Υπολογιστικά Συστήματα Υψηλής Αξιοπιστίας

Πανεπιστήμιο Δυτικής Αττικής

Ακαδημαϊκό έτος 2019-2020

Παρουσίαση 1η:

Εισαγωγή στον Έλεγχο Ολοκληρωμένων κυκλωμάτων (Integrated circuit Testing)

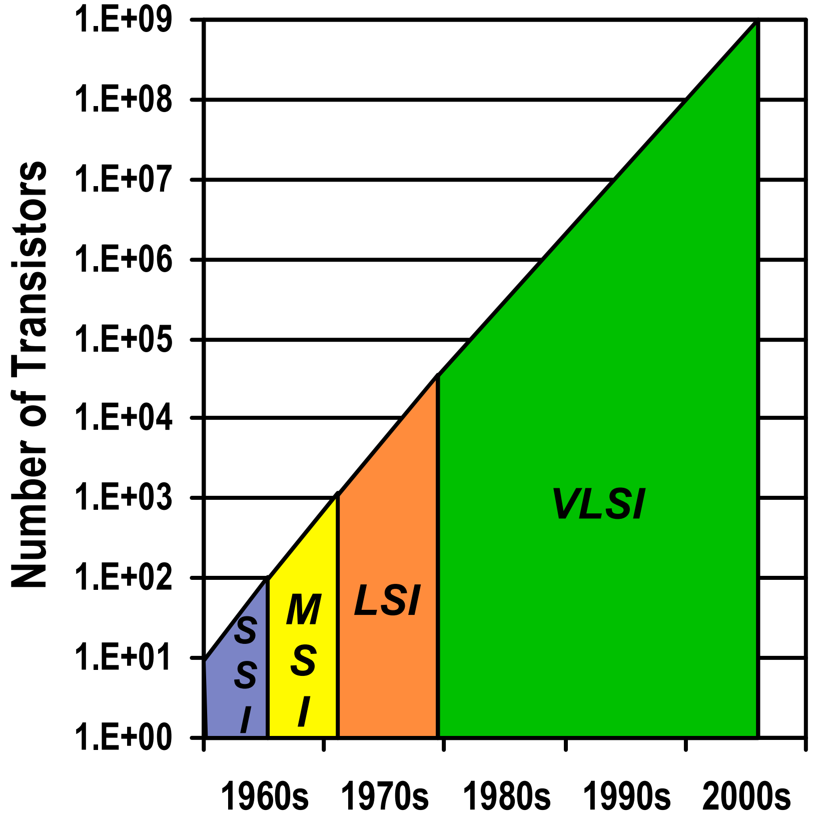
Σκοπός παρουσίασης

- Να παρουσιάσει θέματα που σχετίζονται με τον έλεγχο IC (IC testing) και τη σχετική τεχνολογία
- Έμφαση στα εξής:
  - Σημασία του ελέγχου στη διαδικασία σχεδιασμού και κατασκευής
  - Προκλήσεις στο test generation και το fault modeling

Περιεχόμενα

Introduction

- Integrated Circuits (ICs) have grown in size and complexity since the late 1950's
  - Small Scale Integration (SSI)
  - Medium Scale Integration (MSI)
  - Large Scale Integration (LSI)
  - Very Large Scale Integration (VLSI)
- *Moore's Law*: scale of ICs doubles every 18 months
  - Growing size and complexity pose new testing challenges



Σημασία του Ελέγχου (1)

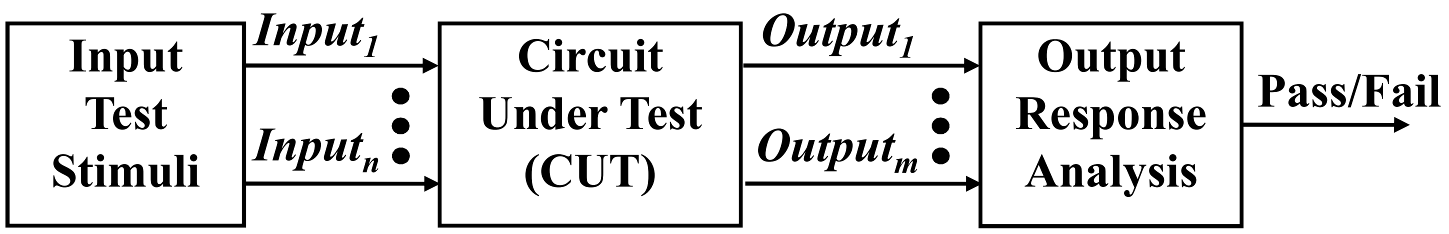
- Decreasing feature size (dimensions)
  - from μm => nm for
    - transistors
    - interconnecting wires
- Operating frequencies have increased
  - from 100KHz => several GHz
- Decreasing feature size increases probability of defects during manufacturing process
  - A single faulty transistor or wire results in faulty IC
- Hence, testing required
  - to guarantee fault-free products

Σημασία του Ελέγχου (2)

- *Rule of Ten*: cost to detect faulty IC increases **by an order of magnitude** as we move from:
  - device $\rightarrow$ PCB $\rightarrow$ system $\rightarrow$ field operation
  - Testing performed at each of these levels
- Testing required during:
  - Manufacturing
    - to improve yield
  - Field operation to:
    - ensure fault-free system operation
    - initiate repairs when faults are detected

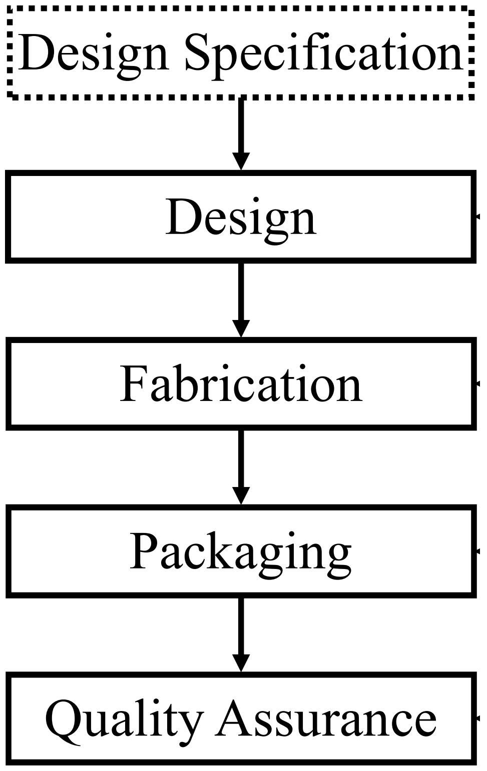
Ορισμός του Ελέγχου

- Testing consists of
  - Applying Input test stimuli to Inputs of *circuit under test* (CUT), and
  - Analyzing output responses
    - If correct (pass), CUT assumed to be fault-free
    - If incorrect (fail), CUT assumed to be faulty



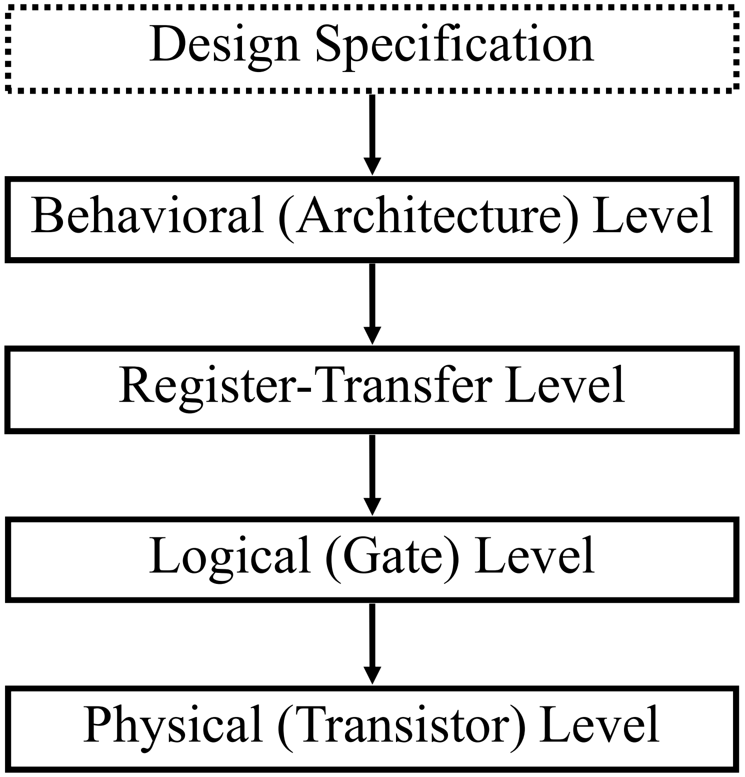
Κύκλος ανάπτυξης IC

- **Design error**
  - Discovered during design verification
  - Corrections made prior to fabrication
- **Manufacturing error (Defect)**
  - a flaw or physical imperfection that can lead to a fault
  - Discovered during Fabrication, Packaging, …



Design Verification

- Different levels of abstraction
- RTL to physical level
  - CAD tools
- Simulation used at various level to test
  - For design errors
  - If the design meets system timing requirements after synthesis



Yield and Reject Rate

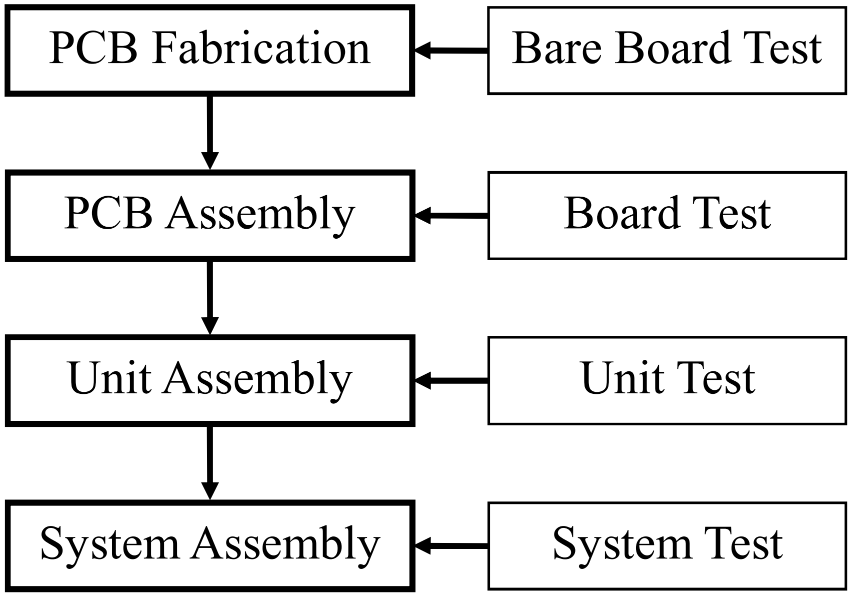
- ❑ Faulty chips due to manufacturing defects ☹
  - the % of good chips:

$$yield = \frac{number\ of\ acceptable\ parts}{total\ number\ of\ parts\ fabricated}$$

- ❑ Yield loss ☹
- ❑ 2 types of yield loss
  - Catastrophic – due to random defects
  - Parametric – due to process variations
- ❑ Undesirable situations during testing
  - Faulty chip appears to be good (passes test)
  - Good chip appears to be faulty (fails test)
    - – Due to overstress during test

PCBs & Electronic Systems

- A system consists of PCBs that consist of ICs
- PCB fabrication similar to IC fabrication
  - Susceptible to defects
- Assembly steps also susceptible to defects
  - Testing performed at all stages of manufacturing
- Τι είναι πιο εύκολο;
  - Ο έλεγχος των PCBs?
  - Ο έλεγχος των ICs?



Είδη ελέγχου

- Off-line testing
  - while system (or portion of it) is taken out of service
  - Performed periodically during low-demand periods
  - Used for diagnosis (identification and location) of faulty replaceable components to improve repair time
- On-line testing
  - concurrent with system operation

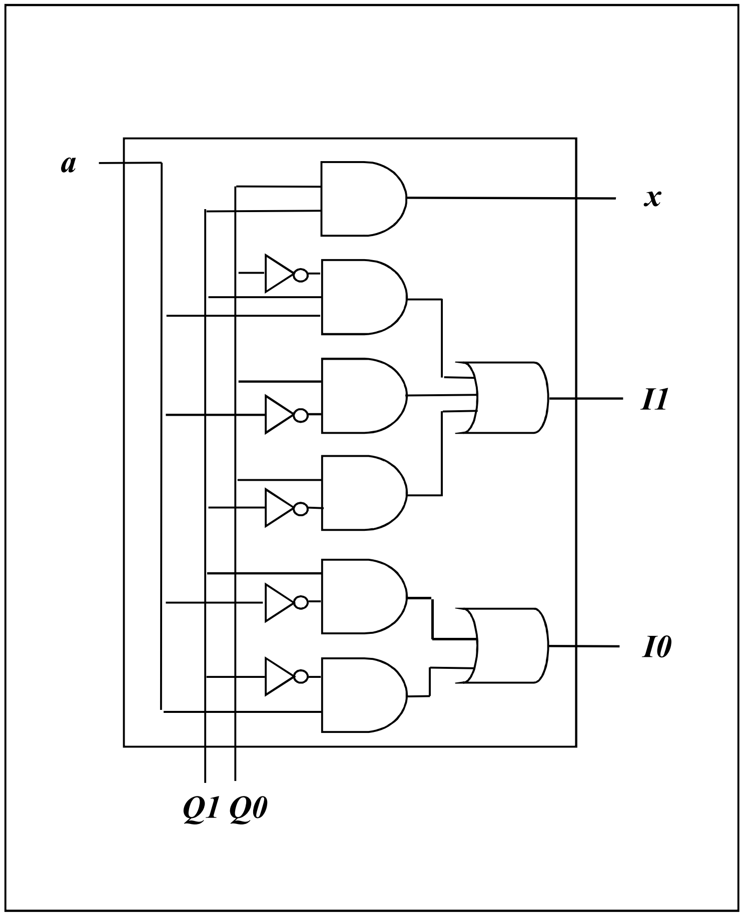
Test Generation

- During testing, a sequence of test patterns, called *test vectors*, is applied to the CUT whose outputs are monitored and analyzed for the correct response
- *Fault coverage* is a quantitative measure of the quality of a set of test vectors
- Brute-force idea: Exhaustive testing – applying all possible test patterns to CUT
  - Is it practical?

Quiz (1)

Έστω το ακόλουθο κύκλωμα.

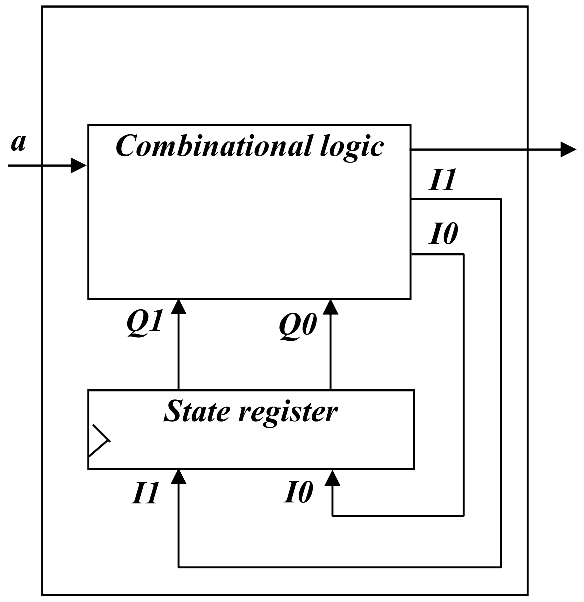
- Είναι συνδυαστικό ή ακολουθιακό;
- Πόσες εισόδους έχει;
- Πόσοι κύκλοι ρολογιού απαιτούνται για εξαντλητικό έλεγχο;



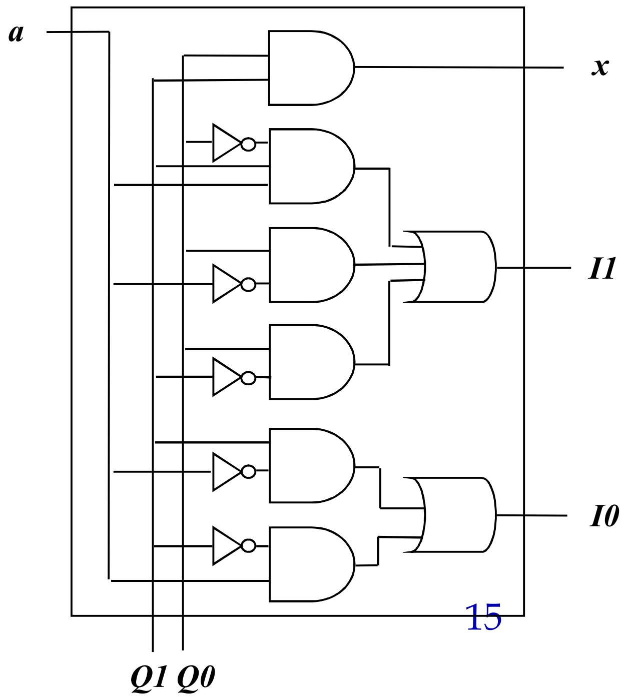
Quiz (2)

Έστω το κύκλωμα.

- Είναι συνδυαστικό ή ακολουθιακό;
- Πόσες εισόδους έχει;
- Σχεδιάστε το διάγραμμα καταστάσεων του κυκλώματος
- Πόσοι κύκλοι ρολογιού απαιτούνται για εξαντλητικό έλεγχο;



Combinational Logic



Quiz (3)

1. Έστω ένα συνδυαστικό κύκλωμα που λειτουργεί στα 100KHz. Πόσος χρόνος απαιτείται για εξαντλητικό έλεγχο αν έχει:
   - 10 εισόδους
   - 20 εισόδους
   - 60 εισόδους
2. Έστω ένα ακολουθιακό κύκλωμα που λειτουργεί στα 100KHz. Πόσος χρόνος απαιτείται για τον εξαντλητικό έλεγχο αν έχει:
   - 10 εισόδους και status register με 10 flip flop
   - 20 εισόδους και status register με 40 flip flop

Test Generation (1)

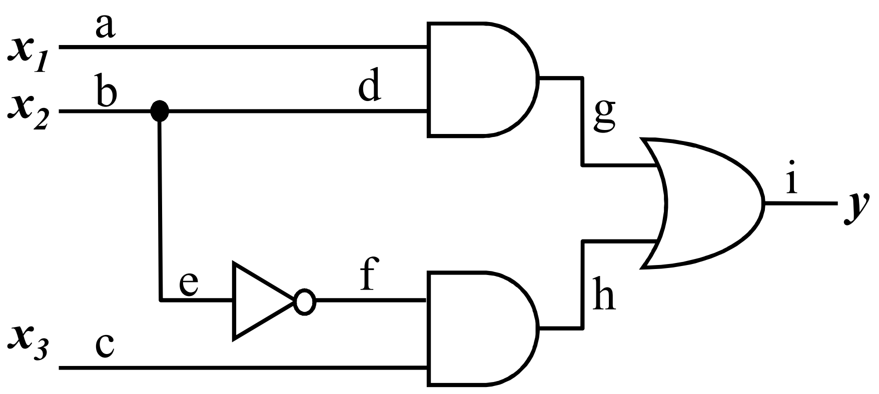
- *Goal*: Εύρεση συνόλου από set of test vectors με μέγιστη κάλυψη ελαττωμάτων (fault coverage)
  - Τι είναι η «**Κάλυψη ελαττωμάτων**»;
- Για να προσομοιώσουμε τη συμπεριφορά των ελαττωμάτων απαιτείται μοντέλο ελαττωμάτων (fault model)
- Για να αποφασίσουμε για το fault coverage χρησιμοποιούμε προσομοίωση ελαττωμάτων (Fault simulation)

Test Generation (2)

- Ένα καλό fault model:
  - Είναι computationally efficient for simulation
  - Περιγράφει με ακρίβεια τη συμπεριφορά των defects
- Κανένα fault model δεν είναι καλό για όλα τα defects ☹

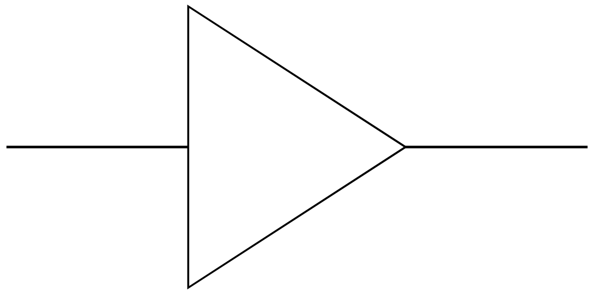
Fault Model

- Μια υπόθεση για τα σφάλματα (faults) που μπορούν να συμβούν
- Παραδείγματα:
  - Ένα καλώδιο να είναι «κολλημένο» στο 0
  - Ένα καλώδιο να καθυστερήσει να πάει από το 0 στο 1



Fault Models

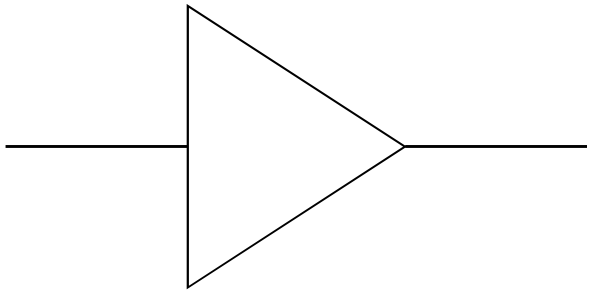
- Ένα fault model έχει συνήθως 2 είδη faults
  - *Π.χ. καλώδιο κολλημένο στο 0 ή κολλημένο στο 1*
- Ένα κύκλωμα έχει *n* fault sites
- Διακρίνουμε Single faults και Multiple Faults
- Single faults: θεωρούμε ότι το κύκλωμα έχει 1 fault
  - Το πλήθος των single faults: *2×n*



2×2 = 4 λάθη
Ποια είναι αυτά;

Fault Models

- Multiple fault model: το κύκλωμα μπορεί να έχει πολλαπλά faults (συμπεριλαμβάνονται τα single faults)
  - Number of multiple fault = 3^n-1
    - Κάθε «σημείο» μπορεί να έχει λάθος ή να είναι fault-free
    - The “-1” represents the fault-free circuit
  - Μη πρακτικό λόγω του πλήθους των faults
- Συνήθως, καλό single fault coverage δίνει καλό multiple fault coverage



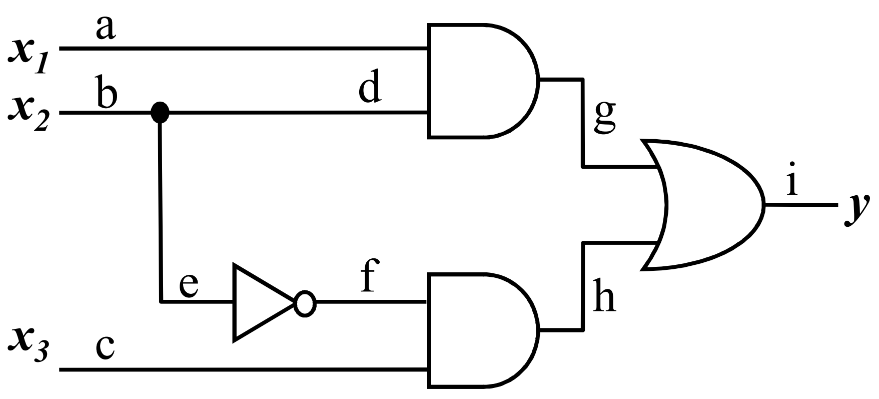
3^2 -1 = 8 λάθη, ποια είναι;

Equivalent Faults & fault collapsing

- Equivalent faults
  - Ένα ή περισσότερα single faults που έχουν ίδια συμπεριφορά για όλους τους συνδυασμούς εισόδων
  - Μόνο ένα fault από το σύνολο των equivalent faults χρειάζεται να προσομοιωθεί
- Fault collapsing
  - Removing equivalent faults
    - Except for one to be simulated
  - Μειώνει το πλήθος των faults και άρα
    - το χρόνο για fault simulation
    - το χρόνο για test pattern generation

Stuck-at Faults

- Any line can be
  - Stuck-at-0 (SA0)
  - Stuck-at-1 (SA1)
- Πώς επηρεάζει ένα fault π.χ.
  - a-SA0?
  - a-SA1?
- Πώς φτιάχνω το διπλανό πίνακα;



Truth table for fault-free behavior

$x_1x_2x_3$	000	001	010	011	100	101	110	111
y	0	1	0	0	0	1	1	1

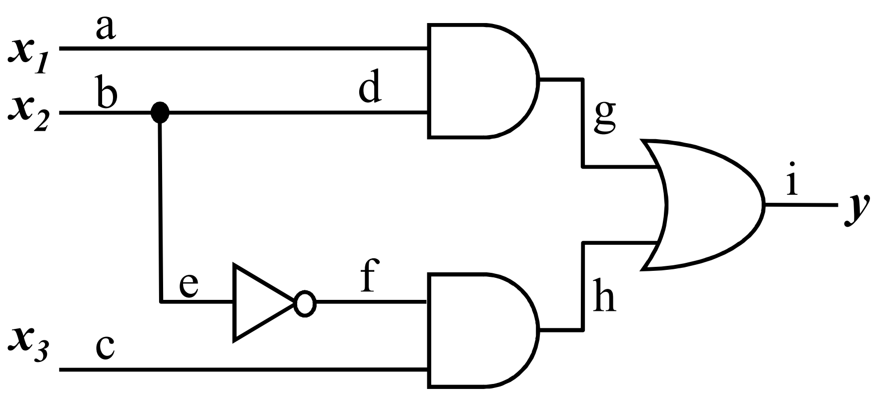
a SA0	0	1	0	0	0	1	▨ 0	▨ 0

a SA1	0	1	▨ 1	▨ 1	0	1	1	1

Stuck-at Faults

Ερωτήματα:

1. Πώς μπορώ να ανακαλύψω το a-SA0?
2. Πώς μπορώ να ανακαλύψω το a-SA1?



Truth table for fault-free behavior

$x_1x_2x_3$	000	001	010	011	100	101	110	111
y	0	1	0	0	0	1	1	1

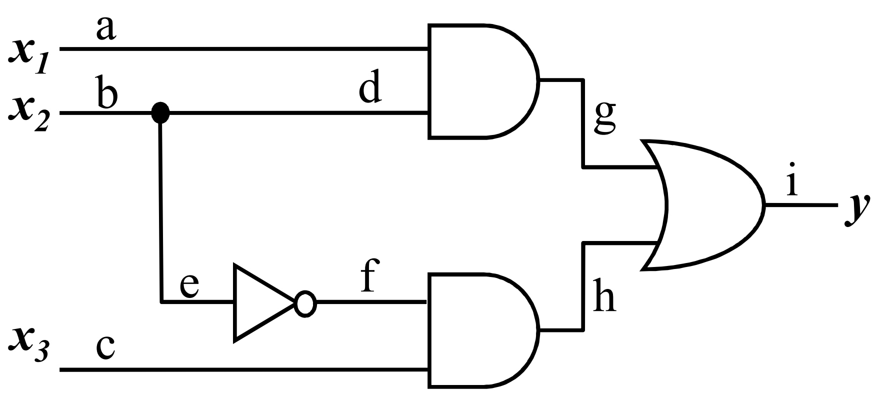
a SA0	0	1	0	0	0	1	▨ 0	▨ 0

a SA1	0	1	▨ 1	▨ 1	0	1	1	1

Stuck-at Faults

Στο παράδειγμά μας:

1. Πόσα λάθη υπάρχουν;
2. Πώς επηρεάζει το καθένα;
3. Πώς μπορώ να τα ανακαλύψω όλα;
   - # fault sites: *n*=9
   - # single faults =2×9=18



Truth table for fault-free behavior and behavior of all possible stuck-at faults

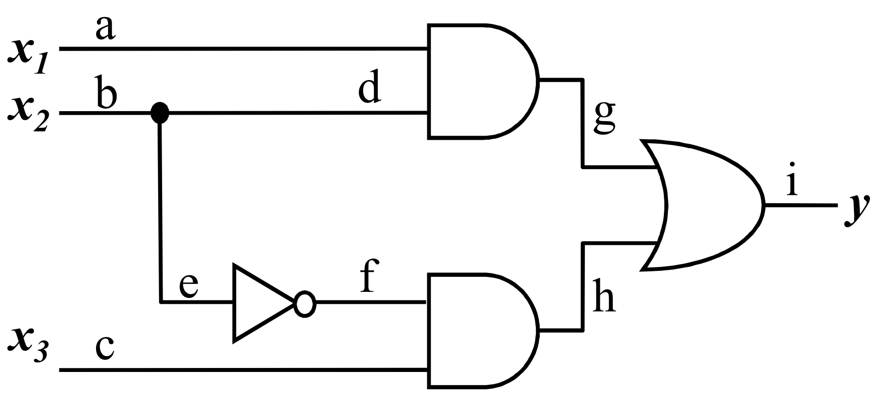
$x_1x_2x_3$	000	001	010	011	100	101	110	111
y	0	1	0	0	0	1	1	1
a SA0	0	1	0	0	0	1	0	0
a SA1	0	1	1	1	0	1	1	1
b SA0	0	1	0	1	0	1	0	1
b SA1	0	0	0	0	1	1	1	1
c SA0	0	0	0	0	0	0	1	1
c SA1	1	1	0	0	1	1	1	1
d SA0	0	1	0	0	0	1	0	0
d SA1	0	1	0	0	1	1	1	1
e SA0	0	1	0	1	0	1	1	1
e SA1	0	0	0	0	0	0	1	1
f SA0	0	0	0	0	0	0	1	1
f SA1	0	1	0	1	0	1	1	1
g SA0	0	1	0	0	0	1	0	0
g SA1	1	1	1	1	1	1	1	1
h SA0	0	0	0	0	0	0	1	1
h SA1	1	1	1	1	1	1	1	1
i SA0	0	0	0	0	0	0	0	0
i SA1	1	1	1	1	1	1	1	1

Stuck-at Faults

- Διακρίνουμε “easy” και “hard” faults
- Hard faults:
  - f-SA1, e-SA0, d-SA1
- Easy faults:
  - g-SA1, h-SA1, i-SA0, i-SA1

Truth table for fault-free behavior and behavior of all possible stuck-at faults

$x_1x_2x_3$	000	001	010	011	100	101	110	111
y	0	1	0	0	0	1	1	1
a SA0	0	1	0	0	0	1	0	0
a SA1	0	1	1	1	0	1	1	1
b SA0	0	1	0	1	0	1	0	1
b SA1	0	0	0	0	1	1	1	1
c SA0	0	0	0	0	0	0	1	1
c SA1	1	1	0	0	1	1	1	1
d SA0	0	1	0	0	0	1	0	0
d SA1	0	1	0	0	1	1	1	1
e SA0	0	1	0	1	0	1	1	1
e SA1	0	0	0	0	0	0	1	1
f SA0	0	0	0	0	0	0	1	1
f SA1	0	1	0	1	0	1	1	1
g SA0	0	1	0	0	0	1	0	0
g SA1	1	1	1	1	1	1	1	1
h SA0	0	0	0	0	0	0	1	1
h SA1	1	1	1	1	1	1	1	1
i SA0	0	0	0	0	0	0	0	0
i SA1	1	1	1	1	1	1	1	1

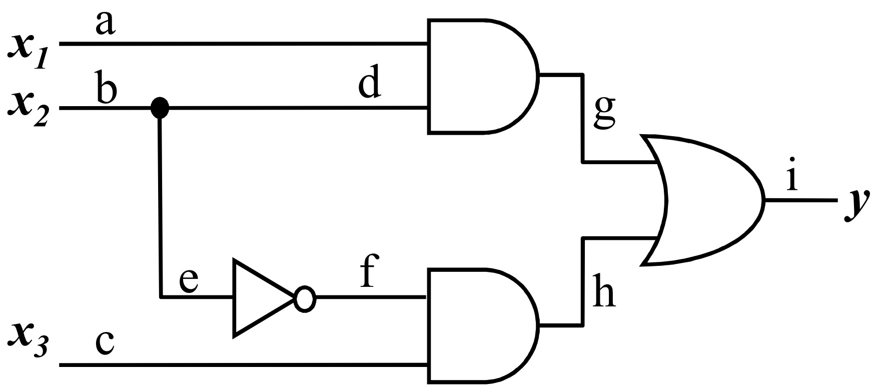


Stuck-at Faults

- Ποια vectors πρέπει να πάρω οπωσδήποτε;
  - 011 για τα f-SA1, e-SA0
  - 100 για τα d-SA1
  - Detect total of 10 faults
  - 001 and 110 detect remaining 8 faults

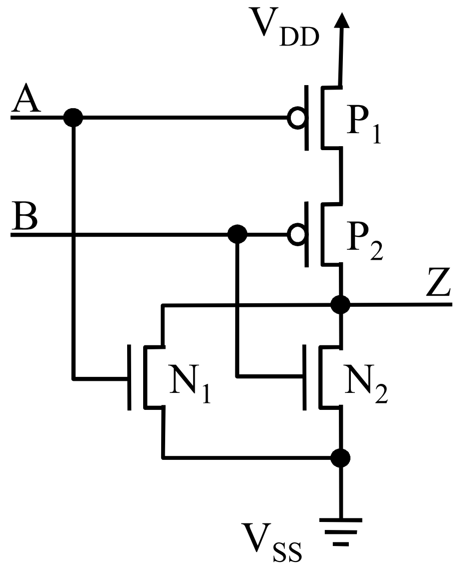
Truth table for fault-free behavior and behavior of all possible stuck-at faults

$x_1x_2x_3$	000	001	010	011	100	101	110	111
y	0	1	0	0	0	1	1	1
a SA0	0	1	0	0	0	1	0	0
a SA1	0	1	1	1	0	1	1	1
b SA0	0	1	0	1	0	1	0	1
b SA1	0	0	0	0	1	1	1	1
c SA0	0	0	0	0	0	0	1	1
c SA1	1	1	0	0	1	1	1	1
d SA0	0	1	0	0	0	1	0	0
d SA1	0	1	0	0	1	1	1	1
e SA0	0	1	0	1	0	1	1	1
e SA1	0	0	0	0	0	0	1	1
f SA0	0	0	0	0	0	0	1	1
f SA1	0	1	0	1	0	1	1	1
g SA0	0	1	0	0	0	1	0	0
g SA1	1	1	1	1	1	1	1	1
h SA0	0	0	0	0	0	0	1	1
h SA1	1	1	1	1	1	1	1	1
i SA0	0	0	0	0	0	0	0	0
i SA1	1	1	1	1	1	1	1	1



Transistor Faults

- Ένα transistor μπορεί να έχει τα εξής λάθη:
  - Stuck-short
    - Also known as stuck-on
    - Άγει συνεχώς
  - Stuck-open
    - Also known as stuck-off
    - Δεν άγει ποτέ
- Παράδειγμα: Πύλη NOR
- Reminder:
  - P:
    - 0=> conduct,
    - 1=> no conduct
  - N:
    - 0: no conduct,
    - 1: conduct



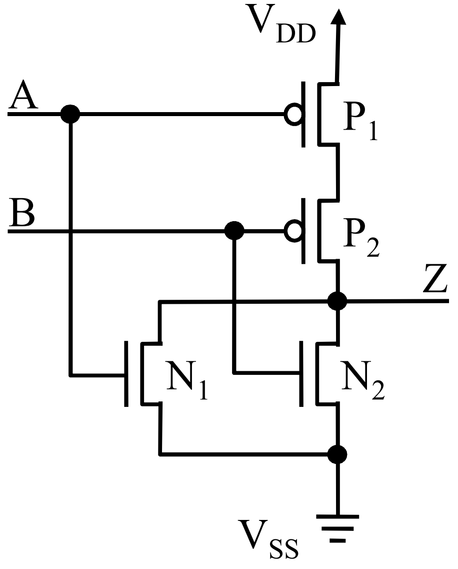
Truth table for fault-free circuit

AB	00	01	10	11
Z	1	0	0	0

Transistor Fault effects

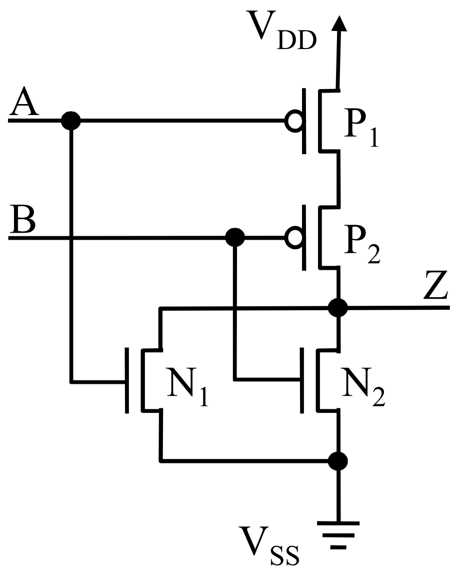
- Stuck-short faults cause conducting path from V_{DD} to V_{SS}
- Stuck-open faults cause output node to store last voltage level

2-input CMOS NOR gate



Transistor Faults

- Example circuit
  - 2-input CMOS NOR gate
  - # fault sites: *n*=4
  - # single faults =2×4=8



Truth table for fault-free circuit and N1 stuck-short transistor fault

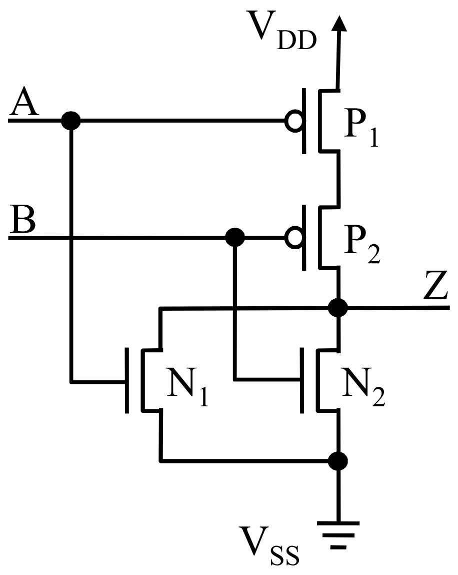
AB	00	01	10	11
Z	1	0	0	0
N_1 stuck-short	I_{DDQ}	0	0	0

Truth table for fault-free circuit and N1 stuck-open transistor fault

AB	00	01	10	11
Z	1	0	0	0
N_1 stuck-open	1	0	last Z	0

Transistor Faults

- Example circuit
  - 2-input CMOS NOR gate
  - # fault sites: *n*=4
  - # single faults =2×4=8



Truth table for fault-free circuit and all possible transistor faults

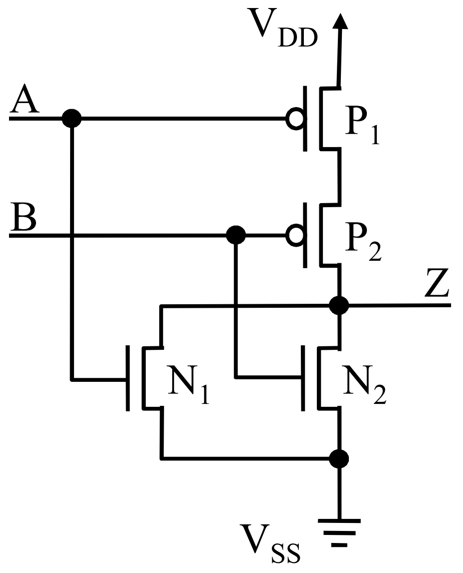
AB	00	01	10	11
Z	1	0	0	0
N_1 stuck-short	I_{DDQ}	0	0	0
N_2 stuck-short	I_{DDQ}	0	0	0
P_1 stuck-short	1	0	I_{DDQ}	0
P_2 stuck-short	1	I_{DDQ}	0	0

AB	00	01	10	11
Z	1	0	0	0
N_1 stuck-open	1	0	last Z	0
N_2 stuck-open	1	last Z	0	0
P_1 stuck-open	last Z	0	0	0
P_2 stuck-open	last Z	0	0	0

Transistor Faults

- ❑ Τι πρέπει να κάνω για να ελέγξω το fault N1 stuck-short;
- ❑ Πόσα διανύσματα (συνδυασμούς εισόδων) χρειάζομαι;
- ❑ Τι πρέπει να κάνω για να ελέγξω το fault N1 stuck-open;
- ❑ Πόσα διανύσματα (συνδυασμούς εισόδων) χρειάζομαι;

2-input CMOS NOR gate



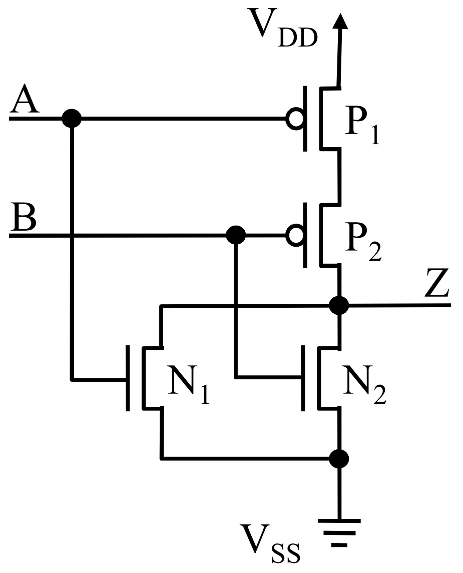
Truth table for fault-free circuit and all possible transistor faults

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

Transistor Faults

- Stuck-short faults cause conducting path from V_{DD} to V_{SS}
  - Can be detected by monitoring steady-state power supply current I_{DDQ}
- Stuck-open faults cause output node to store last voltage level
  - Requires sequence of 2 vectors for detection
    - $00 \rightarrow 10$ detects N_1 stuck-open

2-input CMOS NOR gate



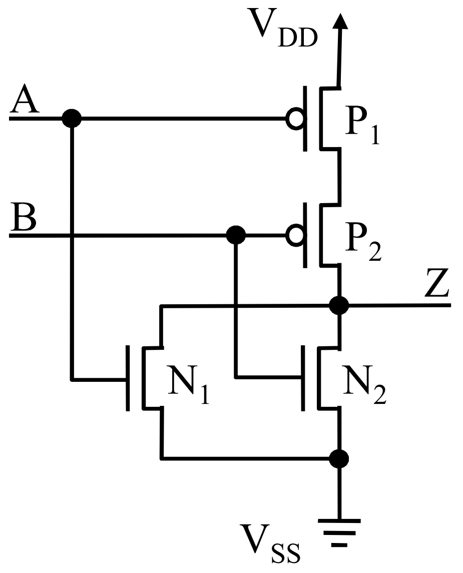
Truth table for fault-free circuit and all possible transistor faults

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

Transistor Faults

- Τι πρέπει να κάνω για να ελέγξω όλα τα stuck-short λάθη;
- Πόσα διανύσματα (συνδυασμούς εισόδων) χρειάζομαι;
- Τι πρέπει να κάνω για να ελέγξω όλα τα stuck-open λάθη;
- Πόσα διανύσματα (συνδυασμούς εισόδων) χρειάζομαι;

2-input CMOS NOR gate



Truth table for fault-free circuit and all possible transistor faults

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

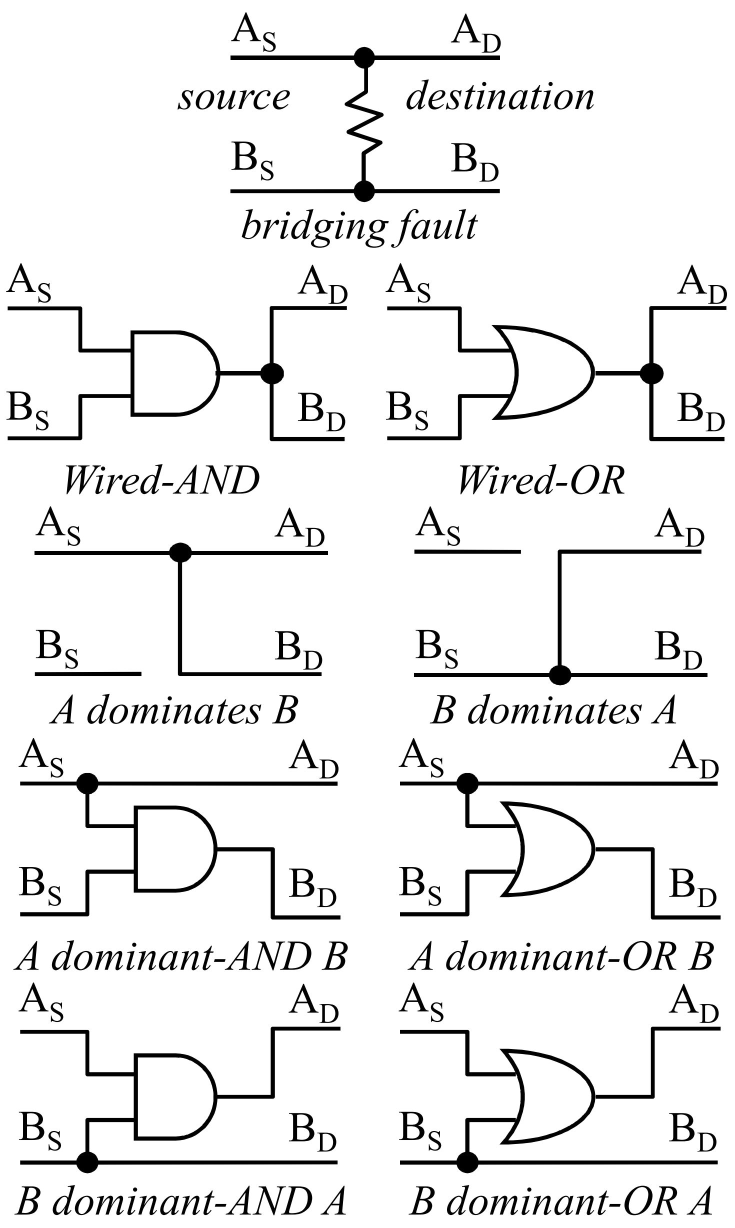
Shorts and Opens

- Wires can be
  - Open
    - Opens in wires interconnecting transistors to form gates
      - behave like transistor stuck-open faults
    - Opens in wires interconnecting gates to form circuit
      - behave like stuck-at faults
    - Opens are detected by vectors detecting transistor and stuck-at faults
  - Short to an adjacent wire
    - Also known as a bridging fault

Bridging Faults

- Three different models
  - Wired-AND/OR
  - Dominant
  - Dominant-AND/OR
- Fault-free case

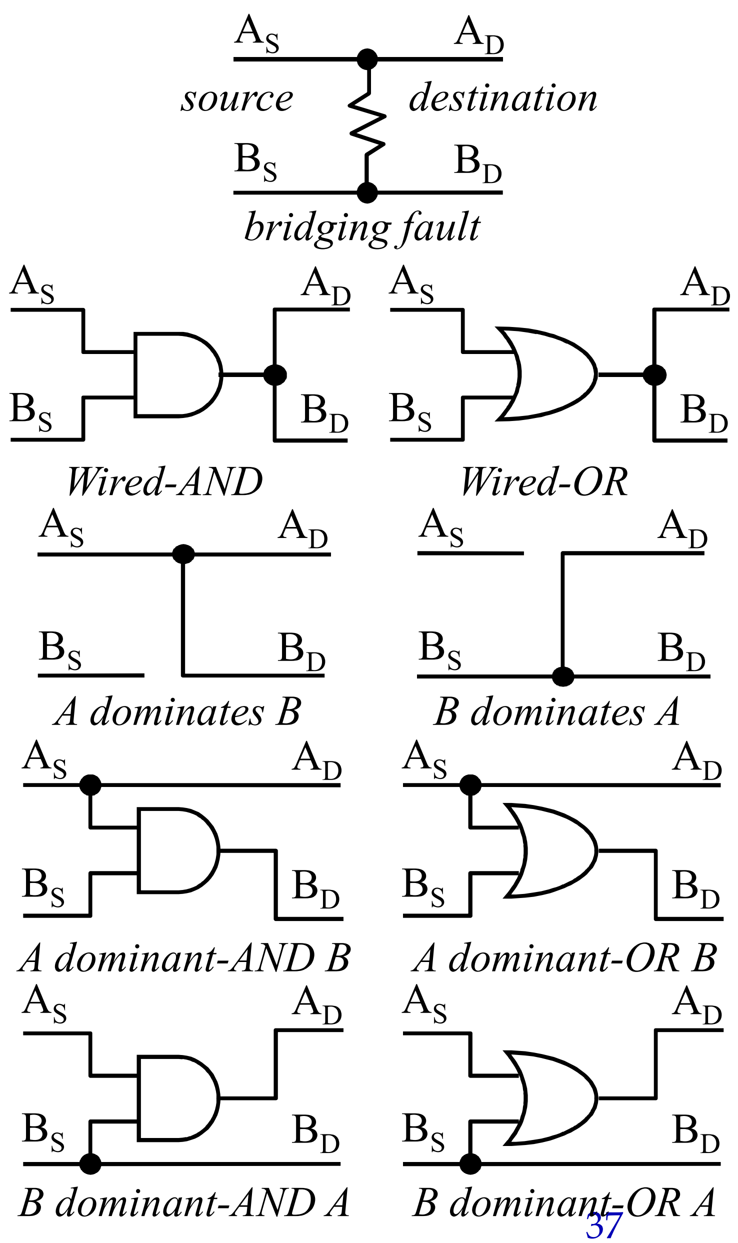
A_S B_S	0 0	0 1	1 0	1 1
A_D B_D	0 0	0 1	1 0	1 1



Bridging Faults

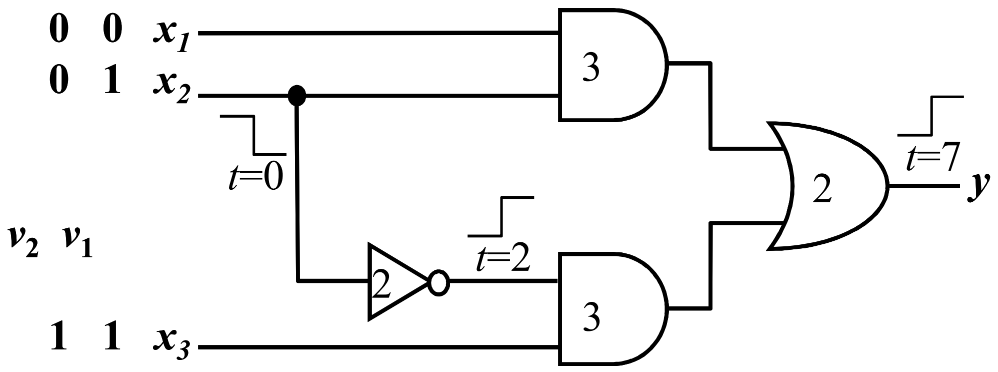
- Three different models
  - Wired-AND/OR
  - Dominant
  - Dominant-AND/OR
- Detectable by I_{DDQ} testing

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



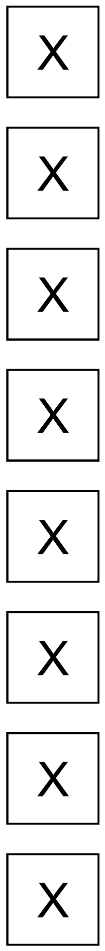
Delay Faults and Crosstalk

- Path-delay fault model considers cumulative propagation delay through CUT
  - 2 test vectors create transition along path
  - Faulty circuit has excessive delay
- Delays and glitches can be caused by crosstalk between interconnect
  - due to inductance and capacitive coupling



Faults in RAMs

- Pattern sensitive fault
  - Contents of memory cell is affected by contents of neighboring cells
- Coupling fault
  - Transition in contents of one memory cell causes change in contents of another cell
- Detected with specific memory test algorithms



Memory March Algorithms

- Consist of March elements
- Example: March C-
- March elements M0-M5

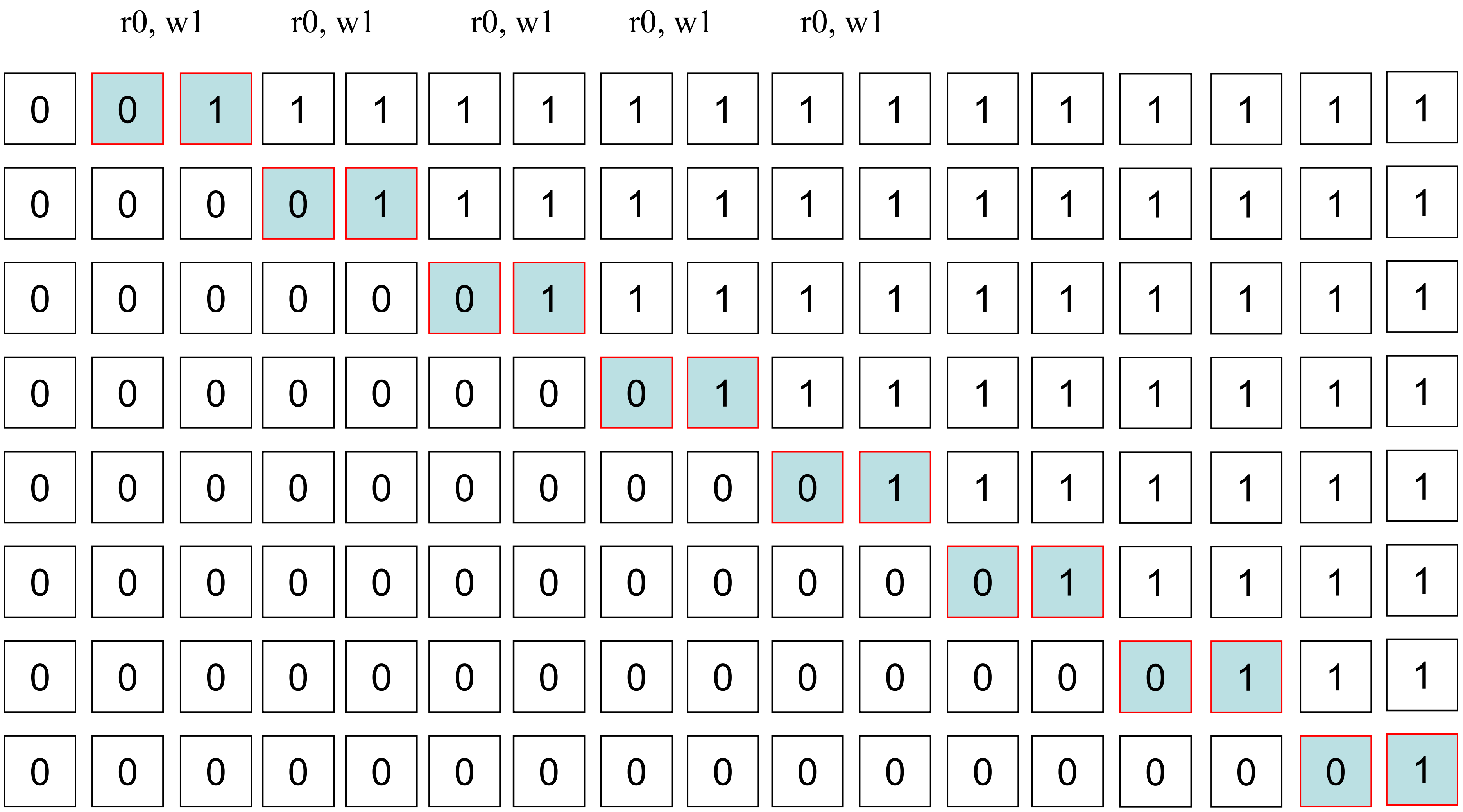
Test Algorithm March C-	March Test Sequence
M0	↕(w0);
M1	↑ (r0, w1);
M2	↑(r1, w0);
M3	↓(r0, w1);
M4	↓(r1, w0);
M5	↑(r0)

Notation:
w0 = write 0 (or all 0's)
r1 = read 1 (or all 1's)
↑= address up
↓= address down
↕ = address either way

M0. ⇔ (w0);

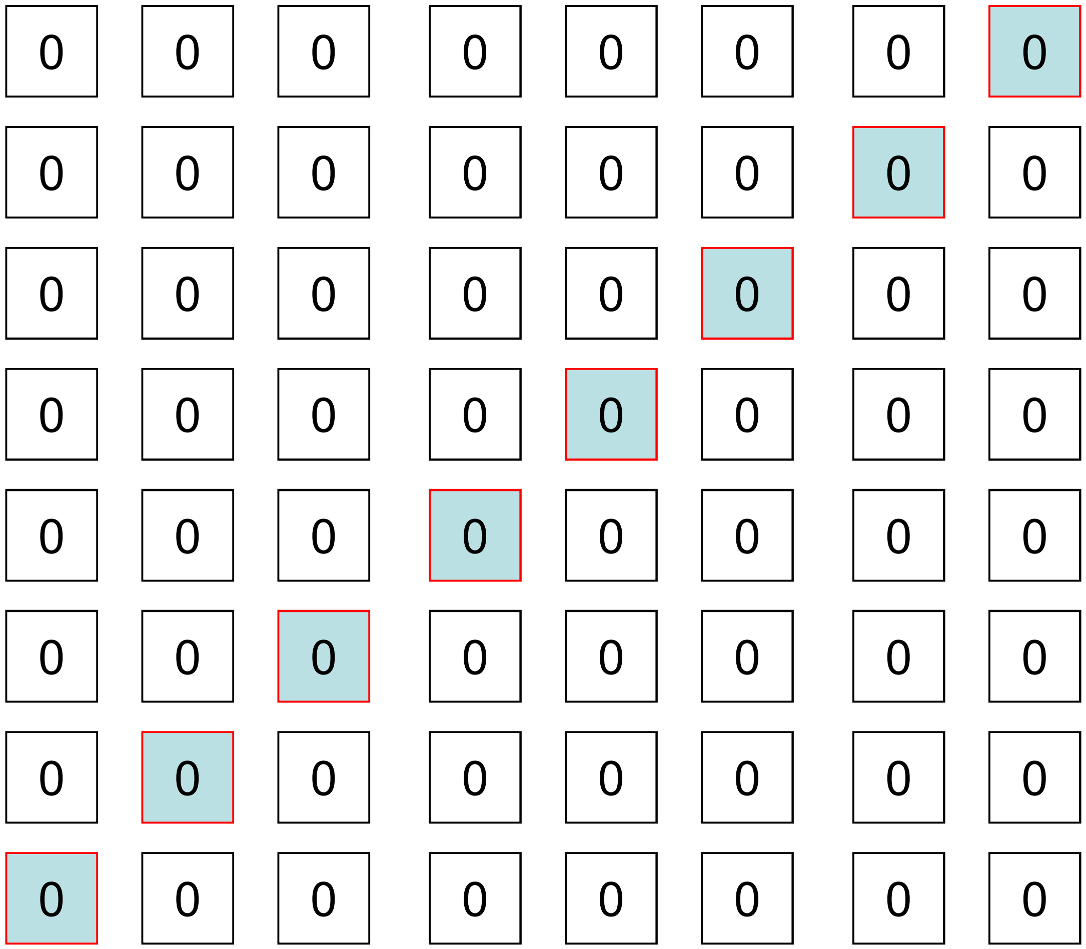
X	0	0	0	0	0	0	0	0
X	X	0	0	0	0	0	0	0
X	X	X	0	0	0	0	0	0
X	X	X	X	0	0	0	0	0
X	X	X	X	X	0	0	0	0
X	X	X	X	X	X	0	0	0
X	X	X	X	X	X	X	0	0
X	X	X	X	X	X	X	X	0

M1. ⇑ (r0, w1);



0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

M5. ⇓ (r0);



Coupling Faults

- Background Data Sequence (BDS) used for word-oriented memories

Test Algorithm	March Test Sequence
March C- w/o BDS	↕(w0); ↑ (r0, w1); ↑(r1, w0); ↓(r0, w1); ↓(r1, w0); ↑(r0)
March C- with BDS	↕(w00); ↑ (r00, w11); ↑(r11, w00); ↓(r00, w11); ↓(r11, w00); ↑(r00) ↕(w01); ↑ (r01, w10); ↑(r10, w01); ↓(r01, w10); ↓(r10, w01); ↑(r01)

Notation:
w0 = write 0 (or all 0's)
r1 = read 1 (or all 1's)
↑= address up
↓= address down
↕ = address either way

Overview of VLSI Test Technology

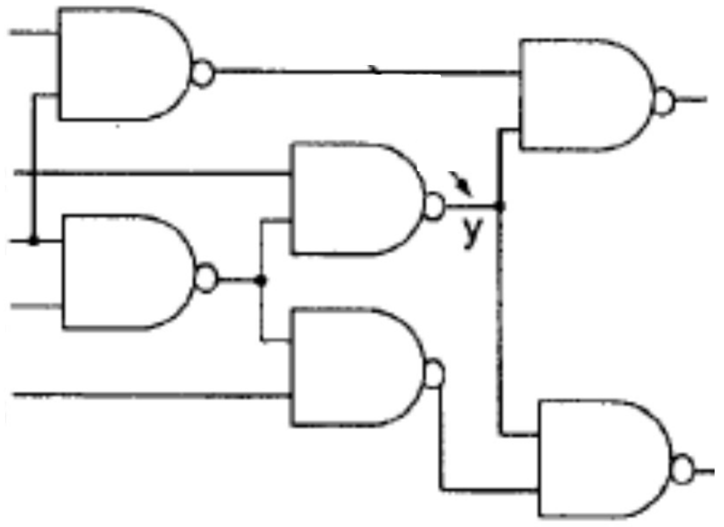
- Automatic Test Equipment (ATE) consists of:
  - Computer
    - for central control and flexible test & measurement for different products
  - Pin electronics & fixtures –
    - to apply test patterns to pins & sample responses
  - Test program –
    - controls timing of test patterns & compares response to known good responses
- Τι ανεβάζει την τιμή του tester?

Overview of VLSI Test Technology

- Automatic Test Pattern Generation (ATPG)
  - Algorithms generating sequence of test vectors for a given circuit based on specific fault models
- Fault simulation
  - Emulates fault models in CUT and applies test vectors to determine fault coverage
  - Simulation time (significant due to large number of faults to emulate)

Case study: ISCAS85 benchmarks

- c17: toy benchmark



Test set:	Responses:
0111	0110
1010	
1100	
1011	
1100	0110

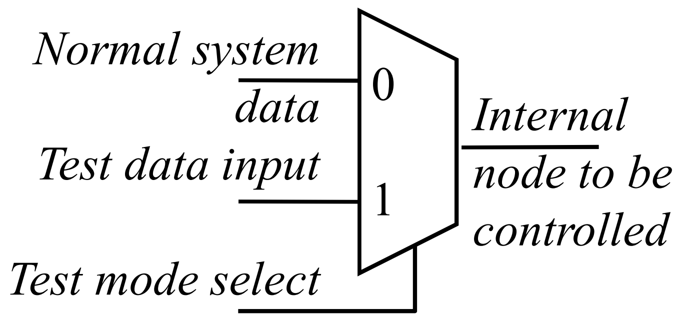
Πόσα διανύσματα χρειάζονται για να βρω όλα τα faults?

Overview of VLSI Test Technology

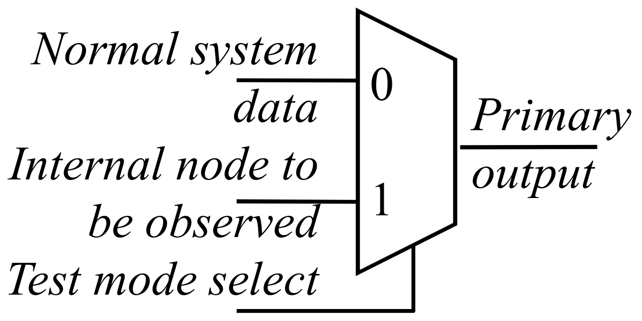
- Design for Testability (DFT)
  - Generally incorporated in design
  - Goal: improve controllability and/or observability of internal nodes of a chip or PCB
- Three basic approaches
  - Ad-hoc techniques
  - Scan design
    - Boundary Scan
  - Built-In Self-Test (BIST)

Design for Testability

- Ad-hoc DFT techniques
  - Add internal test points (usually multiplexers) for
    - Controllability
    - Observability
  - Added on a case-by-case basis
    - Primarily targets “hard to test” portions of chip



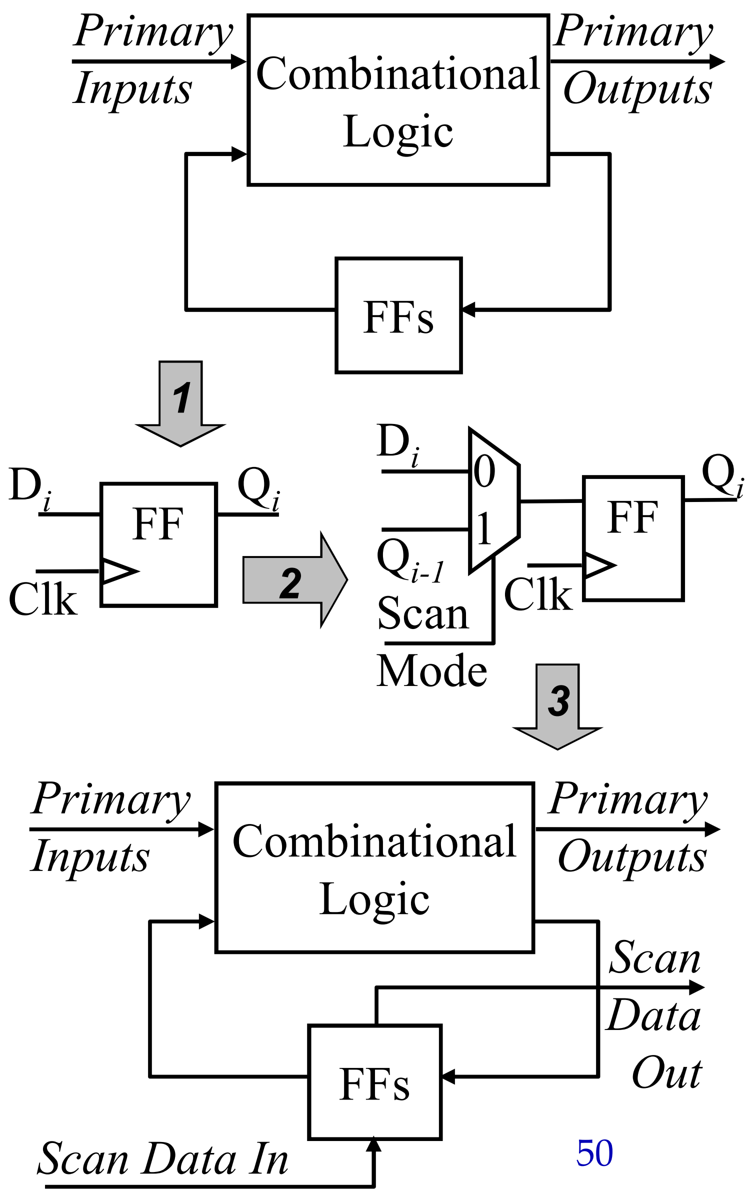
controllability test point



observability test point

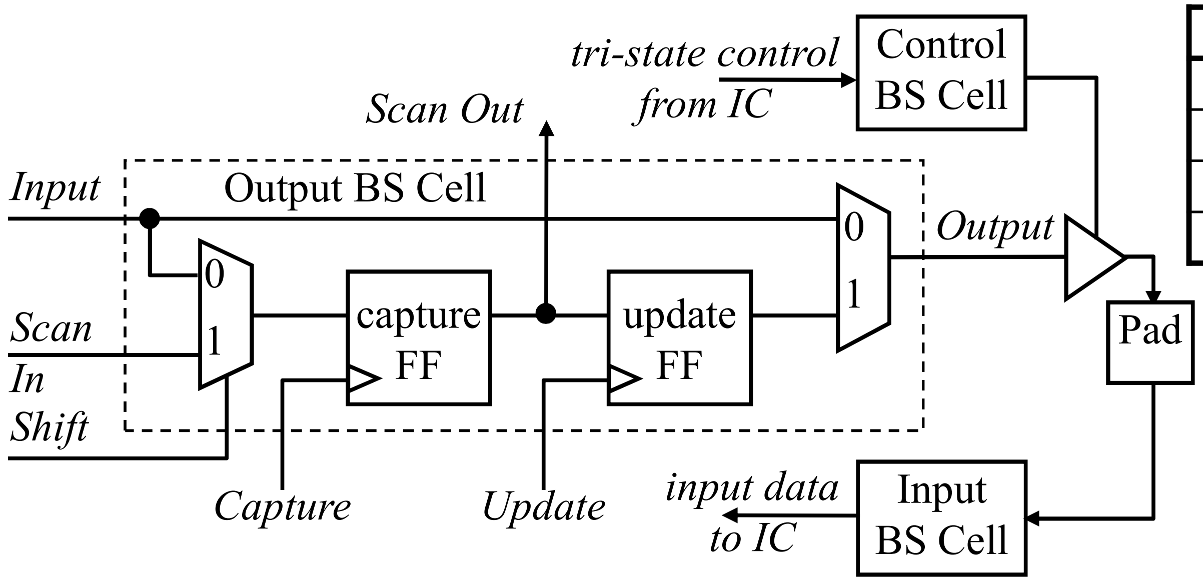
Design for Testability

- Scan design
  - Transforms flip-flops of chip into a shift register
  - Scan mode facilitates
    - Shifting in test vectors
    - Shifting out responses
- Good CAD tool support
  - Transforming flip-flops to shift register
  - ATPG



Design for Testability

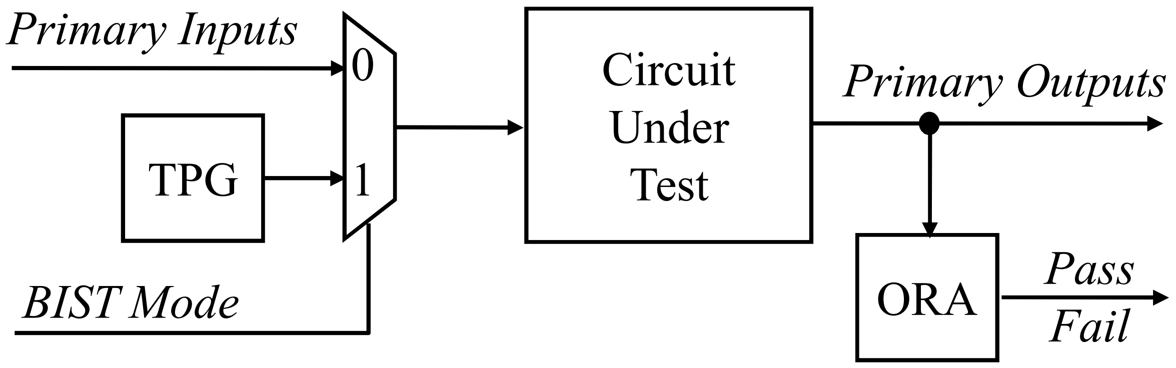
- ❑ Boundary Scan – scan design applied to I/O buffers of chip
  - ▪ Used for testing interconnect on PCB
    - – Provides access to internal DFT capabilities
  - ▪ IEEE standard 4-wire Test Access Port (TAP)



TAP pin	I/O	Function
TCK	input	Test clock
TMS	input	Test Mode Select
TDI	input	Test Data In
TDO	output	Test Data Out

Design for Testability

- Built-In Self-Test (BIST)
  - Incorporates test pattern generator (TPG) and output response analyzer (ORA) internal to design
    - Chip can test itself
  - Can be used at all levels of testing
    - Device → PCB → system → field operation



Concluding Remarks

- Many new testing challenges presented by
  - Increasing size and complexity of embedded devices
  - Decreasing feature size
- Presented introduction to embedded systems testing

Thank you!

Questions?