SoC Testing

Jiun-Lang Huang
Graduate Institute of Electronics Engineering
National Taiwan University
Outline

- **SoC Test Challenges**
- Test Access Mechanism
- Core Test Wrapper
- IEEE P1500
- Test Optimization
Technology Trend: System-on-Board to System-on-Chip

Physical components

Virtual components

System on Board

System on Chip
Why System-on-Chip?

- Complex applications.
- Process technology allows it.
- High performance.
- Miniaturization.
- Battery life.
- Short market windows.
- Cost sensitivity.
Core-Based SoC Design

- Reuse of large modules.
  - Examples: CPU, DSP, MPEG, JPEG, communication modules, memories, analog modules, ...
  - Reduced time-to-market, expertise import.

- ‘Divide-and-conquer’ design methodology
  - Maximize core-level design tasks.
  - Minimize SoC-level design tasks.

- Distributed design: core provider and user
  - Intra-company and inter-company core use.
The Impact

- SoC components are only manufactured and tested in the final system.
Separation of Responsibilities

- The core provider
  - Test pattern generation for the cores.
  - Core internal design-for-testability.

- The core user
  - Test generation for the chip
    - Reuse of core-level test patterns.
    - Additional test patterns for non-core circuitry.
  - Chip-level design-for-testability.
**SoC Test Challenges**

- Distributed design & test
- Test access
- Test optimization
Distributed Design & Test

- In general, the core provider develops the core test including DfT & test patterns.
- However, the core provider does not know the system chip environment
  - Which test method to use?
  - What type of faults to target?
  - What level of fault coverage?
which may lead to
  - inadequate test quality, or
  - waste of resources.
Need a set of standardized set of deliverables.

- Test methods
- Test modes and protocols
- Fault models and fault coverage
- Test pattern data
- Core-internal design-for-test
- Core-internal design-for-diagnosis
- Diagnostics and failure analysis information
Test Access

- Direct access to deeply embedded cores is difficult.
- It’s not uncommon that core’s I/O pin count > SoC’s I/O pin count
- To test each core, we need to provide
  - core test access, and
  - core isolation mechanism.
Test Optimization

- Test access infrastructure optimization
  - Test quality vs. overhead
- Constrained test scheduling
  - Overall test time
  - Power consumption
  - Test access bandwidth
  - Available test resources
  - Other considerations
Outline

- SoC Test Challenges
- **Test Access Mechanism**
- Core Test Wrapper
- IEEE P1500
- Test Optimization
A Conceptual SoC Test Access Architecture

- Test pattern source and sink
  - The source generates the test stimuli for the embedded core.
  - The sink compares the responses to the expected responses.

- Test access mechanism (TAM)
  - Test data transport from the test source to the CUT and from the CUT to the test sink.

- Core test wrapper
  - Connects the terminals of the core to the rest of the IC and the TAM.

[Zorian, Marinissen, Dey – ITC98]
A Conceptual Test Architecture
On/Off-Chip Test Source/Sink

**Off-chip Source/Sink**
- Bandwidth limited by pin count
- More TAM area
- More expensive ATE

**On-chip Source/Sink**
- Closer to CUT
- Less TAM area
- Less dependence on ATE
- BIST area overhead
TAM (Test Access Mechanism)

● Function
  – Deliver test stimuli from the test source to the CUT.
  – Transport test responses from the CUT to the test sink.

● TAM design involves making trade-offs among
  – data transport capacity,
  – test time, and
  – TAM overhead.
TAM Width

- Determines the test data transport bandwidth.
- Considerations
  - A wider TAM shortens the test time, but consumes more wiring area.
  - The width of the test source and sink.
  - Available IC pins if external test source/sink.
- Constraints to meet
  - Test time
  - Area overhead
TAM Length

- Physical distance
- Ways to reduce TAM length
  - On-chip test sources and/or sinks.
  - Sharing TAM among cores can shorten the total TAM length.
    - Reduced wiring area.
    - Possibly reduced test concurrency.
  - Reusing functional hardware as TAM.
    - May not meet the desired test time constraint.
TAM Implementations

- Direct access scheme
  - Immaneni, Raman – ITC90

- Bus-based scheme
  - Varma, Bhatia – ITC98, Harrod – ITC99

- Transparency
  - Beenker – D&T86, Beenker 95, Marinissen – TECS97, Ghosh et al. – ITC97, CAC98

- Boundary-scan based
  - Whetsel – ITC97, Bhattacharya – VTS98, Touba, Pouya – D&T97, ITC97

- Test Rail
  - Marinissen et al. – ITC98
Direct Access Scheme

- Map all core inputs, outputs, and I/O onto package pins.
- In test mode, the I/Os of the selected core are accessible through a group of package pins.
  - Each core can be tested with its standard test program.
- Test isolation provided and cores are tested independently.

[Immaneni, Raman – ITC90]
Direct Access Scheme

- Modification to user logic block.
Direct Access Scheme

- An implementation example
Remarks

● Advantages
  – Embedded cores can be tested and debugged as a stand-alone device.
  – Transition from core-level test to chip level test is simple.
  – A slight increase in overall package pin count and design complexity.

● Drawbacks
  – Not scalable.
    • The complexity of control logic and test circuitry grows with the number of embedded cores.
  – Long test time.
    • Blocks are tested sequentially.
Bus-Based TAM

- Utilizing on-chip system bus or dedicated test bus for test data transport.
  - Varma, Bhatia – ITC98
  - Harrod – ITC99
AMBA Bus-Based Testing

- Test vectors produced for an AMBA-compliant IP block can be reused in any AMBA-based system.
- The AMBA Test Interface Controller (TIC) is responsible for test application and response capture.
- In test mode
  - TIC becomes the AMBA bus master.
  - The external bus interface (EBI) is reconfigured to provide a high-speed, 32-bit, parallel vector interface.

[Harrod – ITC99]
AMBA Bus-Based Testing

- Peripheral test harness
  - Access to I/O’s not connected to the bus.
  - Isolate the core under test from its environment.
AMBA Bus-Based Testing

System on Chip (SoC)

Test Interface Controller (TIC) → Arbiter → CPU Core → On-Chip Memory → ASB to APB Bridge

External Bus Interface

High Bandwidth Peripheral A → AMBA System Bus (ASB) → High Bandwidth Peripheral B

Serial Port → GPIO Interface → Timer → Interrupt Controller

AMBA Peripheral Bus (APB)
Remarks

● Advantages
  – Reusing system bus reduces TAM overhead.
  – Transition from core-level test to chip-level test is simple.

● Drawbacks
  – Fixed bus width may be insufficient for some cores.
  – Difficult to integrate full-scanned cores.
Transparency
**Transparency**

- **Transparent path**
  - A path from input to output which propagates data without information loss.

- **Examples**
  - Scan chains
  - Arithmetic functions: $+0$, $\times 1$
  - Embedded memories
  - Basic gates: AND, OR, INV, MUX

- **Past techniques**
  - *Beenker* – *D&T86, Beenker 95,\n    Marinissen – TECS97,\n   Ghosh et al. ITC97, CAC98*
Remarks

● Advantages
  – Low area overhead

● Drawbacks
  – Transport latency through cores
  – The desired transparency is not guaranteed.
    • Too much transparency – waste.
    • Too little transparency – TAM needed.
  – Non-trivial transition of core-level test to chip-level test.
Boundary Scan

- An IEEE 1149.1 compliant chip.

- 1149.1 board-level view.

![Diagram of Boundary Scan]

- TCK
- TMS
- TDI
- TDO

: Boundary scan cell

: Boundary scan path
Remarks

- Advantages
  - Existing well-known, well-documented standard.
  - Reuse IC level implementation.

- Drawbacks
  - Fixed 1-bit TAM width.
  - Complexity of test control and test data wiring grows with the number of cores.
  - Multiple TAP controllers.
Test Rail

- IC level view.

[Marinissen et al. – ITC98]
Core level view.
**Test Rail**

- An example.
  - A: 4 scan chains.
  - B: BIST
  - C: Functional test
Remarks

● Advantages
  – Flexible
    Enables integration of various core test techniques.
  – Scalable
    Allows trade-offs between area, quality, and test time.

● Drawbacks
  – Difficult to find optimal solution.
Multiplexing Architecture

[Aerts & Marinissen - ITC98]
Daisy Chain Architecture

[Aerts & Marinissen – ITC98]
Distributed Architecture
Remarks

- Different types of TAMs may coexist on a single chip.
- Cores connected to different TAMs can be tested concurrently.
- Cores connected to the same TAM cannot be tested concurrently.
Outline

- SoC Test Challenges
- Test Access Mechanism
- Core Test Wrapper
- IEEE P1500
- Test Optimization
Core Test Wrapper

● Function
  – Interface between the core and its environment.
    • Width adaptation
  – Provision of the following modes
    • Normal: function mode
    • InTest: inward-facing core test mode
    • ExTest: outward-facing interconnect test mode

● Considerations
  – Test time.
  – Performance degradation.
  – Area overhead.
Functional Only Connection

Core A — Core B — Core C

Scan Chain — Scan Chain — Scan Chain

Scan Chain — Scan Chain — Scan Chain

Scan Chain — Scan Chain — Scan Chain
Wrapper & TAM

Core A

Core B

Core C

Test Control

Test Control

Test Control

Scan Chain

Scan Chain

Scan Chain

Scan Chain

Scan Chain

Scan Chain

Bypass

Bypass

Bypass
Normal Operation
InTest
Bypass
Outline

- SoC Test Challenges
- Test Access Mechanism
- Core Test Wrapper
- IEEE P1500
- Test Optimization
Core-Based Test

- The core provider delivers
  - the core design itself, and
  - a set of tests for the core.

- The core user assembles a chip-level test from
  - the pre-defined tests for the various cores, and
  - additional tests for non-core circuitry.

- A test as described above, in which cores are tested as stand-alone units, is called a “core-based test.”
IEEE P1500

- Standard for embedded core test (SECT).
- History
  - Started in 1995 as a TAC of the TTTC of the IEEE Computer Society.
  - In June 1997, Project Authorization Request approved by the IEEE Standards Activities Board.
  - Current target
    • A draft standard for non-merged digital logic and memory cores.
  - Future extensions
    • Mergable cores and analog and mixed-signal cores.
Goals of IEEE P1500

- Standardize a core test architecture which
  - Defines a core test interface between an embedded core and the system chip,
  - Facilitate test reuse for embedded cores through core access and isolation mechanisms, and provide testability for system chip interconnect and logic, and
  - Facilitates core test interoperability, with plug-and-play protocols, in order to improve the efficiency of test between core providers and core users.
Scope of IEEE P1500

• Yes.
  – Standarize core test mechanisms, for core access and isolation, including protocols and test mode control.

• No.
  – System chip test access mechanism.
    • Defined by the system chip integrator.
  – Core test method.
    • Defined by the core provider.
    • P1500 supports, and enables various different methods, e.g., scan, BIST, I_{ddq}, etc.
P1500 Task Forces

- Core test language
- **Scaleable architecture**
- Compliance definition
- Terminology / Glossary
- Documentation
- Mergeable cores test
- Benchmarking
- Industry & media relations
P1500 Core Test Wrapper Architecture

E. J. Marinissen, CTAG Working Group
in DATE’03
P1500 Core Wrapper Architecture

- Wrapper overview and applications
- Wrapper architecture and instructions
- Wrapper harness cells
Block Level Overview

- **WPP**
  - Optional.
  - User defined port for test flexibility.
  - Components:
    - Wrapper Parallel In
    - Wrapper Paralles Control
    - Wrapper Parallel In

- **WSC**
  - Required.
  - Standardized port for plug-and-play.
  - Components:
    - WSI, WSC, WSO
The P1500 Wrapper Boundary Cell

- **Cell modes**
  - Normal, Inward facing, Outward facing, Safe (recommended)

- **Cell events**
  - Shift, Capture, Apply, Update, Transfer
P1500 Wrapper Parameters

- **Bandwidth**
  - Number and/or width of WPI-WPO pairs.

- **Instructions**
  - Optional instructions
  - User-defined instructions
  - OpCodes of instructions

- **WBR functionality**
  - Shared or dedicated wrapper cells
  - Shift-only or Shift + Update cells
  - Storage capacity (one or more bits)
  - Ripple protection (w/ Update register or gate)
  - “Safe State” output values
The P1500 Wrapper Architecture
Wrapper Configuration
Daisy-Chained Wrapper Configuration
Bussed Wrapper Configuration
Direct Access Wrapper Configuration
Wrapper Configuration w/ Local Controllers
P1500 Core Wrapper Architecture

- Wrapper overview and applications
- Wrapper architecture and instructions
- Wrapper harness cells
**Required Wrapper Architecture**

- **WSC**
  - WRST, WCLK, SelectWR, Capture, Shift, Update, Transfer
P1500 Wrapper Test Instructions

- WS_BYPASS Required WSP
- WS_PRELOAD Optional WSP
- WP_PRELOAD Optional WPP
- WS_CLAMP Optional WSP
- WS_SAFE Optional WSP
- WS_INTEST_RING Optional WSP
- WS_INTEST_SCAN Optional WSP
- WP_INTEST_RING* Optional WPP
- WP_INTEST_SCAN* Optional WPP
- WH_INTEST Optional and/or
- WS_EXTEST Required WSP
- WP_EXTEST Optional or
- WH_EXTEST Optional and/or

1. At least one optional INTEST required.
2. *: Examples of user-defined instructions.
Instruction Fields Naming Conventions

[Field 1] [Field 2] [Field 3] [Field 4]

Wrapper Test Access Test Mode Configuration

W S:serial Bypass Ring
P:parallel Preload Scan
H:hybrid Clamp User

Safe Intest

Extest User

WS_Intest_Ring
WS_Bypass

![WS_Bypass Diagram](image-url)
WS_Preload
WP_Preload

Core: Enabled

Test enable(s)

WPC signals are user defined.

Note: WPC signals are user defined.
WS_Clamp

1. Inputs may also be controlled to preloaded states.
2. WS_Clamp is preceded by WS/P_Preload.
1. Inputs may also be controlled to safe states.
2. WS_Safe does no require WS/P_Preload.
WS_Intest_Ring

1. Test is applied via WSC controlled WBR scan path.
**WS_Intest_Scan**

1. Test is applied via WSC controlled WBR and internal scan path.
1. Test is applied via WPC controlled WBR segments.

* Under consideration as a standard instruction.
WP_Intest_Scan

1. Test is applied via WPC controlled WBR and internal scan segments.

* Under consideration as a standard instruction.
1. Test is applied via WSC and WPC controlled WBR and internal scan path.
1. Test is applied via WSC controlled WBR scan path.
WP_Extest

1. Test is applied via WSC controlled WBR segments.

Core: Disabled

Test enable(s)
WP_Extest

1. Test is applied via WSC controlled WBR segments.
1. Test is applied via WSC & WPC controlled WBR segments.
P1500 Core Wrapper Architecture

- Wrapper overview and applications
- Wrapper architecture and instructions
- Wrapper harness cells
Applying Wrappers to Test Only I/O’s
Pass-Through Harness Cell
Scan-Through Harness Cell
Pass/Scan-Through Harness Cell
Pass-Through Harness Cell Example
Scan-Through Harness Cell Example

[Diagram of a scan-through harness cell example with labeled components such as WSI, WSO, Bypass, WIR: WS_Intest, and WSC.]
Pass/Scan-Through Cells: Pass Mode

1. Enables factory testing using parallel access.
Pass/Scan-Through Cell: Scan Mode

1. Enables in-system testing using serial access.
Outline

- SoC Test Challenges
- Test Access Mechanism
- Core Test Wrapper
- IEEE P1500
- Test Optimization
Problem Definition

- SoC test scheduling
  - Given:
    A set of cores, the associated test sets (maybe more than one), and a set of constraints that must be satisfied during testing.
  - Find out:
    The start time(s) of the test set(s) for each core such that the total test time is minimum and none of the constraints are violated.
  - Note:
    The test schedule should be identified efficiently, and the results should be optimum or near optimum.
SoC Test Constraints

- TAM width
- Power dissipation
- Resource sharing
- Precedence
- Multiple test sets
**TAM Width Constraint**

- Usually, the available TAM width for testing is limited and fixed.
- TAM width for a core may be adjustable.
**Power Dissipation Constraint**

- The maximum power consumption is limited.
- Elevated power consumption during testing.

![Diagram of One-level and Two-level Power Models](image-url)
Test Scheduling Methodologies

- Graph based
- ILP based
- Bin-packing based
Graph-Based Scheduling Methodologies

- Developed in 80s’ and early 90s’.
- Aim at exploring parallelism in BIST.
- Test resource conflict is the major constraint.
Resource Allocation & Test Compatibility Graphs

- Resource allocation graph (LHS)
  - TI: test sets
  - Ri: resources

- Test compatibility graph (RHS)
  - Two tests are incompatible if they use the same resource.
  - No edge between incompatible test sets.
Minimum Covering Table

- Tests in the same compatible set can be performed concurrently.
- Cover all tests with minimum number of rows.

<table>
<thead>
<tr>
<th>Test compatible set</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1: {T1, T3, T4}</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G2: {T1, T3, T5}</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G3: {T1, T6}</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G4: {T2, T6}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G5: {T2, T5}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**ILP Based Methodologies**

- **An example**

Maximize

\[ X + Y \]

subject to

\[ 6x + y \leq 30 \]
\[ 3x + 2y \leq 30 \]
\[ x + 4y \geq 32 \]

where

\[ x, y \geq 0 \]
**General Form**

Maximize or minimize

\[ Y = C_1X_1 + C_{12}X_2 + \cdots + C_nX_n \]

subject to

\[
\begin{align*}
A_{11}X_1 + A_{12}X_2 + \cdots + A_{1n}X_n & \geq, =, \leq B_1 \\
A_{21}X_1 + A_{22}X_2 + \cdots + A_{2n}X_n & \geq, =, \leq B_2 \\
& \vdots \\
A_{m1}X_1 + A_{m2}X_2 + \cdots + A_{mn}X_n & \geq, =, \leq B_m 
\end{align*}
\]

where

\[ X_j \geq 0, \ j = 1, 2, \cdots, n \]

\[ A_{ij} \in R, \ i = 1, 2, \cdots, m, \ j = 1, 2, \cdots, n \]

\[ B_i \in R, \ i = 1, 2, \cdots, m \]

\[ C_i \in R, \ i = 1, 2, \cdots, n \]
Mixed-Integer Linear Programming

- MILP
  - Linear constraints and linear objective functions
  - Combine integer in the model

- Allow the use of integer, real number, rational numbers in the model.

- Integers can be used as decision variable and indicator variable which are also called 0-1 variables.
Decision Variables

- Represent the possible decisions
  - Example:
    \[ R_{ik} = \begin{cases} 
    1 & \text{if test set } i \text{ is assigned to resource } k \\
    0 & \text{otherwise} 
    \end{cases} \]
  
  - Example for 2 test sets with 3 resources
    \[
    R_{11} + R_{12} + R_{13} = 1 \\
    R_{21} + R_{22} + R_{23} = 1
    \]
Indicator Variables

- Used to link the logic condition between two constraints.
- Indicate the state of the variable.
- Possible values are 0 and 1.
- Example:
  - Logic condition:
    \[ x > 0 \land \land = 1 \]
  - Constraint:
    \[ x \in [0, M], M \text{ is } x's \text{ upper bound.} \]
**Problem Formulation**

- **Linearize nonlinear item: \( x^d \).**
  - \( x \) is a non-negative real number.
  - \( d \) is a 0-1 variable.

- **Linearization procedure**
  1) \( y = \lfloor x \rfloor \)
  2) List logic condition
     - \( d = 0 \quad y = 0 \)
     - \( d = 1 \quad y = x \)
  3) Add extra constraints to represent
     - \( y \leq M \cdot d \quad \leq 0 \)
     - \( x - y + M \cdot d \geq 0 \)
     - \( y \geq 0 \)

\( M \) is the upper bound of \( x \) and \( y \).
Problem Formulation - cont’d

- Additional constraints can be easily introduced to model.
- The number of variables and constraints increase incredibly fast as the number of cores increase.
- ILP is computationally expensive for large large SoC’s.
Bin Packing Based Methodologies

- Determine how to put the most objects in the least number of fixed space bins.

- More formally, find a partition and assignment of a set of objects such that a constraint is satisfied or an object function is minimized (or maximized).
One-Dimensional Bin Packing

- Given:
  A list of objects with their associated values, and a fixed bin size.

- Find:
  The assignment of objects to bins such that the least number of bins are needed.
A Best-Fit Algorithm Example

• Placing each package in the “best” place that can hold it.
  - The best bin is the one with the least amount of space left after the package is added.
Two-Dimensional Bin Packing

- **Given:**
  A set of orthogonal rectangles and a fixed-width bin.

- **Objective:**
  Pack the rectangles into the bin so as to minimize the height of the bin.

- **Remarks**
  - The rectangles do not overlap with each other or the edges of the bin.
  - The rectangles are packed with their sides parallel to the sides of the bin. (Rotations disallowed.)
An Example
Multi-Dimensional Bin Packing

- 3D is complicated enough.
- When more constraints are considered, multi-dimension problem will be extremely complicated.
- Can be solved within reasonable time.
Dual Rectangle Bin Packing

- Each test set is represented by two rectangles packed into two different open-ended 2-D bins.

- Other constraints are considered when packing the rectangles into the bins.
An Example