Logic Testing and Design for Testability

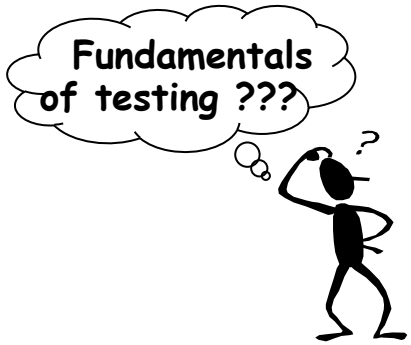
MIT Press, Sept. 1985

Hideo Fujiwara

The basis is necessary for development

- For a tree to grow larger, its root must grow bigger and deeper into the ground.
- Similarly, for test technologies (leaves) to develop, substantial results of the fundamental research (root) are necessary.
- The more enriched the fundamental research results become, the more enriched the practical research results become.

Fundamental problems of testing



What are the fundamentals of testing?

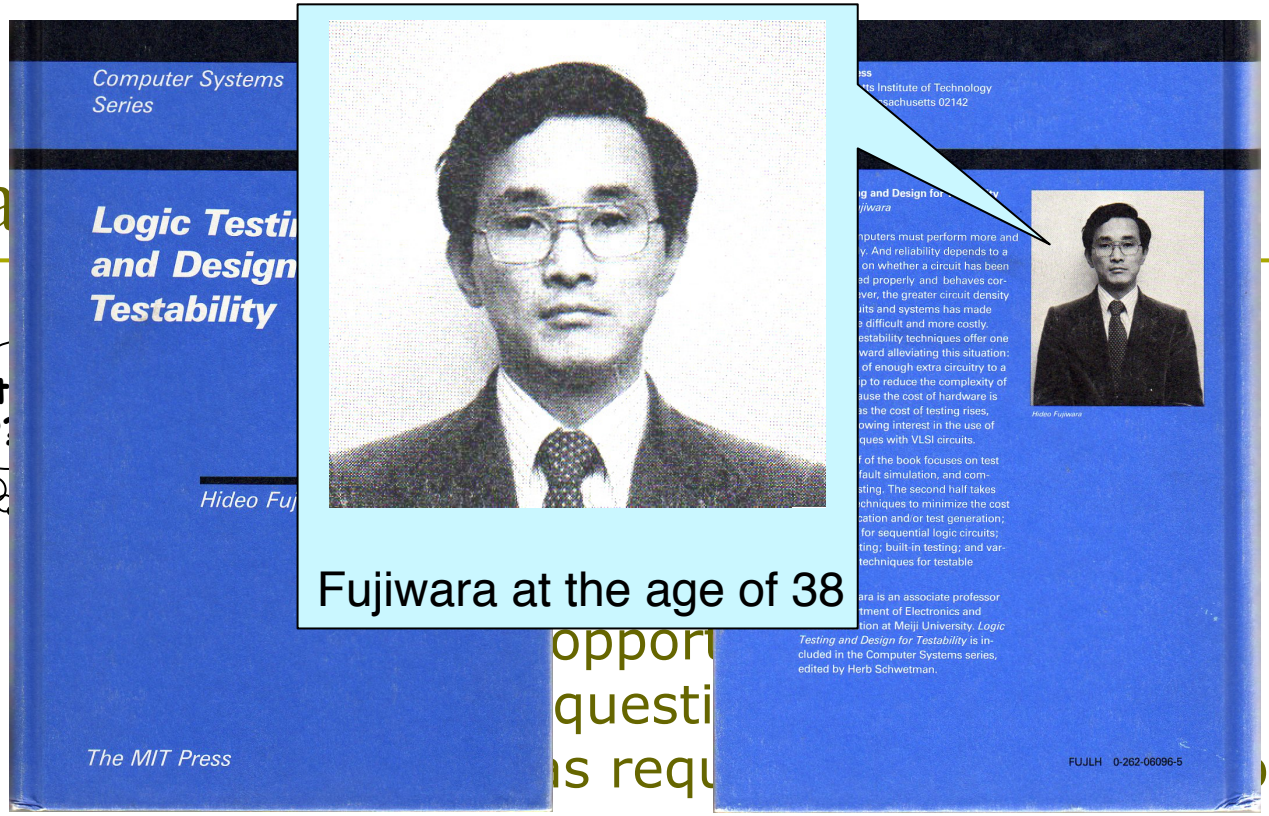
I had the opportunity to ask myself
the same question
when I was requested to write a book.

To educate the fundamentals of testing, I wrote a book.

Hideo Fujiwara, *Logic Testing and Design for Testability*,
The MIT Press, 1985

Funda

Fundament
of testing ?

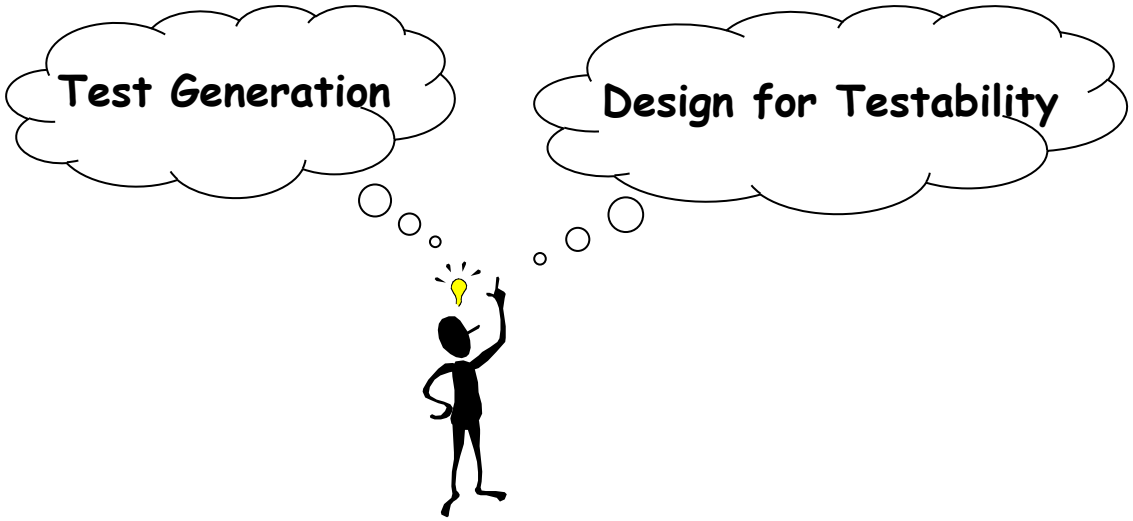


opport
questi
s requ
k.

To educate the fundamentals of testing, I wrote a book.

Hideo Fujiwara, *Logic Testing and Design for Testability*, The MIT Press, 1985

Fundamental problems of testing



- Test generation problem
  - ATPG (Automatic Test Program Generation)
- Design-for-test problem
  - DFT

H. Fujiwara, *Logic Testing and Design for Testability*, MIT Press, 1985

H. Fujiwara, *Logic Testing and Design for Testability*, MIT Press, 1985

Practical items

1.1 Logic Circuits
1.2 Fault Mod
1.3 Testing Pr
1.4 Testing Schem

Test Generation

2 Test Generation

2.1 Boolean Difference
2.2 The D-Algorithm
2.3 The PODE
2.4 The FAN A

Fault Simulation

2.5 Test Generatio for Sequential Circuits

3 Fault Simulation

3.1 Simulation Methodology
3.2 Parallel Fault Simulation
3.3 Deductive Fault Simulation
3.4 Concurrent Fault Simulation
3.5 Hardware Simulators

4 The Complexity of Testing

4.1 NP-Completeness
4.2 Polynomial Time Class
4.3 Closedness under Faults

II DESIGN FOR TESTABILITY

5 Introduction to Design for Testability

5.1 Testability
5.2 Minimization of Testing Cost
5.3 Combinational Logic versus Sequential Logic
5.4 Ad Hoc Design and Structured Design

6 Design to Minimize the Cost of Test Application

6.1 EOR Embedding
6.2 Minimally Testable Design
6.3 Dual-Mode Logic
6.4 Testing with Fixed Reference Values

7 Design to Minimize the Cost of Test Generation

7.1 Partitioning and Exhaustive Testing
7.2 Syndro
7.3 Reed-M
7.4 Programm ble Logic Arrays

Scan Design

8 Scan Design for Sequential Logic Circuits

8.1 State-Shiftable M
8.2 Scan Design Ap
8.3 Variations of Sc
8.4 Incomplete Scan D sign and Enhanced Scan Design

Built-in Self-Testing

9 Design for Built-in Self-Testing

9.1 Signature Analysis
9.2 Built-In Logic Block Observer
9.3 Self-Test with Scan Design
9.4 Self-Verification

H. Fujiwara, *Logic Testing and Design for Testability*, MIT Press, 1985

Theoretical items

1.1 Logic Circuits
1.2 Fault Modeling
1.3 Testi
1.4 Testi

Boolean Difference

2 Test Gen ation

2.1 Boolean Difference
2.2 The D-Algorithm
2.3 The PODEM Algorithm
2.4 The FAN Algorithm
2.5 Test Generation for Sequential Circuits

3 Fault Simulation

3.1 Simulation Methodology
3.2 Parallel Fault Simulation
3.3 Deduc
3.4 Concu
3.5 Hardware ulators

The Complexity of Testing

4 The Complexity of Testing

4.1 NP-Completeness
4.2 Polynomial Time Class
4.3 Closedness under Faults

II DESIGN FOR TESTABILITY

5 I

Design to Minimize the Cost of Test Application

5.1
5.2
5.3 Com ational Logic versus Sequential Logic
5.4 A Hoc Design and Structured Design

6 Design to Minimize the Cost of Test Application

6.1 EOR Embedding
6.2 Minimally Testable Design
6.3 Dual-Mode Logic
6.4 Testing with Fixed Reference Values

7 Design to Minimize the Cost of Test Generation

7.1 Partiti ing and Exhaustive Testing
7.2 Syndrome stable Design
7.3 Reed
rog

Design to Minimize the Cost of Test Generation

an I

8.1 State
8.2 Scan Design Approaches
8.3 Variations of Scan Design
8.4 Incomplete Scan Design and Enhanced Scan Design

9 Design for Built-in Self-Testing

9.1 Signature Analysis
9.2 Built-In Logic Block Observer
9.3 Self-Test with Scan Design
9.4 Self-Verification

Fundamental problems of testing

Practical synthesis problems

- Test generation algorithms.
  - Invent efficient algorithms to provide high fault efficiency.
- Design-for-testability methods.
  - Optimize the DFT under various constraints.

Theoretical analysis problems

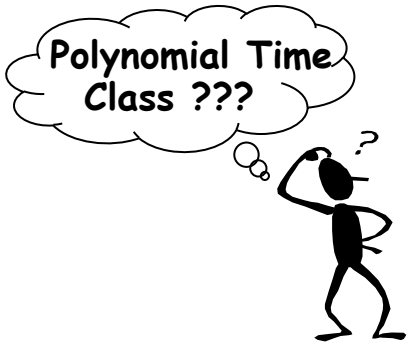
- Analysis of test generation complexity.
  - Clarify the complexity of test generation algorithms.
- Classification of sequential circuits.
  - Class of combinational test generation complexity.
  - Class of acyclic test generation complexity.

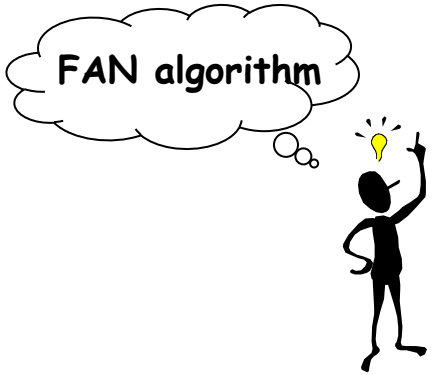
Theory is the mother of practice

Necessity is the mother of invention

Theoretical research / fundamental research are necessary for practical research.

Theoretical research ⇨ Practical ideas





Analysis of test generation complexity ⇨ Efficient ATPG

Classification of sequential circuits ⇨ Optimal design for testability

Complexity of test generation

NP-completeness [Fujiwara, MIT Press, 1985]

- A Boolean expression is *satisfiable* iff there exists some assignment of zeros and ones to the variables that gives the expression the value 1.
- *SAT (Satisfiability)*: Is a Boolean expression satisfiable?
- *Theorem 1* [Cook 1971]: SAT is NP-complete.
- U-SAT: Satisfiability problem for *unate* expression
- *Theorem 2* [Fujiwara 1982]: U-SAT is solvable in time O(L) where L is the length of an expression.

Complexity of test generation

NP-completeness [Fujiwara, MIT Press, 1985]

- *Fault detection* (FD): Is a given single stuck-at fault detectable?
  kM-FD: Fault detection problem for k-level monotone circuits
  kU-FD: Fault detection problem for k-level unate circuits

- *Theorem 3* [Fujiwara 1982]:
  3M-FD is NP-complete.
  Hence,
  3U-FD and FD are NP-complete.

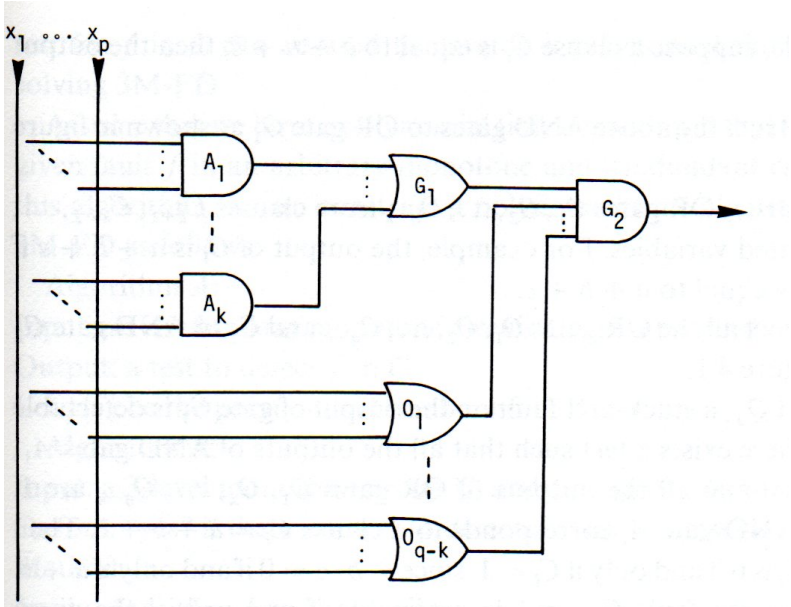


Figure 4.1
A 3-level monotone circuit Q_1

Complexity of test generation

SAT versus FD

[Fujiwara, MIT Press, 1985]

- Observation

SAT is NP-complete. ⟺ FD is NP-complete.

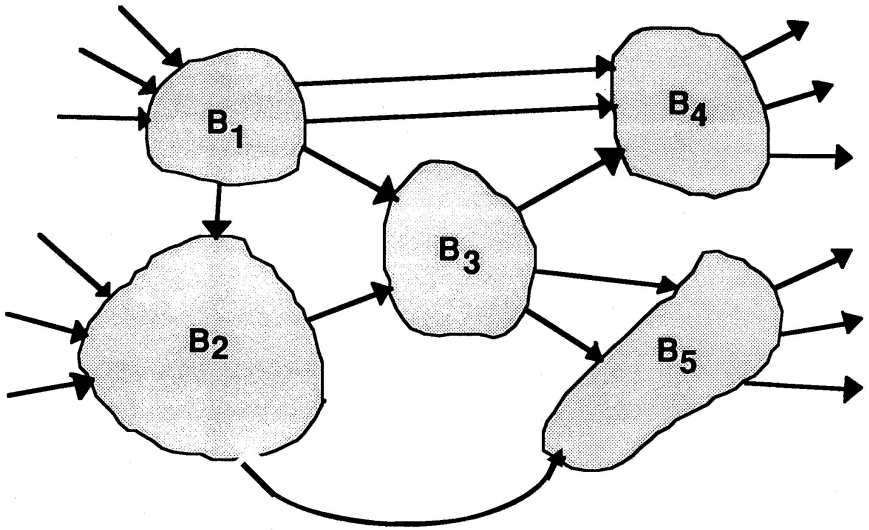
U-SAT is solvable in O(L). ⟺ U-FD is NP-complete.

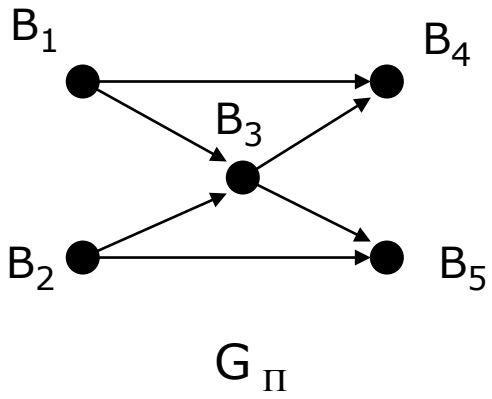
M-SAT is solvable in O(L). ⟺ M-FD is NP-complete.

Complexity of test generation

Polynomial time class [Fujiwara, MIT Press, 1985]

- A combinational circuit C is said to be *k-bounded* if there exists a partition $\Pi=\{B_1, B_2, \ldots, B_t\}$ such that
  - (1) the number of inputs of each block Bi is at most *k*, and
  - (2) graph G_Π has no cycle.





Complexity of test generation

k-bounded circuits [Fujiwara, MIT Press, 1985]

- *Theorem 4* [Fujiwara 1982]: Let C be a *k-bounded* circuit. Then there is an algorithm of time complexity $O(16^k m)$ to find a test for a single stuck-at fault in C, where m is the number of lines in C.

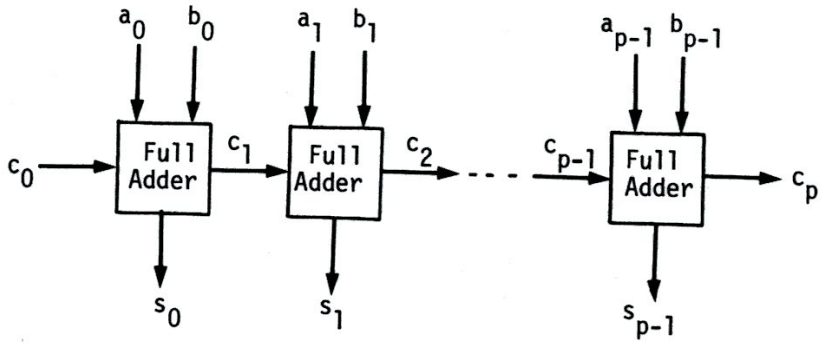


Figure 4.3
Ripple-carry adder

3-bounded circuit

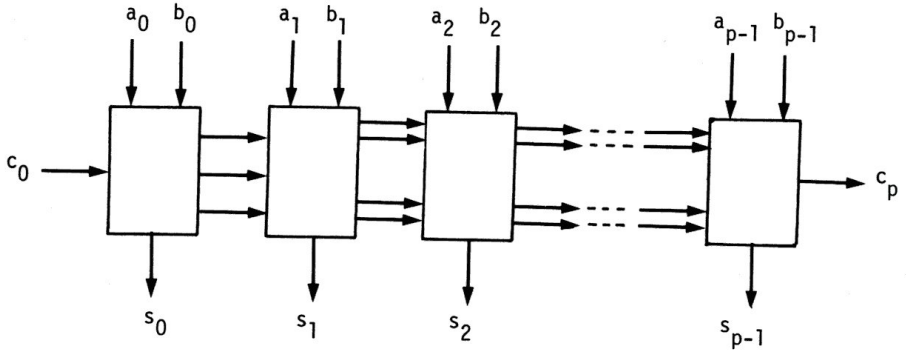


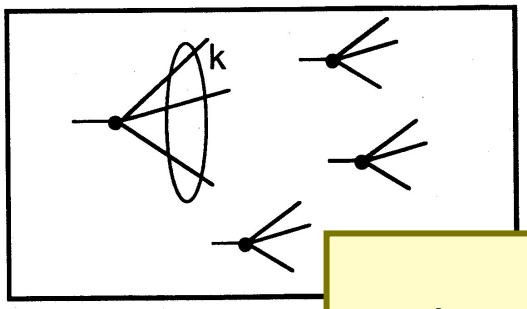
Figure 4.4
Gate-minimum *p*-bit adder

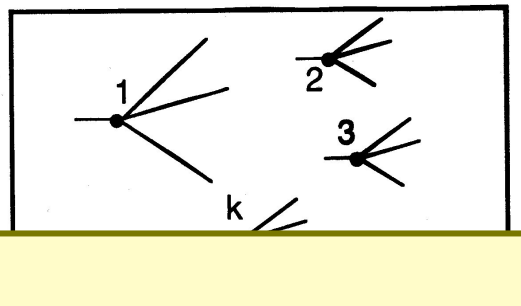
6-bounded circuit

Complexity of test generation

k-fanout-limited vs. k-fanout-point-bounded

[Fujiwara, MIT Press, 1985]





k-fanout-li

- k-FL-FD: Fault d
- k-FPB-FD: Fault d
  circuits

In the proof of this theorem, an algorithm is presented which requires the enumeration of at most 4^k combinations of values (0,1,D,D') on fanout-points.

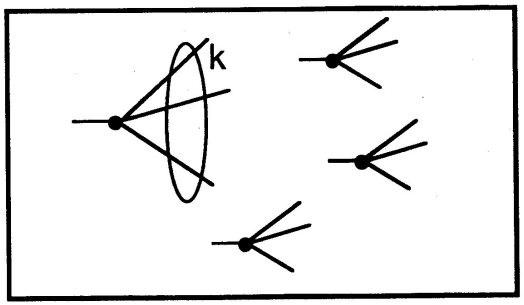
This algorithm is the origin of the FAN algorithm.

- *Theorem 5* [Fujiwara 1982
  k-FL-FD is NP-complete if
  k-FPB-FD is solvable in $O(4^k m)$ where m is the number of lines in C.

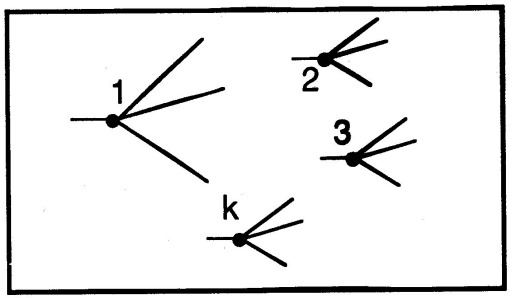
Complexity of test generation

k-fanout-limited vs. k-fanout-point-bounded

[Fujiwara, MIT Press, 1985]



k-fanout-limited



k-fanout-point-bounded

- Observation:

  k-FL-FD is NP-complete even if k is a constant.

  k-FPB-FD is solvable in O(m) if k is a constant.

- The complexity of test generation is affected

  not by the number of fanout branches from a fanout point

  but by the number of fanout points.

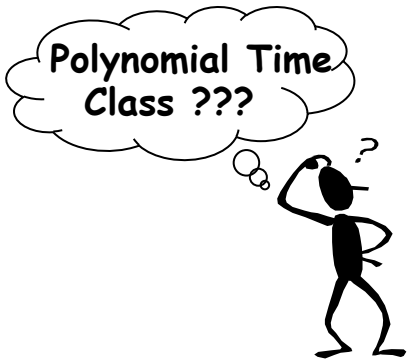
Complexity of test generation

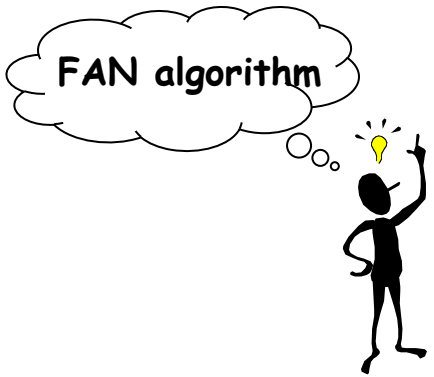
Theory is the mother of practice

[Fujiwara, MIT Press, 1985]

The algorithm shown in the proof of the theorems is the origin of the FAN algorithm.

Analysis of test generation complexity ⇨ Efficient ATPG





Heuristics of the FAN algorithm

[Fujiwara, MIT Press, 1985]

- Strategy 1: In each step of the algorithm, determine as many signal values as possible that can be *uniquely implied*.

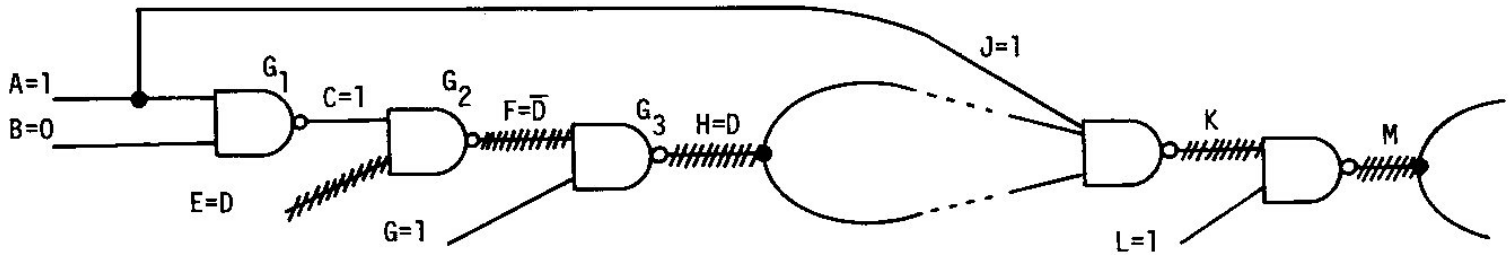
Heuristics of the FAN algorithm

[Fujiwara, MIT Press, 1985]

- Strategy 1: In each step of the algorithm, determine as many signal values as possible that can be *uniquely implied*.
- Strategy 2: Assign a fault signal D or D' that is *uniquely determined* or implied by the fault in question.

H

Unique assignment can reduce backtracks

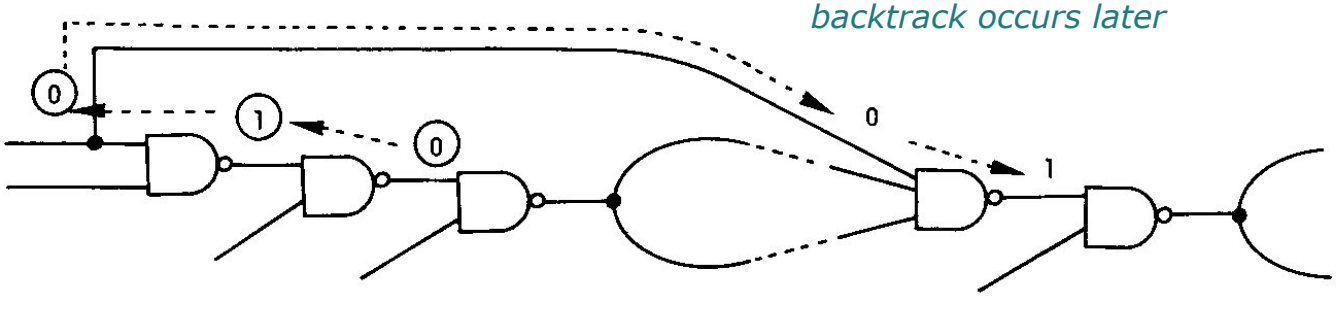


(a) Unique sensitization and implication

implied by the fault in question.

- Strategy 3: When the D-frontier consists of a single gate, apply a *unique sensitization.*

D-frontier becomes empty and backtrack occurs later

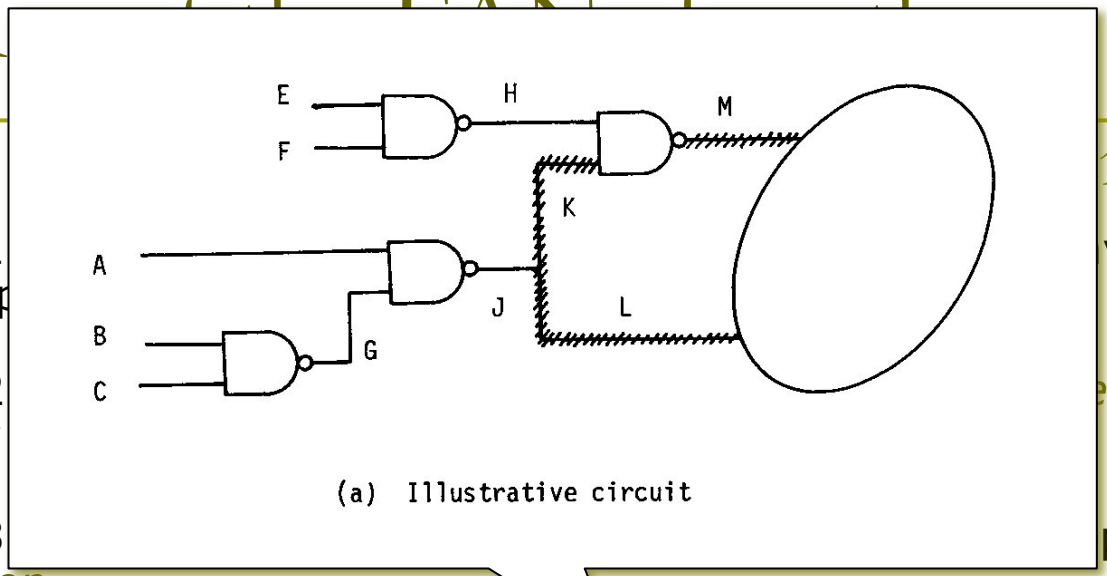


(b) PODEM

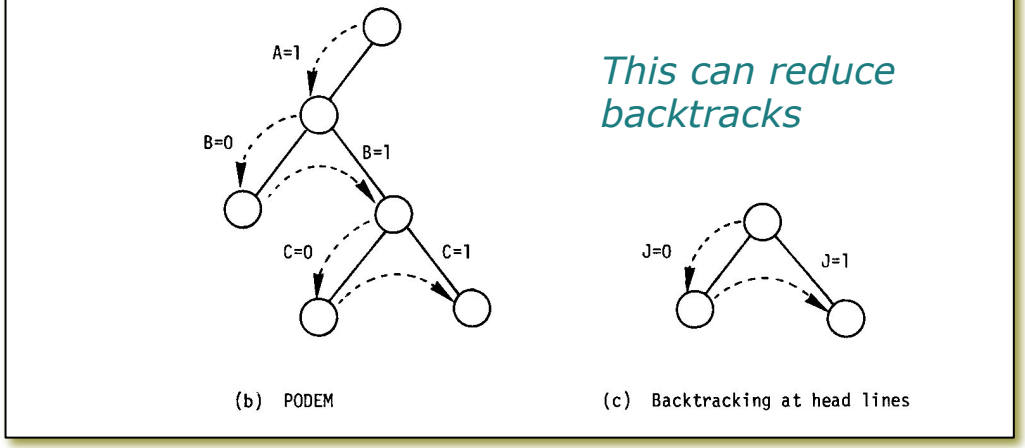
Heuristi

1985]

- Strategy 1 … y signal values as p…
- Strategy 2 … *ermined* or implied by …
- Strategy 3 … ply a *unique sensitization.*
- Strategy 4: Stop the backtrace at a *head line*, and postpone the line justification for the head line to later.



(a) Illustrative circuit

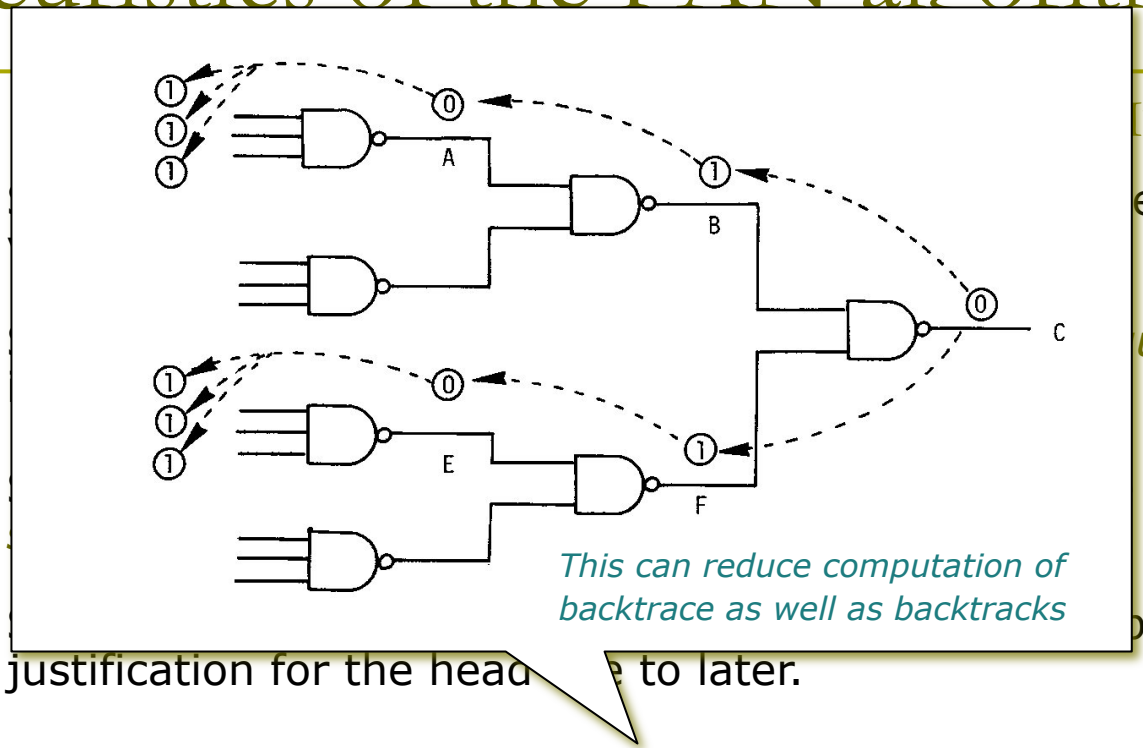


(b) PODEM

(c) Backtracking at head lines

This can reduce backtracks

Heuristics of the FAN algorithm



This can reduce computation of backtrace as well as backtracks

[T Press, 1985]

- e as many signal
- *uely determined* or
- gate, apply a *unique*
- ostpone the line justification for the head e to later.
- Strategy 5: *Multiple backtracing* (concurrent backtracing of more than one path) is more efficient than backtracing along a single path.

Heuristics of the FAN algorithm

[Fujiwara, MIT Press, 1985]

- Strategy 1: In each step of the algorithm, determine as many signal values as possible that can be *uniquely implied*.
- St
  in
- St
  se
- St
  ju
- St
  pa

PODEM assigns a binary value only to primary inputs.
So, backtracks occur at primary inputs.

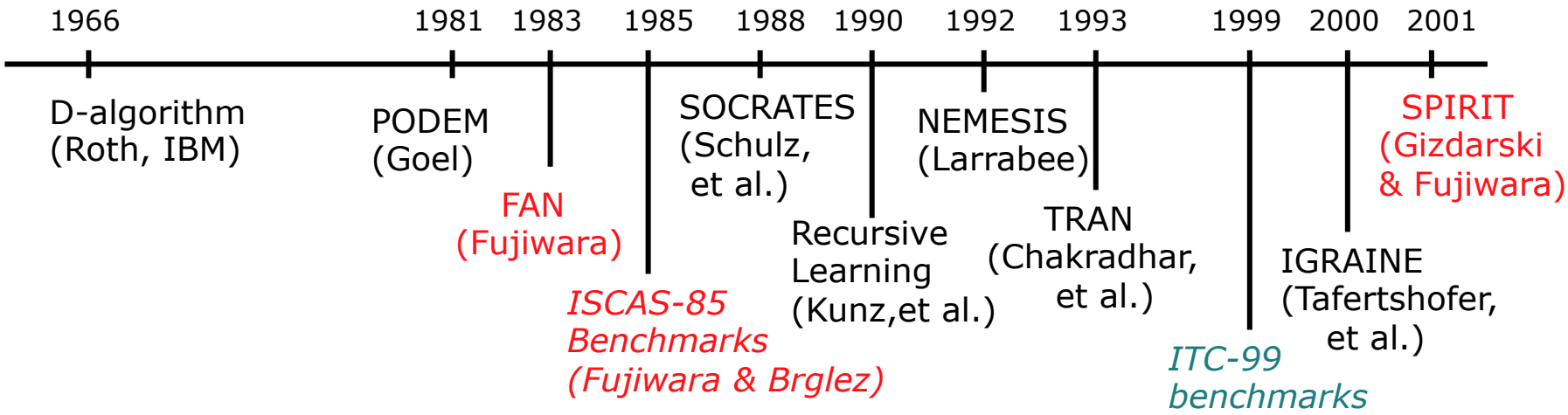
FAN assigns a binary value only to head lines and fanout points.
So, backtracks occur at headlines and fanout points.

This is effective to reduce the number of backtracks.

- Strategy 6: In the multiple backtrace, if an objective at a fanout point p has a contradictory requirement, stop the backtrace so as to *assign a binary value to the fanout point.*

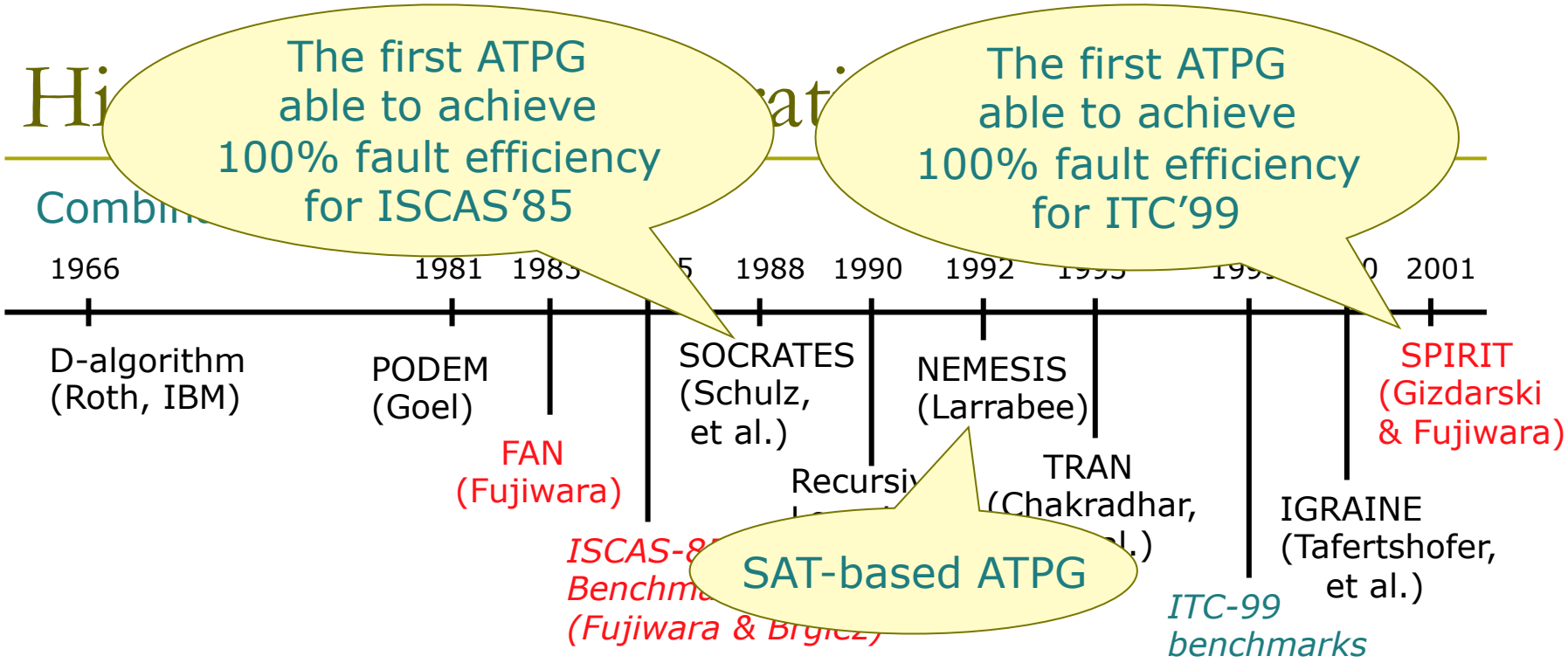
History of test generation algorithms

Combinational ATPG

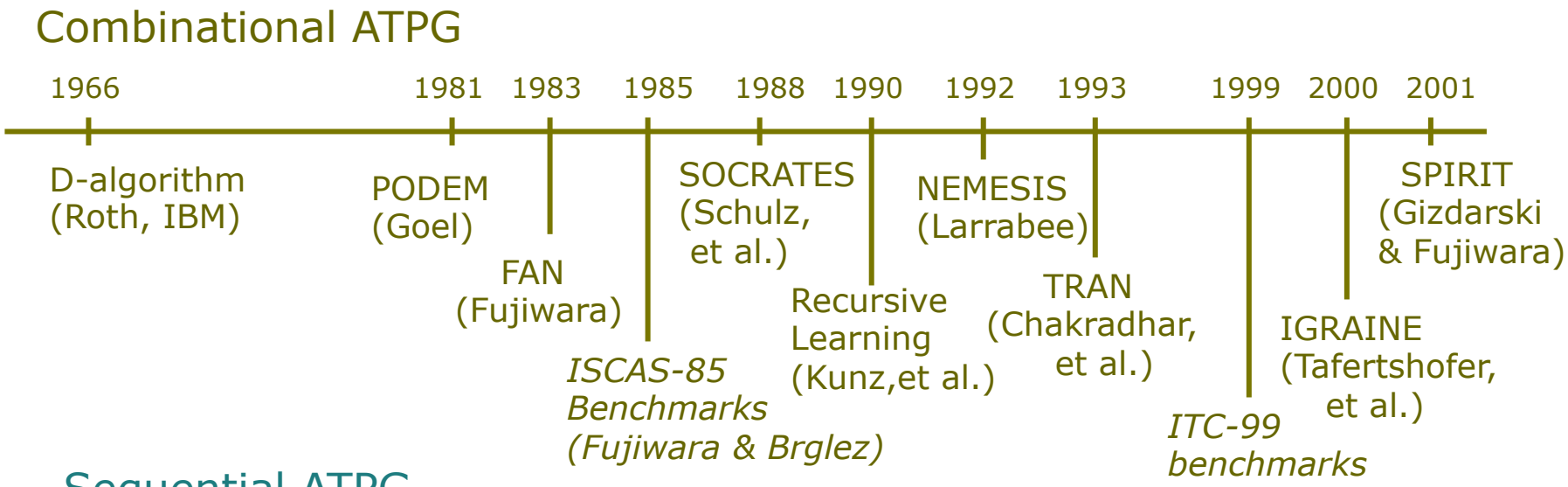


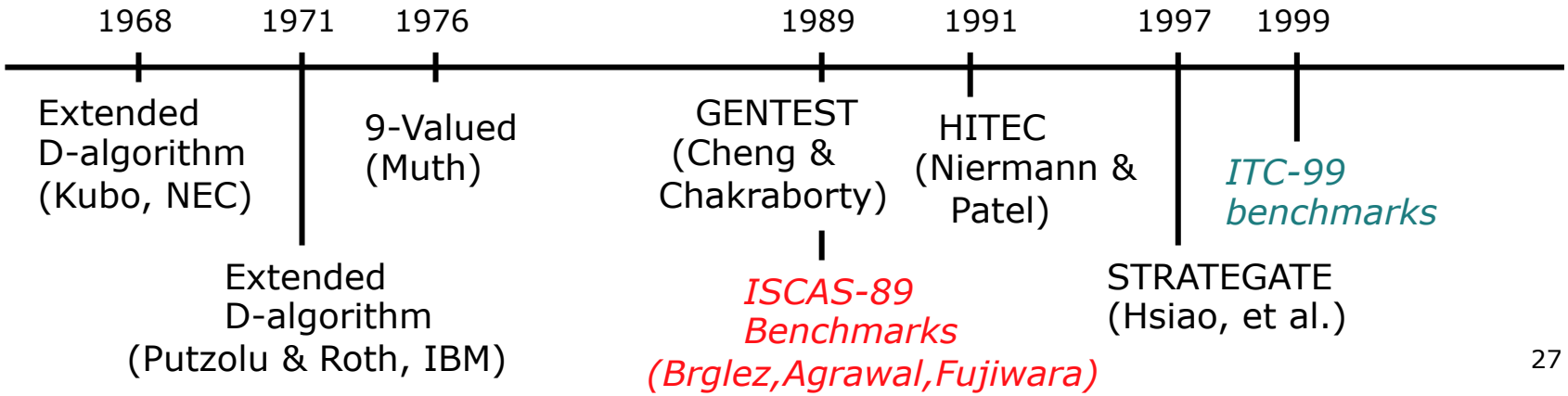
Hi…at…

Combin…


The first ATPG able to achieve 100% fault efficiency for ISCAS'85
The first ATPG able to achieve 100% fault efficiency for ITC'99
1966
1981
1983
1988
1990
1992
2001
D-algorithm (Roth, IBM)
PODEM (Goel)
FAN (Fujiwara)
SOCRATES (Schulz, et al.)
NEMESIS (Larrabee)
TRAN (Chakradhar,
IGRAINE (Tafertshofer, et al.)
SPIRIT (Gizdarski & Fujiwara)
ITC-99 benchmarks
SAT-based ATPG

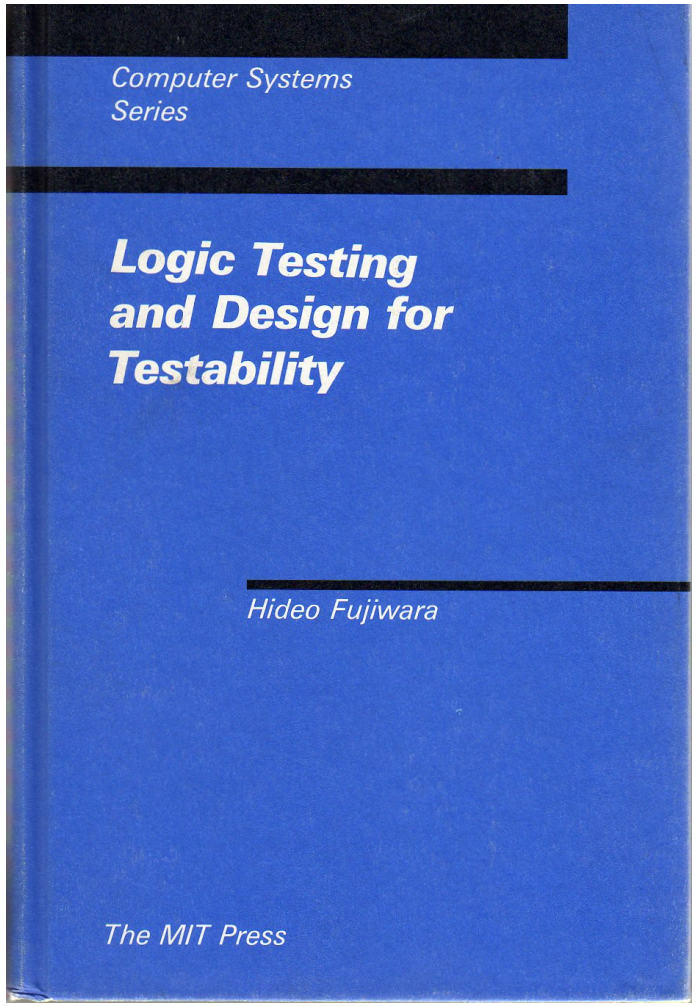
History of test generation algorithms





Hideo Fujiwara, Logic Testing and Design for Testability, The MIT Press, 1985

https://mitpress.mit.edu/index.php?q=books/logic-testing-and-design-testability



Hardback

Paperback

Thank you