*ECE 538*

# VLSI System Testing

Krish Chakrabarty

Test Generation: Part 1



## Introduction

- Classification of test generation methods
- Fault table analysis
- Boolean difference method
- Propagation, implication and justification procedures
- D-algorithm
- 9-V algorithm (multiple path sensitization)

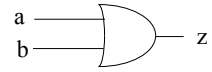
# General Approaches

- Systematic (algorithmic)
  - Fault table analysis
  - Boolean difference method
  - Fault-oriented methods (D-algorithm, PODEM)
  - Fault-independent methods (critical path tracing)
- Exhaustive/pseudoexhaustive
- Random/pseudorandom



# Fault Table Analysis

2-input OR gate




| Test | Fault: a/0 | a/1 | b/0 | b/1 | z/0 | z/1 |
|---|---|---|---|---|---|---|
| 00 | | x | | x | | x |
| 01 | | | x | | x | |
| 10 | x | | | | x | |
| 11 | | | | | x | |

- Determine fault table via simulation and solve covering problem
- Covering problems are NP-complete
- Computationally infeasible

# Fault-Oriented ATPG

```
Procedure Generate_Tests()
begin
 repeat
  begin
    Select uncovered fault f
    Generate test for f
    Evaluate current fault coverage
  end
 until fault coverage > limit, or time runs out
end
```



# Boolean Difference

The Boolean difference of $F(x_1, x_2, x_3, \ldots, x_n)$ with respect to $x_i$ is given by

$$\frac{\partial F}{\partial x_i} = F(x_1, x_2, \ldots, 1, x_3, \ldots, x_n) \oplus F(x_1, x_2, \ldots, 0, x_3, \ldots, x_n)$$

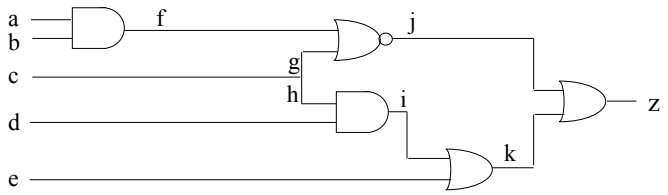
Cofactor w.r.t. $x_i$ Cofactor w.r.t. $\overline{x_i}$

Boolean difference provides input combination for sensitized path

A test pattern for $x_i/0$ is an input combination that makes $x_i \frac{\partial F}{\partial x_i} = 1$

A test pattern for $x_i/1$ is an input combination that makes $\overline{x}_i \frac{\partial F}{\partial x_i} = 1$

# Boolean Difference




To find a test for a/0 or a/1, determine the Boolean difference of output z with respect to a:

$$\frac{\partial z}{\partial a} = ?$$

How to handle internal faults, e.g. j/0?



# Key Terminology

- *Backtrace*: move a goal value backwards in a circuit to a primary input
- *Backtrack*: return to a previous decision point in an algorithm and make an alternative decision
- *D-frontier*: Set of gates closest to primary output with D or $\bar{D}$ on some input
- *Implication*: Determine unique signal values that are forced by signal values already assigned
- *Justification*: Determine values to unspecified inputs of gates whose outputs are specified (backward)
- *Propagation* (D-drive): Determine path values needed to propagate an error signal to a primary output (forward)

# Testing Fanout-Free Circuits

- No (reconvergent) fanout
  - Propagation path from any line is unique
  - Each line justification problem is independent of all others

Test generation for line *l*/*v* in a fanout-free circuit

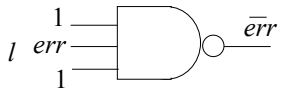
**begin**
 set all values to X (unknown)
 *Justify*($l,\bar{v}$)
 if $v = 0$ then *Propagate*($l,D$)
 **else** *Propagate*($l,\bar{D}$)
**end**



# Testing Fanout-Free Circuits

Error propagation:

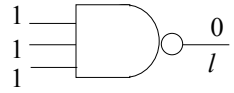
*Propagate*($l,err$)
/* *err* is D or $\bar{D}$ */
**begin**
 set *l* to *err*
 if *l* is PO **then return**
 *k* = the fanout gate of *l*
 *c* = controlling value of *k*
 *i* = inversion of *k*
 for every input *j* of *k* other than *l*
  *Justify*($j,\bar{c}$)
 *Propagate*($k,err \oplus i$)
**end**

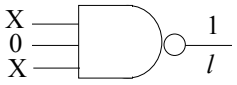


# Testing Fanout-Free Circuits

```
Justify(l,val)
begin
  set l to val
  if l is PI then return
/* l is a gate (output) */
  c = controlling value of l
  i = inversion of l
  inval = val ⊕ i
  if(inval = c̄)
    then for every input j of l
      Justify(j,inval)
    else
      begin
       Select one input (j) of l
       Justify(j,inval)
end
```

Line Justification






# Functional vs Structural Testing

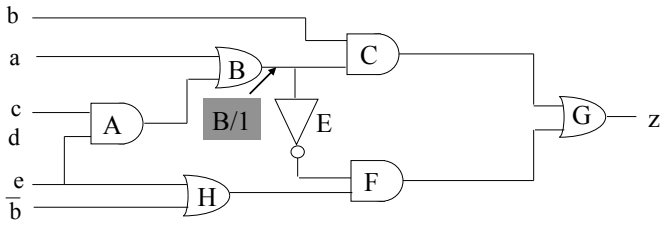
- Functional ATPG – generate complete set of tests for circuit input-output combinations
  - 129 inputs, 65 outputs:
  - $2^{129}$ patterns
  - Using 1 GHz ATE, would take 2.15 x $10^{22}$ years
- Structural test:
  - No redundant adder hardware, 64 bit slices
  - Each with 27 faults (using fault equivalence)
  - 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 $98^{+}$ %

# Algorithm Completeness

- Definition: Algorithm is c*omplete* if it ultimately can search entire binary decision tree, as needed, to generate a test
- *Untestable fault* – no test for it even after entire tree searched
- Combinational circuits only – untestable faults are *redundant,* showing the presence of unnecessary hardware



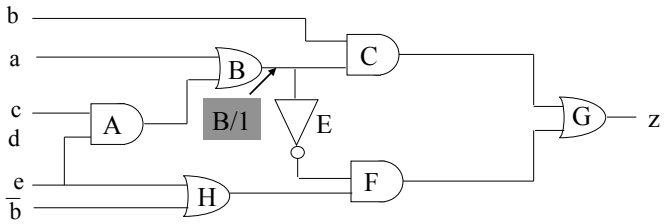
# D-Algorithm




| Decision | Implication | Comment |
|---|---|---|
| $B = \overline{D}$ | $E = \overline{D}$, A = 0, a = 0 | Activate fault |
| b = 1 | $C = \overline{D}$ | Propagate via C |
| F = 0 | z = D | End of D-drive |
| H = 0 | e = 0 | Justify F |
| c = 0 (1) | | Justify A |

*Test: abce = 0100*

# D-Algorithm

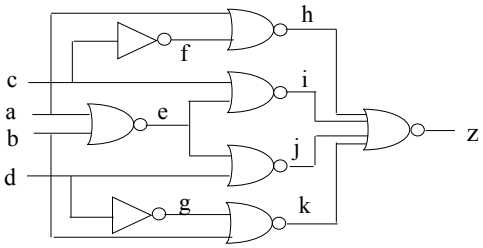



| Decision | Implication | Comment |
|---|---|---|
| B = $\overline{D}$ | E = $\overline{D}$, A = 0, a = 0 | Activate fault |
| H = 1 | F = $\overline{D}$ | Propagate via F |
| C = 0 | z = D | End of D-drive |
| b = 0 | e = 0 | Justify C |
| c = 0 or e = 0 | | Justify A |
| | *Test: abce = 0000* | |



# D-Algorithm

Fault e/0




Single path sensitization does not always work!

| Decision | Implication | Action |
|---|---|---|
| e = D, c = 0 | i = $\overline{D}$, f =1, h = 0 | Activate fault, propagate via i |
| j = k = 0 | z = D | End of D-drive |
| d = 1, b = 1 | g = 0, e = 0 *Contradiction* | Justify j = 0, k = 0 *(Backtrack)* |
| c = 0, d = 0 | i = j = $\overline{D}$ | propagate through i and j |

# Comments on the D-Algorithm

- Based on propagation (primary procedure), justification and implication
- In complete form, guarantees test generation but may require multiple path sensitization (computationally expensive)
- Practical restrictions:
  - Single path sensitization only
  - Limits placed on backtracking time (aborted faults)

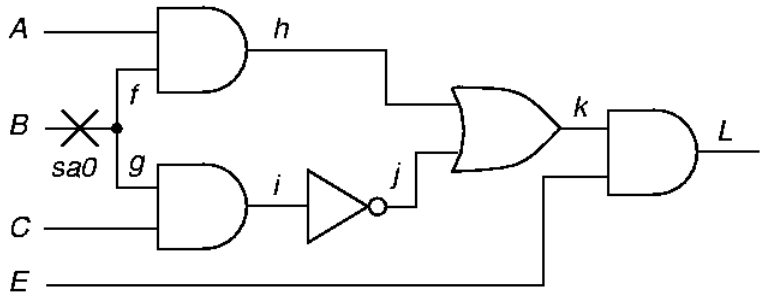


# 9-V Algorithm

- Nine logic values, specified as good/bad pairs: 1/1, 0/0, 1/0, 0/1, u/u, 1/u, 0/u, u/0, u/1
- Example of logic operations: D.X = 1/0.u/u = u/0, i.e. provides more information than the X outcome for 5-valued algebra
- Reduces backtracking
  - When there are k possible paths for error propagation, D-algorithm may try all $2^k$-1 combinations of paths
  - 9-V enumerates only k ways of propagation

# Path Sensitization

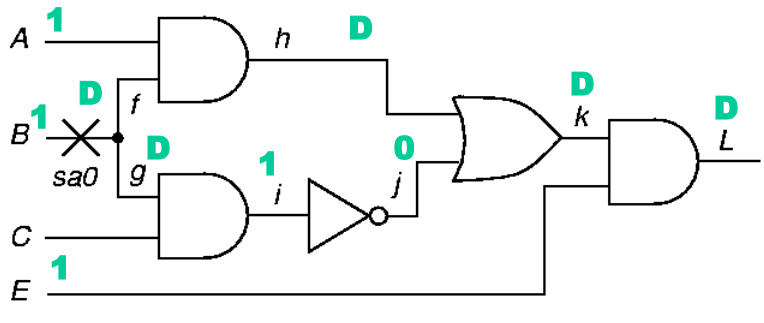
1 **Fault Sensitization**
2 **Fault Propagation**
3 **Line Justification**





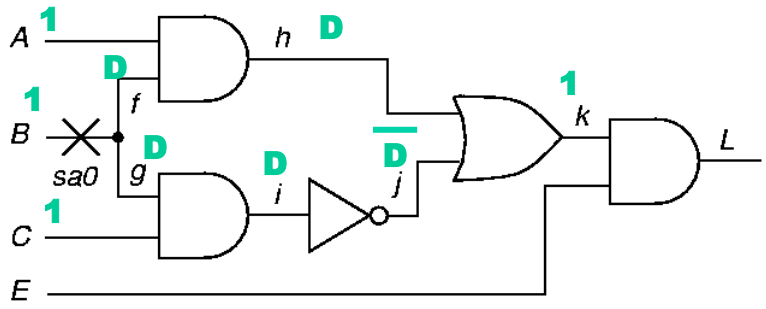

# Path Sensitization

- **Try path *f* – *h* – *k* – *L* blocked at *j*, since there is no way to justify the 1 on *i***



# Path Sensitization

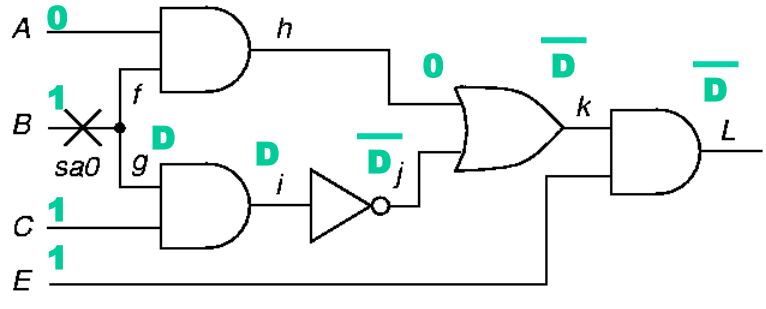
- Try simultaneous paths $f - h - k - L$ and $g - i - j - k - L$ blocked at $k$ because *D-frontier* (chain of $D$ or $\overline{D}$) disappears






# Path Sensitization

- **Final try: path $g - i - j - k - L$ – test found!**



# Random Test Generation

- Flow chart for method
- Use to get tests for 60-80% of faults, then switch to D-algorithm or other ATPG for rest