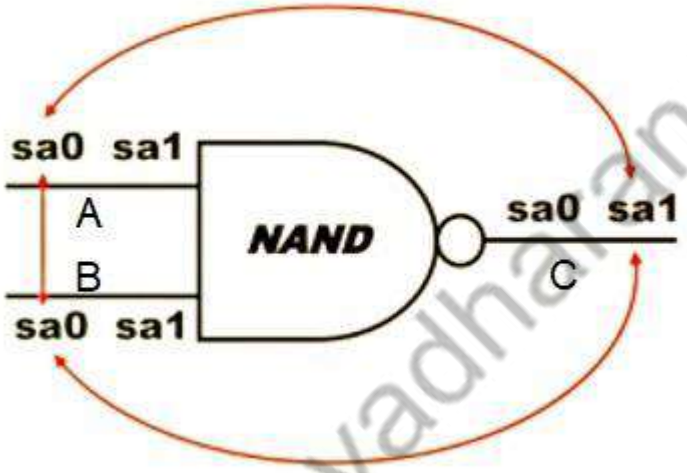
Input		Output						
A	*B*	good	*A/0*	*C/0*	*B/0*	*A/1*	*C/1*	*B/1*
0	0	0	0	0	0	0	1	0
0	1	0	0	0	0	1	1	0
1	0	0	0	0	0	0	1	1
1	1	1	0	0	0	1	1	1

Fault Equivalence



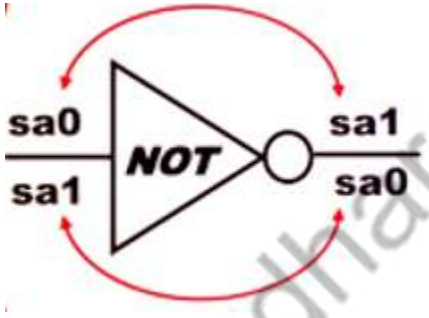
Input		Output						
A	B	Good	A/0	B/0	C/0	A/1	B/1	C/1
0	0	0	0	0	0	1	1	1
0	1	1	1	0	0	1	1	1
1	0	1	0	0	0	1	1	1
1	1	1	1	1	0	1	1	1

Fault Equivalence



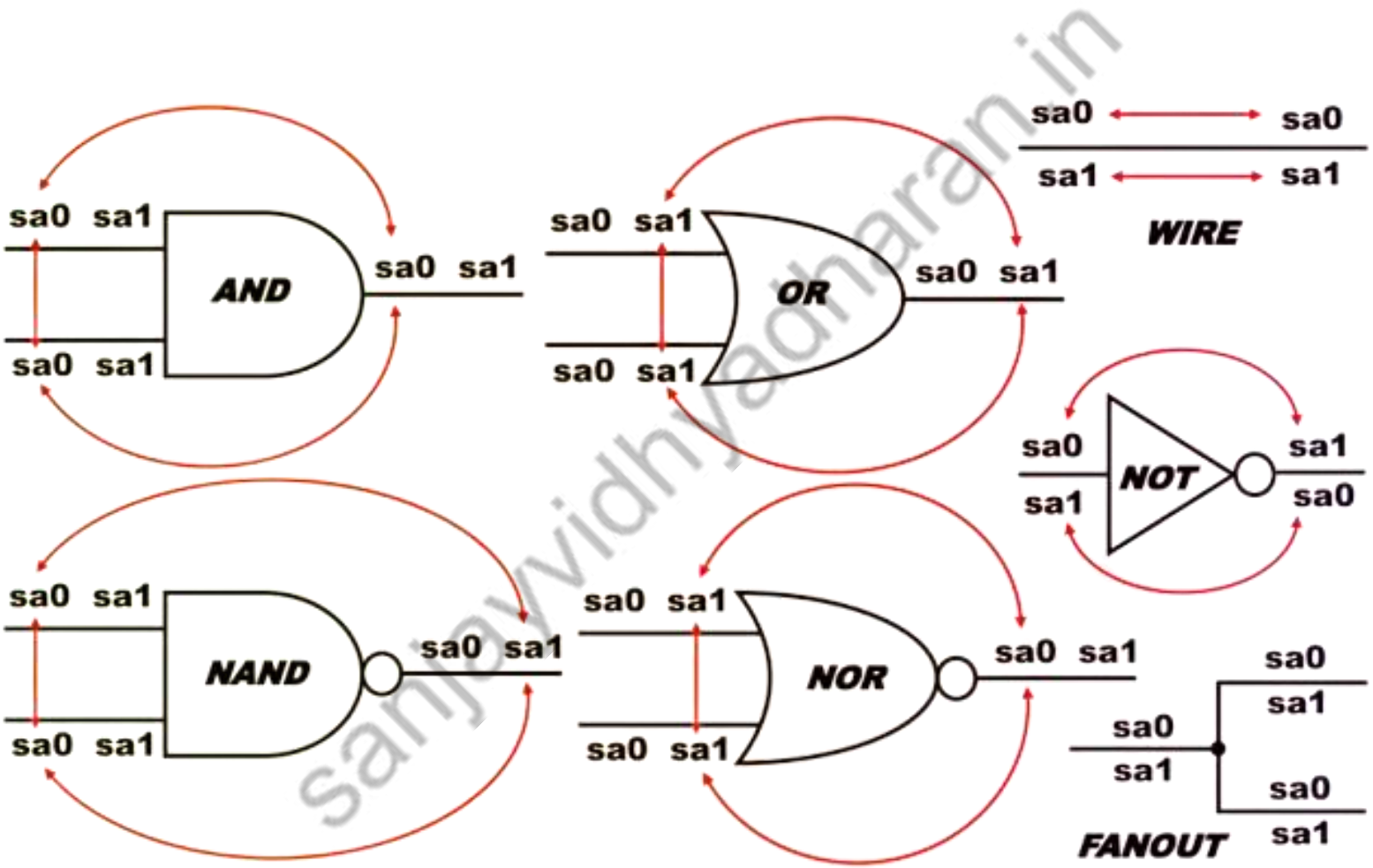
Input		Output						
A	B	Good	A/0	B/0	C/0	A/1	B/1	C/1
0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	1	1
1	0	1	1	1	0	0	0	1
1	1	0	1	1	0	1	1	1

Fault Equivalence



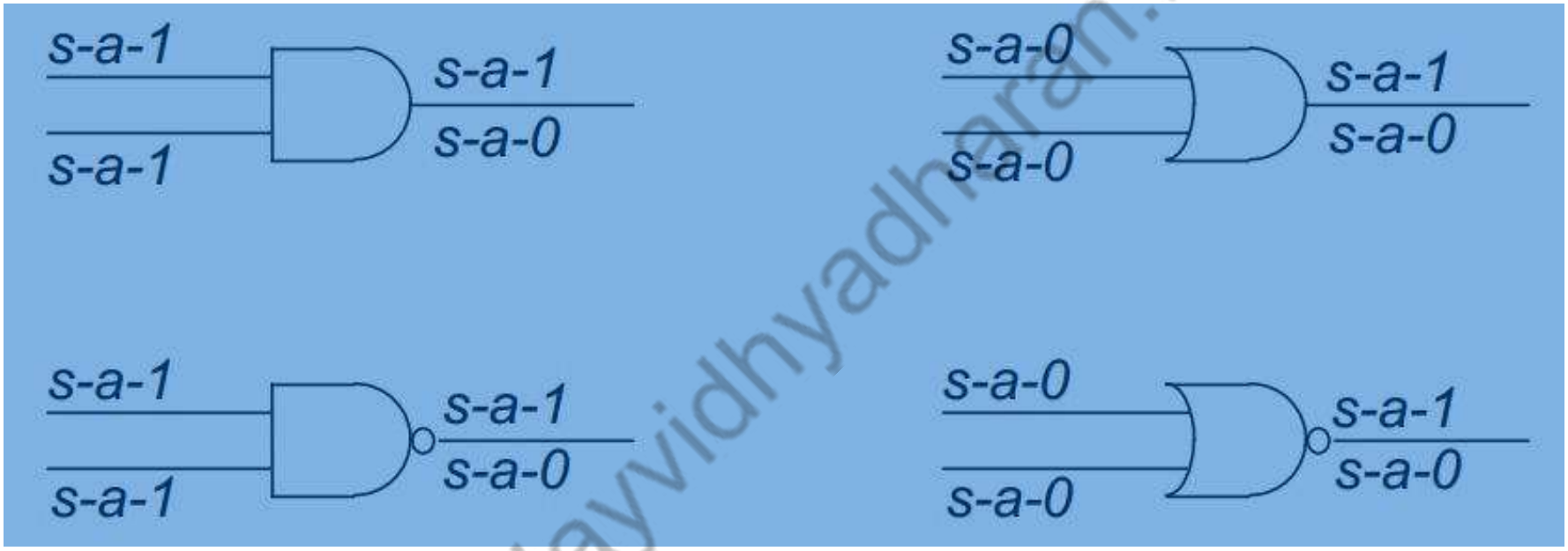
Input	Output				
	Good	in/0	in/1	out/0	out/1
0	1	1	0	0	1
1	0	1	0	0	1

Fault Equivalence



Equivalence Fault Collapsing

***n+2* instead of *2n+2* faults need to be considered for an *n*-input gate**



Why Equivalence Fault Collapsing (EFC)?

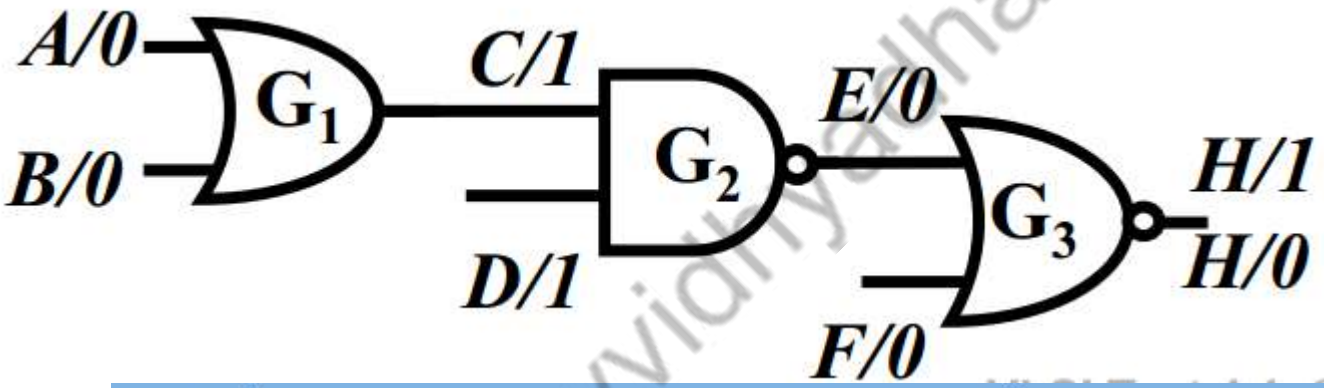
- Reduce number of faults so that
- Speed up ATPG
- Shorten test set (6 to 4 sa faults for 2 i/p gates)

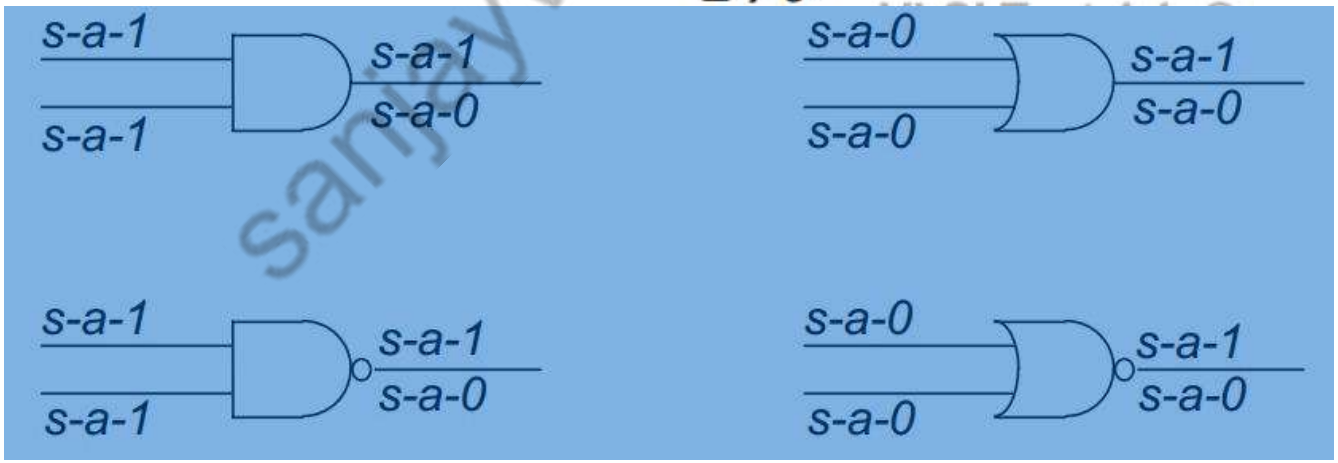
Equivalence Fault Collapsing

EFC on Fanout-free Circuits

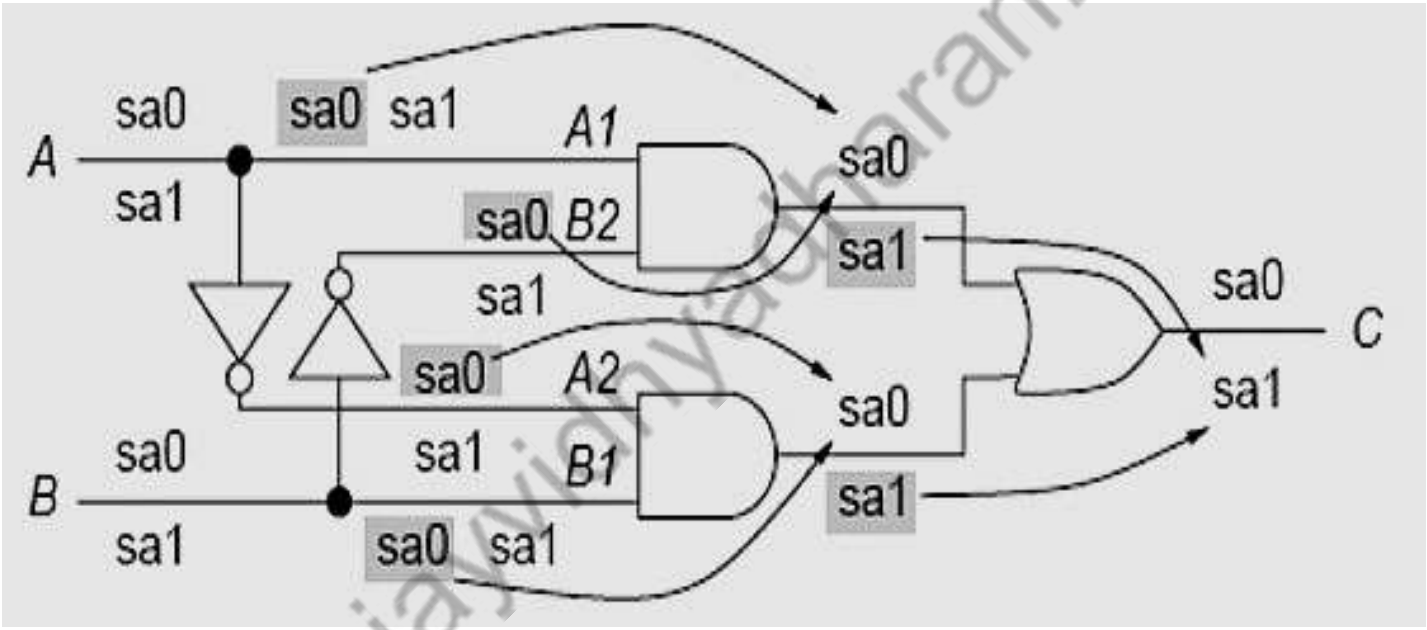
EFC Rules

- (1) both stuck-at one and zero faults for every primary output
- (2) one collapsed fault for each gate input





Equivalence Fault Collapsing



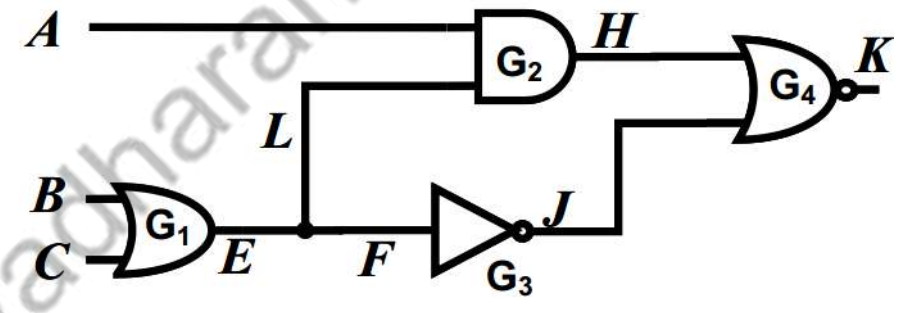
Fault collapsing reduces 18 s-a faults to 12

Equivalence Fault Collapsing

Fanout stem faults are NOT always equivalent to fanout branch faults

Example:

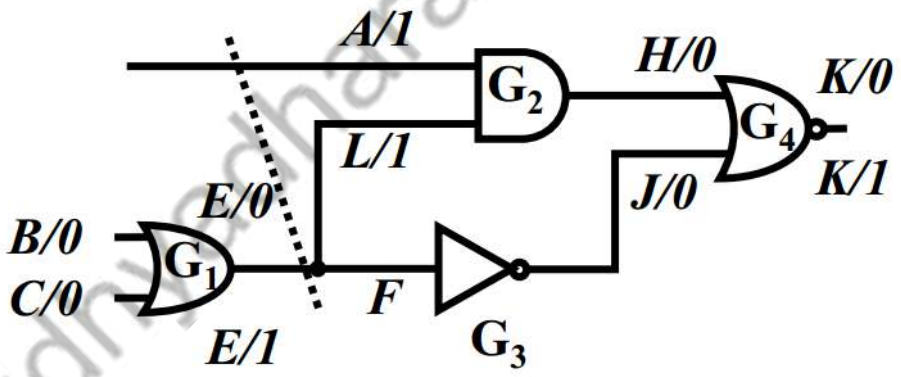
- **E/0 is equivalent to F/0**
- **but not equivalent to L/0**
- **The other faults are NOT equivalent**



Input			Output						
A	*B*	*C*	good	*E/0*	*F/0*	*L/0*	*E/1*	*F/1*	*L/1*
0	0	0	0	0	0	0	1	1	0
0	0	1	1	0	0	1	1	1	1
0	1	0	1	0	0	1	1	1	1
0	1	1	1	0	0	1	1	1	1
1	0	0	0	0	0	0	0	1	0
1	0	1	0	0	0	1	0	0	0
1	1	0	0	0	0	1	0	0	0
1	1	1	0	0	0	1	0	0	0

Equivalence Fault Collapsing

- 2 partitions
- Originally 18 faults, → after EFC 10 faults

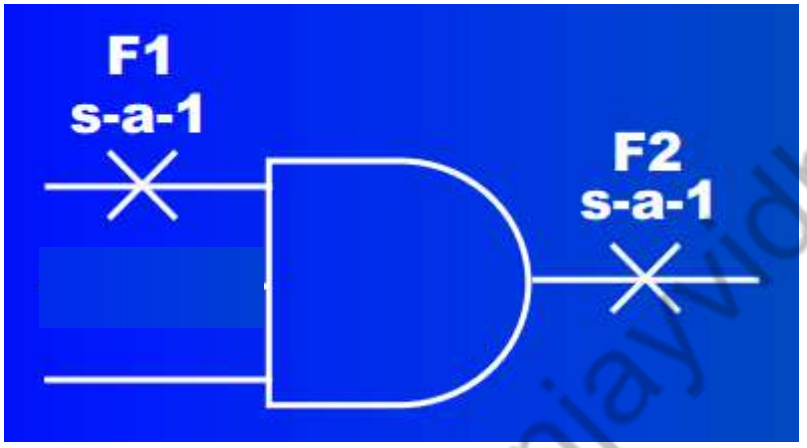


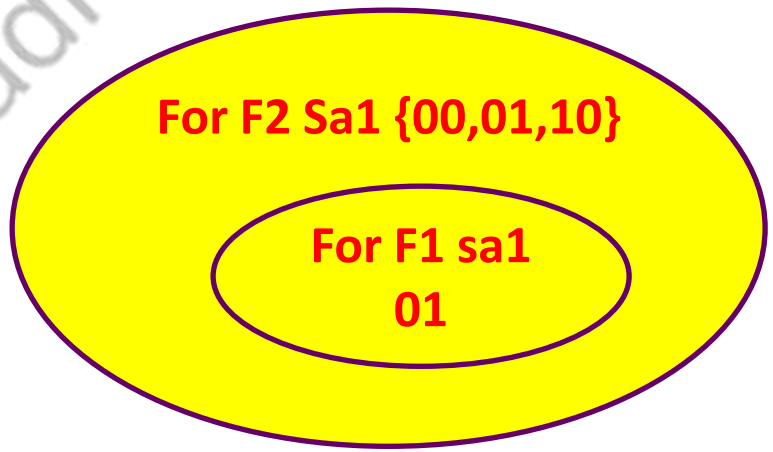
- NOTE:
  - Inverter G_3 ignored
  - because its input fault s@0 is always equivalent to its output s@1 fault

Dominance Fault Collapsing

Detecting set* of fault *f* (T_f) = set of all possible test patterns that detect fault *f

Fault *f* *dominates* fault *g* if the detecting set of *f* contains that of *g*



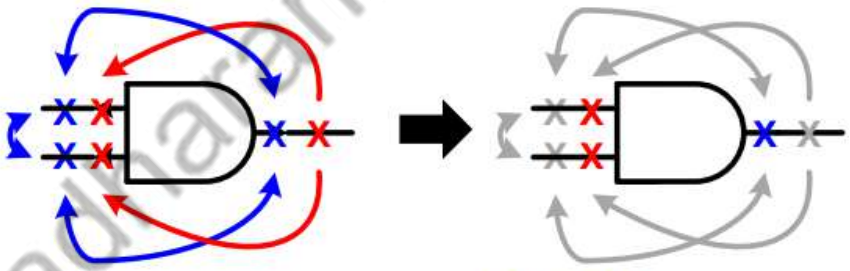


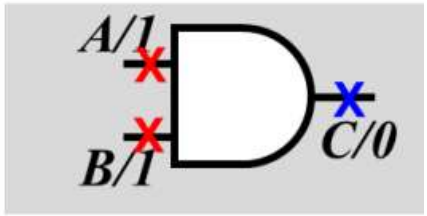
Fault *F2* *dominates* fault *F1*

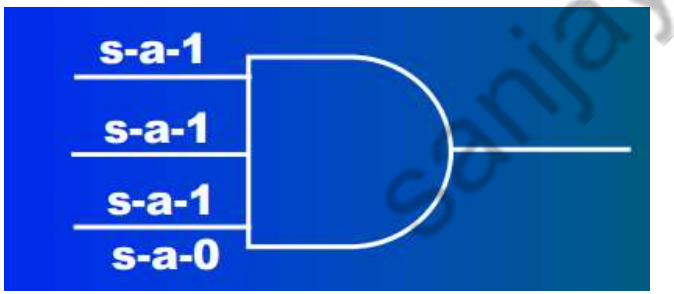
If fault F2 dominates F1, then F2 is removed from the fault list

Dominance Fault Collapsing

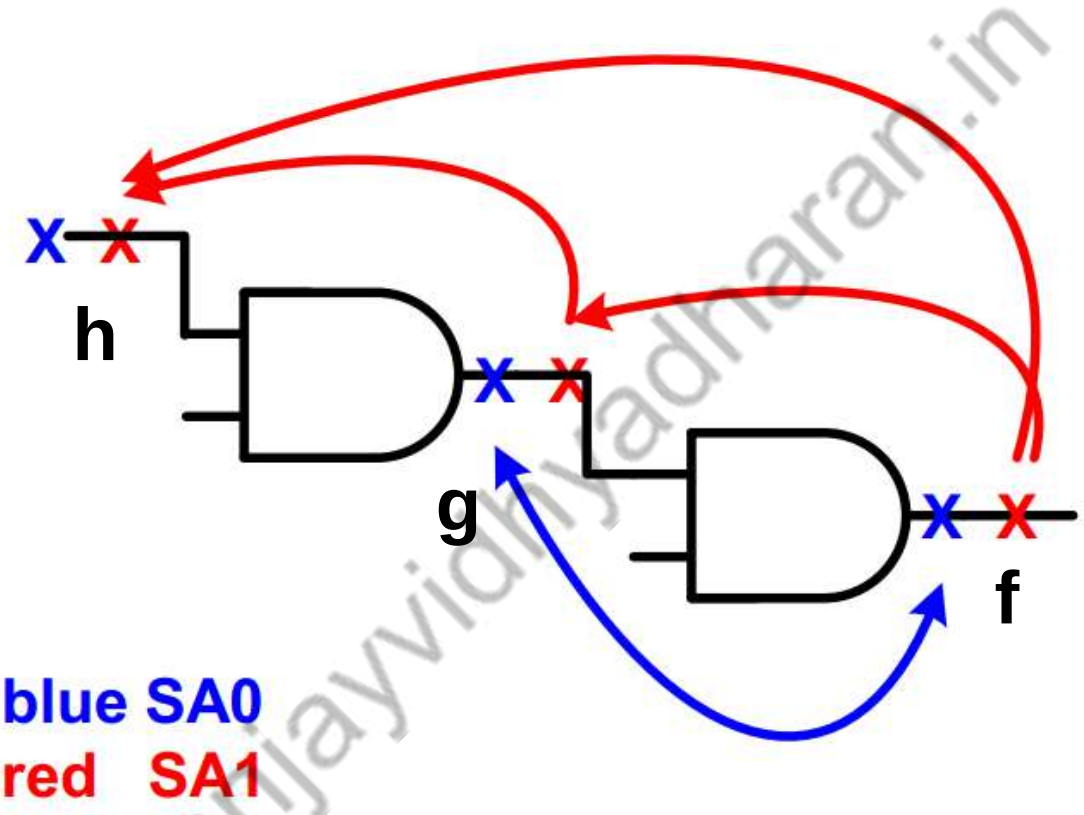
Input		Output						
A	*B*	Good	*A/0*	*C/0*	*B/0*	*A/1*	*C/1*	*B/1*
0	0	0	0	0	0	0	1	0
0	1	0	0	0	0	1	1	0
1	0	0	0	0	0	0	1	1
1	1	1	0	0	0	1	1	1







Dominance Fault Collapsing

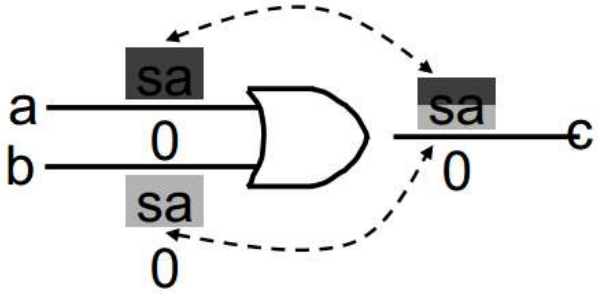


$f \leftrightarrow g \leftrightarrow h$

$f \rightarrow g \rightarrow h$

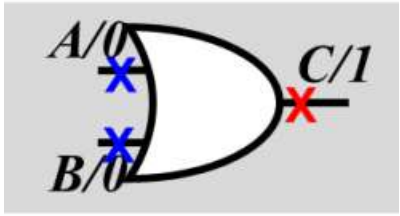
[2] Video lectures by Professor James Chien-Mo Li

Dominance Fault Collapsing

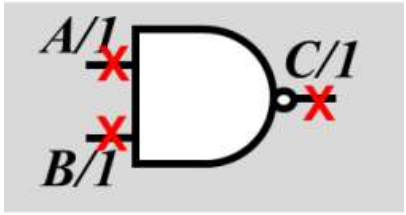


Fault *c:s-a-0* dominates faults: *a:s-a-0* and *b:s-a-0*

a	b	c	a: s-a-0	a: s-a-1	b: s-a-0	b: s-a-1	c: s-a-0	c: s-a-1
0	0	0	0	1	0	1	0	1
0	1	1	1	1	0	1	0	1
1	0	1	0	1	1	1	0	1
1	1	1	1	1	1	1	0	1



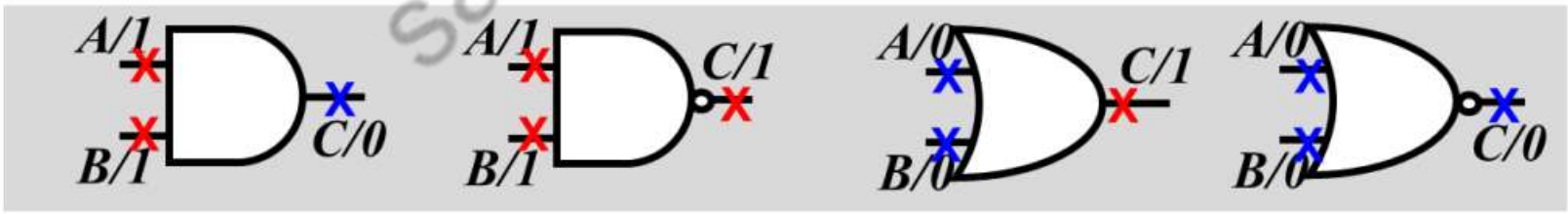
Dominance Fault Collapsing



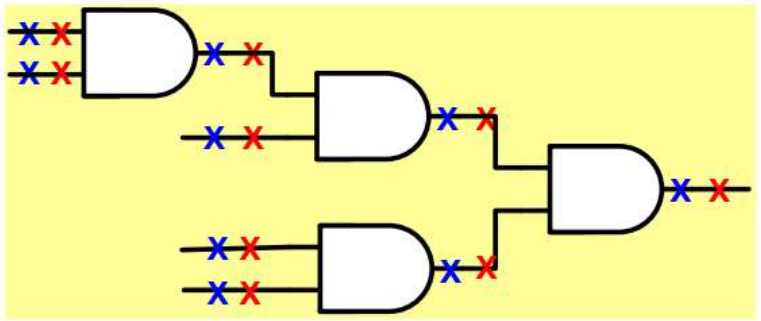
If this is passed obviously C is not Sa0

Input		Output						
A	*B*	Good	*A/0*	*B/0*	*C/0*	*A/1*	*B/1*	*C/1*
0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	1	1
1	0	1	1	1	0	1	0	1
1	1	0	1	1	0	0	0	1

If this is passed obviously A and B are is not Sa0

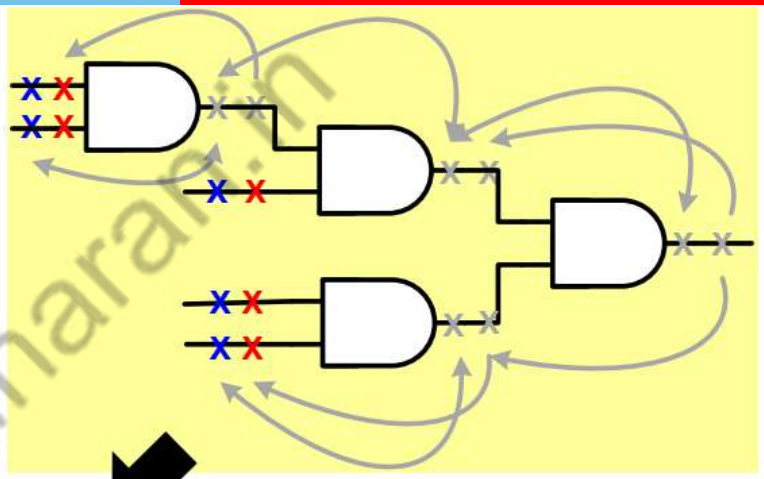


Dominance Fault Collapsing

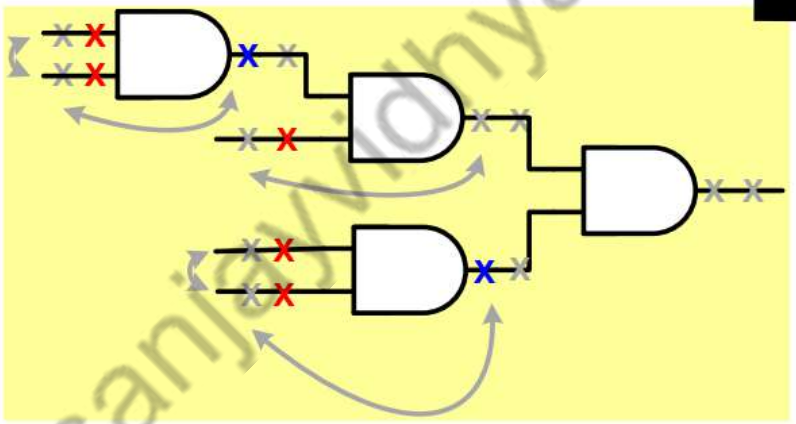


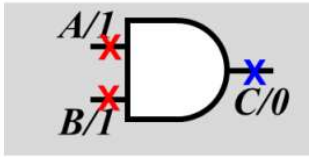
blue SA0
red SA1

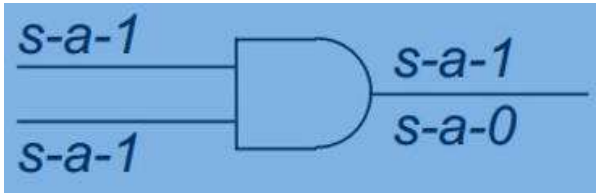
push to PI



all-PI gates reduce further





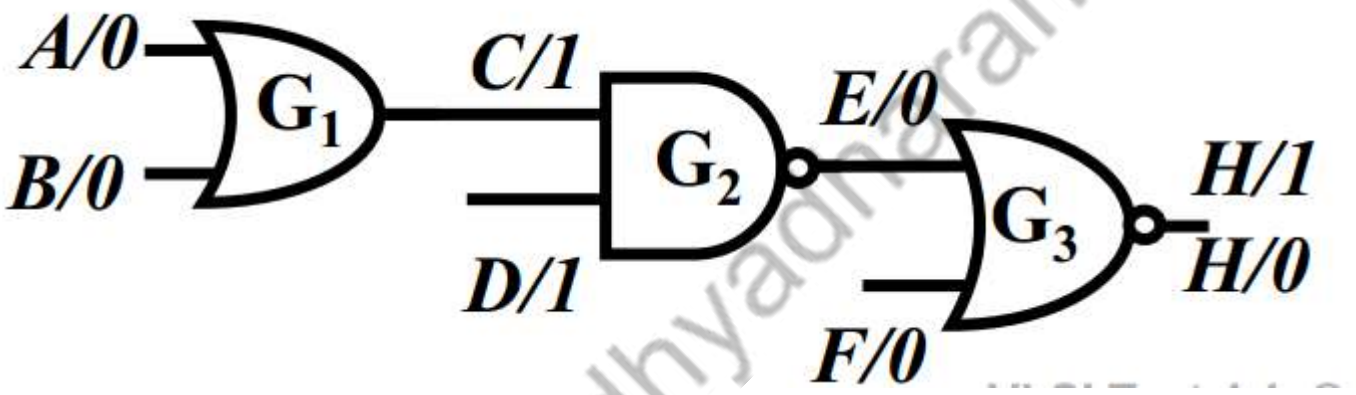


- Originally 18 faults, → after EFC 10 faults → DFC 7 faults

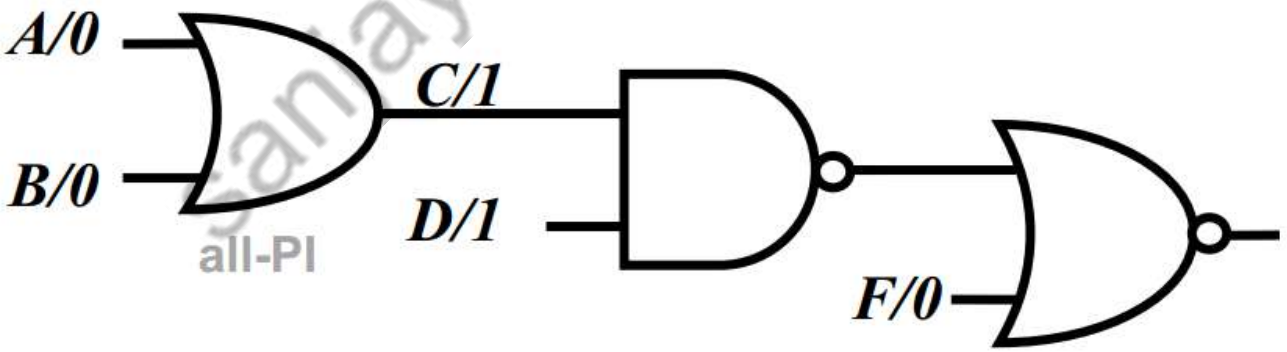
[2] Video lectures by Professor James Chien-Mo Li

Dominance Fault Collapsing

14 faults → **8** faults after EFC



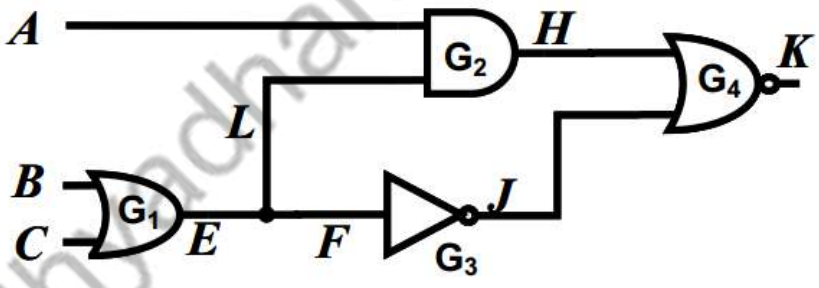
→ **5** faults after DFC

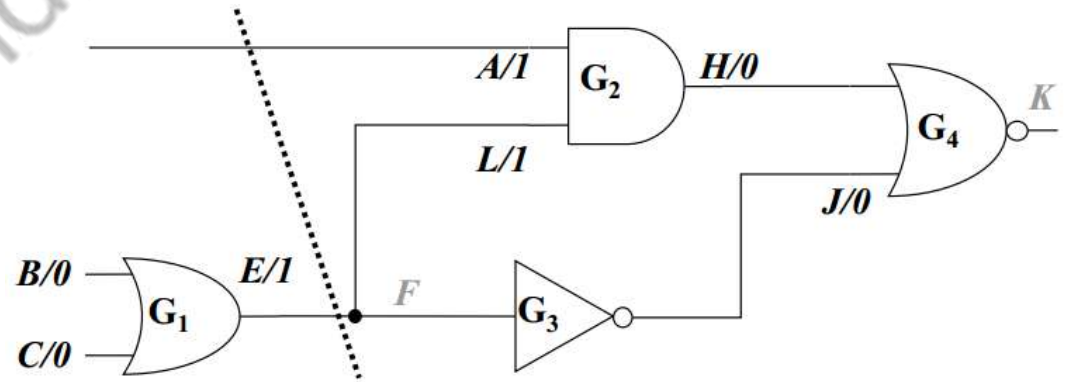


Dominance Fault Collapsing

Fanout Stem and Branches

Input			Output						
A	*B*	*C*	good	*E/0*	*F/0*	*L/0*	*E/1*	*F/1*	*L/1*
0	0	0	0	0	0	0	1	1	0
0	0	1	1	0	0	1	1	1	1
0	1	0	1	0	0	1	1	1	1
0	1	1	1	0	0	1	1	1	1
1	0	0	0	0	0	0	0	1	0
1	0	1	0	0	0	1	0	0	0
1	1	0	0	0	0	1	0	0	0
1	1	1	0	0	0	1	0	0	0



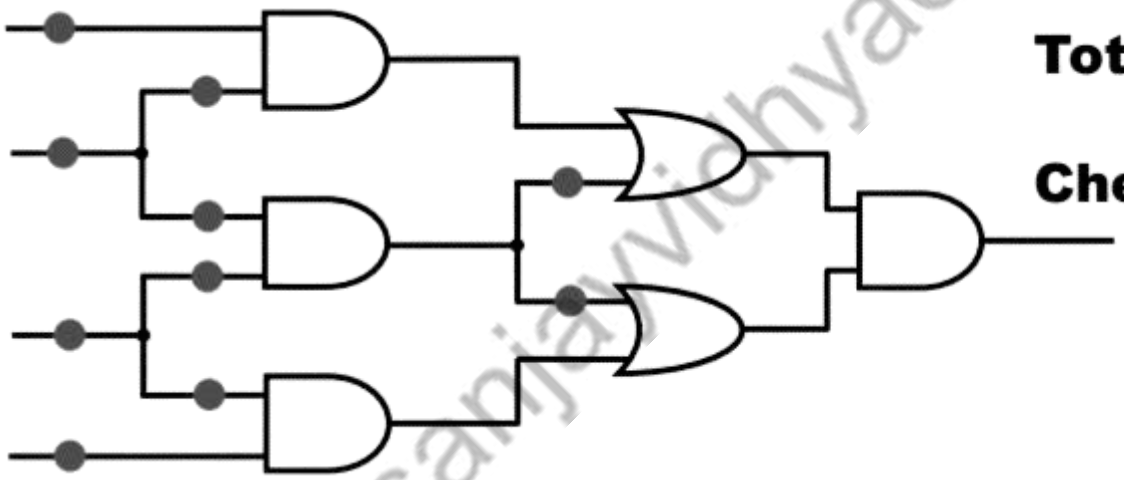


- Originally 18 faults, → after EFC 10 faults → DFC 7 faults

Checkpoint Theorem

Primary inputs and fanout branches of a combinational circuit are called *checkpoints*

Checkpoint theorem: *"A test set that detects all single (multiple) stuck-at faults on all checkpoints of a combinational circuit, also detects all single (multiple) stuck-at faults in that circuit."*

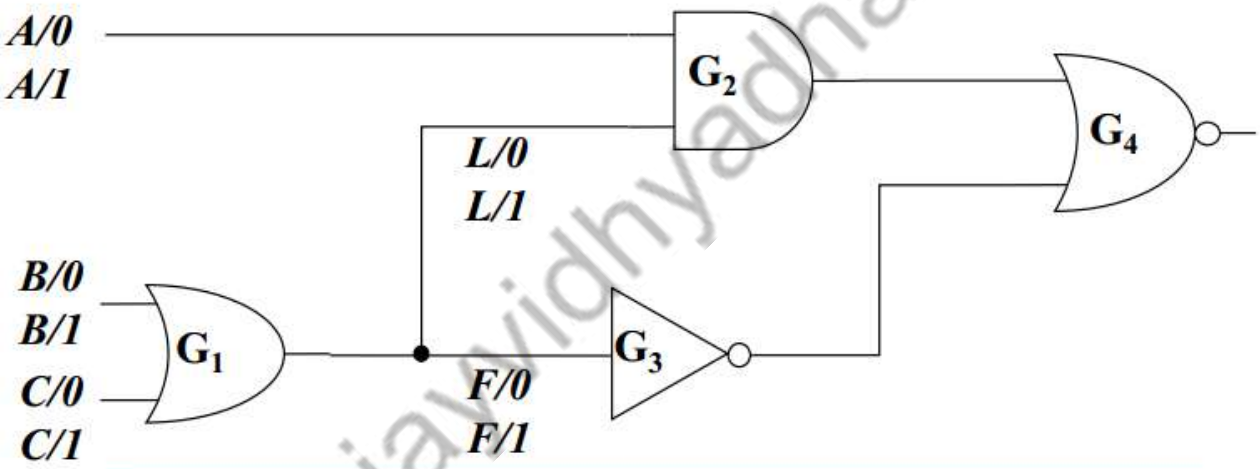


Total fault sites = 16

Checkpoints (●) = 10

Checkpoint Theorem

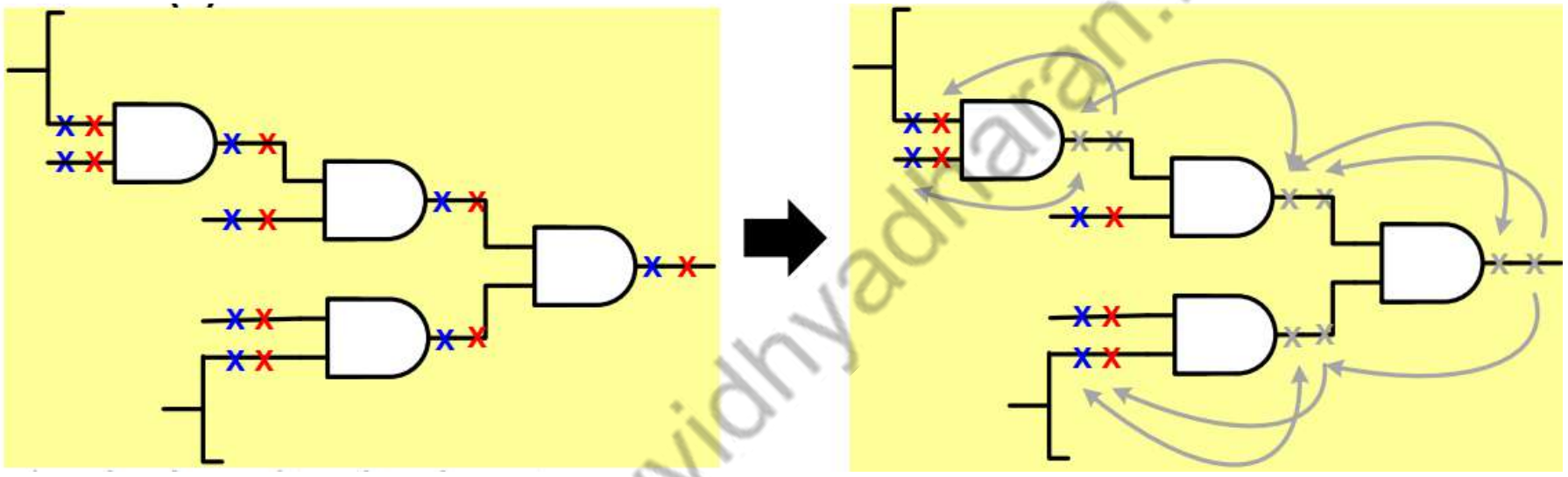
- 10 faults on checkpoints
- Originally 18 faults, after EFC 10 faults, after DFC 7 faults



$$|CHKPT| \geq |EFC| \geq |DFC|$$

[2] Video lectures by Professor James Chien-Mo Li

Checkpoint Theorem



Chkpt is a Simpler Alternative to EFC/DFC

DFC has issues in sequential circuits and EFC is most preferred technique for ATPG

[2] Video lectures by Professor James Chien-Mo Li

Collapse Ratio

	uncollapsed	Collapsed (typically EFC)
Total Faults	1,234	800
Detected faults	1,000	700
Untestable faults	230	98
Aborted faults	4	2
Fault Coverage	1,000/1,234	700/800

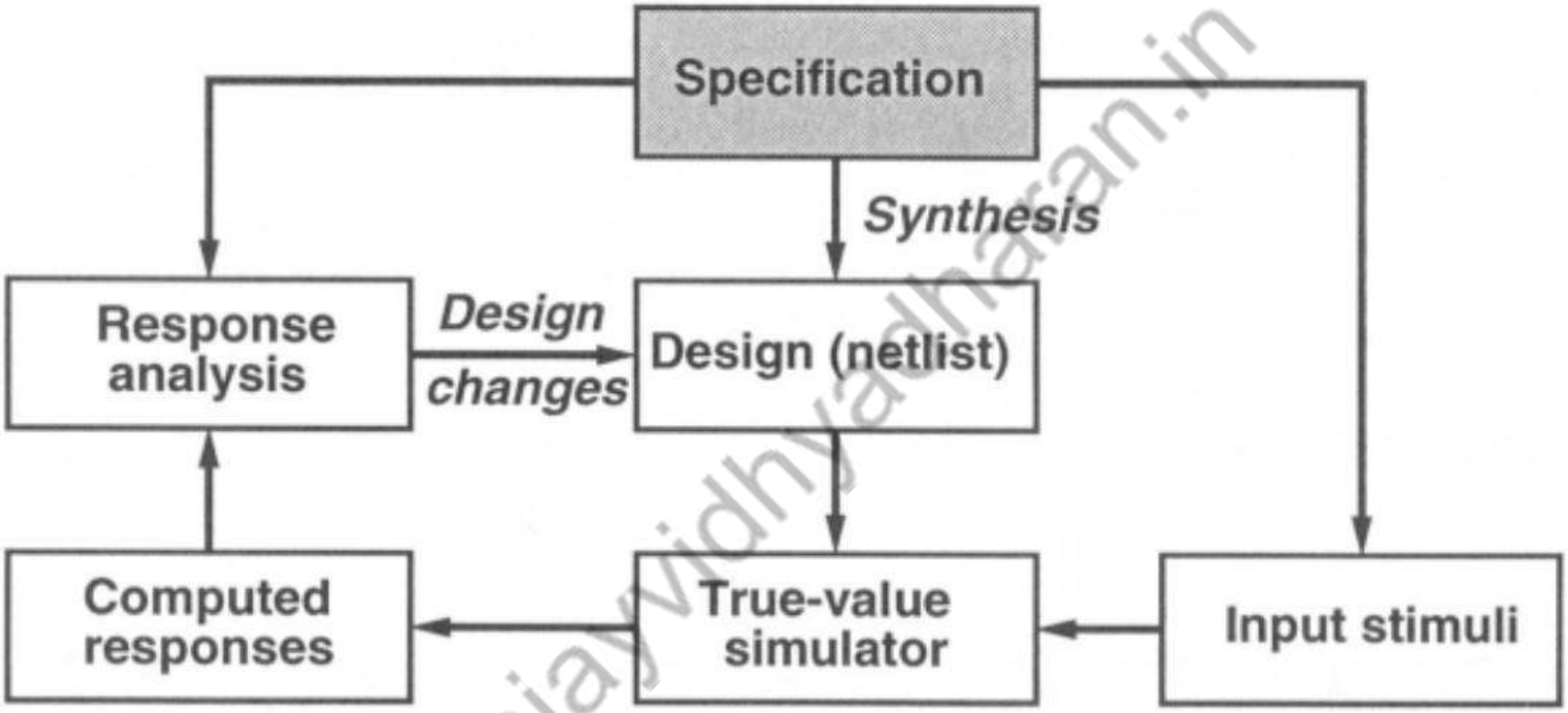
$$CollapseRatio = \frac{\text{number of } collapsed \text{ faults}}{\text{number of } uncollapsed \text{ faults}} \times 100\% = \frac{800}{1234} \approx 64.8\%$$

$$uncollapsed\ F.C. = \frac{\text{detected } uncollapsed \text{ faults}}{\text{total } uncollapsed \text{ faults}} \times 100\%$$

$$Collapsed\ F.C. = \frac{\text{detected } collapsed \text{ faults}}{\text{total } collapsed \text{ faults}} \times 100\%$$

[2] Video lectures by Professor James Chien-Mo Li

Simulation for Design Verification



True-value means that the simulator will compute the response for given input stimuli without injecting any faults in the design. The input stimuli are also based on the specification.

A frequently used strategy is to exercise all functions with only *critical* data patterns. This is because the simulation of the exhaustive set of data patterns can be too expensive

True Value Simulation

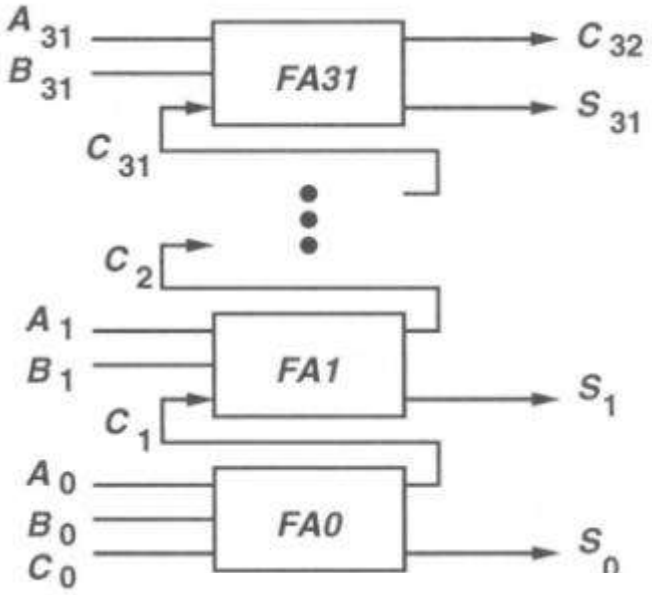
1. A design can be first simulated at a higher behavior level (such as C).
   Netlist not required
   Does not contain the detailed timing information.
   No electrical behavior
2, Once this design is verified, higher-level blocks are replaced by logic-level netlists.
   At this point, a **logic simulator** is used for verification.
3. The process may be repeated by replacing some or all portions by transistor-level or circuit-level implementations.

Simulation is used in this way for verifying very large electronic systems.

The weakness of this method is its dependence on the designer's heuristics used in generating the input stimuli.

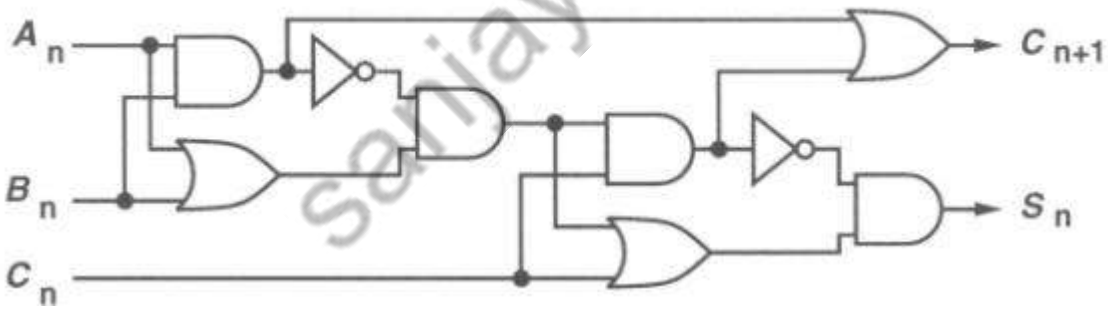
Simulation for Design Verification

Logic design of a 32-bit ripple-carry adder.



How Many Test vectors Required?

Vector no.	Bits: $C_0A_0B_0A_1B_1A_2B_2A_3B_3$	Input $C_nA_nB_n$ to FAn
1	000000000	000 applied to all FAs
2	001010101	001 applied to all FAs
3	010101010	010 applied to all FAs
4	011001100	011 applied to FA0, FA2 & 100 applied to FA1, FA3
5	100110011	100 applied to FA0, FA2 & 011 applied to FA1, FA3
6	101010101	101 applied to all FAs
7	110101010	110 applied to all FAs
8	111111111	111 applied to all FAs

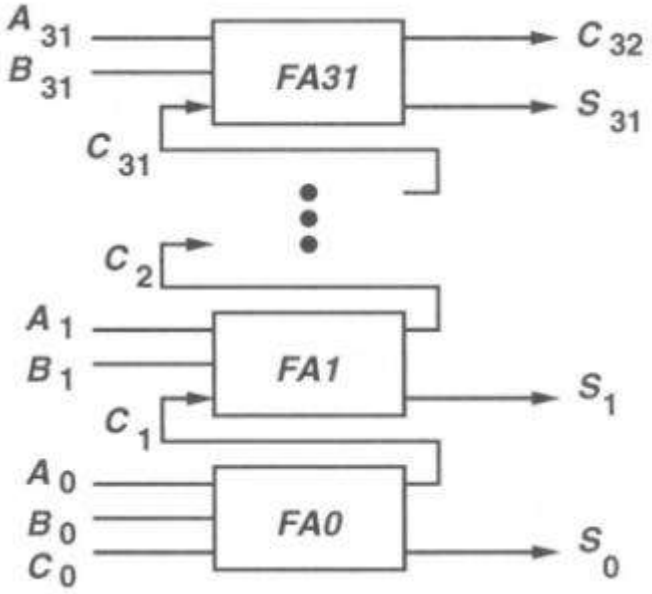


The first seven vectors cover all stuck-at faults. One may, therefore, use only the first seven vectors in the manufacturing test.

Note: This optimization is possible because of same blocks (FA) being used and each test vector verifying similar faults in all blocks

Simulation for Design Verification

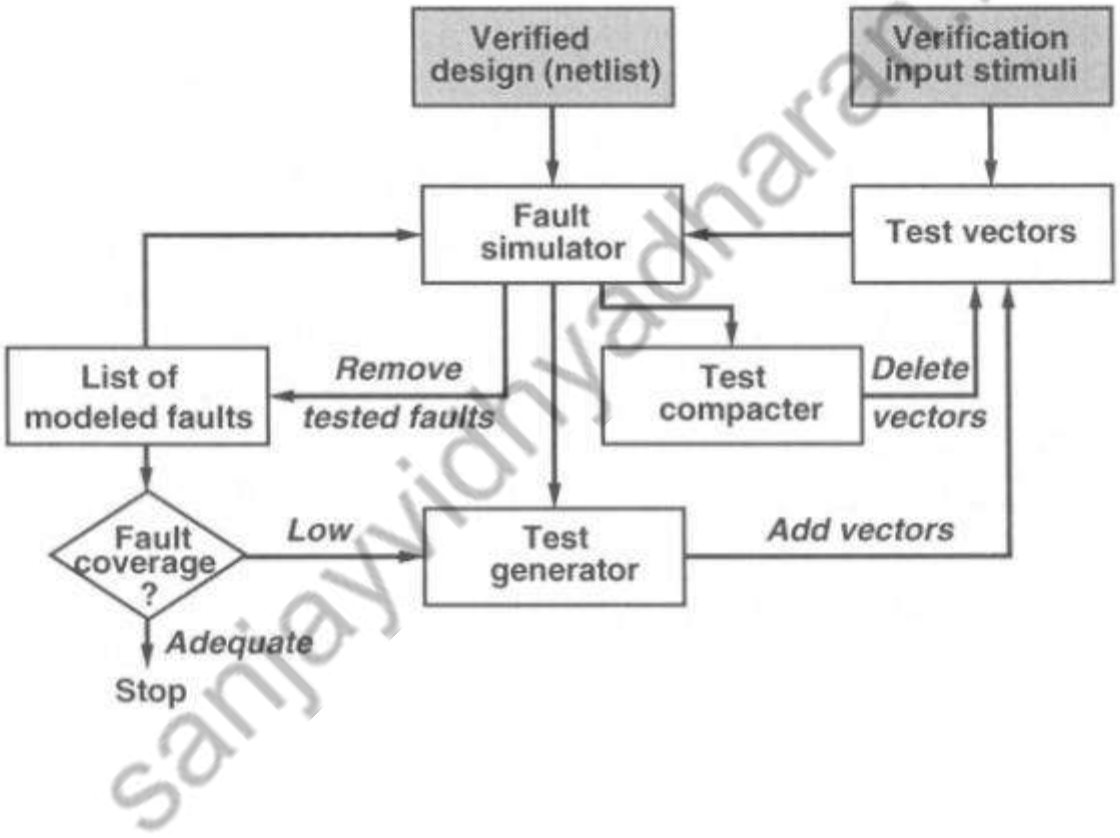
Logic design of a 32-bit ripple-carry adder.



Vector no.	Bits: $C_0A_0B_0A_1B_1A_2B_2A_3B_3$	Input $C_nA_nB_n$ to FAn
1	000000000	000 applied to all FAs
2	001010101	001 applied to all FAs
3	010101010	010 applied to all FAs
4	011001100	011 applied to FA0, FA2 & 100 applied to FA1, FA3
5	100110011	100 applied to FA0, FA2 & 011 applied to FA1, FA3
6	101010101	101 applied to all FAs
7	110101010	110 applied to all FAs
8	111111111	111 applied to all FAs

Timing analysis of 2 followed by 6 or 3 followed by 7 where carry propagates through the chain

Fault simulation for test generation



References

1. “Essentials of Electronic Testing, for Digital, Memory and Mixed-Signal VLSI Circuits”, Michael L. Bushnell and Vishwani D. Agrawal, – Kluwer Academic Publishers (2000).

2. Video lectures by Professor James Chien-Mo Li
Lab. of Dependable Systems Graduate Institute of Electronics Engineering
National Taiwan University
https://www.youtube.com/watch?v=yfcoKOUV5DM&list=PLvd8d-SyI7hjk_Ci0zpTqImAtpEjdK5JF&index=1

Thankyou