Testing of Digital System-on-Chip (SoC)
Outline of the Talk

- Introduction to system-on-chip (SoC) design
- Approaches to SoC design
- SoC test requirements and challenges
- Core test wrapper
- P1500 core test standard
- Summary
What is a System-on-Chip (SoC)?

- The integration of a complete system on an IC chip.
  - Earlier consisted of multiple IC chips.
  - May include multiple types of design blocks and intellectual property (IP).
    - Digital logic blocks, processors, memories and analog circuitry.
- Typically SoC’s are designed using embedded reusable cores.
Traditional vs. SoC Design

- **Traditional IC design**
  - Design the whole IC from scratch.
  - Reuse: standard cell library, memory, etc.

- **SoC (core-based) design**
  - Reuse: large modules, like CPU, DSP, MPEG, I/O controllers, analog modules, etc.
  - Reduced time to market.
  - Sharing / importing expertise.
SoC Chip Evolution

UDL: user-defined logic

Original Logic Design

Reused Logic Design

UDL

SRAM

ROM

Reused Logic Design (IP Core)

uP (IP Core)

DRAM

Analog

ROM

ATM (IP Core)

MPEG (IP Core)

DSM ASIC

Block-Based SOC

Core-Based Plug & Play SOC
## PCB/SoB vs. SoC

<table>
<thead>
<tr>
<th>PCB/SoB</th>
<th>SoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tested parts</td>
<td>High reliability</td>
</tr>
<tr>
<td>In-circuit test (ICT)</td>
<td>Fast interconnects</td>
</tr>
<tr>
<td>Easy test access</td>
<td>Low cost</td>
</tr>
<tr>
<td>Bulky</td>
<td>Untested cores</td>
</tr>
<tr>
<td>Slow</td>
<td>No internal test access</td>
</tr>
<tr>
<td>High assembly cost</td>
<td>Mixed-signal devices</td>
</tr>
</tbody>
</table>
SoB vs. SoC Test Process

System-on-Board (SOB) Process
- IC Design
- IC Manuf.
- IC Test
- SOB Design
- SOB Manuf.
- SOB Test

System-on-Chip (SOC) Process
- Core Design
- UDL Design
- SOC Integration
- SOC Manuf.
- SOC Test
SoC realization process is analogous to SoB using standard parts.
SoC cores and UDL not manufactured and tested individually.
Cores and UDL are tested together.
SoC test integration requires
- test data provided with each core.
- core test integration methodology and tools.
Core-based Design

- Cores are pre-designed, verified but untested functional blocks.
  - **Soft core** ➔ synthesizable RTL
  - **Firm core** ➔ gate-level netlist
  - **Hard core** ➔ non-modifiable layout (legacy core)
- Core is the intellectual property of vendor.
  - Internal details not available to user.
- Core-vendor supplied tests must be applied to embedded cores.
Core Types

- A **soft core** is a technology-independent synthesizable RTL circuit description.
  - Specified in a high-level description language, and hence technology independent.
  - Layout is completely flexible.
  - Performance and area depend on cell library used.
  - Synthesis, test, timing analysis, and verification are required.

- A **firm core** is a technology-dependent gate-level netlist that meets timing constraints.
  - Layout is flexible.
  - Performance and area are more predictable.
  - May be encrypted to protect IP.
Contd.

- A **hard core** includes layout and timing information.
  - Given as layout files that cannot be modified.
  - Highly optimized for area and performance, and synthesized to a specific technology.
  - Includes behavioral model for simulation.
  - May be encrypted.
  - Test sets (test stimuli and test responses) are given.
  - Gives less flexibility to the core integrator, but saves design time and effort.
SoC Test Requirements

- Deeply embedded cores
  - Need *Test Access Mechanism* to access the input/output pins of the core being tested.

- External ATE inefficiency
  - Need “on-chip” ATE.

- Mixing technologies: logic, processor, memory, analog components
  - Need various DFT/BIST techniques.
SoC Test Requirements (contd.)

- Multiple hardware description level for cores
  - Need to insert DFT/ BIST at various levels.
- Different core providers and SoC test developers
  - Need standard for test integration.
- Core/ test reuse
  - Need plug-and-play test mechanism.
- IP protection
  - Need core test standard/ document.
SoC Test Methodology

- Study functions and architectures in each module of a general SoC.
- Design each module.
- Apply proper testing methods to each module.
- Add *wrapper* to each core (module).
- Integrate the IP testing using some standard.
  - P-1500 core test standard.
**Test Challenges in SoC Design**

- For conventional ASIC’s and standard IC’s mounted on a circuit board:
  - The individual chips are tested separately before being assembled on the board.
- With SoC design,
  - The individual cores are not tested before the complete SoC is fabricated.
Test Challenges in SoC Design (contd.)

- There is no direct access to the core I/O ports from the chip I/O’s.
  - Often involves an additional DFT effort.
  - Core integration

- Use of multiple cores within one design.
  - Cores may use different DFT strategies and test methods.

- Compose an integrated test and its control mechanism for the overall system chip.
  - Requires scheduling to meet chip-level requirements.
Test-Wrapper for a Core

- **Test-wrapper (or collar)**
  - Logic added around a core.
  - Provides test access to the embedded core.
- **Test-wrapper provides:**
  - For each core input terminal:
    - **A normal mode:** Core terminal driven by host chip.
    - **An external test mode:** Wrapper element observes core input terminal for interconnect test.
    - **An internal test mode:** Wrapper element controls state of core input terminal for testing the logic inside core.
For each core output terminal:

- **A normal mode**: Host chip driven by core terminal.
- **An external test mode**: Host chip is driven by wrapper element for interconnect test.
- **An internal test mode**: Wrapper element observes core outputs for core test.
A Test-Wrapper (similar to P1500)
DFT Architecture for SoC

User defined test access mechanism (TAM)

Test source

Functional inputs

Module 1

Test wrapper

Instruction register control

Test access port (TAP)

Module N

Test wrapper

Test sink

Functional outputs

SOC inputs

TDI, TCK, TMS, TRST, TDO

SOC outputs

Serial instruction data

Functional inputs

Functional outputs
SoC DFT Components

1. **Test source:**
   - Provides test vectors via on-chip LFSR, counter, ROM, or off-chip ATE.

2. **Test sink:**
   - Provides output verification using on-chip signature analyzer, or off-chip ATE.
Contd.

3. **Test access mechanism (TAM):**
   - User-defined test data communication structure.
   - Carries test signals from source to module, and module to sink.
   - Tests module interconnects via test-wrappers.
   - May contain bus, boundary-scan and analog test bus components.
4. **Test controller:**
   - Boundary-scan *test access port* (TAP).
   - Receives control signals from outside.
   - Serially loads test instructions in test-wrappers.
IEEE P1500 Organization

- Aim is to standardize interface between core provider and core user.
- P1500 specifies:
  1. Standardized, scalable core test wrapper.
  2. Core test information model.
     - Described in core test language (CTL).
  3. Two compliance levels.
     - IEEE 1500 unwrapped.
     - IEEE 1500 wrapped.
IEEE P1500 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.
  - Does not address test access mechanism (TAM) design.
  - Does not specify type of test to be applied.
P1500 Standard Wrapper

Wrapper Modes
1. Normal
2. Serial Test
3. In Test
4. Bypass
5. Isolation
6. Ex Test
Wrapper Elements

- **Wrapper Instruction Register (WIR)**
  - Controls operation of wrapper.
  - Mandatory, optional and user-defined instructions.
  - Controlled directly from WIP signals.
  - Instructions loaded via WSI-WSO.

- **Wrapper Bypass Register (WBY)**
  - Mandatory bypass for serial TAM.
  - Bypass path between WSI and WSO.
Wrapper Elements (contd.)

- Wrapper Boundary Register
  - Provides accessibility to core terminals.
  - Built from library of wrapper cells.
  - Configured into one or more scan chains in test mode.
  - Test data loaded from WSI-WSO or WPI-WPO.
Wrapper Interface Pins

- Functional inputs and outputs
  - Corresponding to the core’s functional input/output pins.

- Wrapper Interface Port (WIP)
  - 6-bit control port.

- Serial Interface: WSI-WSO
  - Load instruction into WIR (test control).
  - Load test data to WBR and WBY.

- Parallel Interface: WPI-WPO
  - Test data into WBR.
  - User defined width.
  - Zero or more parallel ports (typically one).
Wrapper Overview

Test stimuli

Functional inputs

WSI

WPI

WBR

WBR

WBR

WSO

WPO

WBPY

WIR

WIP

CORE

test i/o

functional i/o
Example: without wrapper
Hidden

Core B

Scan chain
Scan chain

Test Control Block
Example: normal operation
Example: InTest mode
Example: ExTest mode
Example: Bypass mode
Test Wrapper Design

Priority 1: Balanced Wrapper Scan Chains

- Unbalanced
  - Minimize length of longest wrapper scan in/out chain

- Balanced
Reducing TAM Width

Priority 2: Minimize wrapper scan chains created

<table>
<thead>
<tr>
<th>Scan chain – 32 FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 FF</td>
</tr>
<tr>
<td>8 FF</td>
</tr>
<tr>
<td>8 FF</td>
</tr>
</tbody>
</table>

4 Wrapper scan chains

2 Wrapper scan chains

<table>
<thead>
<tr>
<th>Scan chain – 32 FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 FF</td>
</tr>
</tbody>
</table>
TAM Design

1. Partial isolation rings
2. Multiplexing
3. Core Transparency
TAM Design & Test Scheduling

- Given the test set parameters for the cores and the total TAM width $W$.
- Assign a part of $W$ to each core, design a wrapper for each core, and determine the test schedule, such that
  - $W$ is not exceeded at any time.
  - Testing time is minimized.
### SoC Testable Design Flow

<table>
<thead>
<tr>
<th>Design Requirement/Test Requirement</th>
<th>P1500 Ready Core?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Architecture</td>
<td>P1500 Compliance Checker/Add Wrapper</td>
</tr>
<tr>
<td>System Partition and IP Survey</td>
<td></td>
</tr>
<tr>
<td>UDL Behavior/RTL/Gate Level</td>
<td>SOC Testable Design Rules</td>
</tr>
<tr>
<td>UDL DFT Insertion</td>
<td></td>
</tr>
<tr>
<td>System Integration</td>
<td></td>
</tr>
<tr>
<td>Test Integration</td>
<td></td>
</tr>
<tr>
<td>System Specification Verification</td>
<td></td>
</tr>
</tbody>
</table>

- **Test Requirement**
  - Test Access Mechanism Synthesis
  - Test Controller Synthesis
  - Test Bench Integration
  - Test Bench Verification/Illegal Test Pattern Checker
Summary

- SoC test technologies are as varied as the cores in the chip.
- An integrated set of different test technologies offers the most effective SoC test solution.
- SoC complexity requires a hierarchical, reusable test architecture.
Testing of SoC’s must be addressed:

- Need to provide a mechanism to access core I/O’s from chips I/O’s.
- Need to target relevant fault models.
- Need to limit test application time.
- Power budget not exceeded during test.
References (selected few)


