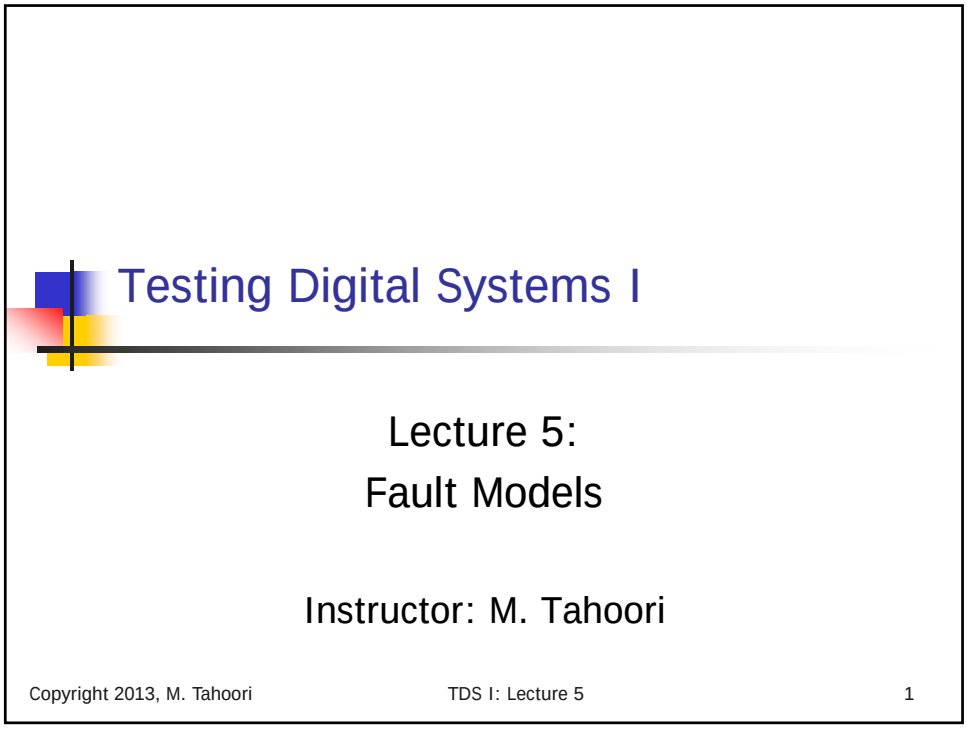


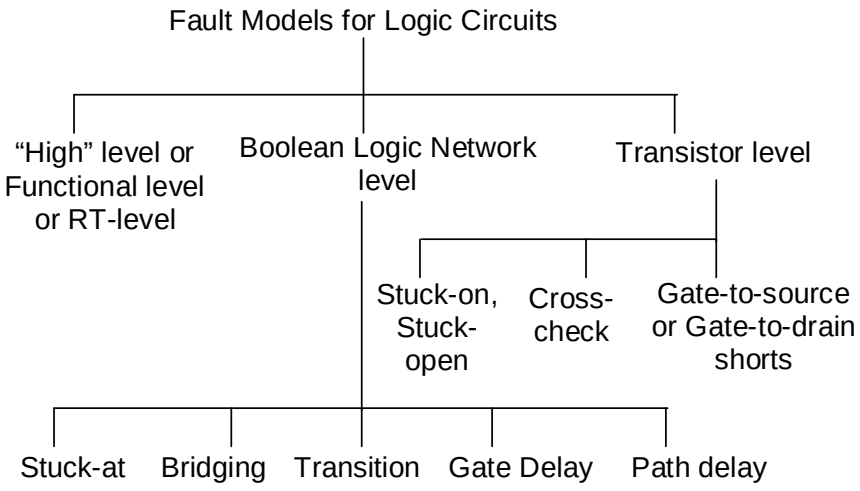
Introduction

- Digital logic networks tested by
  - applying input signals, called **test patterns or vectors,**
  - analyzing the resulting output response.
- The thoroughness and cost of the test depend on the particular test patterns applied.
  - Choosing which input signals to use as test patterns is critically important for the success of the test
- Fault models are used
  - To guide the test pattern selection process.
  - As the bases for **test metrics** which quantify the thoroughness of the test patterns
  - **Diagnosis**, the determination of the location of the defect causing the chip to fail the test.

Fault Model

- Fault model
  - Models effect of physical failure on logic network
    - Abstraction of physical situation
    - Used to describe the change in the logic function of a device caused by the defect.
- Various levels of abstraction are used
  - Functional (Board, Chip) level
  - Register transfer (Behavioral) level
  - Logic level
    - Gate library level
    - Elementary gate level
  - Switch level
  - Transistor (Spice) level

Fault Model Taxonomy



Fault Model Taxonomy (cont)

- Transistor-level fault models
  - More accurate than logic-level fault models
  - complexity of handling all transistor-level faults can be huge
    - may not be manageable by existing CAD tools.
  - Some transistor-level faults are important for modeling defects that can cause timing failures and/or early-life failures
    - transistor gate-to-source and gate-to-drain shorts
- High-level fault models
  - having no knowledge about the actual gate-level representations of logic blocks
  - not very effective in detecting manufacturing defects

Logic Level Faults

- Why needed?
  - I/O function tests inadequate for manufacturing testing
    - No automation for design verification vectors
  - Real defects too numerous and often not analyzable
    - Fault abstractions reduce the number of conditions that must be considered in deriving tests
  - A fault model identifies targets for testing
    - test set generation
  - A fault model makes analysis possible
    - test set evaluation

Fault Model Effectiveness

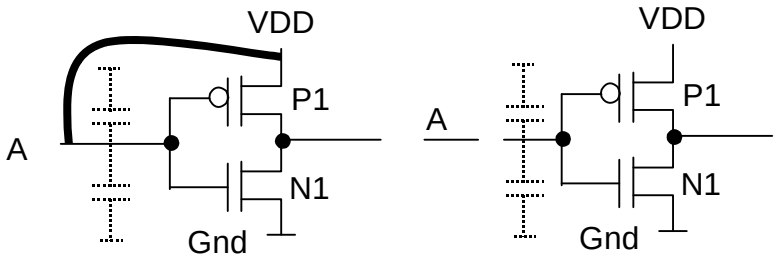
- Effectiveness of a fault model
  - The effectiveness of test patterns generated using the fault model in detecting defective parts.
  - The accuracy with which it represents the effects of failures.
  - Its tractability as a design tool.
  - Scalability of its complexity with the increasing size of VLSI circuits.
  - Its usefulness in determining the location of a defect on a chip.

Stuck-at Fault Model

Stuck Fault Models

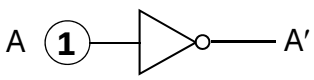
- Structural logic-level fault model
  - Start with the circuit represented as a netlist of Boolean gates
  - Assumes faults only affect the interconnection between gates
- Single Stuck Fault
  - Logic network of elementary gates
    - AND, OR, NAND, NOR, NOT
  - One Line has Fixed 0 or 1 Value
  - Independent of other signal values
  - One fanout branch can be stuck
  - Most common model for Boolean test
  - Written L_i / h, h = 0 or 1
- Multiple Stuck Fault
  - One or More Stuck Line Faults Present
- Pin Fault
  - Stuck Fault on I/O Connection of a Module

Stuck-at Fault Model



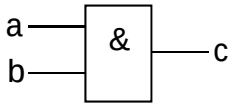
Short to VDD

Open signal lead



Notation: **A/1** or **A_1**

Stuck-at Fault Table for AND Gate



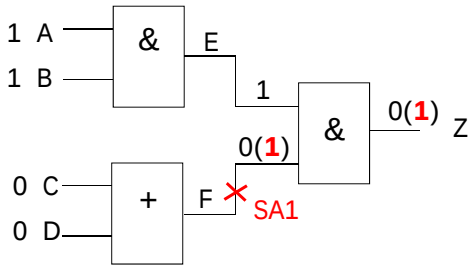
Input	Fault-free	Output Value with Stuck-at Fault					
a b	Output	a/0	a/1	b/0	b/1	c/0	c/1
0 0	0	0	0	0	0	0	1
0 1	0	0	1	0	0	0	1
1 1	1	0	1	0	1	0	1
1 0	0	0	0	0	1	0	1

Fault Detection

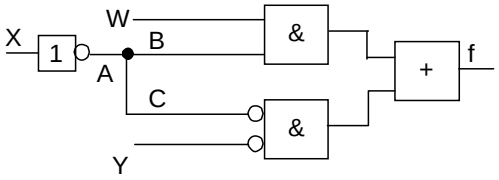
- An input combination *detects* a fault in a logic network if
  - the response of the faulty logic network to that input combination is different from that of the fault-free network
  - The input combination is called a *test pattern* for the fault
- Fault detection requires:
  - A test t activates or provokes the fault f.
  - t propagates the error to an observation point
    - e.g. primary output
- A line whose value changes with f present is said to be sensitized to the fault site.

Single Stuck-at

- 14 faults
  - 2 faults (SA0, SA1) per each line
- ABCD = 1100 detects F/1
  - Faulty and fault-free outputs different
- ABCD = 1101 does NOT detect F/1
  - Faulty and fault-free outputs are the same



Stuck-Fault Table for Fanout Faults



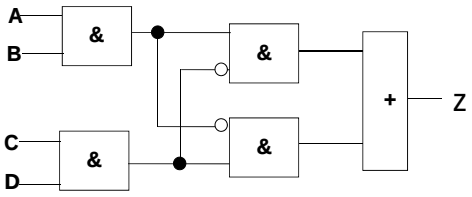
Input *W X Y*	Fault-free Output	Output Value with Stuck-at Fault					
		A/0	*B*/0	*C*/0	*A*/1	*B*/1	*C*/1
0 0 0	0	1	0	1	0	0	0
0 0 1	0	0	0	0	0	0	0
0 1 0	1	1	1	1	0	1	0
0 1 1	0	0	0	0	0	0	0
1 0 0	1	1	0	1	1	1	1
1 0 1	1	0	0	1	1	1	1
1 1 0	1	1	1	1	1	1	0
1 1 1	0	0	0	0	1	1	0

- *W X Y* = 1 0 1 and 1 1 1 detect all SAF on *A*
  - but neither of them detects *C*/0 or *C*/1

Multiple Stuck-at Faults

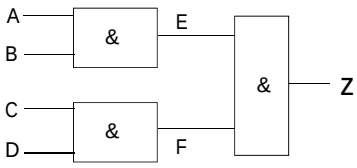
- A multiple stuck-at fault means that any set of lines is stuck-at some combination of (0,1) values.
- The total number of single and multiple stuck-at faults in a circuit with k single fault sites is $3^k - 1$
  - The number of single stuck-at faults is $2K$
- A single fault test can fail to detect the target fault if another fault is also present
  - however, such masking of one fault by another is rare.
- Statistically, single fault tests cover a very large number of multiple faults.

Multiple Stuck-at Faults



- all single stuck-at faults detected by
  - {*ABCD* = 0111, 1101, 1111, 1010}
- Multiple stuck-at faults (*B*/1, *D*/1)
  - Only ABCD=1010 from above set provokes (B/1, D/1)
    - Z = 0 in faulty and fault-free cases
  - **Multiple faults not detected!**

Multiple Stuck-at Faults



- ABCD = 1010
  - E/1 not detected
  - F/1 not detected
  - Multiple faults (E/1,F/1) detected

Multiple Faults

- Exhaustive Simulation of 181 ALU
  - 14-input ALU
  - All 79,600 Double Faults
  - 16 different Single-stuck Fault Test Sets
  - Minimum Double Stuck Fault Coverage 99.963 %
- L lines
  - 2L Single-Stuck Faults
  - $2^2C(L,2)$ Double-Stuck Faults = 2L(L-1)
    - number of m-stuck-at faults = $2^mC(L,m)$

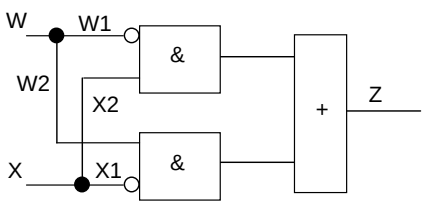
Double Faults in 181 ALU

- All set achieve 100% single stuck-at coverage
- Lowest Double-Stuck-at Fault Coverage is
  - 1 - 30/79,600 = 99.963 %

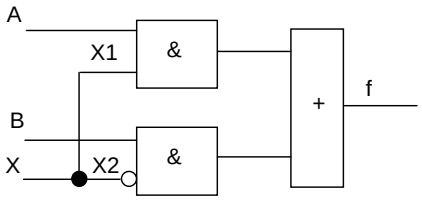
Test Set	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Length	12	12	12	12	12	12	12	14	14	14	14	17	35	124	135	352
Undetected Double-stuck faults	9	8	1	9	28	13	19	4	14	11	3	30	0	0	0	0

PIN Faults

- Exclusive-OR gate
  - 6 Pin faults
    - W/0, W/1, X/0, X/1, Z/0 and Z/1
  - 100% pin fault coverage
    - {*WX* = 00, 01, 10} or {*WX* = 01, 10, 11} or
    - {*WX* = 00, 01, 11} or {*WX* = 00, 10, 11}
  - 100% flattened fault coverage
    - Requires all 4 vectors: {*WX* = 00, 01, 10,11}



Pin Faults on 2-bit Mux



Input X A B	Fault-free Output	Output Value with Stuck-at Fault									
		X/0	X/1	A/0	A/1	B/0	B/1	f/0	f/1	X1/1	X2/1
0 0 0	0	0	0	0	**1**	0	0	0	**1**	0	0
0 0 1	0	0	**1**	0	**1**	0	0	0	**1**	0	**1**
0 1 0	1	1	**0**	**0**	1	1	1	**0**	1	1	1
0 1 1	1	1	1	**0**	1	1	1	**0**	1	1	1
1 0 0	0	0	0	0	0	0	**1**	0	**1**	0	0
1 0 1	1	**0**	1	1	1	**0**	1	**0**	1	1	1
1 1 0	0	**1**	0	0	0	0	**1**	0	**1**	**1**	0
1 1 1	1	1	1	1	1	**0**	1	**0**	1	1	1

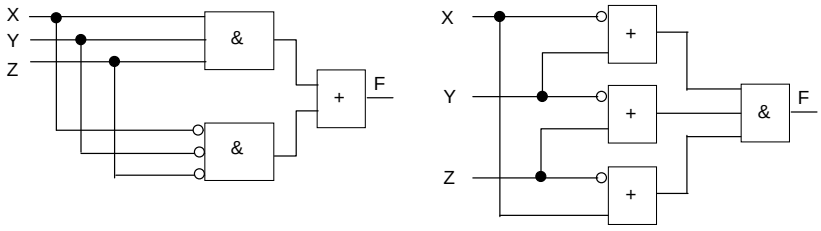
Pin Faults

- Fault coverage for 2-to-1 MUX
  - Different implementations

Test Set	AND-OR		OR-AND	
	Single Stuck	Pin faults	Single Stuck	Pin faults
$S1$ = {XAB = 001, 011, 110, 111}	100%	100%	80%	100%
$S2$ = {XAB = 000, 010, 100, 101}	80%	100%	100%	100%

Diverse Implementation

- F = XYZ + X′Y′Z′
- Left Implementation (A)
  - 100% SSF requires all 8 vectors
- Right Implementation (B)
  - 100% SSF requires only 4 vectors (*XYZ* = 000,010,100,111)



(A) (B)

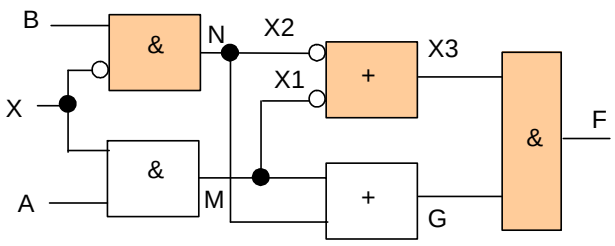
Untestable Faults

Untestable Fault

- A fault that does not affect the logical behavior of a circuit (redundant fault)
  - Untestable by Particular Test Procedure
- Causes
  - Redundant Circuitry
    - Design Error
  - Hazard Control Circuitry
  - Error Detection Circuitry
    - Parity Check
  - Excess Components
    - Needed for Performance, not Functionality

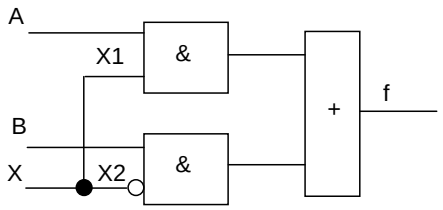
Untestable Fault

- Unexpected redundancy
  - the fault can occur in some portion of the circuit that is redundant:
    - it has no effect on the circuit function
- X3/1, X1/0, X2/0 untestable
  - Either M or N always 0 ⇒ X3 always 1
  - Output can be taken from G instead of F



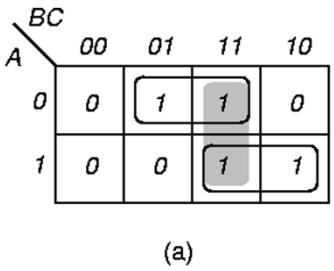
Internal Signal Dependencies

- Cannot set 1 on both inputs to the OR gate
- Untestable fault
  - OR changed to XOR

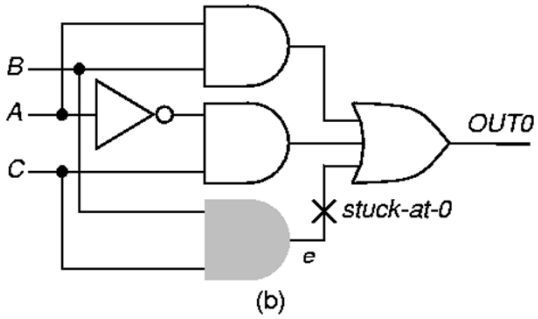


Untestable Fault

- Hazard Elimination Redundancy
  - Problem: Output changes when A changes while BC = 11
  - Solution: Intentional Redundant Implicant *BC*
- Untestable fault: e/0



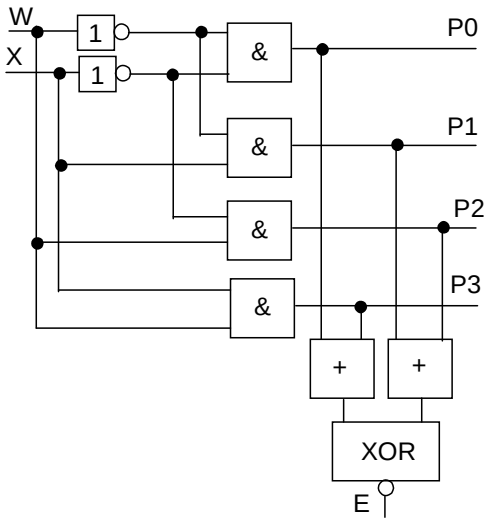
(a)



(b)

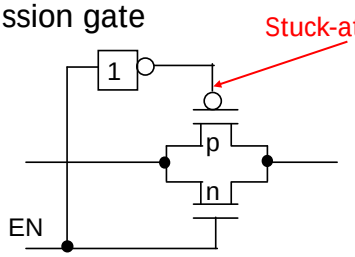
Error Detection Redundancy

- Fault-free decoder
  - Only one output = 1
  - E = 0
- E/0 untestable
  - If no fault in decoder

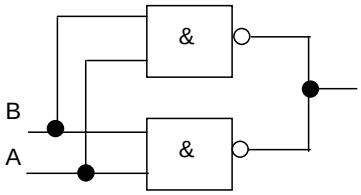


Redundant Transistors and gates

- CMOS transmission gate

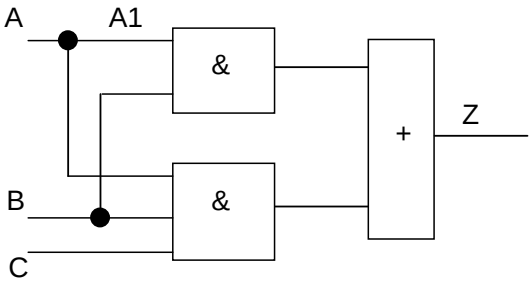


- Extra drive NAND gate



Untestable Faults

- C/1 untestable
  - Z = *AB* + *ABC* = *AB*
- Testable fault *A*1/0 (ABC = 110)
  - Untestable in the presence of the untestable fault *C*/1.



Bridging Faults

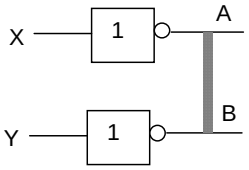
Bridging Fault Models

- Logical Fault Model
- Normally Distinct Signal Lines Shorted Together
  - Valid logic levels preserved
  - Restricted to Signal Lines
  - Short may be resistive
- Excludes
  - Internal gate shorts
  - Shorts between signal line and power rail

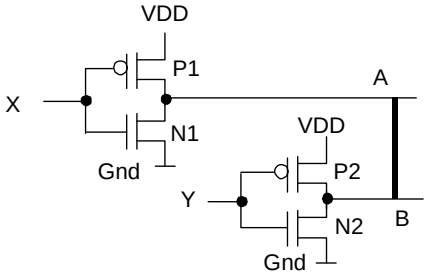
Definitions

- *Bridging faults* appear when two or more normally distinct signal lines in a Boolean logic network are unintentionally shorted together and create wired logic.
- A *feedback bridging fault* is a special type of bridging fault which is created when one of the two shorted signal lines depends on the other signal line in the fault-free circuit.
  - May cause oscillation or latch
- If a fanout branch of a signal line is involved in a bridge
  - Logic value on the fanout stem and the other fanout branches of that signal line will be the same as the logic value on the fanout branch which is involved in the bridge

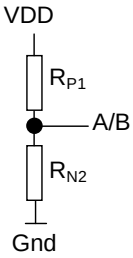
Bridging Fault



Logic-level model



Transistor-level model



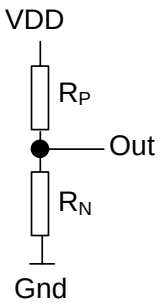
Electrical model when X = 0 and Y = 1

Wired Logic Bridging Fault Models

- Simplest bridging fault model
- Lines A and B connected
  - Both A and B
    - Same logic signal value
  - Wired AND (AND-bridging fault)
    - Signal value = AND function of fault-free values
    - TTL or CMOS technologies
  - Wired OR (OR-bridging fault)
    - Signal value = OR function of fault-free values
    - ECL or CMOS technologies

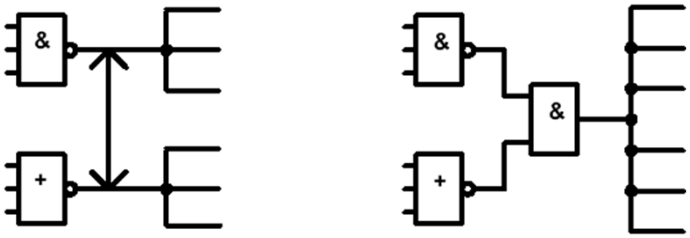
Wired Bridging Fault Model

- Vout = VDD×R_N/(R_N+R_P)
  - Output depends on relative sizing and strength
  - R_N >> R_P
    - Wired-OR
  - R_N << R_P
    - Wired-AND



Wired AND Bridging Fault

- Non-feedback AND Bridge Logical Model



Other Bridging Fault Models

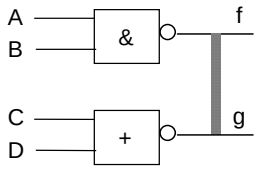
- Lines A and B connected
  - Both A and B
    - Same logic signal value
- Dominating signal model
  - Signal value based on driving gate types
- Voting model
  - Signal value based on relative drive strengths of A and B
    - depends on inputs to driving gates

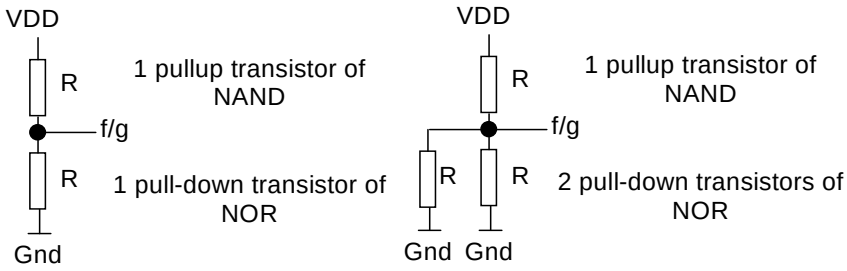
Other Bridging Fault Models

- Logic Behaviors of two signal lines *A* and *B* under various Bridging Fault Models

Fault-free Behavior	Faulty Behaviors			
	Wired-AND	Wired-OR	A-Dominant	B-Dominant
A B	A B	A B	A B	A B
0 0	0 0	0 0	0 0	0 0
0 1	0 0	1 1	0 0	1 1
1 0	0 0	1 1	1 1	0 0
1 1	1 1	1 1	1 1	1 1

Voting Bridging Fault Model





Voltage divider when ABCD = 0110

Voltage divider when ABCD = 0111

Which Bridging Fault Model to Use?

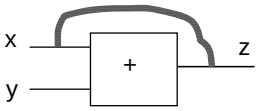
- based on the bridging fault simulation data on the AMD-K6 microprocessor
  - error introduced due to use of computationally less expensive and less accurate bridging fault models is very little
    - By using a less accurate bridging fault model we may overestimate the number of detected shorts by 1%

Other Bridging Fault Models

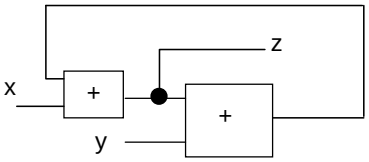
- Feedback Bridging Fault
  - One signal depends on the other signal (fault-free)
    - Can construct asynchronous feedback loop
      - Additional State
    - Can Construct a Latch
      - Additional State
    - Can Cause Oscillation

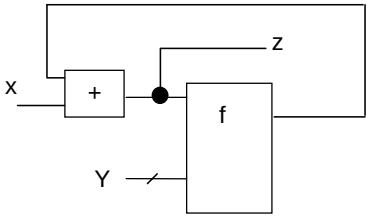
Feedback Bridging Fault

OR-gate with OR-type feedback BF



Logical model for the feedback BF





Model for OR feedback BF in a general combinational circuit

Feedback Bridging Fault

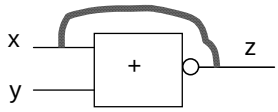
- Logic Behavior of an OR-gate with OR-type Feedback Bridging Fault

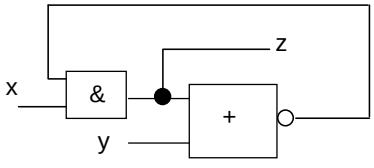
Input Sequence	Inputs x y	Fault-free Output	Faulty Output
Sequence 1	0 0	0	0/1 *
	0 1	1	1
	1 0	1	1
	1 1	1	1
Sequence 2	0 1	1	1
	0 0	0	**1**
	1 0	1	1
	1 1	1	1

* Depends on the initial state

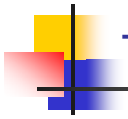
Feedback Bridging Fault

- Oscillation from feedback bridging fault.
  - NOR-gate with AND-type feedback bridging fault
    - xy = 00 followed by 10



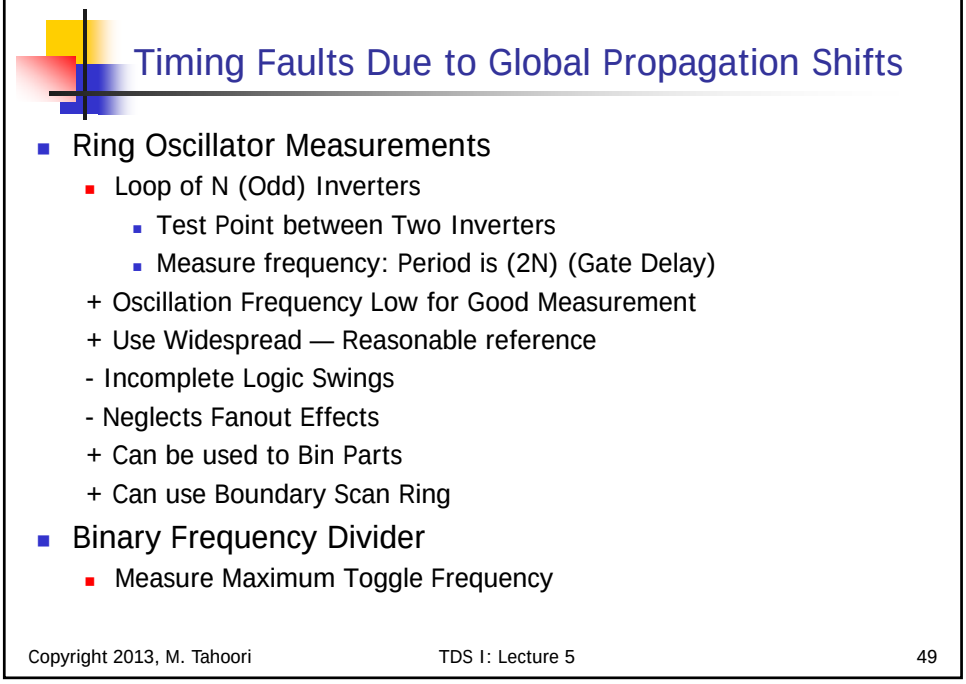


Timing and Delay Faults



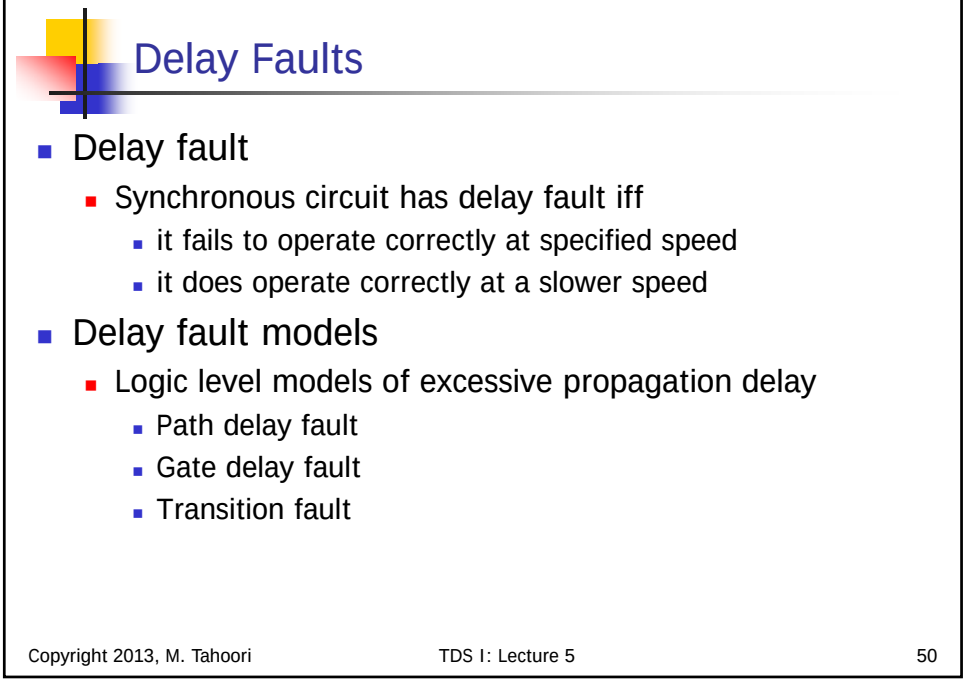
Timing Failures

- Logic Network has a Timing Failure if and only if
  - it fails to operate correctly at its specified speed
- BUT
  - may produce correct outputs when operated at either
    - a slower or faster speed
- Compared to Stuck-at or bridging fault models
  - Static faults
    - Incorrect values at any speed
- Sequential Circuit Causes of Timing Failures
  - excessive propagation delay
    - setup time (long path) violation
  - inadequate propagation delay
    - hold time (short path) violation



Timing Faults Due to Global Propagation Shifts

- Ring Oscillator Measurements
  - Loop of N (Odd) Inverters
    - Test Point between Two Inverters
    - Measure frequency: Period is (2N) (Gate Delay)
  - \+ Oscillation Frequency Low for Good Measurement
  - \+ Use Widespread — Reasonable reference
  - \- Incomplete Logic Swings
  - \- Neglects Fanout Effects
  - \+ Can be used to Bin Parts
  - \+ Can use Boundary Scan Ring
- Binary Frequency Divider
  - Measure Maximum Toggle Frequency



Delay Faults

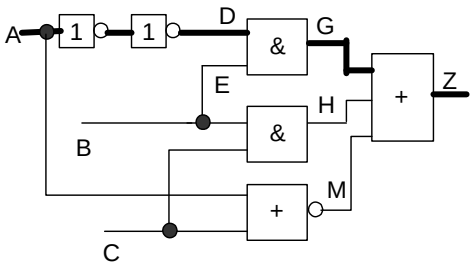
- Delay fault
  - Synchronous circuit has delay fault iff
    - it fails to operate correctly at specified speed
    - it does operate correctly at a slower speed
- Delay fault models
  - Logic level models of excessive propagation delay
    - Path delay fault
    - Gate delay fault
    - Transition fault

False path

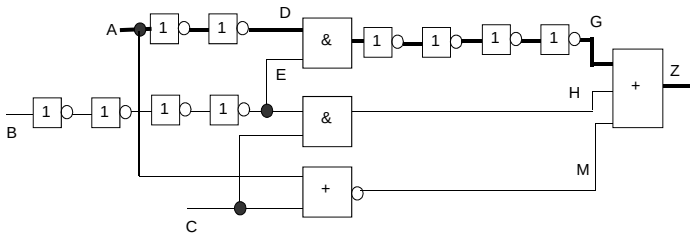
- A path from the input to the output of a combinational circuit is a false path
  - if it does not affect the operation of the circuit.
- A false path is not sensitizable under any timing conditions

False path

- Example
  - Z = AB + BC + A′C′
  - All gates unit 1 delay
  - path B-H-Z not a false path
    - A = 0 and C = 1
  - path C-M-Z not a false path
    - A = B = 0
  - statically sensitizable paths
  - path A-D-G-Z is a false path
    - Must set E = 1 at time t = 2, and H = M = 0 at time t = 3
    - If C = 0
      - For 1-to-0 transition on A,
        - there will be a 0-to-1 transition on M at time t = 1
      - transition on A will not be propagated
    - If C = 1
      - E and H settle to the same value at time t = 0 and 1



False path



- Same function: Z = AB + BC + A′C′
- 1 to 0 transition on A along the path A-D-G-Z
  - E = 1 at time t = 2 and H = M = 0 at time t = 7
    - C = 1 making M = 0
    - let A and B transition from 1 to 0 at time t = 0
- Not a false path!

Sensitization

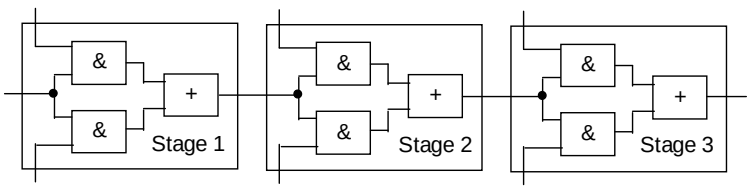
- Dynamic sensitization
  - Previous example
- Static sensitization
  - During static sensitization of a path, we try to find an input combination such that an event can propagate along a path after all signals have settled down
    - All side inputs of a NAND/AND gate through which a path passes are 1
    - All side inputs of a NOR/OR gate through which a path passes are 0
  - Static sensitization is not a necessary condition for a path to be sensitizable
    - But is a sufficient condition

Path Delay Fault

- Path delay fault present
  - propagation delay of at least one path
    - from primary input to primary output exceeds clock interval
  - Models multiple or distributed defects
  - Issue: can path be sensitized, occur in operation?
- Each path delay fault
  - associated with a particular path
    - between primary input and output
  - either
    - all paths
    - all sensitizible paths
    - longest paths
      - static timing analyzer

Path Delay Fault Model Problems

- Number of paths could be too large
  - Example : iterative logic array
    - Number of paths from primary inputs to primary outputs
      - A circuit with n stages will have $3 \times 2^n - 2$ paths
- A sensitizable path in the test mode
  - may not be sensitized during normal operation



Gate Delay Fault

- Definition
  - a localized timing failure at a gate causes the propagation delay of at least one path in the circuit through the fault site to exceed the specified cycle time
- Each gate delay fault
  - Gate has delay that causes incorrect circuit operation
  - Delay of some path through gate
    - exceeds clock interval

Transition Fault

- Definition
  - A localized timing failure is large enough such that the delay of all paths through some gate to observable outputs exceed the clock interval
- Each transition fault
  - associated with a particular gate input or gate output
  - either a 0 to 1 transition or a 1 to 0 transition
    - (two transition faults)
      - slow-to-rise, slow-to-fall
  - propagated to some primary output

Transition Fault

- two input combinations are needed
  - initialization pattern
    - places an initial value at the fault site.
      - The initial value is 0 for a slow-to-rise transition fault,
      - and 1 for a slow-to-fall transition fault.
  - transition propagation pattern
    - places the final transition value
      - 1 for a slow-to-rise transition fault, and 0 for a slow-to-fall transition fault
    - propagates the transition to an observable output

Transition Fault: NAND gate

			Faulty Outputs					
Pattern	Inputs	Fault-free	Slow-to-rise Transition Faults			Slow-to-fall Transition Faults		
Sequence	A B	Output Z	A	B	Z	A	B	Z
1	0 0	1	1	1	1	1	1	1
	0 1	1	1	1	1	1	1	1
2	0 0	1	1	1	1	1	1	1
	1 0	1	1	1	1	1	1	1
3	0 0	1	1	1	1	1	1	1
	1 1	0	**1**	**1**	0	1	1	**1**
4	0 1	1	1	1	1	1	1	1
	0 0	1	1	1	1	1	1	1
5	0 1	1	1	1	1	1	1	1
	1 0	1	1	1	1	1	**0**	1
6	0 1	1	1	1	1	1	1	1
	1 1	0	**1**	0	0	1	0	**1**
7	1 0	1	1	1	1	1	1	1
	0 0	1	1	1	1	1	1	1
8	1 0	1	1	1	1	1	1	1
	0 1	1	1	1	1	**0**	1	1
9	1 0	1	1	1	1	1	1	1
	1 1	0	0	**1**	0	0	0	**1**
10	1 1	0	0	0	0	0	0	0
	0 0	1	1	1	**0**	1	1	1
11	1 1	0	0	0	0	0	0	0
	0 1	1	1	1	**0**	**0**	1	1
12	1 1	0	0	0	0	0	0	0
	1 0	1	1	1	**0**	1	**0**	1

Transition Fault: Detection

- A rising (falling) transition fault is detected if and only if:
  - (1) The first input combination places a 0 (1) at the fault site; and,
  - (2) The second input combination detects a stuck-at-0 (stuck-at-1) fault at the transition fault site.

Different Delay Fault Models

- The transition fault model very practical
  - The test generation effort required for transition faults
    - Almost the same as the test generation effort required for stuck-at faults
  - Most commercial test pattern generation tools support the transition fault model
- Relation between different delay fault models
  - $\text{Stuck-at Fault} \subset \text{Transition Fault} \subset \text{Gate Delay Fault} \subset \text{Path Delay Fault} \subset \text{Delay Fault}$