System-on-Chip Test: Methodology & Experiences

Y. Zorian, D. Burek, LogicVision
S. Mukherji, Fujitsu Microelectronics
Contents

- Introduction
- SOC Test Requirements
- SOC Test Architecture
- SOC Test Methodology
- Case Study
- P1500 Standardization
- Conclusions
System-on-Chip Drivers

- Primary SOC Market Drivers - Consumer Electronics:
  - complex products
  - shrinking market windows
  - cost sensitivity
  - high reliability
  - miniaturization
System-on-Chip Paradigm

- Emergence of -
  - Very large transistor counts on a single chip
  - Mixed technologies on the same chip
    - Logic, Analog, Memory, Processor
  - Creation of Intellectual Property (IP)
  - Reusable core-based design
    - Cores replacing standard parts, such as DSP, DRAM, MCU, Flash, and FPGA
System-on-Chip Evolution

Original Logic Design

Reused Logic Design

Reused Logic Design (IP Core)

uP (IP Core)

DRAM

Analog

ROM

UDL

UDL

ATM (IP Core)

MPEG (IP Core)

DSM ASIC

Block-Based SOC

Core-Based Plug & Play SOC
Contents

- Introduction
- SOC Test Requirements
- SOC Test Architecture
- SOC Test Methodology
- Case Study
- P1500 Standardization
- Conclusions
SOC Test Specifics

- SOC realization process analogous to SOB using standard parts
- SOC cores and UDL not manufactured and not tested individually
- Cores and UDL are tested simultaneously
- SOC test integration requires test data provided with each core and core test integration methodology and tools
SOC Test Requirements

1. Deeply Embedded Cores
   ♦ *access* to core ports limited
   ⇒ need *Test Access Mechanism* to transport test from source to core and from core to sink

2. Mixing Technologies on Same Process
   ♦ cause *yield* limitations
   ⇒ allow different *types of testability*: logic, analog, memory, processor
   ⇒ *debug and diagnosis* modes for each technology type

3. More, Higher-Performance Core Pins than SOC Pins
   ♦ limited SOC I/O *bandwidth*
   ⇒ need *source/sink* that scales to bandwidth
**SOC Test Requirements**

4. Ability to reuse same core in different SOCs
   - efficiency obtained by ease of plug-and-play
     ⇒ need reusable core test solution

5. High-Performance Operation
   - increased yield loss due to guard-banding
     ⇒ need at-speed test execution using system clock

6. Protection of Intellectual Property
   - limited core design knowledge by chip integrator
     ⇒ need self-contained test without need for core design knowledge
SOC Test Requirements

7. Cost of ATE - Clustered or Multiple Insertion
   ♦ substantially more expensive than conventional ATE
     ⇒ need more cost-effective solution to complement single
        insertion on existing ATE
     ⇒ reduce test time by time sharing (core test parallelism)

8. Cost Sensitivity of SOC Market
   ♦ low-cost consumer products
     ⇒ need cost-effective testability solution that optimizes
        implementation (space sharing)

9. Multiple Hardware Description Levels for Cores
   ♦ Behavioral, RTL, and layout (soft, firm, hard cores)
     ⇒ allow core testability at different levels (flexibility)
10. Hierarchical Reuse of Cores

♦ vertical integration: chip becomes next generation core
⇒ need hierarchical core test integration for testing complex cores, chips, and beyond

11. Limited Time-to-Market Window

♦ tighter schedules for SOC development
⇒ need automation for core test preparation and SOC test integration

12. Numerous Core Providers and SOC Test Developers

♦ diverse core test interfaces and information models
⇒ need standardized test interface between core and UDL (User-Defined Logic)
⇒ need standardized core test information model
Summary of Test Requirements

- Sockets and Test Access Mechanisms
- Bandwidth-Scalable Test Mechanism
- Test Solutions for Different Technology Types
- Automation of Core Test and SOC Test
- Different Core Testability Techniques
- Debug and Diagnosis
- Hierarchical Core Test Integration
Contents

- Introduction
- SOC Test Requirements
- SOC Test Architecture
- SOC Test Methodology
- Case Study
- P1500 Standardization
- Conclusions
Summary of Test Requirements

- Sockets and Test Access Mechanisms
- Bandwidth-Scalable Test Mechanism
- Test Solutions for Different Technology Types
- Automation of Core Test and SOC Test
- Different Core Testability Techniques
- Debug and Diagnosis
- Hierarchical Core Test Integration
SOC Test Architecture

- **Core or UDL**
  - random logic often with Test-Enabling (scan, test points)
  - memory, analog, & processor require no Test-Enabling

- **Source**
  - provide test stimulus from either on-chip or off-chip ATE

- **Sink**
  - provide test response to either on-chip or off-chip ATE

- **Socket**
  - isolation and access
  - connect core terminals to Test Access Mechanism

- **Test Access Mechanism**
  - transport patterns from source to core, from core to sink
SOC Test Architecture

S/S : Source/Sink
TAM: Test Access Mechanism
Legacy Core Socket

- Selectable Socket Elements Per-Pin
- Scan Chain Segmentation
- Socket Element Grouping
- Capture-By-Domain
Memory Socket

- XOR Tree for Address Observation (selectable width)
- XOR Tree for Control Observation (selectable width)
- Shared DataIn Observation with DataOut Control (per bit)
- Random Variable for Mux Select (bistOn signal)
Summary of Test Requirements

- Sockets and Test Access Mechanisms
- Bandwidth-Scalable Test Mechanism
- Test Solutions for Different Technology Types
- Automation of Core Test and SOC Test
- Different Core Testability Techniques
- Debug and Diagnosis
- Hierarchical Core Test Integration
Embedded ATE

Low-Cost External ATE

Embedded ATE
(on chip)
Pattern Generation
Result Compression
Precision Timing
Diagnostics
Power Management
Test Control
Support for
Board-Level Test
System-Level Test
(about 10k gates)

Memory
SRAM, DRAM, ROM, 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

At-Speed Test Using Only a Low-Cost Digital Tester with Limited Speed and Accuracy!

Reduced Pin-Count, Low Bandwidth External Interface

High-Bandwidth Internal Interfaces
Summary of Test Requirements

- Sockets and Test Access Mechanisms
- Bandwidth-Scalable Test Mechanism
- Test Solutions for Different Technology Types
- Automation of Core Test and SOC Test
- Different Core Testability Techniques
- Debug and Diagnosis
- Hierarchical Core Test Integration
Different Technology Types

Random Logic

Embedded Memories

Analog Logic

ADC

adcBIST

Test Access

© LogicVision, Inc., 1998, V 1.5 DB
System-on-Chip Test - P1500
Summary of Test Requirements

- Sockets and Test Access Mechanisms
- Bandwidth-Scalable Test Mechanism
- Test Solutions for Different Technology Types
- Automation of Core Test and SOC Test
- Different Core Testability Techniques
- Debug and Diagnosis
- Hierarchical Core Test Integration
Automation

- **Design Analysis and Specification**
  - Rules checking, default configurations
  - Flexibility based on test requirements
    - Area, coverage, performance, test autonomy, IP protection

- **Generation of Design Objects**
  - RTL Objects, scripts, high-level description
  - Modular interfaces (supports test hierarchy and scalability)

- **Assembly and Integration**
  - Supports test resource sharing

- **Verification and Test Data Generation**
  - Protocol Migration for core level to chip level
  - Supports design verification and manufacturing test

- **Diagnostic Test Data Generation**
  - Supports Manufacturing Test / Failure Analysis
Summary of Test Requirements

- Sockets and Test Access Mechanisms
- Bandwidth-Scalable Test Mechanism
- Test Solutions for Different Technology Types
- Automation of Core Test and SOC Test
- Different Core Testability Techniques
- Debug and Diagnosis
- Hierarchical Core Test Integration
Core Testability Techniques

1: Legacy Cores
   • Access and Isolation (socket)

2: ATPG Enabled
   • Scan Chain Insertion

3: Reusable ATPG
   • Access and Isolation (socket)
   • Test Pattern Generation (reusable)

4: BIST Enabled
   • Scan Chain and Test Point Insertion

5: Embedded BIST
   • BIST Controller Insertion
1: Legacy Cores

- **Analyze/Specify**
  - socket segments
  - socket type

- **Generate**
  - socket elements

- **Assemble**
  - socket core

- **Verify**
  - functional vectors through socket elements
2: ATPG Enabled

- Analyze/Specify
  - scan design rules
- Assemble
  - scan chains
- Verify
  - scan vectors through simulation
3: Reusable ATPG

- **Analyze/Specify**
  - core interface
- **Generate**
  - socket
- **Assemble**
  - scan chains
  - socket and core
- **Verify**
  - reusable scan vectors through simulation
4: BIST Enabled - Random Logic

- **Analyze/Specify**
  - scan design rules
  - test point candidates

- **Assemble**
  - scan chains
  - test points

- **Verify**
  - scan vectors through simulation
4: BIST Enabled - Memory

- **Analyze/Specify**
  - serial test or parallel test

- **Generate**
  - sockets

- **Assemble**
  - socket memories
  - route socket control

- **Verify**
  - memory BIST through simulation
5: Embedded Memory BIST

- **Analyze/Specify**
  - serial test or parallel test

- **Generate**
  - memory BIST controller
  - sockets

- **Assemble**
  - socket memories
  - instantiate controller

- **Verify**
  - memory BIST through simulation
2: ATPG Enabled
4: Memory BIST Enabled

- Analyze/Specify
  - scan design rules
  - serial or parallel memory test

- Generate
  - sockets

- Assemble/Generate
  - insert scan chains
  - socket memories

- Verify through Simulation
  - scan vectors
  - memory BIST
2: ATPG Enabled

5: Embedded Memory BIST

- Analyze/Specify
  - scan design rules
  - serial or parallel test

- Generate
  - sockets (collars)
  - memory BIST controller

- Assemble
  - insert scan chains
  - socket memories
  - instantiate BIST controller

- Verify through Simulation
  - scan vectors
  - memory BIST
3: Reusable ATPG
5: Embedded Memory BIST

- Analyze/Specify
  - core interface
- Generate
  - socket (wrapper)
- Assemble
  - insert scan chains
  - socket memories and RL Core
- Verify
  - simulate scan vectors
  - simulate memory BIST
4: Random Logic BIST Enabled
5: Embedded Memory BIST

- **Analyze/Specify**
  - scan design rules
  - serial or parallel test

- **Generate**
  - memory sockets
  - memory BIST controller

- **Assemble**
  - scan chains and test points
  - socket memories
  - instantiate BIST controller

- **Verify**
  - simulate scan vectors
  - simulate memory BIST
5: Embedded Random Logic BIST
5: Embedded Memory BIST

- **Analyze/Specify**
  - socket segments
  - socket type

- **Generate**
  - socket elements
  - BIST controllers
  - Embedded Test Manager

- **Assemble**
  - integrate design objects with core

- **Verify**
  - BIST through simulation
Summary of Test Requirements

- Sockets and Test Access Mechanisms
- Bandwidth-Scalable Test Mechanism
- Test Solutions for Different Technology Types
- Automation of Core Test and SOC Test
- Different Core Testability Techniques
- Debug and Diagnosis
- Hierarchical Core Test Integration
Logic BIST Diagnostics

**Step 1:**
Compare signatures on a vector by vector basis

**Step 2:**
Dump all flip-flop values of failing vector
Memory BIST Diagnostics

Parallel Status Lines

Stop-On-Nth-Error
Summary of Test Requirements

- Sockets and Test Access Mechanisms
- Bandwidth-Scalable Test Mechanism
- Test Solutions for Different Technology Types
- Automation of Core Test and SOC Test
- Different Core Testability Techniques
- Debug and Diagnosis
- Hierarchical Core Test Integration
Daisy Chain Control Mechanism

- Boundary Scan Chain
- Logic BIST
- ETM
- TAM
- BIST Enabled Core
- UDL
- Memory BIST
- Legacy Core
- Test Access Mechanism
- Logic BIST Manager
- Embedded Test Manager
- Test Access Mechanism
- 1149.1 TAP

© LogicVision, Inc., 1998, V 1.5 DB
System-on-Chip Test - P1500
Star Test Access Mechanism (TAM)
Daisy Chain TAM

- Logic BIST
- Embedded Test Manager
- Test Access Mechanism

Boundary Scan Chain

Socketed Core

UDL

BIST Enabled Core

Legacy Core

Memory BIST

ATPG Enabled Core

Logic BIST

ETM

TAM

1149.1 TAP

TDI

TCK

TMS

TRST

TDO

© LogicVision, Inc., 1998, V 1.5 DB
System-on-Chip Test - P1500
Contents

- Introduction
- SOC Test Requirements
- SOC Test Architecture
- SOC Test Methodology
- Case Study
- P1500 Standardization
- Conclusions
SOC Test Process

- UDL Test Enabling
- SOC Decomposition
- SOC Test Merger
- SOC Test Integrator
  - Core Test Augmentation
    - Wrapper
    - Sources/Sinks
    - TAM/ETM
  - Non-Merged Cores
  - Merged Cores
  - Core Test Enabling
    - Wrapper
    - Sources/Sinks
    - TAM/ETM
Contents

- Introduction
- SOC Test Requirements
- SOC Test Architecture
- SOC Test Methodology
- Case Study
- P1500 Standardization
- Conclusions
SOC Case Study

- Chip Contains:
  - 250K gates of digital logic
  - 2 distinct Cores, 5 additional Blocks
  - 29 embedded memories
- CORE 1(VD): 100 K gates, 10 embedded memories
- CORE 2(GPIO): Legacy Core, functional patterns exist
- BLOCK MIU: < 40 K gates, 10 embedded memories
- BLOCK TSD: < 20 K gates, 3 embedded memories
- BLOCK AO: < 20 K gates, 3 embedded memories
- BLOCK VO: < 20 K gates, 3 embedded memories
- BLOCK ICC: < 10 K gates
Set-Top-Box (JSAT) System

Tuner → QPSK → Viterbi Decoder → Read-Solomon

Descrambler → Transport Demux → MPEG Video → MPEG Audio

RAM ← CPU ← SDRAM

MB86372 MPEG2-Chip
MPEG2(MB86372)/JSAT3
Test Architecture

Introduction:
This presentation describes a test architecture proposal for the MPEG2/JSAT3 chip design. This design is a pilot project for validating concepts described in the Socket Program.

Objective:
The objective of the Socket Program is to demonstrate a SOC test methodology and to provide a vehicle to prove in silicon test techniques for core-based designs.
Pre-Test Inserted Chip Core
CORE 1 (VD): Memory BIST Insertion

- **Analyze/Specify**
  - Serial test or parallel test

- **Generate**
  - generate controller
  - generate sockets (collars)

- **Assemble**
  - replace RAM with socketed RAM
  - instantiate controller

- **Verify**
  - simulate memory BIST
CORE 1(VD): Scan/Test Point Insertion

- Analyze/Specify
  - scan design rules
- Generate
  - generate Test Points
- Assemble
  - insert scan chains
  - insert test points
- Verify
  - generate and simulate scan vectors
CORE 1(VD): Logic BIST Insertion

- **Analyze/Specify**
  - socket segments
  - socket type

- **Generate**
  - socket elements
  - Logic BIST controller

- **Assemble**
  - integrate design objects with core

- **Verify**
  - generate and simulate logic BIST vectors
CORE 2 (GPIO): Legacy Socket Insertion

- Analyze/Specify
  - socket segments
  - socket type

- Generate
  - socket elements

- Assemble
  - integrate core with socket elements

- Verify
  - retarget functional vectors through socket elements
BLOCK MIU: Memory BIST Insertion

- Analyze/Specify
  - Serial test or parallel test
- Generate
  - generate controller
  - generate sockets (collars)
- Assemble
  - replace RAM with socketed RAM
  - instantiate controller
- Verify
  - simulate memory BIST
BLOCK MIU: Scan/Test Point Insertion

- **Analyze/Specify**
  - scan design rules

- **Generate**
  - generate Test Points

- **Assemble**
  - insert scan chains
  - insert test points

- **Verify**
  - generate and simulate scan vectors
BLOCK TSD: Memory Collar Insertion

- Analyze/Specify
  - Serial test or parallel test
- Generate
  - generate sockets (collars)
- Assemble
  - replace RAM with socketed RAM
- Verify
  - simulate memory BIST
BLOCK TSD: Scan/Test Point Insertion

- **Analyze/Specify**
  - scan design rules
- **Generate**
  - generate Test Points
- **Assemble**
  - insert scan chains
  - insert test points
- **Verify**
  - generate and simulate scan vectors
BLOCK AO: Memory Collar Insertion

- **Analyze/Specify**
  - Serial test or parallel test

- **Generate**
  - generate sockets (collars)

- **Assemble**
  - replace RAM with socketed RAM

- **Verify**
  - simulate memory BIST
BLOCK AO: Scan/Test Point Insertion

- Analyze/Specify
  - scan design rules

- Generate
  - generate Test Points

- Assemble
  - insert scan chains
  - insert test points

- Verify
  - generate and simulate scan vectors
**BLOCK VO: Memory Collar Insertion**

- **Analyze/Specify**
  - Serial test or parallel test

- **Generate**
  - generate sockets (collars)

- **Assemble**
  - replace RAM with socketed RAM

- **Verify**
  - simulate memory BIST
BLOCK VO: Scan/Test Point Insertion

- Analyze/Specify
  - scan design rules
- Generate
  - generate Test Points
- Assemble
  - insert scan chains
  - insert test points
- Verify
  - generate and simulate scan vectors
BLOCK ICC: Scan/Test Point Insertion

- **Analyze/Specify**
  - scan design rules

- **Generate**
  - generate Test Points

- **Assemble**
  - insert scan chains
  - insert test points

- **Verify**
  - generate and simulate scan vectors
Core Flow: Memory BIST Insertion

- **Analyze/Specify**
  - Serial test or parallel test

- **Generate**
  - generate controller

- **Assemble**
  - instantiate controller
  - route control signals

- **Verify**
  - simulate
Core Flow: Scan/Test Point Insertion

- **Analyze**
  - merge core random logic
  - scan design rules

- **Generate**
  - generate Test Points

- **Assemble**
  - insert scan chains
  - insert test points

- **Verify**
  - generate and simulate scan vectors
Top Flow

- Analyze/Specify
  - Boundary Scan Cell types, segments
  - Test Schedule

- Generate
  - Boundary Scan elements
  - Logic BIST controller
  - Test Access Port

- Assemble
  - integrate design objects with core

- Verify
  - generate and simulate
Top Flow (cont’d)

Boundary Scan Chain

Core 2

ICC

Core 1

MIU

TSD

AO

VO

Memory BIST

Memory BIST

logic BIST controller

Chain Router

1149.1 TAP

TDI TCK TMS TDO

© LogicVision, Inc., 1998, V 1.5 DB
System-on-Chip Test - P1500
Contents

- Introduction
- SOC Test Requirements
- SOC Test Architecture
- SOC Test Methodology
- Case Study
- P1500 Standardization
- Conclusions
P1500 Standardization

- At-Speed Testing
- Diagnostics and Failure Analysis
- Test Reuse: Board, System, and Field
- Interface Standards: Embedded Test Manager
- Legacy Core Test Data Format (STIL?)
- On-Chip-Bus (OCB) for Test Access
- Routability, Signal Integrity, and Cross Talk
- BIST Friendly Architecture
Contents

- Introduction
- SOC Test Requirements
- SOC Test Architecture
- SOC Test Methodology
- Case Study
- P1500 Standardization
- Conclusions
Conclusions

- 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.
- Embedded ATE provides bandwidth scalability and at-speed testing.