

ECE 1767

Design for Test and Testability

Andreas Veneris
Department of Electrical and Computer Engineering
University of Toronto

Outline

- Testing vs. Design Verification
- Fault Models
- Fault Simulation
- Test Generation
- Fault Diagnosis
- Design for Testability
- Modeling at Logic Level
- Binary Decision Diagrams (BDDs)
- Simulation with Unknown Values

Testing vs. Design Verification

- Design/Test issues:
 - ◆ **DIGITAL** meets **ANALOG**
 - ◆ Byzantine world: **Hard** vs. **Soft** faults
 - ◆ Leakage Currents
- Structural issues:
 - ◆ Scaling and new defects
 - ◆ Interconnect related defects

CONCLUSION:

“ The greatest of all faults is to be conscious of none”

Testing vs. Design Verification

- Testing - Is the fabricated chip working?
 - ◆ **Physical defects (shorts, opens, bridging)**
- Design Verification - Is the design correct?
 - ◆ **Wrong wiring, incorrect gates, incorrect interpretation of specs, incorrect algorithm**
- Error Detection - Is the result correct?
 - ◆ **On-line testing**

Testing vs. Design Verification

- In contrast to design verification, **testing** is task of verifying that manufactured chip functions as designed.
 - ◆ The design is assumed to be correct.
- Every chip is checked before shipping to see if it works.
 - ◆ Typically, go/no-go tests are performed.
- Input signals are applied to chip inputs, and chip outputs are monitored and compared to the expected correct response.
 - ◆ This test should be **very fast**, because tester time is expensive.

Physical Faults vs. Design Errors

Physical Faults

- shorts, opens, stuck-at-0, stuck-at-1 resulting from fabrication defects
- functional faults

Design Errors

- forgot to complement
- wrong connection
- wrong gate
- wrong algorithm

Functional Errors

Errors resulting from physical defect or design error; e.g., produces wrong addition in an adder, or decoder selects wrong address.

Three Classes of Physical Faults

- Permanent
 - ◆ **Shorts, opens, etc.**
- Intermittent
 - ◆ **Can be excited with a nonzero probability**
- Transient
 - ◆ **Caused by random events**
 - ◆ **e.g., alpha particles, electrical noise**

Design Verification

- **Design verification** is the task of verifying that a design meets its specification
 - ◆ Typically, it is performed **in parallel** with integrated circuit design.
 - ◆ Both **simulation** and **formal verification** are used to verify a design.
- Design verification is done at various levels of abstraction
 - ◆ At the **architectural level**, use functional or behavioral simulator
 - ◆ At the **register-transfer level** (RTL), use an RTL simulator
 - ◆ At the **logic level**, use a logic/timing simulator
 - ◆ At the **transistor level**, use a switch/circuit simulator

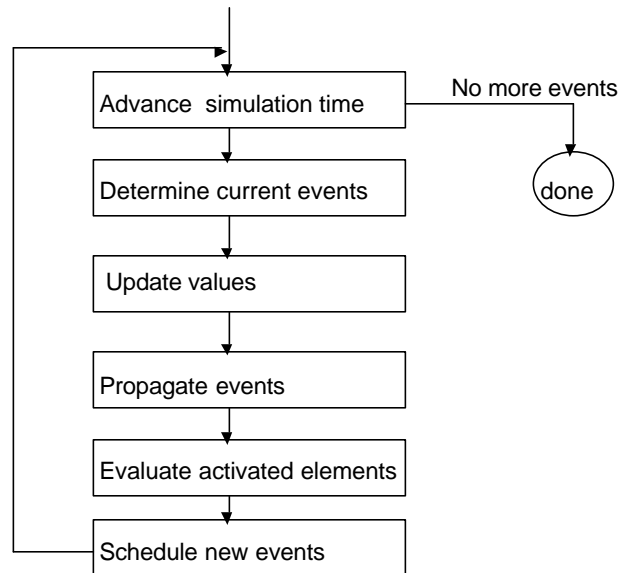
Formal Verification

- Recently, **formal verification** techniques have become popular.
- **Equivalence checkers** prove an implementation is correct because it is **functionally equivalent** to its specification (a higher-level implementation).
 - ◆ e.g., compare RTL design to gate-level design
 - ◆ This proof is done **formally**, i.e., mathematically, using binary decision diagrams (**BDDs**).
- **Model checkers** verify specified properties of a design.
 - ◆ The designer must specify what aspects of the design to check.

Simulation

- With simulation, the output response is computed for a given set of inputs
 - ◆ The output response must then be compared to the expected response.
 - ◆ A set of input vectors must be generated that will fully exercise the circuit being simulated.
 - ◆ Commercial tools for simulation at various levels of abstraction have been available for many years.
 - ◆ Tools to generate vectors, both random and biased random, have typically been developed in-house.
 - ◆ Commercial tools to assist in vector generation have become available

Simulation

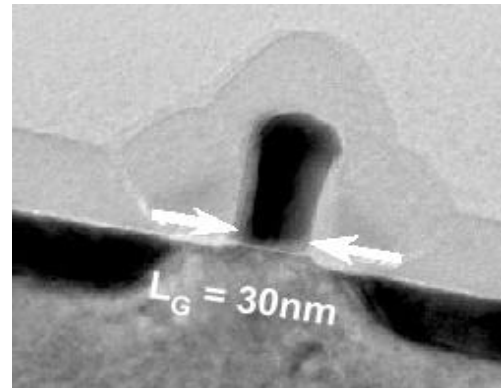
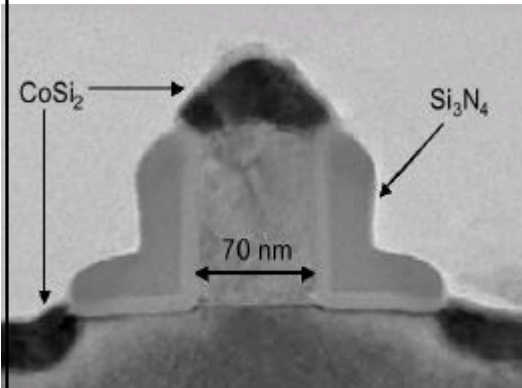


Why Testing is Emerging?

- New nanometric devices
- Miles of multiple level of **interconnect**
- New HARD and SOFT defects:
 - ◆ **Parametric or Catastrophic**
- Growing complexity of SoC:
 - ◆ **70nm technology to be produced in 2005**
 - ◆ **Developed technology is 25nm MOS transistor (MIT)**

Why Testing is Emerging?

70 nm Technology Generation (in production 2005)

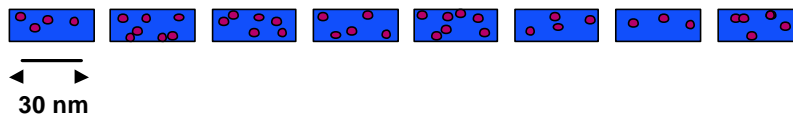


ECE 1767 Courtesy INTEL Corporation

University of Toronto

Why is Testing Emerging?

Large Parameter Variations

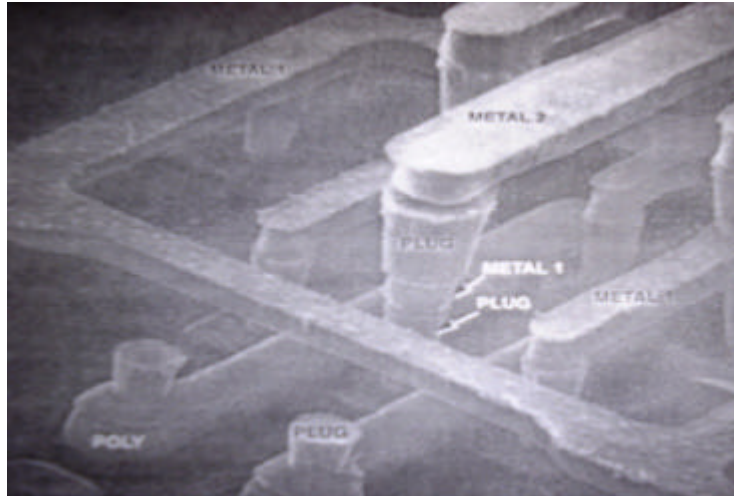


- Average = 5.25 dopant atoms/transistor (variation -43% to +33%)
- Complex architecture: 6 to 10 levels
- Crosstalk and long distances (miles)

ECE 1767

University of Toronto

Multi-level INTERCONNECT

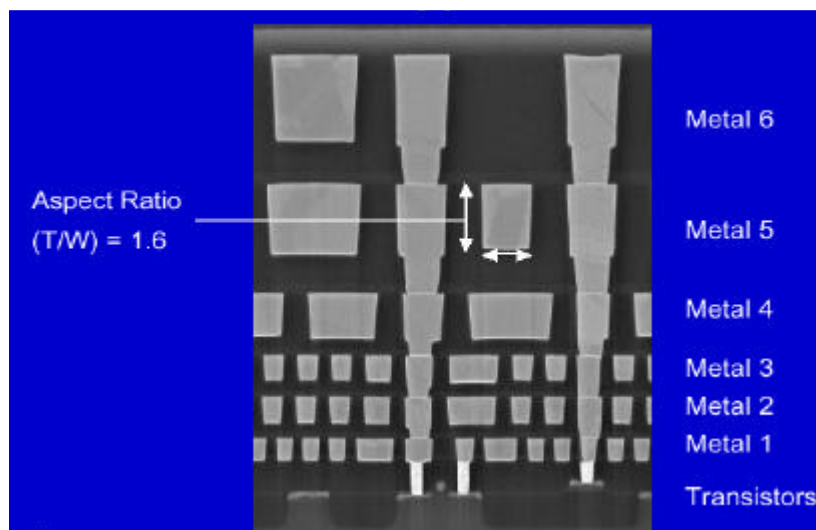


Courtesy of ICE. Integrated Circuit Engineering Co.

ECE 1767

University of Toronto

Interconnect Architecture

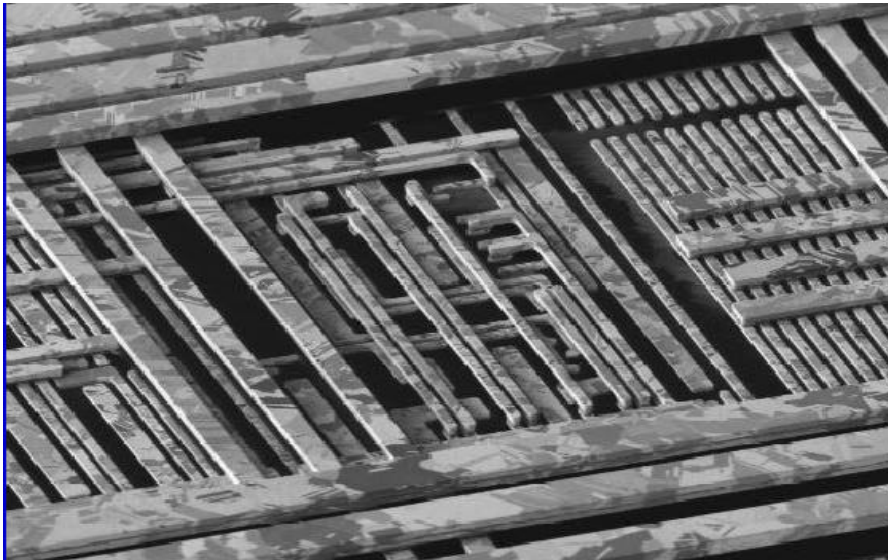


ECE 1767 Courtesy INTEL Co

S.Tyagy et al. IEDM December 2000

University of Toronto

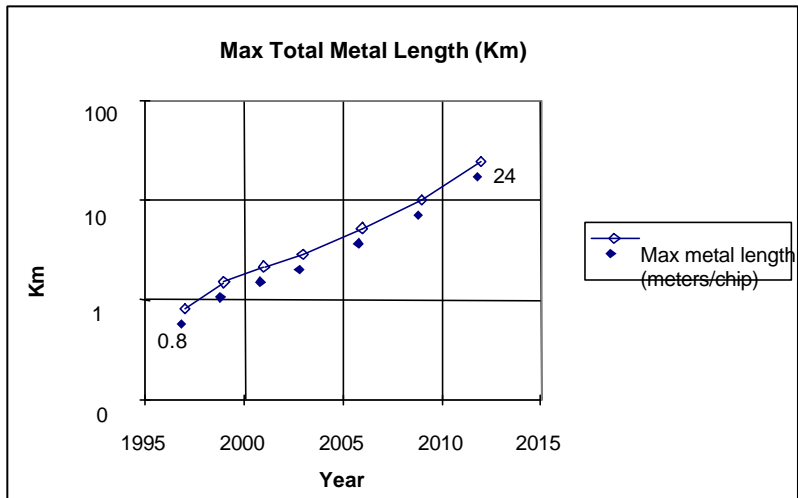
Interconnect Architecture



ECE 1767

Courtesy INTEL Co

University of Toronto



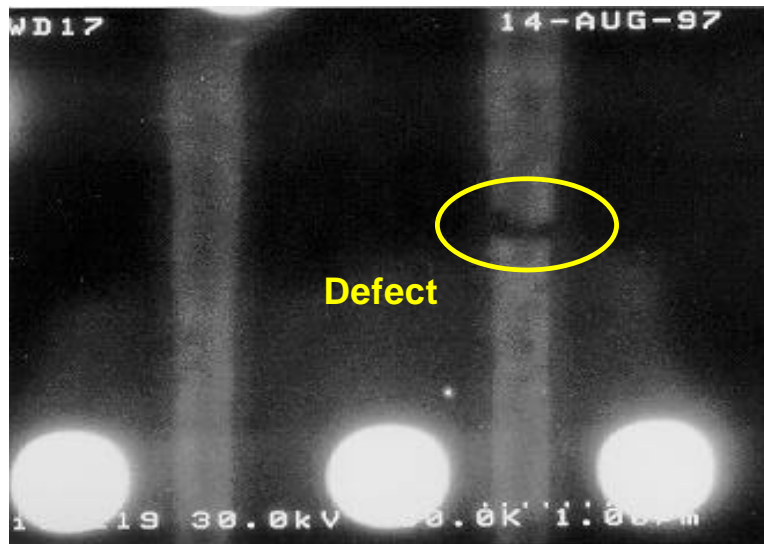
SIA Roadmap 1997

- **Internal interconnect distances**

ECE 1767

University of Toronto

Metal OPEN Defect



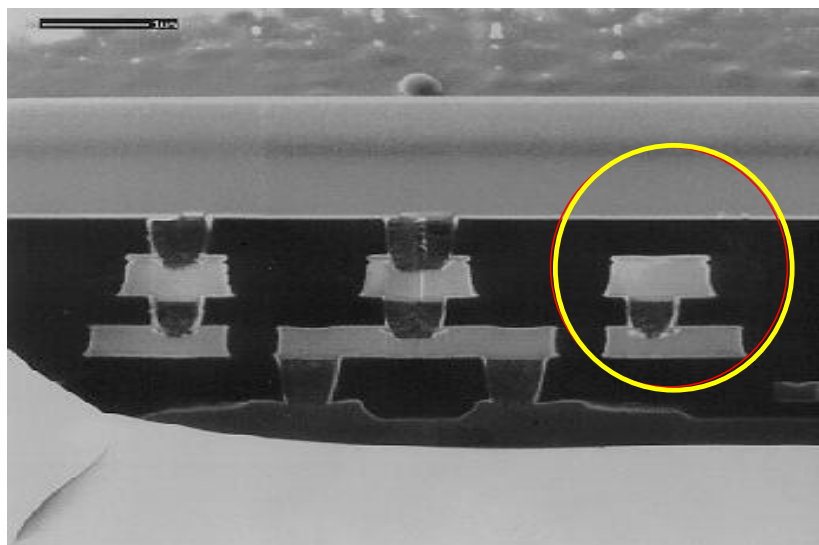
Courtesy INTEL Co

Cheryl Prunty ITC 1998

ECE 1767

University of Toronto

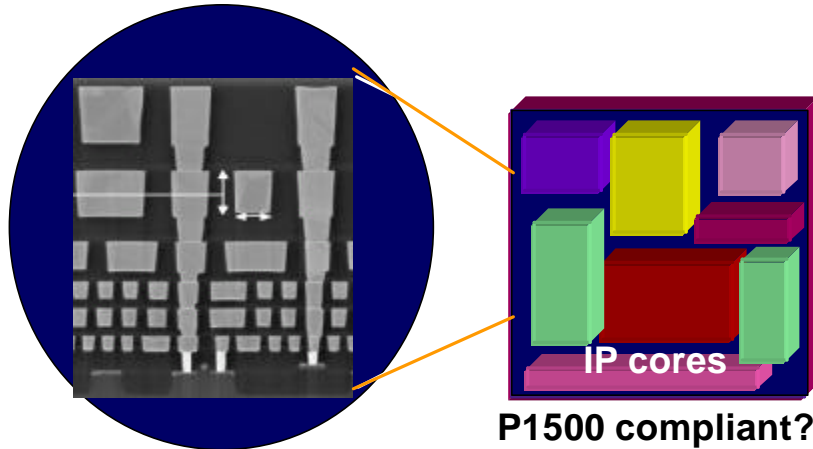
Full OPEN – Missing VIA Defect



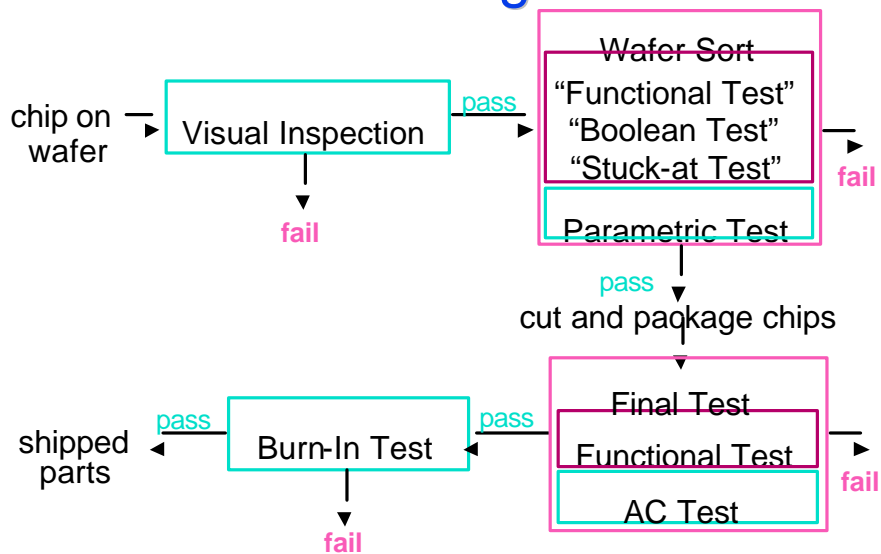
ECE 1767

University of Toronto

Complexity related issues: SoC



Testing



Wafer Sort

- Immediately after wafers are fabricated, they undergo preliminary tests in **Wafer Sort**.
- The tests applied at wafer sort may be a subset of the tests applied at **Final Test** after the chips are packaged.
- If a die fails a wafer sort test, it is marked with a drop of ink.
 - ◆ **Only unmarked dice are packaged.**
- Yield is defined as the fraction of dice that pass the tests.

Process and Product Qualification

- New processes and products must undergo an extensive **qualification** step at the end of the manufacturing process.
 - ◆ Parametric tests are performed to check on noise margins, propagation delays, and maximum clock frequencies under various temperatures and supply voltages.
 - ◆ Testing is also performed after a **burn-in** process, in which a sample of packaged chips is placed in high temperature **ovens** under electrical stress.
- With an immature technology, or a new design, the qualification results are used to improve the process or alter the design.

Testers

- A chip is tested by a complex machine, called a **tester**.
 - ◆ The tester is set up to apply specific inputs to the chip and monitor its outputs.
 - ◆ A **probe card** is needed to interface the tester to a wafer, and a **DUT** (Device Under Test) **Board** is needed to interface the tester to a packaged chip.
 - ▲ The probe card used depends on the pad arrangement, and the DUT board depends on the package.
- In order to avoid discarding good chips, the tester must be highly accurate: timing and noise are critical.
- As a result, a tester usually includes a computer and highly precise instrumentation and is usually very expensive.

Functional Testing

- The tester applies a set of 0/1 patterns to the chip inputs, called **test vectors**.
- The test vector set should be as small as possible to minimize time.
- The **central problem** in testing is to generate a test vector set that will detect as many potential defects as possible.
- A combinational circuit with n inputs accepts 2^n different input patterns.
- An exhaustive test would require the application of 2^n test vectors.

Functional Testing

- A sequential circuit with n primary inputs and m latches requires 2^{n+m} vectors, assuming the latches are all directly controllable, as in full scan.
- Thus, it would take about 3.8×10^{22} test vectors to exhaustively test a circuit with 25 inputs and 50 latches.
 - ◆ **No one does exhaustive testing!**
- Instead, research since the early 1960's has led to several test generation techniques and computer programs.
- Furthermore, designers may supply small sets of test vectors they know test certain important aspects of their designs.

Why Does a Chip Fail?

- All failures not due to design errors are caused by manufacturing defects:
 - ◆ Contamination (dust particles) causing metal bridges
 - ◆ Defects in silicon substrate
 - ◆ Photolithographic defects
 - ◆ Process variations and abnormalities
 - ◆ Oxide defects
- When a failure is described in terms of actual physical defects, it is referred to as a **physical failure**.

Fault Modeling

Modeling the effects of physical defects on the logic function and timing

- Physical Defects

- ♦ Silicon defects
- ♦ Photolithographic defects
- ♦ Mask contamination
- ♦ Process variations
- ♦ Defective oxide

- Logical Effects

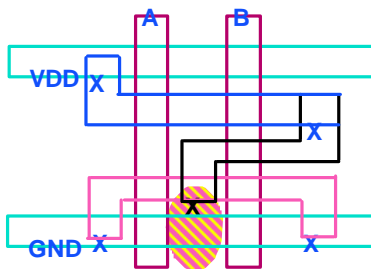
- ♦ Logic s-a-0 or 1
- ♦ Slower transition (delay faults)
- ♦ AND-bridging, OR-bridging

- Electrical Effects

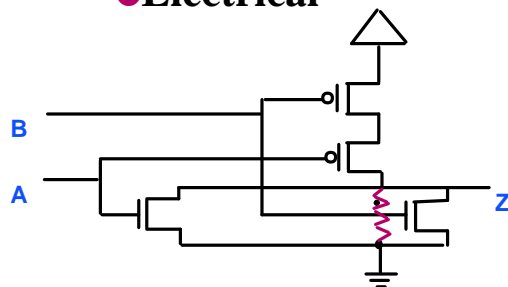
- ♦ Shorts (bridging faults)
- ♦ Opens
- ♦ Transistor stuck-on, stuck-open
- ♦ Resistive shorts and opens
- ♦ Change in threshold voltage

Fault Modeling

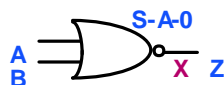
- Physical Defect



- Electrical



- Logical



Single Stuck-At Fault Model

- Why Use Single Stuck-At Fault (SSF) Model?
 - ◆ **Complexity is greatly reduced. Many different physical defects may be modeled by the same logical stuck-at fault.**
 - ◆ **SSF is technology independent**
 - ▲ Has been successfully used on TTL, ECL, CMOS, etc.
 - ◆ **Single stuck-at tests cover a large percentage of multiple stuck-at faults.**
 - ◆ **Single stuck-at tests cover a large percentage of unmodeled physical defects.**

Single Stuck-At Fault Model

- There are two possible faults for every node:
 - ◆ **s-a-0 and s-a-1**
- The difficult problem is to find tests that can be applied at the primary inputs of the circuit and will produce an error at the primary outputs if the fault is present.
- This problem is NP-hard; i.e., it may be impossible to find an efficient algorithm in the general case.
- The bulk of the research since the 1960's has focused on finding efficient approximate ways of solving this problem (heuristics).
- Furthermore, sequential circuits are considerably more difficult to test than combinational circuits.

Fault Coverage Measures

$$\text{Fault Coverage} = \frac{\text{Detected Faults}}{\text{Total Faults}}$$

$$\text{Testable Coverage} = \frac{\text{Detected Faults}}{\text{Total Faults} - \text{Untestable}}$$

$$\text{ATG Efficiency} = \frac{\text{Detected} + \text{Untestable Faults}}{\text{Total Faults}}$$

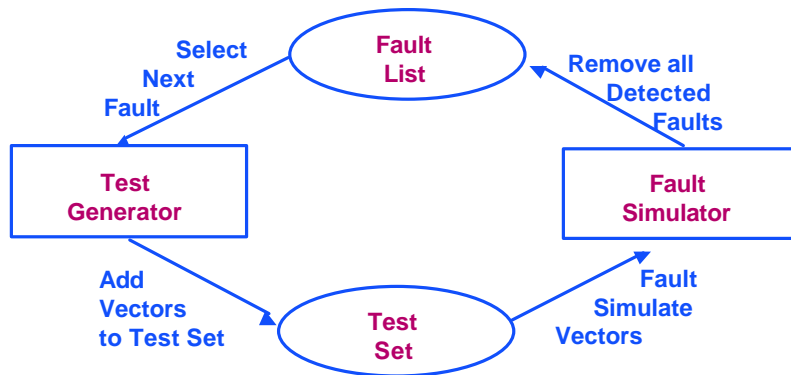
NOTE: 'Untestable' definition may be tool dependent!

Why Automatic Test Generation?

- Test generation can be the longest phase of the design cycle if done manually because virtually every other critical CAD problem is well automated (e.g., synthesis, placement, routing, and simulation).
- Functional verification vectors that the designer of the ASIC has provided, generally will not give high stuck-at-fault coverage. Functional vectors must be augmented by repeated use of a fault simulator.
- ASICs made with a synthesis tool are especially hard for manual test generation, because human insight is missing in a machine generated net list.

Test Generation System

- Since ATPG is much slower than fault simulation, the fault list is trimmed with use of a fault simulator after each vector is generated



Fault Diagnosis

- **Failure Analysis** is the task of determining the cause of a manufacturing defect.
- For chips that fail the go/no-go tests, failure analysis may be performed to characterize the defects.
 - ◆ **Fault diagnosis:** fault dictionaries or simulation-based techniques can be used to logically isolate the problem down to several candidate fault locations.
 - ◆ The chip may then be probed at internal points using an e-beam tester and detailed information collected on its behavior.
- Design or process changes can then be made to avoid similar failures in the future.

Design for Testability

- Generating tests for large digital circuits is very time consuming.
- One way around this problem is to **modify** the design in a way that makes test generation easier.
- If testing is not considered during the design phase, very low fault coverages can result in addition to high test generation times.
- The objective of **design for testability** (DFT) is to improve the **controllability** and **observability** of internal circuit nodes so that the circuit can be tested effectively.

Course Outline

PART I: Testing and ATPG

- **Logic Simulation:** 3-value, event driven simulation
- **Fault modeling:** stuck-at, delay, bridge, I_{ddq} , open. Fault dominance, fault equivalence, fundamentals of testing and method of Boolean difference
- **Parallel Fault Simulation:** serial, parallel, deductive
Inactive Fault Removal: critical path tracing, star algorithm
Statistical Fault Analysis: STEFAN
- **Intro to ATPG:**
 - ♦ D-algorithm (AND-OR decision trees, backtracking), PODEM, FAN, circuit learning, random TPG, test compaction
 - ♦ Sequential machine testing, bridge fault and memory testing

Course Outline

PART II: Design for Testability

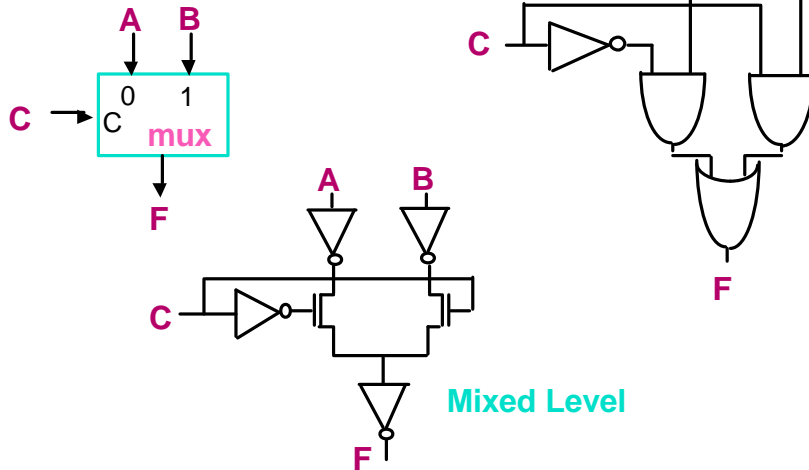
- **Design for Testability:** scan registers, scan chains, scan architectures and algorithms, system testing and partitioning
- **Build-in Self Test:** LFSRs and BIST architectures
- **Core Testing:** architecture and operation

Modeling of Circuits, Structures and Functions

- Logic diagrams, block diagrams, mixed switch and gate levels (graphical)
- Tables: Logic functions, state transitions
- Graphs
 - ◆ **Circuit structures, interconnects**
 - ◆ **logic functions (e.g., BDD's)**
- Symbolic expressions: Boolean, arithmetic
- Procedural Descriptions (text-based)
 - ◆ **RTL, C, VHDL, Verilog**

Modeling at Logic Level

Graphical:



ECE 1767

University of Toronto

Modeling at Logic Level

- Procedural
 - ◆ if C = 0 then F <-- A else F <-- B
 - ▲ if C = 0 then F <-- A
 - ▲ if C = 1 then F <-- B
 - ▲ if C = unknown then F <-- unknown
- Symbolic
 - ◆ $\underline{F} = \underline{A}\underline{C} + \underline{B}\underline{C}$
 - ◆ $F = AC + BC$

ECE 1767

University of Toronto

Modeling at Logic Level

Truth Table

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Singular Cover

A cover consisting of all prime implicants (1's) and prime implicants (0's) (prime implicants of F)

A	B	C	F
1	x	0	1
x	1	1	1

on-set
F is "on"

A	B	C	F
0	x	0	0
x	0	1	0

off-set
F is "off"

set of all **primitive cubes**
complete "cover"

Modeling at Logic Level

Singular cover of the multiplexer

	00	01	11	10
0	0	0	1	1
1	0	1	1	0

A	B	C	F
1	1	x	1
1	x	0	1
x	1	1	1
0	0	x	0
0	x	0	0
x	0	1	0

Prime Implicants
of F

Prime Implicants
of F

Compiled code for the multiplexer

A1 = A AND NOT C; delay 2 ns

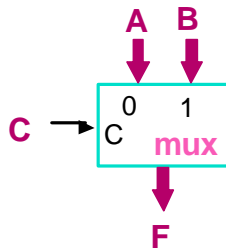
B1 = B AND C; delay 2 ns

F = A1 OR B1; delay 1 ns

Delays optional

RTL Models

- RTL models allow word level operations



0, 1 values

```
if C = 0
  then F <-- A
else
  F <-- B
```

Alternative:
0, 1, x values

```
if C = 0
  then F <-- A
else if C = 1
  then F <-- B
else
  F <-- x
```

Binary Decision Diagrams

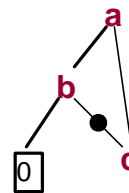
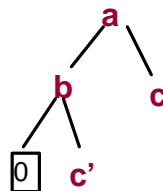
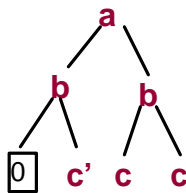
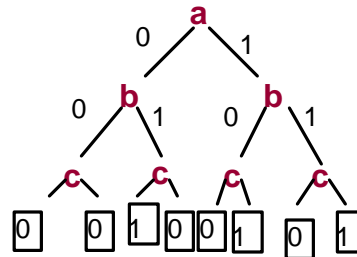
- A **BDD** is a graph-like model (data structure) of a function that has a canonical (ordering) property. It is based on Shannon's expansion:

$$f = x_i f_{x_i=1}(x_1, \dots, x_n) + \bar{x}_i f_{x_i=0}(x_1, \dots, x_n)$$

- To find value of function start at the top and go left if the variable is 0, right if it is 1. Complement result when you encounter a dot. Can be exponential in size (multipliers)
- BDD construction/compaction process

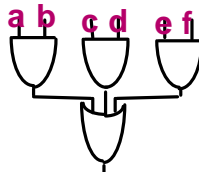
Binary Decision Diagrams: Construction/Compaction Process

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



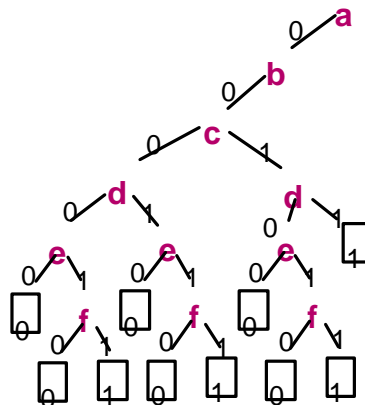
Variable Order Example

Order: a, b, c, d, e, f



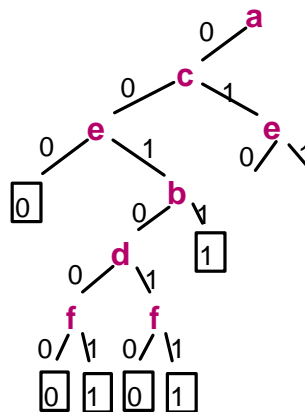
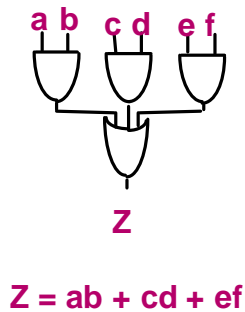
Z

$$Z = ab + cd + ef$$



Variable Order Example

Order: a, c, e, b, d, f



Variable Order Example

Dynamic Order

