Chapter 4

System/Network-on-Chip Test Architectures
What is this chapter about?

- Introduce basic and advanced architectures for:
  - System-on-Chip (SOC) Testing
  - Network-on-Chip (NOC) Testing
- Further focus on:
  - Testing on On-Chip Networks
  - Design and Test Practices in Industry
**Introduction to SoC Testing**

- SoC testing is a composite test comprised of individual tests for each core, user-defined logic (UDL) tests, and interconnect tests.
- To avoid cumbersome format translation for IP cores, SoC and core development working groups such as virtual socket interface alliance (VSIA) have been formed to propose standards.
- IEEE 1500 standard has been announced to facilitate SoC testing.
- IEEE 1500 specifies interface standard which allows cores to fit quickly into *virtual sockets* on SoC.
- Core vendors produce cores with an uniform set of interface features. SoC integration is simplified by plugging cores into standardized sockets.
Challenges of SoC Testing

- Generally, core users cannot access core net-lists and insert design-for-testability circuits. Core users rely on test patterns supplied by core vendors.
- Care must be taken to make sure that undesirable test patterns and clock skews are not introduced into test streams.
- Cores are often embedded in several layers of user-defined or other core-based logic, and are not always directly accessible from Chip I/Os.
- Test data at I/Os of an embedded core might need to be translated into a format for application to the core.
Mainly, three structural elements are required. They are test pattern source and sink, test access mechanism (TAM), and core test wrapper.
More Test Challenges

- Once test data transport mechanism (TAM) and test translation mechanism (test wrapper) are determined, major challenge for system integrator is test scheduling.

- Test scheduling must consider several conflicting factors: (a) SoC test time minimization, (b) resource conflicts due to sharing of TAMs and on-chip BIST engines, (c) precedence constraints among tests, and (d) power constraints.

- Finally, analog and mixed-signal core testing must be dealt with. Testing analog and mixed-signal cores is challenging because their failure mechanisms and test requirements are less known than digital cores.
Talk Outline for SoC Testing

- Introduction to testing
  - Motivation for modular testing of SOCs
- Wrapper design
  - IEEE 1500 standard, optimization
- Test access mechanism design and optimization
- Test scheduling
- Exploiting port scalability to test embedded cores at multiple data rates
  - Virtual TAMs
  - Matching ATE data rates to scan frequencies of embedded cores
- Conclusions
System Chips

~50 million transistors

1 cm

Viruses: Size 300 nm

Intel Itanium (2006): 1.7 billion transistors

EE Times:
Intel crafts transistor with 20-nm gate length
David Lammers, David Lammers
(06/11/2001)
Motivation for Testing: XBox 360 Technical Problems

- The "Red Ring of Death": Three red lights on the Xbox 360 indicator, representing "general hardware failure" (http://en.wikipedia.org/wiki/3_Red_Lights_of_Death)

- The Xbox 360 can be subject to a number of possible technical problems. Since the Xbox 360 console was released in 2005 the console gained reputation in the press in articles portraying poor reliability and relatively high failure rates.

- On 5 July 2007, Peter Moore published an open letter recognizing the problem and announcing 3 years warranty expansion for every Xbox 360 console that experiences the general hardware failure indicated by the three flashing red lights on the console.
XBox 360 Technical Problems (Cont’d)

- July 5, 2007, Xbox issues to cost Microsoft $1 billion-plus. Unacceptable number of repairs leads to company extending warranties.

- Matt Rosoff, an analyst at the independent research group Directions on Microsoft, estimates that Microsoft’s entertainment and devices division has lost more than $6 billion since 2002.
Testing Principles (2-minute primer)

- Screen defective chips (wafer, package)
- Stress test (burn-in)
- Diagnosis: Locate defects, yield learning

- Speed binning
- Design-for-testability (DFT) typically used
- Test generation, scan design

System-on-Chip Test Architectures
Motivation for Core-Based SOC Testing

- System-on-chip (SOC) integrated circuits based on embedded intellectual property (IP) cores are now commonplace
  - SOCs include processors, memories, peripheral devices, IP cores, analog cores
  - Low cost, fast time-to-market, high performance, low power
  - Manufacturing test needed to detect manufacturing defects

![Graph showing cost per transistor from 1980 to 2015](https://example.com/graph.png)

Manufacturing cost
Test cost
ITRS’03
### System-on-Chip (SOC)

- **I/O pads**
  - CPU core
  - Memory array
  - Legacy core
  - IP hard core

- **User-defined logic**
  - Self-test control
  - DSP core
  - Interface control
  - Embedded DRAM

- **1149.1 TAP controller**

- **Test access is limited**
- **Test sets must be transported to embedded logic**
- **High test data volume & test time**

**Philips Nexperia™ PNX8550 SOC:**
- 338,839 flip-flops, 274 embedded cores, 10M logic gates, 40M logic transistors!
Cost of Test

- “The emergence of more advanced ICs and SOC semiconductor devices is causing test costs to escalate to as much as 50 percent of the total manufacturing cost.” [Kondrat 2002]

- “As a result, semiconductor test cost continues to increase in spite of the introduction of DFT, and can account for up to 25-50% of total manufacturing cost”. [Cooper 2001]

- “Test may account for more than 70% of the total manufacturing cost - test cost does not directly scale with transistor count, dies size, device pin count, or process technology.” [ITRS’03]
Modular Testing

- Test embedded cores using patterns provided by core vendor (test reuse)
- Test access mechanisms (TAMs) needed for test data transport: TAMs impact test time and test cost
- Test wrappers translate test data supplied by TAMs
- TAM optimization, test scheduling, and test compression are critical
  - Test data volume and testing time in 2010 will 30X that for today’s chips [ITRS’05]
Test Planning

Optimizing Test Access to Cores and Scheduling Test Hardware

**Test hardware planning**

- Core import
- Core integration
- Test wrapper & TAM design
- Top-level DFT
  - Test control blocks
  - IEEE 1149.1

**Test software planning**

- Core test import
- Top-level ATPG
  - Glue logic, soft cores
  - Test wrappers
- Test scheduling
- Test assembly
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**

**Wrapper Modes:** (1) Normal; (2) Serial Test; (3) 1-N Test; (4) Bypass; (5) Isolation; (6) Extest [Marinissen 2002]
Wrapper Boundary Cells

Wrapper boundary input cell

- shift
- wci
- from chip
- from WSI/WPI
- clk
- to core
- to WSO/WPO

Wrapper boundary output cell

- shift
- wco
- from core
- from WSI/WPI
- clk
- to chip
- to WSO/WPO
Wrapper Usage
Wrapped Embedded Cores
Wrapper Operation Modes (I)

Normal Mode

Serial Bypass Mode
Wrapper Operation Modes (II)

Serial Internal Test Mode

Serial External Test Mode
Wrapper Operation Modes (III)

Parallel Internal Test Mode

Parallel External
Test Wrapper Optimization

Priority 1: Balanced Wrapper Scan Chains

Minimize length of longest wrapper scan in/out chain

System-on-Chip Test Architectures
Reducing TAM Width

Priority 2: Minimize wrapper scan chains created

Scan chain – 32 FF

4 Wrapper scan chains

Scan chain – 32 FF

2 Wrapper scan chains
Two-Priority Wrapper Design Algorithm

1. Minimize length of longest wrapper scan in/out chain
2. Minimize number of wrapper scan chains

Design_wrapper algorithm uses the BFD heuristic for Bin Design
**Test Access Mechanisms**

**Types of TAMs**

- Multiplexed access [Immaneni, ITC’90]
- Reuse system bus [Harrod, ITC’99]
- Transparent paths [Ghosh, DAC’98]
- Isolation rings [Whetsel, ITC’97]
- Test Bus [Varma, ITC’98]
- Test Rail [Marinissen, ITC’98]
Test Bus Architecture

- Combination of multiplexing and distribution
- Supports only serial schedule
- Core-external testing is cumbersome or impossible
TestRail Architecture [Goel ITC ’02]

- Combination of Daisychain and Distribution architectures
- Cores connected to a TestRail can be tested simultaneously as well as sequentially
- Multiple wrappers can be activated simultaneously for Extest
- TestRails can be either fixed-width or flexible-width

**Fixed-width TestRails**

- Cores connected to a fixed-width TestRail can be tested simultaneously.

**Flexible-width TestRails**

- Multiple cores can be tested simultaneously due to the flexible-width configuration.

Ch. 4 – SOC and NOC Testing - P. 30
Step-by-Step Approach to Wrapper/TAM Co-optimization

1. $P_W$: Wrapper design
2. $P_{AW}$: Core assignment + $P_W$
3. $P_{PAW}$: TAM width partitioning + $P_{AW}$
4. $P_{NPAW}$: Number of TAMs + $P_{PAW}$
Mathematical Programming Model for TAM Partitioning

- Variable $x_{ij} = 1$, if core $i$ assigned to TAM $j$
- Testing time of core $i$ on TAM width $w_j = T_i(w_j)$
- Testing time on TAM $j = \sum_i T_i(w_j) \times x_{ij}$
- **Objective:** Minimize $T = \max_j \sum_i T_i(w_j) \times x_{ij}$
- **Constraints**
  1. $\sum_i x_{ij} = 1$, every core connected to exactly one TAM
  2. $\sum_i w_j = W$, total TAM width is $W$
  3. $w_j \leq w_{\text{max}}$, maximum width of any TAM is $w_{\text{max}}$
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 and
- *Testing time is minimized*
Architectures Determine Schedules

[Goel ’03]

Slide provided by Erik Jan Marinissen, Philips Research Labs
Rectangle Model for Test Buses

Three test buses
Each core on same bus gets equal, fixed TAM width

Bus 1
Core 1
Core 3
Core 9
Core 8

Bus 2
Core 2
Core 4

Bus 3
Core 5
Core 6
Core 7
**Test Scheduling**

- Test scheduling determines sequence of core tests on the TAMs
- Avoid test resource conflicts
- Minimize testing time
- Ineffective scheduling can increase tester data volume: *Idle bits*

![Diagram](image_url)

System-on-Chip Test Architectures
Rectangle Representation

- Testing time $T_i(w_j)$ for Core $i$ and TAM width $j$
- Rectangle $R_{ij}$
- Set of rectangles $R_i$ for each core
- Collection of rectangles $R$ for SOC
Rectangle Packing Problem

- Given collection \( R \) of rectangle sets for the SOC cores,
- Select one rectangle \( R_{ij} \) for each Core \( i \)
- Pack the selected rectangles into a bin of fixed height,
- Such that bin width is minimized

![Diagram of rectangle packing problem]

Collection \( R \)

- Core 1
- Core 2
- Core 3

System-on-Chip Test Architectures

Ch. 4 – SOC and NOC Testing - P. 38
Packed Bin = TAM Design + Test Schedule

- Bin height = total TAM width
- Bin width = SOC testing time
- Rectangle area = tester memory for core test
- Empty space = wasted tester memory

System-on-Chip Test Architectures
Preferred TAM Widths

- Only Pareto-optimal TAM widths are considered
- **Procedure:** Tests are scheduled at current time in decreasing order of preferred TAM width until no TAM width remains
Non-Preferred Rectangles: Fill Idle Time

- Core 1-P
- Core 2-P
- Core 3-P
- Core 2
- Core 1

Current_time

Next_time

Total TAM width
Increasing Current TAM Widths

- Modify current rectangle that will benefit the most from an increase in TAM width

If idle time is inevitable, advance Current_time and repeat procedure from the start
Current-Generation ATEs

- Port scalability features
- Digital speeds of up to 2.5 Gbps
- Application flexibility

Every port of a tester, consisting of multiple channels, can be configured at a desired data rate.
Virtual TAMs

- Embedded core test frequency is limited by scan frequency
  - Scan frequencies are low to meet power, routing, and clock skew constraints
- Virtual TAMs allow use of high frequency ATE pins
- How can we match fast ATE data rates to slow scan frequencies?
Bandwidth Matching

ATE pins: $W_{ATE} = 4$

ATE frequency factor: $n = 4$

High frequency pins $U = 2$

$$U \times f_{ATE} + (W_{TAM} - U) \times f_{TAM} = U \times n \times f_{TAM} + (W_{ATE} - U) \times f_{TAM}$$
**Implementation of Bandwidth Matching**

Diagram showing connections between ATE, SOC, Embedded core, and Parallel-In/Serial-out Registers. The diagram illustrates the implementation of bandwidth matching using Low-speed and High-speed TAMs. 

- **ATE** connected to Low-speed TAM
- **SOC** with Serial-In/Parallel-out Registers
- **Embedded core**
- **Parallel-In/Serial-out Registers**

**System-on-Chip Test Architectures**
Selection of $U$ and $n$

- Testing of SOC is often dominated by the testing time of bottleneck cores.
- Testing time of SOCs containing bottleneck cores does not decrease for TAM widths greater than $W^*$.
- The lower bound on test time in such SOCs is $T^*$ corresponding to TAM width $W^*$.
### SOCs with Bottleneck Cores

<table>
<thead>
<tr>
<th>SOC</th>
<th>$W^*$ (bits)</th>
<th>$T^*$ (clock cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>u226</td>
<td>48</td>
<td>5333</td>
</tr>
<tr>
<td>d281</td>
<td>48</td>
<td>3926</td>
</tr>
<tr>
<td>g1023</td>
<td>40</td>
<td>14794</td>
</tr>
<tr>
<td>p34392</td>
<td>36</td>
<td>544579</td>
</tr>
<tr>
<td>t512505</td>
<td>36</td>
<td>5228420</td>
</tr>
<tr>
<td>h953</td>
<td>16</td>
<td>119357</td>
</tr>
<tr>
<td>f2126</td>
<td>16</td>
<td>335334</td>
</tr>
<tr>
<td>q12710</td>
<td>16</td>
<td>2222349</td>
</tr>
</tbody>
</table>
Relationship of U, n and W*

- U and n should be chosen such that total virtual TAM width W does not exceed W*

\[
W \leq W^* \\
W = nU + (W_{ATE} - U) \\
U(n - 1) \leq W^* - W_{ATE}
\]
Variation of $U$ with $n$

\[ U(n-1) = W^* - W_{ATE} \]
U vs n for ITC ’02 Benchmarks

(a) SOC p34392

(b) SOC h953

(c) SOC d281

(d) SOC g1023
Multiple-Speed TAM Architectures

- Exploit port-scalability of ATEs
- Facilitate efficient use of high data-rate tester channels
- Unlike virtual TAMs, avoid on-chip hardware overhead
- Reduce testing time of bottleneck cores
Problem Formulation

- Dual-speed optimization problem

Given:

Determine the wrapper design, TAM width and test data rate for each core, and the SOC test schedule such that:

- the total number of TAM wires utilized at any moment does not exceed $W$
- the number of TAM wires driven at the high data rate does not exceed $V$
- the SOC testing time is minimized
### Selection of Data Rate for a Core

**Core 5 in SOC p93791**

<table>
<thead>
<tr>
<th>Data Rate</th>
<th>Delay Time (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f=2$</td>
<td>V=10</td>
</tr>
<tr>
<td></td>
<td>$T = 14026.9 \mu s$</td>
</tr>
<tr>
<td>$f=1$</td>
<td>$W-V=23$</td>
</tr>
<tr>
<td></td>
<td>$T = 11398.9 \mu s$</td>
</tr>
</tbody>
</table>
Matching Core Scan Frequencies to ATE Data Rates

Baseline Case 1

Core A, Core B, Core C, Core D

- $f_1 = 40\text{MHz}$
- $f_2 = 40\text{MHz}$
- $w_1 = 8$
- $w_2 = 2$
- $T = 456\ \mu\text{s}$

Test time: $T = 456\ \mu\text{s}$
Matching Core Scan Frequencies to ATE Data Rates

Baseline Case 2

\[ T = 275 \, \mu s \]

Core A
Core B
Core C
Core D

\[ f_1 = 80MHz \]

\[ f = 40MHz \]

\[ w_1 = 8 \]

\[ w_1 = 2, \quad f_2 = 40MHz \]

Test time
Matching Core Scan Frequencies to ATE Data Rates

- Core A
- Core B
- Core C
- Core D

$f = 40\text{MHz}$

$T = 246 \mu s$

$w_1 = 5$

$f_1 = 80\text{MHz}$

$w_1 = 5$

$f_2 = 40\text{MHz}$

Test time
Problem Statement

Given
- Test data parameters for $N$ embedded cores
- Maximum scan frequency $f_i^*$ for each core $i$
- SOC-level TAM width $W$

Determine
- The number of TAM partitions $B$
- Width $w_j$ and scan frequency $f_j$ of each TAM partition $j$
- Assignment of cores to TAM partitions

Such that
- TAM frequency does not exceed the maximum scan frequency of any core assigned to that TAM partition
- The overall test time is minimized
- The sum of the widths of all the TAM partitions does not exceed $W$
Solution Techniques

- Lower bound on test time based on geometric arguments (rectangle packing)
- Integer linear programming
  - Exact optimization method, limited to small problem instances
- Fast heuristic method
  - Scalable, close to optimal results
Comparison with Baseline

p22810
(5 frequencies: 10 to 50 MHz)

Test time ($\mu$s)

TAM Width

LB
baseline
proposed

37%
Comparison with Exact Method and Baseline

d695

(2 frequencies: 40 MHz and 50 MHz)

Test time (μs)

TAM Width

ILP
baseline
proposed

(X 100)
Conclusions

- Test reuse, test time minimization, and test compression are necessary to reduce test cost for SOCs.
- Wrapper/TAM optimization and test scheduling can reduce test time for core-based SOCs.
- Virtual TAMs offer several advantages for SOC testing:
  - On-chip TAM wires are not limited by the number of available pins on the SOC.
  - Better utilization of high-speed ATE channels reduces testing times.
- TAM architectures can match port-scalable ATE channels to different scan frequencies of embedded cores.
Introduction to Network-On-Chip Testing

- For future SoCs with large number of cores and increased interconnect delay, traditional point-to-point or bus-based communication architecture becomes new bottleneck.

- Traditional communication architectures cannot meet system requirements of bandwidth, latency, and power consumption.

- Integrated switching network has been proposed as an alternative approach to interconnect cores in SoC.

- Such networks rely on a scalable and reusable communication platform, called network-on-chip (NoC) system, to meet two major requirements: reusability and scalable bandwidth.
The figure shown below represents a 2-D mesh NoC.

- Cores are connected to NoC by routers or switches.
- Data are organized by packets and transported through interconnection links.
- Various network topologies and routing algorithms can be used to meet requirements of performance, hardware overhead, power consumption.
The greatest difference between NoC testing and SoC testing is on test access mechanism design.

On-chip-network of a NoC can be reused as a TAM for test packet delivery. Theoretically, no TAM interconnects are required to be invested.

Test time can be reduced by network reuse even under power constraints, with minimized pin count and area overhead.

Generally, more cores can be tested in parallel than TAM-based SoC testing, due to large NoC channel bandwidth.
Talk Outline for Testing Embedded Cores in NoC

- Reuse of On-Chip Network for Testing
- Test Scheduling
- Test Access Methods and Test Interface
- Efficient Reuse of Network
- Power-Aware and Thermal-Aware Testing
Network-on-Chip

Current Design Methodology: System-on-Chip (SoC)

Interconnection schemes:

- Shared bus
  - Core A
  - Core B
  - Core C
  - Core D

- Dedicated connection
  - Core A
  - Core B
  - Core C
  - Core D
Need for Network-on-Chip (NOC)

Current Design Methodology: System-on-Chip (SoC)

Design
- Communication infrastructure is becoming new bottleneck
  - Wire delay
  - Signal integrity
  - Power dissipation
  - Area vs. speed
- New interconnection schemes needed.

Test
- Test of SoC has been well understood
  - TAM, wrapper
  - Test scheduling
  - IEEE 1500
- Test needs dedicated hardware
- Hardware for mission-mode communication can not be reused for testing
NOC-based System

tester

SoC

core

core

SoC

core

core

core

System-on-Chip Test Architectures
### Possible next-generation SoC paradigm: Network-on-Chip (NoC)

**Design**
- High performance
  - High bandwidth
  - Low signal delay
- Reasonable overhead
- Suitable for large number of cores
- Network design is versatile
- Methodology of next generation VLSI design

**Test**
- Test of NoC has not received much attention
  - Core testing
  - Router and interconnection testing
  - Test wrapper design
  - Test scheduling
- No need for dedicated TAMs
- Network can be reused for testing
NoC-based System

- Packet-switching
- Bidirectional channel
- 2-D mesh, XY routing
- Channels, routers used as TAM
- Input/output ports associated with cores
- Ports, channels are assigned a time tag

d695 from ITC’02 benchmark
Test Scheduling Using Dedicated Routing Path: Non-preemptive

- Each core is associated with a routing path
- All resources are reserved until test completed
- Test pipeline maintained
- No complex logic
- Similar to a circuit switching
- Efficiently assign I/Os and channels to core
Test Scheduling: Problem Formulation

How to assign I/Os and channels to each core for testing such that the overall test time is minimized?

In an NoC system using dedicated routing path, given $N_C$ cores, $N_I$ inputs, $N_O$ outputs, routing algorithm and the network topology, determine an assignment of cores to input/output pairs and a schedule such that the total test time is minimized.

- Equivalent to the resource-constrained multi-processor scheduling problem
- If the number of input/output pairs $\geq 2$, NP-complete
Test Scheduling: Optimal Solution Using ILP

- Problem can be solved exactly using an ILP model
- Large number of non-zero constraints
- CPU time is prohibitive

- Can be simplified using enumeration
- Enumerate the assignment of cores to I/O pairs
- Number of constraints reduced
- A few seconds for small instances with smaller number of I/Os

- For large instances, or larger number of I/Os, CPU time is still prohibitively high
- Not suitable for large systems
**Test Scheduling: Heuristic Algorithm**

- Sort cores and I/O pairs in decreasing order of testing time
- Permute cores and I/O pairs
- Assign cores with higher priority to free I/O pairs
- Check resource conflicts using time tag: I/Os, channels, cores
- Complexity: $O(N_C^M)$
- CPU time: a few minutes for all benchmarks
Test Access Method and Test Interface

Problems targeted:

- Test access scheme for testing routers at NoC level
- Possible hardware overhead
- Efficient test scheduling that can handle both routers and embedded functional cores
Test Access Method

Reuse the network resources for test access.
Test data and test control delivered in packet.
Responses can be processed by ATE or on-chip.
Test Responses

Can be handled on-chip

Minimum overhead
Probability of aliasing

Similar to prior work
Faster, with aliasing
Test Wrapper

On top of the 1500 compliant wrapper
Can wrap both router and core
Packing/unpacking mechanism reused from mission mode
Test Wrapper

From adjacent cores

Unpacking

Router

Pack ing

Core

To adjacent cores

Mission mode
Integrated Test Scheduling

Based on network reuse and dedicated routing path
- Permute cores in the order of test time
- Permute all input/output pairs
- For each permutation
  - Find free I/O pair
  - Check for resource conflicts
  - schedule a core
- Routers on a path should be all tested before functional cores on that path to be tested
- Routers can be tested concurrently with cores
- At least one I/O pair should be used for router testing at any time
After these routers are tested, one of the two I/O pairs can be used for core testing.
### Efficient Channel Width Utilization

Fixed channel width, not fully utilized

<table>
<thead>
<tr>
<th>Cores</th>
<th># of packets</th>
<th>Channel width = 16</th>
<th>Channel width = 32</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>flits/packet</td>
<td>test cycles</td>
<td>flits/packet</td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td>2</td>
<td>38</td>
</tr>
<tr>
<td>2</td>
<td>146</td>
<td>13</td>
<td>1029</td>
</tr>
<tr>
<td>3</td>
<td>150</td>
<td>32</td>
<td>2507</td>
</tr>
<tr>
<td>4</td>
<td>210</td>
<td>54</td>
<td>5829</td>
</tr>
<tr>
<td>5</td>
<td>220</td>
<td>109</td>
<td>12192</td>
</tr>
<tr>
<td>6</td>
<td>468</td>
<td>50</td>
<td>11978</td>
</tr>
<tr>
<td>7</td>
<td>190</td>
<td>43</td>
<td>4219</td>
</tr>
<tr>
<td>8</td>
<td>194</td>
<td>46</td>
<td>4605</td>
</tr>
<tr>
<td>9</td>
<td>24</td>
<td>128</td>
<td>1659</td>
</tr>
<tr>
<td>10</td>
<td>136</td>
<td>109</td>
<td>7568</td>
</tr>
</tbody>
</table>
Utilization of Idle Channel Width

Variable on-chip test clocks

- Use *faster* wrapper test clocks on cores with idle channel width
- Channel width $w$, wrapper scan chain $w'$, $n$ flits can be transported in parallel to core in one clock

$$n = \left\lfloor \frac{w}{w'} \right\rfloor$$

- Additional cores can be selected to further reduce test time
Utilization of Idle Channel Width

Slower tester clock

Test data

faster on-chip clock

Network channel

System-on-Chip Test Architectures  Ch. 4 – SOC and NOC Testing - P. 85
Variable on-chip test clocks

- Use *slower* wrapper test clocks on cores with high power dissipation
- No change on wrapper design
- Physical channel is viewed as \( n \) virtual channels

Channel Width Utilization Under Power Constraints

![Diagram showing channel width utilization with different test clocks on cores](image-url)
Power-Aware Test Scheduling

Variable on-chip test clocks in NoC-based system

- $N$ cores, tester clock $f_T$
- Faster on-chip clocks $2f_T$, $3f_T$, …
- Slower on-chip clocks $f_T/2$, $f_T/3$, …
- Determine a clock for each core, such that
  - No network resource conflicts
  - System test application time is minimized
  - Power constraints are not violated
Power-Aware Test Scheduling

- Each core associated with a set of on-chip clocks \( \{ \ldots 3f_T, 2f_T, f_T, f_T/2, f_T/3, \ldots \} \)
- Each clock corresponds to a power \( P(i,j) \), and the corresponding test time \( T(i,j) \)
- Selection of clock for each core controlled by a priority calculated from \( \Delta P/\Delta T \)
- More than one cores use slower clocks to utilize virtual channels
- Use dedicated routing path
- Power constraints are evaluated
Thermal-Aware Test Scheduling

High power density causes hot spots

- Existence of hot spots may increase test time because of thermal unbalance
- Layout redesign is impossible
- Layout not optimized for test
- Higher power generation
- Larger thermal variation
- Removal of hot spots can lead to thermal balance and reduced test time
Variable Clocking in Test Session

- Still rely on using multiple variable clocking for thermal management
- Clock assigned to each core can be varied during test application
- A more flexible scheme
- More efficient thermal management
- Extra test control
Variable Clocking in Test Session

Thermal safe constraints are not violated
Test time reduced

t_2 < t_1
Variable Clocking in Test Session

Thermal safe constraints guaranteed
Test time not compromised
Clock Selection

Unpack reused
Test control can be carried in packet
Clock varies only when the test of a core finished or started
Problem Formulation

- Test set information of core set $C$
- $N_C$ cores, $N_I$ inputs, $N_O$ outputs,
- Set of on-chip variable-rate clock $CLK$
- Set of thermal parameters $P_{thermal}$
- Chip floorplan, and maximum temperature $T_{TH}$

Determine: (1) clock variation of each core during test application, (2) test scheduling of cores on I/Os and channels, such that:

- Test application time is minimized
- Maximum temperature not over $T_{TH}$
Talk Outline for On-Chip Network Testing

- Testing of interconnect infrastructures [Grecu 2006]
- Testing of routers [Amory 2005]
- Testing of network interfaces and integrated system testing [Stewart 2006]
- Unless on-chip network of an NoC has been completely tested, it cannot be used to test the embedded cores.
Interconnect testing has been discussed in many papers.

This discussion is mainly based on the well-known maximal aggressor fault (MAF) model.

Apply identical transitions to all wires except the victim line to create maximal integrity loss in the victim line.

Contains six crosstalk errors in victim line: rising/falling delay, positive/negative glitch, and rising/falling speed-up.

For an interconnect structure with N lines, totally 6N faults are to be tested using 6N two-vector test patterns.
**Self-Test Structure**

- A pair of test data generator (TDG) and test error detector (TED) is inserted to each set of interconnects between two routers (switches).

- This is called *point-to-point* MAF self-test.
- Test patterns are launched before line drivers, and sampled after receiver buffers.
- Highly parallel testing if power consumption is within the power budget.
Test Application by Unicast

- MAF test patterns can be broadcast to all interconnects by test packets with only one TDG.

Only one set of interconnects between a pair of routers can be tested for each test pattern broadcast.

- A global test controller (GTC) and many TEDs are required.
Test Application by Multicast

- Test packets are broadcast to interconnects of different pairs of routers to achieve maximum parallelism.

- Multicast is a good compromise between test application time and hardware overhead.

- Point-to-point (unicast) test method has the smallest (largest) test application time but the largest (smallest) hardware overhead.
Testing of Routers

- Routers are used to implement functions of flow control, routing, switching and buffering of packets.

- Router testing can be treated as sequential circuit testing by taking its special property of regularity.

- Test pattern broadcasting can be applied to reduce test time.
Testing A Router

- Testing a router consists of testing the control logic (routing, arbitration, and flow control modules) and first-in first-out (FIFO) buffers.

- Control logic can be tested by typical sequential circuit testing methods such as scan testing.
- A smart way to test FIFO is to configure the first register of FIFO as scan register, and others can be tested by the scan register.
Testing All Routers

- Since all routers are identical, all can be tested in parallel by test pattern broadcasting.

- Comparator is implemented by XOR gates. It can also support diagnosis.
**Router Test wrapper Design and Test**

- IEEE-1500 compliant test wrapper is designed to support test pattern broadcasting and test response evaluation.

Modifications required for the test wrapper

---

**System-on-Chip Test Architectures**  
**Ch. 4 – SOC and NOC Testing - P. 103**
For example, all SC1 chains of these routers share the same set of test patterns.

Similarly, all Din[0] (i.e., Din-R0[0], ..., Din-Rn[0]) data inputs of these routers share the same set of test patterns.

The wrapper also supports test response comparison for scan chains and data outputs.

Diagnosis control block can activate diagnosis.

Small hardware overhead (about 8.5%) and small number of test patterns (several hundreds) due to test broadcasting. Small test application time (several thousands test cycles) using multiple, balanced scan chain and test broadcasting. *The method is scalable.*
Network Interface Testing

- Network interface (NI) is used to receive data bits from its corresponding IP core (router), packetize (de-packetize) the bits, and perform clock domain conversions between the router and the core.

- NI might be the most difficult to test component in an on-chip network, because clock domain conversion introduces non-deterministic device behavior.

- Current test methods rely on deterministic stored responses.

- The following discussion mainly based on functional test method, though new structural test solutions must be developed soon.
A NI Functional Test Model

- The NI of AEthereal NoC architecture.

- **Master-controller** (IP masters initiate transactions by issuing requests); **slave-controller** (IP slaves receive and execute transactions); **multicast connection** (one master, multiple slaves, all slaves executing each transaction); **narrowcast connection** (one master, multiple slaves, a transaction executed by only one slave).
NI Functional Fault Representation

- NI faults in AEthereal can be represented with four-tuple \( NI(c_1, c_2, o_1, o_2) \) where:
  - \( c_1 \): ID of NI under test,
  - \( c_2 \): whether the NI under test is a source (S) or destination (D),
  - \( o_1 \): transmission mode (BE or GT) of NI,
  - \( o_2 \): connection type (U, N, M) of NI.

- Notation: BE – best effort, GT – time guarantee, U – unicast, N – narrowcast, M – multicast. Note that \( o_1 \) and \( o_2 \) are optional.

- Each NI must be tested based on different combinations of these tuples.
Number of Functional Faults

- For each NI represented by NI(ID, c2, o1, o2), it must be tested as a source (master) and as a destination (slave). In each case, the NI must be tested with both BE and GT transmission modes. So, four faults must be considered.

- Two additional tests are required to test narrowcast (N) and multicast (M) for the NI. Totally, six faults must be dealt with for thoroughly testing each NI.

- Unicast (U) is not required to be added, because it has been applied during the first four faults.

- By following the same process, ten functional faults can be identified for each router.

- Test patterns must be generated to detect all six (ten) faults for each NI (router).
Test Scheduling for Functional testing

- It is important to develop an efficient method that can generate test patterns shared for NI faults and router faults.
- Initially, a preprocessing step is used to broadcast data packets (GT data and BE data) from I/O pins to local memory of each core.
- During test phase, an instruction packet is sent from input port of the NoC to the source router by GT transmission mode.
- Instruction packet contains information of destination core, transmission path, time at which test pattern application should take place.
- Destination node generates a signature packet.
Notes for NoC Functional Testing

- Functional testing for NI is not sufficient, and efficient structural test methods must be investigated.

- Testing NoC-based system by separating core testing from on-chip network testing is inadequate.

- Interactions between cores and on-chip network must be tested using extensive functional testing.

- Interactions between on-chip network components (routers, interconnects, and NIs) must be thoroughly tested by functional testing as well.
Talk Outline for Design and Test Practices

- SoC testing for PNX8550 system chip [Goel 2004].
- NoC testing for high-end TV system [Steenhof 2006].
Case Study: Soc Testing for PNX8550 System Chip

- PNX8550 is a chip designed based on Nexperia digital video platform by Philips [Goel 2004].
- Fabricated using 0.13um process, six metal layers, with 1.2V supply voltage.
- Entire chip contains 62 logic cores (5 hard, 57 soft), 212 memory cores, and 94 clock domains.
- Five hard cores: one MIPS CPU, two TriMedia CPUs, a custom analog block (PLLs and DLLs), and a D-to-A converter.
- All 62 logic cores are partitioned into 13 chiplets.
- Each chiplet is a group of cores placed together, and is connected to a specific set of TAM wires.
Structure of PNX8550

- Nexperia home platform
PNX8550 Structure and Test Methods

- Two device control and status (DCS) networks enable each processor to observe on-chip modules.
- A bridge is used to allow both DCS networks to communicate.
- Soft logic cores include MPEG decoder, UART, PIC 2.2 bus interface, etc.
- CPUs and many modules have access to external memory via a high-speed memory access network.
- PNX8550 allows test reuse through test wrappers (TestShell), and test access mechanism (TestRail).
- Test methods: random logic – full scan test with 99% stuck-at fault coverage, small embedded memories – scan test, large memories – BIST.
PNX8550 Test Strategies

- There are 140 TAM wires (i.e., 280 chip pins) for the entire chip.
- Design issue: how to assign these TAM wires to different cores and how to design the wrapper for each core.
- Requirement: each channel must provide 28M of test data volume and test application time must be minimized.
- Philips developed a tool called TR-ARCHITECT to deal with these core-based testing requirements.
- TR-ARCHITECT supports three test architectures: daisy chain, distribution, and hybrid (of daisy chain and distribution).
TR-ARCHITECT Inputs

- Requires two different kinds of inputs: SoC data file and a list of user options.

- SoC data file: SoC parameters such as number of cores in the SoC, number of test patterns and number of scan chains in each core.

- User options: test choices such as number of SoC test pins, type of modules (hard or soft), TAM type (test bus/test rail), architecture type (daisy chain, distribution, or hybrid), test schedule type (serial or parallel for daisy chain), and external bypass per module (yes/no).
TAM Wires Distribution and Test Architecture

- Distribution of 140 TAM wires to 13 chiplets is done *manually*, because TR-ARCHITECT became available halfway of PNX8550 design process.
- Assignment of TAM wires for a chiplet ranges from 2 to 21.
- Next step is to design the test architecture inside each chiplet.
- Distribution test architecture is used for all except two chiplets: UMDCS and UTDCS.
- For these two chiplets (hybrid test architecture), some wires are shared by two or more cores using daisy chain; some cores are connected by distribution architecture.
Test Architecture Design for Each Chiplet

- Test architecture design is trivial if chiplet under consideration has only one core. Test wrapper of the core can be designed based on TAM wires assigned and core parameters.

- For a chiplet containing multiple cores and using distribution test architecture, TR-ARCHITECT determines the number of TAM wires assigned to each core and design the test wrapper for the core.

- For both chiplets with hybrid test architecture, TR-ARCHITECT determines the number of TAM-wire groups, the width assigned to each group, assignment of cores to each group, and design the test wrapper for each core.
TR-ARCTITECT Major Procedures

- There are four major steps: create-start-solution, optimize-bottom-up, optimize-top-down, reshuffle.

- Create-start-solution: assign at least one TAM wire for each core.

- If there are cores left unassigned, they are assigned to least occupied TAMs.

- If there are TAM wires left unassigned, they are added to the most occupied TAMs.

- Optimize-bottom-up: merge the TAM (maybe several wires) with shortest test time with another TAM, such that wires free up in this process can be used for overall test reduction.
Example for Optimize-bottom-up

- TAM-1 has three wires with 500 test cycles for Core-1.
- TAM-2 has four wires with 200 test cycles for Core-2.
- TAM-3 has two wires with 100 test cycles for Core-3.
- Core-1 is the test bottleneck and number of total test cycles is 500.
- Merge Core-3 to TAM-2, and number of overall test cycles for Core-2 and Core-3 is 300 (by assumption), still smaller than 500.
- Two wires freed up by TAM-3 can be added to TAM-1 to reduce number of Core-1 test cycles from 500 to 350 (by assumption).
- Finally, number of overall test cycles can be reduced from 500 to 350.
TR-ARCHITECT Major Procedures and Results

- Optimize-top-down and Reshuffle follow the same idea and can be found in [Goel 2002].
- Each of the four procedures requires information of wrapper design and test time for each assignment of TAM wires, which can be provided by [Marinissen 2000].
- By manually assigning 140 TAM wires to 13 chiplets, total test time is dominated by UTDCS with 3,506,193 test cycles.
- If these 140 TAM wires are distributed to 13 chiplets by TR-ARCHITECT and hybrid test architecture is used, total test time is reduced to 2,494,687 test cycles (dominated by UMCU). Note: UTDCS is assigned three more TAM wires by TR-ARCHITECT, and changed to be non-dominant.
Case Study: NoC Testing for High-End TV Companion Chip by Philips

- The following figure outlines a high-end TV system with two chips: main chip (PNX8558 discussed above), and companion chip (implementing more advanced technologies that will not be released to competitors) [Steenhof 2006].
Main TV Chip and Companion Chip

- Main TV chip (PNX 8550 discussed in SoC testing case study) controls entire system and interacts with users, TV sources, TV display, peripherals, and configuration of companion chip.
- Companion chip contains nine IP blocks for enhancing video quality.
- Main and companion chips have their own dedicated interconnect structures. They are connected using a high-speed external link (HSEL).
- Idea of partitioning a complex system into main and companion chips has many advantages: reducing development risk, managing different innovation rates in different market segments, encapsulating different functionality.
System Tasks

- Functionality of whole system contains several hundreds of tasks controlled by main chip.
- Dash lines in following figure represent a task involving 11 IP blocks in main, companion chips and two memories. Notation: I (input), O (output), H (horizontal scaler), C (control processor).
On-chip network of companion chip contains routers (R), interconnects, and network interface (NI). Each NI contains one kernel (K), one shell (S), and several ports. Mainly, it is a 2x2 mesh NoC.
Companion Chip - NoC Implementation (Contd.)

- Numbers of master (M) and slave (S) ports are indicated in each NI.
- Ports are connected to IPs of microprocessors, DSPs, or memory arrays. New HSEL is used to attach another companion chip (e.g., FPGA).
Test Methods for Philips’ AEthreal NoC

- Test methods for Philips’ AEthreal NoC architecture can be found in [Vermeulen 2003].
- On-chip network can be treated as a core for testing.
- Knowledge about on-chip network can be used to enhance standard core-based test approach to get better results. For example, routers can be tested by test broadcasting, while test responses can be compared to each other.
- Timing test is extremely important because: (1) long wires in NoC may cause crosstalk errors, and (2) clock boundaries between cores are in NIs and timing errors can occur.
- Long wire testing can be dealt with by [Grecu 2006], but point (2) is still waiting for a good solution.
Test Methods for Philips’ AEthreal NoC (Contd.)

- Once on-chip network has been fully tested, it can be used to transfer data for core testing.

- No TAM wires are required for testing, and NoC is fully reused for core testing.

- NoC structure also supports parallel testing if channel capacity can support parallel data transportation with a specific power budget.
**Concluding Remarks**

- State-of-art techniques for SoC testing have been described.
- Modular test techniques for digital, mixed-signal, and hierarchical SoCs must be developed further to keep pace with technology advances.
- Test data bandwidth needs for analog cores are very different from digital cores, and unified top-level testing of mixed-signal SoCs remains a major challenge.
- Research is also needed to develop wrapper design techniques and test planning methods for multi-frequency core testing.
- Revolutionary RF interconnect technology might emerge to address future SoC testing.
Concluding Remarks (Contd.)

- Advances in testing NoC-based systems have been discussed.
- Key point: how to utilize on-chip network as a TAM without compromising fault coverage or test time.
- Research on NoC testing is still premature when compared to industrial needs, and future research and development are needed.
- Wrapper design techniques for SoC testing can be adopted by NoC-based systems.
- Case studies for SoC testing and NoC testing have been provided to demonstrate efforts in testing real-world SoC and NoC designs.