LaDS
Laboratory of Dependable Systems

Fault Modeling

Jiun-Lang Huang

GIEE/ICDA, National Taiwan University

A Recap of the Single-Stuck-At Fault Model

Stuck-At Faults

- Circuit model

  An interconnection (called a *netlist*) of Boolean gates.

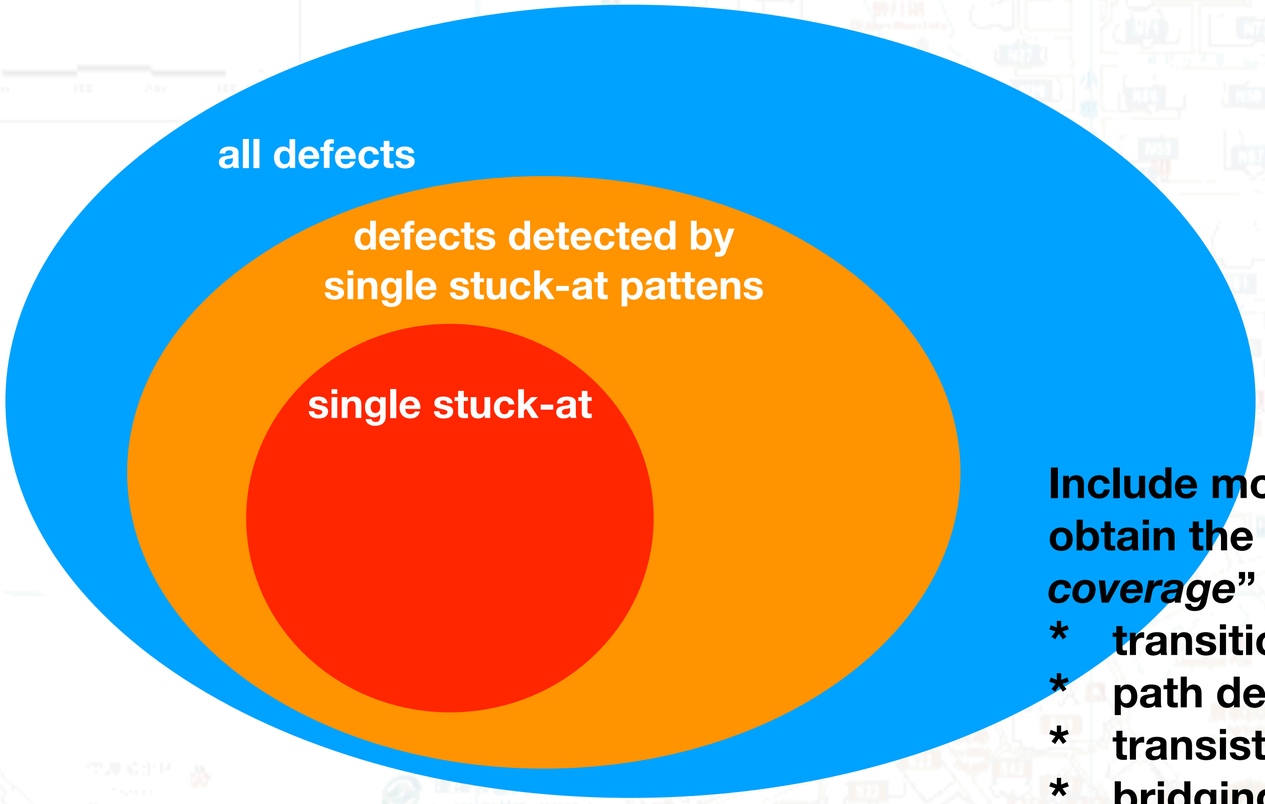
- Definition

  A stuck-at fault is assumed to affect only the interconnections between gates (functionally correct). The faulty line is permanently set to 0 or 1 (stuck-at-0 or stuck-at-1).

- A circuit with n lines has a total of $3^n - 1$ stuck-at faults!

Single-Stuck-At Fault

- Properties
  - Only one line is faulty.
  - The faulty line is permanently set to either 0 or 1.
  - The fault can be at an input or output of a gate.
- An n-line circuit has at most $2n$ single stuck-at faults.
- The classical fault model — the most widely used and studied.



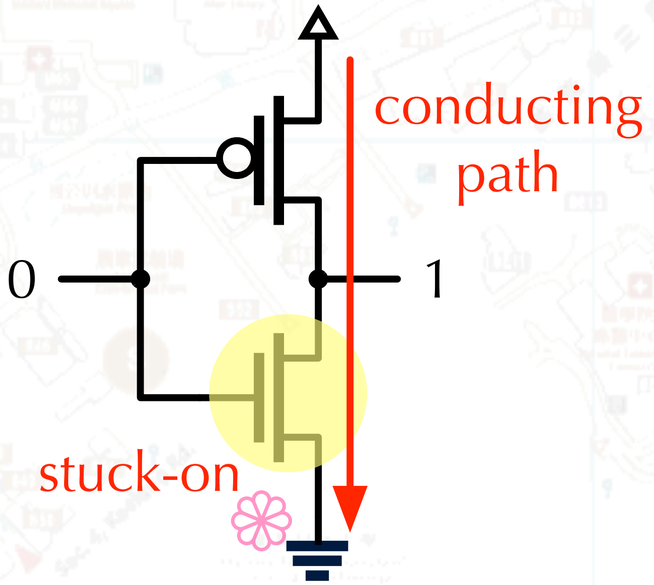
Include more fault models to obtain the desired "*defect coverage*"

- **transition delay fault**
- **path delay fault**
- **transistor stuck-on**
- **bridging**
- **…**

Other Fault Models

CMOS Transistor Stuck-On

- The effect is a possible conducting path between V_{dd} and the ground.
  - The logic state at the output depends on the relative impedance of the transistors.
  - Detected by quiescent current (IDDQ) measurement.
- Example
  - Setting the inverter input to 0 establishes a conducting path from V_{DD} to the output.
  - In the existence of NMOS stuck-on, there will be a conducting path between VDD and the ground.



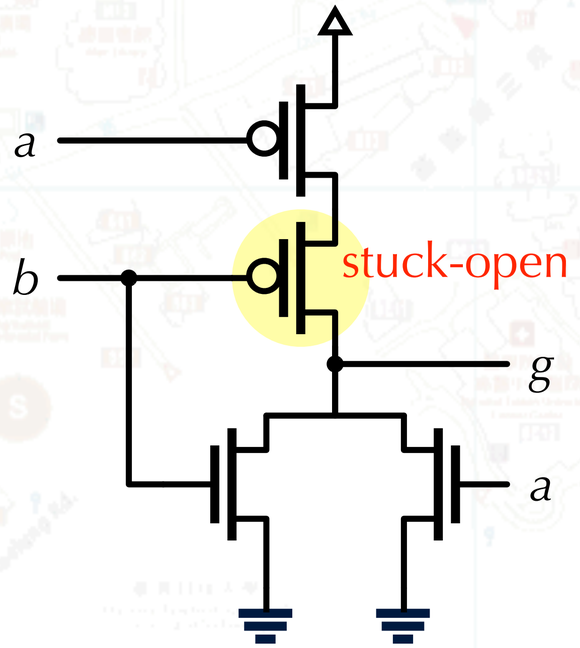
CMOS Transistor Stuck-Open

- The possible effect is a floating output value.
- Detection of stuck-open faults in CMOS circuits requires two-vector tests!
- Example:

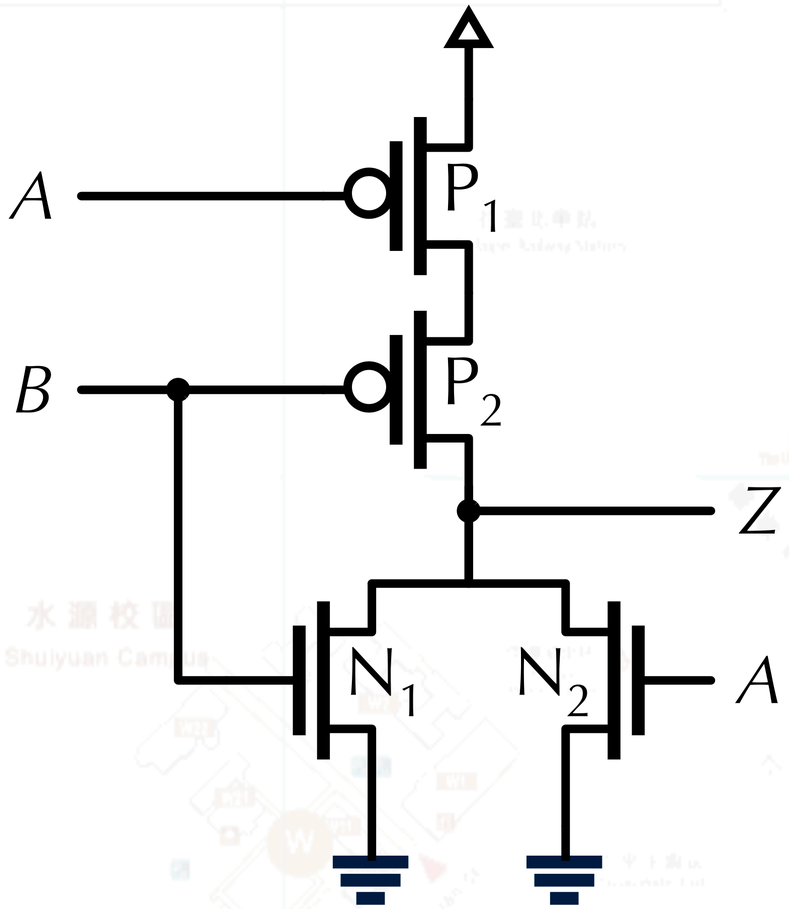
  With the indicated PMOS stuck-open, β, the path from g to V_{DD} cannot be established.

  β can be detected by applying $ab = 10$ followed by $ab = 00$.

  - Without β, g is 1.
  - With β, g is floating and remains 0.



Example of Transistor Fault Detection



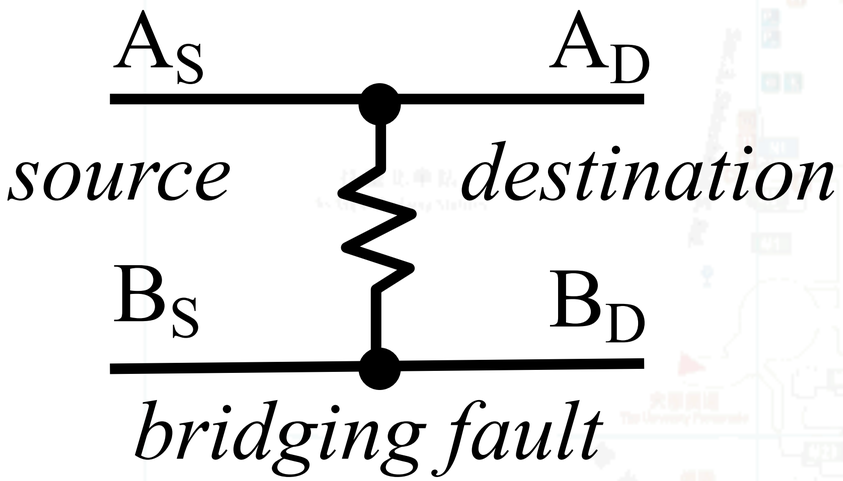
Truth Table for Fault-Free and Faulty NOR

AB	00	01	10	11
Z	1	0	0	0
N_1 stuck-open	1	0	last Z	0
N_1 stuck-short	I_{DDQ}	0	0	0
N_2 stuck-open	1	last Z	0	0
N_2 stuck-short	I_{DDQ}	0	0	0
P_1 stuck-open	last Z	0	0	0
P_1 stuck-short	1	0	I_{DDQ}	0
P_2 stuck-open	last Z	0	0	0
P_2 stuck-short	1	I_{DDQ}	0	0

[L.-T. Wang, C.-W.Wu, and X.Wen, *VLSI Test Principles and Architectures*, Morgan Kaufmann Publishers, 2006]

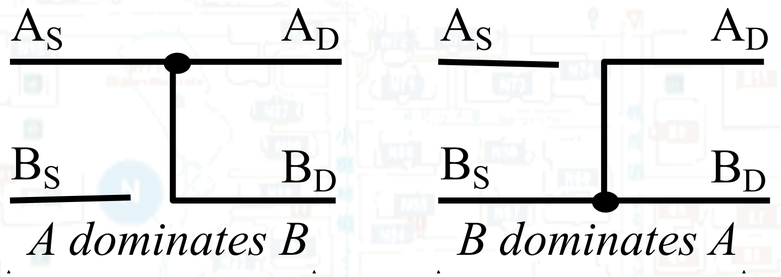
Bridging Faults

- Two or more normally distinct points (lines) are shorted together.

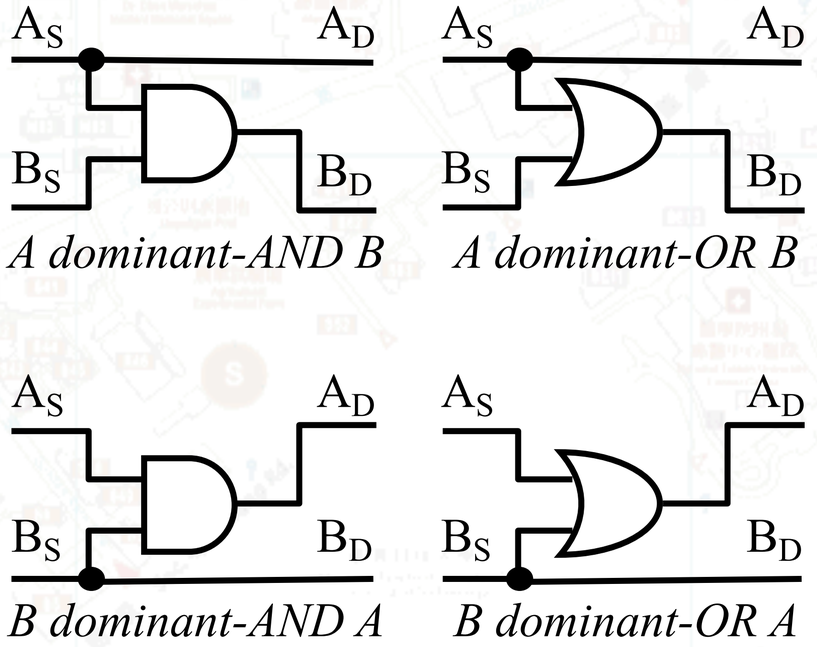


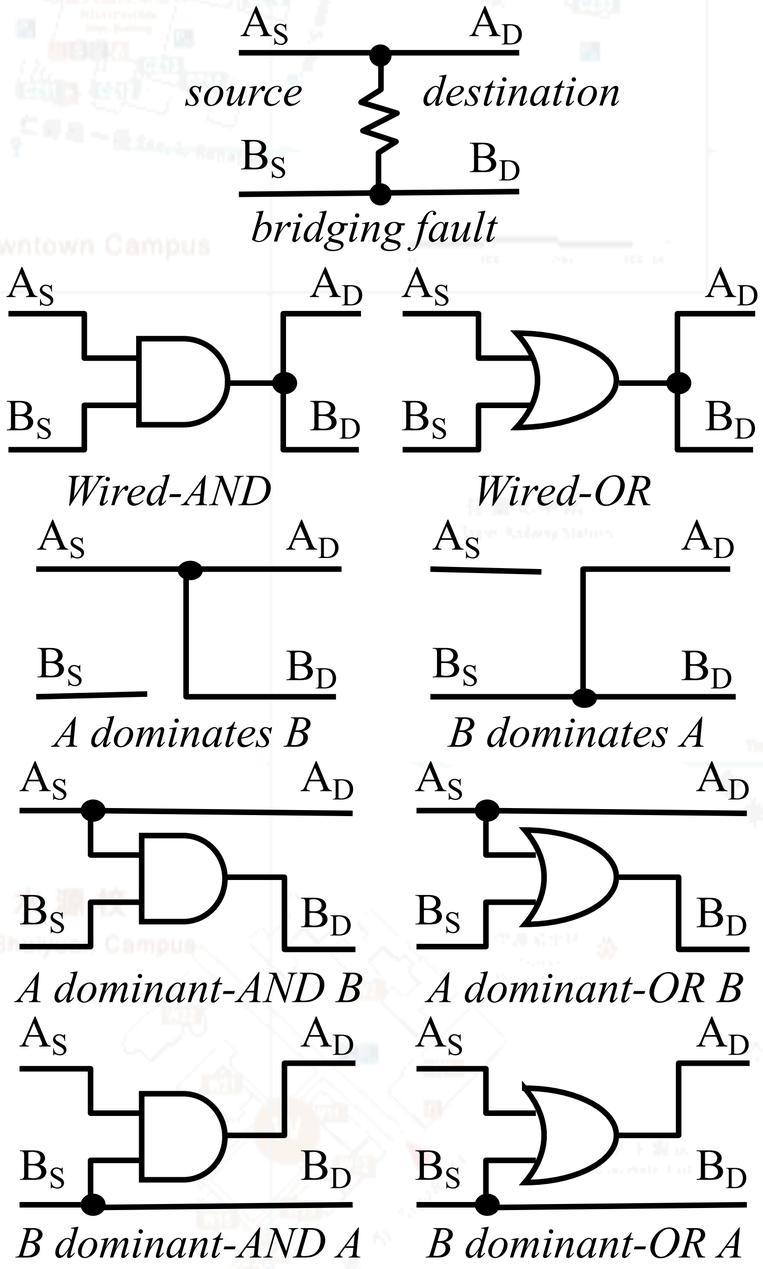
- The logic effect is technology-dependent.

- Dominant bridging fault
  - Proposed for CMOS logic.
  - One driver is assumed to dominate the logic value on the two shorted nets.
  - Two faults per fault site.

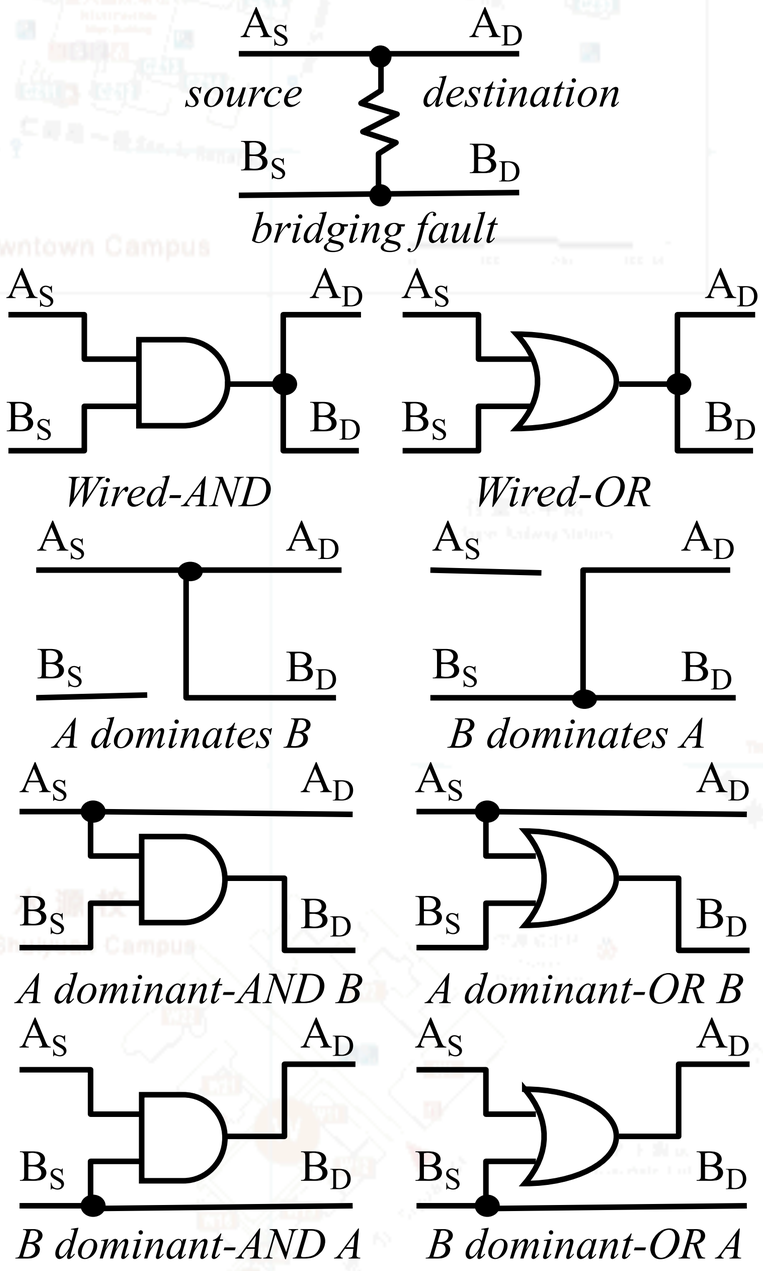


- Dominant-AND/Dominant-OR bridging fault
  - To complement the dominant bridging fault model.
  - One driver dominates the logic value only for a given logic value.
  - Four faults per fault site.





A_S B_S	0 0	0 1	1 0	1 1
A_D B_D	0 0	0 1	1 0	1 1
Wired-AND	0 0	0 0	0 0	1 1
Wired-OR	0 0	1 1	1 1	1 1
A dominates B	0 0	0 0	1 1	1 1
B dominates A	0 0	1 1	0 0	1 1
A dominant-AND B	0 0	0 0	1 0	1 1
B dominant-AND A	0 0	0 1	0 0	1 1
A dominant-OR B	0 0	0 1	1 1	1 1
B dominant-OR A	0 0	1 1	1 0	1 1



A_D B_D	A_S B_S							
	0	0	0	1	1	0	1	1
Fault free	0	0	0	1	1	0	1	1
Wired-AND	0	0	0	0	0	0	1	1
Wired-OR	0	0	1	1	1	1	1	1
A dominates B	0	0	0	0	1	1	1	1
B dominates A	0	0	1	1	0	0	1	1
A dominant-AND B	0	0	0	0	1	0	1	1
B dominant-AND A	0	0	0	1	0	0	1	1
A dominant-OR B	0	0	0	1	1	1	1	1
B dominant-OR A	0	0	1	1	1	0	1	1

Remarks on Bridging Faults

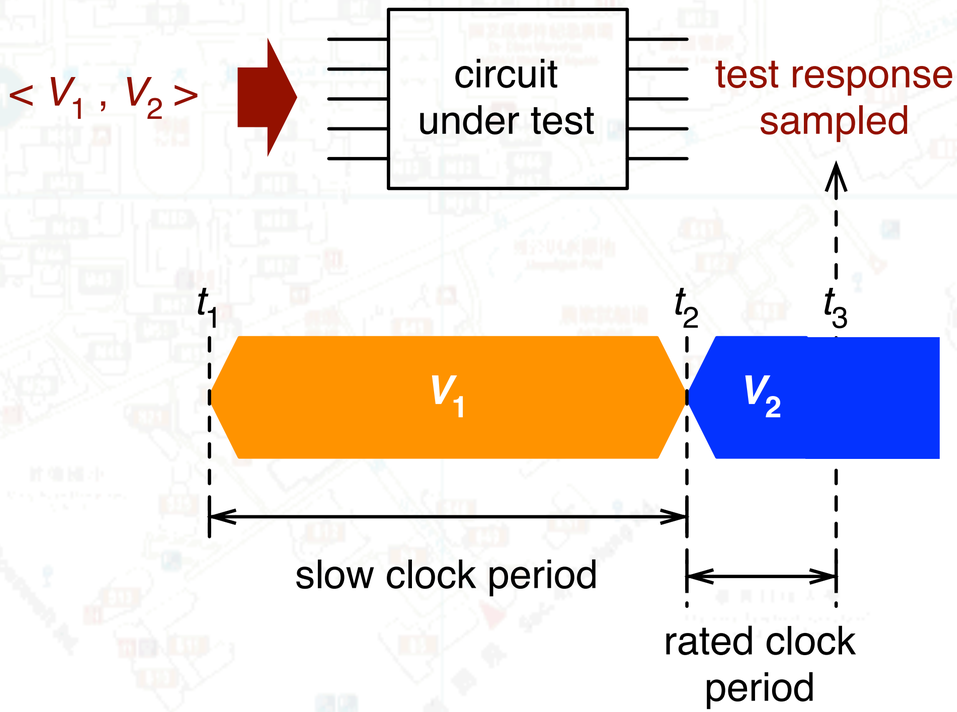
- Many bridging faults can be detected by I_{DDQ} testing.
- *N*-detect single stuck-at-fault test patterns are also effective.
- Bridging incurred feedback paths may lead to sequential behavior.
- Fault sites: Utilize physical design information to derive a more realistic list of fault sites.

Wire Open Fault

- Opens occurring in wires interconnecting transistors to form gates behave like transistor stuck-open faults.
- Opens occurring in wires interconnecting gates to form circuits behave like stuck-at faults.
- Wire opens are detected by vectors detecting transistor faults and stuck-at faults.

Delay Faults

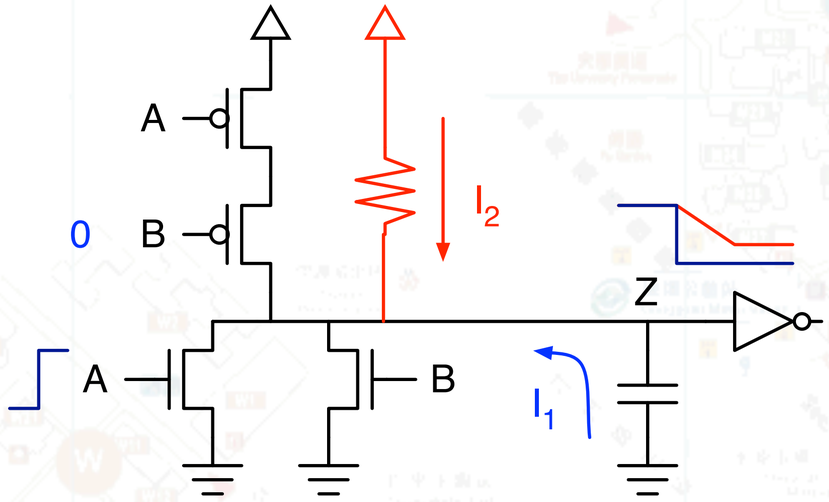
- Cause excessive delay along a path such that the total propagation delay falls outside the specified limit.
- Chips with delay faults may pass DC testing, e.g., stuck-at-fault testing, but fail when tested at speed.
- Delay fault testing has become mandatory.
- Delay fault testing generally requires two-vector tests.



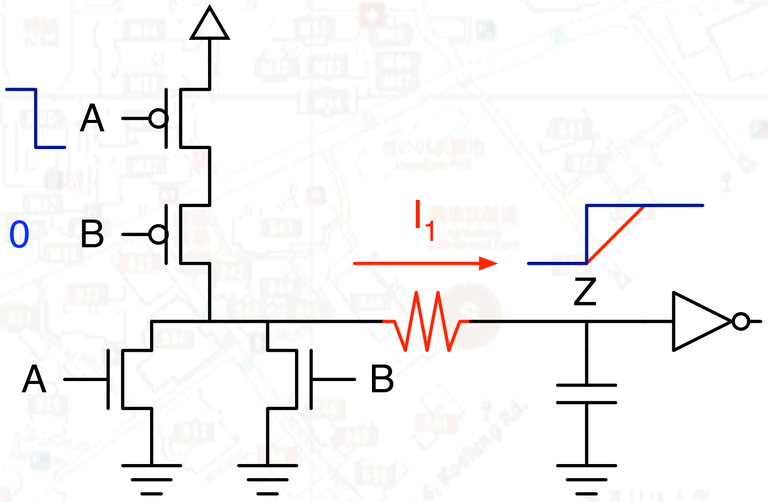
Causes of Delay Faults

- Manufacturing defects
  - Certain manufacturing defects do not change the logic function but can cause timing violations.

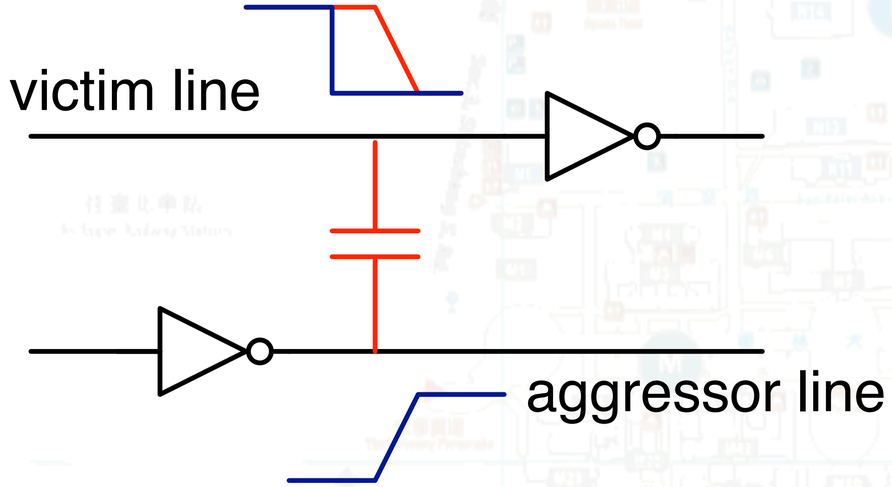
Resistive Bridges
0-to-1 on A delayed, 1-to-0 on A accelerated



Resistive Opens
both 0-to-1 & 1-to-0 on A delayed



- Design/manufacturing defects
  - Aggressive place & route



  - Abnormal statistical variations in geometry
  - Process variations

Need of Delay Fault Testing

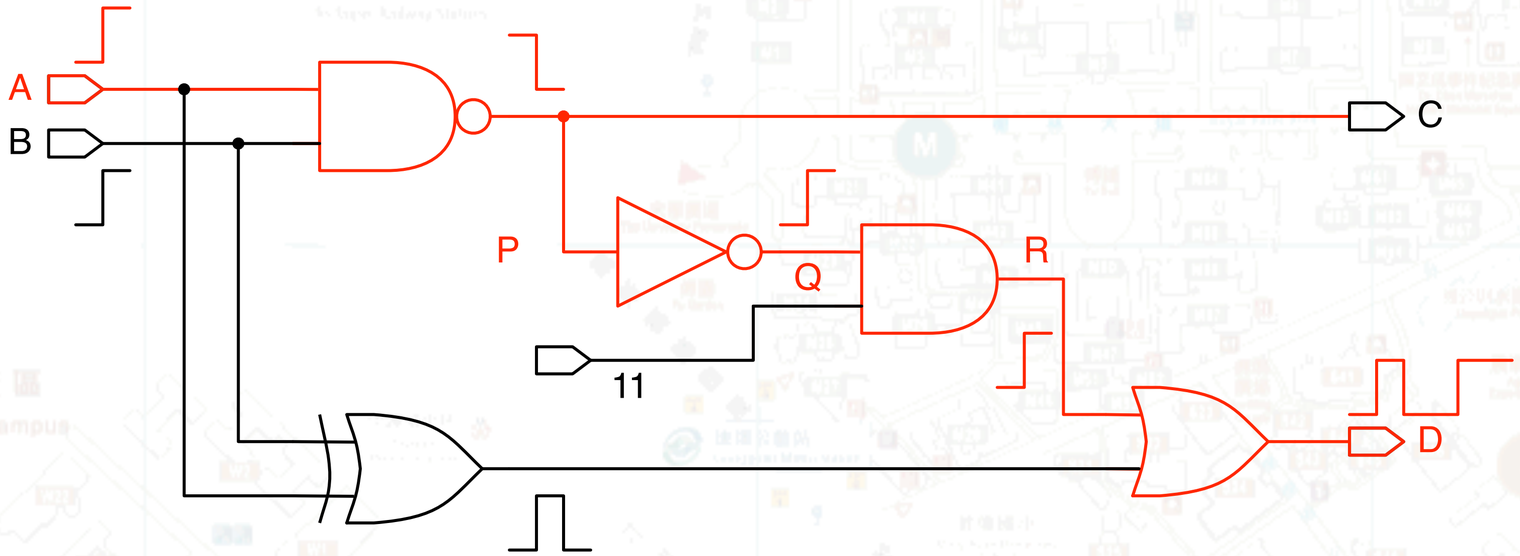
- Hard shorts and opens are testable by stuck-at tests with high confidence.
- Resistive shorts may be testable by stuck-at tests but more likely to be detected by delay tests.
- Resistive opens and coupling faults can only be detected by delay tests.
- Resistive power supply lines cause excessive IR-drop, which can be detected by delay tests.
- Process variations can only be detected by delay tests.

Path Delay Fault Model

- Assume a distributed delay along a combinational path from FF to FF.
  - A path is a sequence of connected gates from a PI (PPI) to a PO (PPO).
  - A path delay fault is said to have occurred if the delay of a path exceeds the specified clock period.
- Features
  - Models distributed delay defects
  - More likely to detect small delay defects
  - Much more complex than the transition delay model
  - Low fault coverage

Example

- Path A-P-Q-R-D is tested for rising transition at pin A.
- Small delay defects distributed along the path will be tested if the cumulative delay exceeds specification.

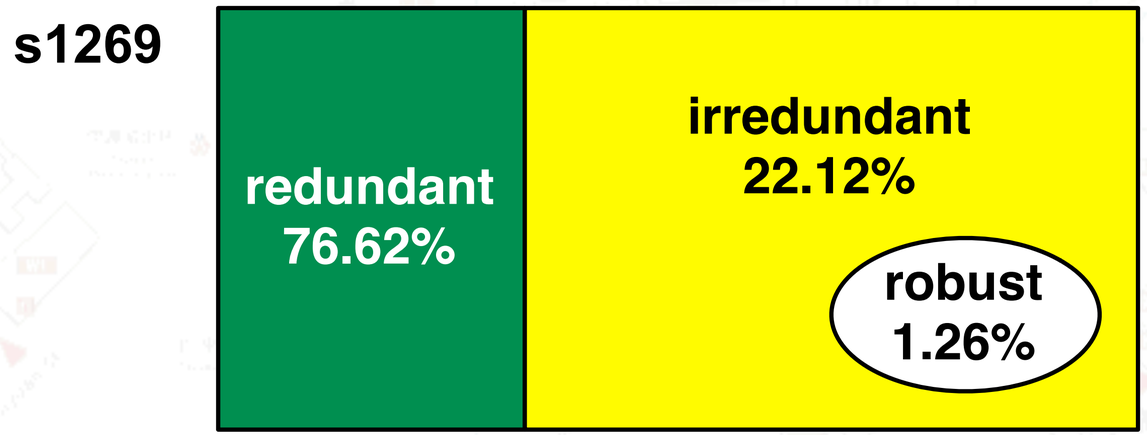


Robust and Non-robust Path Delay Tests

- A robust path delay test guarantees to detect the delay fault on the target path regardless of other delays in the circuit.
- A non-robust path test guarantees to detect the delay fault on the target path only if no other path delay is increased.
  - Many situations exist where a non-robust test is invalidated.

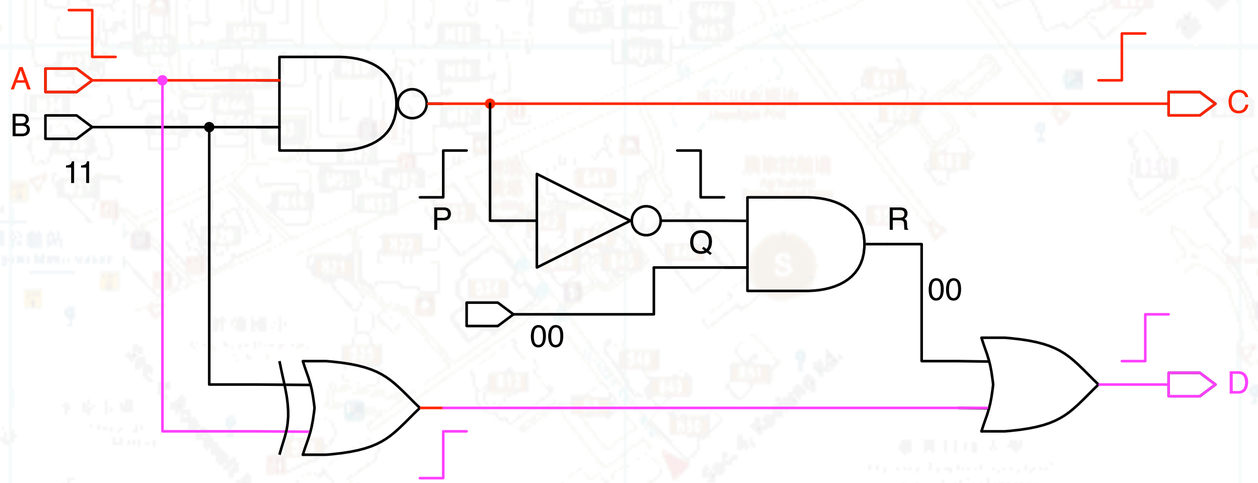
Limitations of Path Delay Fault Model

- The number of paths in a circuit can grow exponentially with the circuit size.
  - s38584: 3.6×10^4 stuck at faults, 2.2×10^6 path delay faults.
- The number of robustly testable paths in typical circuits is much less than the number of irredundant paths.

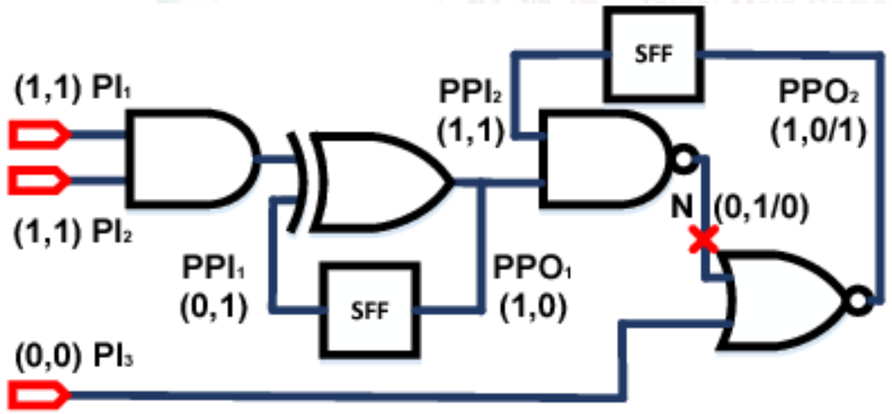


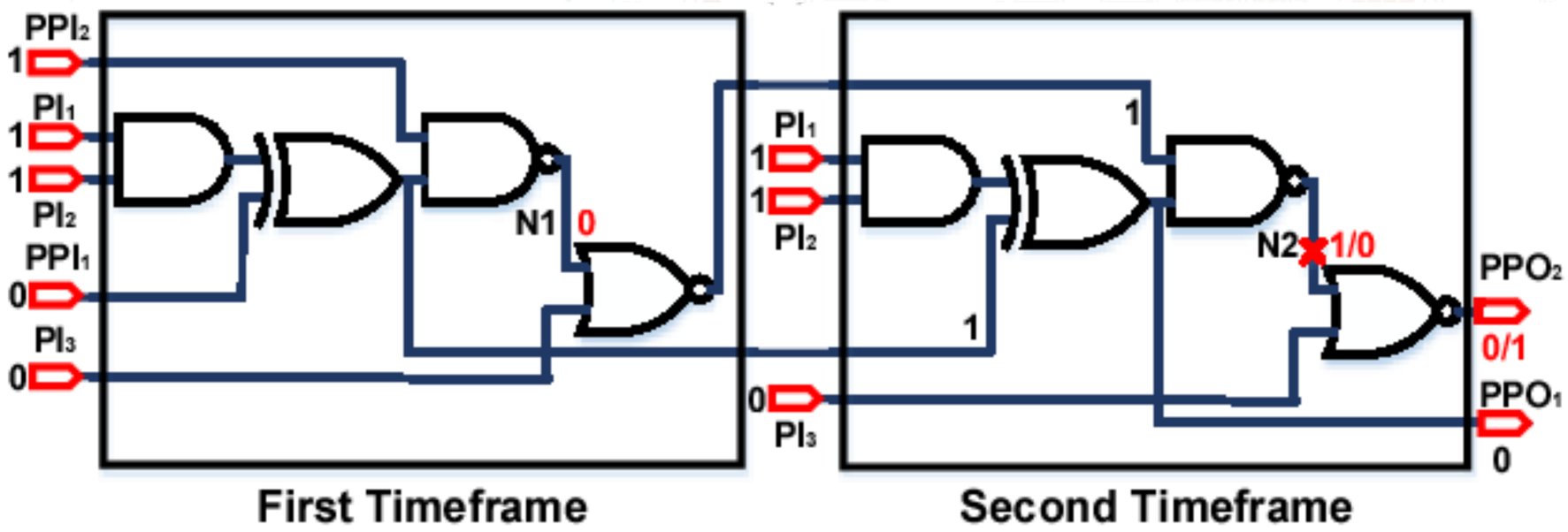
Transition Delay Fault Model

- Assume a large delay defect concentrated at one logical node.
- Any signal transition passing through this node will be delayed past the clock period.
  - Slow-to-rise or slow-to-fall
- Example:
  Slow-to-fall on A may be observed at C or D.



- Example: slow-to-rise at *N*
  - In the first time frame, set *N* to 0.
  - In the second time frame, test *N* stuck-at-0.



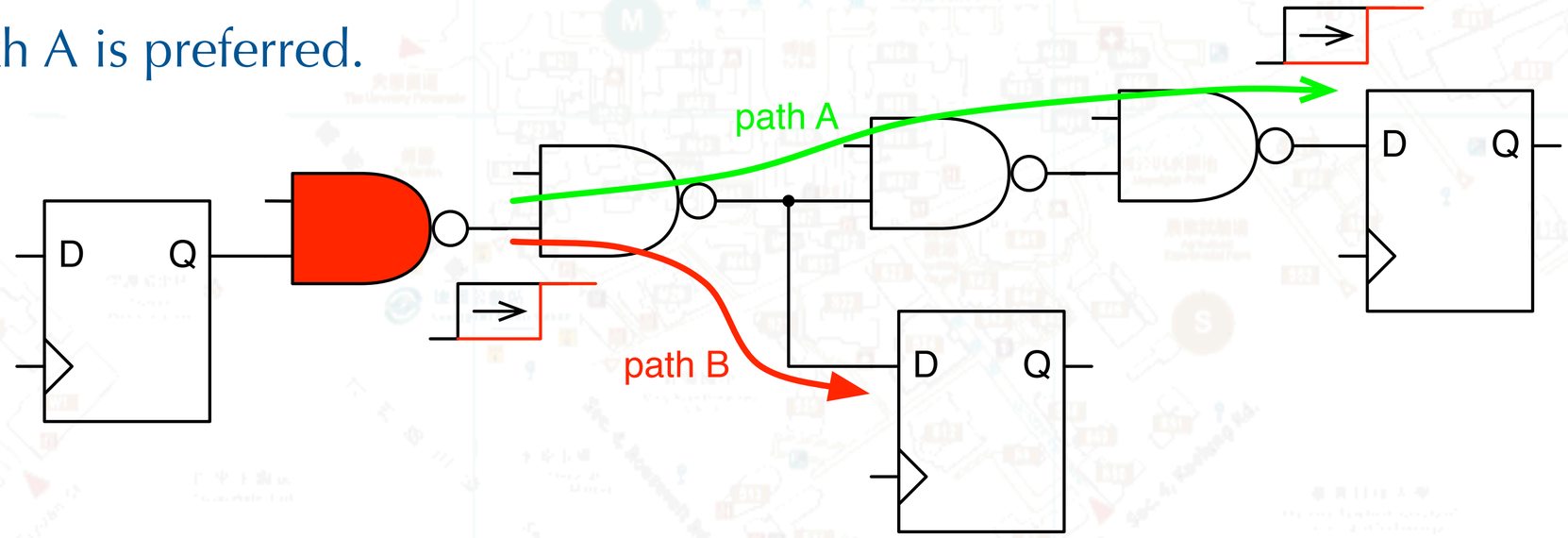


Pros and Cons of Transition Delay Fault Model

- Advantages
  - May detect delay defects missed by stuck-at tests.
  - Test generation with stuck-at-fault tools with minor modifications.
  - The fault lists and coverage metrics are similar to those of stuck-at faults.
- Disadvantages
  - May miss distributed small delay defects.

Small Delay Defects

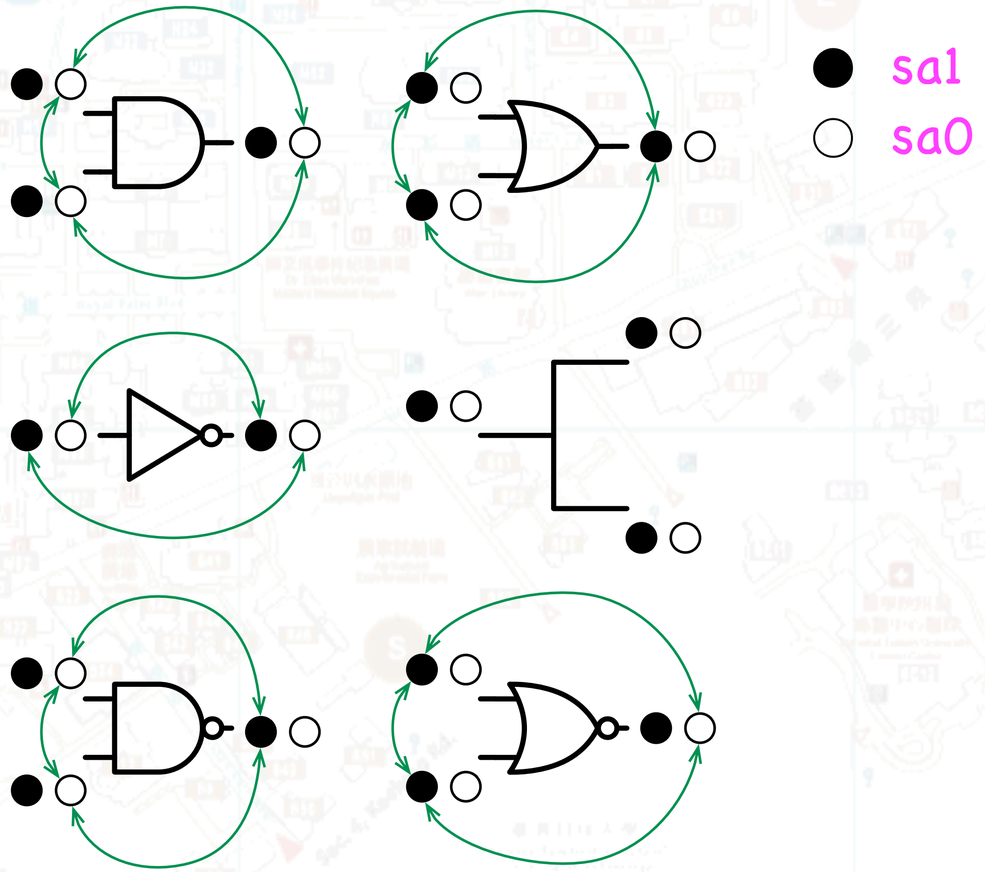
- A type of timing defect that introduces a small amount of extra delay to the design.
- Use a slack-based transition path selection to increase the detection probability.
  - Example: Path A is preferred.



Fault Collapsing

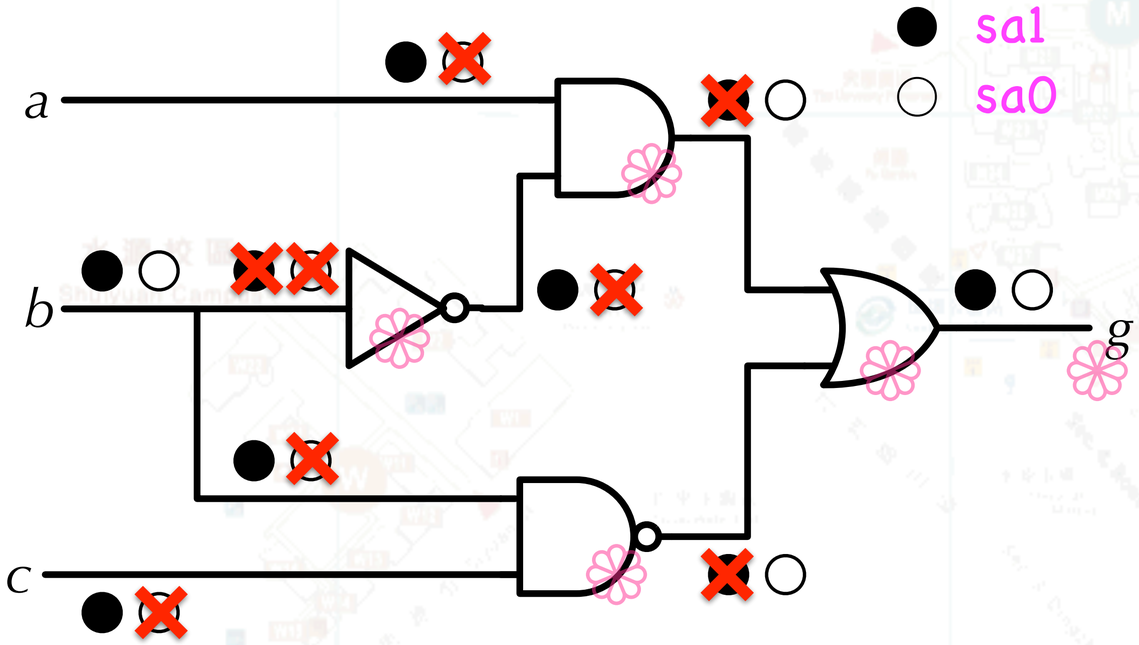
Fault Equivalence

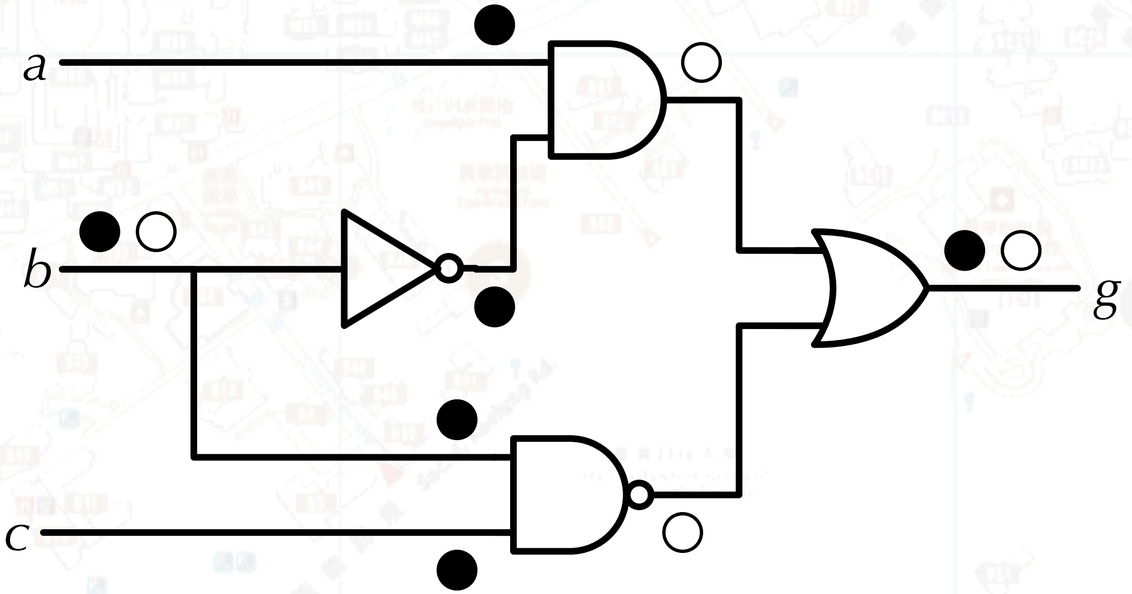
- Two faults α and β are equivalent iff $Z_\alpha(x) = Z_\beta(x)$.
  - Equivalent faults are indistinguishable.
- Equivalence of single stuck-at faults
  - n+2 faults, instead of 2(n+1), need to be considered for an n-input primitive gate.
  - Two faults for an inverter.
  - No equivalence between the fanout stem and branches.



Equivalence Fault Collapsing

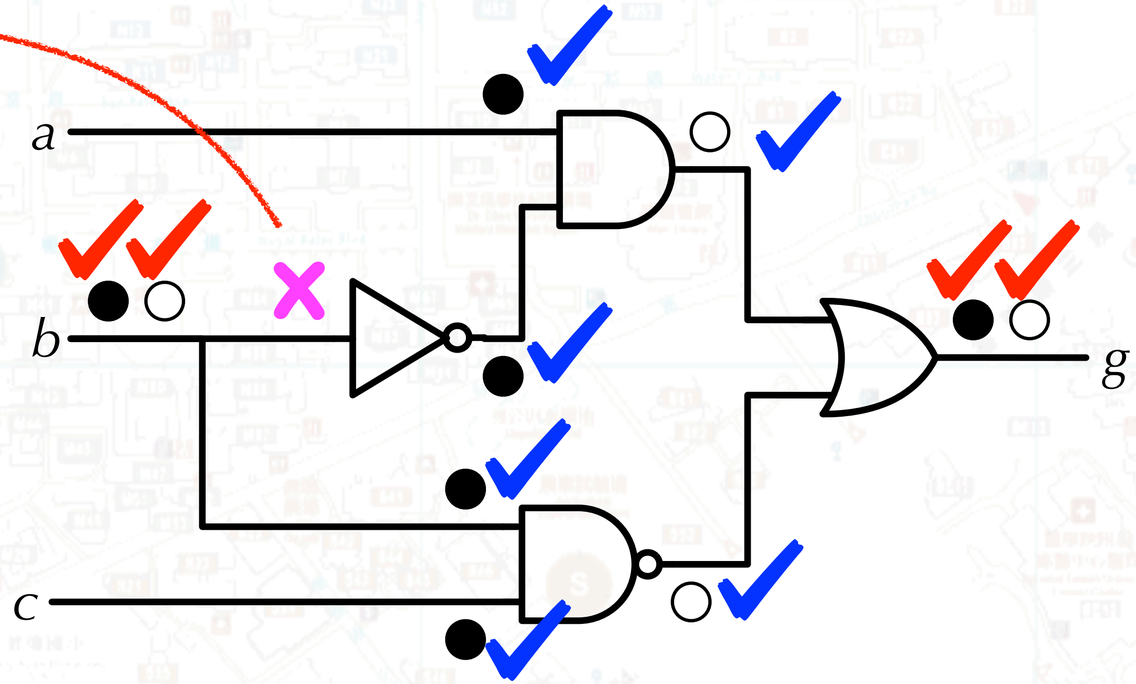
- The relation of equivalence partitions all faults into equivalence classes.
- For test generation, keep one fault from each equivalence class.
- For combinational circuits, apply the rules of primitive gates from PI to PO or vice versa.



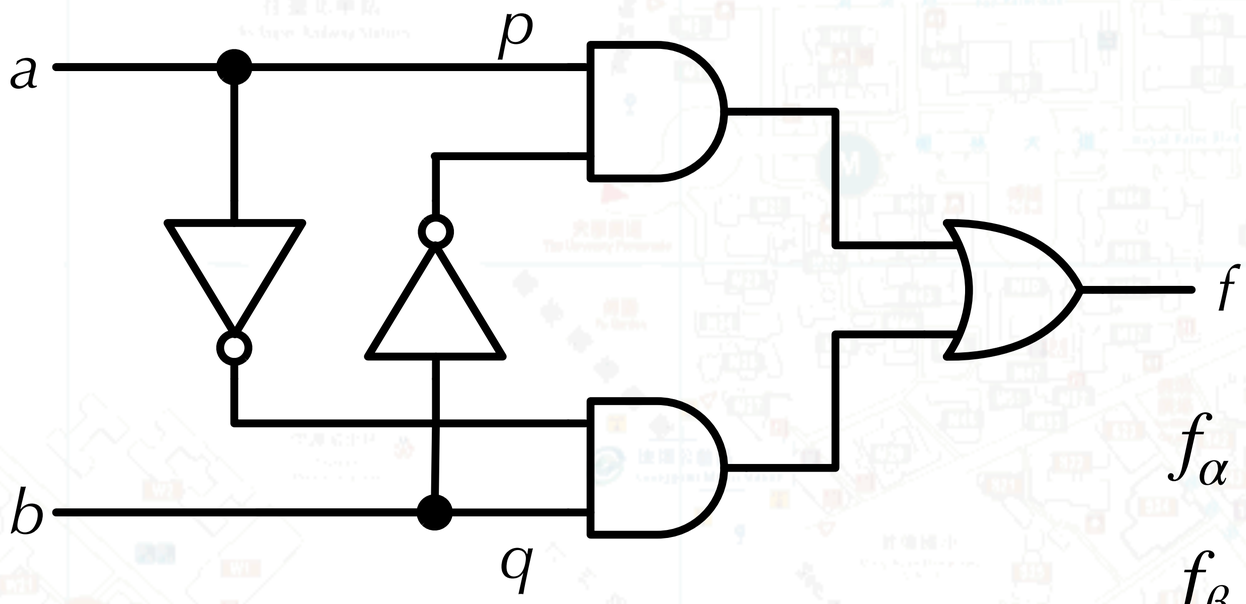


- After equivalence fault collapsing, the number of faults to consider is $2 \times (P_O + F_O) + G_i - N_i$ where
  - P_O is the number of primary outputs,
  - F_O is the number of fanout stems,
  - G_i is the number of gate inputs, and
  - N_i is the number of inverters

  which is about a 50 to 60% reduction.



- Simple structural analysis may not cover all equivalence relationships.
  - Example:
    p s-a-1 (α) and q s-a-1 (β) are equivalent, but the equivalence cannot be identified by simple structure analysis.

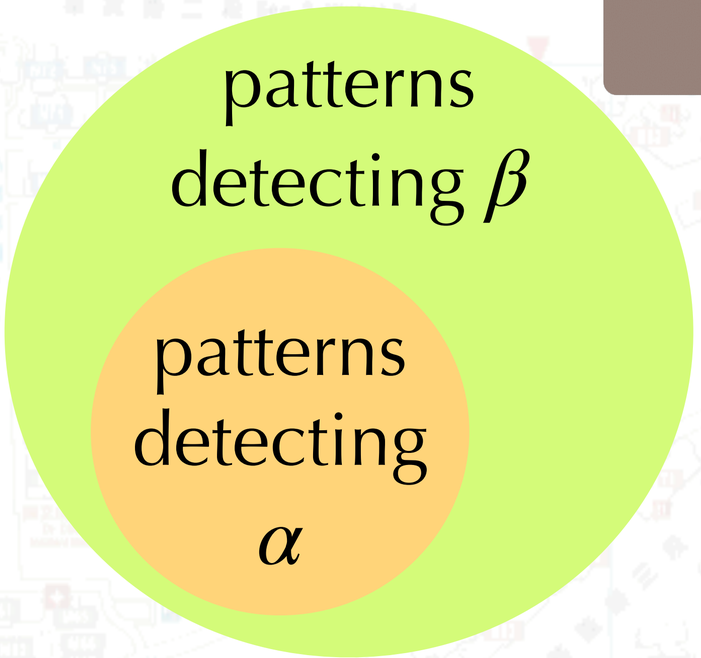


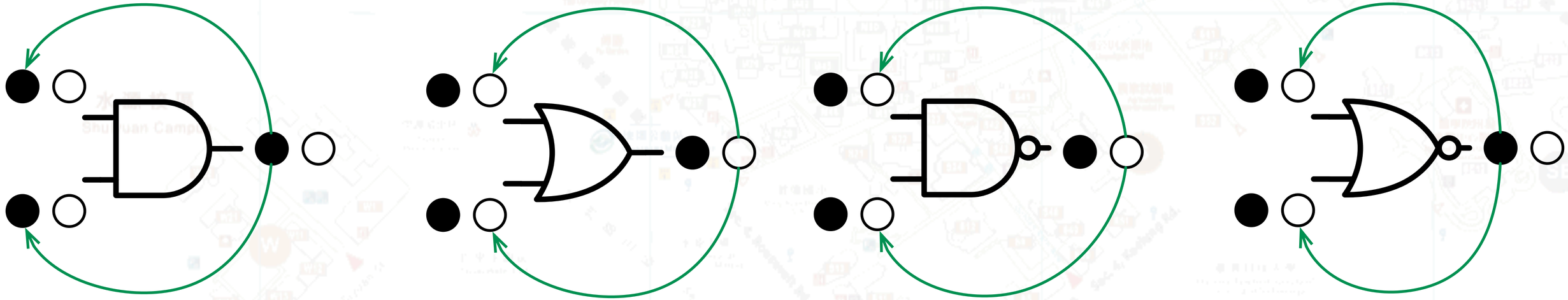
$$f_\alpha = b' + a'b = a' + b'$$

$$f_\beta = a' + ab' = a' + b'$$

Fault Dominance

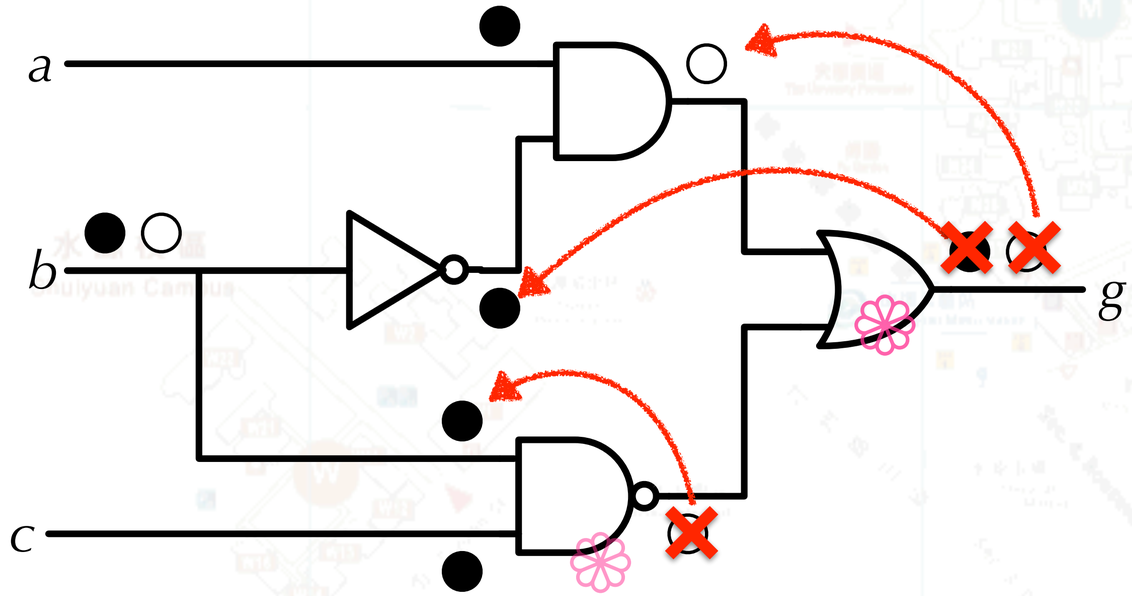
- If any test that detects fault α also detects the fault β, then we say fault β dominates fault α.
- For test generation, we can remove the fault β from the fault list.

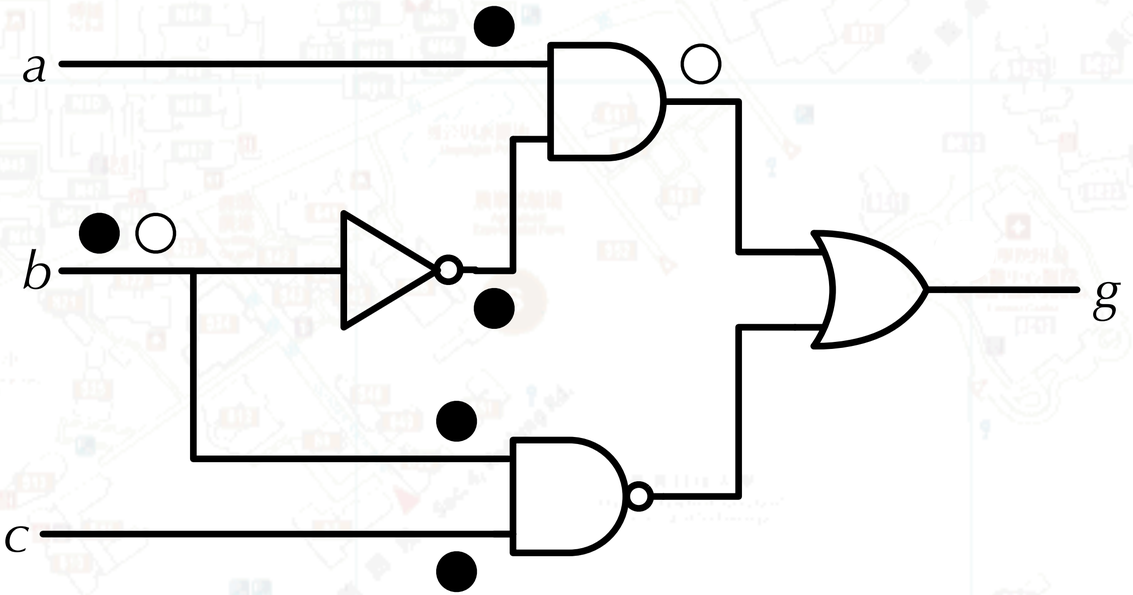




Dominance Fault Collapsing

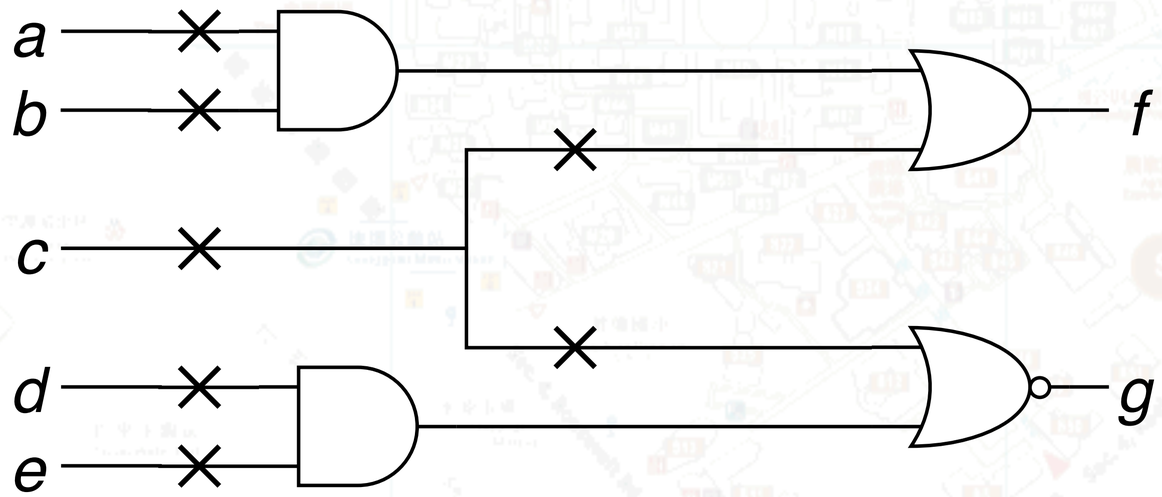
- Eliminating dominating faults from the equivalence collapsed set.
- An *n*-input primitive gate requires *n*+1 single stuck-at faults to be modeled.
- No collapsing is possible for a fanout.





Checkpoint Theorem

- Checkpoints: primary inputs and fanout branches.
- A test set that detects all single stuck-at faults of the checkpoints of a combination circuit detects all single stuck-at faults in that circuit.
  - For fanout-free circuits, we only have to model input faults.



Defect-Oriented Test

Challenges

- The SSF (Single-Stuck-At Fault) model alone cannot provide the desired defect coverage for advanced technology nodes.
  - Not all real defects are detected by test patterns that achieve high SSF coverage.
  - Being a *pin fault model*, the SSF model can only describe relatively simple defect behavior.
- Solutions?
  - Leverage the well-studied and mature SSF test technologies. *N-Detect*
  - Develop new fault models that more accurately describe the target defects' behavior. *Cell-Aware Test*

N-Detect

- Detect each SSF by at least *N* times.
- Improve the probability of fortuitously detecting unmodeled faults.
- The test generation procedure in [Benware ITC03]:
  - Often results in a severe test set inflation.
- In practice, bias the test generation process to favor the target defects.

1. Perform *single-detect* fault simulation with *single-detect* pattern set T_1 for all faults
2. Save all faults detected by single-detect fault simulation with pattern set T_1 (TF_{MD})
3. Set the number of detections N
4. For $K = 1$ to $(N-1)$
   - Perform *multiple-detect* fault simulation with pattern sets T_1 to T_K for TF_{MD} faults
   - Save faults detected K times (F_K)
   - Target faults F_K and perform *single-detect* ATPG to increase the number of detections by one
   - Save the patterns to $T_{(K+1)}$
5. Perform *multiple-detect* fault simulation with pattern sets T_1 to T_N for all faults to obtain multiple-detect fault coverage profile

Figure 4 Multiple-detect ATPG methodology

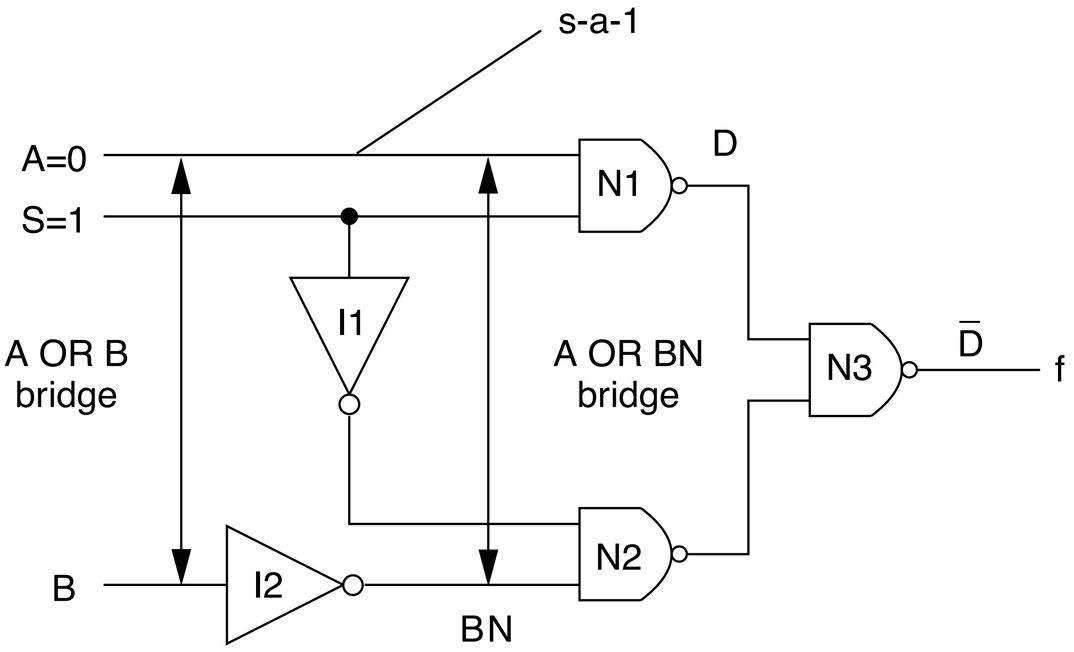
Table 2 Experimental Results to Compare N-Detect Pattern Set Size Before and After ***ILP*** Optimization

	N=3				N=5				N=10			
	# Original Patterns	# Patterns After Optimize	Save (%)	CPU Time (s)	# Original Patterns	# Patterns After Optimize	Save (%)	CPU Time (s)	# Original Patterns	# Patterns After Optimize	Save (%)	CPU Time (s)
B01	23	21	8.7	5.5	50	46	8	7.2	68	65	4.6	8.0
B02	18	17	5.6	3.1	18	18	0	3.6	18	18	0	3.5
B03	58	51	12.1	43.2	81	70	13.6	125	159	149	6.3	317
B04	238	192	19.3	479	323	286	11.5	618	558	505	9.5	1335
B05	150	110	26.7	356	206	167	18.9	577	320	285	10.9	2014
B06	25	20	20	8.4	27	22	18.5	8.9	27	26	3.7	9.3
B07	176	153	13.1	415	265	234	11.7	932	451	414	8.2	2307
B08	109	96	11.9	378	155	140	9.7	662	272	259	4.8	1973
B09	66	54	18.2	176	83	76	8.4	405	140	133	5	1455
B10	92	85	7.6	308	152	139	8.5	624	257	249	3.1	1941
B11	165	149	9.7	531	236	223	5.5	1798	413	391	5.3	3129
B12	456	374	18	1115	621	553	11	2734	943	908	3.7	4009
B13	102	83	18.6	967	147	128	12.9	1590	229	211	7.9	3638
B14	753	605	19.7	1732	1100	988	10.2	4208	1838	1704	7.3	7124
B15	1001	790	21.1	3773		---			2476	2244	9.4	9436
B17		---			1389	1144	17.6	7946		---		
CK1	220	179	18.6	2700	384	326	15.1	4116	675	605	10.4	7021
CK2	188	159	15.4	3172	329	282	14.3	8353		---		
Total	3840	3138	18.3	16161	5566	4842	13	34708	8844	8166	7.7	45720

[Huang ISQED06]

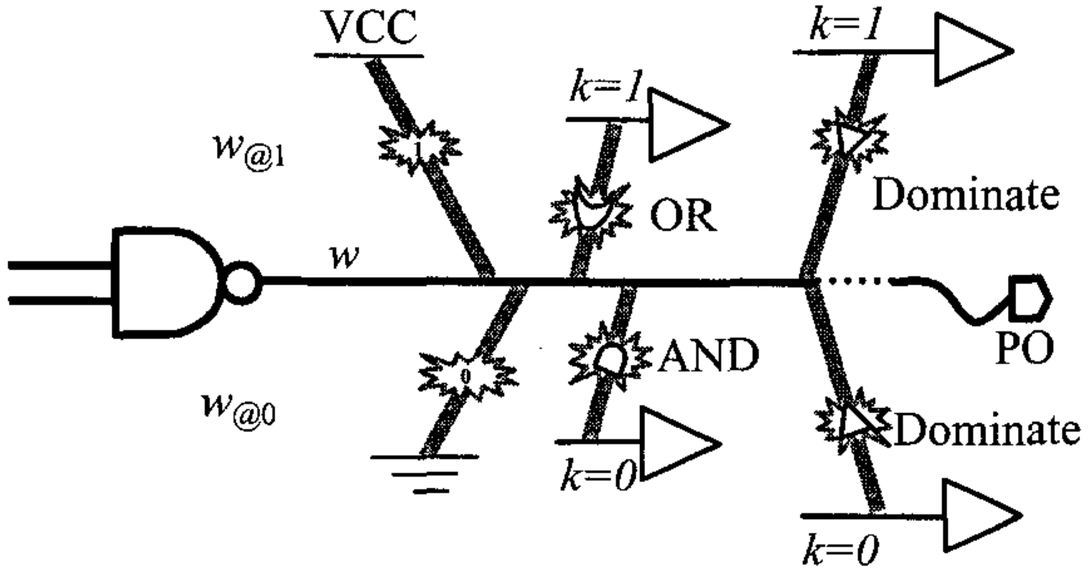
How does *N*-Detect work?

- Fact:
  SSF test patterns fortuitously detect many unmodeled faults.
  - The Murphy chip data showed that only about one-third of the defects caused the affected CUT to respond to inputs with the same output responses as if an SSF was present in the CUT [McCluskey ITC96].
- Example:
  - *ASB* = 01x detects *A* sa1 and may detect the A OR B and A OR BN bridges.
  - Detecting *A* sa1 multiple times improves the probability of detecting the bridging faults.



Detecting Bridging Faults w/ *N*-Detect

- [Benware ITC03]



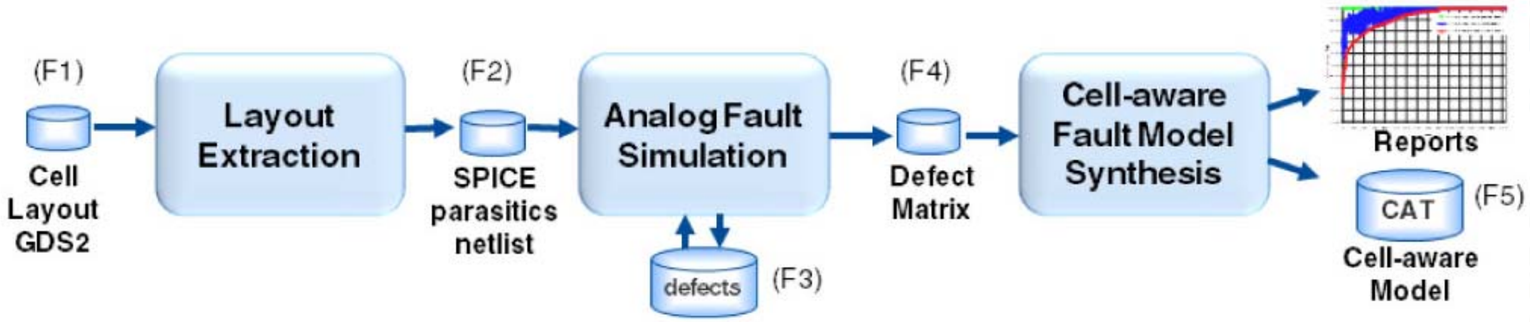
	w	*k*	OR	AND	D	V_{cc}	V_{ss}
$w_{@1}$	0	0	0	0	0	1	0
$w_{@1}$	0	1	1	0	1	1	0
$w_{@0}$	1	0	1	0	0	1	0
$w_{@0}$	1	1	1	1	1	1	0

Cell-Aware Test (CAT)

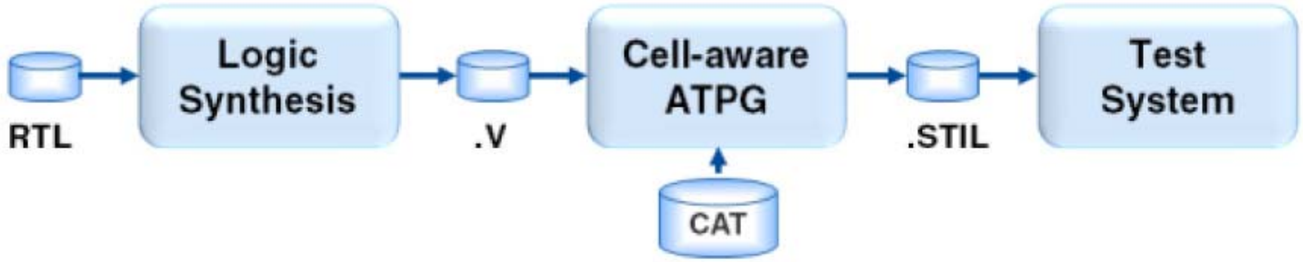
- Available fault models are still insufficient to deliver acceptable defect coverage.
  - Stuck-at (SA), bridge, transition, N-detect, gate-exhaustive, embedded-multi-detect (EMD), timing-aware, and layout-aware fault models on interconnect lines.
  - **Many defects that escape testing are in fact defects within the standard library cells [Hapke TCAD14].**
- The CAT fault model is based on a post-layout transistor-level netlist including parasitic objects, resulting in a defect-based ATPG approach, which can be applied to large, state-of-the-art designs.

CAT Test Methodoloty

- CAT test library generation:



- CAT test generation:



Layout and Defect Extraction

- From GDS II, extract the netlist of circuit elements, including transistors and parasitics.
- Layout-aware defect extraction, e.g., cell-internal bridge, open.

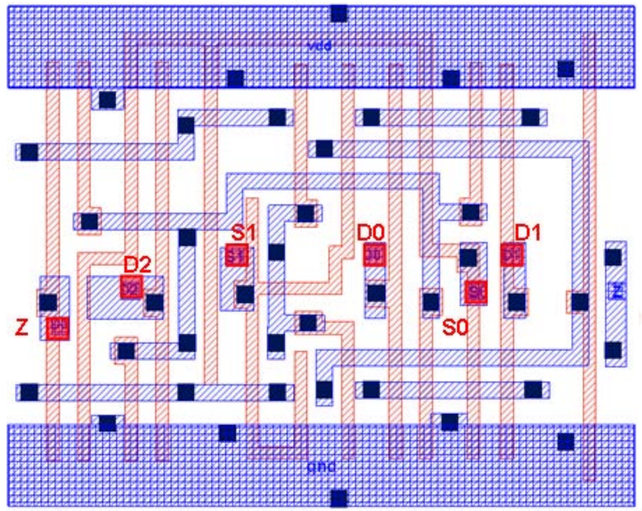


Figure 2: MUX31X4 layout

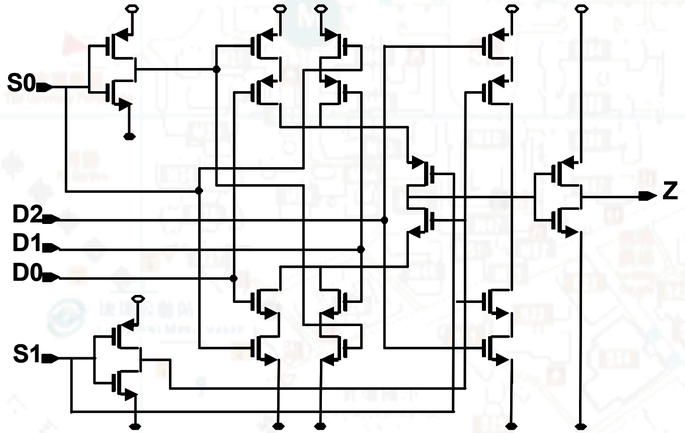


Figure 3: Extracted Transistor Netlist

d1 = S0N, gnd	d17 = D0, gnd	d33 = net38, D1
d2 = S1N, gnd	d18 = vdd, gnd	d34 = net81, D0
d3 = net65, gnd	d19 = Z, net65	d35 = net38, D0
d4 = net57, gnd	d20 = S1, S0N	d36 = S1, S0
d5 = net19, gnd	d21 = S1N, S1	d37 = D2, S1
d6 = net81, gnd	d22 = net65, S1	d38 = S0, D1
d7 = net38, gnd	d23 = S0N, S0	d39 = vdd, S0N
d8 = net85, gnd	d24 = net81, S1	d40 = vdd, S1N
d9 = net35, gnd	d25 = S1, net38	d41 = vdd, net65
d10 = net31, gnd	d26 = D2, S1N	d42 = D0, S0
d11 = net69, gnd	d27 = net81, S0	d43 = net38, vdd
d12 = Z, gnd	d28 = net65, D2	d44 = vdd, Z
d13 = S1, gnd	d29 = net38, S0	d45 = vdd, S1
d14 = S0, gnd	d30 = S0N, D1	d46 = vdd, S0
d15 = D2, gnd	d31 = net81, D1	d47 = vdd, D2
d16 = D1, gnd	d32 = S0N, D0	d48 = vdd, D1

Figure 4: List of Defects of MUX31X4

[Hapke ITC09]

TABLE II
DESIGN DATA/RESULTS OF INDUSTRIAL DESIGNS

Design	# Gates [million]	# FFs [thousands]	# Chains	# SA Faults [million]	# CAT Defects [million]
# 1	1.3	136	300	5.0	23.0
# 2	2.1	148	70	8.2	30.8
# 3	2.4	222	1250	13.9	37.8
# 4	2.1	215	312	9.7	38.0
# 5	2.6	271	600	10.0	46.1
# 6	2.4	174	114	9.6	36.2
# 7	3.2	318	1145	22.8	100.4
# 8	3.6	751	408	85.3	138.3
# 9	3.9	407	900	14.9	69.1
# 10	5.6	458	1020	22.4	92.6

[Hapke ITC14]

Analog Fault Simulation

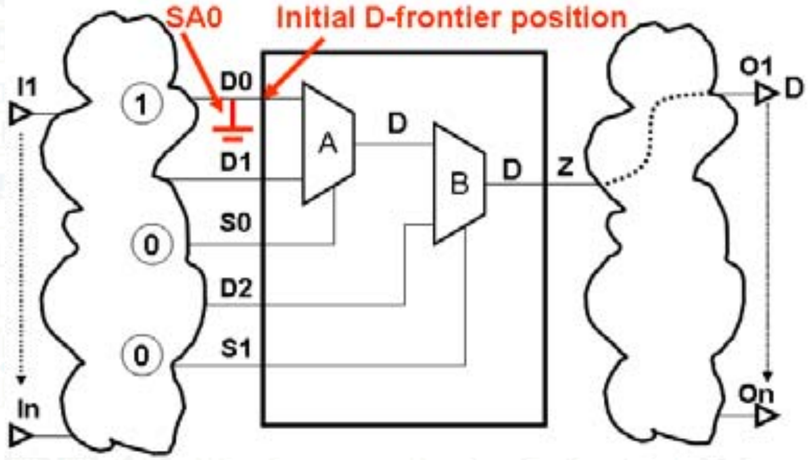
- An exhaustive analog simulation determines the complete cell-input combinations that detect the defect.
- The resulting defect matrix summarizes the detection results for each cell.
- Derive an optimized set of input assignments required to detect individual cell-internal faults.

stimuli	d1	d2	d3	d4	...	d41	d42	d43	d44	d45	d46	d47	d48
00000	-	-	-	-		-	-	-	D	-	-	-	-
00001	D	D	-	-		D	D	D	-	D	D	-	-
00010	-	-	-	-		-	-	-	D	-	D	-	-
00011	-	D	-	-		D	-	D	-	D	-	-	-
00100	-	-	-	-		-	-	-	D	D	-	-	-
00101	D	-	-	-		D	D	D	-	-	D	-	-
00110	-	-	-	-		-	-	-	D	D	D	-	-
00111	-	-	-	-		D	-	D	-	-	-	-	-
01000	-	-	-	-		-	-	-	D	-	-	-	D
...													
11111	-	-	-	-		D	-	-	-	-	-	-	-

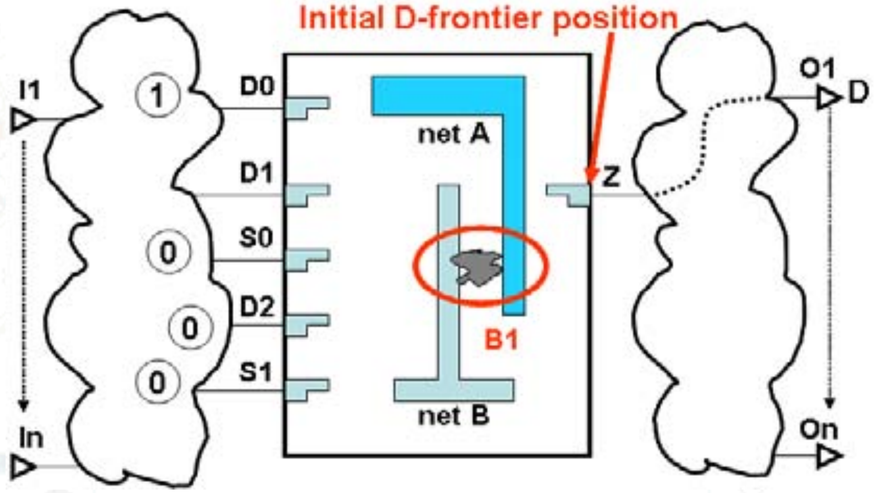
Figure 5: Defect Matrix [Hapke ITC09]

Cell-Aware Test Generation

- CAT models fault differently from conventional fault models.
- For conventional fault models, ATPG excites the fault and propagates the response according to predefined conditions for primitive gates, e.g., D0 = 0, S0 = 0, S1 = 0.



- For the CAT fault models, the fault effect and the fault excitation condition are placed directly at the cell output and inputs.



Defect Coverage Gain & Pattern Count

TABLE III
NUMBER OF TEST PATTERNS RELATED TO DEFECT COVERAGE GAIN

Design	Defect Coverage Gain with				Maximum #Pattern	
	+ 0% Pattern	+ 25% Pattern	+ 50% Pattern	+max Pattern	Additional CAT Static	Additional CAT Delay
# 1	2.5%	3.8%	4.2%	5.0%	51%	89%
# 2	2.5%	5.0%	5.9%	6.1%	72%	75%
# 3	2.8%	3.4%	3.5%	4.1%	14%	50%
# 4	2.7%	3.8%	4.7%	5.5%	77%	77%
# 5	2.1%	3.5%	4.2%	5.0%	58%	87%
# 6	2.5%	3.9%	4.6%	5.8%	13%	85%
# 7	2.2%	2.5%	3.0%	3.5%	57%	63%
# 8	2.6%	4.5%	5.3%	5.3%	45%	26%
# 9	1.9%	3.4%	3.9%	4.4%	56%	71%
# 10	3.5%	5.9%	6.6%	6.8%	47%	72%
Average	**2.5%**	**4.0%**	**4.6%**	**5.1%**	**49%**	**70%**

[Hapke TCAD14]

- +0% pattern: Use the same number of CAT patterns as the SA + TR pattern set.
- The last two columns show the maximum number of CAT patterns compared to SA and TR patterns, respectively.

Conclusions

- Fault modeling is the basis of test technologies.
- SSF is a must, and transition delay fault has become mandatory.
- New fault modeling technologies are needed to guarantee acceptable defect coverage.
  - Leverage current fault models (*N*-detect) or develop new fault models (CAT).
  - Conventional fault models are insufficient to achieve high defect coverage.