EEDG/CE 6303: Testing and Testable Design

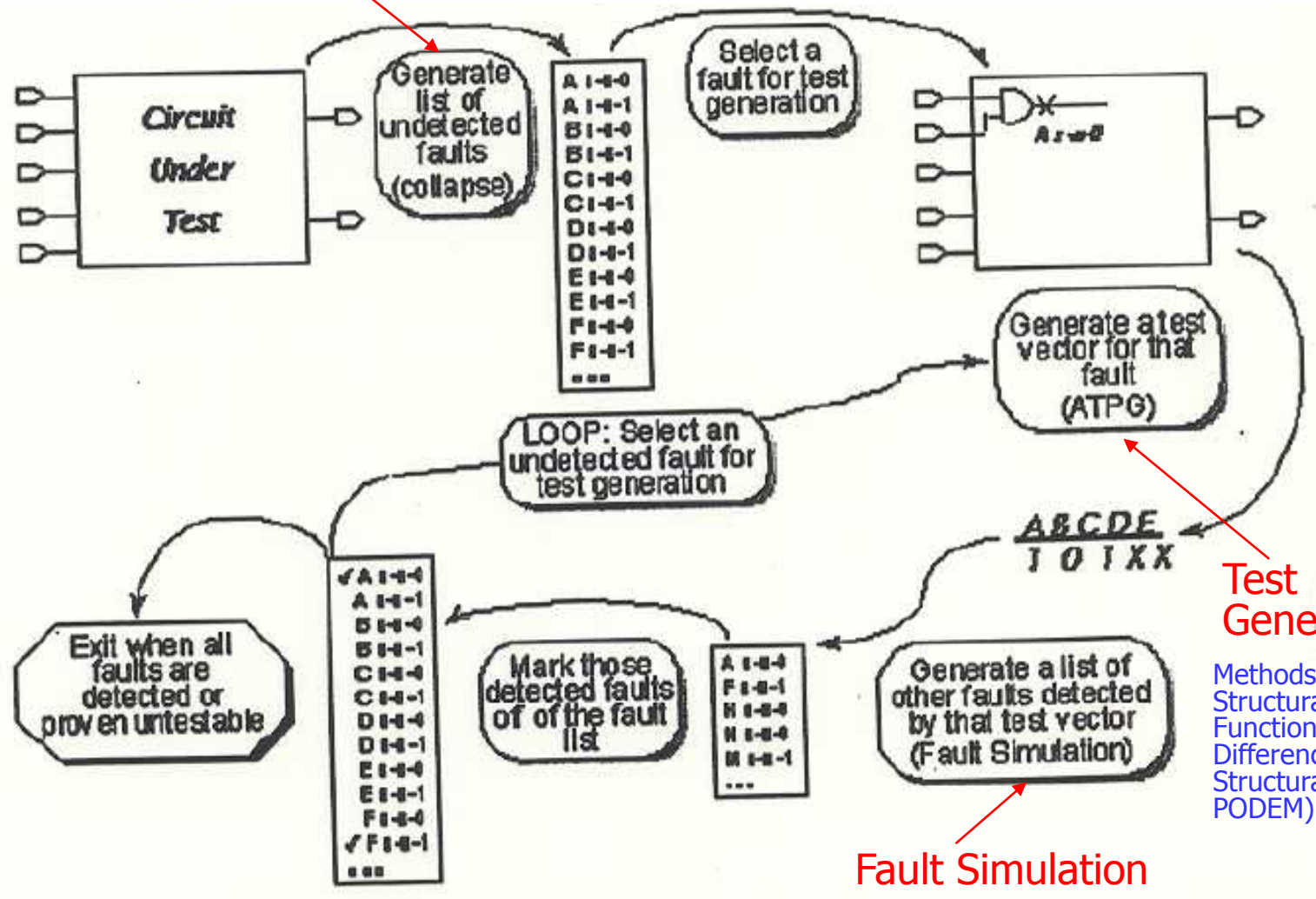
Mehrdad Nourani

Dept. of ECE
Univ. of Texas at Dallas

Session 03

Test Generation for Combinational Circuits

Fault Analysis System (Review)



Methods: 1. Non-Structural (Direct Functions, Booolean Difference, CNF), 2. Structural (D-Algo., PODEM)

Test Generation Techniques

- There are two main test vector generation techniques
  1. Non-Structural (Analytical)
     - Analyzes the gate-level description of a circuit and implicitly enumerate all possible input combinations to find a test vector for a target fault.
     - Methods: (i) Direct Function, (ii) Boolean Difference, (iii) CNF (product-of-sum form), …
  2. Structural
     - Analyzes the structure of a given circuit to generate a test vector for a given target fault, or declare it untestable.
     - Methods: (i) D-Algorithm, (ii) PODEM, …

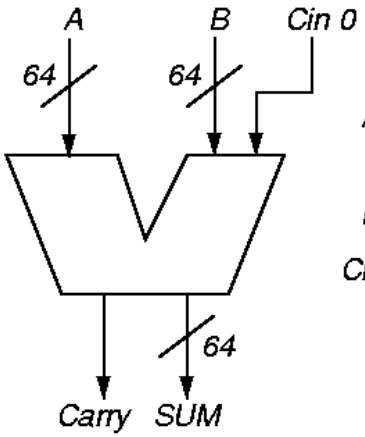
Structural Test Generation

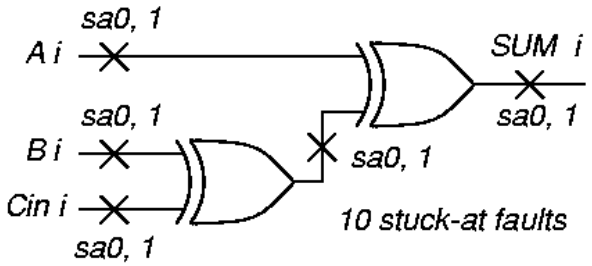
Structural Test Generation

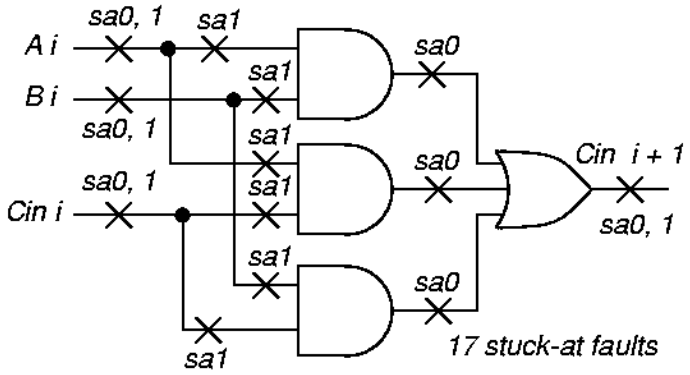
- **Analyzes the structure** of a given circuit to generate a test vector for a given target fault, or declare it untestable.
  - Methods: (i) **D-Algorithm**, (ii) PODEM, …

Motivation

- Consider a 64-bit ripple-carry adder







Motivation (cont.)

- Functional (exhaustive) ATPG – generate complete set of tests for circuit input-output combinations
  - 129 inputs, 65 outputs:
  - 2^{129} = 680,564,733,841,876,926,926,749,214,863,536,422,912 patterns
  - Using 1 GHz ATE, would take 2.15×10^{22} years
- Structural test:
  - No redundant adder hardware, 64 bit slices
  - Each with 27 faults (using fault equivalence)
  - At most 64 x 27 = 1728 faults (tests)
  - Takes 0.000001728 s on 1 GHz ATE
- Designer gives small set of functional tests – augment with structural tests to boost coverage to 98^{+} %

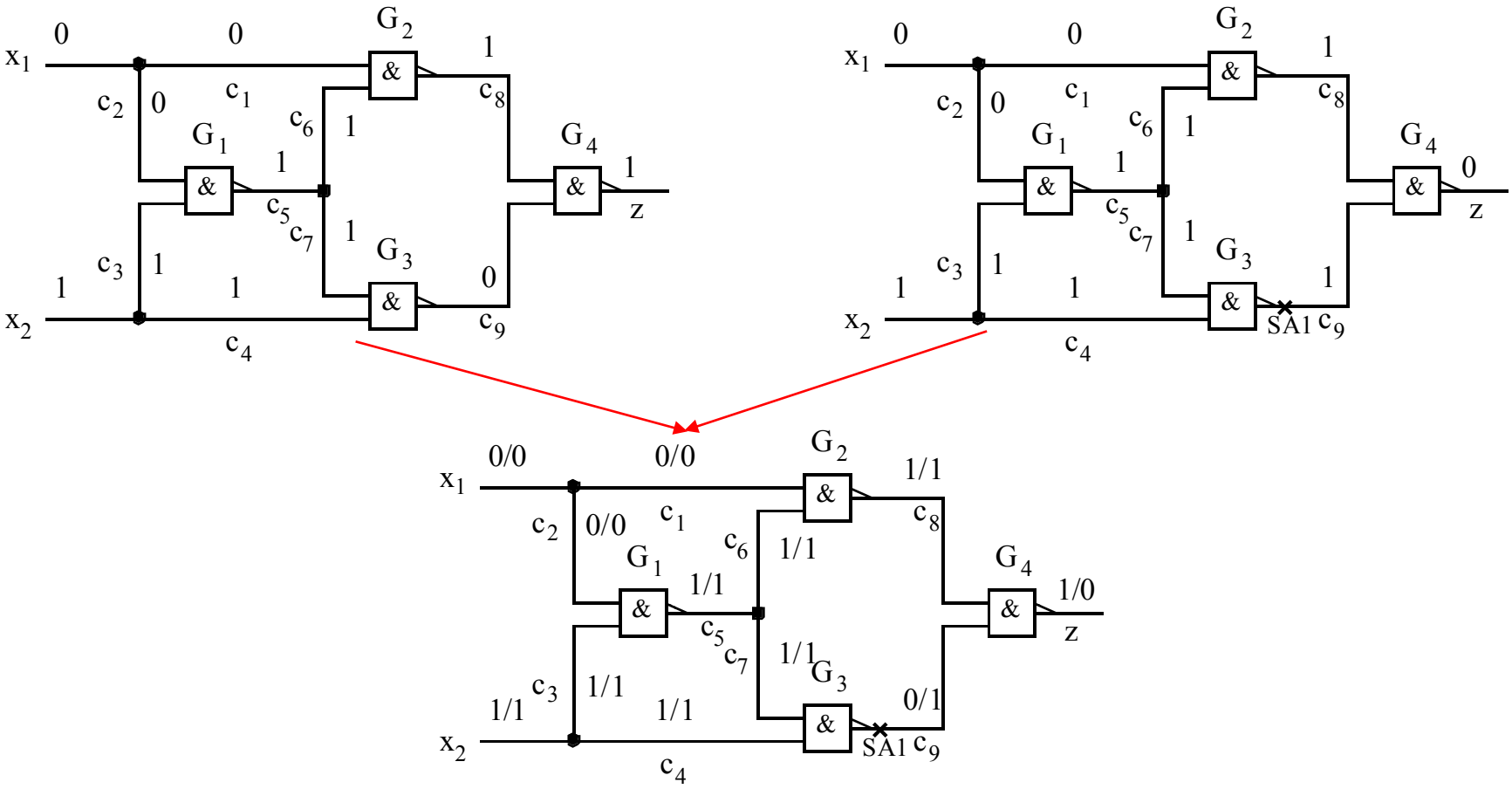
Algebra for Structural Test

- Represent two machines, which are simulated simultaneously by a computer program:
  - Good circuit machine (1st value)
  - Bad circuit machine (2nd value)

- Better to represent both in the algebra:
  - Need only 1 pass of ATPG to solve both
  - Good machine values that preclude bad machine values become obvious sooner & vice versa

Circuit Representation

- It's more efficient to carry (simulate) fault-free and faulty values at the same time.



Algebra for Structural Test (cont.)

- Roth's 5-Valued and Muth's 9-Valued Algebra

Symbol	Meaning	Fault-Free (Good)	Faulty (Bad)
0	0/0	0	0
1	1/1	1	1
D	1/0	1	0
$\overline{D}$	0/1	0	1
X	X/X	X	X
G0	0/X	0	X
G1	1/X	1	X
F0	X/0	X	0
F1	X/1	X	1

5-Valued

9-Valued

Operations in Multi-Valued Logic

- Examples (2-input AND and OR gates)

AND	0	1	D	$\overline{D}$	x
0	0	0	0	0	0
1	0	1	D	$\overline{D}$	x
D	0	D	D	0	x
$\overline{D}$	0	$\overline{D}$	0	$\overline{D}$	x
x	0	x	x	x	x

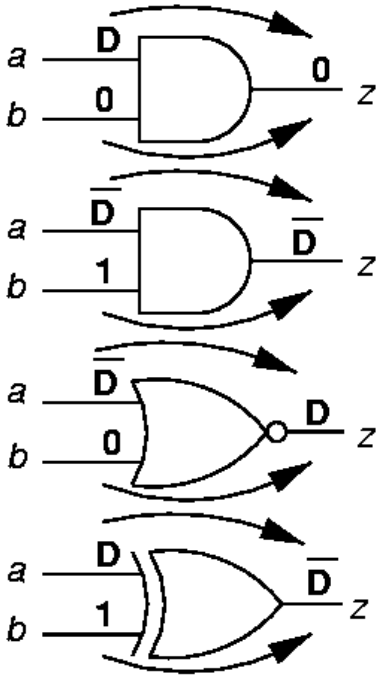
OR	0	1	D	$\overline{D}$	x
0	0	1	D	$\overline{D}$	x
1	1	1	1	1	1
D	D	1	D	1	x
$\overline{D}$	$\overline{D}$	1	1	$\overline{D}$	x
x	x	1	x	x	x

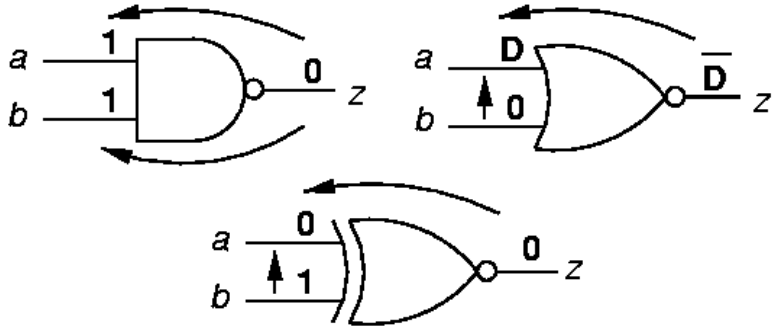
- e.g. D+0=1/0+0/0=1/0=D

Path Sensitization

- Three key tasks are needed
  1. Fault Sensitization (Stimulation)
  2. Fault (Effect) Propagation
  3. Line Justification
- Forward and Backward Implications

Examples of Forward Implications

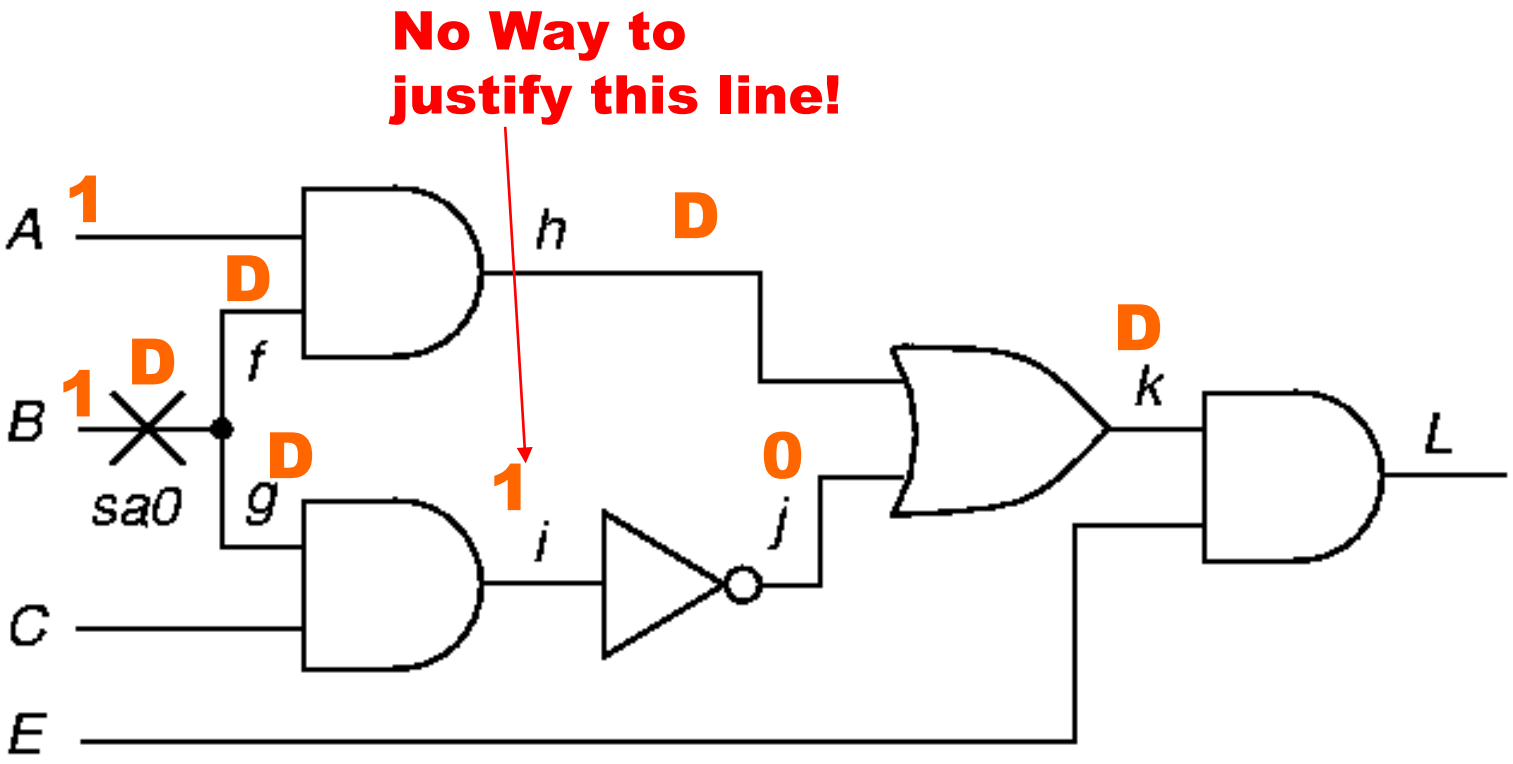




Examples of Backward Implications

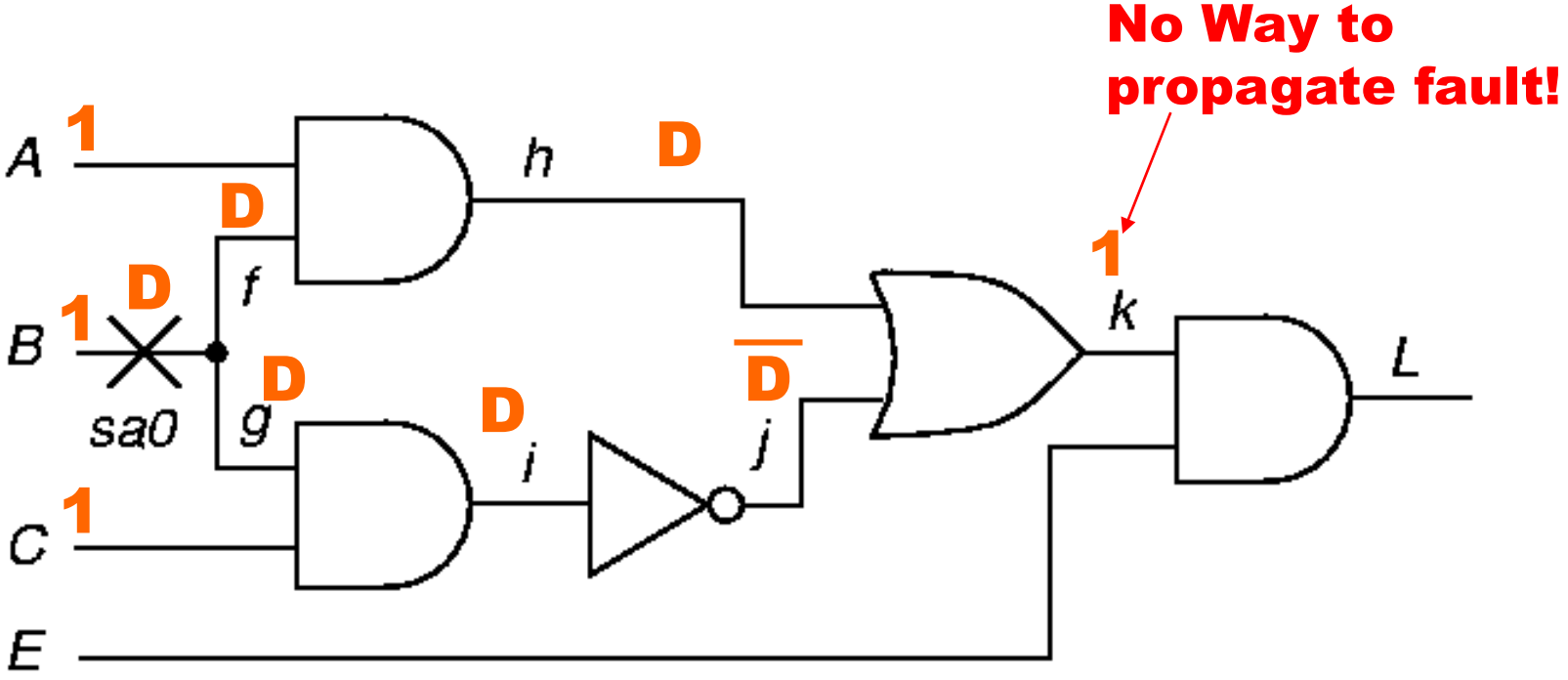
Path Sensitization Example

- Propagate through top path.
- Using 5-valued logic



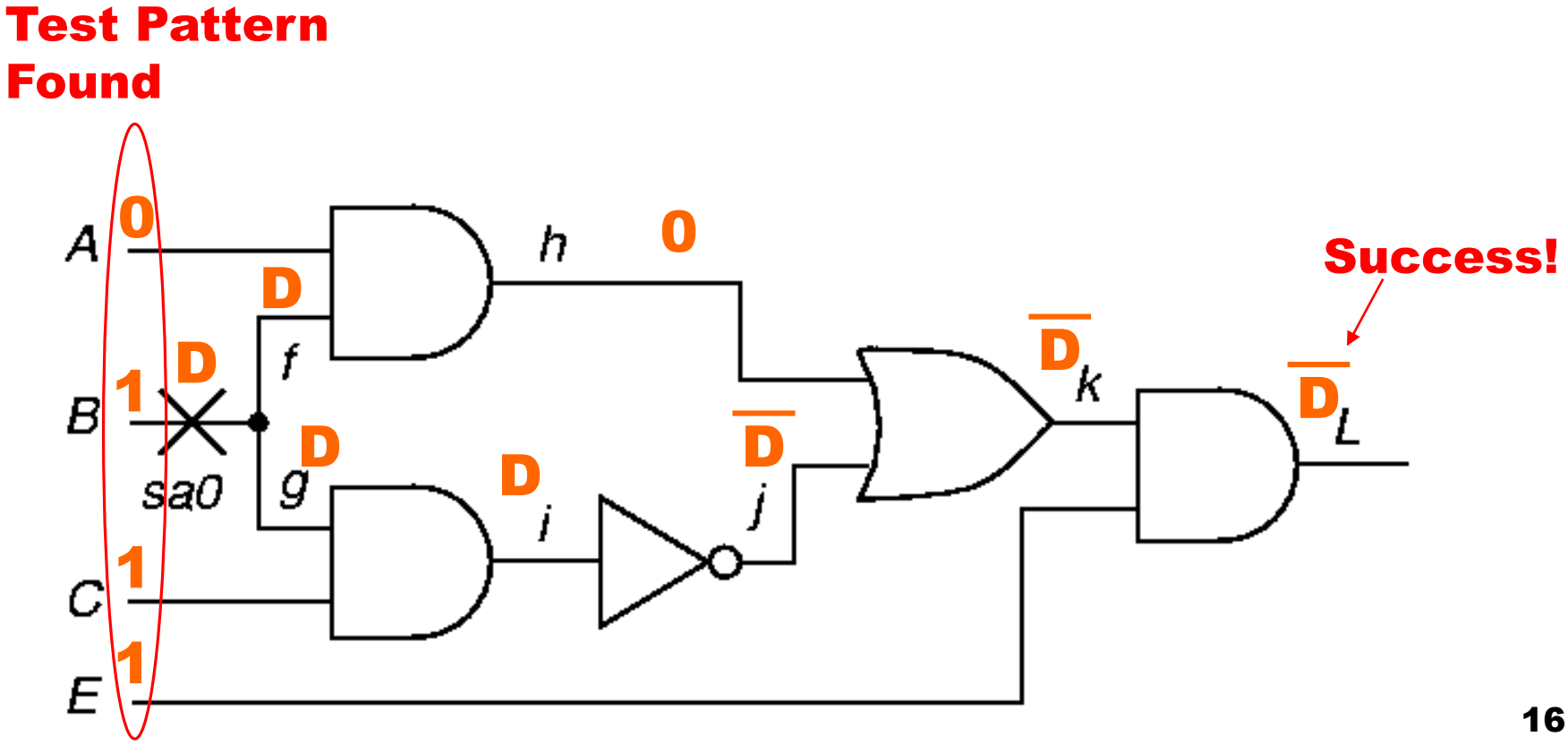
Path Sensitization Example (cont.)

- Propagate through top and bottom paths simultaneously
- Using 5-valued logic



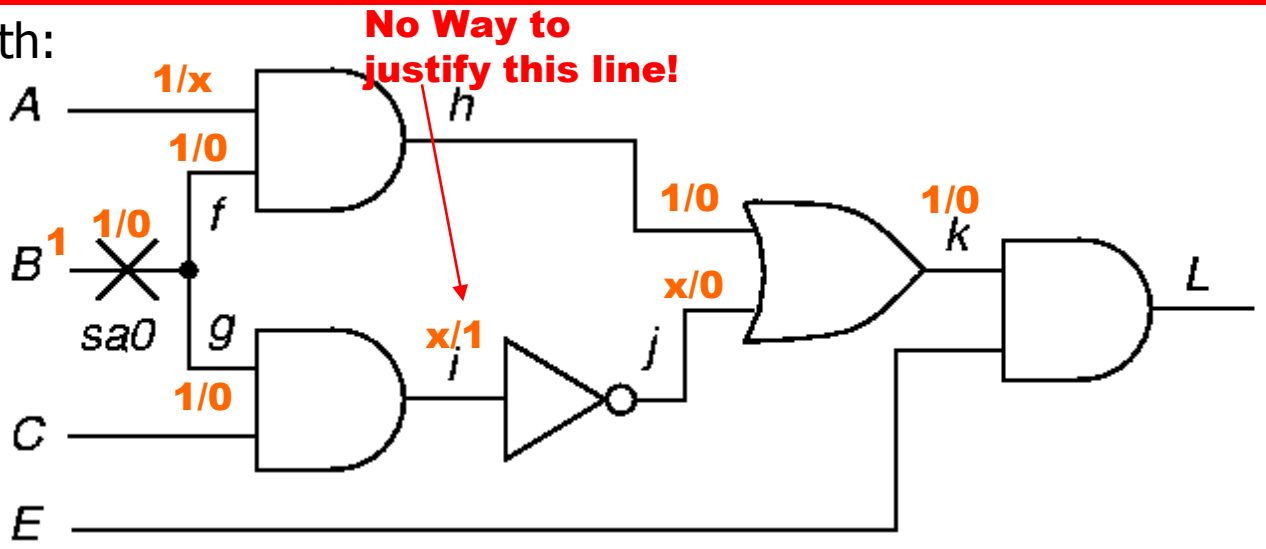
Path Sensitization Example (cont.)

- Propagate through the bottom path
- Using 5-valued logic

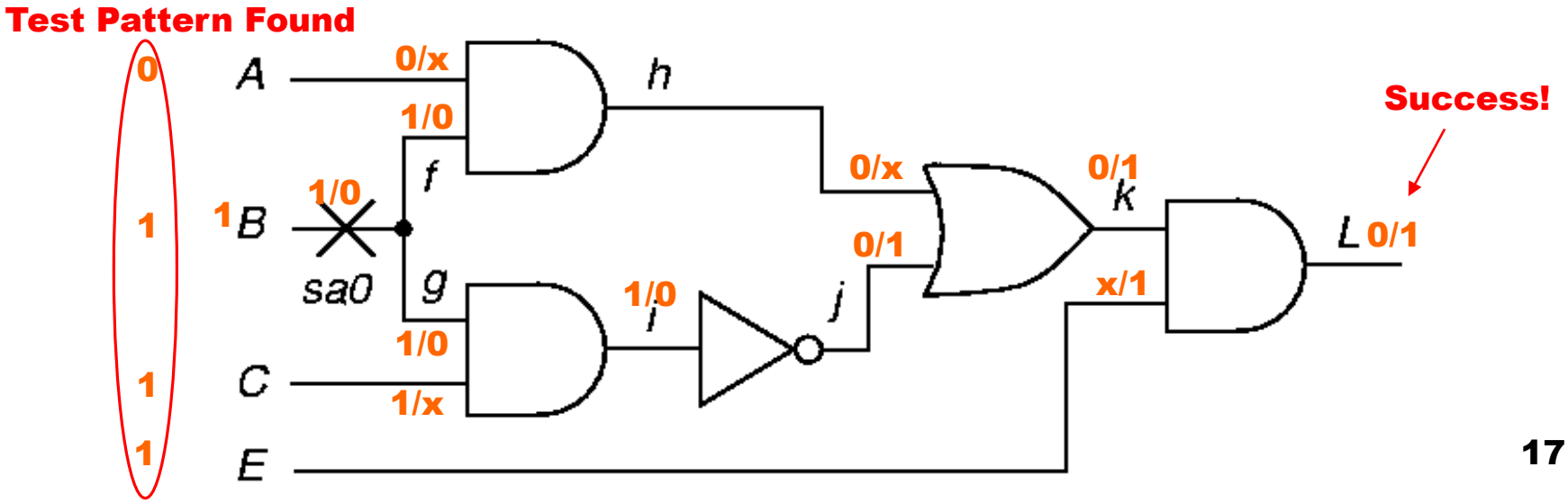


Path Sensitization Example (9-Valued Logic)

- Trying top path:

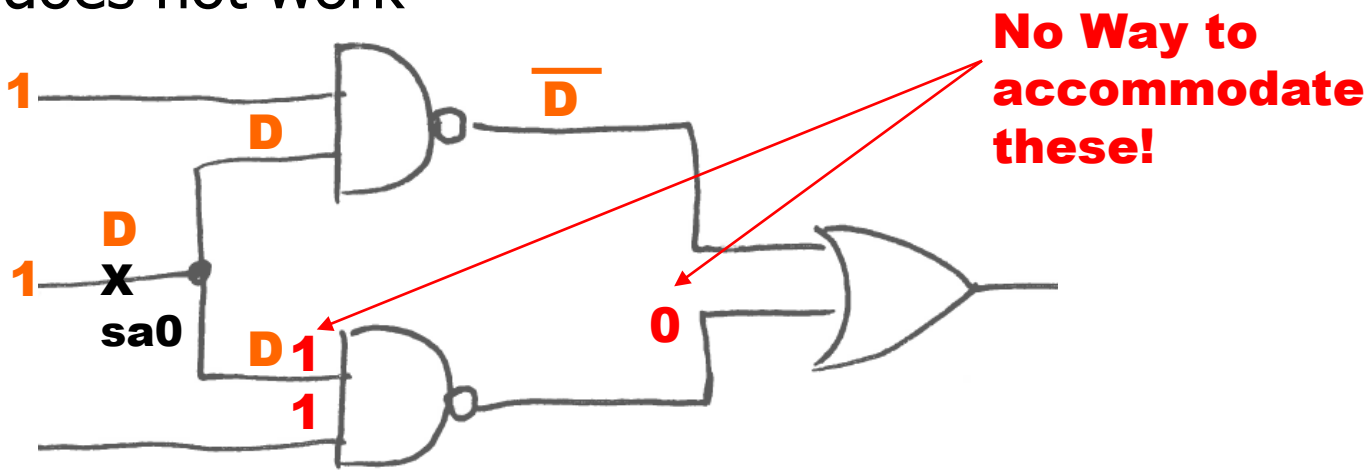


- Trying bottom path:

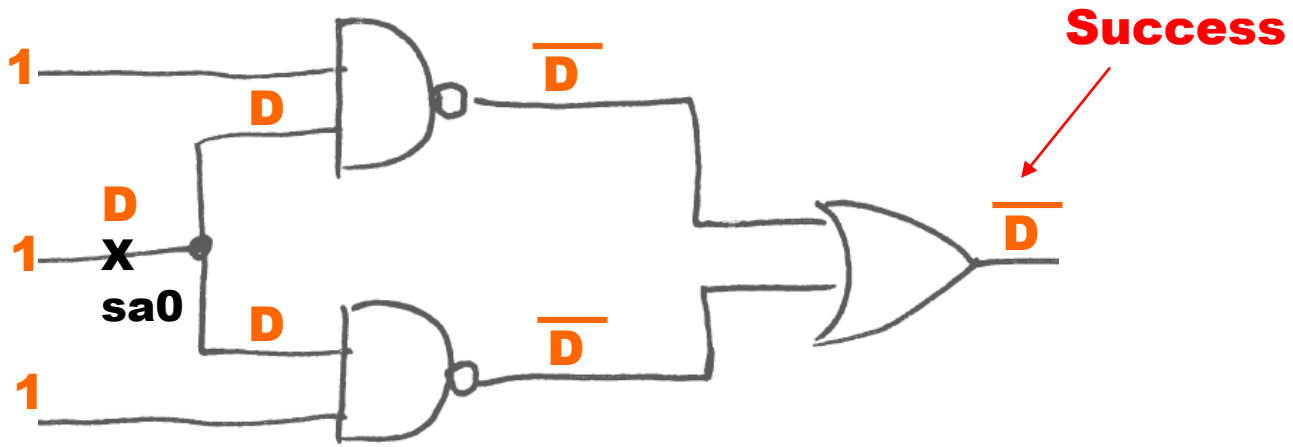


Path Sensitization – Example 2

- Top path does not work



- Bottom path does not work either (circuit is symmetrical)
- Both paths together can test the fault



History of Algorithms

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

Symbols Used in Value Systems

- In the **five-valued system**
  - Unique symbols – 0_5, 1_5, D_5, and $\overline{D}_5$ – are assigned to each set denoting a completely-specified value
  - One single symbol – X_5 – denotes all sets with more than one basic value

	Symbols in other value systems		
16-valued	Five-valued	Six-valued	Nine-valued
$\{\}$	—	—	—
$\{\underline{0}\}$	0_5	0_6	0_9
$\{\underline{1}\}$	1_5	1_6	1_9
$\{D\}$	D_5	D_6	D_9
$\{\overline{D}\}$	$\overline{D}_5$	$\overline{D}_6$	$\overline{D}_9$
$\{\underline{0},\underline{1}\}$	X_5	X_6	xx_9
$\{\underline{0},D\}$	X_5	X_6	$x0_9$
$\{\underline{0},\overline{D}\}$	X_5	X_6	$0x_9$
$\{\underline{1},D\}$	X_5	X_6	$1x_9$
$\{\underline{1},\overline{D}\}$	X_5	X_6	$x1_9$
$\{D,\overline{D}\}$	X_5	X_6	xx_9
$\{\underline{0},\underline{1},D\}$	X_5	X_6	xx_9
$\{\underline{0},\underline{1},\overline{D}\}$	X_5	X_6	xx_9
$\{\underline{0},D,\overline{D}\}$	X_5	X_6	xx_9
$\{\underline{1},D,\overline{D}\}$	X_5	X_6	xx_9
$\{\underline{0},\underline{1},D,\overline{D}\}$	X_5	X_6	xx_9

Symbols Used in Value Systems (cont.)

- In the **six-valued system**
  - Unique symbols – 0_6, 1_6, D_6, and $\overline{D}_6$ – are assigned to each set denoting a completely-specified value
  - One single symbol – χ_6 – denotes the set {$\underline{0}$, $\underline{1}$}, i.e., the only incompletely-specified set that does not contain D or $\overline{D}$
  - One single symbol – X_6 – denotes all sets with more than one basic value (including {$\underline{0}$, $\underline{1}$})

16-valued	Symbols in other value systems		
	Five-valued	Six-valued	Nine-valued
$\{\}$	—	—	—
$\{\underline{0}\}$	0_5	0_6	0_9
$\{\underline{1}\}$	1_5	1_6	1_9
$\{D\}$	D_5	D_6	D_9
$\{\overline{D}\}$	$\overline{D}_5$	$\overline{D}_6$	$\overline{D}_9$
$\{\underline{0},\underline{1}\}$	X_5	χ_6	xx_9
$\{\underline{0},D\}$	X_5	X_6	$x0_9$
$\{\underline{0},\overline{D}\}$	X_5	X_6	$0x_9$
$\{\underline{1},D\}$	X_5	X_6	$1x_9$
$\{\underline{1},\overline{D}\}$	X_5	X_6	$x1_9$
$\{D,\overline{D}\}$	X_5	X_6	xx_9
$\{\underline{0},\underline{1},D\}$	X_5	X_6	xx_9
$\{\underline{0},\underline{1},\overline{D}\}$	X_5	X_6	xx_9
$\{\underline{0},D,\overline{D}\}$	X_5	X_6	xx_9
$\{\underline{1},D,\overline{D}\}$	X_5	X_6	xx_9
$\{\underline{0},\underline{1},D,\overline{D}\}$	X_5	X_6	xx_9

Behavior of Gates Using Multi-Valued System

$V(c_{i_1})$ \ $V(c_{i_2})$	$\underline{0}$	$\underline{1}$	D	$\overline{D}$
$\underline{0}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$
$\underline{1}$	$\underline{1}$	$\underline{0}$	$\overline{D}$	D
D	$\underline{1}$	$\overline{D}$	$\overline{D}$	$\underline{1}$
$\overline{D}$	$\underline{1}$	D	$\underline{1}$	D

2-Input NAND

$V(c_{i_1})$ \ $V(c_{i_2})$	$\underline{0}$	$\underline{1}$	D	$\overline{D}$
$\underline{0}$	$\underline{1}$	$\underline{0}$	$\overline{D}$	D
$\underline{1}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$
D	$\overline{D}$	$\underline{0}$	$\overline{D}$	$\underline{0}$
$\overline{D}$	D	$\underline{0}$	$\underline{0}$	D

2-Input NOR

$V(c_{i_1})$ \ $V(c_{i_2})$	$\underline{0}$	$\underline{1}$	D	$\overline{D}$
$\underline{0}$	$\underline{0}$	$\underline{1}$	D	$\overline{D}$
$\underline{1}$	$\underline{1}$	$\underline{0}$	$\overline{D}$	D
D	D	$\overline{D}$	$\underline{0}$	$\underline{1}$
$\overline{D}$	$\overline{D}$	D	$\underline{1}$	$\underline{0}$

2-Input XOR

$V(c_i)$	$V(c_{j_1})$	$\cdots$	$V(c_{j_l})$	$\cdots$
$\underline{0}$	$\underline{0}$	$\cdots$	$\underline{0}$	$\cdots$
$\underline{1}$	$\underline{1}$	$\cdots$	$\underline{1}$	$\cdots$
D	D	$\cdots$	D	$\cdots$
$\overline{D}$	$\overline{D}$	$\cdots$	$\overline{D}$	$\cdots$

A fanout system

6-Valued Cube Cover

- A **cube** represents a combination of inputs.
  - **singular cover:** cubes with 0_6 or 1_6 at output (i.e., no fault effect)
  - **propagation cubes:** cubes with D_6 or $\overline{D}_6$ at output
  - **cube cover:** together, all the cubes

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
0_6	X_6	1_6
X_6	0_6	1_6
D_6	$\overline{D}_6$	1_6
$\overline{D}_6$	D_6	1_6
1_6	1_6	0_6
$\overline{D}_6$	1_6	D_6
1_6	$\overline{D}_6$	D_6
$\overline{D}_6$	$\overline{D}_6$	D_6
D_6	1_6	$\overline{D}_6$
1_6	D_6	$\overline{D}_6$
D_6	D_6	$\overline{D}_6$

2-Input NAND

Fault Excitation Cubes

- For test generation for single stuck-at fault in a combinational circuit
  - Only fault-free values may appear at the inputs of faulty circuit element
  - It is necessary to apply a combination of values at the inputs of the faulty element that can cause a D or $\overline{D}$ to appear at its outputs
  - **Fault excitation cubes** capture this aspect of a faulty circuit element

Fault Excitation Cubes (cont.)

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
1_6	1_6	$\overline{D}_6$

Two-input NAND with output SA1

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
0_6	χ_6	D_6
χ_6	0_6	D_6

Two-input NAND with output SA0

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
0_6	1_6	D_6

Two-input NAND with input c_{i_1} SA1

$V(c_i)$	$V(c_{j_1})$	$\cdots$	$V(c_{j_l})$	$\cdots$
0_6	0_6	$\cdots$	$\overline{D}_6$	$\cdots$

A fanout system with branch c_{j_l} SA1

$V(c_i)$	$V(c_{j_1})$	$\cdots$	$V(c_{j_l})$	$\cdots$
0_6	$\overline{D}_6$	$\cdots$	$\overline{D}_6$	$\cdots$

A fanout system with stem SA1

More Cube Covers

NAND

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
0_6	X_6	1_6
X_6	0_6	1_6
D_6	$\overline{D}_6$	1_6
$\overline{D}_6$	D_6	1_6
1_6	1_6	0_6
$\overline{D}_6$	1_6	D_6
1_6	$\overline{D}_6$	D_6
$\overline{D}_6$	$\overline{D}_6$	D_6
D_6	1_6	$\overline{D}_6$
1_6	D_6	$\overline{D}_6$
D_6	D_6	$\overline{D}_6$

AND

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
1_6	1_6	1_6
0_6	X_6	0_6
X_6	0_6	0_6
D_6	$\overline{D}_6$	0_6
$\overline{D}_6$	D_6	0_6
D_6	1_6	D_6
1_6	D_6	D_6
D_6	D_6	D_6
$\overline{D}_6$	1_6	$\overline{D}_6$
1_6	$\overline{D}_6$	$\overline{D}_6$
$\overline{D}_6$	$\overline{D}_6$	$\overline{D}_6$

OR

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
1_6	X_6	1_6
X_6	1_6	1_6
D_6	$\overline{D}_6$	1_6
$\overline{D}_6$	D_6	1_6
0_6	0_6	0_6
D_6	0_6	D_6
0_6	D_6	D_6
D_6	D_6	D_6
$\overline{D}_6$	0_6	$\overline{D}_6$
0_6	$\overline{D}_6$	$\overline{D}_6$
$\overline{D}_6$	$\overline{D}_6$	$\overline{D}_6$

XOR

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
1_6	0_6	1_6
0_6	1_6	1_6
D_6	$\overline{D}_6$	1_6
$\overline{D}_6$	D_6	1_6
0_6	0_6	0_6
1_6	1_6	0_6
D_6	D_6	0_6
$\overline{D}_6$	$\overline{D}_6$	0_6
0_6	D_6	D_6
D_6	0_6	D_6
1_6	$\overline{D}_6$	D_6
$\overline{D}_6$	1_6	D_6
1_6	D_6	$\overline{D}_6$
D_6	1_6	$\overline{D}_6$
0_6	$\overline{D}_6$	$\overline{D}_6$
$\overline{D}_6$	0_6	$\overline{D}_6$

Fanout System

$V(c_i)$	$V(c_{j_1})$	$\cdots$	$V(c_{j_l})$	$\cdots$
1_6	1_6	$\cdots$	1_6	$\cdots$
0_6	0_6	$\cdots$	0_6	$\cdots$
D_6	D_6	$\cdots$	D_6	$\cdots$
$\overline{D}_6$	$\overline{D}_6$	$\cdots$	$\overline{D}_6$	$\cdots$

More Fault Excitation Cubes

2-Input NAND

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
1_6	1_6	$\overline{D}_6$

output SA1

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
0_6	χ_6	D_6
χ_6	0_6	D_6

output SA0

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
0_6	1_6	D_6

input c_{i1} SA1

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
1_6	1_6	$\overline{D}_6$

input c_{i1} SA0

2-Input XOR

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
0_6	0_6	$\overline{D}_6$
1_6	1_6	$\overline{D}_6$

output SA1

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
0_6	1_6	D_6
1_6	0_6	D_6

output SA0

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
0_6	1_6	D_6
0_6	0_6	$\overline{D}_6$

input c_{i1} SA1

$V(c_{i_1})$	$V(c_{i_2})$	$V(c_j)$
1_6	0_6	D_6
1_6	1_6	$\overline{D}_6$

input c_{i1} SA0

Fanout System

$V(c_i)$	$V(c_{j_1})$	$\cdots$	$V(c_{j_l})$	$\cdots$
0_6	0_6	$\cdots$	$\overline{D}_6$	$\cdots$

branch c_{jl} SA1

$V(c_i)$	$V(c_{j_1})$	$\cdots$	$V(c_{j_l})$	$\cdots$
1_6	1_6	$\cdots$	D_6	$\cdots$

branch c_{jl} SA0

$V(c_i)$	$V(c_{j_1})$	$\cdots$	$V(c_{j_l})$	$\cdots$
0_6	$\overline{D}_6$	$\cdots$	$\overline{D}_6$	$\cdots$

stem c_i SA1

$V(c_i)$	$V(c_{j_1})$	$\cdots$	$V(c_{j_l})$	$\cdots$
1_6	D_6	$\cdots$	D_6	$\cdots$

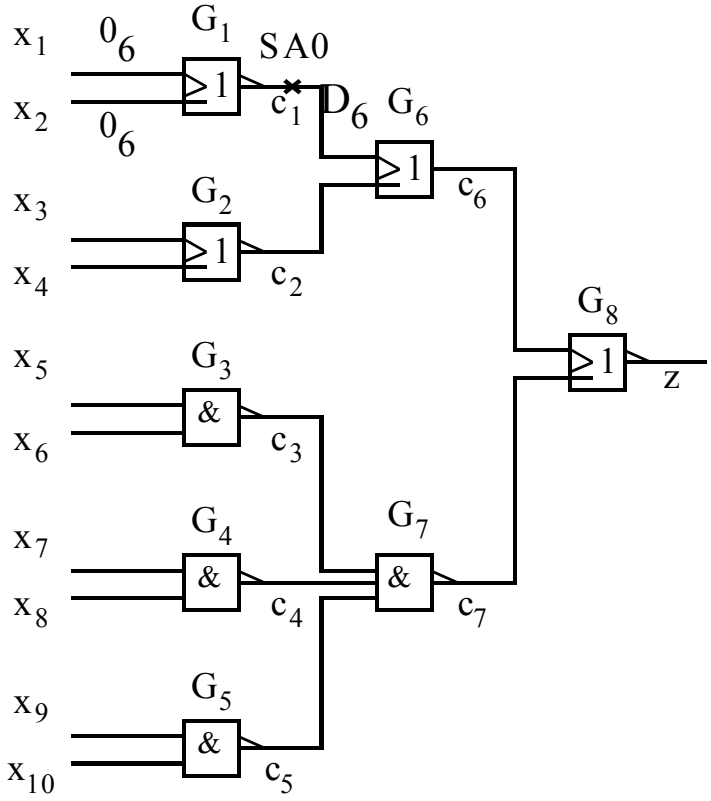
stem c_i SA0

Test Generation Basics

- **Fault effect excitation (FEE)** is the process of creating a fault effect (D or $\overline{D}$) at one or more outputs of the faulty circuit element
- **Fault effect propagation (FEP)** is the process of assigning values to circuit lines such that a fault-effect propagates from an output of the faulty circuit element to a primary output
  - One or more paths must exist between the fault site and a primary output, such that every line along the path has D_6 or $\overline{D}_6$

Test Generation Example

- In our example, only a single path c_1c_6z exists for such propagation, i.e., FEP via G_6 and G_8 necessary

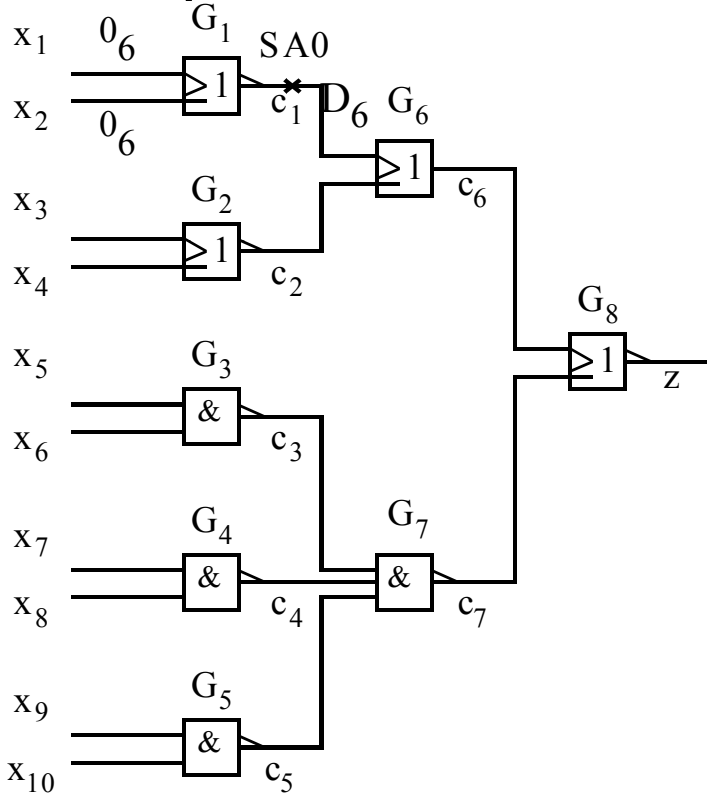


Fault excitation cube for G_1

$V(x_1)$	$V(x_2)$	$V(c_1)$
0_6	0_6	D_6

Test Generation Example (cont.)

- To propagate via G_6, first use propagation cubes for G_6
- Next, eliminate rows incompatible with currently assigned values
- Make one of the remaining assignments
  - Only one remained in this case

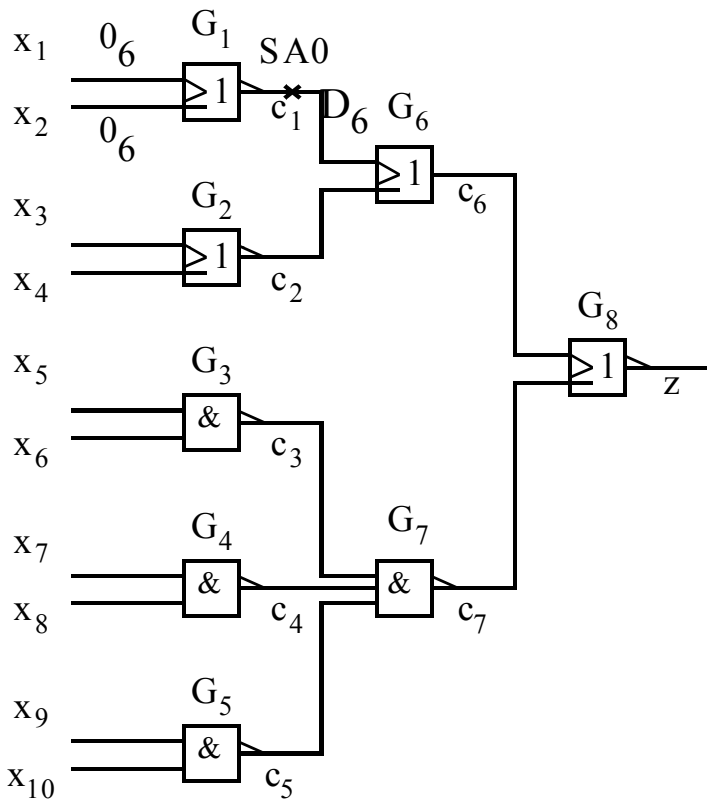


Propagation cubes for G_6

$V(c_1)$	$V(c_2)$	$V(c_6)$
D_6	0_6	D_6

Test Generation Example (cont.)

- Similarly, propagate via G_8

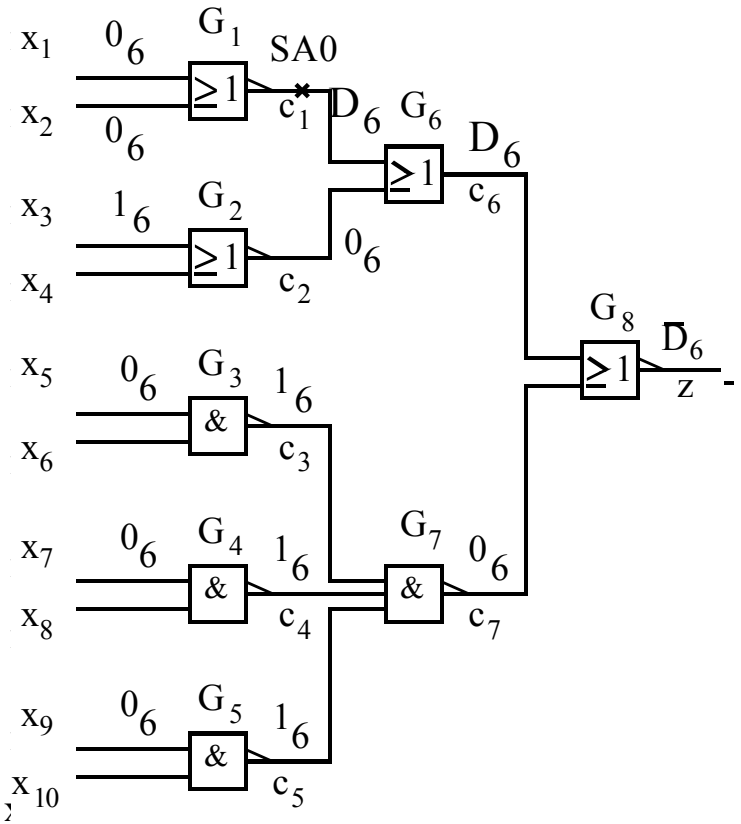


Propagation cubes for G_8

V(c6)	V(c7)	V(z)
D_6	0_6	$\overline{D_6}$

Test Generation Example (cont.)

- **Justification** is the task of assigning values to primary inputs so as to imply a desired value at an internal line
  - Values at c_2 and c_7 are **not** justified

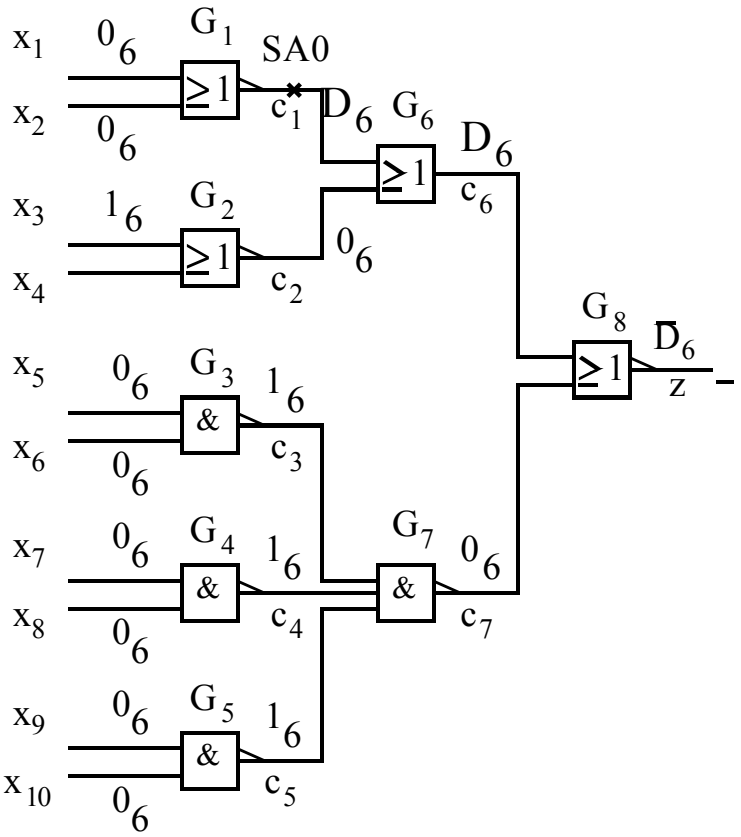


Singular cover of G_2

$V(x_3)$	$V(x_4)$	$V(c_2)$
1_6	X_6	0_6
X_6	1_6	0_6

Test Generation Example (cont.)

- **Justification** is the task of assigning values to primary inputs so as to imply a desired value at an internal line
  - Values at c_2 and c_7 are **not** justified
  - Justify value at c_2
    - Identify all cubes in singular cover of G_2
    - Eliminate cubes not consistent with currently applied values
      - Two choices remain
        - Either 1_6 at x_3
        - Or 1_6 at x_4
      - In this **fanout-free circuit** use either one
  - Similarly, justify value at c_7

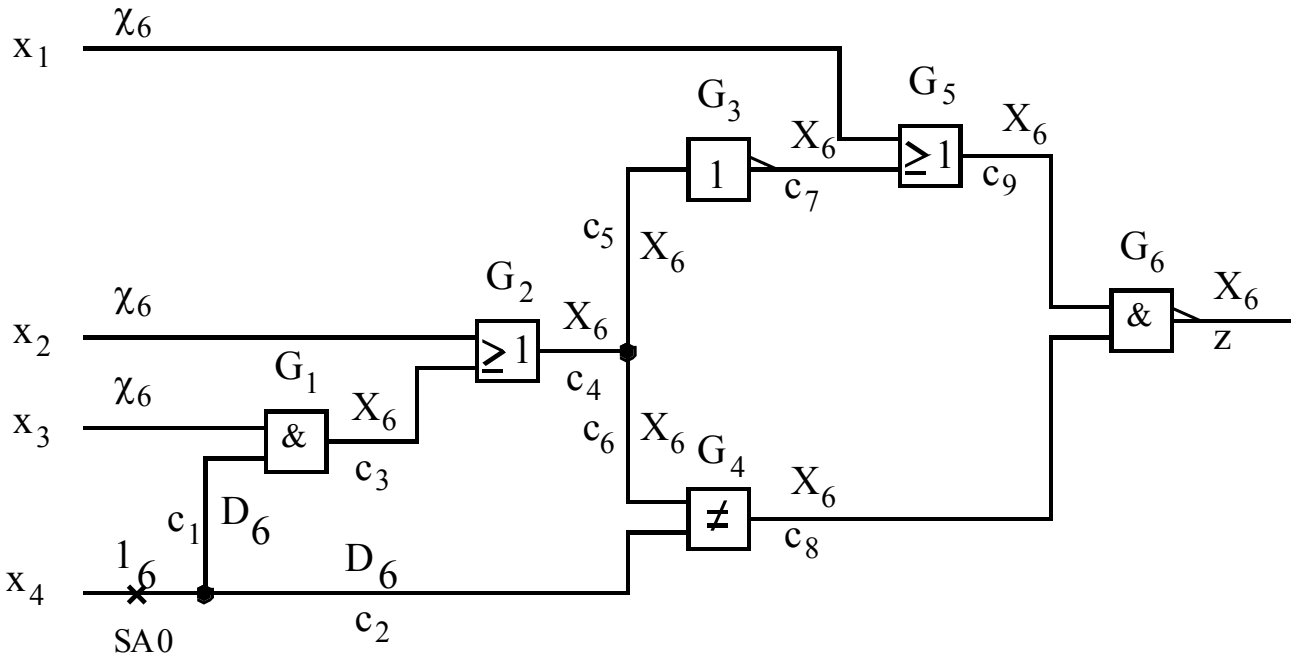


Singular cover of G_2

$V(x_3)$	$V(x_4)$	$V(c_2)$
1_6	X_6	0_6
X_6	1_6	0_6

Test Generation Example (cont.)

- Consider a circuit with fanouts for a more general case
  - Initialization: All lines in fanout of fault site x_6, other lines χ_6
  - Fault-effect excitation
  - G_1 and G_4 as possible sites for FEP
    - **D-frontier** is a set of gates where each gate has
      - D_6 or $\overline{D}_6$ at one or more inputs, and
      - X_6 at output
    - Each gate in D-frontier is a candidate for FEP
    - Here, $D = \{G_1, G_4\}$

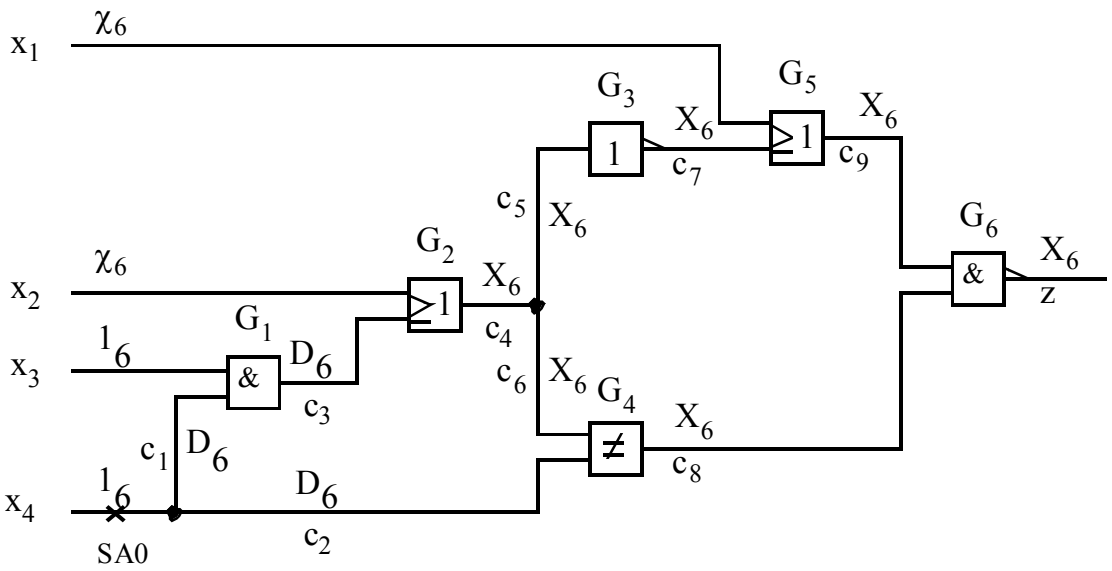


Fault excitation cubes for fanout system with stem x_4 with SA0 at x_4

$V(x_4)$	$V(c_1)$	$V(c_2)$
1_6	D_6	D_6

Test Generation Example (cont.)

- Since D = {G_1, G_4} has more than one gate
  - Select, only say, G_1 for FEP
  - Perform FEP via G_1
    - Assign 1_6 at x_3
  - Now the D-frontier D = {G_2, G_4}

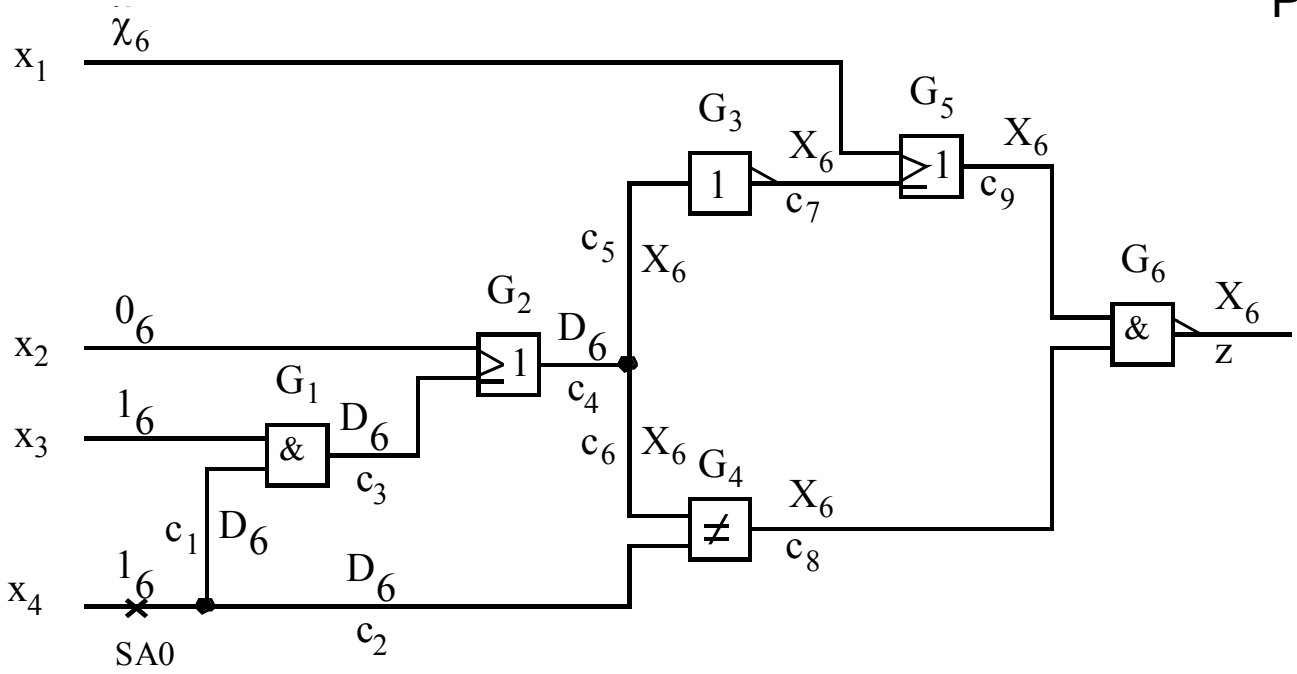


Propagation cubes for G_1

$V(x_3)$	$V(c_1)$	$V(c_3)$
1_6	D_6	D_6

Test Generation Example (cont.)

- Since D = {G_2, G_4}
  - Select one, say G_2
  - FEP via G_2
    - Assign 0_6 to x_2

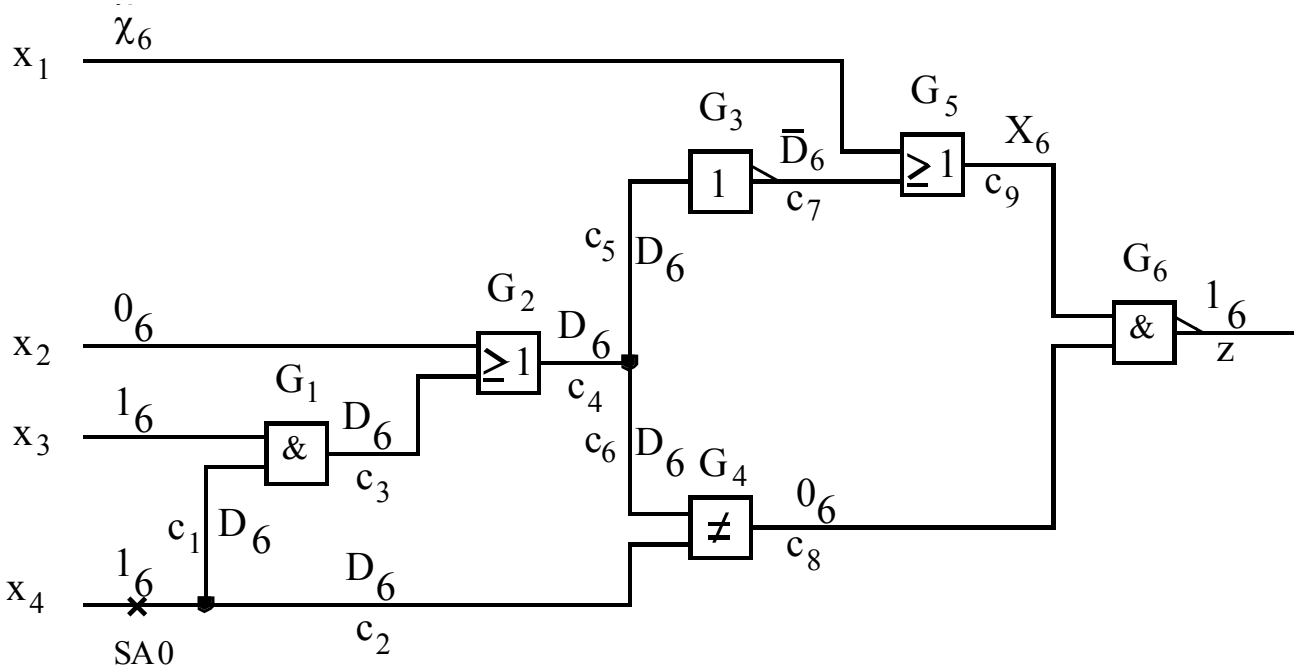


Propagation cubes for G_2

x_2	c_3	c_4
0_6	D_6	D_6

Test Generation Example (cont.)

- Implications of currently assigned values
  - — Only when a unique value can be identified



Relevant parts of cube covers of:

Fanout system with stem c_4

$V(c_4)$	$V(c_5)$	$V(c_6)$
D_6	D_6	D_6

NOT gate G_3

$V(c_5)$	$V(c_7)$
D_6	$\overline{D}_6$

XOR gate G_4

$V(c_2)$	$V(c_6)$	$V(c_8)$
D_6	D_6	0_6

OR gate G_5

$V(x_1)$	$V(c_7)$	$V(c_9)$
1_6	X_6	1_6
0_6	D_6	D_6

NAND gate G_6

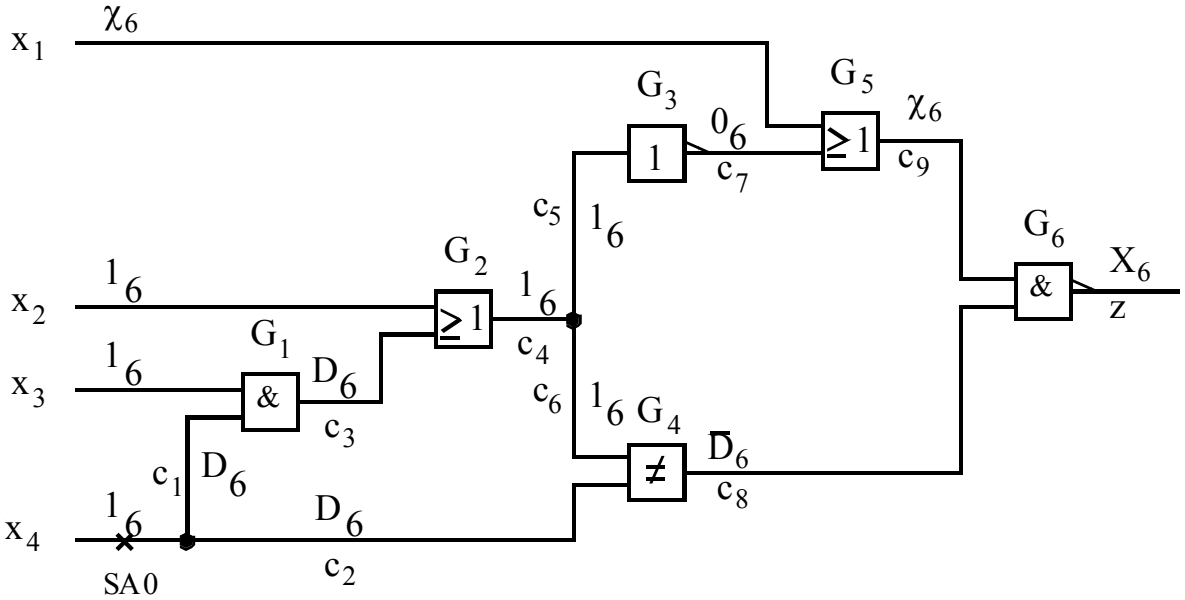
$V(c_8)$	$V(c_9)$	$V(z)$
0_6	X_6	1_6

Test Generation Example (cont.)

- Current status
  - $D = \{G_5\}$
  - But $V(z) = 1_6$, i.e., no possibility of obtaining D_6 or $\overline{D}_6$ at output
  - In other words, proceeding further will **not** lead to a test vector
  - Hence we must **backtrack**, i.e.,
    - Identify the most recent primary input value assignment for which an untried alternative exists
      - Restore the state to prior to that assignment was made
      - Make the alternative value assignment

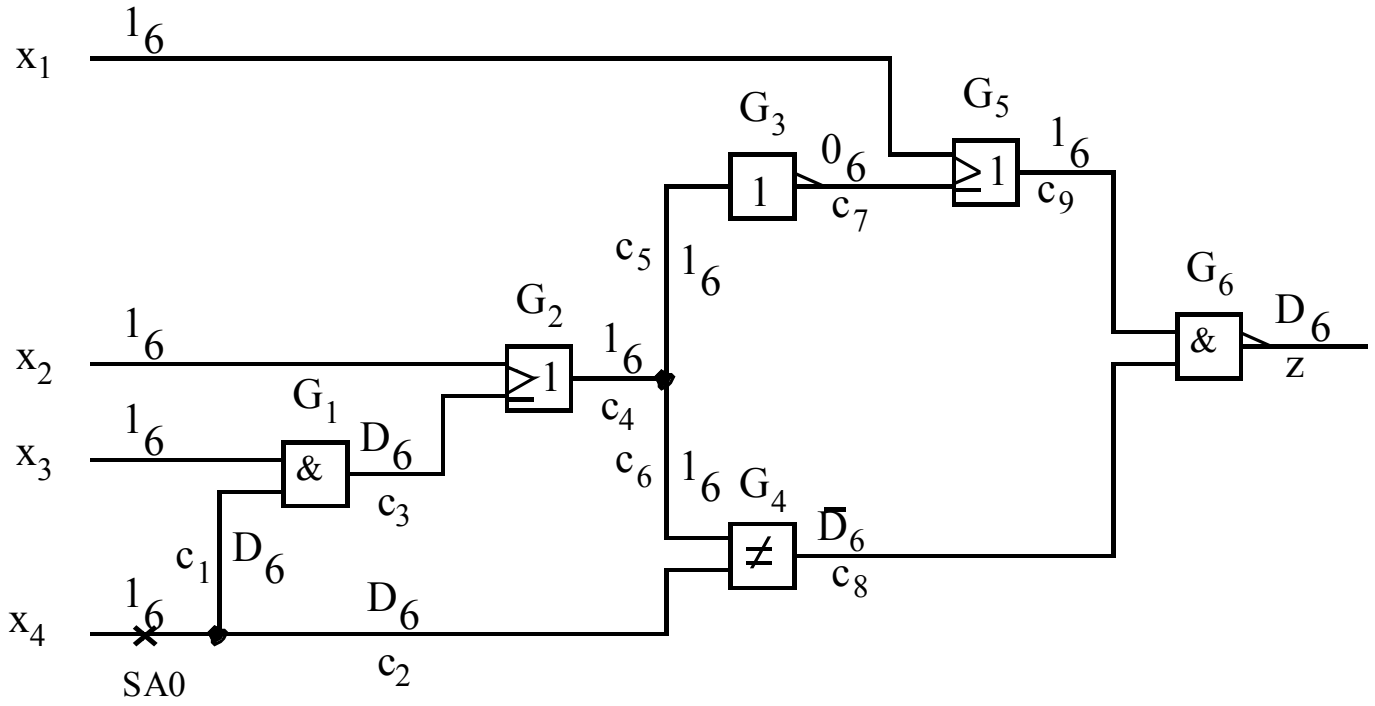
Test generation basics

- Identifying the most recent primary input value assignment for which an untried alternative exists
  - During assignment of 0_6 to x_2, alternative 1_6 at x_2 untried
- Executing backtrack
  - Restore circuit values to before 0_6 was assigned to x_2
  - Assign 1_6 at x_2 (no more alternatives exist)
  - Perform implication



Test Generation Example (cont.)

- Now D = {G_6}
  - Select G_6 for FEP
    - Objective 1_6 at c_9, which can be attained by 1_6 at x_1
      - Assign 1_6 at x_1
    - Perform implication



Test Generation Example (cont.)

- Now D_6 at output and no unjustified lines
  - Hence test generation complete with vector (1_6, 1_6, 1_6, 1_6)
  - In reality, need to apply only the value corresponding to fault-free circuit in each of above composite values
    - That is, apply test (1, 1, 1, 1)

Importance of Implication

- **Implication** is the process of determining logic values that appear at circuit lines as a consequence of the values assigned to some of its lines
- It also
  - Identifies any inconsistency between the values assigned
  - Updates D-frontier and list of unjustified lines
- An **implication procedure** iteratively uses **implication operation**
  - A **direct implication** operation determines values at inputs or outputs of a circuit element, given values assigned at the element's inputs and/or outputs
    - **Forward** implication operation proceeds from lines to those in its fanout
    - **Backward** implication proceeds from lines to those in fanin

Generic Test Generation Algorithm

1. Read the circuit under test (CUT)
2. Preprocessing
   a. Identify static indirect implications
   b. Compute testability metrics
3. For each target fault in the fault list
   a. Insert the target fault by using the appropriate description of the behavior of the faulty element
   b. Initialize circuit lines
      - In five-valued system, assign X_5 to all lines
      - In six-valued system, assign X_6 to all lines in transitive fanout of fault site and χ_6 to all other lines
      - In sixteen-valued system
        + Assign {$\underline{0}$, $\underline{1}$, D, $\overline{D}$} to all lines in transitive fanout of the fault site
        + Assign {$\underline{0}$, $\underline{1}$} to all the other lines
        + Perform forward implication starting at each output of the faulty circuit element

Generic Test Generation Algorithm (cont.)

c. Identify test generation sub-tasks (TGSTs)
   - Analyze values at primary outputs of CUT and outputs of the faulty circuit element
   - Analyze gates in the D-frontier D
   - Analyze the set of unjustified lines U
   - Determine, whether
     + Test generation is complete – print vector, manage fault list, go to Step 3 and process next fault
     + Conflict, i.e., not possible to generate a vector for target by further specifying currently assigned vector (values) – initiate **backtrack**
     + Continue search by carrying out a TGST
       1. FEE – fault-effect excitation,
       2. FEP – fault-effect propagation, or
       3. Justification

d. Identify a value assignment to accomplish the selected TGST

e. Assign selected value
   - Save the state of the test generation: e.g., values at circuit lines, untried alternatives TGSTs and/or value assignments
   - Assign selected value
   - Perform implications
     + If successful, continue
     + Else, backtrack

Identification of TGSTs

- Backtrack
  - Find the most recent value assignment at which an untried alternative existed
  - Restore the state of the test generation algorithm to prior to that value assignment
  - Make an alternative TGST/value assignment and try it, starting at Step 3d or 3e
- Assignment of each logic value by test generator followed by implication that
  - Either returns CONFLICT and initiates backtrack
  - Else, returns SUCCESS after updating
    - Values at circuit lines
    - D-frontier D
    - List of unjustified lines U
- Subsequently, values at circuit lines, D, and U are analyzed to identify TGSTs with following possible outcomes
  - Fault-effect excitation (FEE)
  - Fault-effect propagation (FEP)
  - Justification
  - Backtrack
    - FEE impossible
    - FEP impossible
  - Test generation complete

Identification of TGSTs