A quick look at Circuit Testing

Alfredo Benso & Paolo PRINETTO
Politecnico di Torino (Italy)
www.testgroup.polito.it

Goals

– Presenting some of the major issues related to Digital Circuit Testing
Outline

- Introduction
- Definitions
- Testing a Circuit
- Historical Evolution
- Why testing
- The basic approach to test
- Testing diversifications
- Conclusions.

How can I discover faults in my systems?
A first look at System Testing

Slide # 2.5

Just test them !!!

Slide # 2.6

The result is:
• probabilistic
• valuable at the moment of the test, only

Test
Set of operations aiming at checking whether a manufactured unit works properly w.r.t. its specifications, or not.
Test

Determine if a system is mission-ready.
If not, help establishing why not.

Diagnostic test

Testability

Capability of a product to be tested, satisfying a set of given constraints in terms of quality, cost, time, ...
Testability

Capability of a design to guarantee that the final product will be testable, satisfying a set of given constraints in terms of quality, cost, time, ...

Design for Testability (DfT)

Modify the logic in a way to make it easily testable (e.g., Scan Design, BICS, Boundary Scan, ...)
Outline

- Introduction
- Definitions
- Testing a Circuit
- Historical Evolution
- Why testing
- The basic approach to test
- Testing diversifications
- Conclusions.

What is a fault?

- Impairments:
  - Faults
  - Errors
  - Failures
- Means
- Attributes & Measures
Caveat

• A plenty of different definitions & view points exists !!!

Fault

A physical defect, imperfection, or flaw that occurs within some hardware or software components.

[Pradham 96]
**Fault**

A malfunction in a hardware, software or human component of the system, which may introduce errors and may lead to failures.

Excited Faults

A fault is excited when a different behavior occurs, at the fault site, between the good and the faulty machine.
Be careful !!!

Even if excited, a fault doesn’t necessarily show up, since the fault site can be not observable from outside.

Error

A different behaviour, between good and faulty machine, due to the existence of one or more excited faults and observable from the outside.
Error

A manifestation of a fault.

Error

A deviation from accuracy or correctness.

[Pradham 96]
**Error**

An invalid state of the system, such as an incorrectly stored or transmitted items of data.

- [Gibb_76]

---

**Failure**

It occurs when an error results in the system performing one of its functions incorrectly.

- [Pradham 96]
A fault condition in the system or a functional unit that has influence on service.

[Ref: Gibb_76]

From Faults to Errors:

1. No fault
2. A fault occurs
   - The fault disappears
   - A ≠ appears at the fault site
3. A fault is present but latent
   - The ≠ disappears but the fault is still present
4. The fault is excited
   - The ≠ reaches an observability point
5. An error appears

[Gibb_76]
3-universe model

- If no tolerance for specific faults exists, a fault may lead to a failure:

![Diagram](https://example.com/diagram)

[Avizienis 82]

Error effects

- No fault
- A fault is present but latent
- A fault is excited
- An error appears
- System failure

[Avizienis 82]
**Error effects**

- No fault
- A fault is present but latent
- A fault is excited
- An error appears
  
  - System failure
  - Partial failure

**Partial Failure**

*Degrades service, but does not interrupt it completely.*

[Lan_86]
Graceful degradation

The ability of a system to automatically decrease its level of performance to compensate for hardware and/or software faults.

Error effects

- No fault
- A fault is present but latent
- A fault is excited
- An error appears
- System failure
- Partial failure
- Total failure
Total Failure

Interrupts the service until the system/functional unit is recovered or repaired or replaced

Error effects

- No fault
- A fault is present but latent
- A fault is excited
- An error appears

System failure

Partial failure  Total failure  Fail silent violation
Fail silent violation

The system/application produces incorrect results, while it looks providing correct ones.

Error effects

- No fault
- A fault is present but latent
- A fault is excited
- An error appears
- System failure
- The error is detected
Error effects (cont’d)

– it can be detected:
  . by one of the system Error Detection Mechanism (e.g., hardware exceptions handling, software checks, ...).
  The system reaches a safe state.
  . during a test session.

Error effects

- No fault
- A fault is present but latent
- A fault is excited
- An error appears
  - System failure
  - The error is detected
  - The error has no effect
**Error effects** (cont’ed)

– It can *have no effect*, since:
  - it has been overwritten and thus disappeared
  - it represents a *potential Hazard*: it is active and could eventually have effects
  - it is not significant from the system behavior point of view.

**Error effects diagram**

- No fault
- A fault is present but latent
- A fault is excited
- An error appears
- System failure
- The error is detected
- The error has no effect
- The error is latent
- The ≠ disappears but the defect is still present
- The ≠ and the defect disappear
- The ≠ is no longer on an observability point
**Fault Latency**

The length of time between the occurrence of a fault and the appearance of an error due to that fault.

**Error Latency**

The length of time between the occurrence of an error and the appearance of the resulting failure.
Fault characteristics

- Cause
- Nature
- Duration
- Extent
- Value

Fault characteristics

- **Cause** (that which leads to the fault)
  - Specification Mistakes
  - Implementation Mistakes
  - External Disturbances
  - Physical Hardware Component Defects
  - Misuses
- Nature
- Duration
- Extent
- Value
Fault characteristics

- **Cause**
- **Nature** (relates to the intent of the cause of fault):
  - *Hardware*
  - *Software*
- **Duration**
- **Extent**
- **Value**

Fault characteristics

- **Cause**
- **Nature**
- **Duration** (length of time for which the fault persists)
  - *Permanent*
  - *Intermittent*
  - *Transient*
- **Extent**
- **Value**
Fault characteristics

- **Cause**
- **Nature**
- **Duration** (length of time for which the fault persists)
  - *Permanent*
  - *Intermittent*
  - *Transient*
- **Extent**
- **Value**

It remains in existence indefinitely if no corrective action is taken

Fault characteristics

- **Cause**
- **Nature**
- **Duration** (length of time for which the fault persists)
  - *Permanent*
  - *Intermittent*
  - *Transient*
- **Extent**
- **Value**

It appears, disappears, and then reappears repeatedly
Fault characteristics

- **Cause**
- **Nature**
- **Duration** (length of time for which the fault persists)
  - Permanent
  - Intermittent
  - Transient
- **Extent**
- **Value**

It can appear and disappear within a given period of time

Fault characteristics

- **Cause**
- **Nature**
- **Duration**
- **Extent** (how far a fault propagates)
  - Local
  - Global
- **Value**
Fault characteristics

- Cause
- Nature
- Duration
- Extent
- Value (consequence of the fault)
  - Determinate
  - Indeterminate

E.g., the so called “Byzantine faults”
**Byzantine faults**

- It occurs when the system can fail & stop, or execute slowly, or execute at normal speed but produce erroneous values.

---

**Genesis**

- Specification mistakes
- Implementation mistakes
- External Disturbances
- Component Defects

**Errors**

- Software Faults
- Hardware Faults

**System Failures**
Misuses

- Operator error is the most common cause of failure
- Nevertheless many errors attributed to operators are actually caused by designs that require an operator to choose an appropriate recovery action without much guidance and without any automated help
Faults can occur in any moment of the product life cycle.

Defect

The lack of something necessary or desirable for completion or perfection; deficiency.
Physical Defects

Defects that can occur in the physical structure of a system.

Physical Defects are obviously not numerable in terms of type, location, time of occurrence.
Hardware Fault model

A model of a defect to make it numerable, and thus tractable, making restricting hypothesis on location and types.

Some Hardware Fault Models

- single (multiple) stuck-at
- short (bridge)
- stuck-on & stuck-open
- slow-to-rise & slow-to-fall
- single state-transition fault
- coupling
- pattern sensitive faults
- functional
- ...

© P.Prinetto 2003 - all rights reserved
Stuck-at Faults

– Any given node in the network permanently stuck at 0 or 1.

Selection conditioners

The fault model selection is heavily influenced by:

– target system:
  . technology
  . type (IP core, chip, board, system;
    random logic, memory, microprocessor;
    ...)

– available support tools (Automated Synthesis, Fault Simulation, ATPG, ...)

– available test equipments (ATE, BIST)

– ...
Status

The single stuck-at fault model, although introduced in 1958, is still a de-facto standard, since:
- it’s a metric
- it’s technology independent
- it’s easy to use
- it’s managed by any CAD tool
- ...

Outline

- Introduction
- Definitions
- Testing a Circuit
- Historical Evolution
- Why testing
- The basic approach to test
- Testing diversifications
- Conclusions.
How many Stuck at Faults?

- The total number of faults is $2N$, where $N$ is the number of gate terminals

\[ \times = \text{fault site} \]

Equivalent Faults

- Let $F_1$ and $F_2$ be the functions performed by $C$ in the presence of $f_1$ and $f_2$, respectively. Then faults $f_1$ and $f_2$ are equivalent if and only if $F_1 = F_2$

- Fault collapsing
- Generate only one test for a group of equivalent faults
Testing of a Circuit

\[ \begin{align*}
a &= 1 \\
b &= 0 \\
c &= 0 \\
d &= \text{G1} \\
e &= \text{G2} \\
s-a-1 &= c
\end{align*} \]

Controlling and Non-controlling Value

- **Controlling value**: when it present on at least one input of a gate, it forces the output to a known value
  - AND gate, NAND gate: 0
  - OR gate, NOR gate: 1

- **Non-controlling value**: the complement of (Sensitizing value) the controlling value
  - AND gate, NAND gate: 1
  - OR gate, NOR gate: 0
Automatic Test Generation

• Three steps
  – Set up (fault sensitizing)
  – Propagation (path sensitizing)
  – Justification (consistency check)

Test Generation (D-Algorithm)

• The setup step is to produce a difference in the output signal at the gate where the fault is located between the two cases when the fault is present or it is absent.

\[
D = \begin{cases} 
0 & \text{when fault occurs} \\
1 & \text{no fault}
\end{cases}
\]

• D is called frontier
Test Generation (D-Algorithm)

• The propagation step derives the D (or $\overline{D}$) condition from the faulty gate to a output

\[ D_1 \]

\[ D_0 \]

\[ \overline{D} \]

Test Generation (D-Algorithm)

• The last step is to force the logic values needed to sensitize the assumed fault from the primary inputs
Backtracking

1. excitation condition \( a = b = 1 \)
2. sensitization condition \( f = 0 \)
3. choose \( d = 1 \Rightarrow b = 0 \) (conflict)
   - try \( c = 1 \) (succeed)

*Backtracking: returning on one’s step and reversing a previous choice*

Untestable Fault

1. excitation condition \( b = 0 \)
2. sensitization condition \( c = 0 \)
3. justification \( a = 1 \) and \( b = 1 \) (conflict)

*There is no test for b s-a-1 fault*
*\( b \) is redundant*
*Replacing \( b \) by 1 \( \Rightarrow d = 0 \)*
*The conflicting requirement derived from reconvergent fanout (paths have a common source and a common sink)*
Outline

- Introduction
- Definitions
- Testing a Circuit
- Historical Evolution
- Why testing
- The basic approach to test
- Testing diversifications
- Conclusions.

First Design, then Test

Historical Evolution
We cannot adopt Design for Testability techniques, since our system will be bigger and slower. The resulting overhead is unacceptable for us!!
If you consider testability as part of the specifications, its cost cannot be considered an overhead.

First Design, then Test

Historical Evolution

Design & Test
First Design, then Test

Design & Test

Testable Synthesis & Built-In Self Testing

Concurrent Engineering

Historical Evolution
Designers ARE Test Engineers

Test responsibilities must be owned by every member of the design team!!
Testable Synthesis

Automated synthesis methodology that considers testability as a design constraint.

Test Synthesis

Automatic Testability enhancement
Outline

- Introduction
- Definitions
- Testing a Circuit
- Historical Evolution
- Why testing
- The basic approach to test
- Testing diversifications
- Conclusions.

Why should I test?
“If anything can go wrong ... 
... it will!”

[Murphy]

“All customers are named Murphy!”

[Any test engineer]
Sooner or later nearly every computer company suffers a glitch.

Intel

- In 1994 Intel relented under pressure and replaced flawed Pentium processors.
- The chips contained an error that could have caused them to make mathematical mistakes.
- Cost:

\[ 480 \text{ M US$} \]

\[ \equiv \text{the overall EU Esprit budget for '94 !!!} \]
Apple

– In 1995, Apple Computer recalled Powerbook 5300 laptops after some units burst into flames.
– A lithium ion battery was overheating.
– Only 1,000 units had been shipped.

Toshiba

– In 1999 Toshiba settled a lawsuit alleging that the company sold 5 million defective laptops.
– The culprit was a semiconductor for controlling floppy drives.
– Cost: $2.1 G US$
Intel

– May 12, 2000
– Intel this week announced it has found defects in a chip called the memory translator hub (MTH) used to route signals from Intel’s 820 chipset to the SDRAM in Pentium IIIs.
– Intel said noise in the MTH could cause PCs to reset, reboot, or freeze, and in some cases, cause data corruption. The company has not fixed the problem, but has offered to replace all defective motherboards with new ones fitted with Rambus memory, a faster memory than SDRAM.
– An Intel spokesman said about one million motherboards have been shipped to end customers since November 1999 and could cost the company a few hundred million dollars. Analysts expect the recall to cost Intel $300 million and $400 million, placing it on a par with the infamous Pentium recall.

An Italian Car manufacturer

– 5,000 68HC11 processors, not burned-in, with bonding problems
– Mounted on cars
– 2,000 cars sold before discovering the problem
– 2,000 systems to replace:
  . 1 year to find all of them
  . global cost of 700 ML !!!
The “Rule of 10”

According to the EU law, if a product damage you, you can prosecute both the final manufacturer and the manufacturers of all the sub-components.
TEST to:
• Improve quality
• Improve yields
• Reduce TTM, TTV, TT$
• …

Time to Market (TTM)

Time to Volume (TTV)

Time to Money (TT$)

Profits

Costs

Revenues
Outline

– Introduction
– Definitions
– Testing a Circuit
– Historical Evolution
– Why testing
– The basic approach to test
– Testing diversifications
– Conclusions.

Which is the basic approach to testing?
The basic approach to Testing

A proper sequence of values

Reference System

Target System Under Test

Comparator

Good / Faulty indication

Test Sequences

Fault free

Faulty

Fault free

Faulty

0
Basic steps

Step #1

Off-line, generate the Test Sequences to be applied to the UUT inputs.

Tools

- ATPG (Automatic Test Pattern Generator)
- Ad hoc Software
- Ad hoc Hardware
- Hand, brain, experience,
- ...

© P.Prinetto 2003 - all rights reserved

Vers. 1.0

2.53
Basic steps

Step #1
Off-line, generate the test sequences to be applied to the UUT inputs.

Step #2
Off-line, determine the behavior of the reference (i.e., non faulty) unit when Test Sequences are applied.

Practical approach

Step #1
Off-line, determine the behavior of the reference (i.e., non faulty) unit when Test Sequences are applied.

Test sequences
UUT description
Reference output behavior
Basic steps

Step #1
Off-line, generate the test sequences to be applied to the UUT inputs.

Step #2
Off-line, determine the behavior of the reference (i.e., non faulty) unit when Test Sequences are applied.

Step #3
At test time:
- apply Test Sequences to UUT inputs
- compare UUT outputs with the reference output behavior

Step #4
When a faulty unit is found:
- UUT is repaired (if possible)
- Statistics are collected to fix the production and/or the test process.
How can we evaluate how good our tests really are?

Carefully evaluate your Defect Levels
Defect level

The Defect Level depends on:
- the quality of the production process (yield)
- the quality of the test (coverage).
Yield

The % of elements of a given set that are considered to be defect free.

Be careful !!!

For TTM and TT$ reasons, in any new IC technology, the production starts when just 10% of the chips provided by the new process line works properly.
Coverage

Given:
• a circuit
• a set of possible faults
• a test sequence

Fault Coverage = \frac{\text{# detected faults}}{\text{# possible faults}}

Defect Coverage = \frac{\text{# detected defects}}{\text{# possible defects}}

Detected (or covered) fault

A test sequence detects (covers) a fault if the values it generates on the UUT outputs when the fault is present are different from the values generated when the UUT is fault free.
Detected Faults

- A Test Pattern detects a single fault iff:
  - the fault is excited (opposite values on the fault site)
  - the difference is propagated to at least one output.

Example of Defect Level evaluation

\[ DL = 1 - Y^{(1-T)} \]

where:
- \( Y \) = process yield
- \( T \) = coverage


Too approximated when the coverage is estimated in terms of a single fault model, only.
Outline

- Introduction
- Definitions
- Testing a Circuit
- Historical Evolution
- Why testing
- The basic approach to test
- Testing diversifications
- Conclusions.
Is there just one kind of test or are there alternatives to exploit?

There is no “one-size fits-all” solution!
Testing Diversification (1)

Product Life Cycle

Levels of Integration

Technology

In-Field
End-of-Production
Design
Core
Chip
MCM
Board
System

Levels of Integration:
- Core
- Chip
- MCM
- Board
- System

Testing Diversification (2)

When

On-line Testing
Off-line Testing
Built-In Self Test
External Testing

How

Structural Testing
Functional Testing
Parametric Testing
Current-based Testing
• Let’s now focus on some of the above mentioned attributes
EOP test goals

- Minimize the *Defect Level*, by identifying faulty units
- Diagnose fault type and location
- Improve:
  - the production process
  - the test process.
EOP test goals

Defect level

Test station

Fix the production process

F, ~OK

Fix the product

F, OK

F, ~OK

Fault Analysis & Repair

Fix the test process

PASS

P, OK

P, ~OK

FAIL

F, OK

F, ~OK

EOP test goals

Test station

Fix the production process

F, ~OK

Fix the product

F, OK

F, ~OK

Fault Analysis & Repair

Fix the test process

PASS

P, OK

P, ~OK

FAIL

F, OK

F, ~OK

Testing Diversification (1)

Product Life Cycle

Levels of Integration

Technology

In-Field

End-of-Production

Design

Core, Chip, MCM, Board, System

Analog, Memory, Processor, Random Logic

© P.Prinetto 2003 - all rights reserved

Vers. 1.0

2.66
In-field test goals

- Detect Physical Defects, Fault and Errors as soon as possible
- Diagnose fault type and location
- Identify faulty SRUs (Smallest Replaceable Units).

Testing Diversification (2)
When

On-line Testing
Off-line Testing

How

Built-In Self Test
External Testing

What

Structural Testing
Functional Testing
Parametric Testing
Current-based Testing

Testing Diversification (2)

Test performed when the system is idle.

Slide # 2.137
© Prinetto - 2003

Slide # 2.138
© Prinetto - 2003
On-line Test

Test performed while the system is normally working

- Concurrent, if it guarantees a zero error latency
- Not concurrent otherwise.

Testing Diversification (2)

When

- On-line Testing
- Off-line Testing

How

- Built-In Self Test
- External Testing

What

- Structural Testing
- Functional Testing
- Parametric Testing
- Current-based Testing

Slide # 2.139

Slide # 2.140
**Structural test**

It looks for faults that can occur in the physical structure of the UUT.

---

**Pro’s & Con’s**

+ Tools available for automatic generation
+ Precise Coverage evaluation
+ Diagnostic capabilities
+ Design independence
- Applicable if the netlist is known, only.
Status

- One of standard EOP test performed by any IC manufacturer.

Functional test

It aims at checking the correct behavior of the target system, w.r.t. its specs, but regardless its actual implementation.
An uncompleted example

- Reset
  - Shift four 1 and check that $\text{SHFT}_{\text{OUT}}$ be 0
  - Shift four 0 and check that $\text{SHFT}_{\text{OUT}}$ be 1

Pro’s & Con’s

+ Rather Easy to write
+ No structural knowledge required
+ Allow at-speed test
  - Very hard coverage evaluation
  - Specs dependence
  - Manual generation, only.
Status

– Usual approach to test memories and microprocessors, at the user level.
Structural Test verifies the correct implementation at each level.

Parametric test

It aims at measuring the correctness of some electrical entities of the target system.
Status

– One of the standard approaches for both products and manufacturers qualification.

At-speed Test
Test performed at the target nominal frequency.
Status

- 100% of microprocessors and memories are tested, at EOP, at-speed, too.

Experimental results

Data collected on 4,349 faulty devices of a sample of 26,415 dies.

[Maxwell et al, ITC’92]
Testing Diversification (2)

When

On-line Testing
Off-line Testing
Built-In Self Test
External Testing

How

Structural Testing
Functional Testing
Parametric Testing
Current-based Testing

What

External Testing

Test is performed resorting to an Automatic Test Equipment (ATE)
The chip to be tested is inserted here!!
**ATE cost**

Test-per-pin architecture: 5 K US $ / pin

From 3 to 6 M US $ !!
A first look at System Testing

Self Healing in Dependable Systems

ATE for boards

Bad-of-Nails

Some “Nails”
Testing Diversification (2)

- **When**
  - On-line Testing
  - Off-line Testing
  - Built-In Self Test
  - External Testing

- **How**
  - Structural Testing
  - Functional Testing
  - Parametric Testing
  - Current-based Testing

- **What**
  - Structural Testing
  - Functional Testing
  - Parametric Testing
  - Current-based Testing

*Built-In Self Test (BIST)*

Modify the logic in a way to make it test itself.
Migrate on board most of the ATE capabilities !!

Embedded ATE or BIST (Built-In Self Test)

Embedded ATE

Is a total test solution embedded in silicon for test, diagnostic and measurement, at all the levels, from cores to systems, from EOP test to in-field maintenance.
LogicVision’s Solution

- External ATE
  - Standard Digital Tester
  - Limited Speed/Accuracy
  - Low Cost-per-Pin

- Embedded ATE
  - Pattern Generation
  - Result Compression
  - Power Management
  - Test Control
  - Support for Board-level Test
  - System-Level Test
  - (about 10k gates)

- Memory
  - SRAM, DRAM, RDM, Flash, FIFO, CAM, etc.

- Logic
  - Processor, I/O, Audio, Video, Glue Logic, etc.

- Mixed-Signal
  - PLL, ADC/DAC, Filter, Power Supplies, etc.

- I/Os & Interconnects
  - Drivers/Receivers, Boundary Scan, etc.

Very Deep Submicron Chip, SOC, Board or System

Reduced Pin-Count, Low Bandwidth External Interface
High-Bandwidth Internal Interfaces

Slide # 2.167
© Prinetto - 2003

BIST costs

- External ATE
  - Standard Digital Tester
  - Limited Speed/Accuracy
  - Low Cost-per-Pin

- Embedded ATE
  - Pattern Generation
  - Result Compression
  - Power Management
  - Test Control
  - Support for Board-level Test
  - System-Level Test
  - (about 10k gates)

- Memory
  - SRAM, DRAM, RDM, Flash, FIFO, CAM, etc.

- Logic
  - Processor, I/O, Audio, Video, Glue Logic, etc.

- Mixed-Signal
  - PLL, ADC/DAC, Filter, Power Supplies, etc.

- I/Os & Interconnects
  - Drivers/Receivers, Boundary Scan, etc.

Very Deep Submicron Chip, SOC, Board or System

10 K gate equivalent

Slide # 2.168
© Prinetto - 2003
BIST usage

BIST structures are today widely used:
- in a big variety of “off-the-shelf” products:
  - Microprocessors (Intel, Motorola, Toshiba, Sun, …)
  - Personal computers
  - Workstations
  - Sawing machines
  - Automotive applications
  - …

BIST usage (cont’d)

- in almost all embedded memory IP cores
- In a lot of SoC’s
- …
Is it enough to test just at the end of the overall production process?

NO !!!
Remember the “Rule of 10”. Test sub-modules asap.
The “Rule of 10”

Cost for replacing a faulty device

Device | Board | System | In-field

Abstraction level

An example of IC production test

IC processing → Water probe Test → IC packaging → Packaged IC Test

Burn-In → Post burn-in Test → Ship
A first look at System Testing

An example of board production test

Outline

- Introduction
- Definitions
- Testing a Circuit
- Historical Evolution
- Why testing
- The basic approach to test
- Testing diversifications
- Conclusions.
BE CAREFUL:
If you don’t need testing any longer, your microelectronic technology is not aggressive enough, and, quite soon, you’ll be out of the market!!!

References