Introduction to structured VLSI design
Design for Test (DfT) - Part 2

Erik Larsson
EIT, Lund University

Outline
- Electronics
- Manufacturing
- Test, diagnosis, and verification
- Test generation
- Test points and Scan
- Built-In Self-Test (BIST)
- Systems-on-chip test
- Boundary scan (IEEE 1149.1)

Making Electronic Products

Design specification
Design
Production
Product

AND-gate
Integrated Circuits (ICs)

IC Manufacturing

- The cost to set up a modern 45 nm process is $200–500 million.
- The purchase price of a photomask can range from $1,000 to $100,000 for a single mask.
- As many as 30 masks (of varying price) may be required to form a complete mask set.

IC Defects

- Salt
- Seed
Fault Models

- A defect manifests itself as a fault
- A fault is modeled by a fault model
- Example of fault models:
  - Stuck-at Fault, Bridging Fault, Shorts (Resistive shorts), Opens, Delay Faults, Transient Fault
- So far stuck-at fault model is the most used one:
- Motivations: Simple and covers quite well possible defects

Verification, Test and Diagnosis

- **Verification** is to verify the correctness of the design. It is performed through simulation, hardware emulation, or formal methods. It is performed once prior to manufacturing. Responsible for quality of design.
- **Test** verifies the correctness of manufactured hardware. Test is a two-part process:
  - Test generation: software process executed once during design, and
  - Test application: electrical tests applied to hardware. Test application performed on every manufactured device. Responsible for quality of devices.
- **Diagnosis**: Identification of a specific fault that is present on DUT.

Test

- **Device under test (DUT)**
- **Stimulus** → **Response**
- **Stimulus**: test vectors
- Test pattern: test vector + expected test response (ordered n-tuple of binary values)
  - Produced test response is compared against expected test response

Types of Test

- **Wafer sort** - tests the logic of each die on the wafer
- **Final test** - tests the logic of each packaged IC
- **Board test** - tests interconnections (soldering errors)
Stuck-at Fault (SAF) Model

- A line is fixed to logic value 0 (stuck-at-0) or 1 (stuck-at-1)
- For the stuck-at fault model there are for a circuit with \( n \) lines \( 2^n \) possible faults

Quality of a test is given by:

\[
\text{fault coverage} = \frac{\text{faults detected}}{\text{total number of faults}}
\]

Example: 12 lines (24 faults) detect 15 faults: f.c.=15/24 (63%)

Deterministic Test Generation

While fault coverage < desired limit {
Select an uncovered fault \( f \)
Generate test for the fault \( f \)
Evaluate fault coverage
}

Needed functions to generate a test:
- Excite (provoke) the fault
- Sensitize (propagate) the results to primary outputs
- Justify other values in the circuit

ATPG:
- D-algorithm
- Path-Oriented Decision-Making (PODEM)
- Fanout-oriented Test Generation (FAN)
- Structure-oriented cost-reducing automatic test pattern generation (SOCRATES)

D-notation

- Five-valued algebra \((0,1,X,D,D')\)
- \(D=1/0\)
- \(D'=0/1\)

\[
\begin{array}{cccccccc}
\text{OR} & 0 & 1 & D & D' & X \\
0 & 0 & 1 & D & D' & X \\
1 & 1 & 1 & 1 & 1 & 1 \\
D & D & 1 & D & 1 & X \\
D' & D' & 1 & 1 & D' & X \\
X & X & 1 & X & X & X \\
\text{AND} & 0 & 1 & D & D' \\
0 & 0 & 0 & D & D' \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & D & D' & X \\
0 & D & D & 0 & X \\
D' & 0 & D' & 0 & D' & X \\
X & 0 & X & X & X & X & X \\
\end{array}
\]

- Stuck-at 0 on \( A \rightarrow Z \)
- Line \( A = D \)
- To propagate \( D \) (fault effect) to \( Z \) (check table) set \( B=1 \)

D-Algorithm

- \(D=1/0\)

\[
\begin{array}{ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
Fault Simulation

Problem and Motivation

Given

- A circuit
- A sequence of test vectors
- A fault model

Determine

- Fault coverage - fraction (or percentage) of modeled faults detected by test vectors
- Set of undetected faults

Motivation

- Determine test quality and in turn product quality
- Find undetected fault targets to improve tests

Fault Simulation

- Fault simulation consists of a fault free and a faulty circuit simulation.
- First, a fault free simulation takes place to find the fault free output responses for all patterns.
- Second, a series of simulations take place where. For each fault, fault injection is performed, the circuit is modified to become faulty. Then, the faulty circuit is simulated to find the faulty responses.

Fault Table - Analysis

- Fault simulator may provide fault table (fault dictionary)

<table>
<thead>
<tr>
<th>Pattern</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Test Set Compaction

- ATPG generates too many vectors; faults are covered by several vectors
- Test set compaction tries to reduce number of test vectors without compromising test quality
- Static test set compaction tries to remove vectors after the use of ATPG
- Dynamic test tries to remove vectors during ATPG
Commercial ATPG Tools

- Commercial ATPG tools are often for combinational circuits
- Commercial tools usually make use of a random test generation for 60-80% of the faults (easy to detect) and deterministic test generation for the remaining part (hard to detect)
- Examples of commercial ATPG tools:
  - Encounter Test - Cadence
  - TetraMax - Synopsis
  - FastScan, FlexTest - Mentor Graphics

Test Generation for Sequential Circuits

- Most real circuits are sequential
- A major problem is that the output depends not only on inputs but also on current state

![Sequential Circuit Diagram]

Outline

- Electronics
- Manufacturing
- Test, diagnosis, and verification
- Test generation
- Test points and Scan
- Built-In Self-Test (BIST)
- Systems-on-chip test
- Boundary scan (IEEE 1149.1)
Test Point Insertion

Sequential -> Combinational

- Problem: ATPG works for combinational logic while most ICs are sequential
- Solution: Provide a test mode in which flip flops can be accessed directly
- Register provide virtual primary inputs/primary outputs

Scan Design Concept

- Replace flip flop (FF) with scan flip flop (SFF): extra multiplexer on data input
- Connect SFFs to form one or more scan chains
- Connect multiplexer control signal to scan enable
Sequential -> Combinational

- Circuit can be in two modes: Functional mode and Test mode
- In Test mode test data can be shifted in and shifted out
- Test mode adds virtual PI and PO such that test data can be directly applied to combinational logic
- ATPG for combinational logic works also for sequential

Scan Test Application - first attempt

Scan Test Application - second attempt

Scan Benefits and Costs

Scan Benefits
- Automatic scan insertion
- ATPG
- High fault coverage
- Short test development time

Scan Costs
- Silicon area - Mux, scan chain, scan enable
- Performance reduction - Multiplexer in time-critical path
- IC pins - Scan-in (SI), scan-out (SO), scan_enable (SE)
- Test time - Serial shifting is slow

EDA tools
- For scan insertion (converting flip flops to scan flip flops)
- Connection
- Partial scan selection
- Scan stiching
Delay Test

- Stuck-at-fault test consist of one vector. Each vector applied at slow speed (DC-scan).
- Timing related faults need two vectors and they are to be applied on consecutive clock cycles (at normal clock speed) (AC-scan)
- At speed test:
  - Vector V1 is applied to set the circuit in its state
  - Vector V2 is applied
  - Response is captured
- Three approaches:
  - Launch-on-capture
  - Launch-on-shift
  - Enhanced scan

Launch on shift (LOS) and launch on capture (LOC)

- Launch on capture (broadsided or double capture)
  - shift in test stimuli (usually at low speed). For an n-bit shift register, shift in n bits.
  - apply a capture to create transition
  - apply another capture cycle to capture the response
- Launch on shift (skewed load)
  - shift in test stimuli (usually at low speed). For an n-bit shift register, shift in n-1 bits at low speed.
  - The final bit is shifted at high speed and then a capture is applied in high speed.

LOS and LOC

DC scan

<table>
<thead>
<tr>
<th>SE</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LOC

<table>
<thead>
<tr>
<th>SE</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LOS

<table>
<thead>
<tr>
<th>SE</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
<th>__</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Enhanced Scan

![Enhanced Scan Diagram]

- SE: Scan enable
- SI: Scan input
- SO: Scan output
Outline
- Electronics
- Manufacturing
- Test, diagnosis, and verification
- Test generation
- Test points and Scan
- Built-In Self-Test (BIST)
- Systems-on-chip test
- Boundary scan (IEEE 1149.1)

Built-In Self-Test
- Key component to discuss:
  - Test pattern storage/generation
  - Test stimuli storage/generation
  - Test response analysis
  - Test control

- In a non-BIST environment:
  - Test generation is performed by ATPG; a tool such as FastScan can generate deterministic test patterns,
  - Test stimuli and expected test responses are stored in the ATE, and
  - the ATE controls the testing and performs test evaluation.

Terms
- Test source – where test stimuli are generated/stored
- Test sink – where test responses are stored/analyzed

On-chip/off-chip
Test Pattern Generation

- How store/generate test patterns on-chip?
- Deterministic test patterns
- Exhaustive test patterns
- Pseudo-exhaustive/random test patterns
- Random test patterns

Commercial tools usually make use of a random test generation for 60-80% of the faults (easy to detect) and deterministic test generation for the remaining part (hard to detect).

Test Pattern Generation

- Exhaustive test generation; simple hardware (a counter), 100% fault coverage but too time consuming
- Deterministic test generation; high fault coverage but requires ATE for test pattern storage
- Pseudo-exhaustive test generation using Linear-Feedback Shift-Registers (LFSR)

Random Pattern Resistant Faults

- The effectiveness of a test is given based on the test’s fault coverage, length, and hardware/data storage requirement.
- Probability to create a 1 at the output; $1/2^n$ where $n$ is the number of inputs. $n=2; P=0.25, n=4; P=0.0625$

Test generations

- Some logic takes too long to test with pseudo-random patterns
- Too many specific input bit values are required
- Too many pseudo-random trials needed to achieve the required value combination
Test Response Analysis

- How store/analyze test responses on-chip?
- Compression – does not lose information
- Compaction – does loose information
- Compaction alternatives:
  - Parity check
  - One counting
  - Transition counting
  - Signature analysis

Response Compaction: Motivation

- Compaction of test responses necessary for verifying the test response
- Store compacted response called signature and compare to known fault-free signature

Compaction Options

- MISR (Multiple-Input Signature Register)
  - Very high compression ratio
  - BUT: does not tolerate x-states in test responses
    - May require product logic re-design OR
    - X-state gating logic
  - Error information loss can impact diagnostics
  - Combinational space compaction
- XOR-network
  - Can be x-state tolerant (e.g., X-Compact from Intel)
  - Less compression
  - Possibly better for diagnostics

Time Compaction

- Compress test responses in the temporal dimension
- Compress m-bit (or word) test response stream to q-bit (or word) signature stream
- Time compactor is a sequential circuit (finite-state machine)
Serial signature analysis

- Assume: $f(x) = 1 + x + x^4$. Start pattern (seed): {0000}. Fault-free test response sequence $M^* = \{10011011\}$, gives $R^* = \{1011\}$
- Faulty test response: $M' = \{10001011\}$ gives $R' = \{1110\}$, and $M'' = \{10011011\}$ gives $R'' = \{1011\}$
- The faulty response $M'$ results in $R'$ which is different from $R$ while the response $R''$ from $M''$ is not different from $R$.
- The fault detection problem ($M$ and $M''$) is called aliasing.

Space Compaction

- Compress test response in the spatial dimension
- Compress k-bit-wide response stream to q-bit signature stream ($k >> q$)
- Space compactor is a combinational circuit

![Space compactor diagram](image)

Unknons (X)

- Output from analog blocks
- Memories and non-scan storage elements
- Combinational feed-back loops
- Asynchronous set/reset
- Tristate buses
- False paths (not normal functional paths)
- Critical paths
- Multi-cycle paths
- Floating ports
- Bidirectional I/O

X-compact

![X-compact diagram](image)
X-handling

- X-Blocking; block X’s at source (where it is generated)
- X-Masking; block X’s in front of compactor

MISR Gate

- Requires masking data
  - msk='0’ blocks x-states
  - msk='1’ propagates test responses
- Trade-offs required
  - One mask value for all chains (simple)
  - Vector with unique mask value for each chain (complex)

STUMPS: Self-testing using MISR and parallel shift register sequence generator

Outline

- Test points and Scan
- Built-In Self-Test (BIST)
- Systems-on-chip test
- Boundary scan (IEEE 1149.1)
Outline

- Electronics
- Manufacturing
- Test, diagnosis, and verification
- Test generation
- Test points and Scan
- Built-In Self-Test (BIST)
- Systems-on-chip test
- Boundary scan (IEEE 1149.1)

System-on-Chip

- Viper 2.0 RevB
- Analog/Digital TV Processor
- 10mm x 10 mm (100 mm²)
- ~10 M gates
- ~50 M transistors
- ~100 clock domains

Modular Test Design

- Test Quality
  - Different parts (logic, memory, analog, RF) need different test methods
  - Black-boxed Embedded Core
    - Implementation is not known, forced to use tests developed by provider
- Divide-and-Conquer
  - Very large SOCs are intractable for ATPG/FSim tools
  - Modular test approach allows concurrent development/engineering
- Test Reuse
  - Module will be reused in other designs

Challenges

- Distributed Design and Test Development
  - Standardized set of deliverables
- Test Access to Embedded Modules
  - Standardized on-chip test access hardware
  - Tools for test translation
- Chip-Level Test Optimization
  - Tools to evaluate trade-offs; minimal impact on design (extra silicon, delay) at minimizing test application time and ATE memory requirement
Generic Test Access Architecture

- Test pattern Source and Sink
  - Store/generate test stimuli and store/evaluate test responses
- Test Access Mechanism (TAM)
  - Transports test patterns to/from module under test (MUT)
- Test Wrapper
  - Provides test access to MUT
  - Isolates MUT at test

IEEE 1500 Core Test Standard

- Goals
  - Define test interface between core and SOC
  - Core isolation
  - Plug-and-play protocols
- Scope
  - Standardize core isolation protocols and test modes
  - TAM design
  - Type of test to be applied
  - Test scheduling

IEEE 1500 Wrapper

Test Planning

- Objectives: Optimizing test access to cores and scheduling test hardware

Test hardware planning

Core import

Core integration

Test wrapper & TAM design

Test scheduling

Test software planning

Core test import

Top-level ATPG
  • Glue logic, soft cores
  • Test wrappers

Test assembly

Functional data

Test stimuli

Test responses

Test control+

WIR

Test stimuli

Test control+

Test responses

WIR

Functional data
Test Wrapper

- Test wrapper
  - Interface between module and the rest of the chip
  - makes it possible access core and isolate core from rest of the system.
- Test modes
  - Normal: Functional mode, InTest: test of module itself, ExTest: test of interconnection to other core
- IEEE 1500 Standard for Embedded Core Test

Non-modular Alternative

- Scan chain 1 (20 FFs)
- Scan chain 0 (20 FFs)
- Scan chain 1 (10 FFs)
- Scan chain 0 (10 FFs)
- Test vectors: 10
- Test vectors: 20
- Max(10,20)
- Non-modular alternative: Test time = (20+10+1)*20+(20+10)=650
- Capture

Modular Alternative

- Core 1: Test time = (20+1)*10+(20)=230
- Core 2: Test time = (10+1)*20+(20)=230
- Test vectors: 10
- Test vectors: 20
- Capture
- Total test time: 460

Problem

- For a given SoC:
  - form wrapper chains out of the scan-chains and the wrapper cells at every core
  - connect the wrapper chains to TAMs, and
  - assign a time for testing each core,
- such that the total test time is minimized.
Architecture Design

Wrapper Design

Core To TAM Assignment

Outline

- Electronics
- Manufacturing
- Test, diagnosis, and verification
- Test generation
- Test points and Scan
- Built-In Self-Test (BIST)
- Systems-on-chip test
  - Boundary scan (IEEE 1149.1)
Test Objectives

- Given a Printed Circuit Board (PCB) composed of a set of components (ICs) where each component is tested good.
- The main objectives are to ensure that all components are:
  - correct (the desired ICs are selected)
  - mounted correctly at the right place on the board and
  - ensuring that interconnections are functioning according to specification
- Problems that may occur:
  - A component does not contain logic
  - A component is not placed where it should be,
  - A component is at its place but turned wrongly,
  - A component is correct but the interconnection is not correct, for example due to bad soldering.
Boundary Scan (IEEE std. 1149.1)

- The Joint European Test Action Group (JETAG), formed in mid-80, became Joint Test Action Group (JTAG) in 1988 and formed the IEEE std 1149.1. The standard consists of:
  - Test Access Port (TAP)
  - TAP Controller (TAPC),
  - Instruction Register (IR), and
  - Data Registers (DR)

IEEE Std. 1149.1 and IEEE Std. 1500

- TMS and TCK are used to control the behavior of the TAP.
Instructions

- Mandatory
  - Bypass; used to bypassing an IC
  - Extest; tests interconnection between ICs
  - Sample/Preload; used to sample (snapshot) and preload boundary scan during operation

- Optional
  - Intest
  - Runbist
  - Clamp
  - Idcode
  - Usercode
  - Highz

Die ID - used to backtrack where a die/chip/IC comes from. Can be used to check how a particular die was tested.
Scan and MBIST support with Boundary Scan

IEEE 1149 Standard Family

<table>
<thead>
<tr>
<th>Number</th>
<th>Main objectives</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1149.1</td>
<td>Testing of digital chips and interconnects between chips</td>
<td>Std. 1149.1-1990</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Std. 1149.1a-1993</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Std. 1149.1b-1994 (BSDL)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Std. 1149.1-2001</td>
</tr>
<tr>
<td>1149.2</td>
<td>Extended digital serial interface</td>
<td>Discontinued</td>
</tr>
<tr>
<td>1149.3</td>
<td>Direct access testability interface</td>
<td>Discontinued</td>
</tr>
<tr>
<td>1149.4</td>
<td>Mixed-signal test bus</td>
<td>Std. 1149.4-1999</td>
</tr>
<tr>
<td>1149.5</td>
<td>Standard module test and maintenance (MTM) bus</td>
<td>Std. 1149.5-1995 (not endorsed by IEEE since 2003)</td>
</tr>
<tr>
<td>1149.6</td>
<td>High-speed network interface protocol</td>
<td>Std. 1149.6-2003</td>
</tr>
<tr>
<td>1149.7</td>
<td>Reduced-Pin and Enhanced-Functionality Test Access Port</td>
<td>Std. 1149.7-2009</td>
</tr>
</tbody>
</table>

Introduction to structured VLSI design
Design for Test (DfT) - Part 2

Erik Larsson
EIT, Lund University