Overview

- VLSI realization process
- Role of testing, related cost
- Basic Digital VLSI test concepts
- Fault Modeling, Test Generation
- Design for Testability (SCAN, BIST)
Electronic Systems Design/Fabrication Cycle

Customer Needs
- Determine Requirements
- Write Specifications
- Design & Test Development
- Fabrication
- Manufacturing Test

Good Chips to Customer

Needs to be satisfied by the chip, i.e., function of the application
Testing and Design for Testability

VLSI Realization Process

Customer Needs

Determine Requirements

Write Specifications

Design & Test Development

Fabrication

Manufacturing Test

Good Chips to Customer

Determine Chip characteristics:

- Function (I/O),
- Operating (power, frequency, noise, etc),
- Physical (packaging, etc),
- Environmental (temperature, reliability, etc),
- Other (volume, cost, price, availability, etc).

VLSI Realization Process

Customer Needs

Determine Requirements

Write Specifications

Design & Test Development

Fabrication

Manufacturing Test

Good Chips to Customer

Determine Chip characteristics:

- Function (I/O),
- Operating (power, frequency, noise, etc),
- Physical (packaging, etc),
- Environmental (temperature, reliability, etc),
- Other (volume, cost, price, availability, etc).

System/Macro-level model

LogiGate-level model

Transistor/Device-level model
Abstraction Models

System/Macro Level Model, Ex. Data Stack

Logic/Gate Level Model, Ex. 4-bit Register

Transistor Level Model

MOS Device Level

VLSI Realization Process (cont.)

Customer Needs

- Determine Requirements
- Write Specifications
- Design & Test Development
- Fabrication
- Manufacturing Test

Architectural Design/Test
- System/Macro-level model
- Logic/Gate-level model
- Transistor/Device-level model

Physical Design/Test

Apply Test, detect/locate fabrication defects

Good Chips to Customer
Definitions

- **Design synthesis**: Given an I/O function, develop a procedure to manufacture a device using known materials and processes.

- **Verification**: Predictive analysis to ensure that the synthesized design, when manufactured, will perform the given I/O function.

- **Test**: A manufacturing step that ensures that the physical device, manufactured from the synthesized design, has no manufacturing defect.

Verification vs. Test

- **Verifies correctness of design.**
- **Performed by simulation, hardware emulation, or formal methods.**
- **Performed once prior to manufacturing.**
- **Responsible for quality of design.**

- **Verifies correctness of manufactured hardware.**
- **Two-part process:**
  1. **Test generation**: software process executed once during design
  2. **Test application**: electrical tests applied to hardware
- **Test application performed on every manufactured device.**
- **Responsible for quality of devices.**
Note on Reliability

- Testing is related to Reliability, but often confused…
  - A chip that passes testing is certainly more reliable than the one that has not (if used)
  - However, a chip is not necessarily reliable because it has passed testing!
  - On-line testing contributes to reliability

- Reliability is currently receiving wider attention
  - Necessary for non-critical systems due to scaling and larger integration (difficulty w/ testing, more transient/ware-out faults, by-passing of permanent faults, …)

Digital VLSI test concept

- Basic scheme for testing internal components

```
Fault F

Circuit Under Test (CUT)

Test patterns T

Test response R

Response Comparison

Expected (good) response R'

Test Generation

Test Equipment

Test Outcome
Pass: R=R'
Fail: R≠R'
```
Roles of Testing

- **Detection**: Determination whether or not the device under test (DUT) has some fault.
- **Diagnosis**: Identification of a specific fault that is present on DUT.
- **Device characterization**: Determination and correction of errors in design and/or test procedure.
- **Failure mode analysis (FMA)**: Determination of manufacturing process errors that may have caused defects on the DUT.

Costs of Testing

- **Design for testability (DFT)**
  - Chip area overhead and yield reduction
  - Performance overhead
- **Software processes of test**
  - Test generation and fault simulation
  - Test programming and debugging
- **Manufacturing test**
  - *Automatic Test Equipment* (ATE) capital cost
  - Test center operational cost
Design for Testability (DFT)

DFT refers to hardware design styles or added hardware that reduces test generation complexity.

Motivation: Test generation complexity increases exponentially with the size of the circuit.

Example: Test hardware applies tests to blocks A and B and to internal bus; avoids test generation for combined A and B blocks.

Cost of Manufacturing Testing in 2000AD

- 0.5-1.0GHz, analog instruments, 1,024 digital pins: ATE purchase price
  - = $1.2M + 1,024 x $3,000 = $4.272M
- Running cost (five-year linear depreciation)
  - = Depreciation + Maintenance + Operation
  - = $0.854M + $0.085M + $0.5M
  - = $1.439M/year
- Test cost (24 hour ATE operation)
  - = $1.439M/(365 x 24 x 3,600)
  - = 4.5 cents/second
Automatic Test Equipment (ATE)

ATE consists of:
- Powerful computer
- Powerful 32-bit Digital Signal Processor (DSP) for analog testing
- Test Program (written in high-level language) running on the computer
- Probe Head (actually touches the bare or packaged chip to perform fault detection experiments)
- Probe Card or Membrane Probe (contains electronics to measure signals on chip pin or pad)

LTX FUSION HF ATE
Economics for Design for Testability

- DFT can reduce cost of testing
- Consider life-cycle cost; DFT on chip may impact the costs at board and system levels.
- Can lead to performance degradation
- Consider costs vs benefits
  - Cost examples: reduced yield due to area overhead, yield loss due to non-functional tests
  - Benefit examples: Reduced ATE cost due to self-test, inexpensive alternatives to burn-in test

Benefits and Costs of DFT

**BIST Example**

<table>
<thead>
<tr>
<th>Level</th>
<th>Design and test</th>
<th>Fabrication</th>
<th>Manuf. Test</th>
<th>Maintenance test</th>
<th>Diagnosis and repair</th>
<th>Service interruption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chips</td>
<td>+ / -</td>
<td>+</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boards</td>
<td>+ / -</td>
<td>+</td>
<td>-</td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td>+ / -</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

+ Cost increase - Cost saving +/- Cost increase may balance cost reduction
Fault Modeling

- Bridges gap between physical reality and mathematical abstraction
- Allows application of analytical tools
- Thus, essential in design

Ideal Tests

- Ideal tests detect all defects produced in the manufacturing process.
- Ideal tests pass all functionally good devices.
- Very large numbers and varieties of possible defects need to be tested.
- Difficult to generate tests for some real defects. *Defect-oriented testing is an open problem.*
Real Tests

- Based on analyzable fault models, which may not map on real defects.
- Incomplete coverage of modeled faults due to high complexity.
- Some good chips are rejected. The fraction (or percentage) of such chips is called the yield loss.
- Some bad chips pass tests. The fraction (or percentage) of bad chips among all passing chips is called the defect level.

Defects, Errors, and Faults

- Defect: Unintended difference between manufactured h/w and design
- Error: A wrong output signal produced by a defective system (observable)
- Fault: Representation of a defect at an abstracted level
Some real defects in chips

- Processing defects
  - Missing contact windows
  - Parasitic transistors
  - Oxide breakdown
  - ...
- Material defects
  - Bulk defects (cracks, crystal imperfections)
  - Surface impurities (ion migration)
  - ...
- Time-dependent failures
  - Dielectric breakdown
  - Electromigration
  - ...
- Packaging failures
  - Contact degradation
  - Seal leaks
  - ...


Levels of Fault Models

- Related to the level of circuit model
  - Behavioral/High/Functional Level
  - Logic Level
    - Logic faults, ex. stuck-at, bridging
    - Delay faults
  - Transistor Level
    - Technology dependent
- “Realistic” fault models (ex. $I_{DDQ}$)
Common Fault Models

- Bridging faults
- **Single stuck-at faults**
- Transistor open and short faults
- Memory faults
- PLD faults (stuck-at, cross-point, bridging)
- Functional faults (processors)
- Delay faults (transition, path)
- IDDQ faults
- ...

Single stuck-at fault model

- Three properties define a single stuck-at fault
  - Only one line is faulty
  - The faulty line is permanently set to 0 or 1
  - The fault can be at an input or output of a gate
- Example: XOR circuit has 12 fault sites (●) and 24 single stuck-at faults
Single stuck-at fault model

- Consider the **stuck-at-0** fault at line \( h \) \( (h \ s-a-0) \)
- A test is an input combination s.t. the value at output \( z \) when there is no fault (good cct) is different from the value at output \( z \) where line \( h \) is **s-a-0** (faulty cct).
- A test must:
  - Activate the fault (bring a value 1 at \( h \))
  - Propagate its effect at some primary output

---

Faulty circuit value

Good circuit value

---

Single stuck-at fault model

- Consider the **stuck-at-0** fault at line \( h \) \( (h \ s-a-0) \)
- A test is an input combination s.t. the value at output \( z \) when there is no fault (good cct) is different from the value at output \( z \) where line \( h \) is **s-a-0** (faulty cct).
- A test must:
  - Activate the fault (bring a value 1 at \( h \))
  - Propagate its effect at some primary output
Testing and Design for Testability

**Single stuck-at fault model**

- Consider the **stuck-at-0** fault at line \( h \) (\( h \) s-a-0)
- A test is an input combination s.t. the value at output \( z \) when there is no fault (good cct) is different from the value at output \( z \) where line \( h \) is s-a-0 (faulty cct).
- A test must:
  - Activate the fault (bring a value 1 at \( h \))
  - Propagate its effect at some primary output

![Test vector for \( h \) s-a-0 fault](image)

**Fault Equivalence**

- Number of fault sites in a Boolean gate circuit
  \[ = \#PI + \#gates + \# (fanout branches). \]
- Fault equivalence: Two faults \( f_1 \) and \( f_2 \) are equivalent if all tests that detect \( f_1 \) also detect \( f_2 \), and vice-versa.
- If faults \( f_1 \) and \( f_2 \) are equivalent then the corresponding faulty functions are identical.
- Fault collapsing: All single faults of a logic circuit can be divided into disjoint equivalence subsets, where all faults in a subset are mutually equivalent. A **collapsed fault set contains one fault from each equivalence subset**.
Equivalence Rules

- **AND**
- **OR**
- **NAND**
- **NOR**
- **WIRE/BUFFER**
- **INVERTER**
- **NOT**
- **FANOUT**

Faults in red removed by equivalence collapsing

12 faults collapsed

Collapse ratio = \( \frac{20}{32} = 0.625 \)

Equivalence Example

(MKM - 33)

(MKM - 34)
Fault Dominance

- If all tests of some fault F1 detect another fault F2, then F2 is said to dominate F1.
- Dominance fault collapsing: If fault F2 dominates F1, then F2 is removed from the fault list.
- When dominance fault collapsing is used, it is sufficient to consider only the input faults of Boolean gates. See the next example.
- In a tree circuit (without fanouts) PI faults form a dominance collapsed fault set.
- If two faults dominate each other then they are equivalent.

Dominance Example

A dominance collapsed fault set (after equivalence collapsing)
Primary inputs and fanout branches of a combinational circuit are called **checkpoints**.

**Checkpoint theorem:** A test set that detects all single (multiple) stuck-at faults on all checkpoints of a combinational circuit, also detects all single (multiple) stuck-at faults in that circuit.

**Redundant/Untestable Faults**

Some single stuck-at faults are identified by fault simulators or test generators as:

- **Redundant fault** \( \rightarrow \) No test exists for the fault.
- **Untestable fault** \( \rightarrow \) Test generator is unable to find a test.
Multiple Stuck-at Faults

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

Automatic Test Pattern Generation (ATPG)

- The process of generating patterns to test a circuit.
- Basic steps involved:
  - Fault activation (injection)
  - Fault propagation
- ATPG Algebra (for stuck-at fault model)
  - 5-value composite algebra

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning (Fault-free/Fault Value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0/0</td>
</tr>
<tr>
<td>1</td>
<td>1/1</td>
</tr>
<tr>
<td>X</td>
<td>X/X</td>
</tr>
<tr>
<td>D</td>
<td>1/0</td>
</tr>
</tbody>
</table>

\[
 F = A \cdot B \\
 sa0 \\
 D
\]
Automatic Test Pattern Generation (ATPG)

- The process of generating patterns to test a circuit.
- Basic steps involved:
  - Fault activation (injection)
  - Fault propagation
- Propagation of error value (D or D)

![Diagram of ATPG process]
Automatic Test Pattern Generation (ATPG)

• The process of generating patterns to test a circuit.
• Basic steps involved:
  – Fault activation (injection)
  – Fault propagation
• Propagation of error value (D or \( \bar{D} \))

![Diagram of a circuit with inputs A, B, C, E and outputs h, i, j, k, l.]

Automatic Test Pattern Generation (ATPG)

• The process of generating patterns to test a circuit.
• Basic steps involved:
  – Fault activation (injection)
  – Fault propagation
• Propagation of error value (D or \( \bar{D} \))
• Structural Vs Symbolic ATPG techniques
  – Structural: Fast for easy to test faults
    - Identify one or more tests
  – Symbolic: Identify complete set of tests per fault
    - Depends of boolean function representation
Design For Testability (DFT)

- Design for testability (DFT) refers to those design techniques that make test generation and test application cost-effective.
- DFT methods for VLSI circuits (digital/memory/mixed):
  - Ad-hoc methods
  - Structured methods:
    • Scan for Digital Logic
    • Partial Scan
    • Built-in self-test (BIST) for Memory
    • Boundary scan for access to embedded components
    • Analog test bus
    • Systems (SoCs) test

Ad-Hoc DFT Methods

- Good design practices learnt through experience are used as guidelines:
  - Avoid asynchronous (unclocked) feedback.
  - Make flip-flops initializable.
  - Avoid redundant gates. Avoid large fanin gates.
  - Provide test control for difficult-to-control signals.
  - Avoid gated clocks.
  - Consider ATE requirements (tristates, etc.)
**Ad-Hoc DFT Methods**

- Design reviews conducted by experts or design auditing tools.
  - Modify Circuit
  - Insert test points

- Disadvantages of ad-hoc DFT methods:
  - Experts and tools not always available.
  - Test generation is often manual with no guarantee of high fault coverage.
  - Circuits have become too large for manual inspection
  - Design iterations may be necessary.

**Structured DFT Methods**

- Alternative to Ad-Hoc methods:
  - Extra logic and signals added to facilitate testing according to some predefined procedure.
  - Divided into Scan and Built-In-Self-Test (BIST)
  - Allow for Automatic Test Pattern Generation (ATPG)
  - Larger circuits can be handled
Scan Design - Full/Partial

- Obtain control and observability of all/some flip-flops
  - Test structure (hardware) is added to the verified design:
    - Add a test control (TC) primary input.
    - Replace flip-flops by scan flip-flops (SFF) and connect to form one or more shift registers in the test mode.
    - Make input/output of each scan shift register controllable/observable from PI/PO.
  - Use combinational ATPG to obtain tests for all testable faults in the combinational logic.
  - Add shift register tests and convert ATPG tests into scan sequences for use in manufacturing test.
  - Circuit is designed using pre-specified design rules.

Adding Scan Structure

Diagram showing scan design with test control (TC or TCK) and scan flip-flops (SFF) connecting to the combinational logic. Not shown: CK or MCK/SCK feed all SFFs.
Built-In Self Test (BIST)  
Motivation

- Useful for field test and diagnosis (less expensive than a local automatic test equipment)
- Software tests for field test and diagnosis:
  - Low hardware fault coverage
  - Low diagnostic resolution
  - Slow to operate
- Hardware BIST benefits:
  - Lower system test effort
  - Improved system maintenance and repair
  - Improved component repair
  - Better diagnosis

Costly Test Problems Alleviated by BIST

- Increasing chip logic-to-pin ratio – harder observability
- Increasingly dense devices and faster clocks
- Increasing test generation and application times
- Increasing size of test vectors stored in ATE
- Expensive ATE needed for 1 GHz clocking chips
- Hard testability insertion – designers unfamiliar with gate-level logic, since they design at behavioral level
- Shortage of test engineers
- Circuit testing cannot be easily partitioned
Economics – BIST Costs

- Chip area overhead for:
  - Test controller
  - Hardware pattern generator
  - Hardware response compacter
  - Testing of BIST hardware
- Pin overhead -- At least 1 pin needed to activate BIST operation
- Performance overhead – extra path delays due to BIST
- Yield loss – due to increased chip area or more chips in system because of BIST
- Reliability reduction – due to increased area
- Increased BIST hardware complexity – happens when BIST hardware is made testable

BIST Benefits

- Faults tested:
  - Single combinational / sequential stuck-at faults
  - Delay faults
  - Single stuck-at faults in BIST hardware
- BIST benefits
  - Reduced testing and maintenance cost
  - Lower test generation cost
  - Reduced storage / maintenance of test patterns
  - Simpler and less expensive ATE
  - Can test many units in parallel
  - Shorter test application times
  - Can test at functional system speed
Hierarchical BIST Process

- **Test controller** – Hardware that activates self-test simultaneously on all PCBs
- Each board controller activates parallel chip BIST Diagnosis effective only if very high fault coverage

Chip BIST Architecture

- **Note**: BIST cannot test wires and transistors:
  - From PI pins to Input MUX
  - From POs to output pins
**BILBO – Works as Both a PG and a RC**

- **Built-in Logic Block Observer (BILBO) -- 4 modes:**
  1. Flip-flop
  2. LFSR pattern generator
  3. LFSR response compacter
  4. Scan chain for flip-flops

**Complex BIST Architecture**

- **Testing epoch I:**
  - LFSR1 generates tests for CUT1 and CUT2
  - BILBO2 (LFSR3) compacts CUT1 (CUT2)

- **Testing epoch II:**
  - BILBO2 generates test patterns for CUT3
  - LFSR3 compacts CUT3 response
Bus-Based BIST Architecture

- **Self-test control** broadcasts patterns to each CUT over bus – parallel pattern generation
- Awaits bus transactions showing CUT’s responses to the patterns: serialized compaction

BIST Pattern Generation

- Store in ROM – too expensive
- **Pseudo random** (LFSR) – Preferred method
- Binary counters (Exhaustive) – use more hardware than LFSR
- Modified counters – still hardware intensive
- LFSR and ROM
  - LFSR combined with a few patterns in ROM
Exhaustive Pattern Generation

- Shows that every state and transition works
- For $n$-input circuits, requires all $2^n$ vectors
- Impractical for $n > 20$

Random Pattern Testing

Bottom curve: Random-Pattern Resistant circuit (ex. PLAs)
Pseudo-Random Pattern Generation

- Standard Linear Feedback Shift Register (LFSR, n-stage)
  - Produces patterns algorithmically – repeatable
  - Has most of desirable random # properties
- Need not cover all $2^n$ input combinations
- Long sequences needed for good fault coverage