Testing and Testability

P.RAJESH

DEPT OF E.C.E

J.N.T.U.A C.E.A

Content of Syllabus

- ..\New Microsoft Word Document.docx

1.1 Testing Philosophy

- Example :
- The teacher sets a *domain of knowledge* for testing, called the *course syllabus.*
  - It may be the contents of a book, class notes, lectures, or some (arbitrary!) combination of all those.
  - Next, comes the testing method. The teacher asks questions and analyzes the response, perhaps by matching answers to correct ones from the book.
  - The quality of such a test system depends upon how well the test questions cover the syllabus.
- In VLSI testing also, one should know the *specification* (synonymous to the course syllabus) of the object being tested and then devise tests such that if the object produces the expected response then its conformance to the specification can be guaranteed.

Introduction to Testing

- Electronic testing also uses *fault modeling* and tests are generated for the assumed fault models.
- In testing, successful experience with a fault model gives it credibility, and eventually people expect reliability when a high percentage of the modeled faults is tested.

- **Example 1.1** Testing of students.
- In a course on xyzeeology, 70% of the students deserve to pass. We will call them “pass quality” students. Assuming that the number of students in the class is large, we will study the test process using a statistical basis. For a randomly selected student from the class, we define the following events:

PQ: student is pass quality P: student passes the test

FQ: student is fail quality F: student fails the test

Solution:

- Given Prob(PQ) = 0.7 Assuming that only pass/fail grades are awarded,

  The remaining 30% students are of “fail quality,”

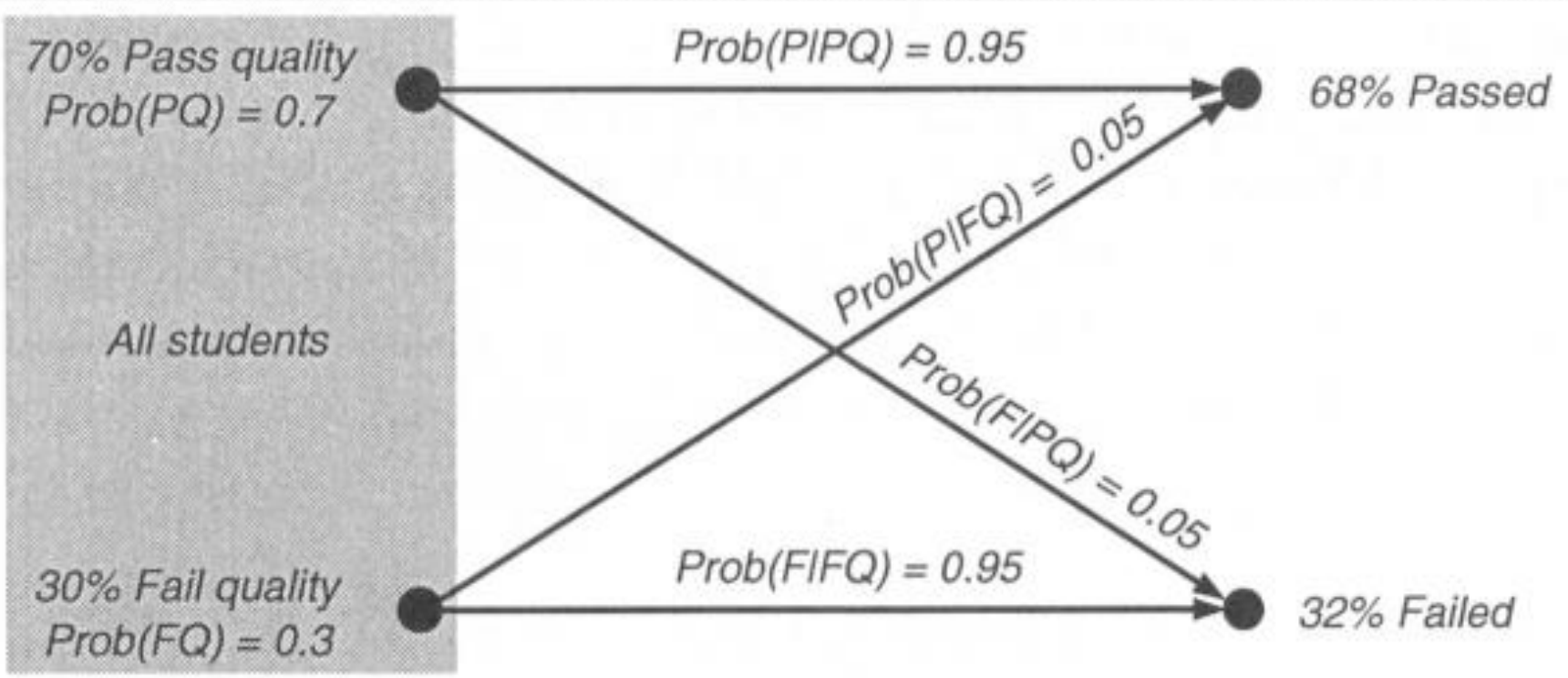
  i.e. Prob(FQ)= 0.3. As we know, it is impossible to design a perfect test.

- However, our teacher does quite well and 95% of pass quality students actually pass the test.
- This is represented by conditional probabilities, and Prob(P/PQ) = 0.95 and

  Prob(F/PQ)=0.05

  Similarly, the test correctly fails 95% of the fail quality students.

- A pass/fail test.



The total probability of passing is,

$$Prob(P) = Prob(P|PQ) \times Prob(PQ) + Prob(P|FQ) \times Prob(FQ) \quad (1.1)$$

$$= 0.95 \times 0.70 + 0.05 \times 0.30 = 0.68$$

Similarly Probability of failing is found to be

$$P(F) = 0.32$$

If original group had 70% Pass quality students, test has passed only 68%

Let Examine conditional Probability Prob (F|P) of student belonging to the fail quality subgroup, given that he (or) she passed.

the Joint Probability of Events FQ and P is

$$Prob(FQ, P) = Prob(FQ/P)\, Prob(P) = Prob\left(\frac{P}{FQ}\right) Prob(FQ) \rightarrow (b)$$

$$\therefore Prob\left(\frac{FQ}{P}\right) = \frac{Prob\left(\frac{P}{FQ}\right) Prob\,(FQ)}{Prob(P)} \rightarrow (c)$$

Teachers Risk

$$Prob\left(\frac{FQ}{P}\right) = \frac{0.05 \times 0.30}{0.68} = 0.022$$

i.e 2.2% Pass students are of Jail quality.

where Eq (3) is known as Baye's rule.

Gen Teacher Can reduce this risk by making test more difficult decreasing $P\left(\frac{P}{FQ}\right)$ close to 0.

w.r.t Students risk

Applying the Bayes rule we obtain

$$Prob\left(\frac{PQ}{F}\right) = \frac{Prob\left(\frac{F}{PQ}\right) Prob(PQ)}{Prob(F)} = \frac{0.05 \times 0.7}{0.32} = 0.11$$

→ (d)

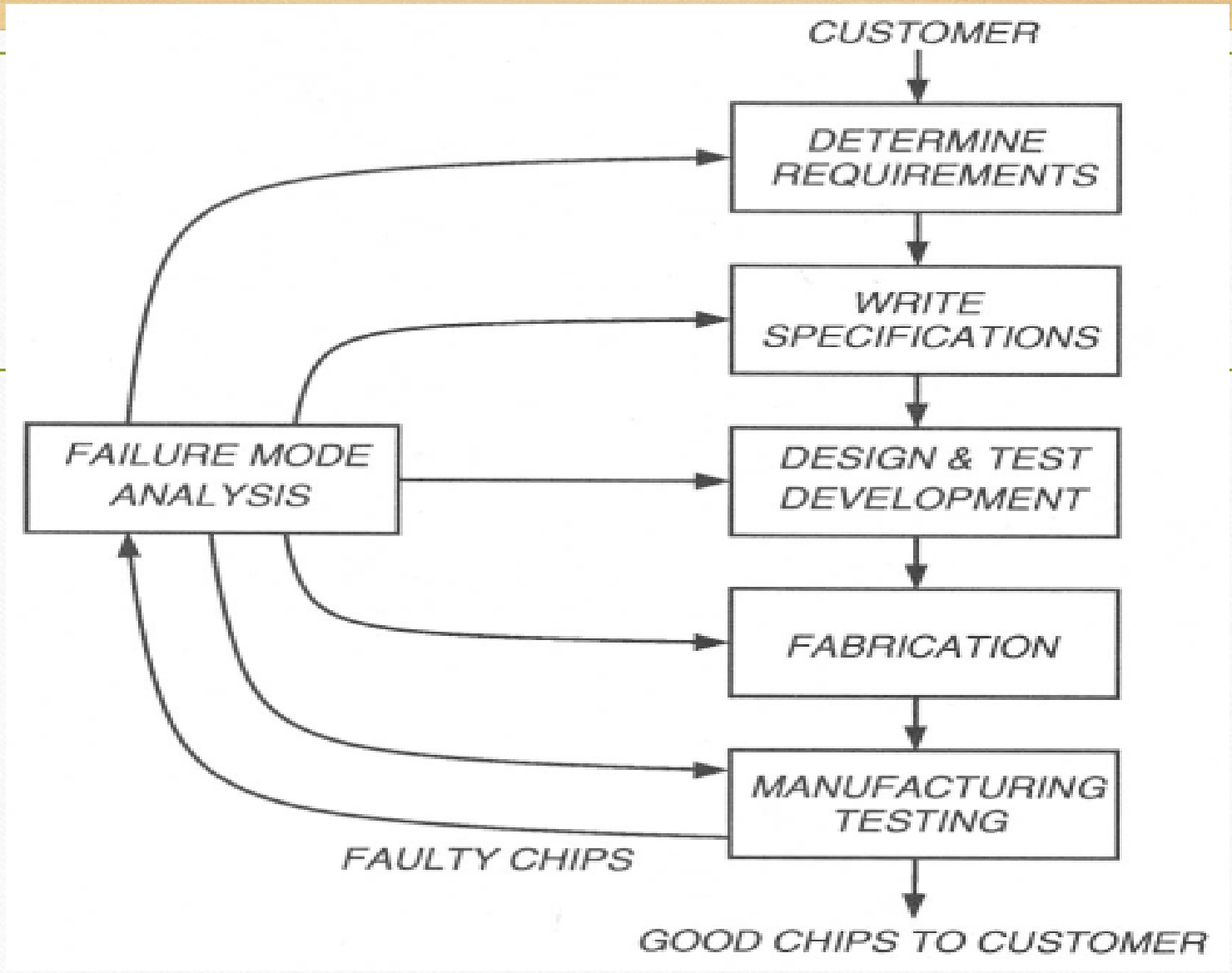
∴ 11% of failed students should have passed.

Role of Testing

- If you design a product, fabricate and test it, and it fails the test, then there must be a cause for the failure.
- Either (1) the test was wrong, or
  (2) the fabrication process was faulty, or
  (3) the design was incorrect, or
  (4) the specification had a problem.

Anything can go wrong.

The role of *testing* is to detect whether something went wrong and the role of *diagnosis* is to determine exactly what went wrong, and where the process needs to be altered.



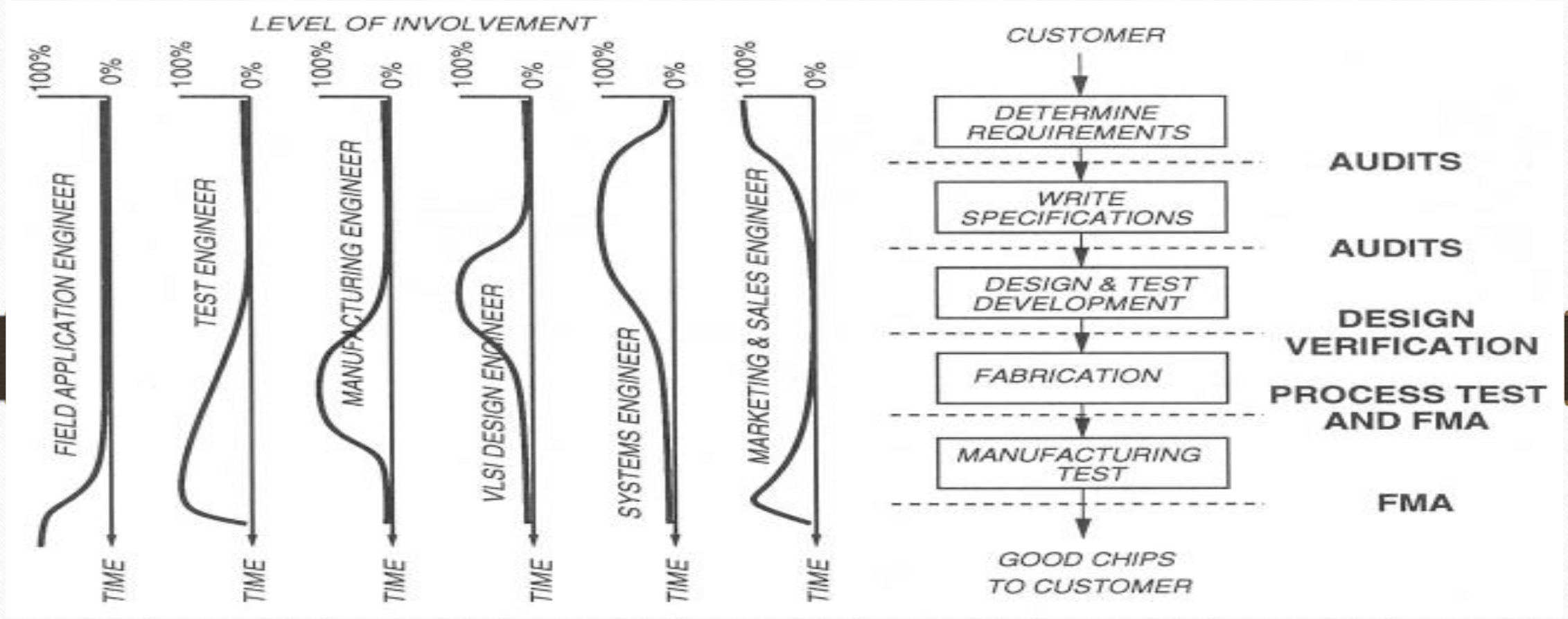
VLSI realization process

Digital and Analog VLSI Testing

- The objective of *design* is to produce data necessary for the next steps of fabrication and testing.
- Design has several stages.
- The first, known as ***architectural design***, produces a system-level structure of realizable blocks to implement the functional specification.
- The second, called ***logic design***, further decomposes blocks into logic gates.
- Finally, the gates are implemented as physical devices (e.g., transistors) and a chip layout is produced during ***physical design***.

- The physical layout is converted into photo masks that are directly used in the fabrication of silicon VLSI chips.
- Fabrication consists of processing silicon wafers through a series of steps involving photoresist, exposure through masks, etching, ion implantation, etc.
- Impurities and defects in materials, equipment malfunctions, and human errors are some causes of defects. The likelihood and consequences of defects are the main reasons for testing!
- Another very important function of testing is the process diagnosis. We must find what went wrong with each faulty chip, be it in fabrication, in design, or in testing. Or, we may have started with unrealizable specifications.
- The faulty chip analysis is called ***failure mode analysis*** (FMA.) FMA uses many different test types, including examination through optical and electron microscopes, to determine the failure cause and fix the process.

- Requirements and specifications are *audited*, design and tests are *verified*, and fabricated parts are *tested*.
- Each testing level performs two functions, and involves different technical personnel.
  - The first function ascertains that the work still conforms to the objectives of previous levels and meets customer requirements.
  - The second ascertains that things have been done according to the capabilities of the later process levels.
- For example, verification of design and test procedures should ensure that the design meets all functional and other specifications, and that it is also manufacturable, testable, and repairable.



A realistic VLSI realization process

- Figure is typically an ***application specific integrated circuit*** (ASIC), it applies to many other electronic devices as well.
- The process begins with a dialogue between the customer and the marketing engineer.
- As specifications are prepared, some involvement of those responsible for later activities (design, manufacture, and test) is advisable to ensure realizable specification.
- The systems engineer then begins by constructing an architectural block diagram.
  - The architecture is verified by high-level simulation and each block is synthesized at the logic-level.
  - The logic circuit is simulated for the same stimuli (often produced by testbenches) as used for the high-level simulation.
  - A *testbench* is hardware description language (HDL) code that, when executed, produces stimuli for the designed circuit Vectors generated by testbenches are compacted or augmented and run through a fault simulator to satisfy some specified fault coverage requirement.

VLSI Technology Trends Affecting Testing

- The complexity of VLSI technology has reached the point where we are trying to put 100 million transistors on a single chip, and we are trying to increase the on-chip clock frequency to 1 *GHz*.

Table 1.1: VLSI chips – present and future [297, 584].

Year	1997–2001	2003–2006	2009–2012
Feature size, μm	0.25–0.15	0.13–0.10	0.07–0.05
Millions of transistors/cm^2	4–10	18–39	84–180
Number of wiring layers	6–7	7–8	8–9
Die size, mm^2	50–385	60–520	70–750
Pin count	100–900	160–1475	260-2690
Clock rate, *MHz*	200–730	530–1100	840–1830
Voltage, *V*	1.2–2.5	0.9–1.5	0.5–0.9
Power, *W*	1.2–61	2–96	2.8–109

Rising Chip Clock Rates.

- Microprocessors represent the leading edge in the VLSI technology
- The exponentially rising clock rate indicates several changes in testing over the next 10 years:
- 1. **At-Speed Testing.** It has been established that *stuck-fault* tests are more effective when applied at the circuit's rated clock speed, rather than at a lower speed.
- *Stuck-fault* testing covers all (or most) circuit signals assuming that a faulty signal may be permanently stuck-at logic 0 or 1.
- For a reliable high-speed test, the ***automatic test equipment*** (ATE) must operate as fast as, or faster than, the ***circuit-under-test*** (CUT.)

- 2. **ATE *automatic test equipment* Cost.**
- At the time of writing, a state of the art ATE can apply vectors at a clock rate of 1 GHz
- As the development of faster ATE continues, other test methods are also emerging.
- An embedded-ATE method, in which ATE functions such as high-speed vector generation and response analysis are added to the chip hardware, have been proposed.
- In another method, controllable delays are inserted in the chip hardware such that the critical path delay can be measured by a slow-speed tester.

Automatic Test Equipment

- The automatic test equipment is an instrument used to apply test patterns to a *device-under-test* (DUT), analyze the responses from the DUT, and mark the DUT as good or bad.
- The DUT is also sometimes called the *circuit-under-test* (CUT.)
- The ATE is controlled by a central UNIX work station or PC, and one or more additional CPUs is often built into it to provide housekeeping and data reduction capability.
- The tester has one or more *test heads*, which contain buffering electronics local to the DUT, but one mainframe with common instrumentation, power supplies, etc.
- The ATE is connected to external equipment that mechanically handles the wafers or IC packages being tested.

- 3. **EMI** *electromagnetic interference :* A chip operating in the GHz frequency range must be tested for *electromagnetic interference* (EMI.)
- This is a problem because inductance in the wiring becomes active at these higher frequencies, whereas it could be ignored at lower frequencies.

  The inherent difficulties are:
- (1) Ringing in signal transitions along the wiring, because signal transitions are reflected from the ends of a bus and bounce back to the source, where they are reflected again
- (2) Interference with signal propagation through the wiring caused by the *dielectric permeability* and the *dielectric permittivity* of the chip package; and
- (3) Delay testing of paths requires propagation of sharp signal transitions, resulting in high-frequency currents through interconnects, causing radiation coupling.
- *Delay testing* is necessary, because many factors may delay a signal propagating along a path. We must also test the chip interconnect carefully for radiation noise induced errors.

- **Increasing Transistor Density.** Transistor feature sizes on a VLSI chip reduce roughly by 10.5% per year, resulting in a transistor density increase of roughly 22.1% every year.
- An almost equal amount of increase is provided by wafer and chip size increases and circuit design and process innovations

1. ***Test complexity***. Testing difficulty increases as the transistor density increases.

This occurs because the internal chip modules (particularly embedded memories) become increasingly difficult to access. Also, test patterns for subassemblies on the chip interfere with each other, due to the need to observe subassembly.

A through sub-assembly

B while stimulating

both sub-assemblies *A* and *B* from circuit inputs.

2. Feature scaling and power dissipation. The *power density* (power dissipation per unit area) of a CMOS chip is given by

$$Power\ density = C \times V_{DD}^2 \times f \qquad (1.6)$$

where *C* is the combined node capacitance per unit area that is switched per clock cycle,
VDD is the supply voltage, and *f* is the clock frequency.
In general, *C* is proportional to the number of transistors per unit area and the average switching probability of signals.

3. Current testing:
This method is called I_{DDQ} *testing* While switching, CMOS circuits exhibit an elevated current in the digital logic, which dies out quickly to a small quiescent current (I_{DDQ}) after the gate output settles to a steady state.
Faults, such as transistors stuck-on, shorted wires, shorts from transistor gates to drains, etc., elevate the quiescent current. testing marks the chip as faulty if the measured quiescent I_{DDQ} current through ground busses of the chip exceeds a prespecified threshold.

Integration of Analog and Digital Devices onto One Chip.

Eliminating chip-to-chip delay between an A/D converter and the *digital signal processor* (DSP) that processes the digitized data. When data goes between two chips, the driving chip inserts a delay to amplify and buffer the output signals, and the receiving chip inserts a delay to condition the signals and propagate them through a “lightening arrester” to eliminate voltage surges coming from people handling the chip.

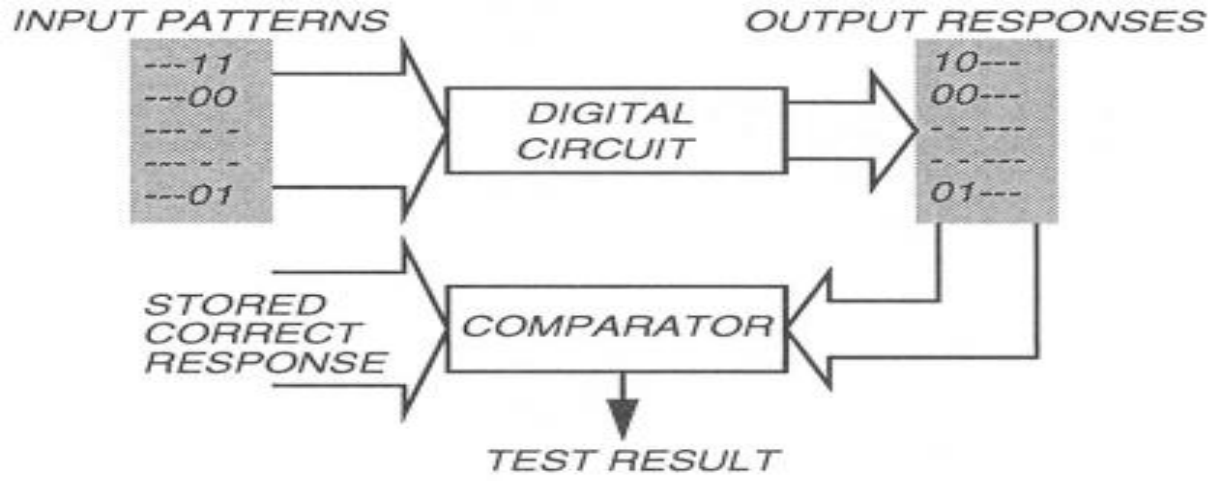
Integration onto one chip eliminates a significant delay, but brings new issues of testing mixed-signal circuits on one chip.

VLSI TESTING PROCESS AND TEST EQUIPMENT

How to Test Chips?

Binary patterns (or *test vectors*) are applied to the inputs of the circuit.
The response of the circuit is compared with the expected response. The circuit is considered good if the responses match. Obviously, the quality of the tested circuit will depend upon the thoroughness of the test vectors.

Principle of testing.



Importance of Testing

- *Rule of Ten*: cost to detect faulty IC increases by an order of magnitude as we move from:
  - device → PCB → system → field operation
    - Testing performed at all of these levels
- Testing also used during
  - Manufacturing to improve yield
    - Failure mode analysis (FMA)
  - Field operation to ensure fault-free system operation
    - Initiate repairs when faults are detected

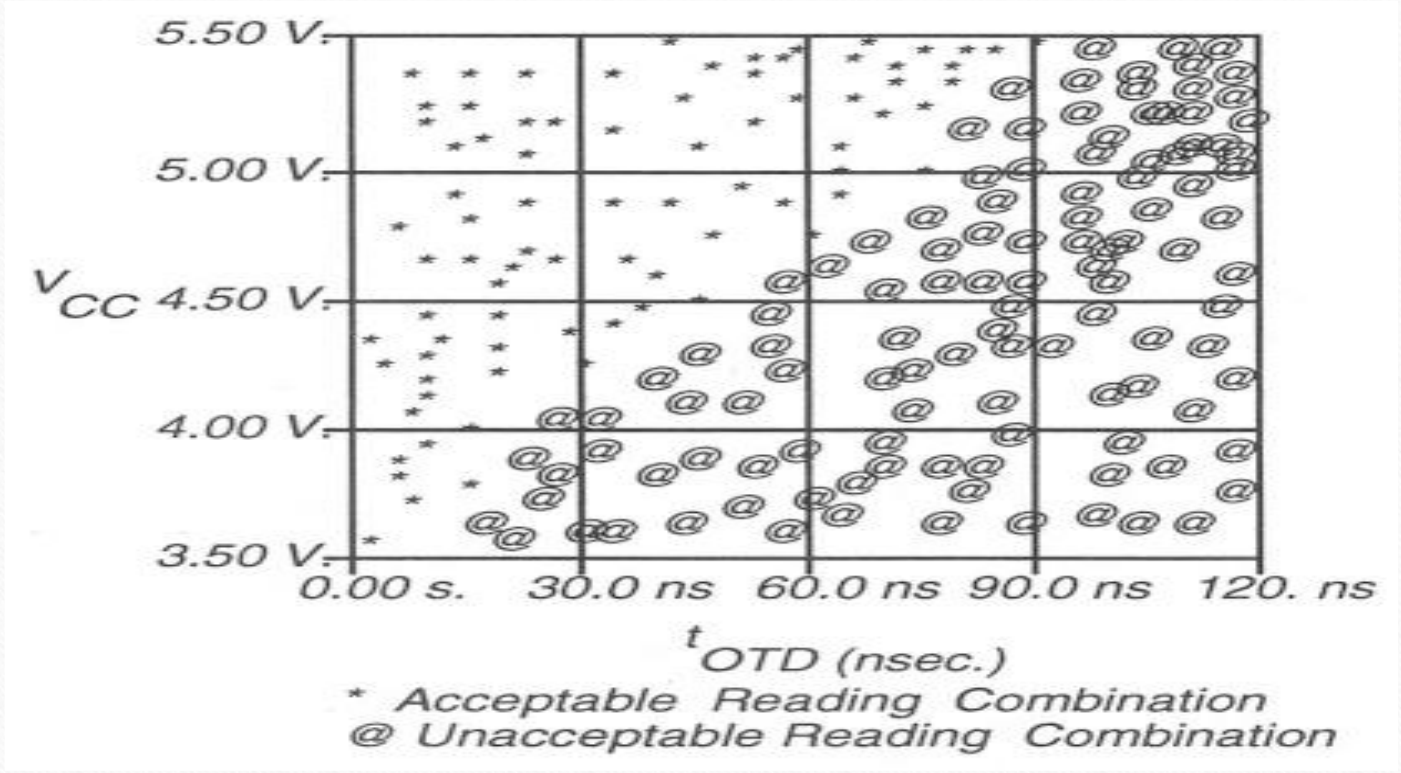
VLSI devices are tested by *automatic test equipment* (ATE) that performs a variety of tests. Modern ATE is a powerful computer operating under the control of a *test program* written in a high level language.

Types of Testing

VLSI testing can be classified into four types depending upon the specific purpose it accomplishes

Characterization: Also known as *design debug* or *verification testing*,
This form of testing is performed on a new design before it is sent to production.
The purpose is to verify that the design is correct and the device will meet all specifications.
Functional tests are run and comprehensive AC and DC measurements are made.
Probing of internal nodes of the chip, commonly not done in production testing, may also be required during characterization.
Use of specialized tools such as *scanning electron microscopes* (SEM) and electron beam testers, and techniques such as *artificial intelligence* (AI) and *expert systems*, can be effective.
A *characterization test* determines the exact limits of device operating values. We generally test for the worst case, because it is easier to evaluate than average cases and devices passing this test will work for any other conditions.

We do this by selecting a test that results in a chip pass/fail decision.
This essentially means repetitively applying functional tests and measuring various DC or AC parameters, as we vary different variables such as (the supply voltage.)
This data is plotted as a *Shmoo plot.*



Shmoo plot.

Production

- Every fabricated chip is subjected to production tests, which are less comprehensive than characterization tests yet they must enforce the quality requirements by determining whether the device meets specifications.
- The vectors may not cover all possible functions and data patterns but must have a high coverage of modeled faults. The main driver is cost, since every device must be tested.
- Test time (and therefore cost) must be absolutely minimized. Fault diagnosis is not attempted and only a *go/no-go* decision is made.
- Production tests are typically short but verify all relevant specifications of the device. It is an outgoing inspection test of each device, and is not repetitive.
- We test whether some *device-under-test* (DUT) parameters are consistent with the device specifications under normal operating conditions.
- We test either at the speed required by the application of the device or at the speed guaranteed by the supplier.

Burn-in

All devices that pass production tests are not identical. When put to actual use, some will fail very quickly while others will function for a long time. *Burn-in* ensures reliability of tested devices by testing, either continuously or periodically, over a long period of time, and by causing the bad devices to actually fail.

Incoming Inspection

- System manufacturers perform incoming inspection on the purchased devices before integrating them into the system.
- the incoming inspection may be done for a random sample with the sample size depending on the device quality and the system requirement.
- The most important purpose of this testing is to avoid placing a defective device in a system assembly where the cost of diagnosis may far exceed the cost of incoming inspection.

Types of Tests:

Although some testing is done during device fabrication to assess the integrity of the process itself, most device testing is performed after the wafers have been fabricated.

- The first test, known as ***wafer sort* or *probe***, differentiates potentially good devices from defective ones.
- Specially designed tests are applied to certain test sites containing specific test patterns. These are designed to characterize the processing technology through measurement of parameters such as gate threshold, polysilicon field threshold, bypass, metal field threshold, poly and metal sheet resistances, contact resistance, etc.

In general, each chip is subjected to two types of tests:

(1) Parametric Tests.

- DC parametric tests include shorts test, opens test, maximum current test, leakage test, output drive current test, and threshold levels test.
- AC parametric tests include propagation delay test, setup and hold test, functional speed test, access time test, refresh and pause time test, and rise and fall time test.
- These tests are usually technology-dependent. CMOS voltage output measurements are done with no load while TTL devices require current load.

(2) ***Functional Tests.***

- These consist of the input vectors and the corresponding responses. They check for proper operation of a verified design by testing the internal chip nodes.
- Functional tests cover a very high percentage of modeled (e.g., stuck type) faults in logic circuits and their generation.
- Often, functional vectors are understood as verification vectors, which are used to verify whether the hardware actually matches its specification.

Functional tests may be applied at an elevated temperature to guarantee specifications:

- For example, testing may be done at 85° C to guarantee 70° operation. This is called *guard banding*.

- Another application is in ***speed binning*** to grade the chips according to performance.

This may be done by applying the tests at several voltages and at varying timing conditions (e.g., clock frequency.)

Memory tests are *functional*:

- No stuck-type fault coverage is evaluated for these tests, these are designed to check functional attributes such as address uniqueness, address decoder speed, cell coupling, column and row coupling, data sensitivity, write recovery, and refresh.
- Elaborate testing may require long vector sequences.
- To increase throughput, memory testers sometimes have parallel testing capability.
- Specialized testers may even be able to repair redundant cells used in large memories (256 M bits) to enhance yield.

Circuit boards consist of previously-tested components.

- A primary objective of board testing is to check the printed wiring and the contacts between wires and components. It is possible to perform a bare-board testing of interconnections before the components are inserted.
- After the components (such as chips) are in place, *in-circuit testing* (through a *bed-of-nails* fixture) is often used to verify the performance of individual components, although the bed-of-nails fixture is becoming obsolete.
- Finally, functional testing determines whether or not individual components, possibly designed with different technologies, function as a system and produce the expected response.

The ATE employed for boards is different from that used for chips.

Test Specifications and Test Plan:

- Functional Characteristics – Algorithms to be implemented, I/O signal characteristics (timing waveforms, signal levels, etc.), data and control signal behavior, clock rate.
- Type of Device – Logic, microprocessor, memory, analog, etc.
- Physical Characteristics – Package, pin assignments, etc.
- Technology – CMOS (or gate array), custom, standard cell, etc.
- Environmental Characteristics – Operating temperature range, supply voltage, humidity, etc.
- Reliability – Acceptance quality level (defective parts per million), failure rate per 1,000 hours, noise characteristics, etc.

Test Plan

- In the test plan the type of test equipment and the type of tests are specified.
- Selection of a tester depends on parameters as throughput, clock rate, timing accuracy, test sequence length, tester availability, and cost.
- The types of test may include parametric, functional, burn-in, margin, speed sorting, etc.
- The fault coverage requirement should also be specified.

Testers:

The basic purpose of a tester is to drive the inputs and to monitor the outputs of a device-under-test.

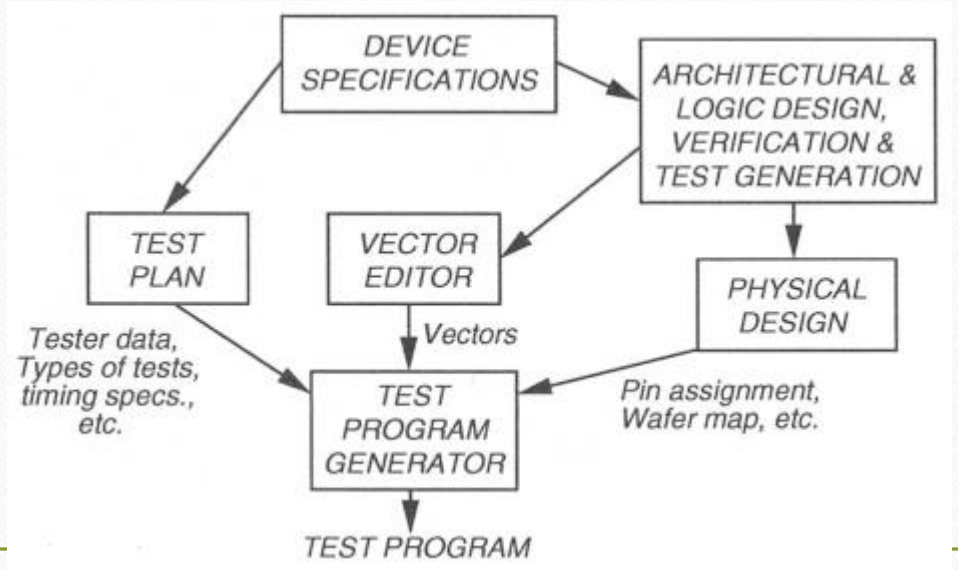
- Testers are popularly known as ATE (*automatic test equipment.*)
- Fast-changing VLSI technology has driven the development of modern ATE.
- Selection of ATE for a VLSI device must consider the specifications of the device.
- Major factors are speed (clock rate of the device), timing (strobe) accuracy, number of input/output pins, etc.
- Other considerations in selecting a tester are cost, reliability, serviceability, ease of programming, etc

Test Programming:

Once the device-under-test has been mounted in the tester, three things are needed to conduct the test.

- These are the *test program*, the digital *test vectors*, and the analog test waveforms.
- Until recently, test programs were written manually.
- However, the use of CAD tools is now becoming widespread in this area.

(1) Tester specification and the information on the types of tests is obtained from the test plan.
(2) Physical data on the device (pin locations, wafer map, etc.) are obtained from the layout.
(3) Timing information on signals and test vectors (inputs and expected responses) are obtained from simulators.



Test program generation:

The test program contains the sequence of instructions that a tester would follow to conduct testing.

For example:

- A simple sequence of events will be apply power, clocks and vectors to input pins, strobe output pins, and compare output signals with stored expected response.
- Modern testers provide a choice of input signal waveforms, mask output signals, sense high impedance state, and have a variety of sophisticated capabilities.
- Since testers differ in their capabilities and programming languages, TPGs commonly generate a *tester-independent* program which can be customized to any specific ATE.
- Also, since software simulators used in design verification differ from the ATE in their handling of signal format (logic values, timing, etc.), *vector editors* are useful tools during test programming.

Test Data Analysis:

The test data obtained from the ATE serves three purposes.

- First, it helps to accept or reject the device-under-test.
- Second, it provides useful information about the fabrication process.
- And third, it provides information about design weaknesses.

Analysis of test data provides information on device quality. Due to the random variations in the fabrication process speed characteristics of devices vary. Test data analysis allows sorting of chips for higher than the nominal performance.

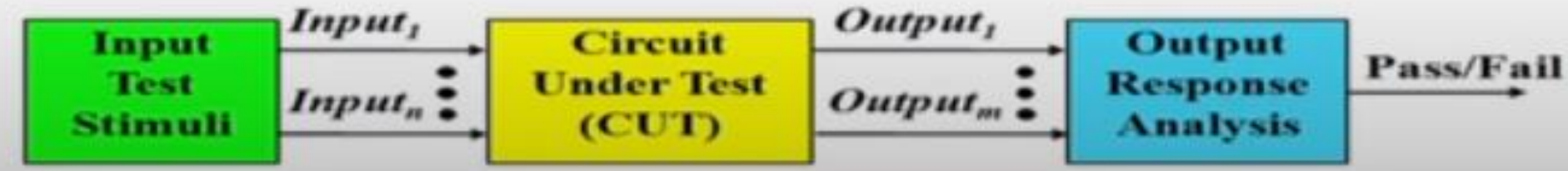
Failure mode analysis (FMA) of the failed devices provides further information for improving the VLSI processing.

Failing devices often show patterns of repeated failures.

The causes of these failures can point to weaknesses (sensitivity to process variations) in the design. Such information is useful for improving logic and layout design rules

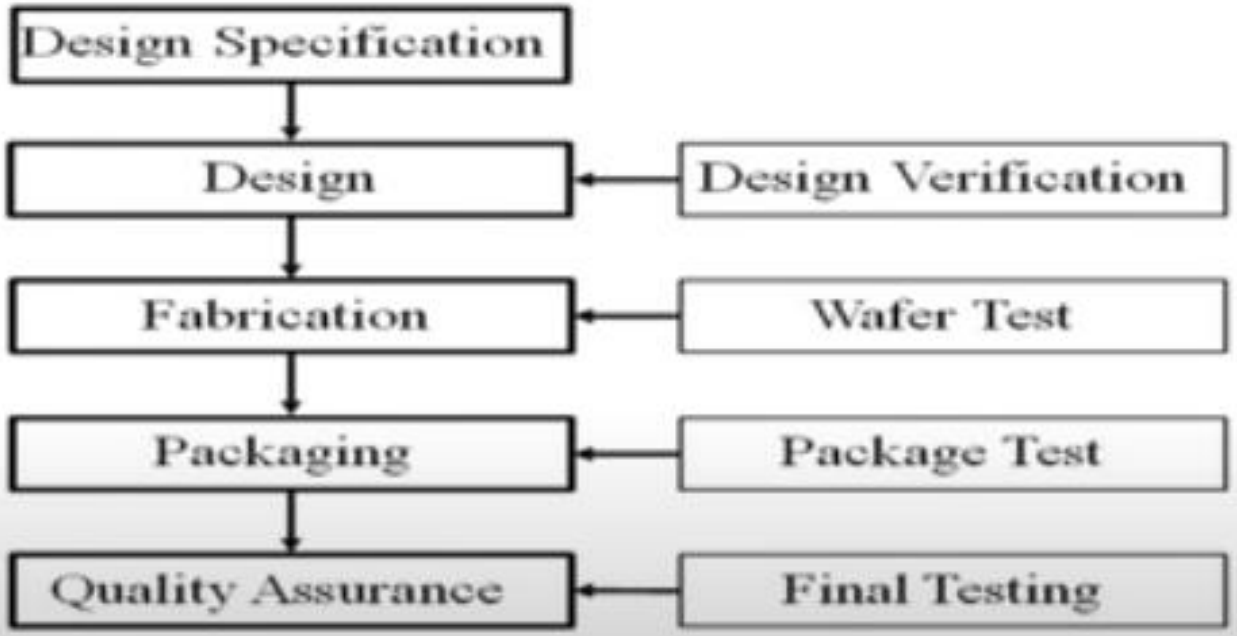
Testing During VLSI Life Cycle

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



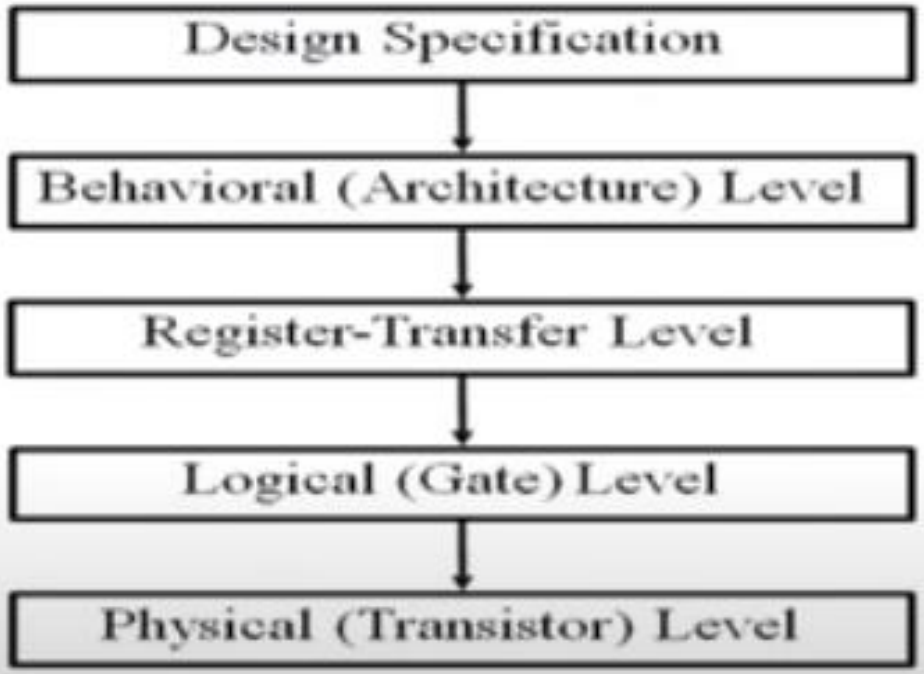
Testing During VLSI Development

- Design verification targets design errors
  - Corrections made prior to fabrication
- Remaining tests target manufacturing defects
  - A defect is a flaw or physical imperfection that can lead to a fault



Design Verification

- Different levels of abstraction during design
  - CAD tools used to synthesize design from RTL to physical level
- Simulation used at various level to test for
  - Design errors in behavioral or RTL
  - Design meeting system timing requirements after synthesis



Yield and Reject Rate

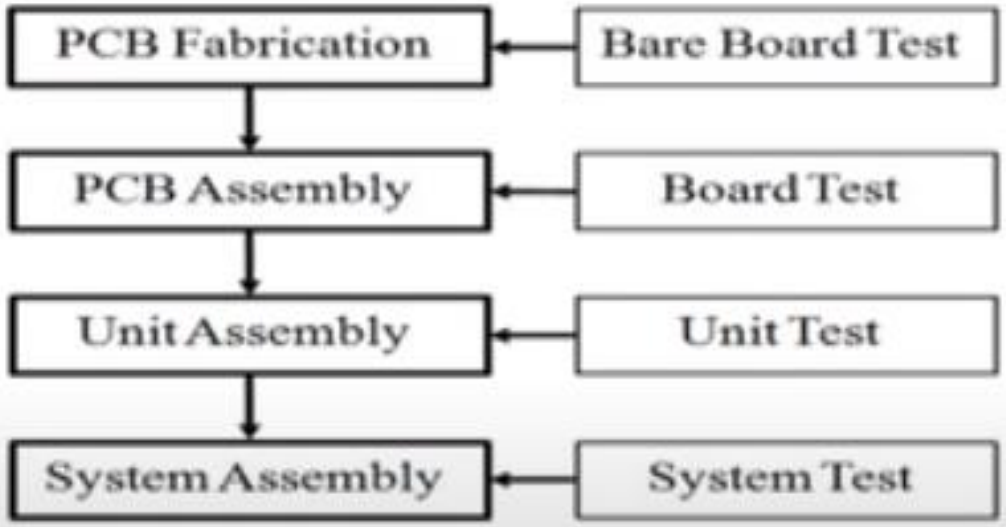
- We expect faulty chips due to manufacturing defects
  - Called yield

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

- 2 types of yield loss
  - Catastrophic – due to random defects
  - Parametric – due to process variations
- Undesirable results during testing
  - Faulty chip appears to be good (passes test)
    - Called reject rate $reject\ rate = \frac{number\ of\ faulty\ parts\ passing\ final\ test}{total\ number\ of\ parts\ passing\ final\ test}$
  - Good chip appears to be faulty (fails test)
    - Due to poorly designed tests or lack of DFT

Electronic System Manufacturing

- A system consists of
  - PCBs that consist of
    - VLSI devices
- PCB fabrication similar to VLSI fabrication
  - Susceptible to defects
- Assembly steps also susceptible to defects
  - Testing performed at all stages of manufacturing



FAULT MODELING

Defects, Errors, and Faults

Incorrectnesses in electronic systems are described in several ways.

A reader may find that the terms *defect*, *error*, and *fault* are sometimes used in confusing ways in the literature on testing.

Table 4.1: Typical printed circuit board (PCB) defects.

Defect type	Frequency of occurrence (%)
Shorts	51
Opens	1
Missing components	6
Wrong components	13
Reversed components	6
Bent leads	8
Wrong analog specifications	5
Defective digital logic	5
Performance defects	5

Definition

Defect : *A* defect *in an electronic system is the unintended difference between the implemented hardware and its intended design.*

Some typical defects in VLSI chips are:

1. Process Defects – missing contact windows, parasitic transistors, oxide breakdown, etc.
2. Material Defects – bulk defects (cracks, crystal imperfections), surface impurities, etc.
3. Age Defects – dielectric breakdown, electro migration, etc.
4. Package Defects – contact degradation, seal leaks, etc.

Defects occur either during manufacture or during the use of devices. Repeated occurrence of the same defect indicates the need for improvements in the manufacturing process or the design of the device.

Procedures for diagnosing defects and finding their causes are known as *failure mode analyses*

Fault Models

- A given fault model has k types of faults
  - $k = 2$ for most fault models
- A given circuit has n possible fault sites
- Multiple fault model –circuit can have multiple faults (including single faults
  - Number of multiple fault = $(k+1)^n-1$
    - Each fault site can have 1-of-k fault types or be fault-free
    - The "-1" represents the fault-free circuit
  - Impractical for anything but very small circuits
- Single fault model – circuit has only 1 fault
  - Number of single faults = $k \times n$
  - Good single fault coverage generally implies good multiple fault coverage

Physical defects

Stuck @ 0 & stuck @ 1

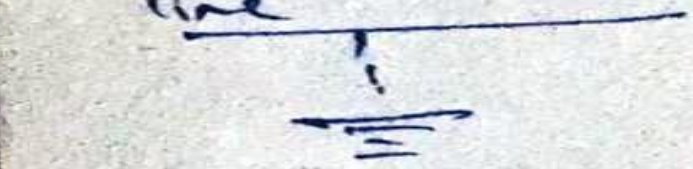
↓

line is taken

Permanently 0 ; (or) Permanently 1

↓

If is line is stacked with ground line is called stode @ 0



If line is stacked with supply line is called stuck @ 1

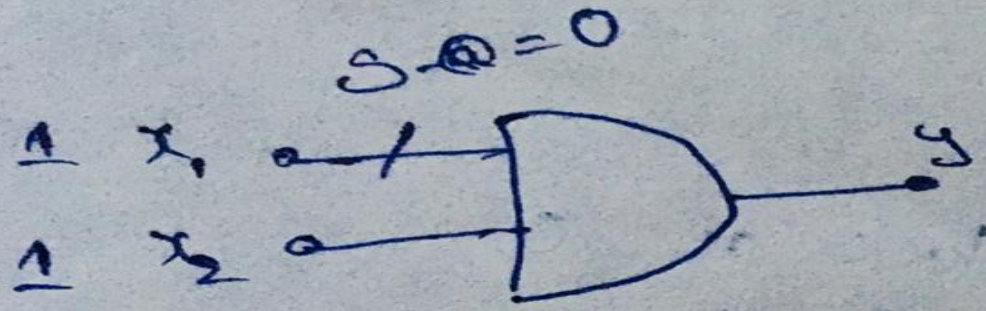
K = 2 are fault types

① stuck @ 0

② " @ 1

Eg :- An AND gate

Case (i)



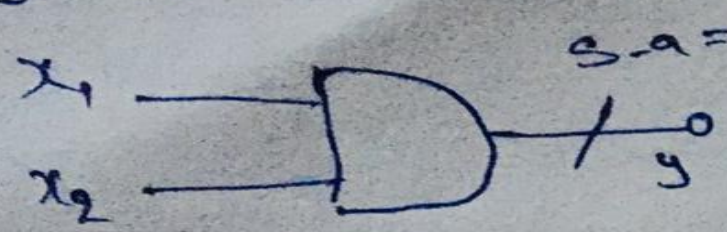
To check x_1 is stuck at 0 then we apply Pattern 1 1 as i/p's and see o/p.

→ If o/p is 1 then x_1 is not stuck at 0

→ If o/p is 0 then x_1 is said faulty.

Case (ii)

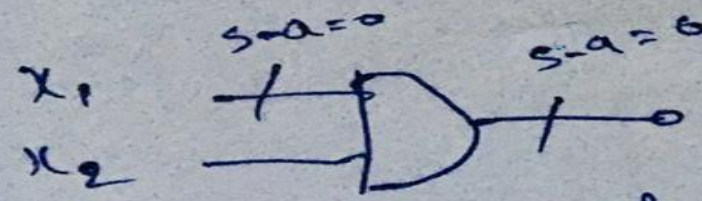
If o/p is stuck at '0'



→ If o/p is 1 then line is not stuck at 0

→ If o/p is 0 then line is stuck at 0

Case (iii) If for both faults x_1 & o/p are stuck at '0'



→ The response of a AND gate is same for all test cases.

Stuck at Faults

Single stuck at fault model is based on following :

a) Only one line is faulty
b) Faulty line is permanently set to either ‘0’ or ‘1’
c) Faulty can be at input or output of a gate

Fault Models

- Equivalent faults
  - One or more single faults that have identical behavior for all possible input patterns
  - Only one fault from a set of equivalent faults needs to be simulated
- Fault collapsing
  - Removing equivalent faults
    - Except for one to be simulated
  - Reduces total number of faults
    - Reduces fault simulation time
    - Reduces test pattern generation time

Definition:

Error: *A wrong output signal produced by a defective system is called an* error.

An error is an "effect" whose cause is some "defect."

Definition

Fault*: A representation of a "defect" at the abstracted function level is called a* fault.

The difference between a defect and a fault is rather subtle. They are the imperfections in the hardware and function, respectively.

Levels of Fault Models

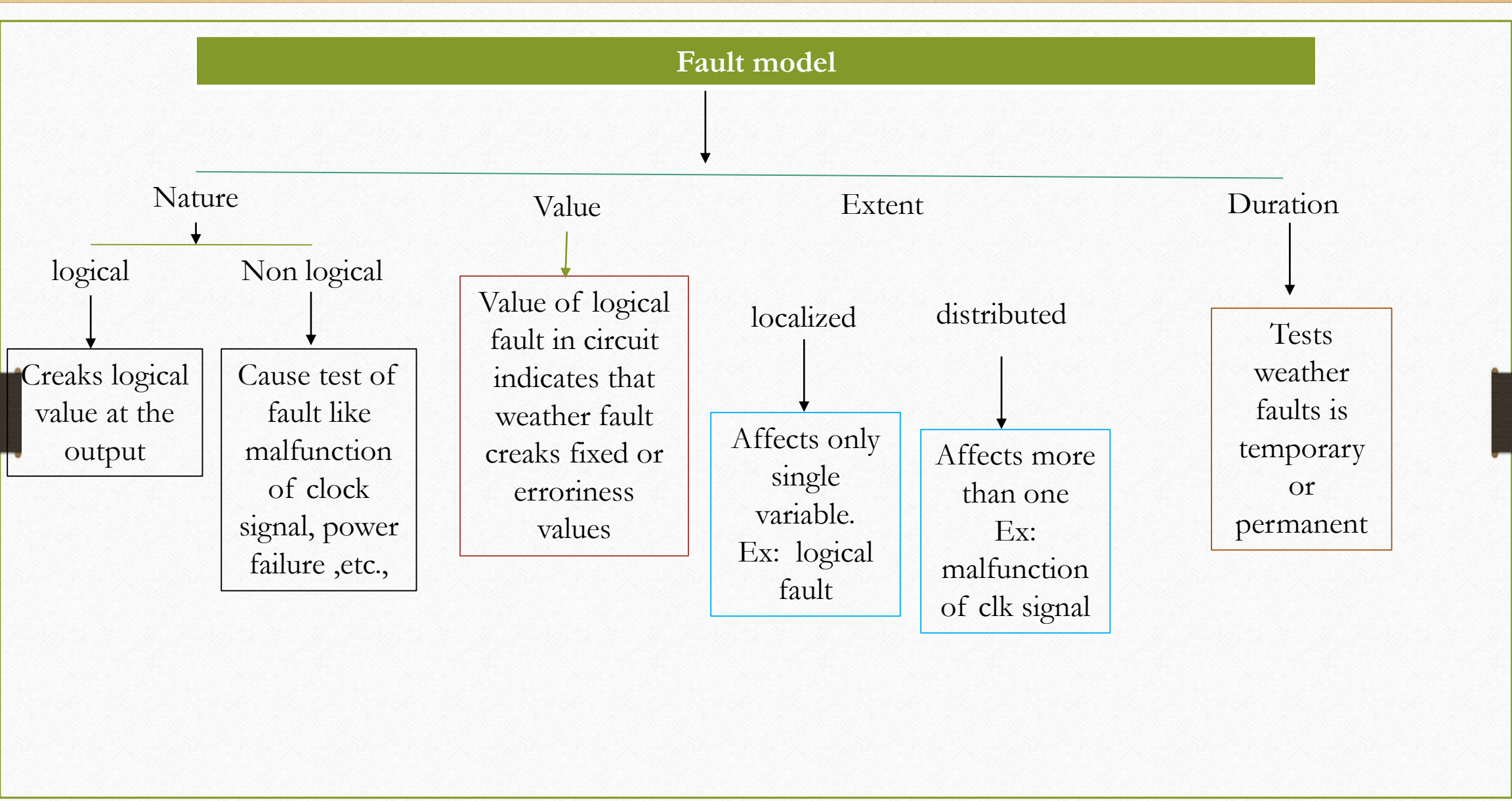
Modeling of faults is closely related to the modeling of the circuit.

- The ***behavioral level*** (sometimes referred to as *high level*).
- Highlevel fault models play a greater role in the simulation-based design verification, than in testing.
- Exceptions are the functional fault models of semiconductor memories.

- The ***register-transfer level*** **(RTL**) or logic level consists of a netlist of gates and the stuck-at faults at this level are the most popular fault models in digital testing.
- Other fault models at this level are *bridging faults* and *delay faults.*
- Transistor and other lower levels (referred to as ***component levels***) include stuck open types of faults that are also known as *technology-dependent* faults.

- Component level faults are mainly modeled in analog circuit testing

A typical example is the *quiescent current (IDDQ).*


Fault model
Nature
logical
Non logical
Creaks logical value at the output
Cause test of fault like malfunction of clock signal, power failure ,etc.,
Value
Value of logical fault in circuit indicates that weather fault creaks fixed or erroriness values
Extent
localized
distributed
Affects only single variable. Ex: logical fault
Affects more than one Ex: malfunction of clk signal
Duration
Tests weather faults is temporary or permanent

Types of Faults

Branch Fault: This fault is modeled at the behavioral level where the circuit function is described in a programming language.
A *branch fault* affects a branch statement and causes it to branch to an incorrect destination.

Bridging Fault: Usually modeled at the gate or transistor level, a *bridging fault* represents a short between a group of signals. The logic value of the shorted net may be modeled as 1-dominant (OR bridge), 0-dominant (AND bridge), or indeterminate, depending upon the technology in which the circuit is implemented.

Bus Fault: A *bus fault* specifies the status for each line in a bus as stuck-at-0, stuck-at-1, or fault-free. Thus, for an *n*-bit bus, there are 3^n -1 bus faults. A *total bus fault* assumes all lines of the bus to be stuck at the same 0 or 1 state

Cross-point Fault: These faults are modeled in *programmable logic arrays* (PLA.)
In the layout of a PLA, input and output variable lines are laid out perpendicular to the product-lines.

There are two types of cross-point faults.

- A *missing cross-point* means a missing connection at a crossing where a connection was intended.
- An *extra crosspoint* means a faulty connection at a crossing where no connection was intended.

Based on their influence on the logic function of the PLA, the cross-point faults are further classified as *shrinkage*, *growth*, *appearance*, and *disappearance* faults.

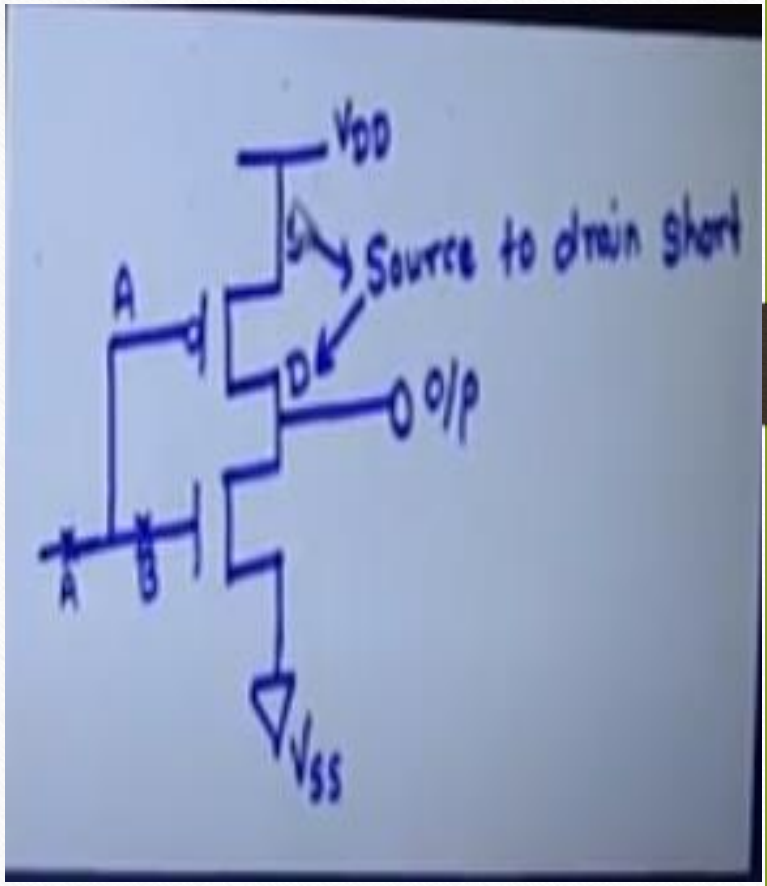
Defect-Oriented Faults: Faults at the physical level that usually occur during manufacture are called *defects*. The electrical or logic-level faults that can be produced by physical defects are classified as *defect-oriented faults*.

Examples of physical defects are broken (open) wires, bridges, improper semiconductor doping, and improperly formed devices.

Functional Faults

Transistor Level Faults

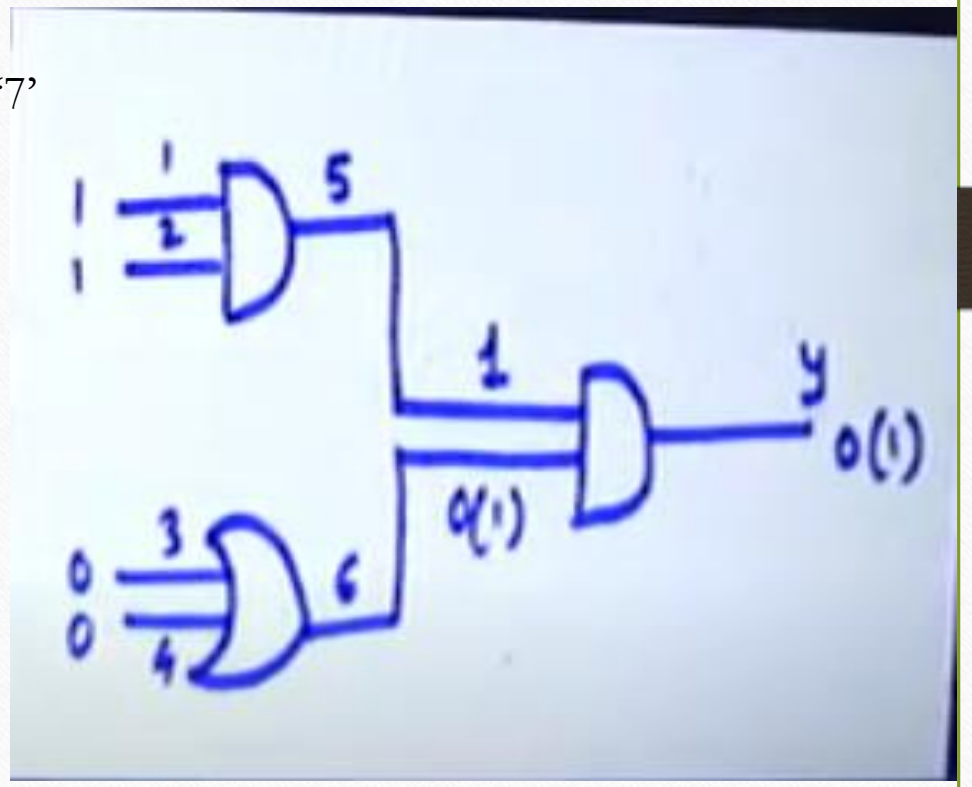
fAULT	INPUT	OUTPUT	COMMENT
Source to drain short	1 0	1 1	Output is always High
Break at “B”	0 1	1 X	PMOS is ON NMOS is Floating
Break at “A	1 0	X X	Floating output depends on threshold voltage (Vbin)
“A” Stuck at 1	1 0	0 0	Output is always Low



Gate Level Faults

Fault is said to be single stuck at fault at a line "L" of the circuit if the line permanently assumes a logic value either '0' or 1

If input "1100" is applied and Line '6' is stuck at '1', the output at '7' will be '1' in place of expected '0'



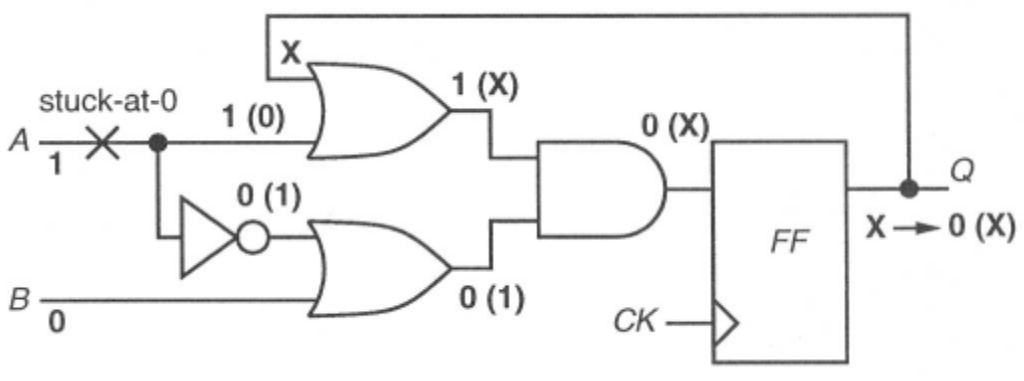
Delay Fault: These faults cause the combinational delay of a circuit to exceed the clock period. Specific delay faults are *transition faults*, *gate-delay faults*, *line-delay faults*, *segment-delay faults*, and *path-delay faults.*

Gate-Delay Fault: The fault increases the input to output delay of a single logic gate, while all other gates retain some nominal values of delay.
The increase in the delay of the faulty gate is called the *size* of the gate-delay fault.

Hyperactive Fault: A hyperactive fault causes a large number of signals in the circuit to differ from their correct values.
The fault thus produces very high faultrelated activity in the circuit.
If not readily detected, fault simulators usually remove hyperactive faults for later consideration to save CPU time and memory

Initialization Fault: Circuits with memory elements (e.g., flip-flops) are designed so that they can be initialized by applying suitable input signals.

Faults that interfere with such an initialization procedure are called *initialization faults.*

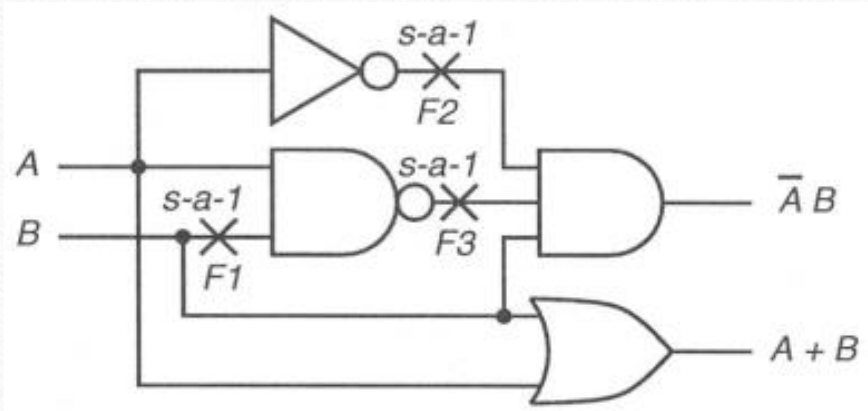


Instruction Fault: Usually modeled in programmable systems like microprocessors or digital signal processors, an *instruction fault* causes an intended instruction to be incorrectly executed.

Intermittent Fault: A fault that appears and disappears as a function of time is called an *intermittent fault.*

Line-Delay Fault: This fault models rising and falling delays of a given signal line.
In contrast with the transition fault where the transition can be propagated through any path, a test for a *line-delay fault* must propagate the transition through the longest sensitizable path.

multiple stuck-at faults



Multiple stuck-at fault	Output at $\overline{A}\,B$	Test
F1, F2	$\overline{A}\,B$	*Redundant*
F1, F3	$\overline{A}\,B$	*Redundant*
F2, F3	*B*	11
F1, F2, F3	*B*	11

Logical Faults: These faults affect the state of logic signals. Normally, the state may be modeled as {0, 1, *X* (unknown), *Z* (high impedance)}, and a fault can transform the correct value to any other value.

Memory Faults: Faults modeled in memory blocks are *single cell stuck-at*-[0,l] faults, *pattern sensitive faults*, *cell coupling faults*, and single stuck-at faults in the address decoder logic

Non-classical Fault: Although a *non-classical fault*, in general, refers to a fault other than a stuck-at fault, the term has been used for the stuck-open and stuck-short faults of MOS technologies

Oscillation Fault: These faults cause oscillating signals in the faulty circuit when the fault-free circuit remains stable. Such a condition can occur due to certain single stuck-at faults in sequential circuits that contain combinational feedback.
Oscillations can also occur in a purely combinational circuit if a bridging fault produces feedback. Oscillation faults are also referred to as *star-faults*

Parametric Fault: Such a fault changes the values of electrical parameters of active or passive devices from their nominal or expected values.
Examples are the threshold voltage of a transistor (active device) and values of resistors and capacitors (passive devices.)

Path-Delay Fault:

- This fault causes the cumulative propagation delay of a combinational path to increase beyond some specified time duration.
- The combinational path begins at a primary input or a clocked flip-flop, contains a connected chain of logic gates, and ends at a primary output or a clocked flip-flop.
- The specified time duration can be the duration of the clock period (or phase), or the vector period.
- Propagation delay is defined for the propagation of a signal transition through the path.
- Thus, for each combinational path there are two path-delay faults, which correspond to the rising and falling transitions, respectively.

Pattern Sensitive Fault: This fault causes an incorrect behavior in a certain part of the circuit only when a specific state occurs in some other part.

Permanent Fault: Any faulty behavior that does not change with time is called a *permanent fault.*

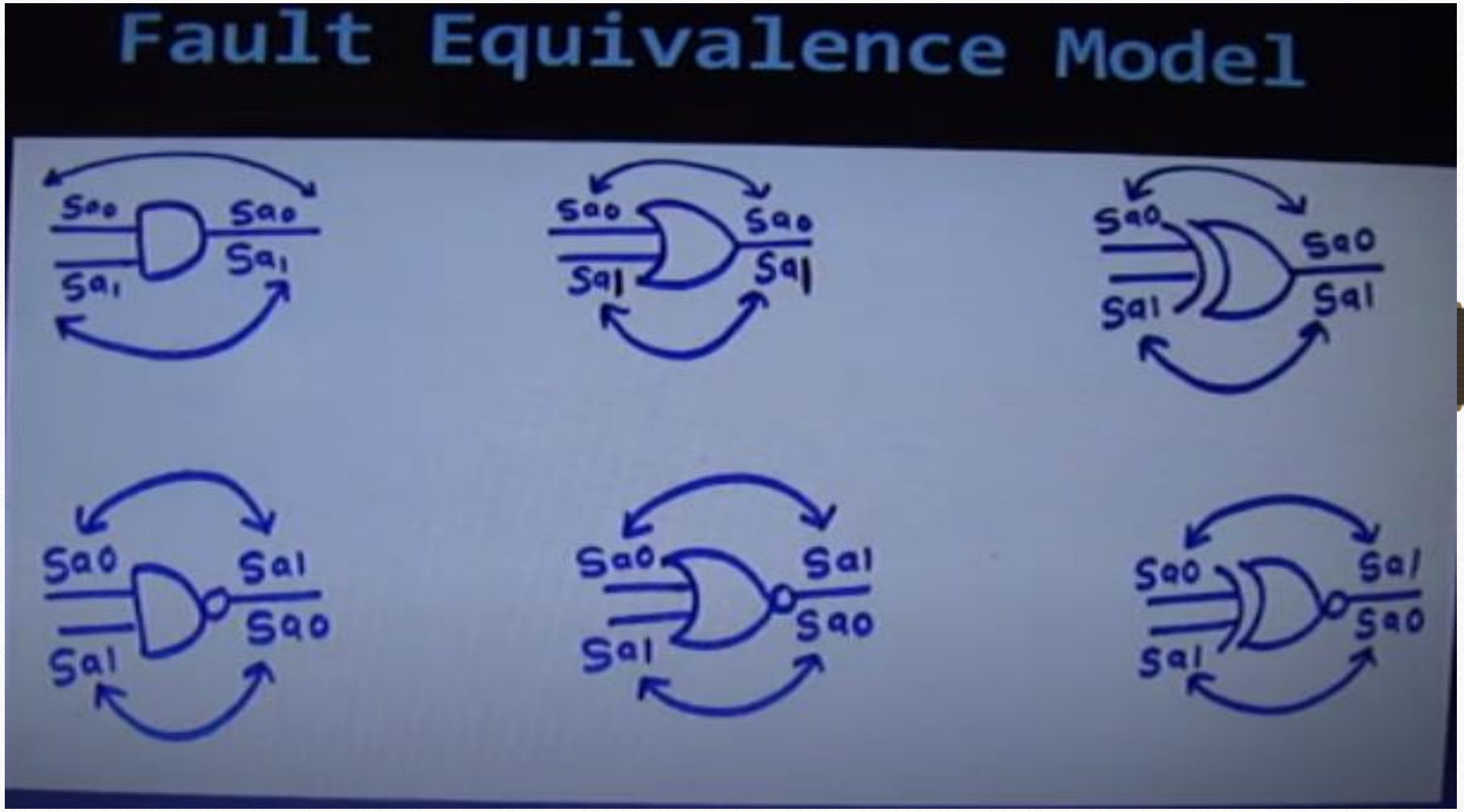
Physical Faults: These faults cause physical changes in the circuit.
Examples of physical faults are broken wires, bridges (shorts) between conductors carrying unconnected signals, shorted or open transistors, etc.

Pin Fault: When a circuit is modeled as an interconnect of modules, the terminals of those modules are referred to as *pins*. This term is adopted from the technology of *printed circuit boards* (PCBs), which contain interconnecting wiring between the pins of the mounted chips.

Fault Equivalence Model for Test Vectors

Ex: gates : AND , OR, EX-0R

NAND, NOR, EX-NOR



Fault Equivalence :

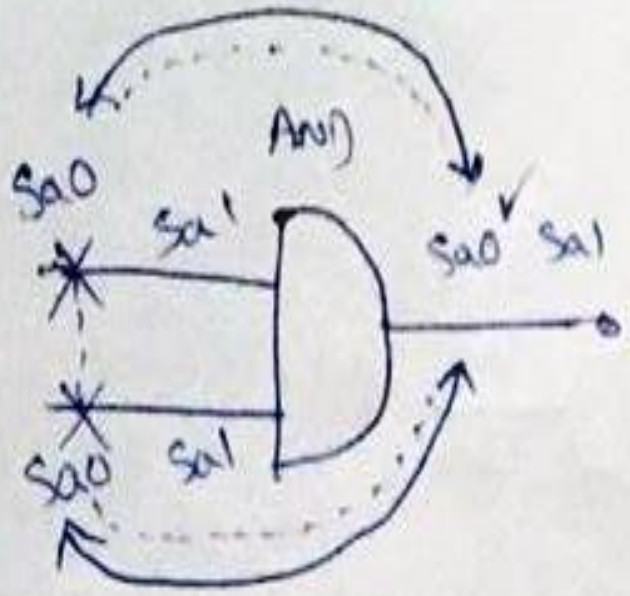
Equivalence Rules : handwritten\1.jpeg
Example : handwritten\2.jpeg

Fault Dominance:

Dominance Rules :
Example : handwritten\3.jpeg

Equivalence Rules:-

(a)



(1). 6-Possible faults in AND gate

(2). Dotted indicates equivalent

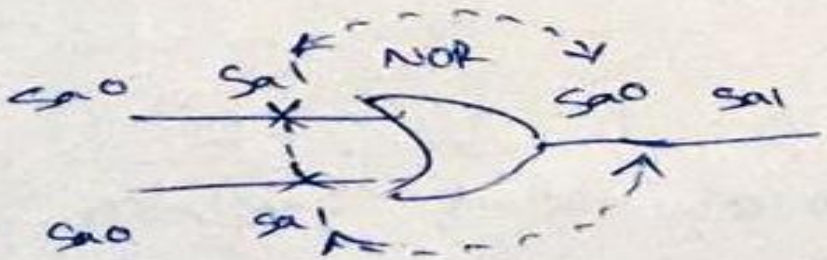
(3). X → Delet possibles (indication)

(4) Out of 6-Possible we get down to 4-faults

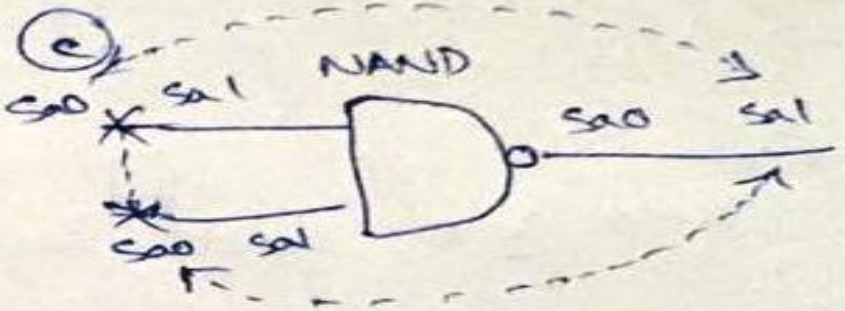
(5) Since at input side sa0 of 2-ip got deleted due to Equivalence.

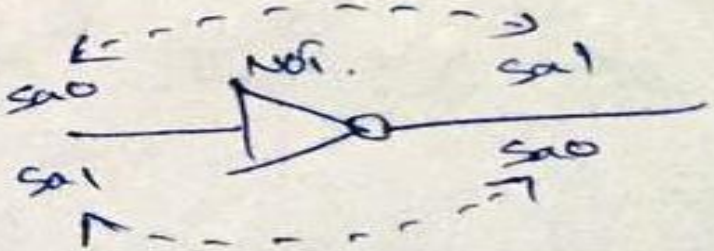
(b)


sa0
sa1
OR
sa0
sa1
sa0
sa1

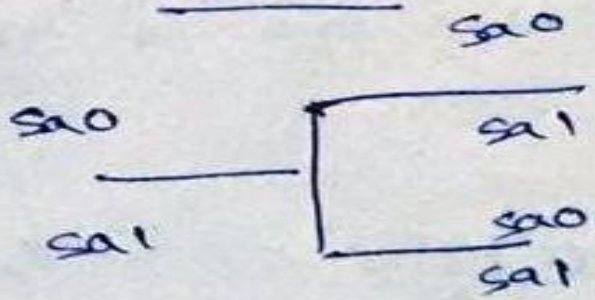

NOR
sa0
sa1
sa0
sa1
sa0
sa1

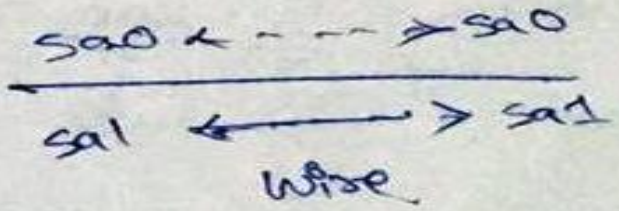
(c)


NAND
sa0
sa1
sa0
sa1
sa0
sa1

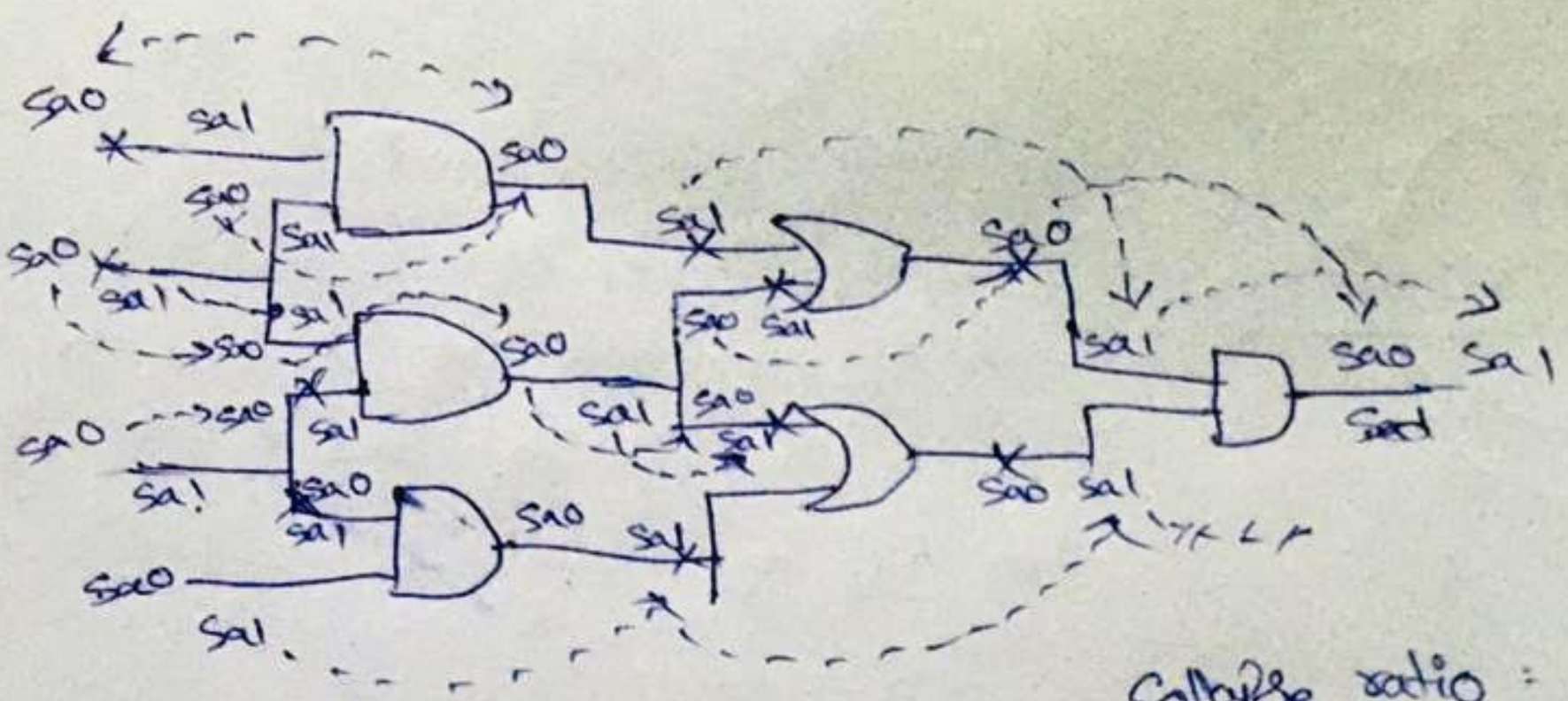

NOT
sa0
sa1
sa1
sa0

FANOUT


sa0
sa0
sa1
sa1
sa0
sa1


sa0
sa0
sa1
sa1
wire

Eg:- Fault Equivalence



Collapse ratio : $\frac{20}{32} = 0.625$

Step 1:-

→ For AND gate delet Sa0 at inputside

Step 2:-

→ for OR gate input delete Sa1

Step :-3 :-

→ For fanout branches we cannot delet anything

→ Total 32 faults in circuit

→ ON deleting of Sa0 & Sa1 at AND and OR gates we have 20 faults remained

$$\therefore \text{Collapse ratio} = \frac{20}{32} = 0.625. \Rightarrow 62\%$$

Fault Dominance :-

→ one fault dominates the another.

* ~~If~~ ~~all~~ ~~tests~~ ~~faults~~ ~~F_1 F_2 F_3~~ ~~detects~~ ~~another~~ ~~fault~~ ~~F_2~~ ~~then~~ ~~F_1~~

→ If all test for some faults F_1 detects another fault F_2 then F_2 is said to dominate F_1.

→ Dominance fault Collapsing :-

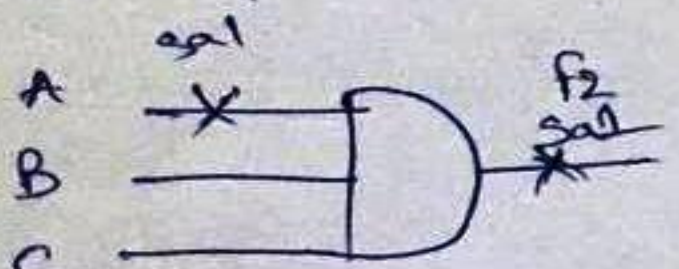
→ If fault F_2 dominates F_1 then F_2 is removed from fault list.

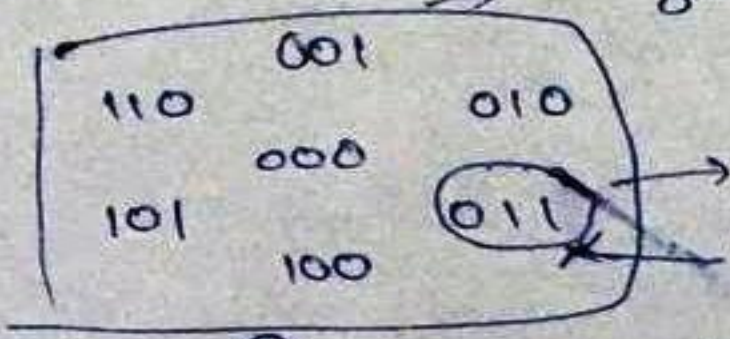
→ If two faults dominate each other then they are Equivalent.

* all pssible

→ If two faults dominate

Eg:- AND





All test of f_2 → remaining will act for f_2 all possible

→ first vector for F_1

→ only test of 11

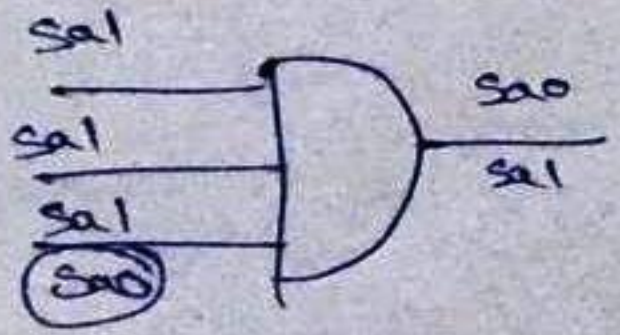
Test vectors

f_1 (To detect fault we have to apply reverse logic I/P with A=0)

A=0
B=1
C=1

(*) where f_2 dominates f_1 so we can delete f_2 from fault list.

Sol



If n-I/P AND gate no:- of stuck at faults to be considered are

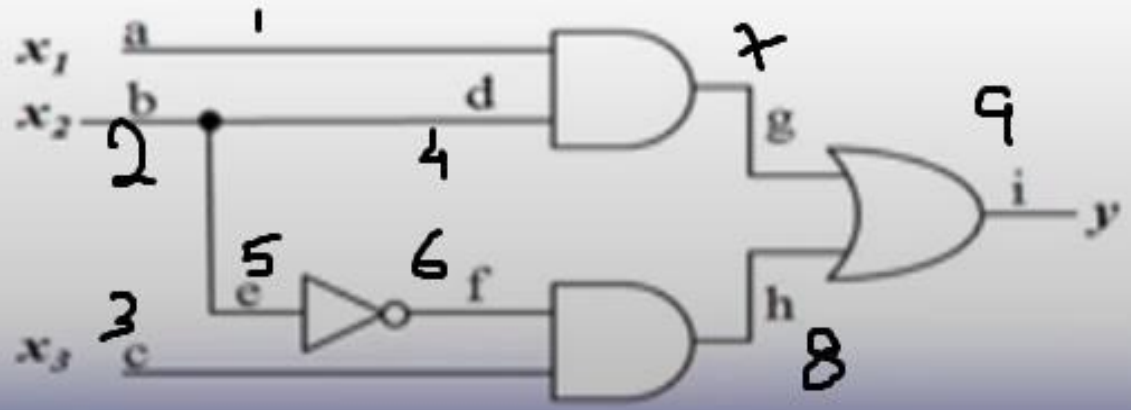
(n+1)

same for OR, AND, & NOT gate.

Stuck-at Faults

- Any line can be
  - Stuck-at-0 (SA0)
  - Stuck-at-1 (SA1)

  # fault types: k=2
- Example circuit:
  - # fault sites: n=9
  - # single faults =2×9=18



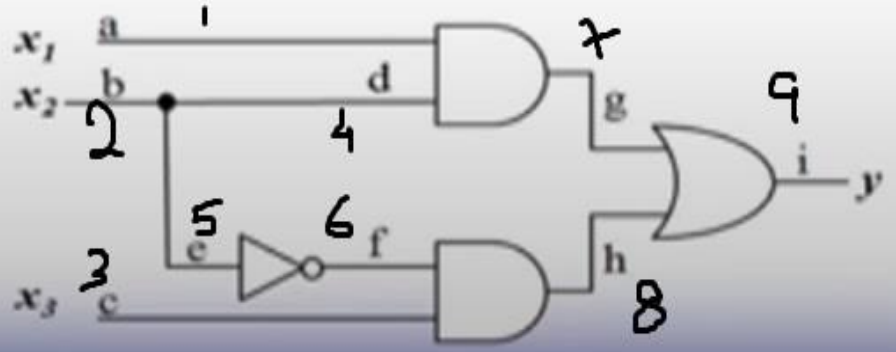
Truth table for fault-free behavior and behavior of all possible stuck-at faults

$x_1x_2x_3$	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

- Any line can be
  - Stuck-at-0 (SA0)
  - Stuck-at-1 (SA1)

  # fault types: k=2
- Example circuit:
  - # 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

indicates Responce in defering

Stuck-at Faults

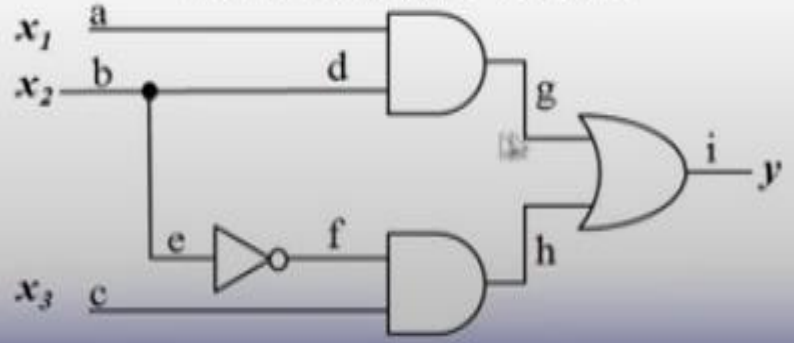
- Valid test vectors
  - Faulty circuit differs from good circuit
  - Necessary vectors:

    011 detects f SA1, e SA0

    100 detects d SA1

    – Detect total of 10 faults

    – 001 and 110 detect remaining 8 faults



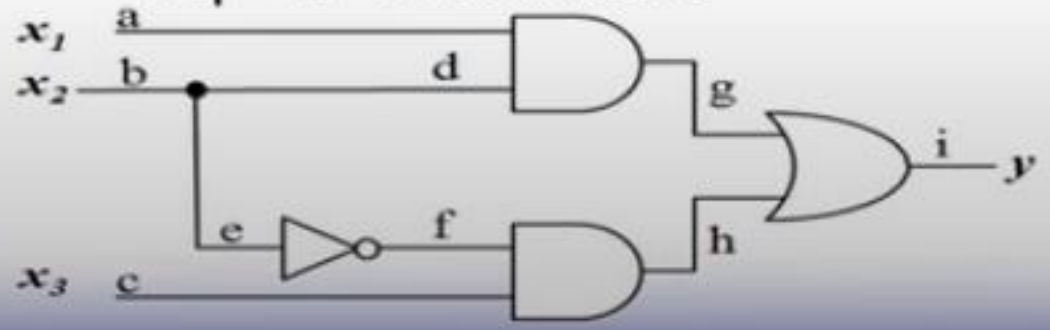
Nessary vectors

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

- 4 sets of equivalent faults
- # collapsed faults = $2\times(P_O+F_O)+G_I-N_I$
  - P_O= # primary outputs
  - F_O= # fanout stems
  - G_I= # gate inputs
  - N_I= # inverters

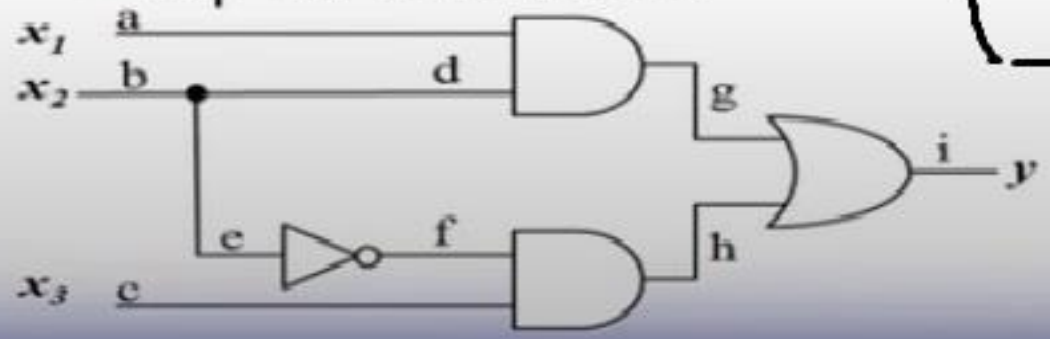


Truth table for fault-free behavior and behavior of all possible stuck-at faults

$x_1x_2x_3$	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

- 4 sets of equivalent faults
- # collapsed faults = $2\times(P_O+F_O)+G_I-N_I$
  - P_O= # primary outputs
  - F_O= # fanout stems
  - G_I= # gate inputs
  - N_I= # inverters

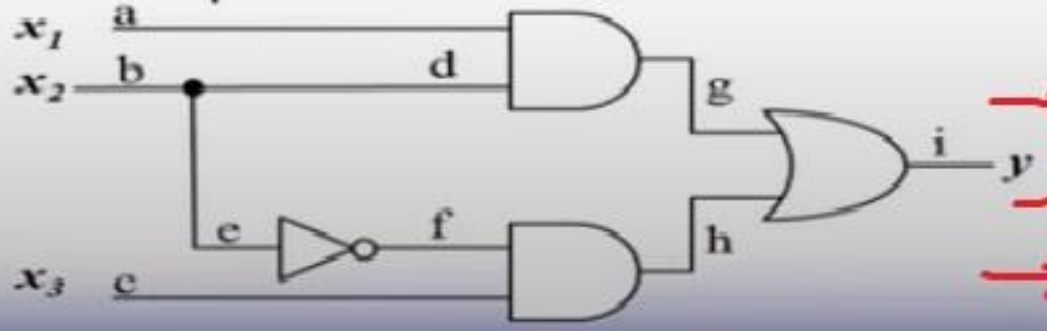


Truth table for fault-free behavior and behavior of all possible stuck-at faults

$x_1x_2x_3$	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

- 4 sets of equivalent faults
- # collapsed faults = $2\times(P_O+F_O)+G_I-N_I$
  - P_O= # primary outputs
  - F_O= # fanout stems
  - G_I= # gate inputs
  - N_I= # inverters



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

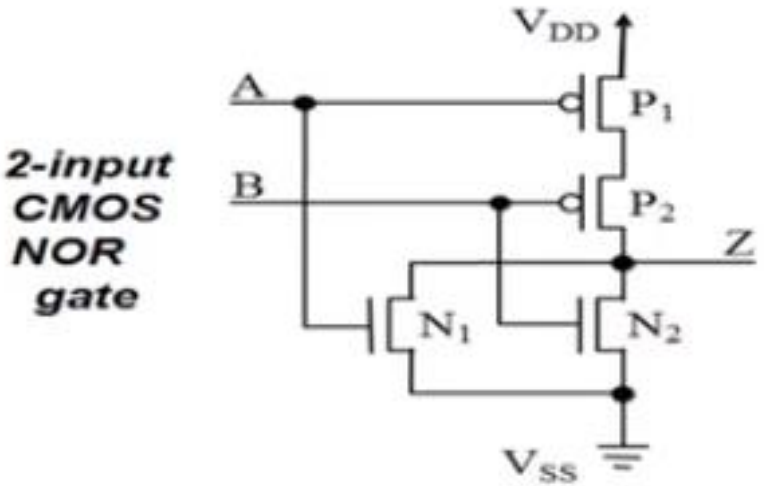
Stuck-at Faults

- # collapsed faults = $2\times(P_O+F_O)+G_I-N_I$
  - P_O= number of primary outputs
  - F_O= number of fanout stems
  - G_I= total number of gate inputs
    for all gates including inverters
  - N_I= total number of inverters
- For example circuit, # collapsed faults = 10
  - P_O= 1, F_O= 1, G_I= 7, and N_I= 1
- Fault collapsing typically reduces number of stuck-at faults by 50% - 60%

Transistor Faults

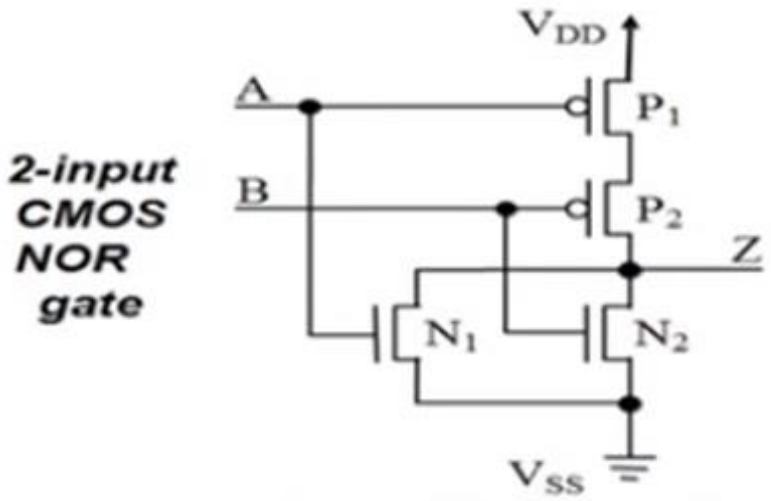
- Any transistor can be
  - Stuck-short
    - Also known as stuck-short
  - Stuck-open
    - Also known as stuck-open

    # fault types: k=2
- Example circuit
  - # 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-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



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

if A=1 and B=0 in case of N1 stuck open then Output is stored with previous value of Z

if N1 short then current flows through so I_{DDQ} Quicent current

Transistor Faults

- # collapsed faults = $2 \times T - T_S + G_S - T_P + G_P$
  - T = number of transistors
  - T_S = number of series transistors
  - G_S = number of groups of series transistors
  - T_P = number of parallel transistors
  - G_P = number of groups of parallel transistors
- For example circuit, # collapsed faults = 6
  - T=4, T_S= 2, G_S= 1, T_P= 2, & G_P= 1
- Fault collapsing typically reduces number of transistor faults by 25% to 35%

THANK YOU