



---

# Metodologie di progetto HW

## Il test di circuiti digitali

Introduzione

---

---

# Metodologie di progetto HW

## Il test di circuiti digitali

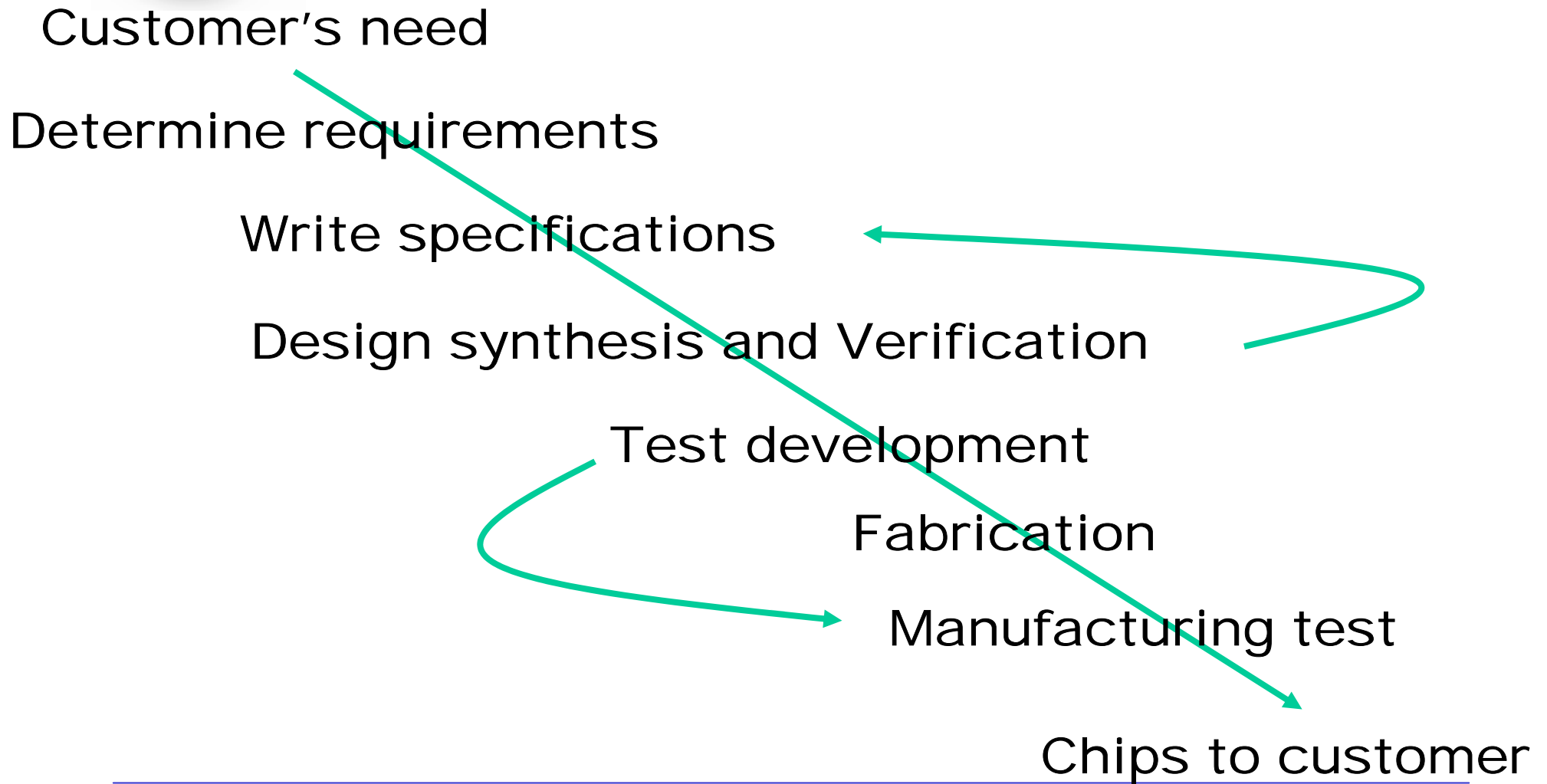
Introduction

---



# VLSI Realization Process

---





## Present and Future\*

---

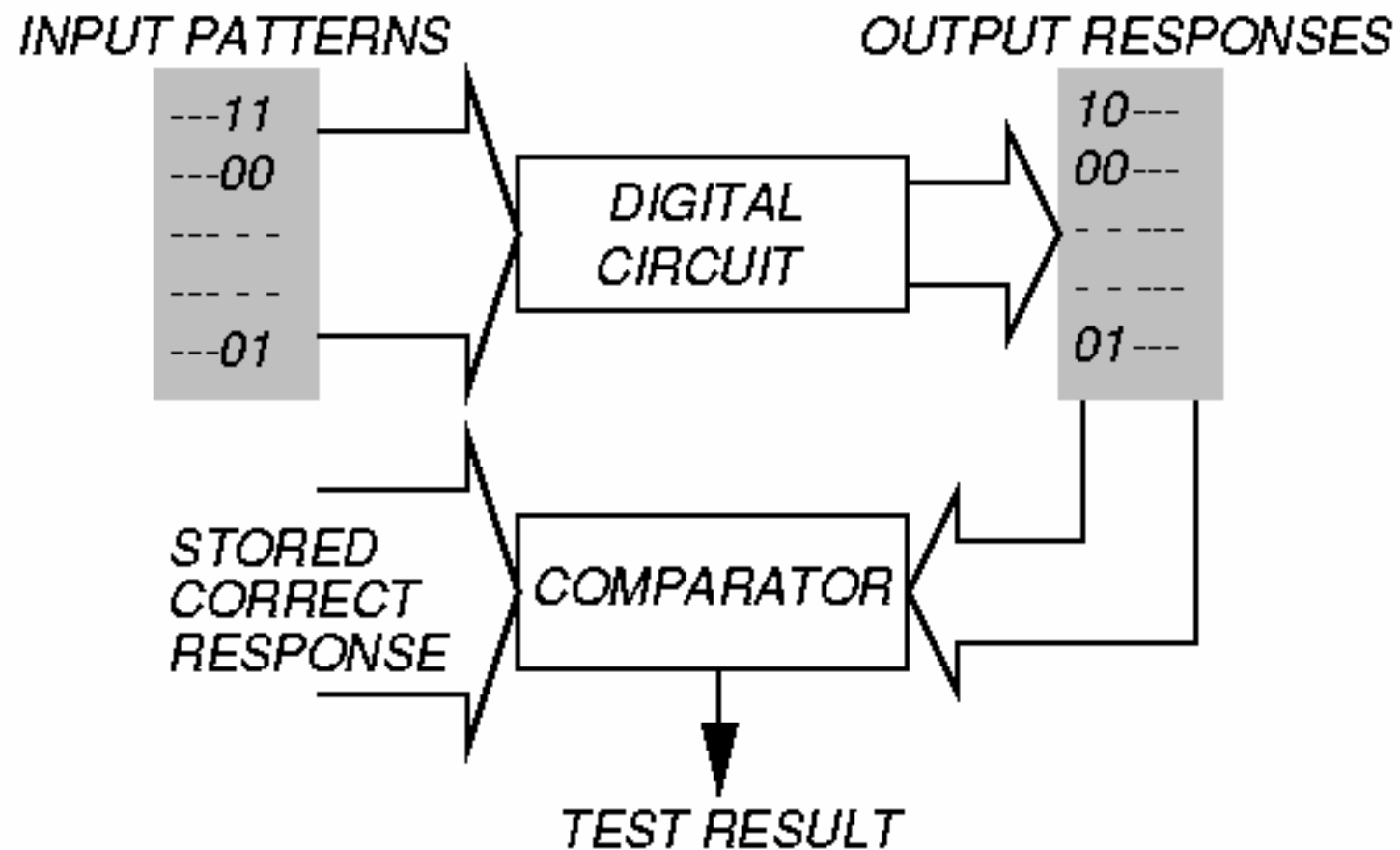
|                       | 1997 - 2001 | 2003 - 2006 |
|-----------------------|-------------|-------------|
| Feature size (micron) | 0.25 - 0.15 | 0.13 - 0.10 |
| Transistors/sq. cm    | 4 - 10M     | 18 - 39M    |
| Pin count             | 100 - 900   | 160 - 1475  |
| Clock rate (MHz)      | 200 - 730   | 530 - 1100  |
| Power (Watts)         | 1.2 - 61    | 2 - 96      |

\* SIA Roadmap, *IEEE Spectrum*, July 1999

---



## Testing Principle





## Contract between design house and fab vendor

---

- ❑ Design is complete and checked (verified)
- ❑ Fab vendor: How will you test it?
- ❑ Design house: I have checked it and ...
- ❑ Fab vendor: But, how would you test it?
- ❑ Design house: Why is that important?  
*complete the story*
  
- ❑ That is one reason for test generation etc.



## Contract between design ...

---

Hence:

- “Test” must be comprehensive
- It must not be “too long”

Issues:

- Model possible defects in the process
  - Understand the process
- Develop simulator and fault simulator
- Develop test generator
- Methods to quantify the test efficiency



# Verification v/s Testing

---

## Definitions

- ❑ Design synthesis: Given an I/O function, develop a procedure to manufacture a device using known materials and processes.
- ❑ Verification: Predictive analysis to ensure that the synthesized design, when manufactured, will perform the given I/O function.
- ❑ Test: A manufacturing step that ensures that the physical device, manufactured from the synthesized design, has no manufacturing defect.



## Need for testing

---

- ❑ Functionality issue
  - Does the circuit (large or small) work?
- ❑ Density issue
  - Higher density  $\Rightarrow$  higher failure prob
- ❑ Application issue
  - Life critical applications
- ❑ Maintenance issue
  - Need to identify failed components
- ❑ Cost of doing business
- ❑ What does testing achieve?
  - Discard only the “bad product”? - see next three slides



## Problems of Ideal Tests

---

- ❑ Ideal tests detect all defects produced in the manufacturing process.
- ❑ Ideal tests pass all functionally good devices.
- ❑ Very large numbers and varieties of possible defects need to be tested.
- ❑ Difficult to generate tests for some real defects.  
*Defect-oriented testing is an open problem.*



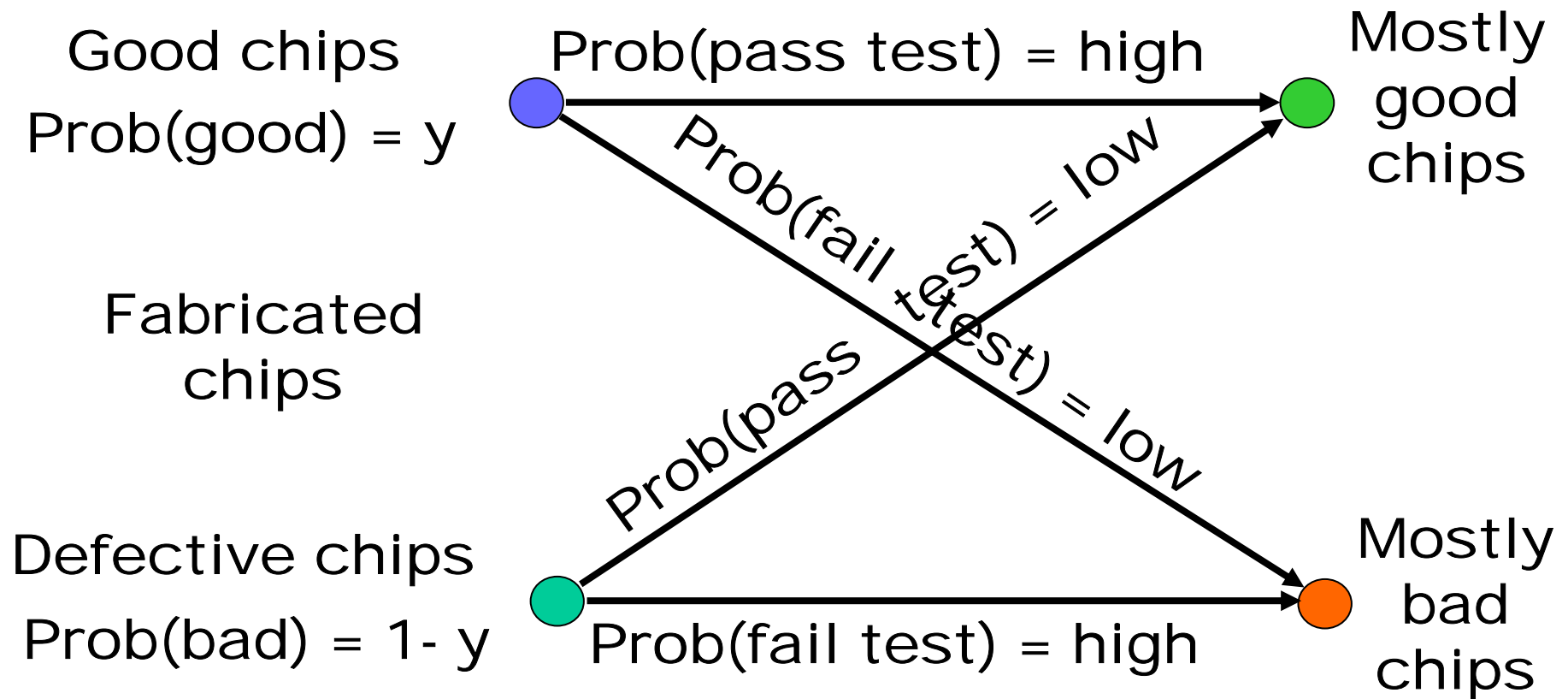
## Real Tests

---

- ❑ Based on analyzable fault models, which may not map on real defects.
- ❑ Incomplete coverage of modeled faults due to high complexity.
- ❑ Some good chips are rejected. The fraction (or percentage) of such chips is called the *yield loss*.
- ❑ Some bad chips pass tests. The fraction (or percentage) of bad chips among all passing chips is called the *defect level*.



## Testing as Filter Process





## Level of testing (1)

---

### □ Levels

- Chip
- Board
- System
  - Boards put together
  - System-on-Chip (SoC)
- System in field

### □ Cost - Rule of 10

- It costs 10 times more to test a device as we move to higher level in the product manufacturing process



## Cost of Manufacturing Testing in 2000AD

---

- ❑ 0.5-1.0GHz, analog instruments, 1024 digital pins: ATE purchase price
  - = \$1.2M + 1,024 x \$3,000 = \$4.272M
- ❑ Running cost (five-year linear depreciation)
  - = Depreciation + Maintenance + Operation
  - = \$0.854M + \$0.085M + \$0.5M
  - = \$1.439M/year
- ❑ Test cost (24 hour ATE operation)
  - = \$1.439M / (365 x 24 x 3,600)
  - = 4.5 cents/second



---

# Metodologie di progetto HW

## Il test di circuiti digitali

Fault Modeling

---



## Why Model Faults?

---

- ❑ I/O function tests inadequate for manufacturing (functionality versus component and interconnect testing)
- ❑ Real defects (often mechanical) too numerous and often not analyzable
- ❑ A fault model identifies targets for testing
- ❑ A fault model makes analysis possible
- ❑ Effectiveness measurable by experiments



# Some Real Defects in Chips

---

- Processing defects
  - Missing contact windows
  - Parasitic transistors
  - Oxide breakdown
  - . . .
- Material defects
  - Bulk defects (cracks, crystal imperfections)
  - Surface impurities (ion migration)
  - . . .
- Time-dependent failures
  - Dielectric breakdown
  - Electromigration
  - . . .
- Packaging failures
  - Contact degradation
  - Seal leaks
  - . . .

Ref.: M. J. Howes and D. V. Morgan, *Reliability and Degradation - Semiconductor Devices and Circuits*, Wiley, 1981.



## Observed PCB Defects

| Defect classes        | Occurrence frequency (%) |
|-----------------------|--------------------------|
| Shorts                | 51                       |
| Opens                 | 1                        |
| Missing components    | 6                        |
| Wrong components      | 13                       |
| Reversed components   | 6                        |
| Bent leads            | 8                        |
| Analog specifications | 5                        |
| Digital logic         | 5                        |
| Performance (timing)  | 5                        |

Ref.: J. Bateson, *In-Circuit Testing*, Van Nostrand Reinhold, 1985.



# Common Fault Models

---

- ❑ Single stuck-at faults
- ❑ Transistor open and short faults
- ❑ Memory faults
- ❑ PLA faults (stuck-at, cross-point, bridging)
- ❑ Functional faults (processors)
- ❑ Delay faults (transition, path)
- ❑ Analog faults



## Stuck-at Faults

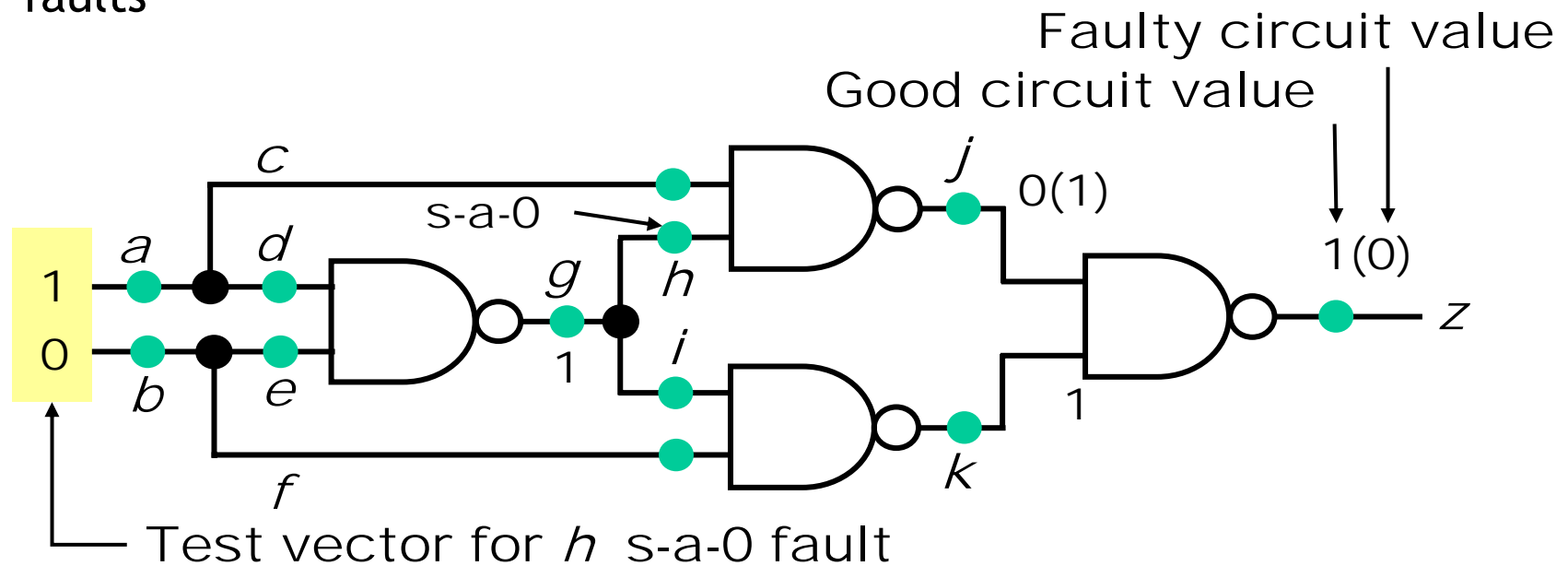
---

- ❑ Single stuck-at faults
- ❑ What does it achieve in practice?
- ❑ Fault equivalence
- ❑ Fault dominance and checkpoint theorem
- ❑ Classes of stuck-at faults and multiple faults



## Single Stuck-at Fault

- Three properties define a single stuck-at fault
  - Only one line is faulty
  - The faulty line is permanently set to 0 or 1
  - The fault can be at an input or output of a gate
- Example: XOR circuit has 12 fault sites (●) and 24 single stuck-at faults





## Single Stuck-at Faults (contd.)

---

- How effective is this model?
  - Empirical evidence supports the use of this model
  - Has been found to be effective to detect other types of faults
  - Relates to yield modeling
  - Simple to use



## Origins of Stuck-Faults

---

- ❑ Eldred (1959) - **First use** of structural testing for the Honeywell Datamatic 1000 computer
- ❑ Galey, Norby, **Roth** (1961) - First publication of stuck-at-0 and stuck-at-1 faults
- ❑ Seshu & **Freeman** (1962) - Use of stuck-faults for parallel fault simulation
- ❑ **Poage** (1963) - Theoretical analysis of stuck-at faults



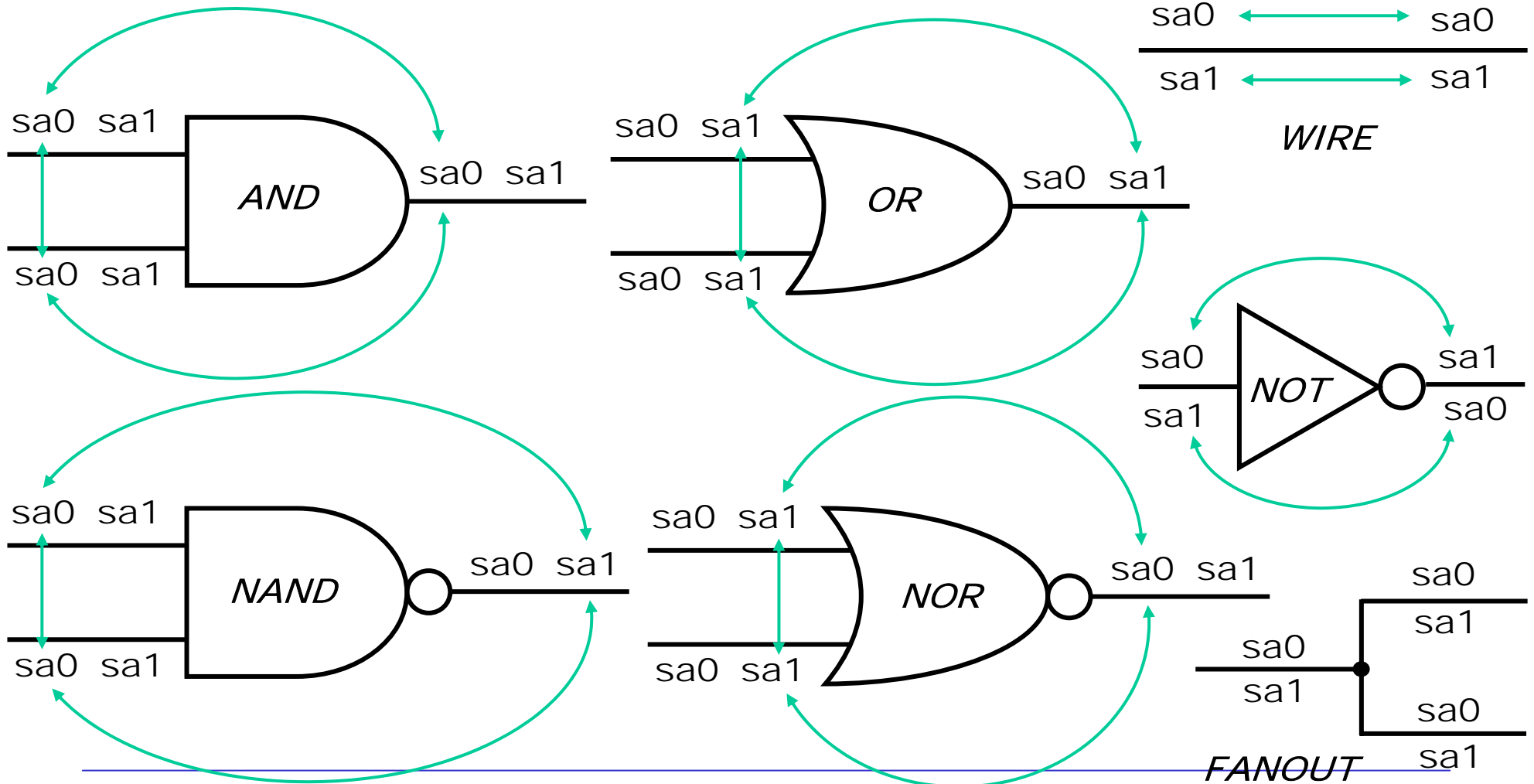
## Fault Equivalence

---

- Number of fault sites in a Boolean gate circuit =  $\#PI + \#gates + \#$  (fanout branches).
- Fault equivalence: Two faults  $f1$  and  $f2$  are equivalent if all tests that detect  $f1$  also detect  $f2$ .
- If faults  $f1$  and  $f2$  are equivalent then the corresponding faulty functions are identical.
- Fault collapsing: All single faults of a logic circuit can be divided into disjoint equivalence subsets, where all faults in a subset are mutually equivalent. A collapsed fault set contains one fault from each equivalence subset.

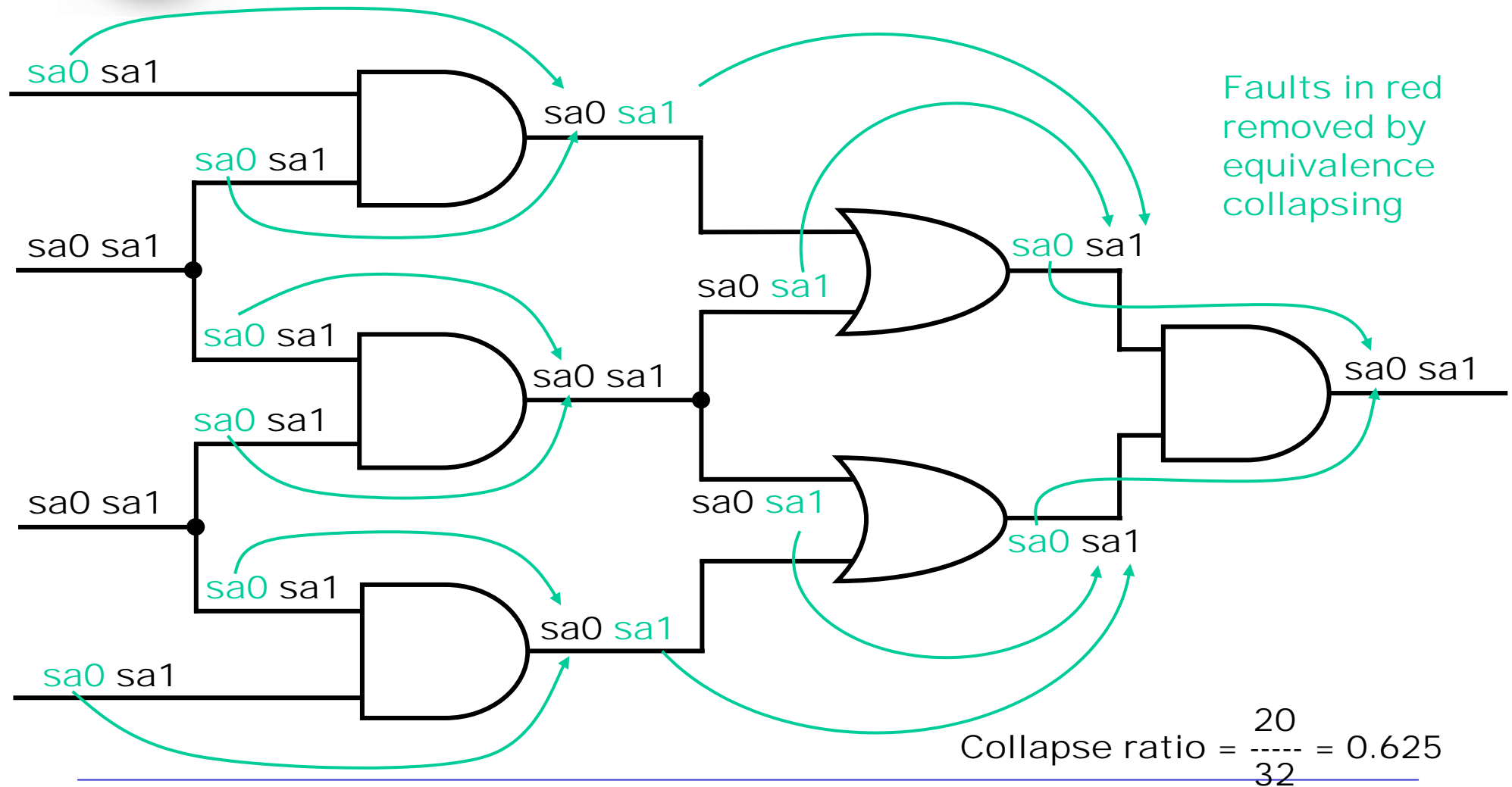


## Equivalence Rules





## Equivalence Example





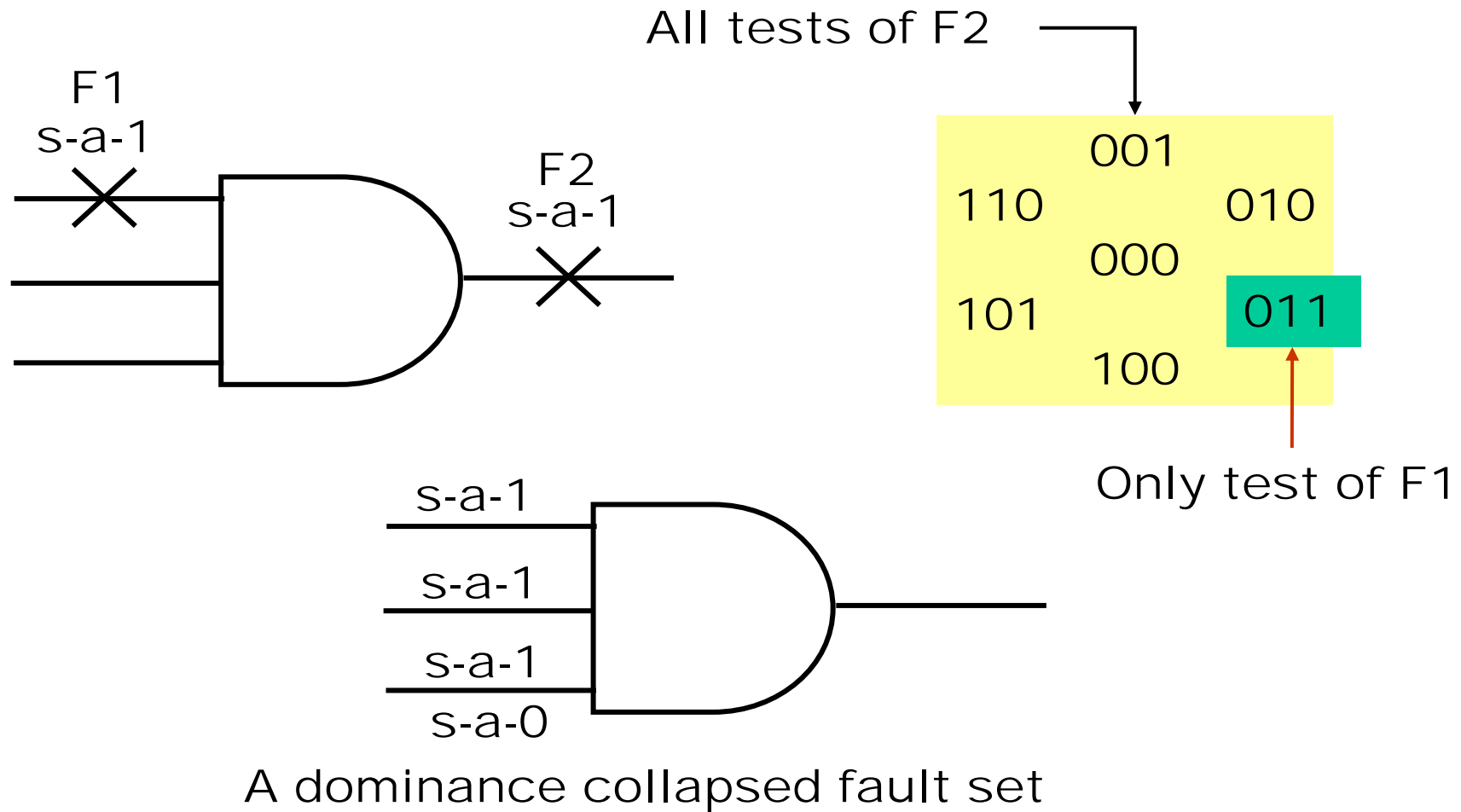
## Fault Dominance

---

- If all tests of some fault F1 detect another fault F2, then F2 is said to dominate F1.
- **Dominance fault collapsing**: If fault F2 dominates F1, then F2 is removed from the fault list.
- When dominance fault collapsing is used, it is sufficient to consider only the **input faults** of Boolean gates. See the next example.
- In a tree circuit (without fanouts) PI faults form a dominance collapsed fault set.
- If **two** faults **dominate** each other then they are **equivalent**.



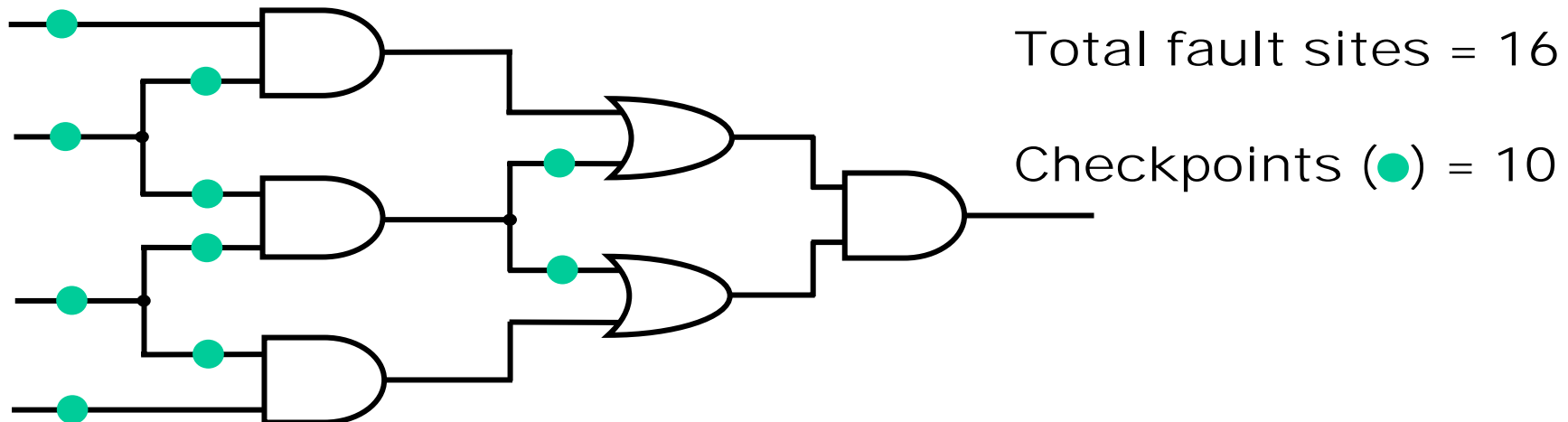
## Dominance Example





## Checkpoints

- 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.





## Classes of Stuck-at Faults

---

- Following **classes** of single stuck-at faults are identified by fault simulators:
  - *Potentially-detectable fault* -- Test produces an unknown (X) state at *primary output* (PO); detection is probabilistic, usually with 50% probability.
  - *Initialization fault* -- Fault prevents initialization of the faulty circuit; can be detected as a potentially-detectable fault.
  - *Hyperactive fault* -- Fault induces much internal signal activity without reaching PO.
  - *Redundant fault* -- No test exists for the fault.
  - *Untestable fault* -- Test generator is unable to find a test.



## Multiple Stuck-at Faults

---

- ❑ A multiple stuck-at fault means that **any set of lines** is stuck-at some combination of (0,1) values.
- ❑ The total number of single and multiple stuck-at faults in a circuit with  $k$  single fault sites is  $3^k - 1$ .
- ❑ A single fault test can fail to detect the target fault if another fault is also present, however, such masking of one fault by another is **rare**.
- ❑ **Statistically**, single fault tests cover a very large number of multiple faults.



## Transistor (Switch) Faults

---

- ❑ MOS transistor is considered an ideal switch and two types of faults are modeled:
  - **Stuck-open** -- a single transistor is permanently stuck in the open state.
  - **Stuck-short** -- a single transistor is permanently shorted irrespective of its gate voltage.
- ❑ Detection of a stuck-open fault requires **two vectors**.
- ❑ Detection of a stuck-short fault requires the measurement of **quiescent current** ( $I_{DDQ}$ ).

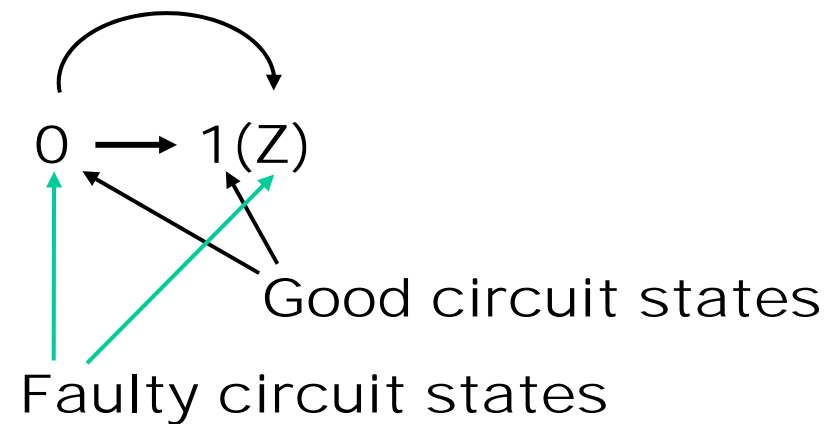
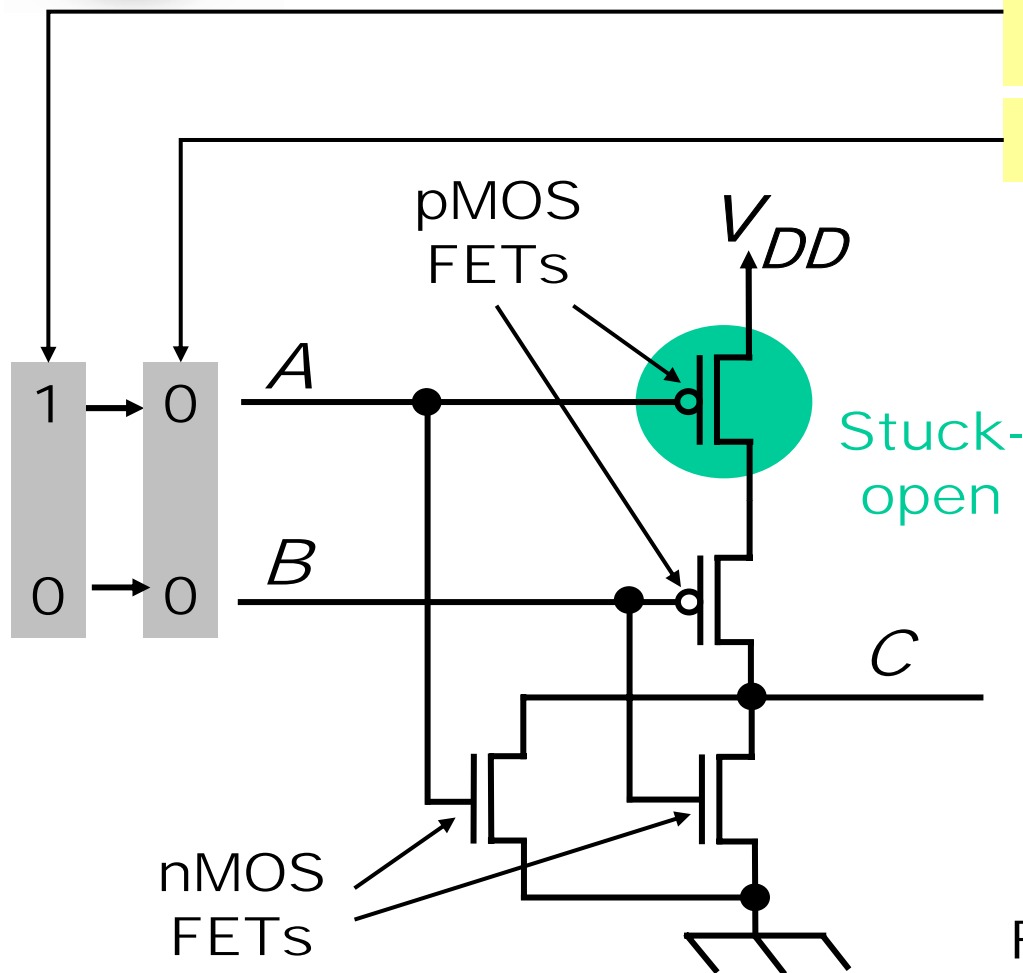


## Stuck-Open Example

Vector 1: test for  $A$  s-a-0  
(Initialization vector)

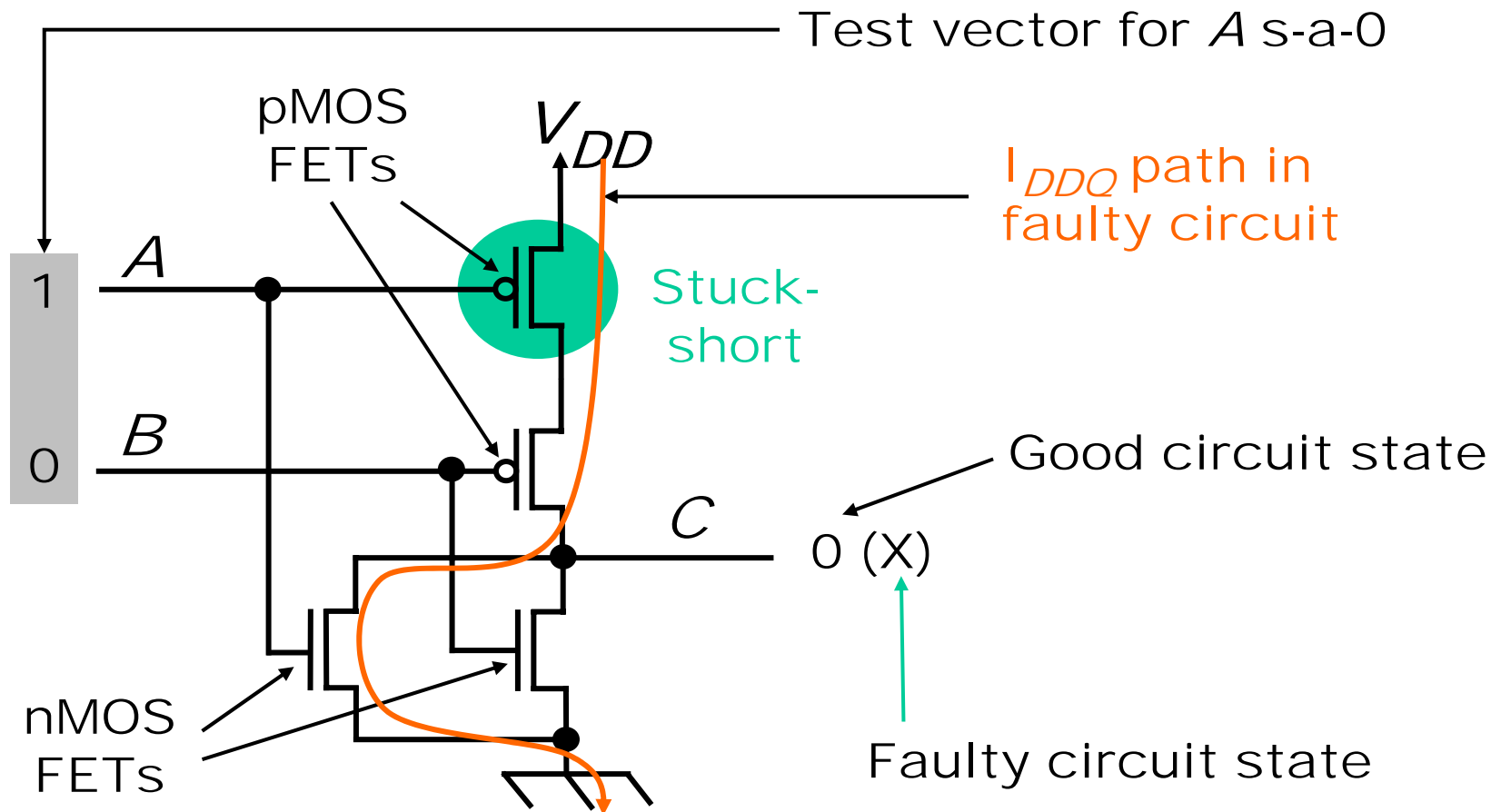
Vector 2 (test for  $A$  s-a-1)

*Two-vector s-op test  
can be constructed by  
ordering two s-at tests*





## Stuck-Short Example





## Summary

---

- ❑ Gate level models are most prevalent in logic testing
- ❑ Fault models are analyzable approximations of defects and are essential for a test methodology.
- ❑ For digital logic single stuck-at fault model offers best advantage of tools and experience.
- ❑ Many other faults (bridging, stuck-open and multiple stuck-at) are largely covered by stuck-at fault tests.
- ❑ Stuck-short and delay faults and technology-dependent faults require special tests.
- ❑ Memory and analog circuits need other specialized fault models and tests.



---

# Metodologie di progetto HW

## Il test di circuiti digitali

Fault Simulation

---



# Overview

---

- ❑ Problem and motivation
- ❑ Fault simulation algorithms
  - Serial
  - Parallel
  - Deductive
  - Concurrent
  - Other algorithms
- ❑ Random Fault Sampling
- ❑ Summary



# Problem and Motivation

---

## □ Fault simulation Problem: Given

- A circuit
- A sequence of test vectors
- A fault model

### - Determine

- Fault coverage - fraction (or percentage) of modeled faults detected by test vectors
- Set of undetected faults

## □ Motivation

- Determine test quality and in turn product quality
- Find undetected fault targets to improve tests



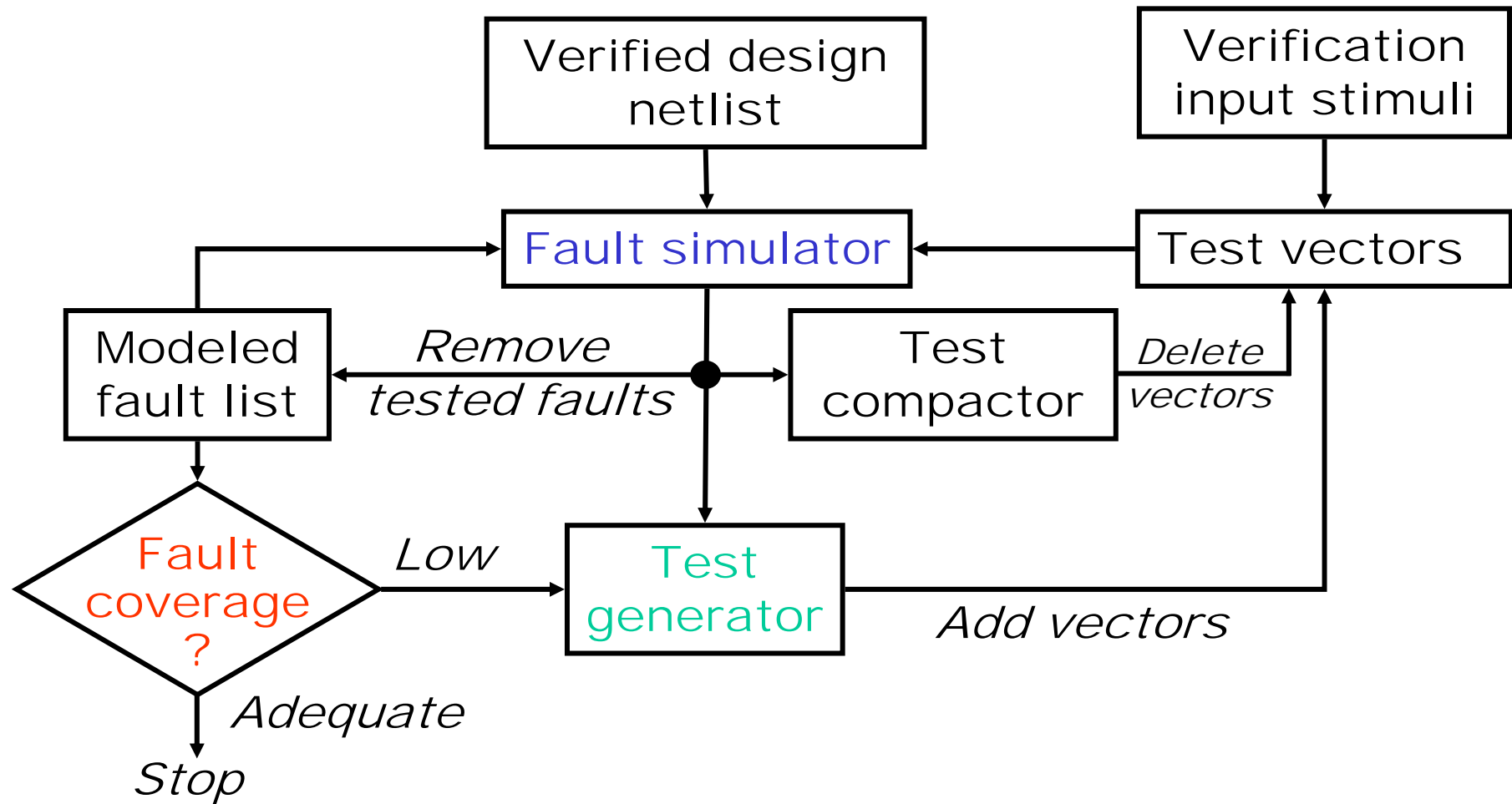
## Usages of Fault Simulators

---

- ❑ Test grading
- ❑ Test Generation
- ❑ Fault diagnosis
- ❑ Design for test (DFT) - identification of points that may help improve test quality
- ❑ Fault-tolerance - identification of damage a fault can cause



# Fault Simulator in a VLSI Design Process





# Fault Simulation Scenario

---

- **Circuit model:** mixed-level
  - Mostly logic with some switch-level for high-impedance (Z) and bidirectional signals
  - High-level models (memory, etc.) with pin faults
- **Signal states:** logic
  - Two (0, 1) or three (0, 1, X) states for purely Boolean logic circuits
  - Four states (0, 1, X, Z) for sequential MOS circuits
- **Timing:**
  - Zero-delay for combinational and synchronous circuits
  - Mostly unit-delay for circuits with feedback



## Fault Simulation Scenario (continued)

---

### □ Faults:

- Mostly **single stuck-at** faults
- Sometimes **stuck-open, transition, and path-delay** faults; analog circuit fault simulators are not yet in common use
- **Equivalence** fault collapsing of single stuck-at faults
- **Fault-dropping** -- a fault once detected is dropped from consideration as more vectors are simulated; fault-dropping may be suppressed for diagnosis
- **Fault sampling** -- a random sample of faults is simulated when the circuit is large



## Fault Simulation Algorithms

---

- ❑ Serial
- ❑ Parallel
- ❑ Deductive
- ❑ Concurrent
- ❑ Others
  - Differential
  - Parallel pattern
  - etc.



## Serial Algorithm

---

- **Algorithm:** Simulate fault-free circuit and save responses. Repeat following steps for each fault in the fault list:
  - Modify netlist by **injecting** one fault
  - **Simulate** modified netlist, vector by vector, comparing responses with saved responses
  - If **response** differs, report fault detection and suspend simulation of remaining vectors
- **Advantages:**
  - **Easy** to implement; needs only a true-value simulator, less memory
  - **Most faults**, including analog faults, **can be simulated**



# Fault Injection

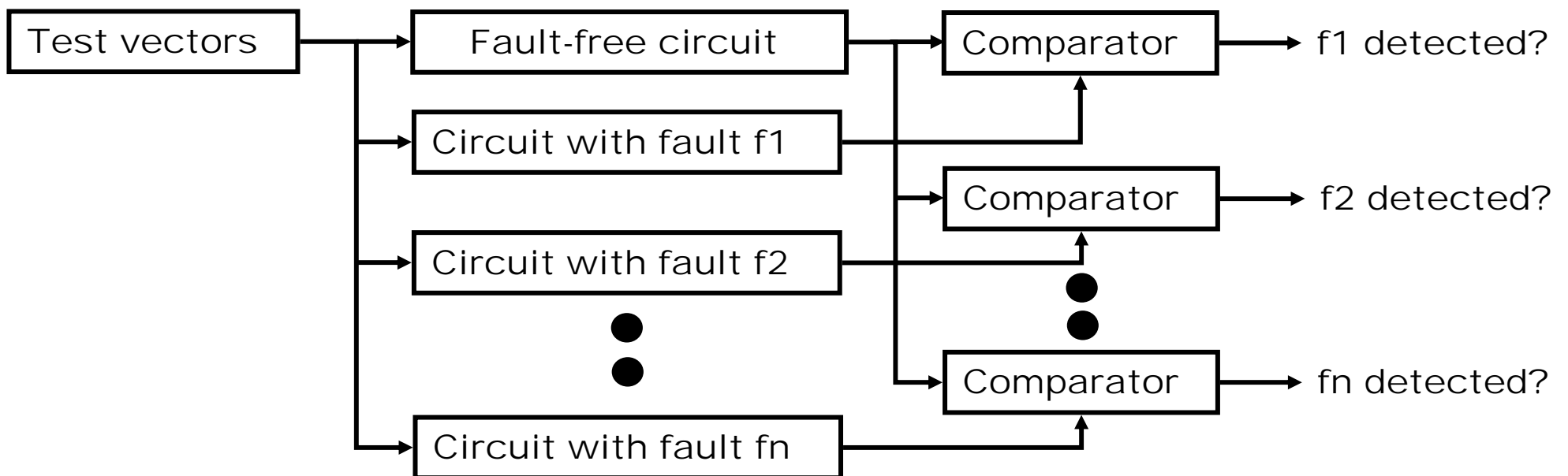
---

- ❑ **Modifying** netlist for every run can be expensive
- ❑ Alternative
  - **Check** if a net is faulty or fault-free
    - If faulty change its value to the stuck-value
    - Else leave it to the computed value
  - **Mux** based fault insertion
    - Use additional variables and computed the value based on the signal value and the value in the additional variable



## Serial Algorithm (Cont.)

- ❑ **Disadvantage:** Much repeated computation; CPU time prohibitive for VLSI circuits
- ❑ Alternative: **Simulate many faults together**





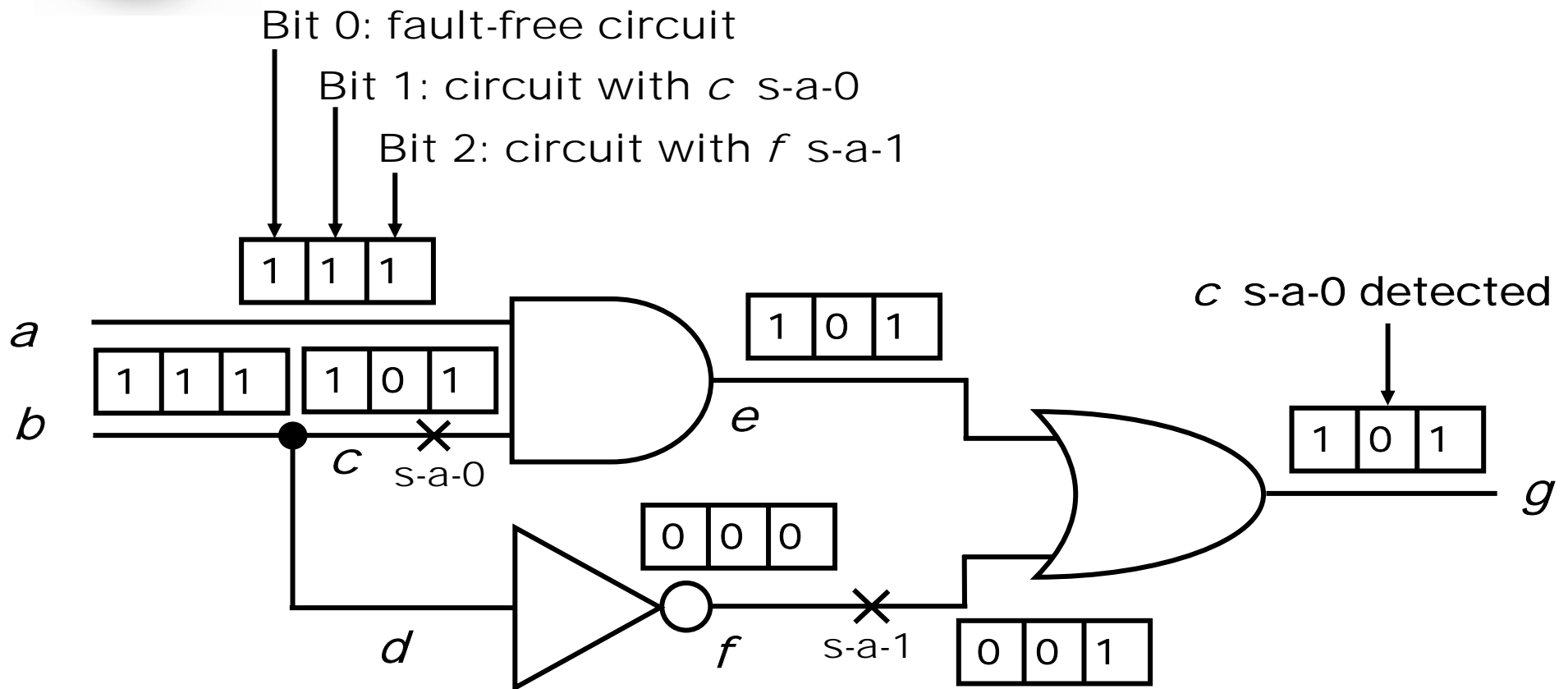
## Parallel Fault Simulation

---

- ❑ **Compiled-code** method; best with **two-states** (0,1)
- ❑ Exploits inherent bit-parallelism of logic operations on **computer words**
- ❑ **Storage**: one word per line for two-state simulation
- ❑ **Multi-pass simulation**: Each pass simulates  $w-1$  new faults, where  $w$  is the machine word length
- ❑ **Speed up** over serial method  $\sim w-1$
- ❑ **Not suitable** for circuits with timing-critical and non-Boolean logic



## Parallel Fault Sim. Example





# Deductive Fault Simulation

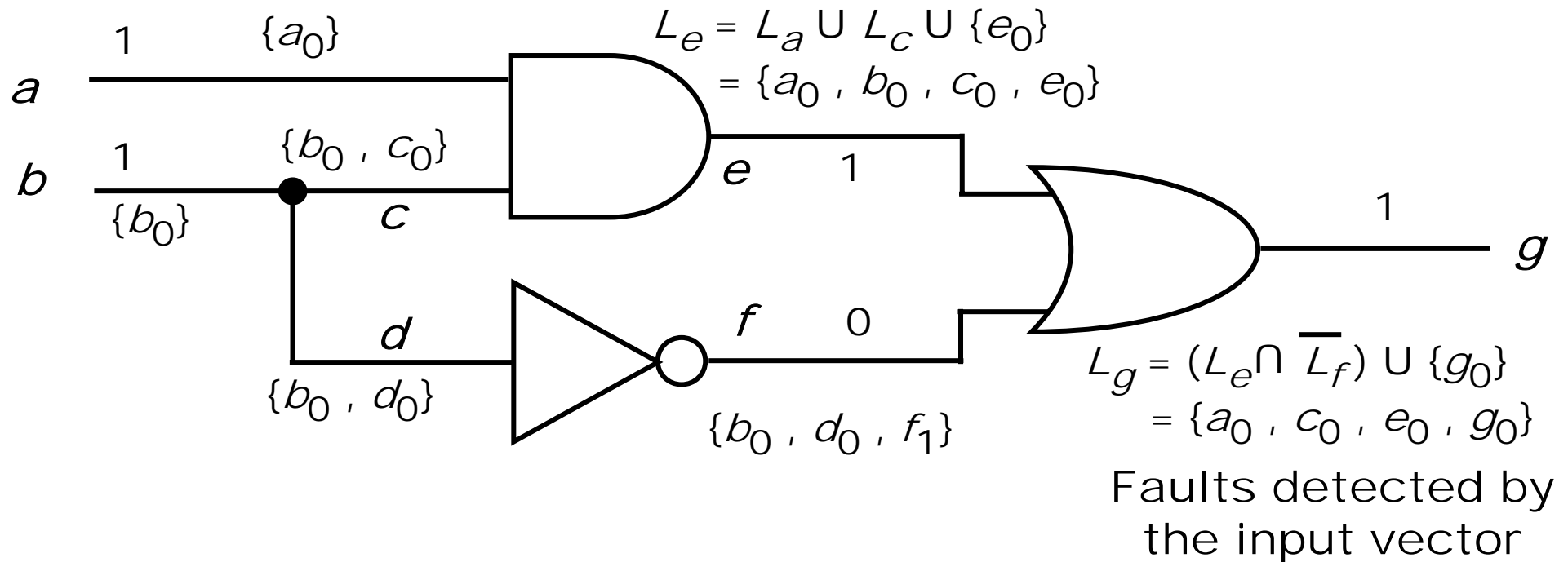
---

- ❑ One-pass simulation
- ❑ Each line  $k$  contains a list  $L_k$  of faults detectable on  $k$
- ❑ Following true-value simulation of each vector, fault lists of all gate output lines are updated using **set-theoretic rules**, signal values, and gate input fault lists
- ❑ PO fault lists provide detection data
- ❑ **Limitations:**
  - Set-theoretic rules difficult to derive for non-Boolean gates
  - Gate delays are difficult to use



## Deductive Fault Simulation: Example

Notation:  $L_k$  is fault list for line  $k$   
 $k_n$  is s-a-n fault on line  $k$





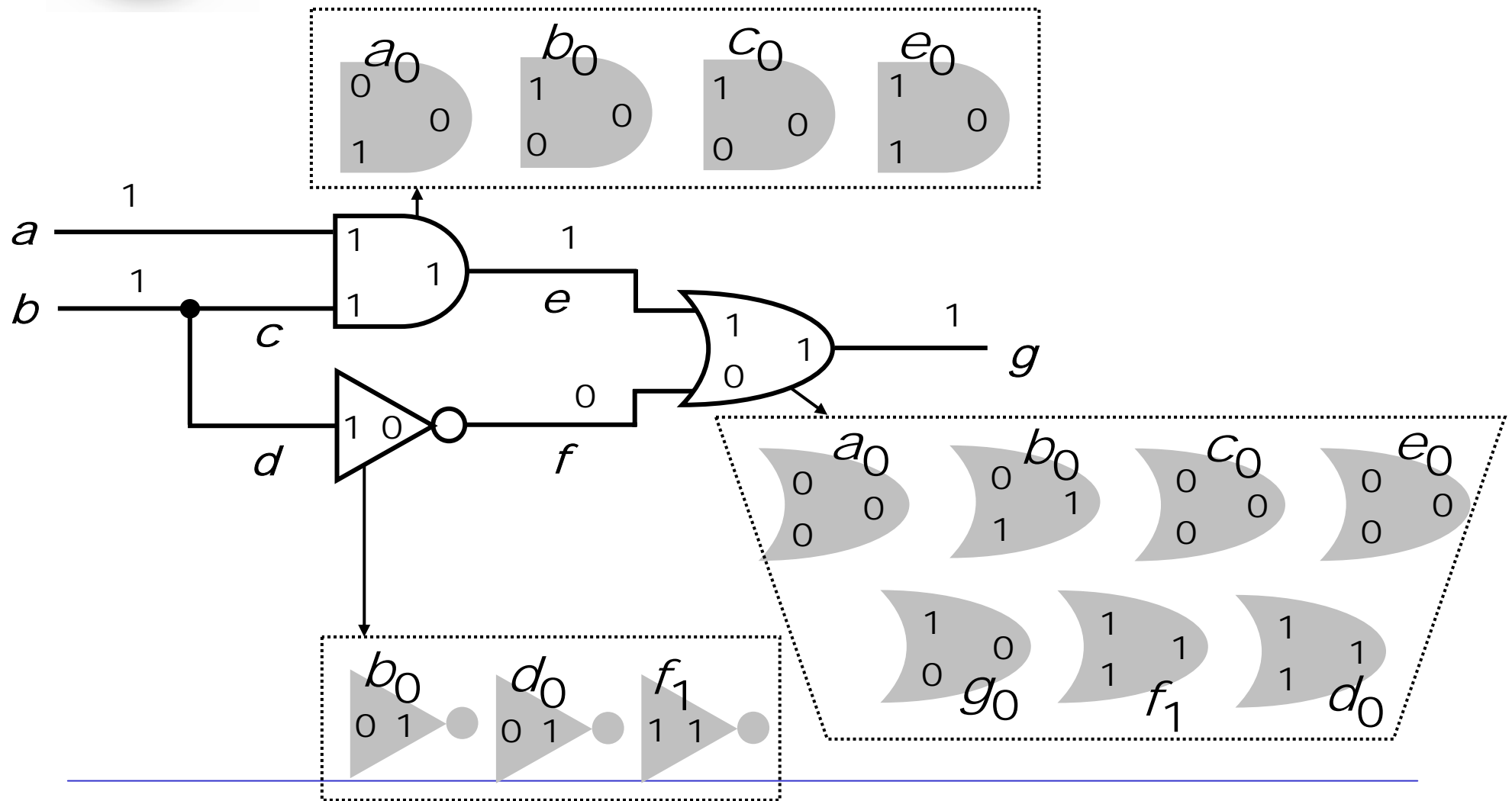
## Concurrent Fault Simulation

---

- ❑ **Event-driven simulation** of fault-free circuit and only those parts of the faulty circuit that differ in signal states from the fault-free circuit.
- ❑ **A list per gate** containing copies of the gate from all faulty circuits in which this gate differs. List element contains fault ID, gate input and output values and internal states, if any.
- ❑ **All events** of fault-free and all faulty circuits are implicitly simulated.
- ❑ Faults can be simulated in **any modeling style** or detail supported in true-value simulation (offers most flexibility.)
- ❑ **Faster** than other methods, but **uses most memory**.



## Conc. Fault Sim. Example





## Other Fault Simulation Algorithms

---

- Parallel pattern single fault simulation (PPSFP)
  - Simulate **many vectors in parallel**
  - Inject only **one fault** - hence one event
  - Simulate the circuit from the fault site
  - Limitation - well suited for combinational circuits only



## Fault Sampling

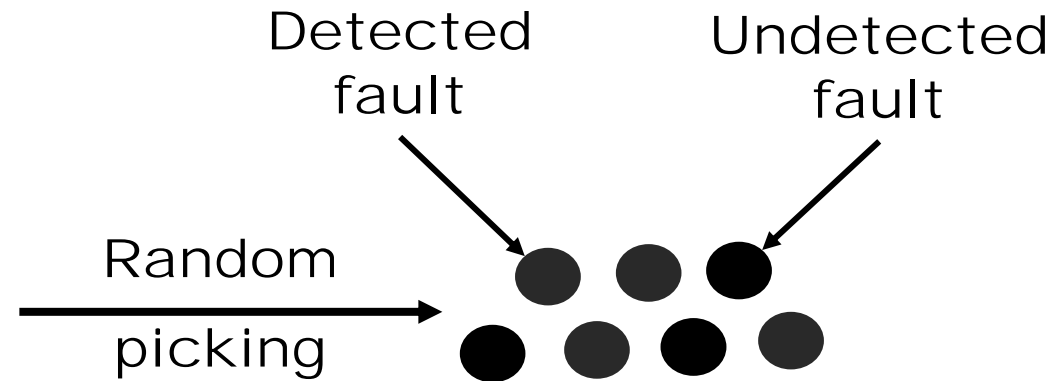
---

- ❑ A randomly selected subset (**sample**) of faults is simulated
- ❑ Measured coverage in the sample is used to **estimate** fault coverage in the entire circuit
- ❑ **Advantage**: Saving in computing resources (CPU time and memory.)
- ❑ **Disadvantage**: Limited data on undetected faults
- ❑ Complexity of **fault simulation** depends on:
  - Number of gates
  - Number of faults
  - Number of vectors



## Random Sampling Model

All faults with  
a fixed but  
unknown  
coverage



$N_p$  = total number of faults  
(population size)

$C$  = fault coverage (unknown)

$N_s$  = sample size

$$N_s \ll N_p$$

$c$  = sample coverage  
(a random variable)



## Summary

---

- ❑ Fault simulator is an **essential tool** for test development
- ❑ **Concurrent** fault simulation algorithm offers the best choice
- ❑ For restricted class of circuits (combinational and synchronous sequential with only Boolean primitives), **deductive** algorithm can provide better speed and memory efficiency
- ❑ For large circuits, the accuracy of random **fault sampling** only depends on the sample size (1,000 to 2,000 faults) and not on the circuit size. The method has significant advantages in reducing CPU time and memory needs of the simulator