Génération de tests basés sur les modèles pour des systèmes sur puce avec cohérence de caches

Massimo Zendri & Abderahman Kriouile

STMicroelectronics DCG / IP dev / FVS
Model based test generation for cache coherent Systems On Chips

Massimo Zendri & Abderahman Kriouile

STMicroelectronics DCG / IP dev / FVS
Where you find us

Our automotive products are making driving safer, greener and more entertaining

Our MEMS & Sensors are augmenting the consumer experience

Our smart power products are allowing our mobile products to operate longer and making more of our energy resources

Our Microcontrollers are everywhere making everything smarter and more secure

Our digital consumer products are powering the augmented digital lifestyle

Our digital consumer products are powering the augmented digital lifestyle
Towards the Home Cloud

- Personal Clients
- Over-The-Top Services
- Operator Managed Network & Services
- Medium Screen Clients
- Small Screen Clients
- Home Automation Clients
- Personal Clients
- On the Move
- Broadcast Set-Top Box
- Home gateway
- Big Screens
- Connected Client & Server
Heterogeneous System-on-Chip (SoC)

Need for System-Level Cache Coherency
ARM proposed ACE specification: standard for system level cache coherency
Simulation-Based Testing

Abstract

CPU with cache

Abstract CPU with cache

Abstract video decoder without cache

Cache Coherent Interconnect (CCI)
Verilog/VHDL
Simulation-Based Testing

State of the Art

Abstract

CPU with cache

Monitor

128b

Abstract CPU with cache

Monitor

128b

Abstract video decoder without cache

Monitor

128b

Cache Coherent Interconnect (CCI)

Verilog/VHDL

• Assertions: CPU behavior
• Constraints: CCI behavior
Expressed as SystemVerilog assertions or PSL properties

Interface-level

Formal blocks

Non-formal blocks

A. KROIULE, W. SERWE

Using Formal Model to Improve Verification of Cache-Coherent SoC
Model Checking
(without running any test)

- Applying restrictions for more exploration
- Limitation due to state-space explosion problem
HW Model Based Test Generator

Graph-Based Stimulus Description

- Stimulus scenario described using Rules
  - Captures data and control flow aspects of test scenario
  - Describes legal stimulus scenario space

State of the Art

Automatic Coverage Closure

Activity diagram for generated scenarios – getting from exclusive to modified by invalidating the line using another core write

Scenario goal: observe any state on core 3 followed by any state on core 3

Unreachable paths – cannot get back from modified to exclusive (without implicit cache maintenance)
Need for System-Level Verification

Abstract CPU with cache
Monitor

Abstract CPU with cache
Monitor

Abstract GPU without cache
Monitor

Cache Coherent Interconnect (CCI)
Verilog/VHDL

System-level
Formal Model of an ACE-based SoC

- Interface transfers modeled by rendezvous
- 3400 lines of LNT code derived from ACE specification
- Parametric: #masters, forbidden ACE transactions, ...
CADP: OCIS (Open/Cæsar Interactive Simulator)

- language-independent
- tree-like scenarios
- save/load scenarios
- source code access
- dynamic recompile
Generation of System-Level Test Cases

- Formal model
- System properties

function CIC

- Restricted model
- Model checker
- Counter-examples

Interesting configurations

Test generation

- [Tretmans-92]
- [Jard-Jeron-05]

Test purposes

Abstract test cases

IVK

Coverage-directed solver

Concrete RTL tests
**Inputs**
Architectural description (TDL) either generated by interconnect designers GUI or through Excel flow

**Outputs**
Full Verification Environment, including sequences and coverage models
Several Kinds of Derived Tests

- **39 + 3 generated CTGs (Complete Test Graphs)**

<table>
<thead>
<tr>
<th>prop.</th>
<th>masters</th>
<th>global CTG states</th>
<th>extr. trans.</th>
<th>extr. time</th>
<th>nb. of CTGs</th>
<th>largest CTG states</th>
<th>smallest CTG states</th>
<th>extr. trans.</th>
<th>extr. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varphi_3$</td>
<td>2ACE</td>
<td>6,402</td>
<td>14,323</td>
<td>$&gt;\frac{1}{2}$ y</td>
<td>18</td>
<td>903</td>
<td>1,957</td>
<td>274</td>
<td>543</td>
</tr>
<tr>
<td>$\varphi_5$</td>
<td>2ACE</td>
<td>23,032</td>
<td>48,543</td>
<td>$&gt;\frac{1}{2}$ y</td>
<td>14</td>
<td>462</td>
<td>888</td>
<td>59</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>1ACE/1Lite</td>
<td>2,815</td>
<td>7,071</td>
<td>$&gt;\frac{1}{2}$ y</td>
<td>7</td>
<td>193</td>
<td>394</td>
<td>59</td>
<td>107</td>
</tr>
</tbody>
</table>

- **296 simple system-level tests**

  - for each correct initial state with two masters possibly sharing a memory line, initiate all permitted transitions
  - check correct behavior of the Cache Coherent Interconnect (e.g., generation of corresponding snoops)

- **10 sequence tests to recreate counter-examples**

  - concurrency between transactions
  - conditioned by response of the Cache Coherent Interconnect
Results

• 300 IVK tests generated

• Many problems identified on the verification environment (VIP components)

• System level assertions to check system behavior

• 100% coverage of system level assertions

• Reproduction of 1 suspected architectural issue

• Used on 2 currently developed products (codenamed Orly3 and Barcelona)