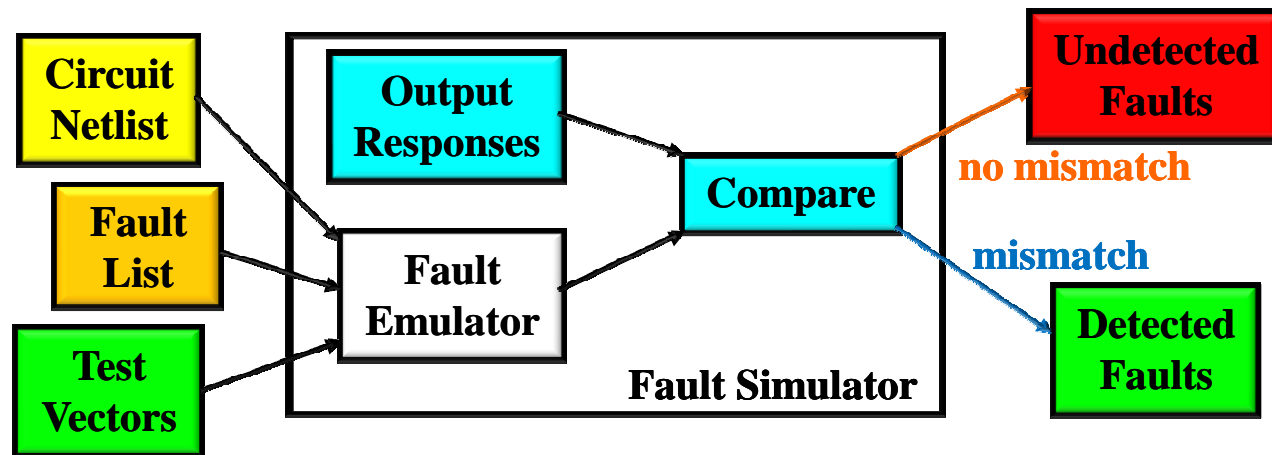


Fault Models, Detection & Simulation - Organization

- Testing & Fault Simulation
- Fault Models & Detection
 - ❖ Single vs. Multiple Fault Models
 - ❖ Gate Level Faults
 - Equivalent Faults & Collapsing
 - ❖ Transistor Level Faults
 - Equivalent Faults & Collapsing
 - ❖ Bridging Faults
 - Bridging Fault Extraction
 - ❖ Delay Faults
- Fault Detection & Fault Coverage
 - ❖ Controllability & Observability
 - ❖ Path Sensitization
 - ❖ Undetectable & Potentially Detected Faults
 - ❖ Fault Coverage
 - ❖ N-Detectability

Fault Simulation

- Fault simulation similar to testing process
 - ❖ evaluates fault detection capabilities of a set of test patterns
 - Fault coverage = percentage of faults detected by set of test vectors
- Fault simulators:
 - ❖ emulate faults
 - ❖ compare output response to known good circuit output response
 - For a given set of input test patterns
 - ❑ At least one mismatch \Rightarrow fault is detected
 - ❑ No mismatches \Rightarrow fault is not detected



Fault Detection

- Fault detection requires:
 - ❖ observation of an error (from the fault) at a primary output
 - **observability** of the fault site
 - The ease at which we can observe the fault behavior
 - ❖ input stimuli that creates an error as a result of fault
 - **controllability** of the fault site
 - The ease at which we can control the fault behavior
 - **controllability** of path from fault site to primary output
 - Also considered part of **observability**
- Testability of fault \propto **controllability** & **observability** of site
- Circuit testability \propto overall circuit **controllability** & **observability**

Fault Coverage (simple view)

- Given a set of test vectors, each fault in the fault set for the circuit can be:

- ❖ **$D = \text{detected faults}$**

- Targeted faults and faults “accidentally” detected

- ❖ **$U = \text{undetected faults}$**

- Could not find a vector to detect fault

- *But there might be one not included in vector set*

- ❖ **$T = \text{total faults} = D + U$**

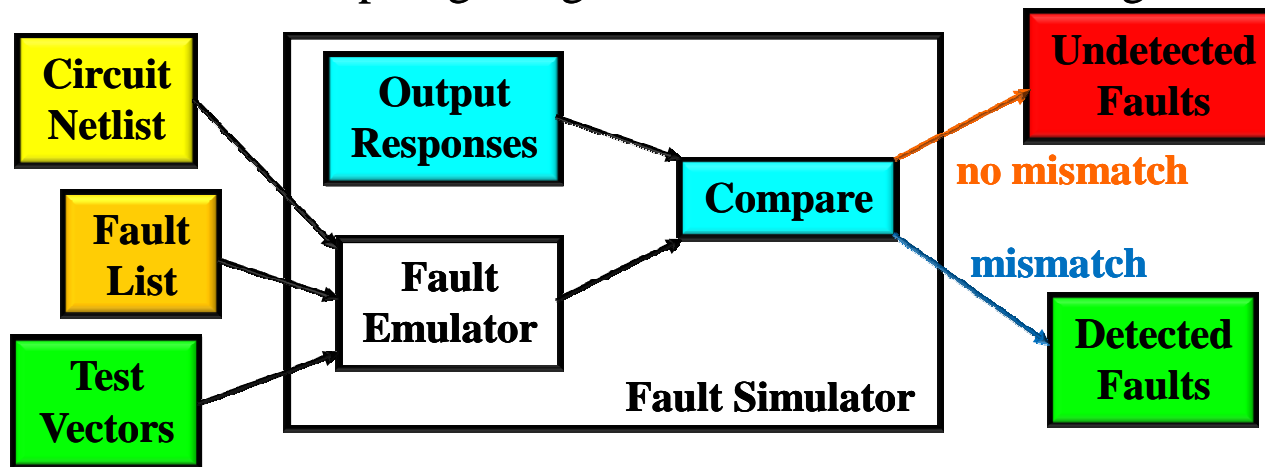
- ❖ **Fault coverage = D / T**

- For that specific set of test vectors

- Also referred to as *fault grading*

Fault Simulation

- Fault simulation time:
 - ❖ Circuit must be simulated for each fault
 - N faults $\Rightarrow N$ simulations of circuit
 - ❖ Fault simulation speed-up by:
 - Simulation of a given fault ends on detection
 - called fault dropping
 - Parallel fault simulation emulates 1 fault/bit of computer word
 - Statistical fault sampling
 - >1000 samples gives good estimate of fault coverage



Simulation Time

- Logic simulation

- ❖ $T_{vec} \times N_{vec}$

- T_{vec} = time to simulate one vector

- N_{vec} = number of vectors

- Serial fault simulation

- ❖ Worst case (no faults dropping):

- N_{flt} = number of faults

- ❖ Best case (faults dropping):

- N_{vec_i} = number of vectors until fault i is detected

$$T_{vec} \times N_{vec} \times N_{flt}$$

$$\sum_{i=1}^{N_{flt}} T_{vec} \times N_{vec_i}$$

- Parallel fault simulation (simulate sets of P_{flt} faults in parallel)

- ❖ Worst case (no faults dropping):

- P_{flt} = number of faults emulated in parallel

- Number of bits in computer integer or long

- ❖ Best case (faults dropping):

- N'_{vec_i} = number of vectors until last fault in set is detected

$$T_{vec} \times N_{vec} \times \left\lceil \frac{N_{flt}}{P_{flt}} \right\rceil$$

$$\sum_{i=1}^{\left\lceil \frac{N_{flt}}{P_{flt}} \right\rceil} T_{vec} \times N'_{vec_i}$$

Fault Models

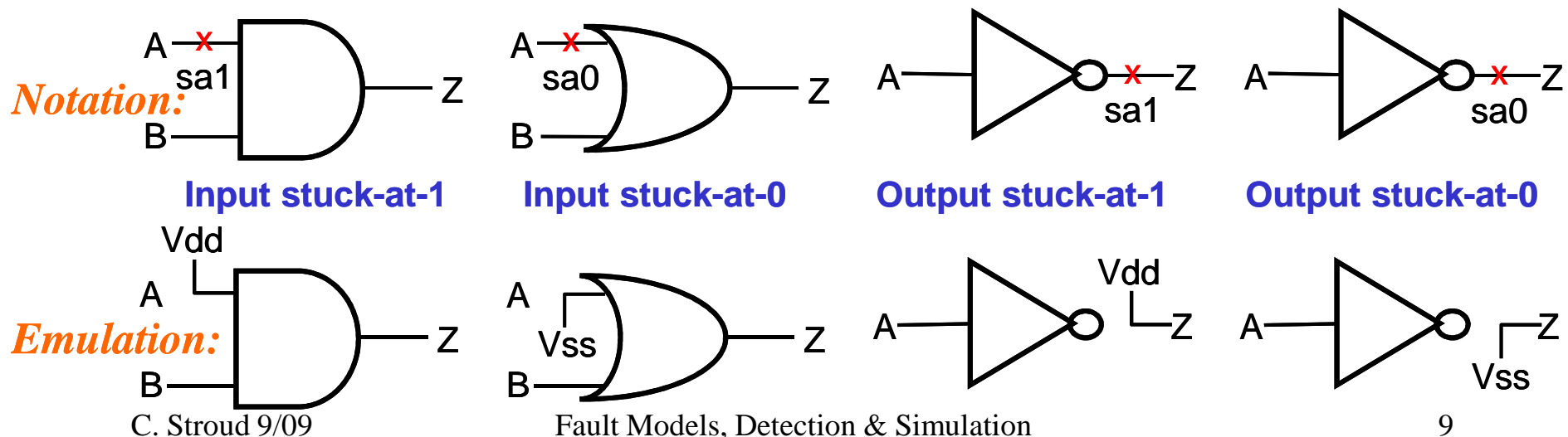
- A good fault model has 2 requirements:
 1. accurately reflects the behavior of a physical defect
 2. is computationally efficient with respect to simulation
- ❖ Single fault model (aka “assumption”) used for # 2
- Current common fault models include:
 - ❖ Gate level stuck-at faults
 - Stuck-at-0 (sa0) & stuck-at-1 (sa1)
 - ❖ Transistor level stuck faults
 - Stuck-on (stuck-closed) & stuck-off (stuck-open)
 - ❖ Bridging faults (shorts between wires)
 - Wired-AND & wired-OR
 - Dominant (one driving source dominates the other)
 - ❑ Note: opens in wires typically covered by stuck-faults
 - ❖ Delay faults
 - Excessive delay in a transition, a gate, or a path
 - ❑ Known as transition, gate, or path delay fault, respectively

Single vs. Multiple Fault Models

- Assume
 - ❖ $N =$ number of fault sites
 - ❖ $K =$ number of possible faults per site
- Single vs. multiple fault models
 - ❖ Single - there can be only one fault at a time
 - # Possible Faulty Circuits = $K \times N$
 - Less accurate but more efficient to simulate
 - ❖ Multiple - there any number of faults at a time
 - # Possible Faulty Circuits = $(K+1)^N - 1$
 - The -1 circuit is the fault-free circuit
 - More accurate but cannot possibly simulate all fault groups
 - Statistical fault sampling is only practical solution
- High single Fault Coverage (FC) \Rightarrow high multiple FC
 - ❖ In general, 100% single FC \Rightarrow $> 98\%$ multiple FC

Gate Level Stuck-at Fault Model

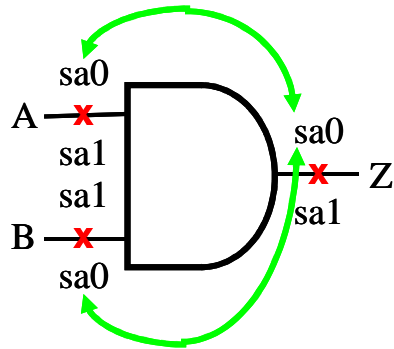
- Gate inputs or outputs can be:
 - ❖ Stuck-at-0 (sa0)
 - As if input or output were disconnected and tied low to Vss
 - ❖ Stuck-at-1 (sa1)
 - As if input or output were disconnected and tied high to Vdd
 - ❖ fault site denoted by 'X' with 'sa0' or 'sa1'
 - Note: there is no feedback of fault value from fault site!
- Also known as line stuck-at fault model
 - ❖ Any line can be sa0 or sa1



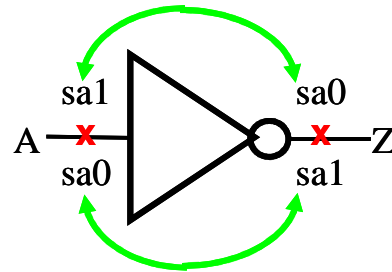
Gate Level Equivalent Faults and Fault Collapsing

- Equivalent faults are indistinguishable & can be collapsed
 - ❖ 1 fault in set of equivalent faults represents all in set
 - ❖ Fewer faults to simulation \Rightarrow faster fault simulations
- Gate level collapsing
 - ❖ **AND (NAND)** gates
 - Any input $sa0 = \text{output } sa0$ ($sa1$)
 - ❑ # Collapsed Faults = $I + 2$ (where $I = \#$ inputs)
 - ❖ **OR (NOR)** gates
 - Any input $sa1 = \text{output } sa1$ ($sa0$)
 - ❑ # Collapsed Faults = $I + 2$ (where $I = \#$ inputs)
 - ❖ **INVERTER**
 - Input $sa0$ ($sa1$) = output $sa1$ ($sa0$)
 - ❑ # Collapsed Faults = 2

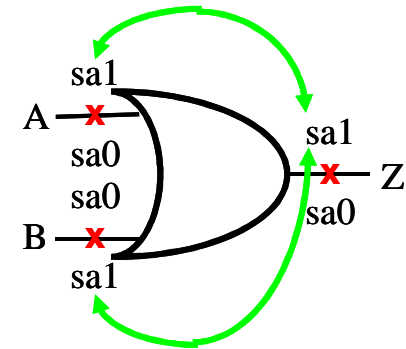
Gate Level Equivalent Faults and Fault Collapsing



Collapsed fault set
 $Z\ sa0 = A\ sa0 = B\ sa0$
 $Z\ sa1$
 $A\ sa1$
 $B\ sa1$



Collapsed fault set
 $Z\ sa1 = A\ sa0$
 $Z\ sa0 = A\ sa1$



Collapsed fault set
 $Z\ sa1 = A\ sa1 = B\ sa1$
 $Z\ sa0$
 $A\ sa0$
 $B\ sa0$

AND

| AB | Z | Asa0 | Asa1 | Bsa0 | Bsa1 | Zsa0 | Zsa1 |
|----|---|------|------|------|------|------|------|
| 00 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 01 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 10 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

INVERTER

| A | Z | Asa0 | Asa1 | Zsa0 | Zsa1 |
|---|---|------|------|------|------|
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |

OR

| AB | Z | Asa0 | Asa1 | Bsa0 | Bsa1 | Zsa0 | Zsa1 |
|----|---|------|------|------|------|------|------|
| 00 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 01 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 11 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

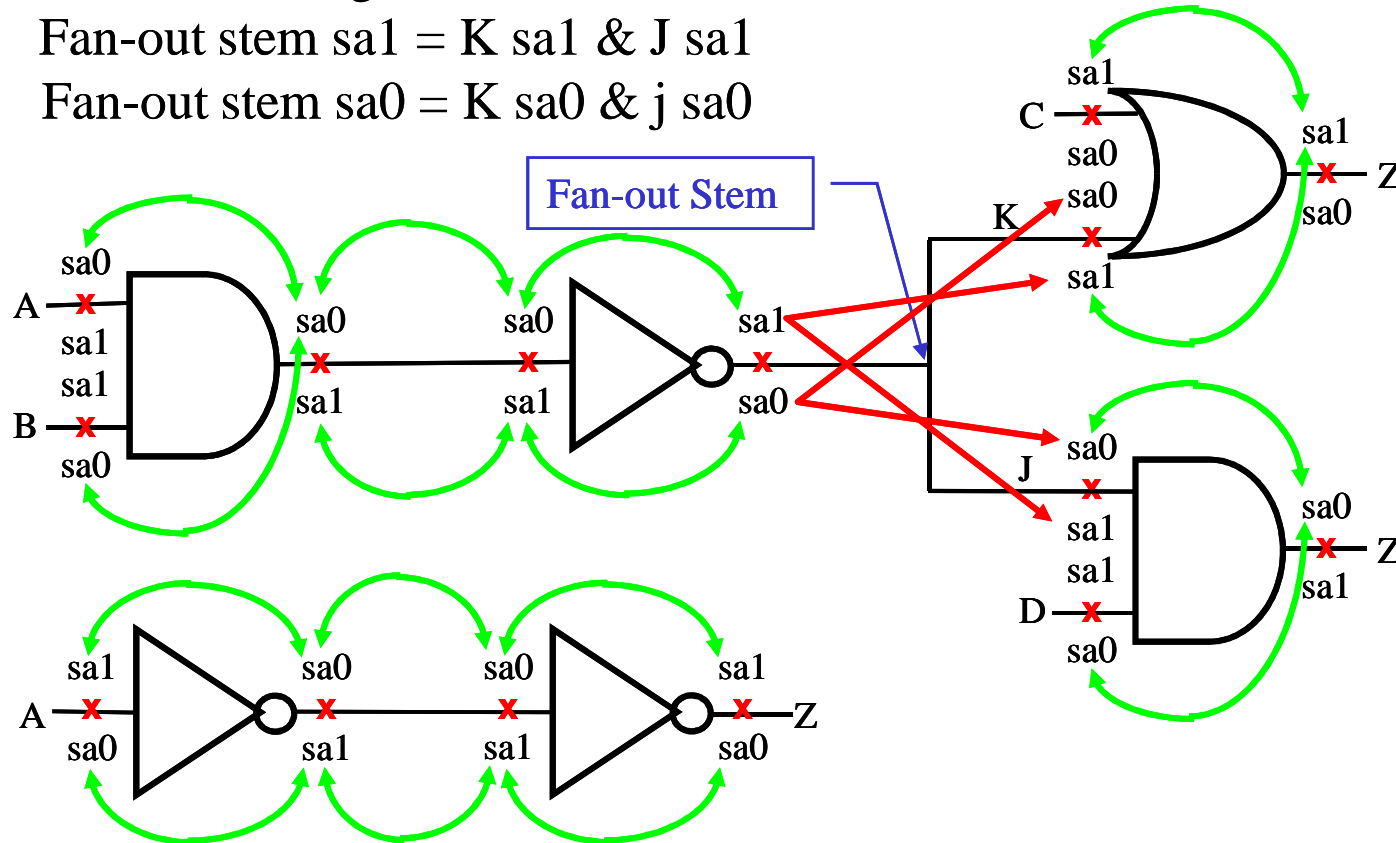
Structural Equivalent Faults and Fault Collapsing

Cannot collapse faults at fan-out stem since it would violate single stuck-at fault model

Fan-out stem sa1 = K sa1 & J sa1

Fan-out stem sa0 = K sa0 & j sa0

collapsed faults = 12

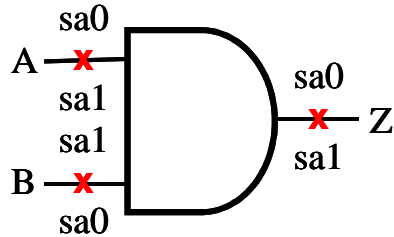


Only 2 collapsed faults for inverter chain (Z sa0 & Z sa1)

Collapsed vs. Uncollapsed Gate Level Fault Sets

- For a given circuit (assuming elementary logic gates)
 - ❖ # uncollapsed faults = $2(G + G_I)$
 - G = total # gates
 - G_I = total # gate inputs
 - ❖ # collapsed faults = $2(O_P + F_S) + G_I - N_I$
 - O_P = # primary outputs
 - F_S = # fan-out stems
 - N_I = # inverters
 - ❖ Typically # collapsed flts $\approx 1/2$ # uncollapsed flts
- Should faults be collapsed?
 - ❖ **YES**: for TPG and fault simulation (more efficient)
 - ❖ **NO**: for computing fault coverage (more accurate)
 - But longer fault simulation times

Gate Level Fault Detection



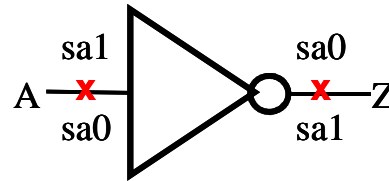
Collapsed fault set
 $Z\ sa0 = A\ sa0 = B\ sa0$
 $Z\ sa1$
 $A\ sa1$
 $B\ sa1$

AND

| AB | Z | A _{sa0} | A _{sa1} | B _{sa0} | B _{sa1} | Z _{sa0} | Z _{sa1} |
|----|---|------------------|------------------|------------------|------------------|------------------|------------------|
| 00 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 01 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 10 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Minimum Set of Test Vectors

01
 10
 11



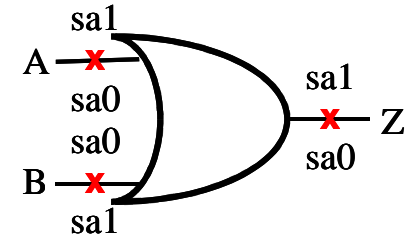
Collapsed fault set
 $Z\ sa1 = A\ sa0$
 $Z\ sa0 = A\ sa1$

INVERTER

| A | Z | A _{sa0} | A _{sa1} | Z _{sa0} | Z _{sa1} |
|---|---|------------------|------------------|------------------|------------------|
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |

Minimum Set of Test Vectors

0
 1



Collapsed fault set
 $Z\ sa1 = A\ sa1 = B\ sa1$
 $Z\ sa0$
 $A\ sa0$
 $B\ sa0$

OR

| AB | Z | A _{sa0} | A _{sa1} | B _{sa0} | B _{sa1} | Z _{sa0} | Z _{sa1} |
|----|---|------------------|------------------|------------------|------------------|------------------|------------------|
| 00 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 01 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 11 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

Minimum Set of Test Vectors

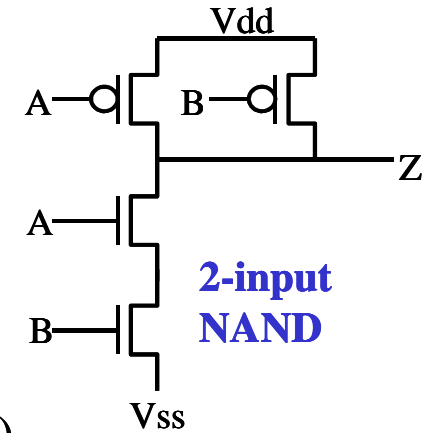
00
 01
 10

Minimum Set of Test Vectors for Gate Level Faults

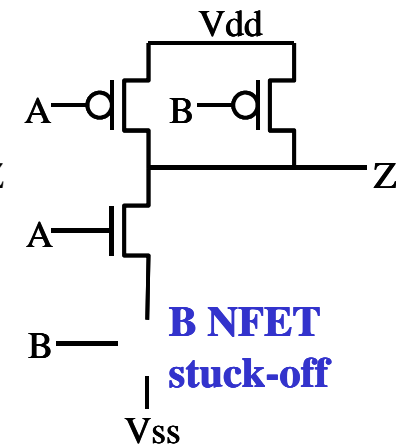
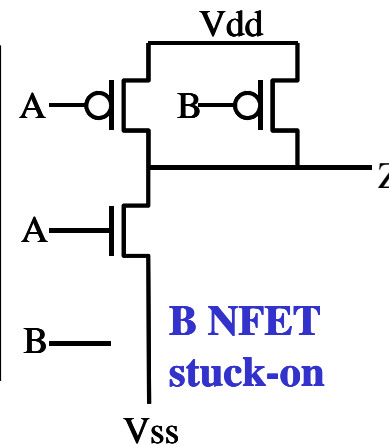
- Inverter requires both input logic values (0 and 1)
 - ❖ # vectors = 2
- N -input AND or NAND gate
 - ❖ # vectors = $N + 1$
 - All 1s
 - Walk 0 through a field of 1s
- N -input OR or NOR gate
 - ❖ # vectors = $N + 1$
 - All 0s
 - Walk 1 through a field of 0s
- XOR is not an elementary logic gate (*made from multiple gates*)
 - ❖ # faults depends on construction of gate
 - 3 vectors required for pin faults {01, 10, and 00 *or* 11}
 - All 4 vectors required for internal faults

Transistor Level Fault Model

- Transistor can be:
 - ❖ Stuck-on (a.k.a. stuck-short)
 - Can result in excessive I_{DDQ}
 - ❖ Stuck-off (a.k.a. stuck-open)
 - Can result in “memory” node (logic gate \Rightarrow latch)
- Gate level fault model accurate for NMOS but not for CMOS
 - ❖ 1 gate input stuck-at = 2 transistors stuck-at in CMOS

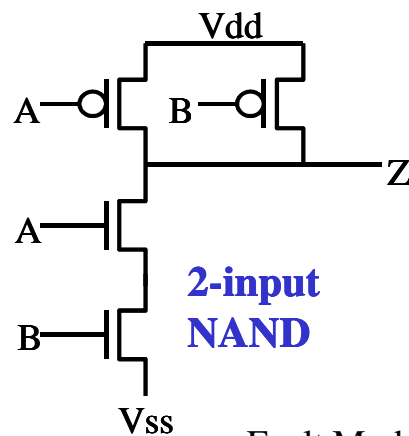


| | | A PFET | | A NFET | | B PFET | | B NFET | |
|----|---|-----------|-------|-----------|-------|-----------|-------|-----------|-------|
| AB | Z | s-on | s-off | s-on | s-off | s-on | s-off | s-on | s-off |
| 00 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 01 | 1 | 1 | mem | I_{DDQ} | 1 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 | 1 | 1 | mem | I_{DDQ} | 1 |
| 11 | 0 | I_{DDQ} | 0 | 0 | mem | I_{DDQ} | 0 | 0 | mem |



Transistor Level Fault Equivalence & Collapsing

- Stuck-off faults in series transistors are equivalent
- Stuck-on faults in parallel transistors are equivalent
- # collapsed transistor faults = $2T - N_{ser} + G_{ser} - N_{par} + G_{par}$
 - ❖ N_{ser} = total # series transistors
 - ❖ G_{ser} = total # groups of series transistors
 - ❖ N_{par} = total # parallel transistors
 - ❖ G_{par} = total # groups of parallel transistors



collapsed faults = 6
 A PFET s-on = B PFET s-on
 A NFET s-off = B NFET s-off
 A PFET s-off
 B PFET s-off
 A NFET s-on
 B NFET s-on

Transistor Level Fault Detection

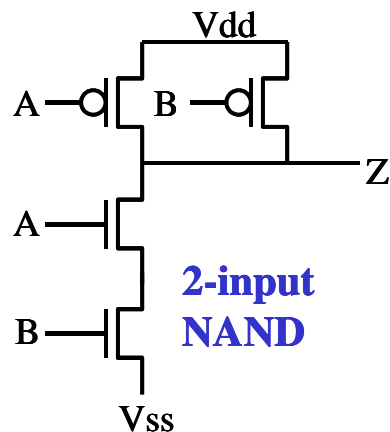
- Transistor fault detection more difficult than gate level

- ❖ Stuck-on faults

- Voltage divider may not produce incorrect logic values
- Monitoring I_{DDQ} is best approach
 - ❑ Small currents may be lost in leakage of $> 2M$ transistors

- ❖ Stuck-off faults

- Will produce wrong logic values, *but*
 - ❑ Need ordered set of 2 vectors to detect



| | | A PFET | | A NFET | | B PFET | | B NFET | |
|----|---|-----------|-------|-----------|-------|-----------|-------|-----------|-------|
| AB | Z | s-on | s-off | s-on | s-off | s-on | s-off | s-on | s-off |
| 00 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 01 | 1 | 1 | mem | I_{DDQ} | 1 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 | 1 | 1 | mem | I_{DDQ} | 1 |
| 11 | 0 | I_{DDQ} | 0 | 0 | mem | I_{DDQ} | 0 | 0 | mem |

Vectors to detect:

A PFET stuck-off

- 11 - to get Z=0
- 01 - to detect Z=0

B PFET stuck-off

- 11 - to get Z=0
- 10 - to detect Z=0

A/B NFETs s-off

- 0x or x0 - to get Z=1
- 11 to detect Z=1

Min. #vectors = 4

detects s-on w/ I_{DDQ}

Bridging Fault Models

- Two common models for wires shorted together:

- ❖ **Wired-AND/Wired-OR** fault model

- Shorted wires perform logical AND or logical OR
 - Developed for bipolar technology

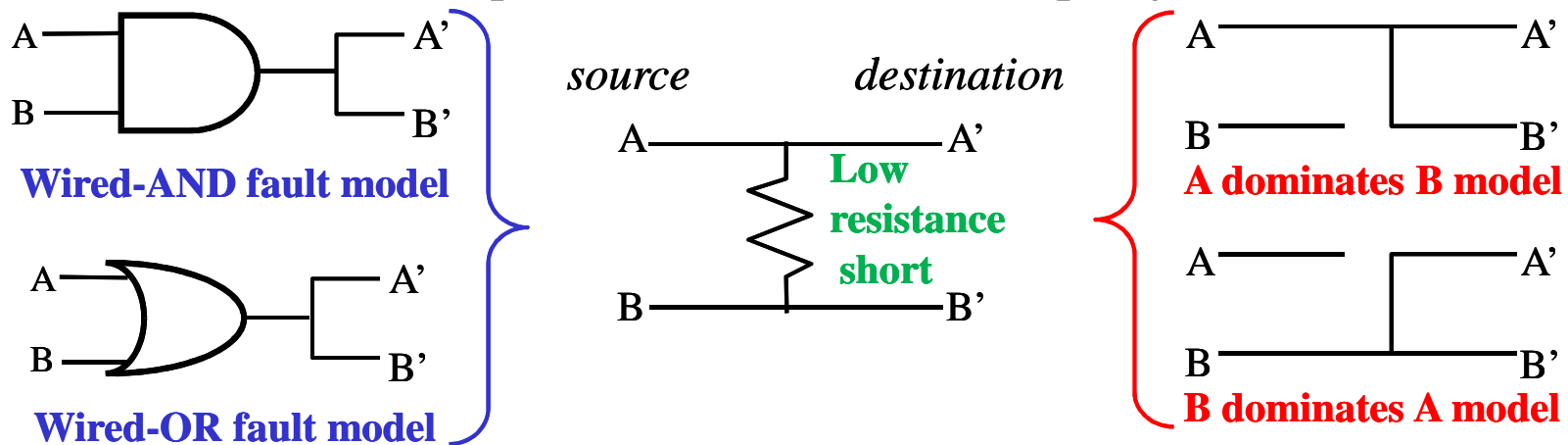
- ❖ **Dominant** fault model

- Stronger driving gate dominates the short
 - More accurate for MOS technology

- Two faults per fault site

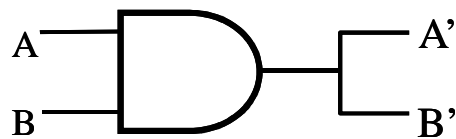
- ❖ For N nets, # pair-wise bridging faults = $N^2 - N$

- No fault equivalence \Rightarrow no fault collapsing

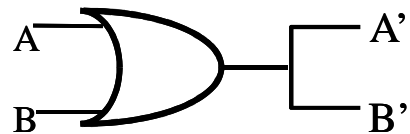


Bridging Fault Detection

- Wired-AND/Wired-OR faults
 - ❖ 1 vector (01 or 10) with 2 outputs (A' and B'), *or*
 - ❖ 1 output (A' or B') with 2 vectors (01 and 10)
- Dominant faults
 - ❖ 1 vector (01 or 10) with 2 outputs (A' and B')
 - Harder to detect than Wired-AND/OR (*less observable*)
 - Detecting all dominant BFs ⇒ detects all Wired-AND/OR BFs

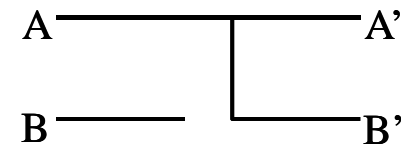


Wired-AND fault model

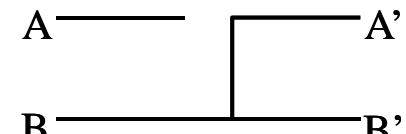


Wired-OR fault model

| AB | A'B' | WAND | WOR | A _{domB} | B _{domA} |
|----|------|------|-----|-------------------|-------------------|
| 00 | 00 | 00 | 00 | 00 | 00 |
| 01 | 01 | 00 | 11 | 00 | 11 |
| 10 | 10 | 00 | 11 | 11 | 00 |
| 11 | 11 | 11 | 11 | 11 | 11 |



A dominates B model

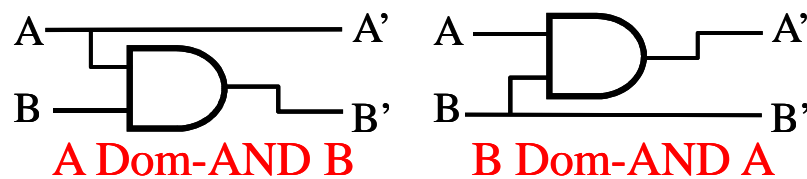


B dominates A model

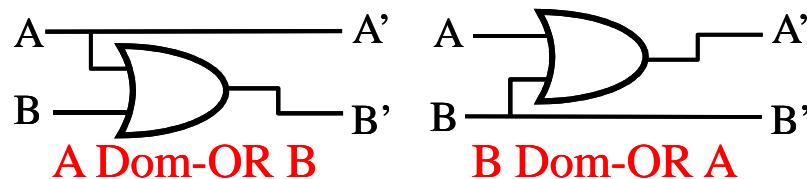
A Newer Bridging Fault Model

- Dominant-AND/Dominant-OR (aka 4-way BF model):
 - ❖ Stronger driving gate dominates short for only one logic value
 - Behavior has been observed in ASICs and FPGAs
 - ❖ Disadvantage: 4 faults per fault site
 - 2 vectors (01 and 10) with 2 outputs (A' and B')
 - ❑ Harder to detect than Wired-AND/OR BF (*less observable*)
 - ❖ Detecting all dominant-AND/OR BFs \Rightarrow detects all dominant BFs and all Wired-AND/OR BFs

Dominant-AND fault model



Dominant-OR fault model



| AB | A'B' | A _{and} B | B _{and} A | A _{or} B | B _{or} A |
|----|------|--------------------|--------------------|-------------------|-------------------|
| 00 | 00 | 00 | 00 | 00 | 00 |
| 01 | 01 | 00 | 01 | 01 | 11 |
| 10 | 10 | 10 | 00 | 11 | 10 |
| 11 | 11 | 11 | 11 | 11 | 11 |

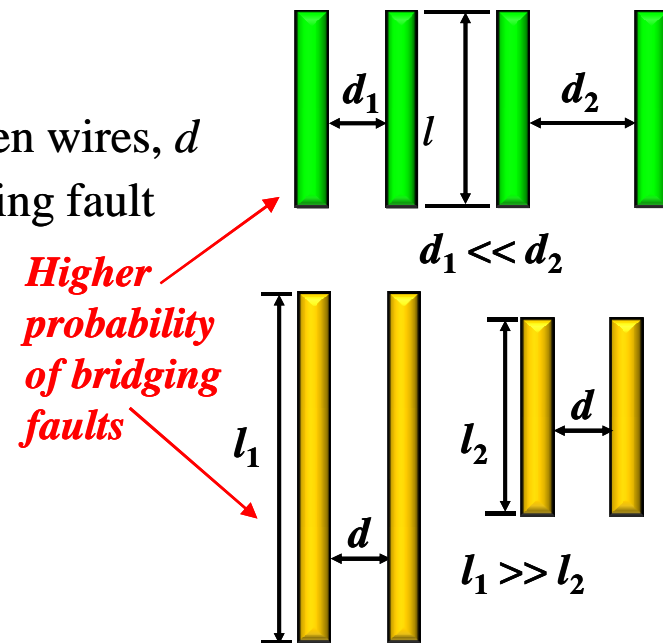
Bridging Fault Extraction

- Considering all possible shorts between any two wires is not practical
 - ❖ Large number of faults to simulate
 - For N wires, number of possible fault sites = N -choose-2 = $(N^2-N)/2$
 - ❑ # faults = N^2-N for dominant and wired-AND/OR bridging faults
 - ❑ # faults = $2(N^2-N)$ for dominant-AND/OR bridging faults
 - ❖ But wires at opposite sides of IC or PCB not likely to short
- Capacitance extraction identifies those wires most likely to short
 - ❖ Parallel plate capacitance, $C = \epsilon A/d$
 - Proportional to area of parallelism, A
 - Inversely proportional to distance between wires, d
 - Also proportional to probability of bridging fault

- ❖ Weighted fault coverage FC_{WBF}

- More accurate FC

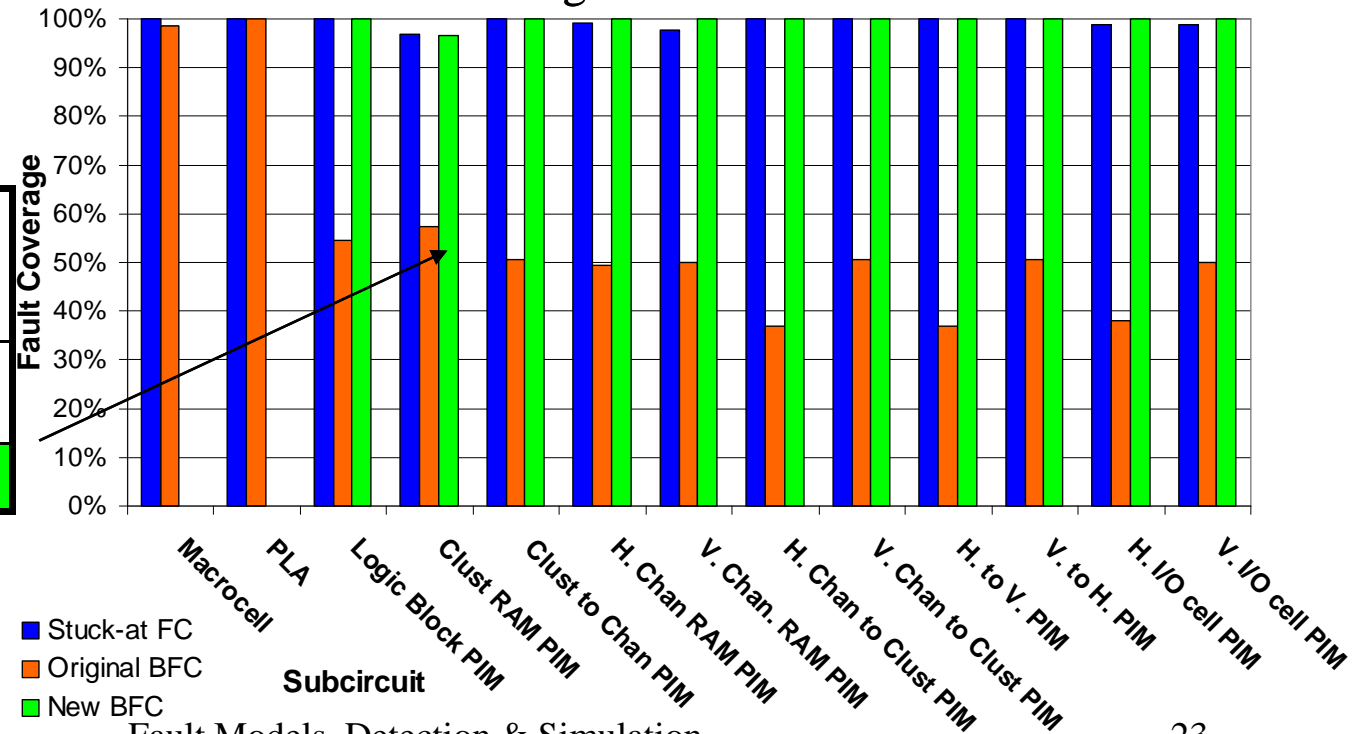
$$FC_{WBF} = \frac{\sum_{i=1}^D C_i}{\sum_{j=1}^T C_j}$$



Bridging Fault Detection

- High gate-level stuck-at FC traditionally assumed to give high BFC
 - ❖ Shown to be true for general sequential logic
 - ❖ Shown not true for non-decoded multiplexer-based routing resources
 - Stroud, et. al. “Bridging Fault Extraction from Physical Design Data for Manufacturing Test Development”, ITC’00
 - Cypress Delta 39K CPLD manufacturing tests

| Bridging Fault Coverage | Original Test Vectors | New BF Test Vectors |
|-------------------------|-----------------------|---------------------|
| Un-weighted | 57.5% | 96.5% |
| Weighted | 29.9% | 99.994% |



Fault Detection

Recall:

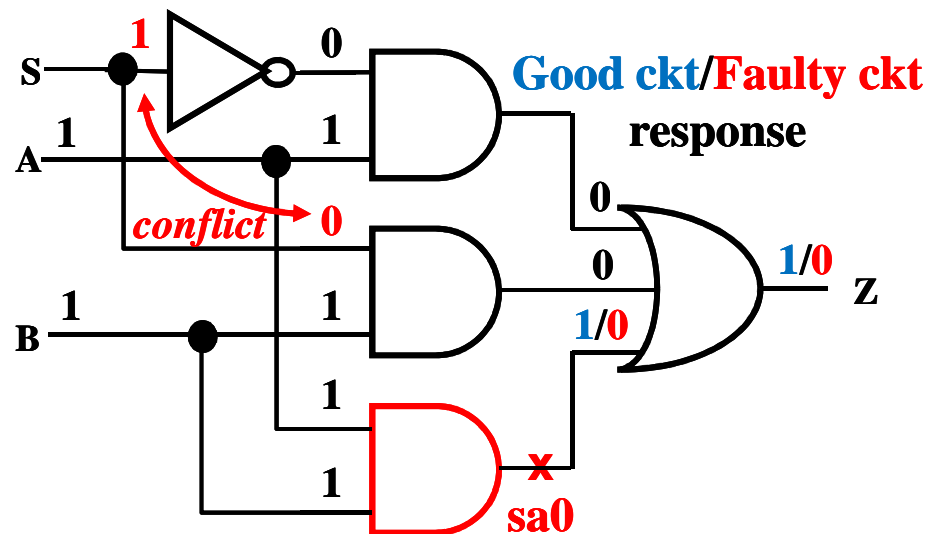
- Fault detection requires:
 - ❖ observation of an error (from fault) at a primary output
 - *observability* of the fault site
 - the ease at which we can observe the fault behavior
 - ❖ input stimuli that creates an error as a result of fault
 - *controllability* of the fault site
 - the ease at which we can control the fault behavior
 - *controllability* of path from fault site to primary output
 - typically considered part of observability
- But any given fault may be:
 - ❖ Detectable
 - ❖ Undetectable
 - ❖ Potentially detectable

Gate Level Fault Detection - Path Sensitization

1. At fault site, assign logic value opposite that of stuck-at fault
2. From fault site, choose a path to a PO assigning non-controlling values to all other inputs to gates in that path
 - 1 = non-controlling value for AND/NAND gates
 - 0 = non-controlling value for OR/NOR gates
3. For all assigned values, back-trace to PIs selecting input values along the way that will produce the assigned values
4. If there is a conflict, repeat Steps 2 & 3 choosing new paths and/or values in Step 3
5. If no path can be found without conflict, the fault *may be* undetectable, otherwise values at PIs form test vector

Undetectable Faults

- No test vector can detect the fault
 - ❖ Usually difficult to prove a fault is undetectable
- Undetectable faults due to:
 - ❖ Re-convergent fan-out, *and*
 - ❖ Redundant logic



Hazard-free multiplexer has undetectable faults due to re-convergent fanout and *redundancy*

| | | AB | | | |
|---|---|----|----|----|----|
| | | 00 | 01 | 11 | 10 |
| S | 0 | 0 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 1 | 0 |

Undetectable Faults (cont.)

- Minimize circuit to remove redundancy & these faults
 - ❖ Undetectable fault may not effect circuit operation
 - Wasted area
 - ❖ Undetectable fault slows down fault simulation
 - All test vectors are simulated (no fault dropping)
- Sometimes undetectable faults are unavoidable
 - ❖ Hazard-free circuits
 - Glitch-free clock multiplexing
 - ❖ Initialization circuitry
 - Power-up presets
 - Global resets
 - ❖ Use these circuits sparingly to minimize undetectable faults

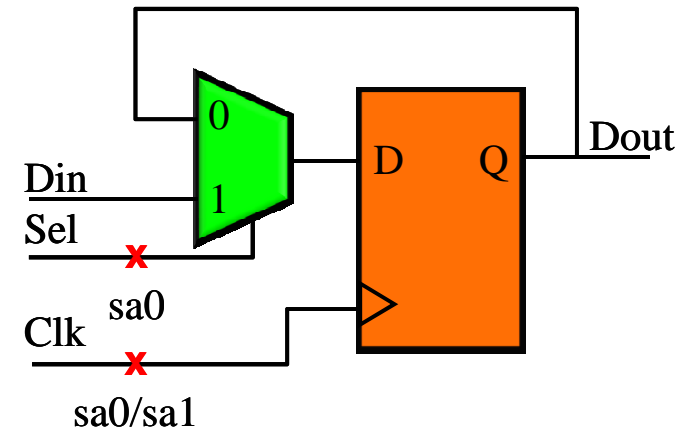
Potentially Detected Faults

- Due to undefined logic values ($U, X, 2, 3$)
 - ❖ An artifact of logic simulation
 - Can be a 0 or a 1 but simulator doesn't know
 - Used for un-initialized logic
 - ❖ Faults preventing initialization produce $U, X, 2, 3$
- Potential detect fault if good ckt = 1/0 & faulty ckt = $U, X, 2, 3$
 - ❖ Potential detect faults may be detected by other vectors
 - ❖ Probability of detection of a fault \propto # potential detects
 - For high data activity, otherwise probability = 0.5
 - ❖ In real circuit, fault may/may not be detected
 - Depends on power-up value & data activity
 - ❖ In simulation, PDFs also show up as undetected faults

Potentially Detected Faults (cont.)

Examples of potentially detected fault

- Select input to MUX stuck-at-0
 - ❖ Flip-flop cannot be initialized
- Clock stuck-at-0 and stuck-at-1
 - ❖ Flip-flop cannot be initialized
- 3 potentially detected faults
 - ❖ High detection probability for
 - High data activity on Sel & Din
 - High clock frequency
 - ❖ Otherwise these may not be detected



Fault Coverage Revisited

- Given a set of test vectors, each fault in fault set can be:
 - ❖ $D = \text{detected faults}$
 - Targeted faults and faults “accidentally” detected
 - ❖ $X = \text{undetectable faults}$
 - There are **NO** test vectors that can detect these faults
 - ❖ $U = \text{undetected faults}$
 - Could not find vector to detect faults (*but there could be some*)
 - ❖ $P = \text{potentially detected faults (PDF)}$
 - Also included in U
 - ❖ $T = \text{total faults} = D + X + U$
- Fault coverage = $(D + P/2) / T$
 - ❖ Detectable FC = $(D + P/2) / (T - X)$
 - Note: assumes PDF detection probability = 0.5

Testability Tidbits

- A set of test vectors is said to be N -detectable if all faults are detected at least N times with N different test vectors
 - ❖ The higher the value of N the higher the fault coverage for *other* fault models
- In the absence of global presets/resets (my experience):
 - ❖ testability \propto initializability
 - ❖ difficult to initialize subcircuits \Rightarrow difficult to test
 - ❖ therefore, logic simulation gives early indication of difficult to test portions of design
 - logic simulation provides cheap testability analysis
- Logic gates are 100% testable & easily testable
 - ❖ Therefore, testability problems come from design
 - the way designers interconnect the gates
 - ❖ Hence, the need for Design for Testability