VLSI Testing
積體電路測試

Combinational ATPG
(Automatic Test Pattern Generation)

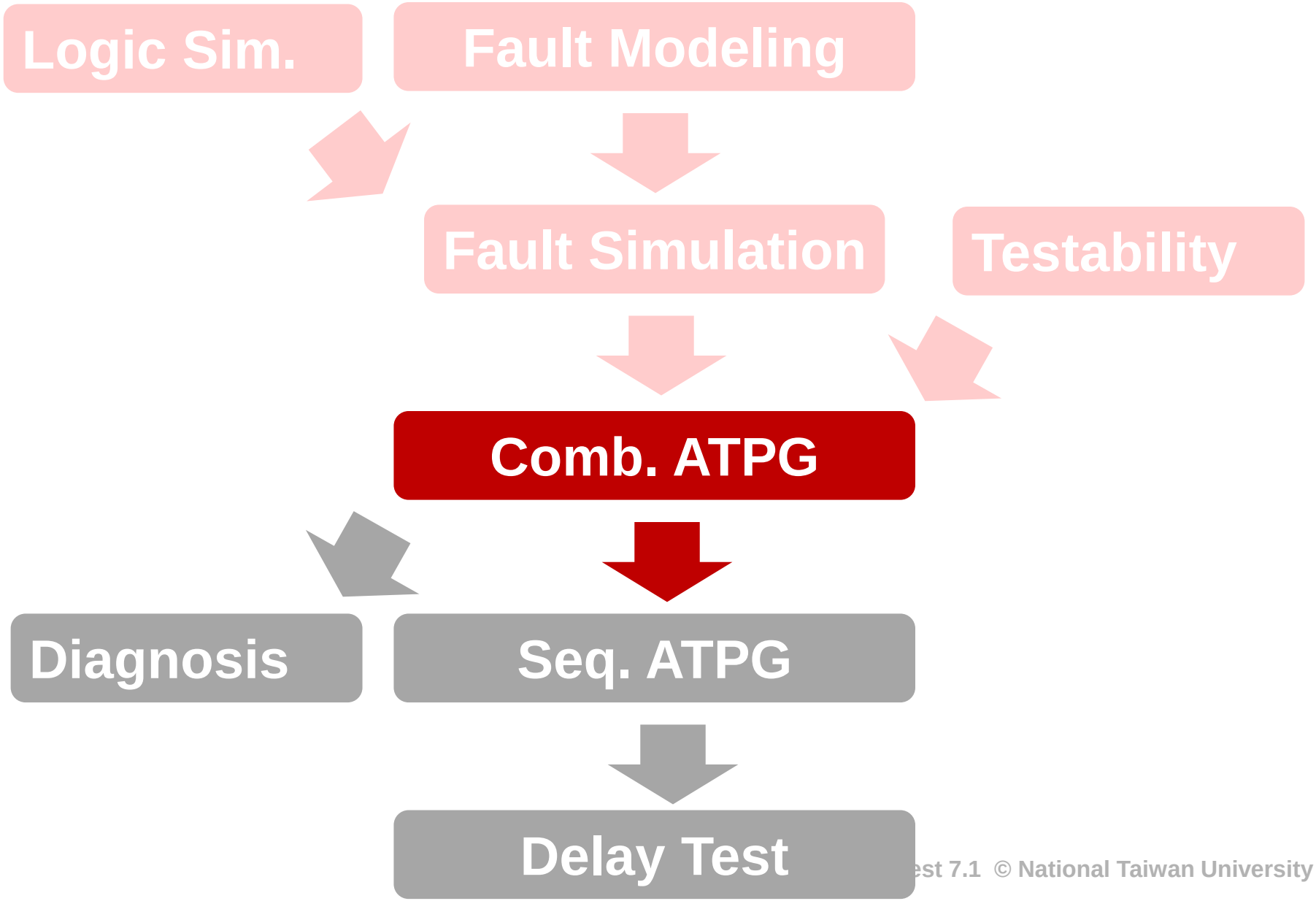
Professor James Chien-Mo Li 李建模

Lab. of Dependable Systems

Graduate Institute of Electronics Engineering
National Taiwan University

* Some pictures are courtesy of Prof. Jiun-Lang Huang, NTU

Course Roadmap (EDA Topics)



Why Am I Learning This?

- **Automatic Test Pattern Generation (ATPG)**
  - **1. Generate high quality test patterns**
  - **2. Reduce Human efforts**

"Testing a product is a learning process."

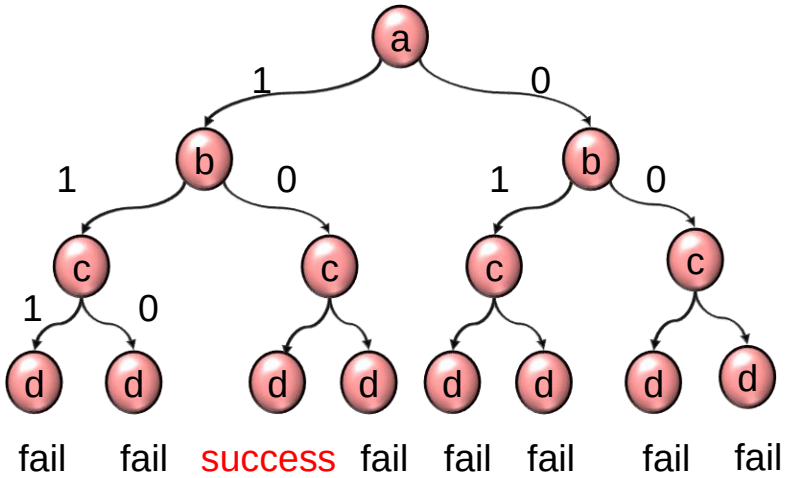
(Brian Marick)

Test Generation

Fault Models	Combinational Circuits (seq. ckt. w/ scan)	Sequential Circuits
No fault model	PET	Checking experiment
Single Stuck-at Fault Model	D PODEM FAN	Extended D 9-valued
Delay Fault Model	Path delay Transition delay	Launch on capture Launch on shift

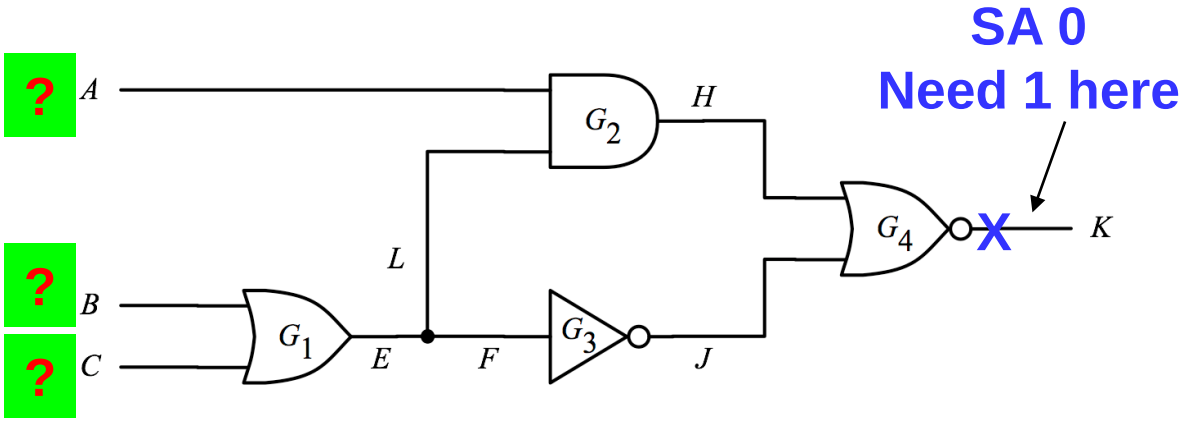
Combinational ATPG

- **Introduction**
- **Deterministic Test Pattern Generation**
- **Acceleration Techniques**
- **Concluding Remarks**



Motivating Problem

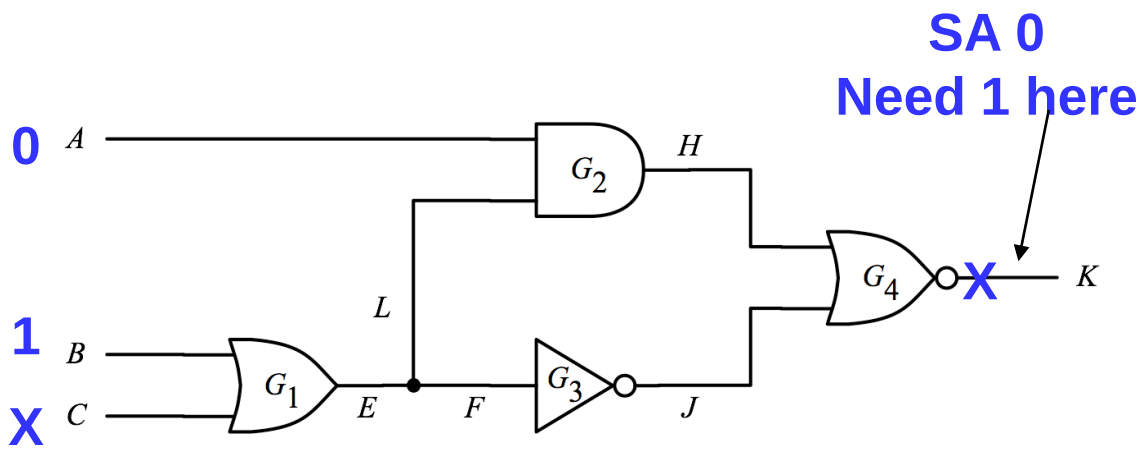
- **Your manger asks you to generate a test pattern for output stuck-at zero fault. What is your answer? (Maybe more than one answer)**



- **Simulation is like *substitution*, easy**
  - ***f(x)=x⁵+3x⁴+2*, x=3.2 ⇒ *f(x)*=?**
- **Finding test pattern is like *finding roots,* very hard**
  - ***f(x)=x⁵+3x⁴+2, f(x)=0*, ⇒ *x*=?**

What is Complexity of TPG?

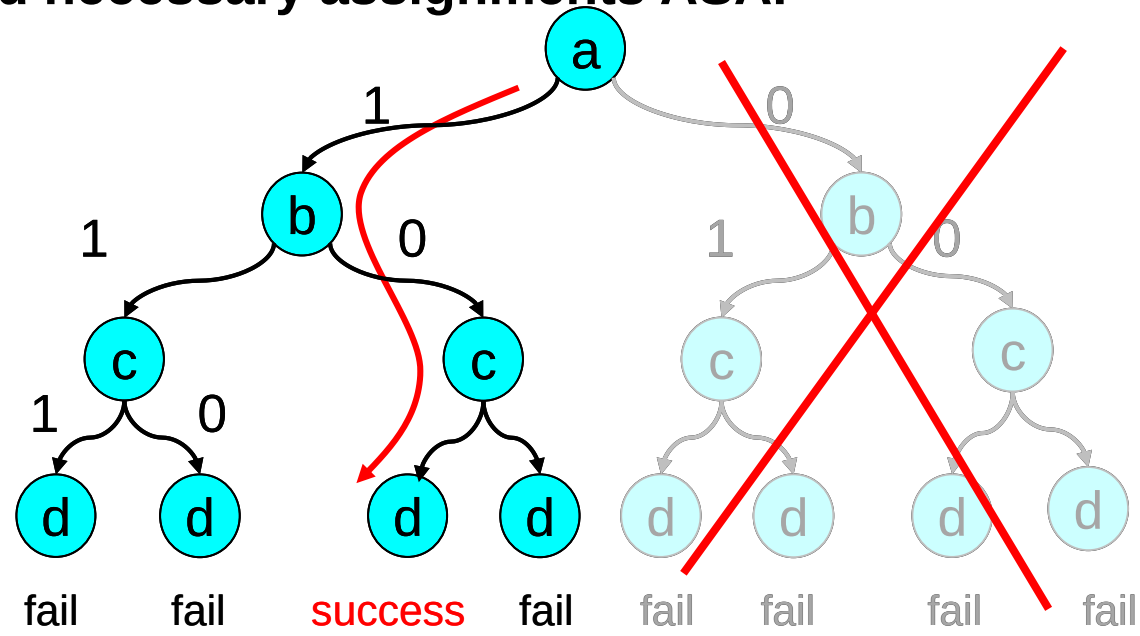
- *Test pattern generation TPG* (aka. *test generation*)
  - generate test patterns for a given fault model
- Generating a test pattern for a fault is *NP-complete* [Ibarra 75]
  - Same as *satisfiability* problem
  - Worst case 2^{number_PI} assignments to try
- Need automatic tool, *ATPG* (*Automatic Test Pattern Generator*)



TPG is NP-complete

ATPG is Decision Problem

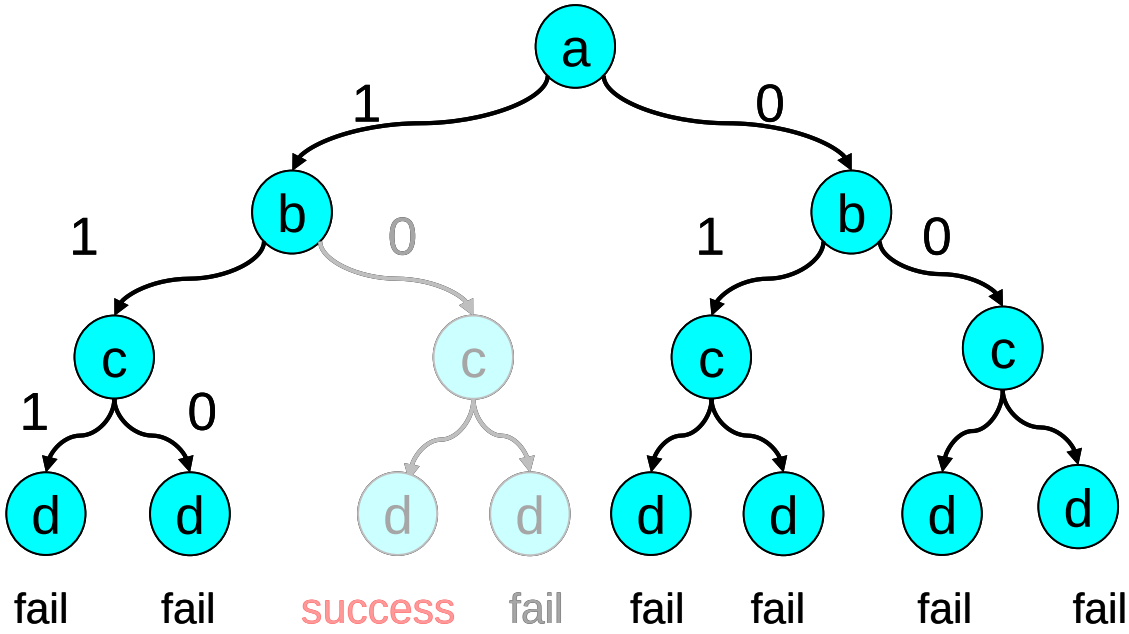
- Huge binary decision tree (exponential size!)
  - There could be *one, many, or even zero* answers
- Need smart heuristic to speed up
  - 1. Prune impossible sub-trees ASAP
  - 2. Find necessary assignments ASAP



Search a Test Pattern in a Huge Tree

Complete ATPG Algorithm

- *Complete ATPG* exhausts the whole search tree (2^n)
  - If a test pattern exists, complete ATPG will find it for sure
- *Incomplete ATPG* does NOT exhaust the whole search tree
  - may not find solution even though a solution DOES exist



Complete ATPG Guarantees to Find Solution (if it exists)

ATPG Components

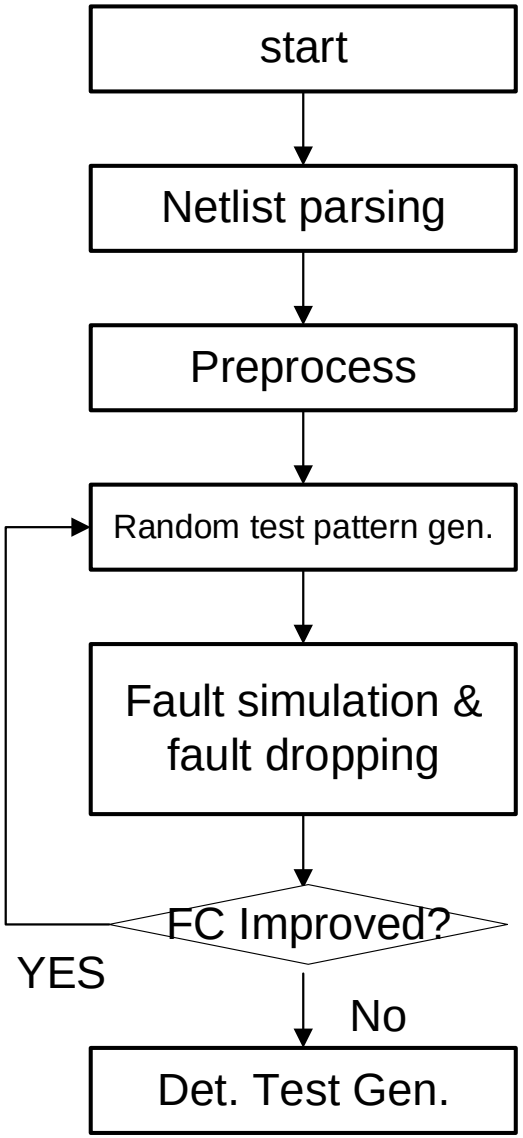
- Netlist parsing
  - Create database
  - Design rule checking
  - Levelization and etc.
- Perprocess
  - Fault collapsing
  - Testability analysis
  - **Learning**
  - **Redundant fault identification**
- **Test Generation**
  - **Random test pattern generation**
  - **Deterministic test pattern generation**
- Test Compaction
  - Dynamic test compaction
  - Static test compaction

Previous chapter

This chapter

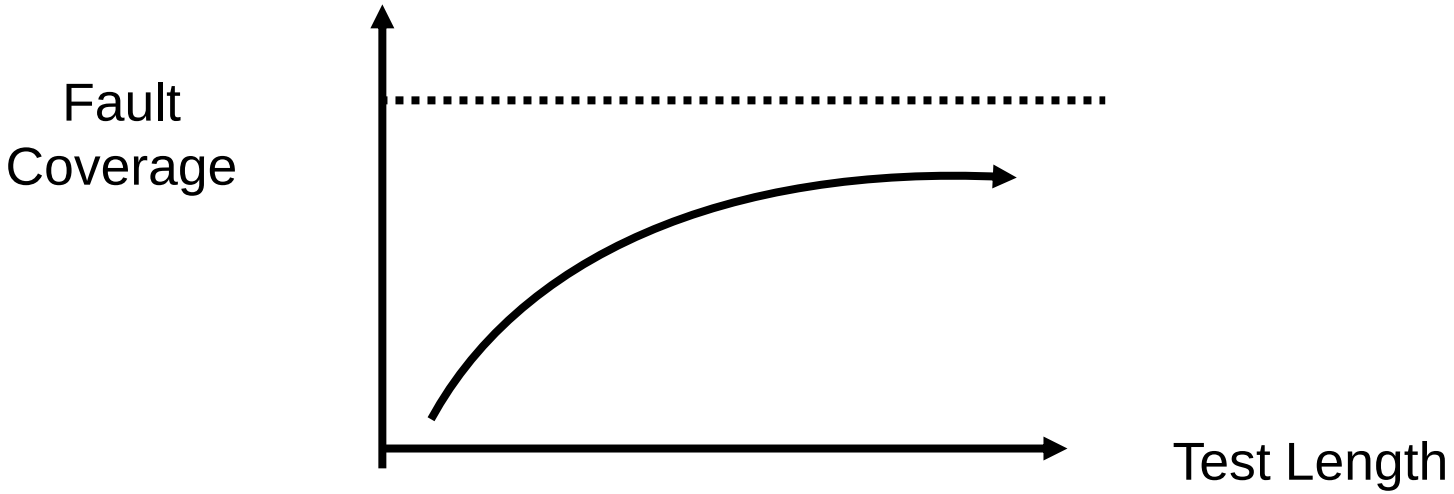
See test compaction

Random Test Pattern Gen.

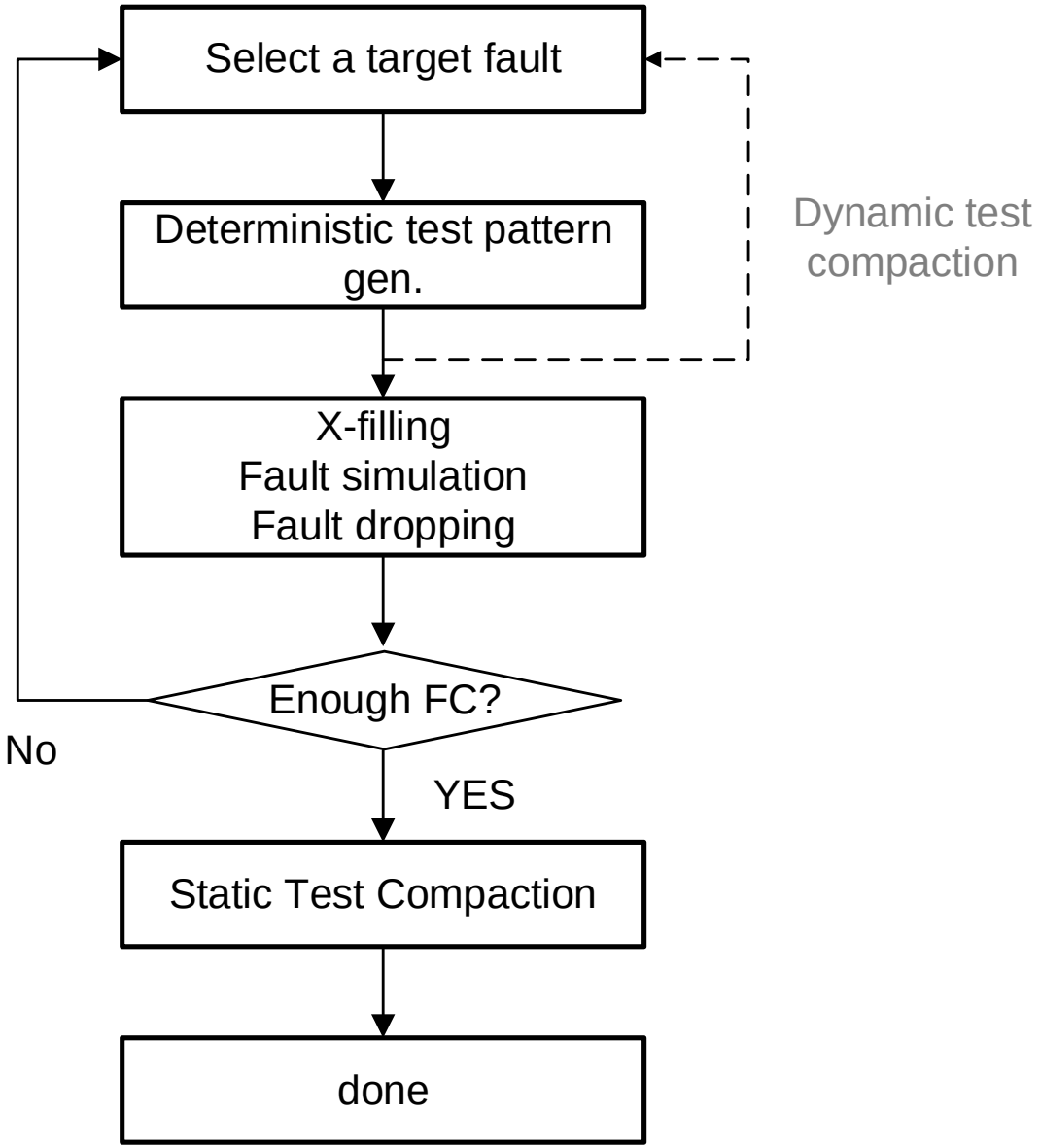


Random Test Pattern Generation

- **Idea: there are many easy-to-detect faults**
  - **First generate random patterns and**
  - **then select patterns that defect undetected faults**
- **Problem**
  - **Fault coverage often saturates after easy faults are detected**
    - ***Random pattern resistant faults*** **not easy to detect**

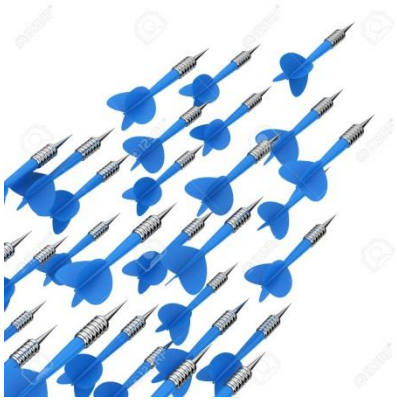


Deterministic Test Pattern Gen.



Summary

- **Introduction**
  - **Test pattern generation is NP-complete**
  - **Complete ATPG guarantees to find a solution if it exists**
  - **Random TPG: no target fault**
  - **Deterministic TPG: one target fault at a time**



Random TPG

Deterministic TPG