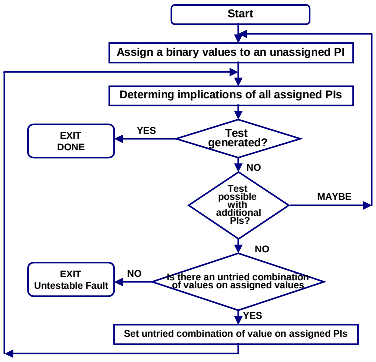
PODEM Algorithm

- IBM introduced semiconductor DRAM memory into its mainframes – late 1970's
- Memory had error correction and translation circuits – improved reliability
  - D-ALG unable to test these circuits
    - Search too undirected
    - Large XOR-gate trees
    - Must set all external inputs to define output
  - Needed a better ATPG tool

PODEM

- PODEM: Path-Oriented DEcision Making (Goel 1981)
- Like D Algorithm, PODEM is circuit-based, fault-oriented ATPG algorithm
- Signal values are explicitly assigned at the primary inputs only; other values are computed by implication
- *Justification* is not needed
- *Backtracking* means reassigning primary inputs when a contradiction occurs; "implicit enumeration" technique
- A simple "*backtrace*" heuristic is used to select the next primary input line and the value to assign to it
- **New concepts introduced:**
  - **Expand binary decision tree only around primary inputs**
  - **Use *X-PATH-CHECK* to test whether *D-frontier* still there**
  - ***Objectives* -- bring ATPG closer to propagating D (D') to PO**
  - ***Backtracing***

PODEM Flow Chart



Step 1. Assign binary value to unassigned PI

Step 2: Determine implications of all assigned PIs

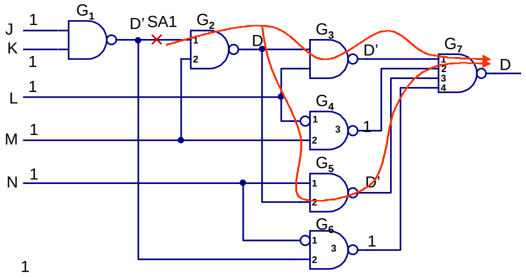
Test Generated? If so, done.

Test possible with more assigned PIs? If maybe, go to Step 1

Is there untried combination of values on assigned PIs? If not, exit: untestable fault

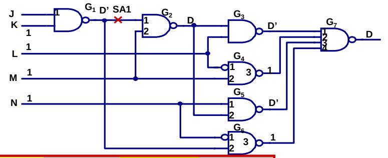
Set untried combination of values on assigned PIs using objectives and backtrace.

Then, go to Step 2



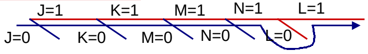
- Find the Test vector that tests SA1 fault at G1.
- D-Drive: G2 && {G3 || G5} && G7.

PODEM Example

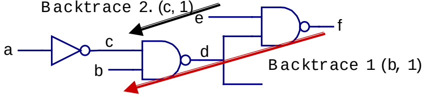


Iteration	Objective	Backtrace	Implication	D-Front
1	$G_2.1 = 0$	J=1		
2	$G_2.1 = 0$	K=1	$G_6.3 = 1$	$\{G_2\}$
3	$G_2.2 = 1$	M=1	$G_2.3 = D$	$\{G_3, G_5\}$
4	$G_5.1=1$	N=1	$G_5.3 = D'$	$\{G_3, G_7\}$
5a	$G_7.1=1$	L=0	$G_7.5 = 1$	$\{G_3, G_7\}$
5b	Retry	L=1	$G_7.5 = D$	None

Test: (JKMNL)=(11111)



How PODEM Works



1. Initial objective (f = 1)

2. Backtrace 1 (b, 1)

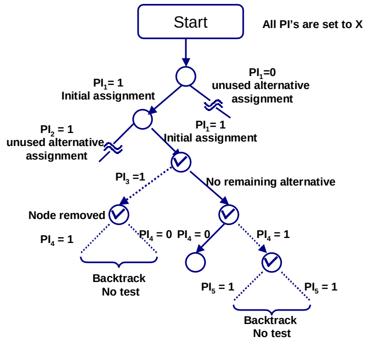
3. PI assignment (b =1)

4. Imply

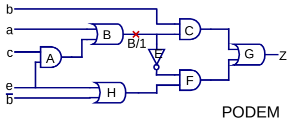
5. Backtrace 2 (c, 1)

6. PI assignment (a = 0)

7. Imply (d = 0)



Comparison Between D-ALG & PODEM



D-Algorithm

	Decision	Implication	Comment
1	B=D'	E=D', A=0, a=0	Activate fault
2	b=1	C=D'	Propagate via G
3	F=0	Z=D'	End of D-Drive
4	H=0	e=0	Jusitify F
5	c=0		

Test = (a, b, c, e)= (0100)
Order of PI assignment: a, b, e, c.

PODEM

	Decision	Implication	Comment
			Initial objective (B, 0)
1	a=0		Backtrace, Imply
2	c=0	A=0, B=D', E=D	Backtrace, Imply
3	b=1	b'=0, C=D'	Objective (b,1), Imply
4	e=0	H=0, F=0, Z=D'	Objective (F,0); Imply; Success

Test = (a, b, c, e)= (0100)
Order of PI assignment: a, c, b, e.

D-ALG v.s. PODEM

D-Algorithm

- **Emphasis on D-propagation**
- **Too much backtracking (undoing a previous decision and making a different choice) in some circuits. All possible choices may have to be tried in the worst case**
- **Some faults require multiple path sensitization**

PODEM

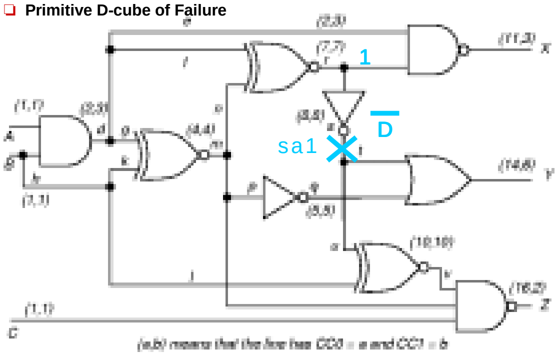
- **Less backtracking**
- **Sets a subset of inputs one by one**
- **Propagates D after initial objective is reached**

Comparison of speed between PODEM & D-Algorithm

	# of Cells	Run Time		Fault Coverage	
		PODEM	D-ALG	PODEM	D-ALG
ckt #1	828	1	34.5	100.0	99.7
ckt #2	935	1	12.8	100.0	93.1
ckt #3	951	1	2.2	99.5	99.5
ckt #4	1,566	1	3.1	97.4	97.4
ckt #5	1042	1	3.2	96.6	96.6

Example – Fault *s* sa1

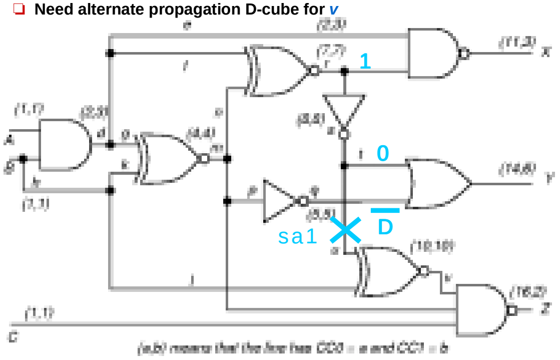
- **Primitive D-cube of Failure**



(a,b) means that the line has CC0 = a and CC1 = b

Example 7.3 – Backtrack

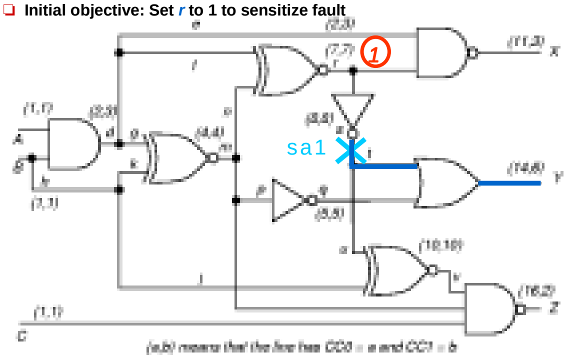
- **Need alternate propagation D-cube for *v***



(a,b) means that the line has CC0 = a and CC1 = b

Example 7.3 -- Step 2 *s* sa1

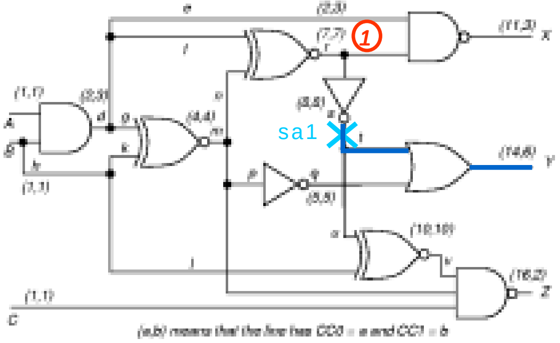
- **Initial objective: Set *r* to 1 to sensitize fault**



(a,b) means that the line has CC0 = a and CC1 = b

Example 7.3 -- Step 3 *s* sa1

- Backtrace from *r*



(a,b) means that the line has CC0 = a and CC1 = b

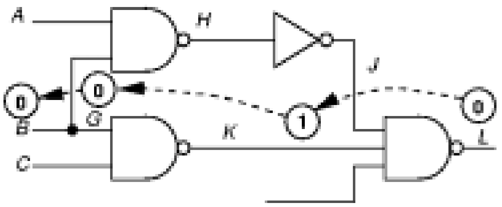
FAN: Fan-out Oriented ATPG Algorithm

- FAN: Fan-out Oriented ATPG Algorithm (Fujiwara & Shimono, 1983)
- Representative of PODEM-based ATPG algorithms that add various heuristic speed-up features
- FAN drops or alters some of basic features of PODEM:
  - it halts backtracking at certain internal lines
  - it tries to satisfy multiple objectives at once (multiple backtrace)
  - it allows backward as well as forward implications
  - it makes quick and easy assignments directly

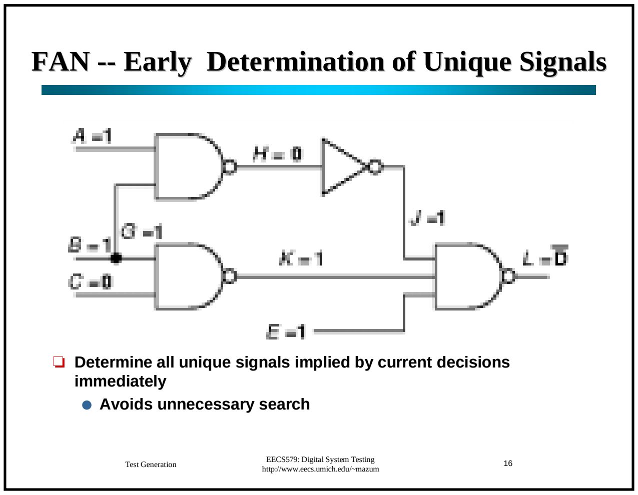
FAN -- Fujiwara and Shimono (1983)

- **New concepts:**
  - **Immediate assignment of *uniquely-determined signals***
  - ***Unique sensitization***
  - **Stop Backtrace at *head lines***
  - ***Multiple Backtrace***

PODEM Fails to Determine Unique Signals

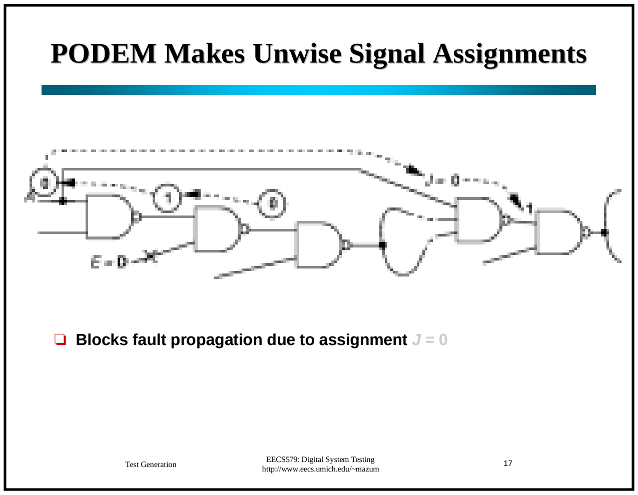


- **Backtracing operation fails to set all 3 inputs of gate *L* to 1**
  - **Causes unnecessary search**



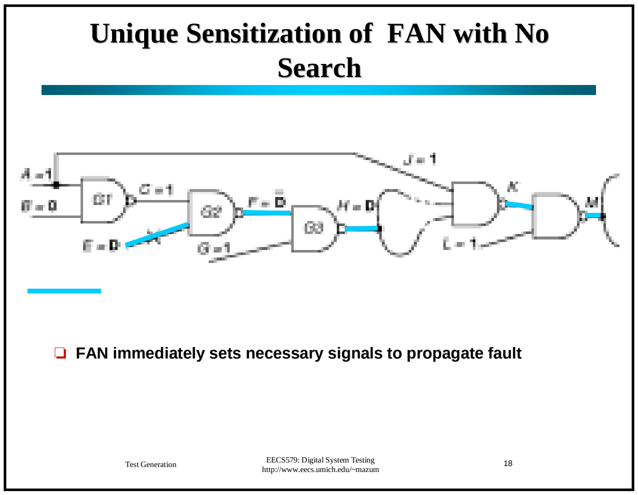
FAN -- Early Determination of Unique Signals

- Determine all unique signals implied by current decisions immediately
  - Avoids unnecessary search



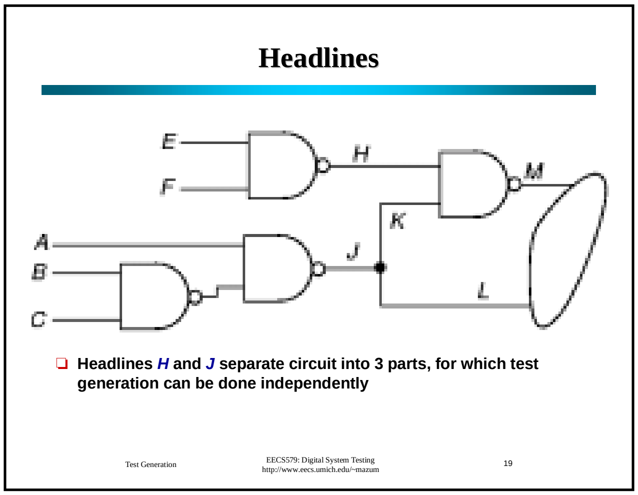
PODEM Makes Unwise Signal Assignments

- Blocks fault propagation due to assignment J = 0



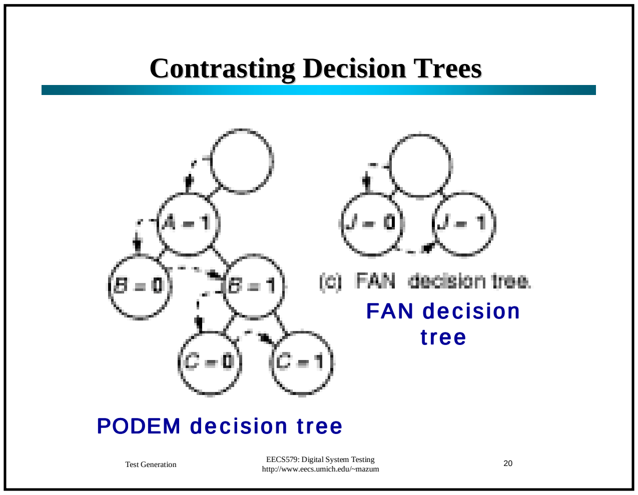
Unique Sensitization of FAN with No Search

- FAN immediately sets necessary signals to propagate fault



Headlines

- Headlines *H* and *J* separate circuit into 3 parts, for which test generation can be done independently



Contrasting Decision Trees

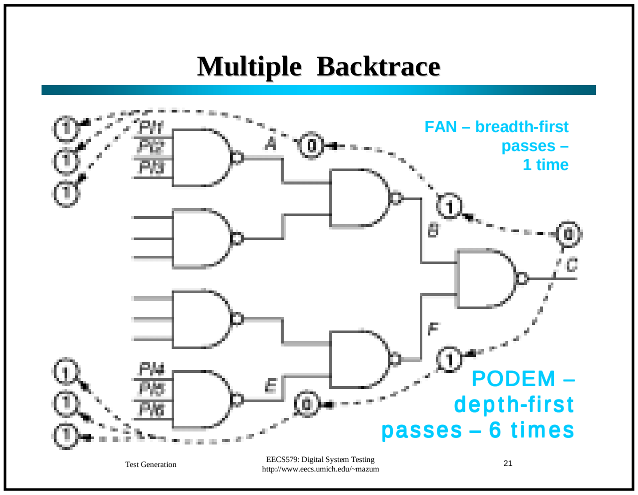
FAN decision tree

PODEM decision tree

Test Generation

EECS579: Digital System Testing
http://www.eecs.umich.edu/~mazum

20



Multiple Backtrace

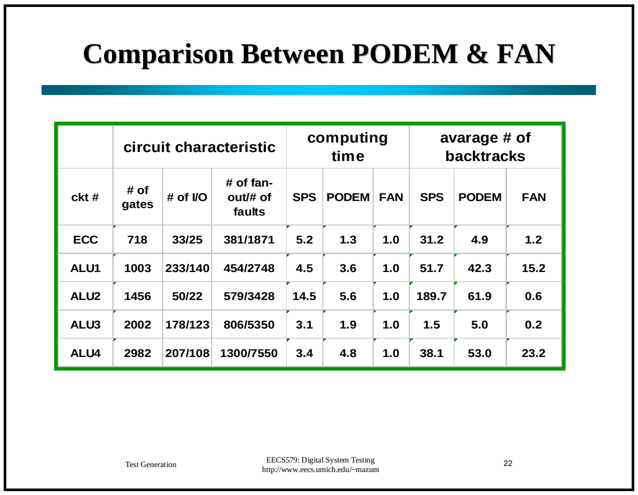
FAN – breadth-first passes – 1 time

PODEM – depth-first passes – 6 times

Test Generation

EECS579: Digital System Testing
http://www.eecs.umich.edu/~mazum

21



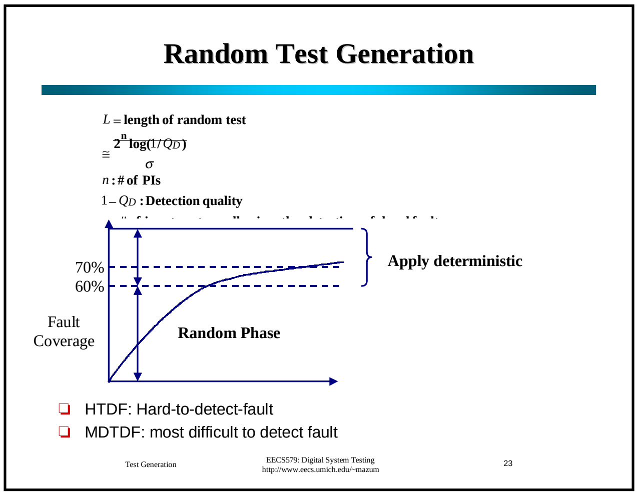
Comparison Between PODEM & FAN

	circuit characteristic			computing time			avarage # of backtracks		
ckt #	# of gates	# of I/O	# of fan-out/# of faults	SPS	PODEM	FAN	SPS	PODEM	FAN
ECC	718	33/25	381/1871	5.2	1.3	1.0	31.2	4.9	1.2
ALU1	1003	233/140	454/2748	4.5	3.6	1.0	51.7	42.3	15.2
ALU2	1456	50/22	579/3428	14.5	5.6	1.0	189.7	61.9	0.6
ALU3	2002	178/123	806/5350	3.1	1.9	1.0	1.5	5.0	0.2
ALU4	2982	207/108	1300/7550	3.4	4.8	1.0	38.1	53.0	23.2

Test Generation

EECS579: Digital System Testing
http://www.eecs.umich.edu/~mazum

22



Random Test Generation

L = **length of random test**

$$\cong \frac{2^n \log(1/Q_D)}{\sigma}$$

n : **# of PIs**

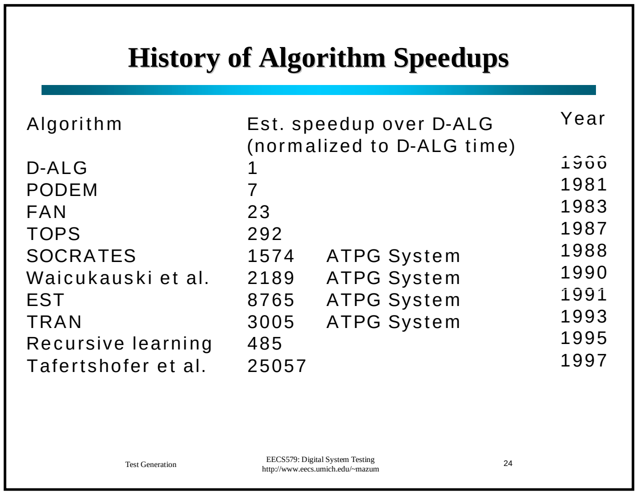
$1 - Q_D$: **Detection quality**

- HTDF: Hard-to-detect-fault
- MDTDF: most difficult to detect fault

Test Generation

EECS579: Digital System Testing
http://www.eecs.umich.edu/~mazum

23



History of Algorithm Speedups

Algorithm	Est. speedup over D-ALG (normalized to D-ALG time)		Year
D-ALG	1		1966
PODEM	7		1981
FAN	23		1983
TOPS	292		1987
SOCRATES	1574	ATPG System	1988
Waicukauski et al.	2189	ATPG System	1990
EST	8765	ATPG System	1991
TRAN	3005	ATPG System	1993
Recursive learning	485		1995
Tafertshofer et al.	25057		1997