

ECE 538

VLSI System Testing

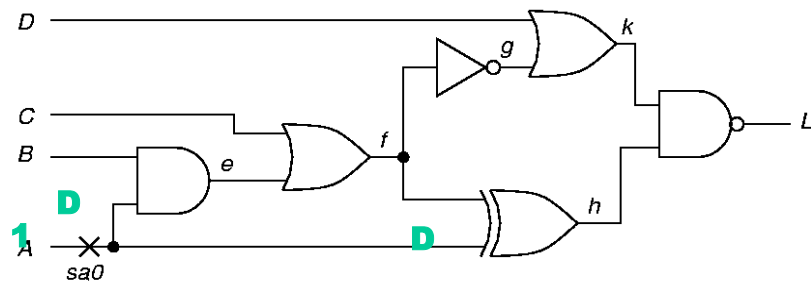
Krish Chakrabarty

Test Generation: 2

Outline

- Problem with D-Algorithm
- PODEM
- FAN
- Fault-independent ATPG
 - Critical path tracing
- Random test generation
- Redundancy identification

- Step 1 – *D-Drive* – Set $A = 1$

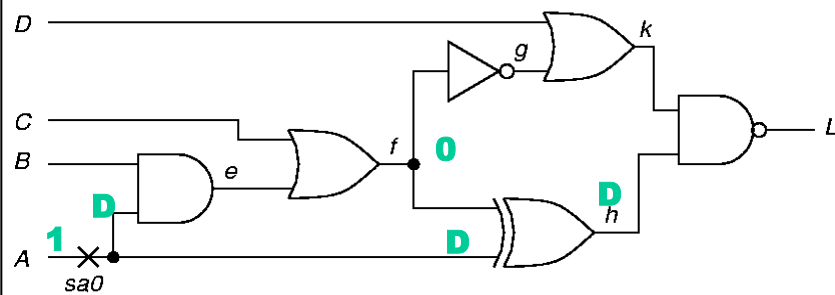


ECE 538

Krish Chakrabarty

3

- Step 2 – *D-Drive* – Set $f = 0$

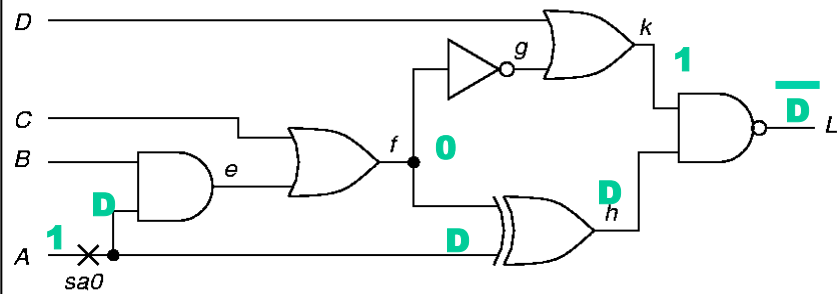


ECE 538

Krish Chakrabarty

4

- Step 3 – *D-Drive* – Set $k = 1$

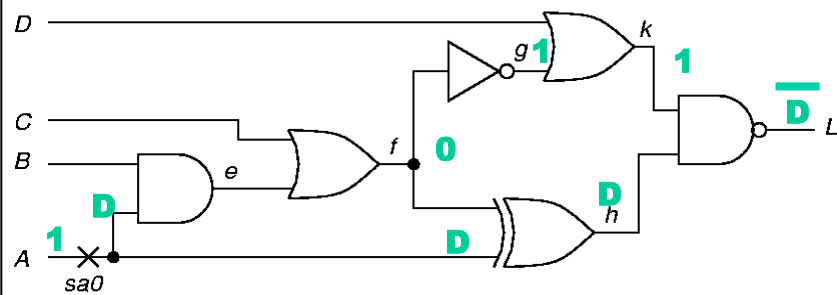


ECE 538

Krish Chakrabarty

5

- Step 4 – *Consistency* – Set $g = 1$

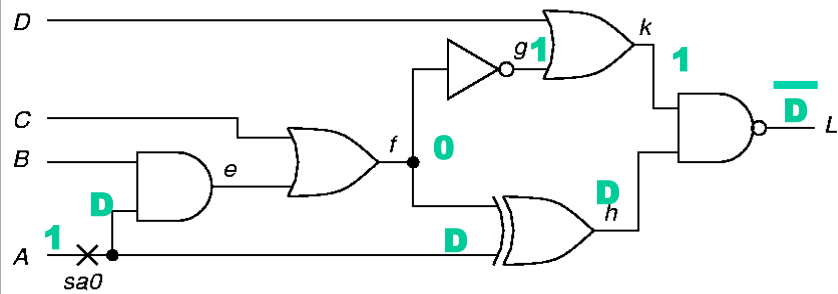


ECE 538

Krish Chakrabarty

6

- Step 5 – Consistency – $f = 0$ Already set

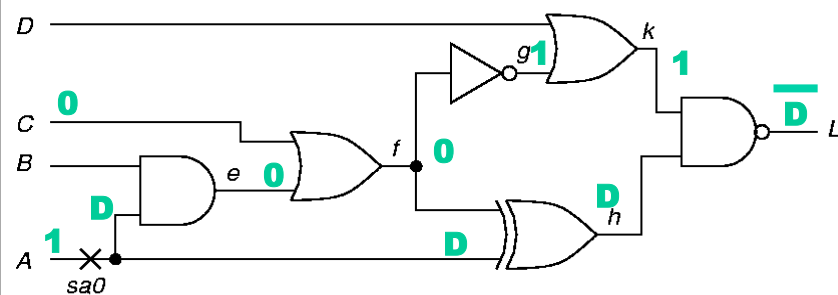


ECE 538

Krish Chakrabarty

7

- Step 6 – Consistency – Set $c = 0$, Set $e = 0$

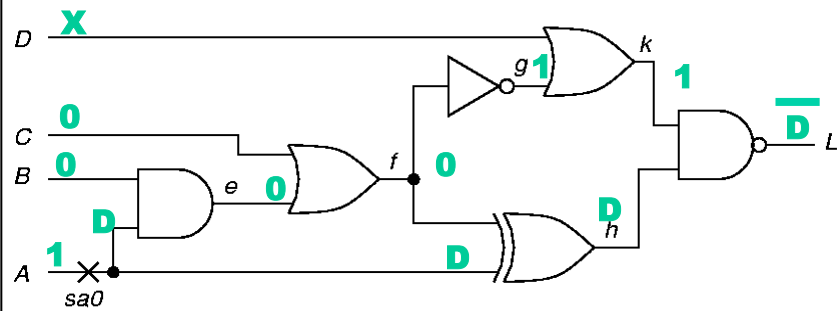


ECE 538

Krish Chakrabarty

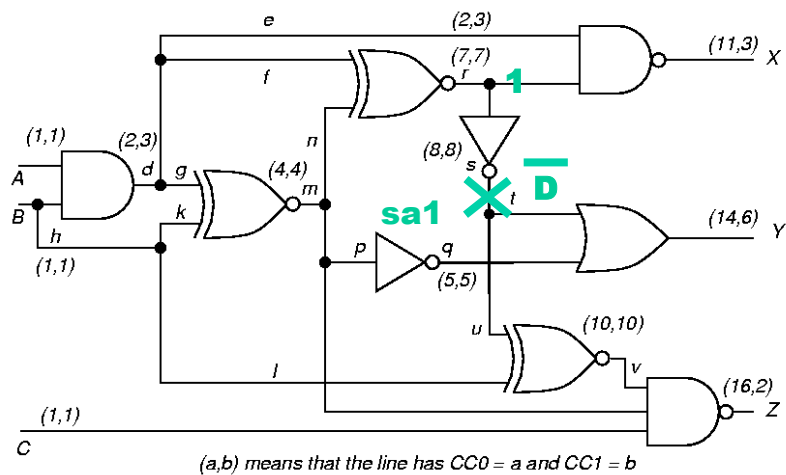
8

- Step 7 – *Consistency* – Set $B = 0$



9

Example 7.3 – Fault s sal



10

Example 7.3 – Step 2 s sa1

The diagram illustrates the propagation of the $sa1$ fault through a logic circuit. The circuit has inputs A , B , and C , and outputs X , Y , and Z . The logic components are AND, OR, XOR, and NOT gates. The diagram shows the propagation of the $sa1$ fault, with annotations indicating the fault's effect on various lines. A green 'X' is placed over the $sa1$ label, indicating that the fault is blocked or results in a don't care value. The diagram also shows the propagation of the fault through the circuit, with some paths being blocked or resulting in don't care values.

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

11

Example 7.3 – Step 2 s sa1

- Forward & Backward Implications

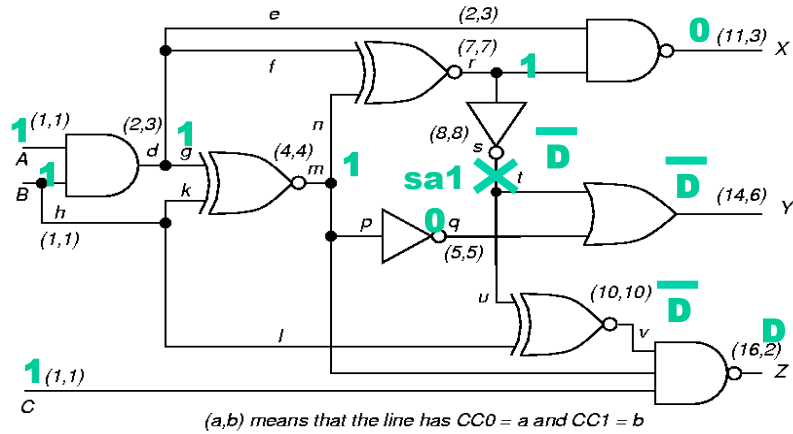
(a,b) means that the line has $CC0 = a$ and $CC1 = b$

- ## Example 7.3 – Step 2 s sa1
- Forward & Backward Implications
-
- The diagram illustrates a circuit with inputs A , B , and C , and outputs X , Y , and Z . Internal nodes are labeled with coordinates (a,b) , where a is the current state of $CC0$ and b is the current state of $CC1$. Green annotations show the propagation of the 'sa1' fault from input A through the circuit. A red 'X' marks a conflict at node t .
- (a,b) means that the line has $CC0 = a$ and $CC1 = b$

12

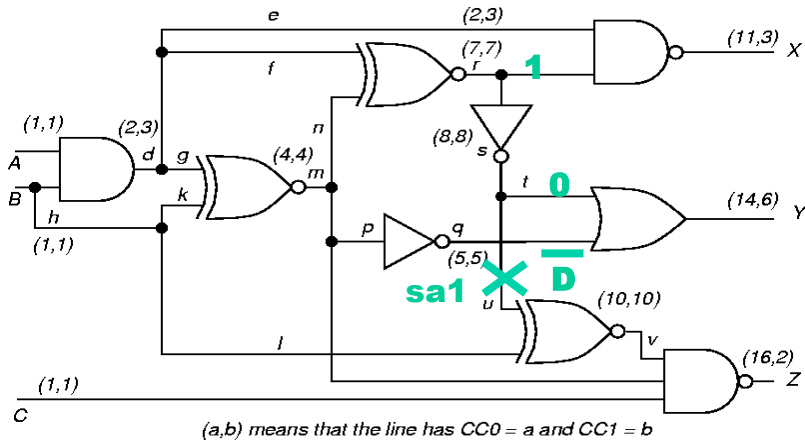
Example 7.3 – Step 3 s sa1

- Test found!



13

Example 7.3 – Fault u_{sa1}



14

Example 7.3 – Step 2 *u* sa1

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

15

Example 7.3 – Step 2 *u* sa1

- Forward and backward implications

The diagram illustrates a circuit with inputs A , B , and C , and outputs X , Y , and Z . The circuit includes several logic gates: AND gates, OR gates, XOR gates, and NOT gates. The circuit is annotated with green values (0, 1, D) and pairs (a,b) indicating the state of the circuit at each node. The pairs (a,b) mean that the line has $CC0 = a$ and $CC1 = b$.

Key annotations and values shown in the diagram:

- Input A is 0, B is 0, and C is 0.
- Node d is 0, g is 0, k is 0, m is 1, n is 1, p is 0, q is 0, r is 1, s is 0, t is 0, u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node h is 1, i is 1, j is 1, l is 1, o is 1, u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is 0.
- Node u is 1, v is 1, w is 1, x is 1, y is 1, z is 1.
- Node u is 0, v is 0, w is 0, x is 0, y is 0, z is

- Forward and backward implications

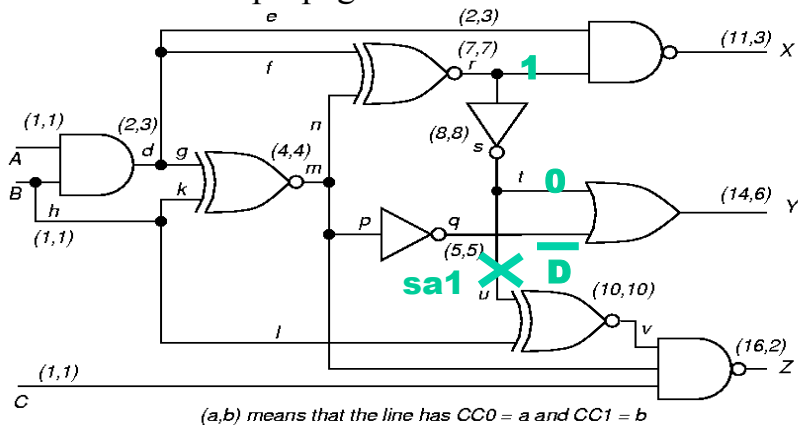
16

Inconsistent

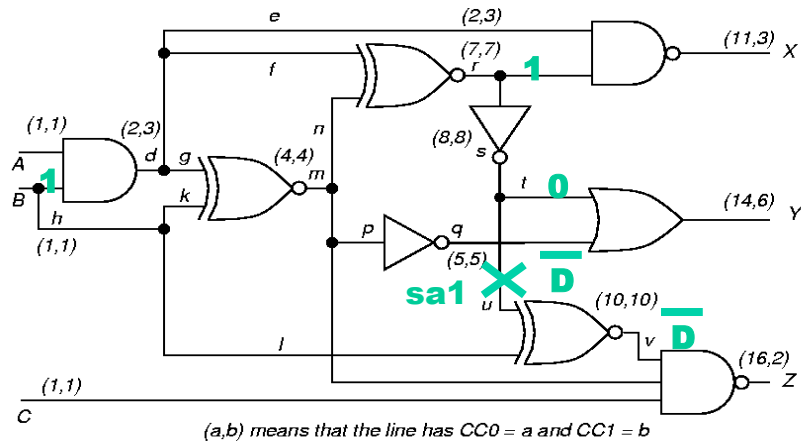
- $d = 0$ and $m = 1$ cannot justify $r = 1$ (equivalence)
 - Backtrack
 - Remove $B = 0$ assignment

Example 7.3 – Backtrack

- Need alternate propagation



Example 7.3 – Step 3 *u sa1*

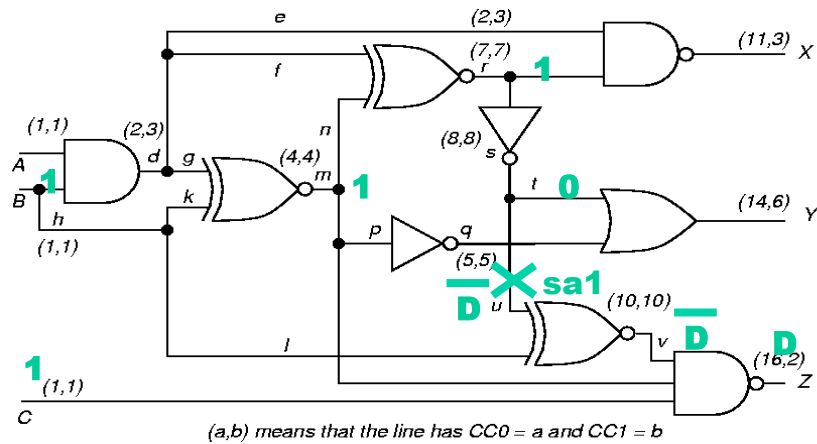


ECE 538

Krish Chakrabarty

19

Example 7.3 – Step 4 *u sa1*



ECE 538

Krish Chakrabarty

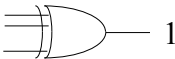
20

[illegible]


21

Problem with D-Algorithm

Excessive backtracking occurs in certain types of circuits

2^{n-1} justifying values 

Causes “ripple effect” in many circuits, e.g. adders, parity circuits, error correcting circuits

2^{n-1} justifying values 

22

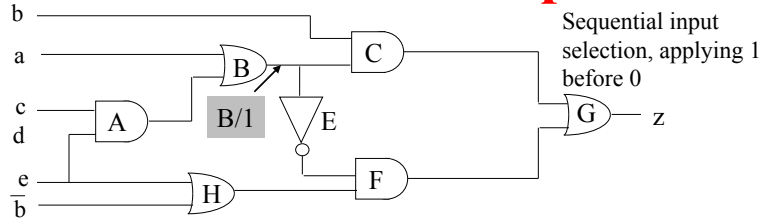
PODEM: “Path-Oriented Decision Making”

- *Similarity with D-algorithm*: circuit-based, fault-oriented
- *Difference*: Signal values explicitly assigned only at primary outputs, others computed by implication
- Justification not needed!
- Backtracking means reassigning primary inputs when contradiction occurs: “implicit enumeration”
- Simple “backtrace” heuristic used to select primary input

Branch and Bound Search

- Efficiently searches binary search tree
- *Branching* – At each tree level, selects which input variable to set to what value
- *Bounding* – Avoids exploring large tree portions by artificially restricting search decision choices
 - Complete exploration is impractical
 - Uses *heuristics*

PODEM Example



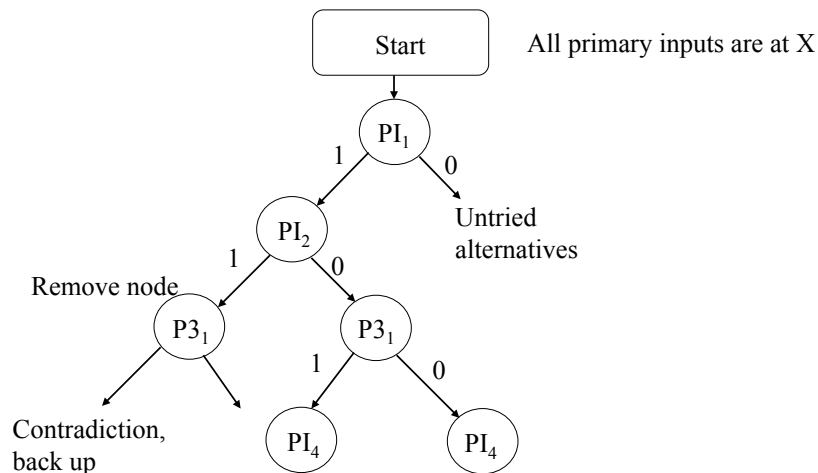
Decision	Implication	Comment
a = 1		Contradiction at fault Backtrack
a = 0		
b = 1		
c = 1		
e = 1	A = 1, B = 1	Contradiction, backtrack
e = 0	A = 0, B = \overline{D} , E = D, H = 0, F = 0, C = \overline{D} , z = \overline{D}	Test found!

ECE 538

Krish Chakrabarty

25

PODEM Decision Tree



ECE 538

Krish Chakrabarty

26

PODEM Steps

- Input Assignment
 - Unassigned PIs are selected and assigned new values systematically
 - All implications of each assignment are determined
 - If D/\bar{D} is implied on a primary output, a test has been found; otherwise a new assignment or a new primary input line is selected

PODEM Steps

- Primary inputs selection:
 - INITIAL OBJECTIVE: A series of “initial” objectives of the form $IO_j = (l, v)$ are determined. The first IO_0 is to apply $v = D/\bar{D}$ to the fault site.
 - BACKTRACING: For each initial objective IO_j , a path is traced backwards through the circuit to a primary input via a series of “current” objectives
 - Current objectives are selected by heuristics

PODEM Procedures

Procedure *Backtrace*(k, v_k)

/ Map objective into PI assignment */*

begin

$v = v_k$;

while k is a gate input

begin

i = inversion value of k ;

select an input (j) of k with value x ;

$v = v \oplus i$;

$k = j$;

end

/ k is a PI */*

return (k, v);

end

Procedure *Objective*()

begin

/ the target is l/v */*

if (value of l is X) **then return** (l, v);

select a gate (G) from the D-frontier;

select an input (j) of G with value X;

c = controlling value of G ;

return (j, c);

end

PODEM Procedures

PODEM()

begin

if (error at PO) **then return** SUCCESS

if (test not possible) **then return** FAILURE

(k, v_k) = *Objective*();

(j, v_j) = *Backtrace*(k, v_k);

Imply(j, v_j);

if *PODEM*() = SUCCESS **then return** SUCCESS

/ reverse decision */*

Imply(j, v_j);

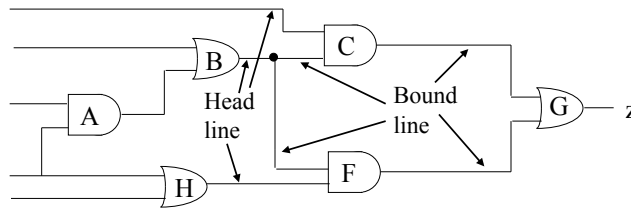
if *PODEM*() = SUCCESS **then return** SUCCESS;

Imply(j, X);

return FAILURE

FAN: “Fanout-Oriented Test Generation”

- Two major extensions to PODEM
 - Backtracing may stop at internal lines
 - Multiple backtrace-procedures attempts to simultaneously satisfy a set of objectives
- Backtracing can stop at *head lines*



ECE 538

Krish Chakrabarty

31

Selection Criteria

- Controllability (CC0 and CC1) and observability measures (CO)
 - Exact values can only be determined by exhaustive simulation
 - Estimates are useful for guiding test generation (more controllable \Leftrightarrow low values, more observable \Leftrightarrow low values)

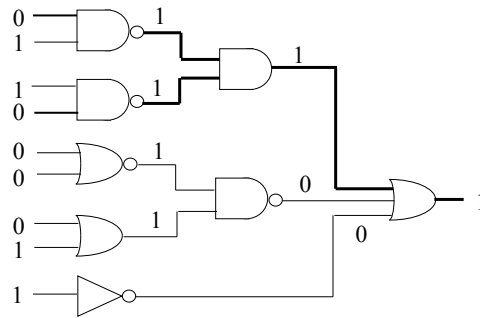
ECE 538

Krish Chakrabarty

32

Critical Path Test Generation

- Recursively determine critical paths

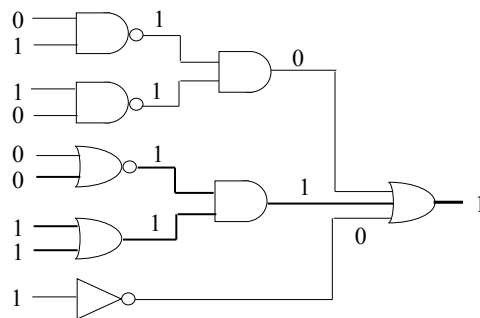


ECE 538

Krish Chakrabarty

33

Critical Path Test Generation



ECE 538

Krish Chakrabarty

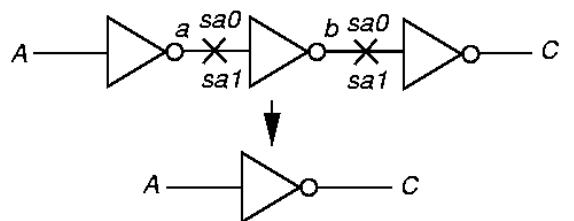
34

Redundancy Removal Using ATPG

- Redundancy identification
- Redundancy removal

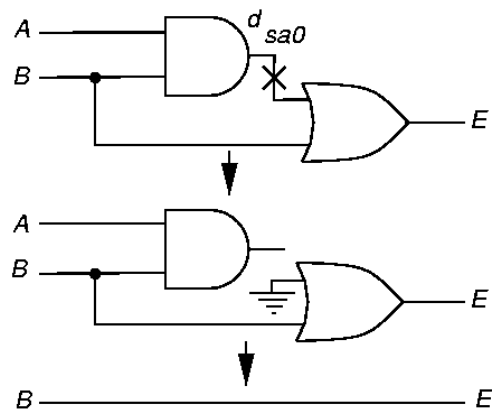
Irredundant Faults

- Combinational ATPG can find redundant (unnecessary) hardware



- | | |
|------------------|---------|
| • Fault | Test |
| $a\ sa1, b\ sa0$ | $A = 1$ |
| $a\ sa0, b\ sa1$ | $A = 0$ |
- Therefore, these faults are not redundant

Redundant Hardware and Simplification

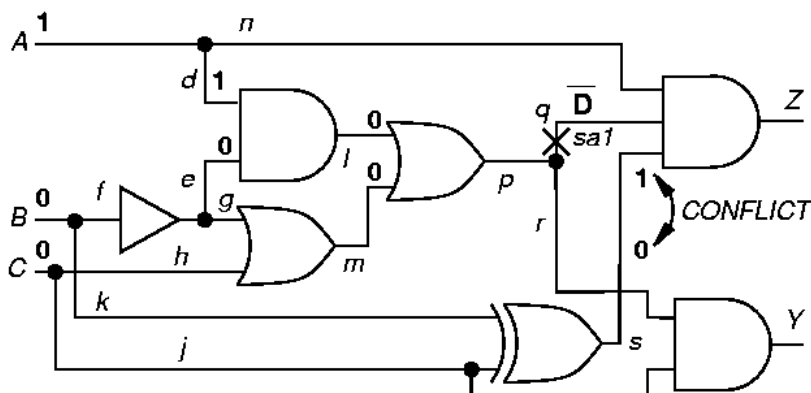


ECE 538

Krish Chakrabarty

37

Redundant Fault Example



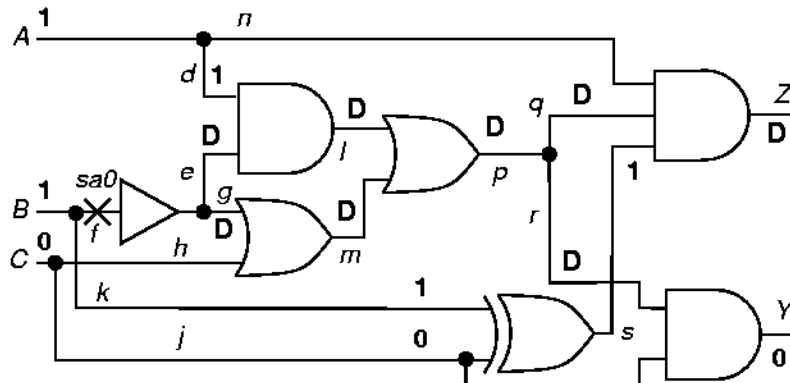
ECE 538

Krish Chakrabarty

38

Multiple Fault Masking

- f sa0 tested when fault q sa1 not there



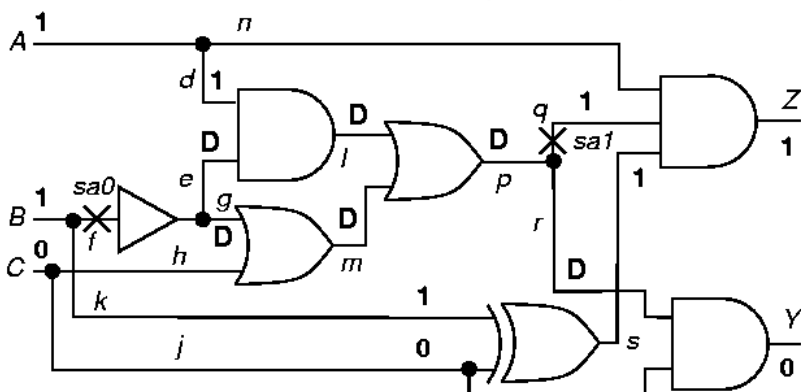
ECE 538

Krish Chakrabarty

39

Multiple Fault Masking

- f sa0 masked when fault q sa1 also present



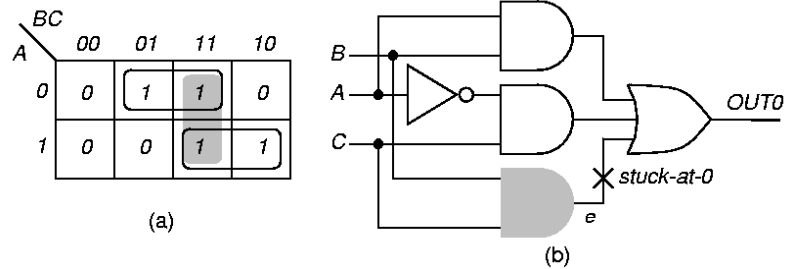
ECE 538

Krish Chakrabarty

40

Intentional Redundant Implicant BC

- Eliminates hazards in circuit output



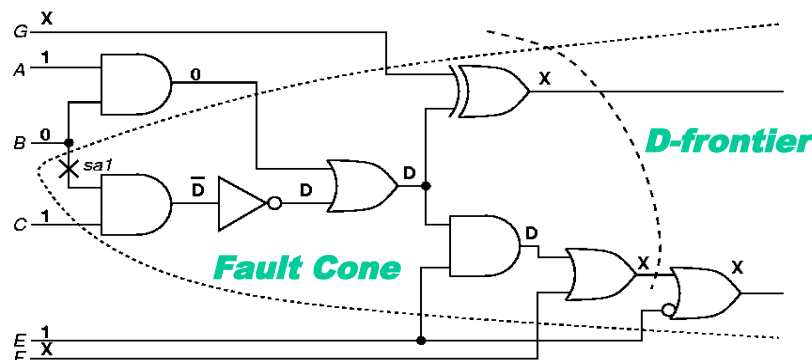
ECE 538

Krish Chakrabarty

41

Fault Cone and D-Frontier

- Fault Cone* -- Set of hardware affected by fault
- D-frontier* -- Set of gates closest to POs with fault effect(s) at input(s)



ECE 538

Krish Chakrabarty

42

Redundancy Removal

Repeat until there are no more redundant faults:

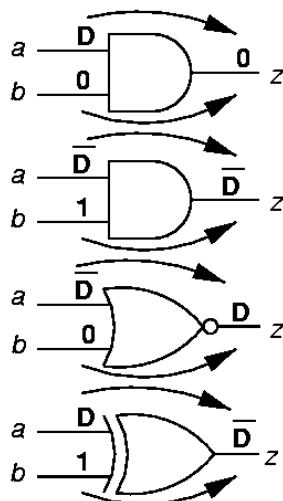
- {
- Use ATPG to find all redundant faults;
- Remove all redundant faults with non-overlapping fault cone areas;
- }

ECE 538

Krish Chakrabarty

43

Forward Implication



- Results in logic gate inputs that are significantly labeled so that output is uniquely determined
- AND gate forward implication table:

$a \backslash b$	0	1	X	D	\bar{D}
0	0	0	0	0	0
1	0	1	X	D	\bar{D}
X	0	X	X	X	X
D	0	D	X	D	0
\bar{D}	0	\bar{D}	X	0	\bar{D}

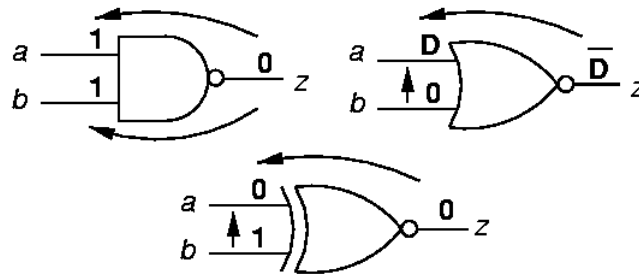
ECE 538

Krish Chakrabarty

44

Backward Implication

- Unique determination of all gate inputs when the gate output and some of the inputs are given



ECE 538

Krish Chakrabarty

45

Implication Stack

- Push-down stack. Records:
 - Each signal set in circuit by ATPG
 - Whether alternate signal value already tried
 - Portion of binary search tree already searched

Signal	Value	Alternative tried
A	1	NO
C	1	NO
E	1	NO
B	0	YES

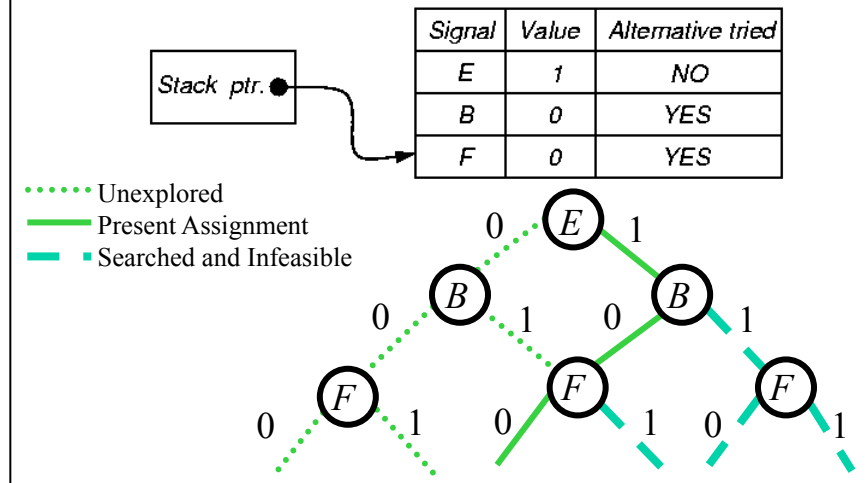
Stack ptr. ● →

ECE 538

Krish Chakrabarty

46

Implication Stack after Backtrack

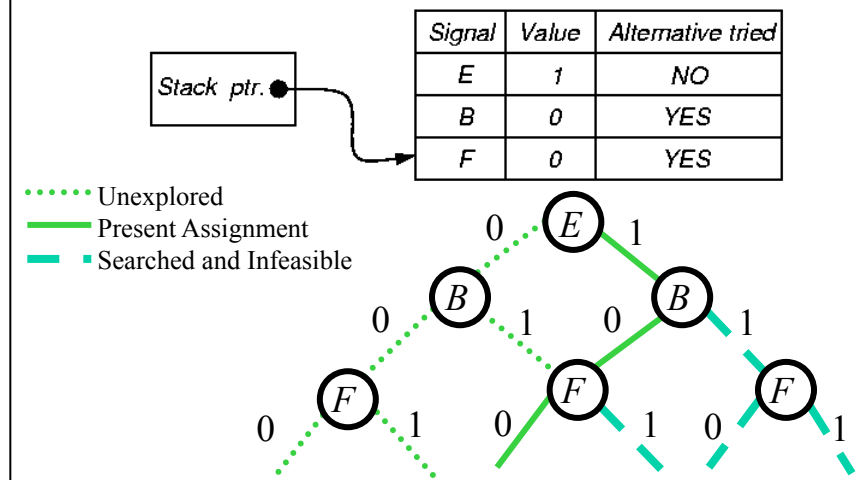


ECE 538

Krish Chakrabarty

47

Implication Stack after Backtrack



ECE 538

Krish Chakrabarty

48