國立清華大學電機系

EE-6250
超大型積體電路測試
VLSI Testing

Chapter 2
Fault Modeling

Functional v.s. Structural Testing

- I/O functional tests inadequate for manufacturing
- Exhaustive testing is prohibitively expensive

Question: How to Generate Compact yet High-Quality Test Vectors?

Why Fault Model ?

- **Fault model identifies target faults**
  - Model faults most likely to occur
- **Fault model limits the scope of test generation**
  - Create tests only for the modeled faults
- **Fault model makes effectiveness measurable by experiments**
  - **Fault coverage can be computed** for specific test patterns to reflect its effectiveness
- **Fault model makes analysis possible**
  - Associate specific defects with specific test patterns

Scientific Study: Hypothesis (Assumption) → Evaluation → Refinement

Fault Modeling

- **Fault Modeling**
  - Model the effects of physical defects on the logic function and timing
- **Physical Defects**
  - Silicon Defects
  - Photolithographic Defects
  - Mask Contamination
  - Process Variation
  - Defective Oxides

Common Fault Types Used To Guide Test Generation

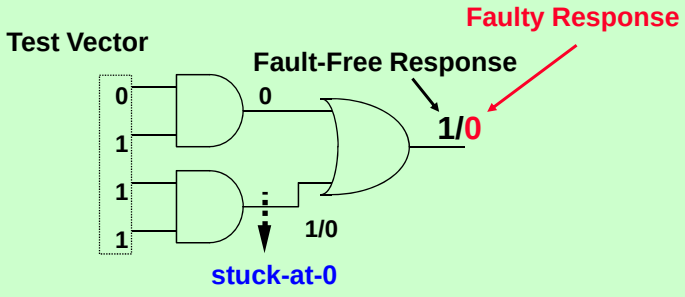
- **Stuck-at Faults**
- **Bridging Faults**
- **Open Faults**
- **Transistor Stuck-On Faults**
- **Delay Faults**
- **IDDQ Faults (Quiescent current at VDD pin)**
- **Memory Faults**

IDDQ Testing: canary in the coalmine, alarming of un-modeled defects

金絲雀

Ch2-5

Single Stuck-At Fault



Assumptions:

- **Only One line is faulty**
- **Faulty line permanently set to 0 or 1**
- **Fault can be at an input or output of a gate**

Ch2-6

Multiple Stuck-At Faults

- **Several stuck-at faults occur at the same time**
  - Mostly used in logic diagnosis
- **For a circuit with k lines**
  - there are 2k single stuck-at faults
  - there are 3^k-1 multiple stuck-at faults
    - A line could be stuck-at-0, stuck-at-1, or fault-free
    - One out of 3^k resulting circuits is fault-free

Why Single Stuck-At Fault Model?

- **Complexity is greatly reduced**
  - Many different physical defects may be modeled by the same logical single stuck-at fault
- **Stuck-at fault is technology independent**
  - Can be applied to TTL, ECL, CMOS, BiCMOS etc.
- **Design style independent**
  - Gate array, standard cell, custom VLSI
- **Detection capability of un-modeled defects**
  - Empirically, many defects accidentally detected by test derived based on single stuck-at fault
- **Cover a large percentage of multiple stuck-at faults**

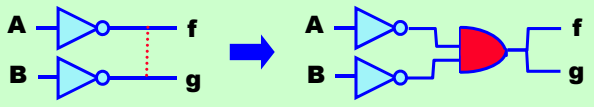
Single SA model survives well (due to its simplicity and effectiveness)

Multiple Faults

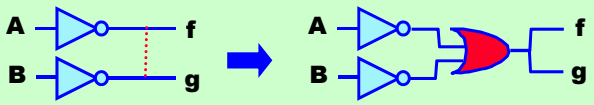
- **Multiple stuck-fault coverage by single-fault tests of combinational circuit:**
  - **4-bit ALU (Hughes & McCluskey, ITC-84)**
    All double and most triple-faults covered.
  - **Large circuits (Jacob & Biswas, ITC-87)**
    Almost 100% multiple faults covered for circuits with 3 or more outputs.

Bridging Faults

- **Two or more normally distinct points (lines) are shorted together erroneously**
  - **Logic effect depends on technology**
  - **Wired-AND for TTL**



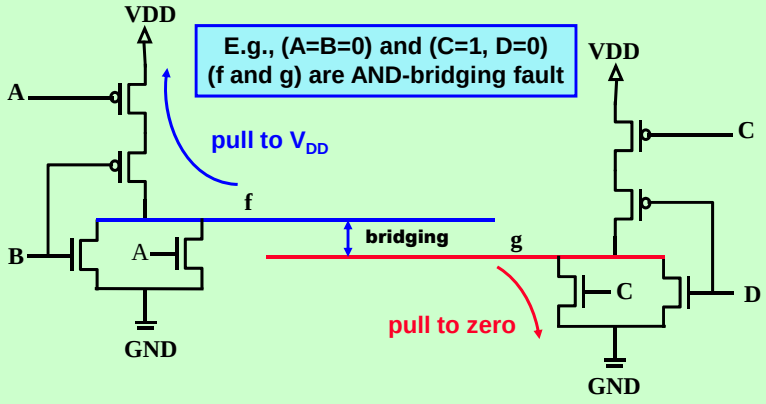
  - **Wired-OR for ECL**



  - **CMOS ?**

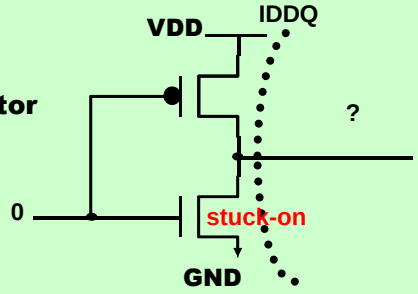
Bridging Faults For CMOS Logic

- **The result**
  - could be AND-bridging or OR-bridging
  - depends on the inputs



CMOS Transistor Stuck-On

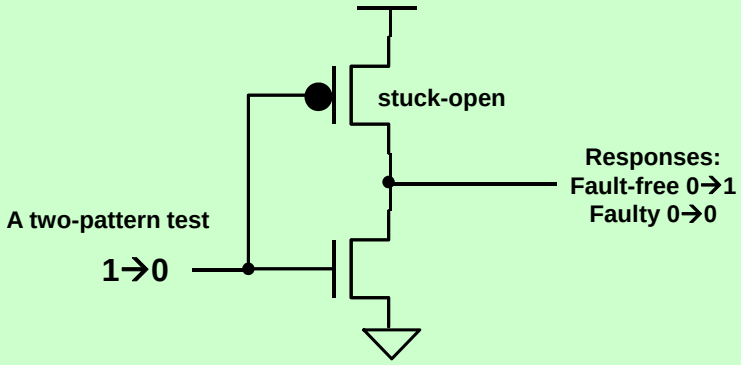
Example:
N-type transistor is always ON



- **Transistor Stuck-On**
  - May cause ambiguous logic level
  - Depends on the relative impedances of the pull-up and pull-down networks
- **When Input Is Low**
  - Both P and N transistors are conducting, causing increased quiescent current, could be detected by IDDQ test

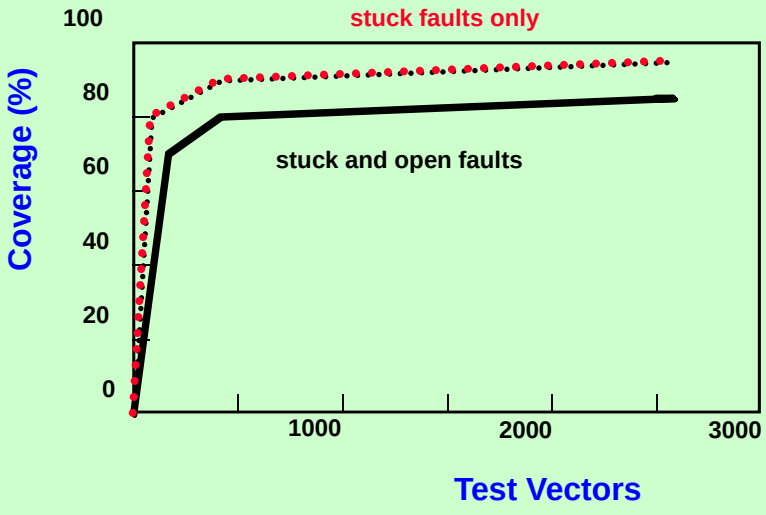
CMOS Transistor Stuck-Open (I)

- **Transistor stuck-open**
  - **May cause the output to be floating**
  - **The fault exhibits sequential behavior**
  - **Need two-pattern test (to set it to a known value first)**



Ch2-13

Fault Coverage in a CMOS Chip



Ch2-14

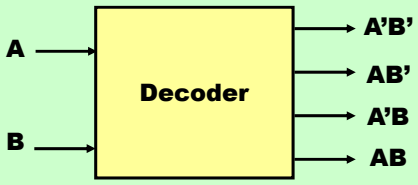
Summary of Stuck-Open Faults

- **First Report:**
  - Wadsack, Bell System Technology, J., 1978
- **Recent Results**
  - Woodhall et. al, ITC-87 (1-micron CMOS chips)
  - 4552 chips passed the test
  - 1255 chips (27.57%) failed tests for stuck-at faults
  - 44 chips (0.97%) failed tests for stuck-open faults
  - 4 chips with stuck-open faults passed tests for stuck-at faults
- **Conclusion**
  - Stuck-at faults are about 20 times more frequent than stuck-open faults
  - About 91% of chips with stuck-open faults may also have stuck-at faults
  - Faulty chips escaping tests for stuck-at faults = 0.121%

Functional Faults

- **Fault effects modeled at a higher level than logic for functional modules, such as**
  - Decoder
  - Multiplexers
  - Adders
  - Counters
  - ROMs

Functional Faults of Decoders



- **$f(L_i/L_k)$:** One active output, but wrong one
  - Instead of input line L_i, L_k is selected
- **$f(L_i/L_{i+k})$:** More than one active outputs
  - In addition to line L_i, L_k is also selected
- **$f(L_i/0)$:** No active output
  - None of the lines is selected

Memory Faults

- **Parametric Faults**
  - Any fault that causes the response to deviate from its fault-free nominal value by some amount
  - Ex. A cell with parametric delay fault (with for example 93% more than normal)
  - Due to all kinds of factors like PVT variation
- **Functional Faults**
  - Stuck Faults in Address Register, Data Register, and Address Decoder
  - Cell Stuck Faults
  - Adjacent Cell Coupling Faults
  - Pattern-Sensitive Faults

Memory Faults

- **Pattern-sensitive faults: the presence of a faulty signal depends on the signal values of the neighboring cells**
  - **Mostly in DRAMs**

0	0	0
0	d	b
0	a	0

a=b=0 ➔ d=0
a=b=1 ➔ d=1

- **Adjacent cell coupling faults**
  - **Pattern sensitivity between a pair of cells**

Memory Testing

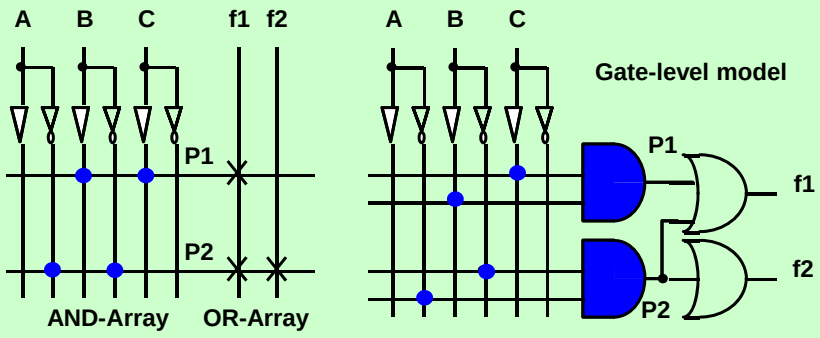
- **Test could be time-consuming**
  - **The length of the test sequence for memory testing could be prohibitively long**
- **Example:**
  - **A pattern sensitive test is $5n^2$ long for an n-bit RAM**
  - **Testing a 1-M bit chip at 10ns pattern would take 14 hours**
  - **For a 64-M bit chip, it would take 6 years**

PLA Faults

- **Stuck-at Faults**
- **Cross-point Faults**
  - **Extra/Missing Transistors**
- **Bridging Faults**
- **Break Faults**

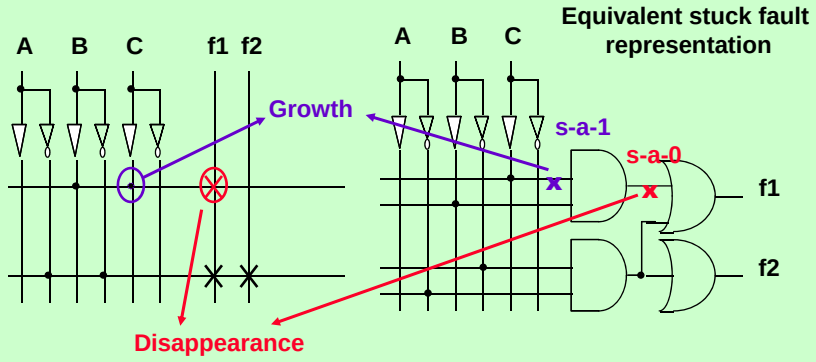
Stuck-at Faults in PLA

- **s-a-0 & s-a-1 faults**
  - **on inputs, input inverters, product lines, and outputs are easy to simulate in its gate-level model**



Missing Cross-Point Faults in PLA

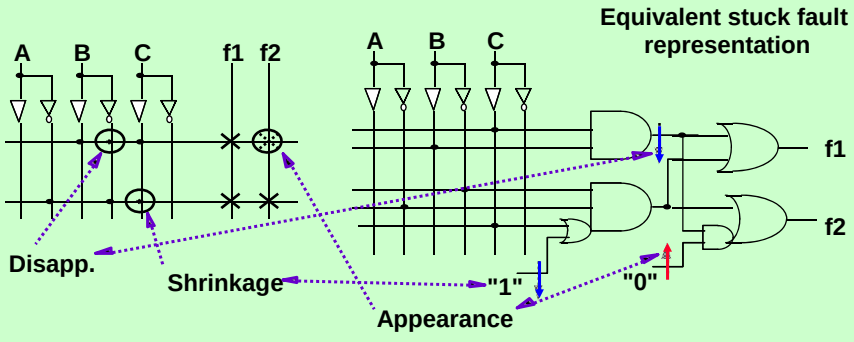
- **Missing Crosspoint in AND-array**
  - **Growth Fault**
- **Missing Crosspoint in OR-array**
  - **Disappearance fault**



Ch2-23

Extra Cross-Point Faults in PLA

- **Extra cross-point in AND-array**
  - **Shrinkage or disappearance fault**
- **Extra cross-point in OR-array**
  - **Appearance fault**



Ch2-24

Summary of PLA Faults

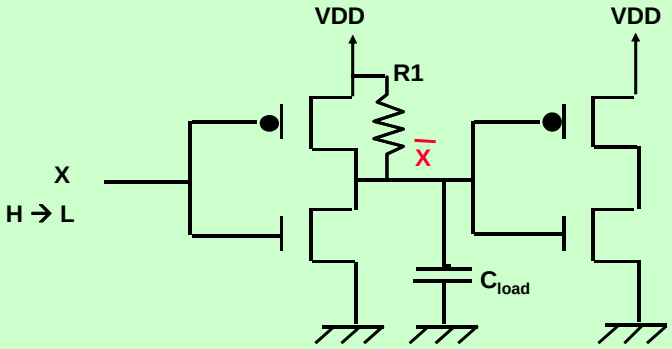
- **Cross-Point Faults**
  - 80 ~ 85% covered by stuck-fault tests
  - Layout-dependence in folded PLA
- **Bridging Faults**
  - 99% covered by stuck-fault tests
  - Layout-dependence in all PLAs
  - (Ref: Agrawal & Johnson, ICCD-86)

Delay Testing

- **Chip with Timing Defects**
  - may pass the DC stuck-fault testing, but fail when operated at the system speed
  - For example, a chip may pass the test under 10 MHz operation, but fail under 100 MHz
- **Delay Fault Models**
  - Gate-Delay Fault
  - Path-Delay Fault

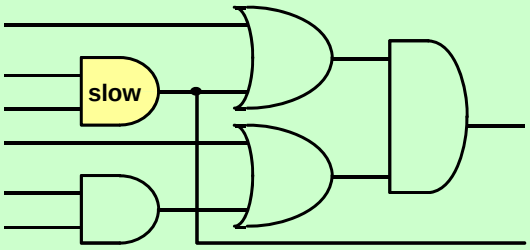
Gate-Delay Fault (I)

- **Slow to Rise**
  - $\overline{x}$ is slow to rise when channel resistance R1 is abnormally high



Ch2-27

Gate-Delay Fault (II)

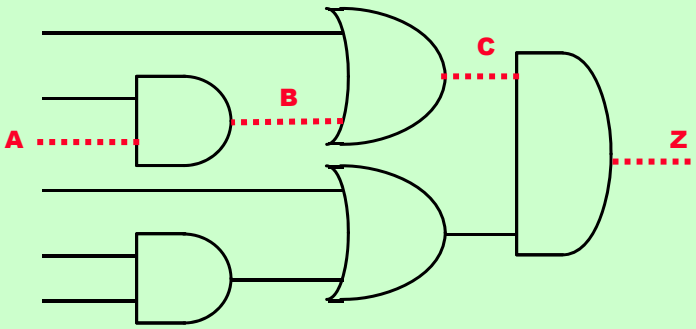


- **Test Based on Gate-Delay Fault**
  - May not detect those delay faults that result from the accumulation of a number of small incremental delay defects along a path !! (Disadvantage)

Ch2-28

Path-Delay Fault

- **Associated with a Path (e.g., A-B-C-Z)**
  - Whose delay exceeds the clock interval
- **More complicated than gate-delay fault**
  - Because the number of paths grows exponentially



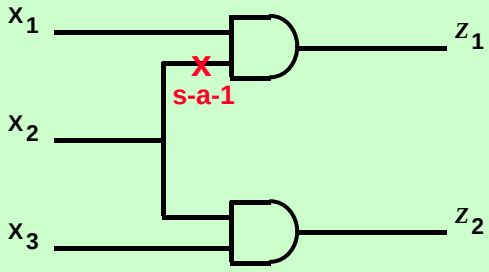
Ch2-29

Fault Detection

- **Fault Activation**
- **Fault Propagation**

Definition Of Fault Detection

- **A test (vector) *t* detects a fault *f* iff**
  - ***t*** **detects** ***f*** **⇔** $z(t) \neq z_f(t)$
- **Example**



$z_1=x_1x_2$ $z_2=x_2x_3$

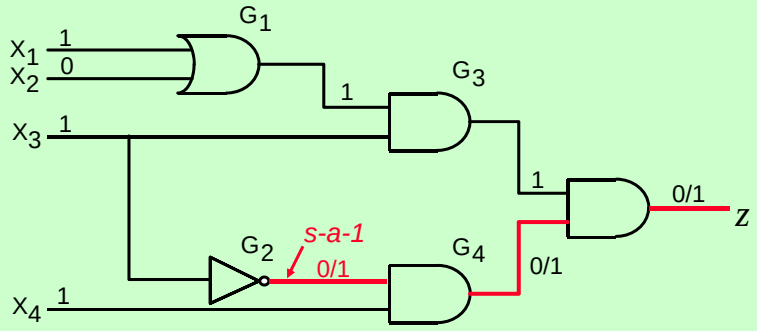
$z_{1f}=x_1$ $z_{2f}=x_2x_3$

The test (x1,x2,x3) = (100) detects *f* because $z_1(100)=0$ while $z_{1f}(100)=1$

Fault Detection Requirement

- **A test *t* that detects a fault *f***
  - **(1) Activate *f* (or generate a fault effect at the site of the fault)**
  - **(2) Propagate the fault effect to a primary output *w***
- **Sensitized Line:**
  - **A line whose faulty value is different from its fault-free one is said to be sensitized by the test in the faulty circuit**
- **Sensitized Path:**
  - **A path composed of sensitized lines is called a sensitized path**

Fault Sensitization



z (1011)=0 z_f (1011)=1

1011 detects the fault f (G_2 stuck-at 1)

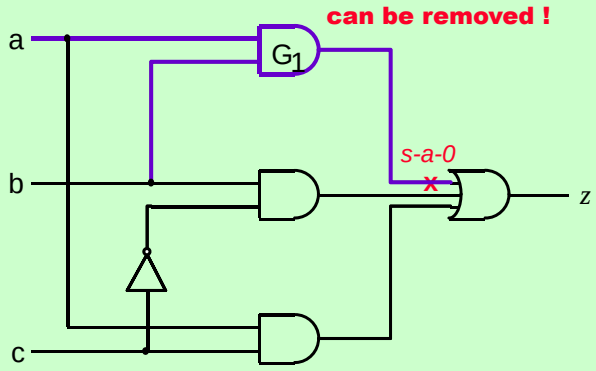
v/v_f : v = signal value in the fault free circuit

v_f = signal value in the faulty circuit

Detectability

- **A fault *f* is said to be detectable**
  - **if there exists a test *t* that detects *f*; otherwise, *f* is an undetectable fault**
- **For an undetectable fault *f***
  - **No test can simultaneously activate *f* and create a sensitized path to a primary output**

Undetectable Fault

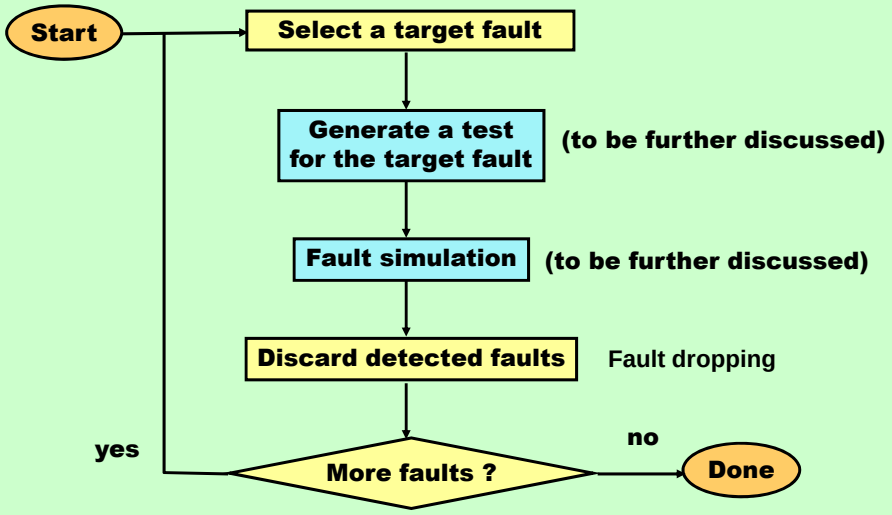


- G_1 output stuck-at-0 fault is undetectable
  - Undetectable faults do not change the function of the circuit
  - The related circuit can be deleted to simplify the circuit

Test Set

- 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
- The fault coverage
  - can be determined by fault simulation
  - >95% is typically required for single stuck-at fault model
  - >99.9% in IBM

Typical Test Generation Flow



Ch2-37

Fault Collapsing

- **Fault Equivalence**
- **Fault Dominance**
- **Checkpoint Theorem**

Fault Equivalence

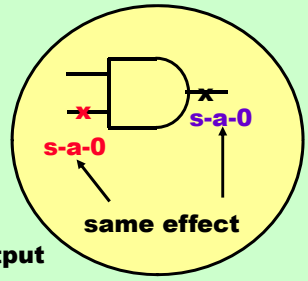
- **Distinguishing test**
  - **A test *t* distinguishes faults α and β if**

$$Z_{\alpha}(t) \oplus Z_{\beta}(t) = 1$$

- **Equivalent Faults**
  - **Two faults, α & β are said to be equivalent in a circuit , iff the function under α is equal to the function under β for any input combination (sequence) of the circuit.**
  - **No test can distinguish between α and β**

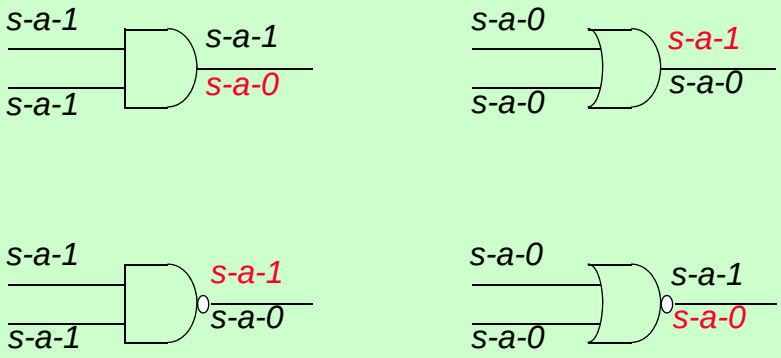
Fault Equivalence

- **AND gate:**
  - **all *s-a-0* faults are equivalent**
- **OR gate:**
  - **all *s-a-1* faults are equivalent**
- **NAND gate:**
  - **all the input *s-a-0* faults and the output *s-a-1* faults are equivalent**
- **NOR gate:**
  - **all input *s-a-1* faults and the output *s-a-0* faults are equivalent**
- **Inverter:**
  - **input *s-a-1* and output *s-a-0* are equivalent**
    input *s-a-0* and output *s-a-1* are equivalent



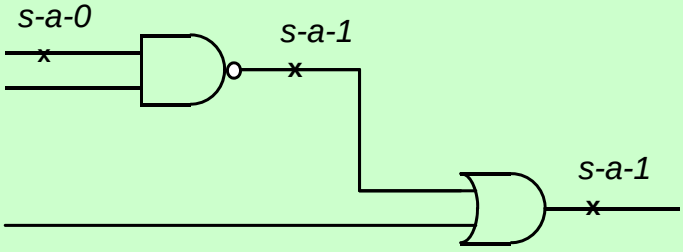
Equivalence Fault Collapsing

- ***n+2*** **instead of** ***2(n+1)*** **faults need to be considered for** ***n*****-input gates**



Equivalent Fault Group

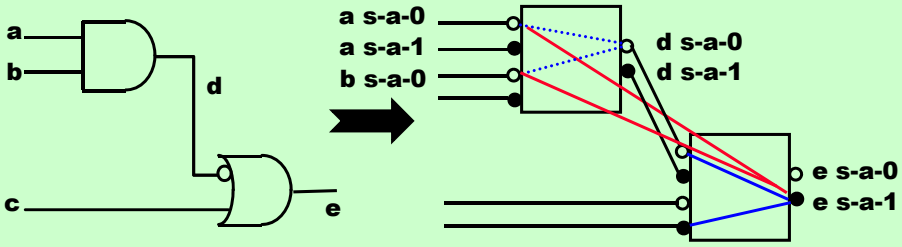
- **In a combinational circuit**
  - **Many faults may form an equivalent group**
  - **These equivalent faults can be found by sweeping the circuit from the primary outputs to the primary inputs**



Three faults shown are equivalent !

Finding Equivalent Group

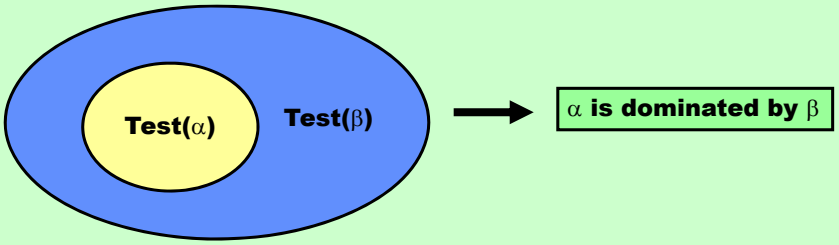
- **Construct a Graph**
  - Sweeping the netlist from PO's to PI's
  - When a fault α is equivalent to a fault β, then an edge is connected between then
  - Transitive Rule:
    - When α connects β and β connects γ, then α connects γ



Equivalent group = { a/0, b/0, d/0, c/1, e/1}

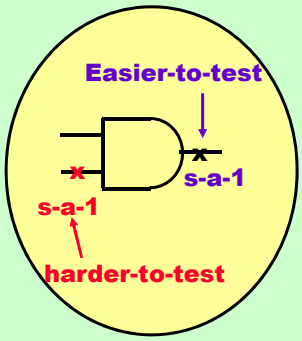
Fault Dominance

- **Dominance Relation**
  - A fault β is said to *dominate* another fault α in a circuit, iff every test (sequence) for α is also a test (sequence) for β.
  - I.e., test-set(β) > test-set(α)
  - No need to consider fault β for fault detection



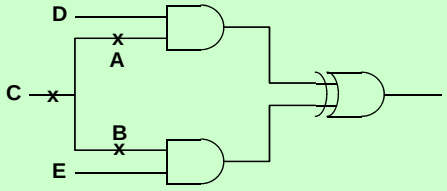
Fault Dominance

- **AND gate:**
  - Output *s-a-1* dominates any input *s-a-1*
- **NAND gate:**
  - Output *s-a-0* dominates any input *s-a-1*
- **OR gate:**
  - Output *s-a-0* dominates any input *s-a-0*
- **NOR gate:**
  - Output *s-a-1* dominates any input *s-a-0*
- **Dominance fault collapsing:**
  - The reduction of the set of faults to be analyzed based on dominance relation



Stem v.s. Branch Faults

C: stem of a multiple fanout
A & B: branches



- **Detect A sa1:**

$$z(t) \oplus z_f(t) = (CD \oplus CE) \oplus (D \oplus CE) = D \oplus CD = 1$$

$$\Rightarrow (C = 0,\ D = 1)$$

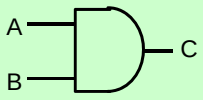
- **Detect C sa1:**

$$z(t) \oplus z_f(t) = (CD \oplus CE) \oplus (D \oplus E) = 1$$

$$\Rightarrow (C = 0,\ D = 1) \text{ or } (C = 0,\ E = 1)$$

- **Hence, C sa1 dominates A sa1**
- **Similarly**
  - C sa1 dominates B sa1
  - C sa0 dominates A sa0
  - C sa0 dominates B sa0
- **In general, there might be no equivalence or dominance relations between stem and branch faults**

Analysis of a Single Gate



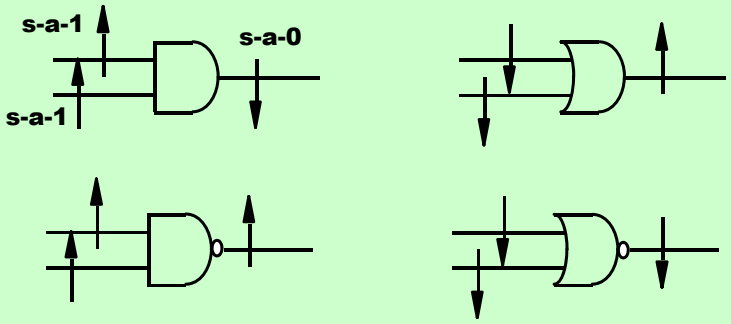
AB	C	A sa1	B sa1	C sa1	A sa0	B sa0	C sa0
00	0			1			
01	0	1		1			
10	0		1	1			
11	1				0	0	0

Negligible fault

- **Fault Equivalence Class**
  - (A s-a-0, B s-a-0, C s-a-0)
- **Fault Dominance Relations**
  - (C s-a-1 > A s-a-1) and (C s-a-1 > B s-a-1)
- **Faults that can be ignored:**
  - A s-a-0, B s-a-0, and C s-a-1

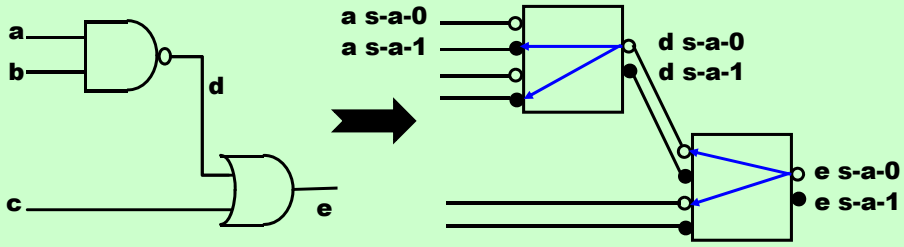
Fault Collapsing

- **Equivalence + Dominance**
  - For each *n*-input gate, we only need to consider *n+1* faults during test generation



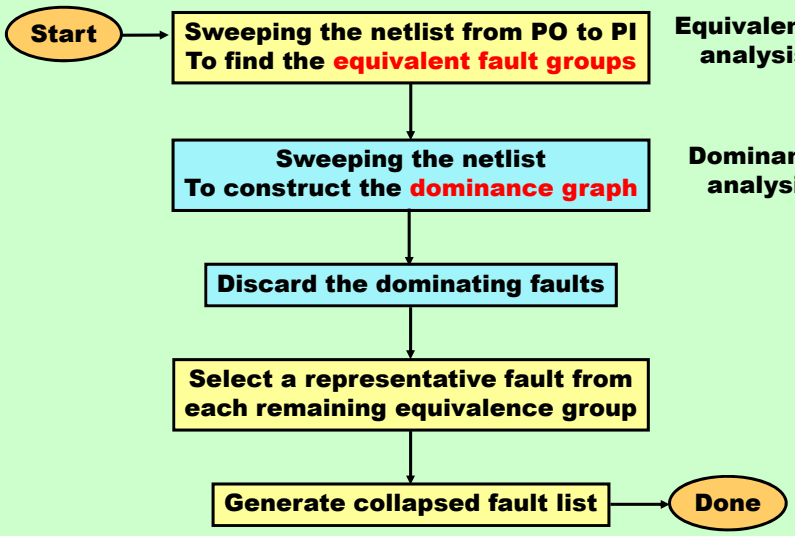
Dominance Graph

- **Rule**
  - When fault α dominates fault β, then an arrow is pointing from α to β
- **Application**
  - Find out the transitive dominance relations among faults



Ch2-49

Fault Collapsing Flow



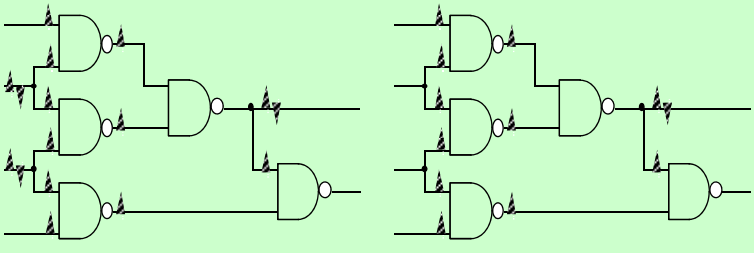
Ch2-50

Prime Fault

- α is a prime fault if every fault that is dominated by α is also equivalent to α

Why Fault Collapsing ?

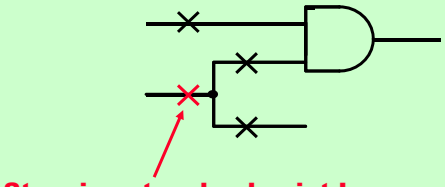
- Memory and CPU-time saving
- Ease testing generation and fault simulation



* 30 total faults → 12 prime faults

Checkpoint Theorem

- **Checkpoints for test generation**
  - A test set detects every fault on the primary inputs and fanout branches is complete
  - I.e., this test set detects all other faults too
  - Therefore, primary inputs and fanout branches form a *sufficient* set of checkpoints in test generation
  - In fanout-free combinational circuits, primary inputs are the sole checkpoints

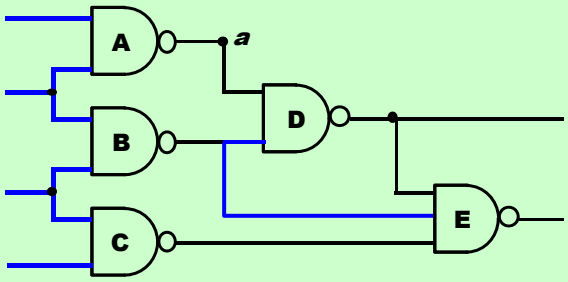


Stem is not a checkpoint !

Why Inputs + Branches Are Enough ?

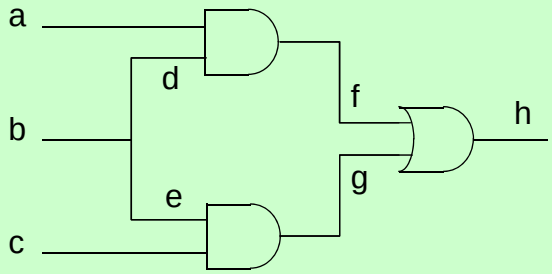
- **Example**
  - Checkpoints are marked in blue
  - Sweeping the circuit from PI to PO to examine every gate, e.g., based on an order of (A->B->C->D->E)
  - For each gate,

output faults are detected if every input fault is detected



Fault Collapsing + Checkpoint

- **Example:**
  - **10 checkpoint faults**
  - **a s-a-0 <=> d s-a-0 , c s-a-0 <=> e s-a-0**
    b s-a-0 > d s-a-0 , b s-a-1 > d s-a-1
  - **6 tests are enough**



Ch2-55