# Gate-level Testing. Detection of SSF's

## Test Tables

**Example 1:** AND gate

| $x_1$ | $x_2$ | $f(x_1,x_2)$ | $\hat{f}(x_1,x_2)$ $x_1/0$ | $x_1/1$ | $x_2/0$ | $x_2/1$ | $y/0$ | $y/1$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

**Test Table:**

| $x_1$ | $x_2$ | $x_1/0$ | $x_1/1$ | $x_2/0$ | $x_2/1$ | $y/0$ | $y/1$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |

- Optimal test $T = \{01, 10, 11\}$ in every column at least one 1

$x_1/0$ $x_2/0$ $y/0$ have the same column ⇒ cannot be locat

## Example 2

OR gate

Test table:

| $x_1$ | $x_2$ | $x_1/0$ | $x_1/1$ | $x_2/0$ | $x_2/1$ | $y/0$ | $y/1$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

Optimal test: $T = \{00, 01, 10\}$

For m-input AND gate: $T = \{11...11, 01...11, ... 11...10\}$

For m-input OR gate: $T = \{00...00, 10...00, ... 00...01\}$

Testing of m-input AND or OR gates: $|T| = m+$
(Linear test complexity)

**Example 3** XOR gate

**Test Table:**

| $x_1$ | $x_2$ | $x_1/0$ | $x_1/1$ | $x_2/0$ | $x_2/1$ | $y/0$ | $y/1$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

**Test** : $T = \{00, 01, 11\}$

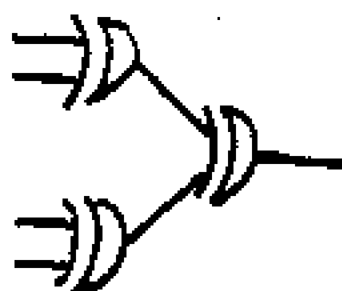
For $m$ input XOR gate $T = \{00...0, 11...1, 00...01\}$

The number of test patterns $|T| = 3$ does **not** depend on a number of inputs $m$.

# Testing Internal Faults -

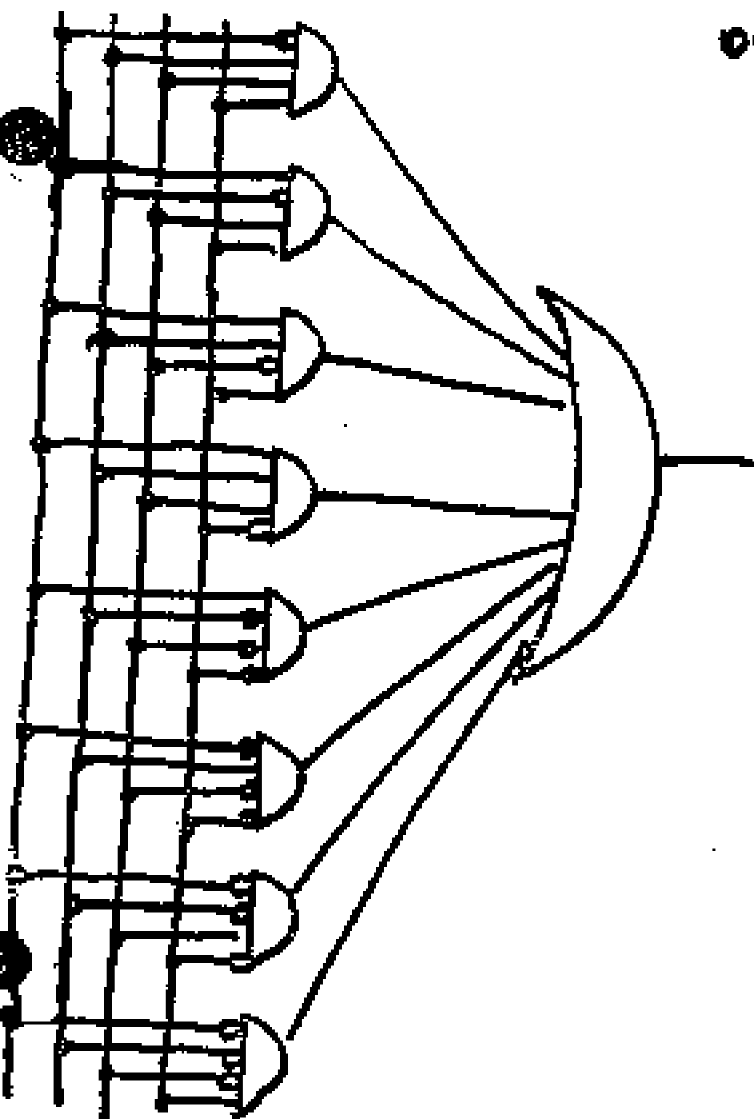
- Dependence on an implementation

Example - parity checkers



$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \Rightarrow$ 3 test patterns:

0000
1101
0111



$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \Rightarrow$

9 test patterns:

0001
0010
0100
1000
1110
1101
1011
0111
0000

For a m-bit binary tree

$$y = x_1 + x_2 + \ldots + x_m$$

And single stuck-at faults

| IMPLEMENTATION | Min. # of test patterns |
|---|---|
| Tree of XOR Gates | 3 |
| Two-Level AND-OR Network | $2^{m-1} + 1$ |

## Generation of a Minimal Test

choose a minimal number of rows in the test table containing at least one 1 in every column

Covering problem ⇒ NP-complete

Good practical solutions for small networks by branch and bounds method

## Diagnostic (Fault Location) for Combinational Networks

Even more difficult than fault detection - mostly for manufacturing (not field) test

**Test table:** $(t_{ij}) = 1$ iff test pattern $t_i$ detects $j$th faults

**Fault Detection:** min number of rows such that every column contains at least one 1

**Fault Location** min number of different nonzero columns

# Lower Bounds for a Number of Test Patterns for Location of Stuck-at Faults

$$|T| \geq \left\lceil \log_2 \sum_{i=0}^{\ell} 2^i \binom{L}{i} \right\rceil$$

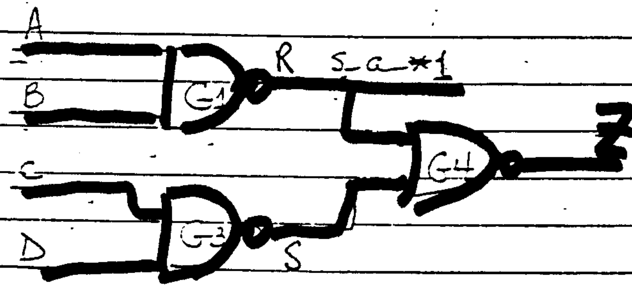
$|T|$ is a number of test patt

$L$ is a number of lines

$\ell$ is a multiplicity of faul

Example $L = 1{,}000$, $\ell = 2 \Rightarrow |T| \geq 2$

# PATH SENSITIZATION




1. SPECIFY INPUTS $X_i$ TO PROVOKE
   ~~FAULTY CONDITION~~ A=B=1

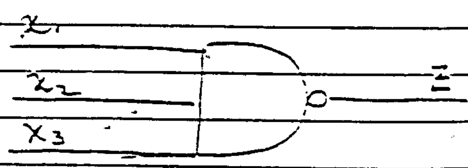
2. ~~SELECT PATH FROM FAULT TO PO~~
   (ERROR PROPAGATION) R-$G_4$-Z

3. SENSITIZE PATH FROM FAULT TO PO
   (LINE JUSTIFICATION)
   S=0 ⇒ C=D=1

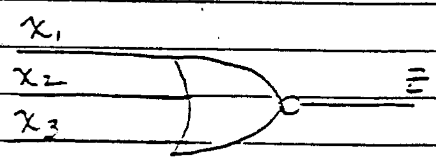
Test pattern ABCD = (1 1 1 1)

# ERROR PROPAGATION

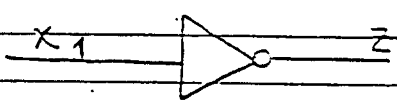
$x_1 \mapsto z$

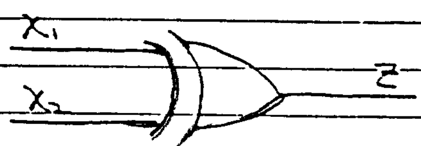



$\Rightarrow X_2 = x_3 = 1$

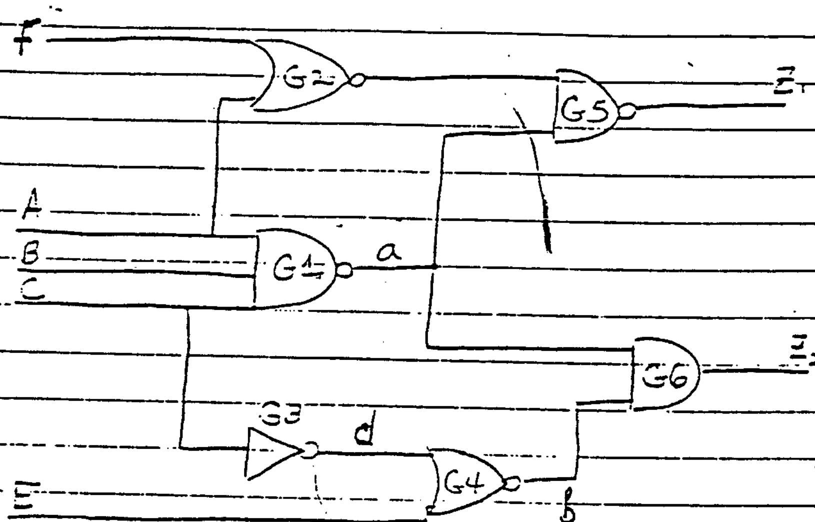



$\Rightarrow x_2 = x_3 = 0$




$\Rightarrow x_2 = *$




Consider: a s-a-0 a/0

1. PROVOKE FAULT
2. PROPAGATE "FAULT EFFECT" } PATH SENSITIZATION
3. LINE JUSTIFICATION }

1. $A \cdot B \cdot C = 0 \Rightarrow A = 0$ or $B = 0$ or $C = 0$
2. Path $a - G6 - Z_2$
3. $b = 1 \Rightarrow E = d = 0 \Rightarrow C = 1$

test patterns:

A B C E F
0 * 1 0 *
* 0 1 0 *

0 * 1 0 * = 00100
01100
00101
01101

* 0 1 0 * = 00100
00101
10100
10101

## Gate-Level Testing. Conclusions

1. SSF model is the most popular

2. Gate-level testing is very complicate even for SSF for VLSI. (test generation time $\sim 2^N$ in the worst case and $\sim N^3$ on average)

3. Most existing algorithms cannot generate a (even not optimal!) test for $N > 2 \cdot 10^4$

4. For complex devices ($\mu$processors) fault-coverage of (95-96)% is sufficient for practical purposes

5. Simulation to estimate fault-coverages
   Software for simulation expensive
   Simulation is slow. (About 50 hours for a chip with $N = 10^5$)

6. Very difficult to generate a test for sequential networks