SoCProbe: Compositional Post-Silicon Validation of Heterogeneous NoC-Based SoCs

Gabriele Tombesi¹, Joseph Zuckerman¹, Paolo Mantovani¹, Davide Giri¹, Maico Cassel dos Santos¹, Tianyu Jia², David Brooks², Gu-Yeon Wei², and Luca P. Carloni¹

Columbia University¹, Harvard University²
Background

Paradigm shift from multicore to heterogeneous computing inherently increases design complexity

- More lines of the RTL code
- **Complex interactions** among components of different nature
- More **physical design** runs
- Increased **verification** effort

Increased likelihood of faults in the manufactured chips
Motivation

❖ Industry promoted and enhanced DFT methods (boundary scan and ATPG) to provide fine-grained structural testability

❖ This work addresses the need for coarse-grained functional validation of:
  • SoC components in the case of interconnect faults
  • Entire system in the case of faults in particular components.

❖ Three main categories of faults can affect post-silicon validation of NoC-based SoCs:
  • Tile faults
  • I/O faults
  • On-chip communications faults

https://www.corelis.com/education/tutorials/jtag-tutorial/jtag-technical-primer/
The SoCProbe approach

- SoCProbe serves as a debug unit that sits between an individual tile and its NoC routers.

- Three key requirements inform our design choices:
  - **Bypassable**
  - **Latency-insensitive**
  - **Low I/O requirements**
SoCProbe microarchitecture

- **4-pin IO interface to meet scalability requirements:**
  - SPI: shift-in reqs
  - SPO: shift-out reqs
  - SPM: toggle op. mode
  - SCLK: provide clock domain

- **Deserializaiton/serializaton layer** to convert between serial IO and NoC bitwidth

- **Multi-plane orchestration layer** to manage the communication between SoCProbe and the EUT:
  - Async-fifos
  - Mux/demux logic
  - Control FSM
SoCProbe Operating modes

- Standard tile-to-router connection
- SoCProbe logic inactive
- The router is disconnected
- SoCProbe provides test access to the tile under test
- The tile is disconnected
- SoCProbe provides test access to the router under test
SoCProbe Validation Flow

For a given application, collect flits at the NoC–tile boundary in full-system RTL simulation and SoCProbe in normal mode.

An FPGA test bed is used to stimulate the fabricated chip by using extracted NoC flits as a test vector.
SoC integration

Target SoC for SoCProbe’s validation:

• Tile-based architecture
• NoC-based interconnects

our choice:

✓ 2-D-mesh NoC with 6 physical planes
✓ Heterogeneous tiles: CPU, accelerator, I/O, and memory.
✓ Modular socket with:
  ▪ Platform services for DMA, cache coherence, monitors and interrupts
  ▪ Latency-insensitive channels to NoC

https://esp.cs.columbia.edu
SoC integration

➢ Different strategies to connect multiple SoCProbe units depending on the tradeoff between design complexity and I/O utilization:

a) ✓ Scalable
   ✗ Complex top-level

b) ✓ Simple
   ✗ Not Scalable

c) ✓ Our tradeoff

[Diagram showing connections and components like aux, μp, acc, mem, slm, and SoCProbe controller]
Fabricated chip

- 12-nm process.
- 16 tiles composed of six different types
- Multiple open-source processors and accelerators
- 4x4 2-D mesh NoC

- **Ariane tile area:** 374.44 μm²
- **SoCProbe area (in red):** 3,370 μm², => 0.9%

Jia et al. ESSCIRC’22
Test Cases and results

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Test Name</th>
<th>Test Mode</th>
<th>Tile Under Test</th>
<th>Total Flits</th>
<th>Reqs</th>
<th>Resps</th>
<th>Norm. Time (µs)</th>
<th>Test Time (µs)</th>
<th>Planes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CSR write</td>
<td>Tile</td>
<td>any</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>$2.13 \times 10^1$</td>
<td>5.3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>PC read</td>
<td>Tile</td>
<td>any</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>$2.14 \times 10^1$</td>
<td>$9.8 \times 10^1$</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>NoC intercept</td>
<td>Tile</td>
<td>IO</td>
<td>60</td>
<td>0</td>
<td>60</td>
<td>$3.64 \times 10^2$</td>
<td>$1.9 \times 10^3$</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>CPU in isolation</td>
<td>Tile</td>
<td>CPU</td>
<td>5842</td>
<td>2938</td>
<td>2904</td>
<td>$4.45 \times 10^2$</td>
<td>$7.97 \times 10^4$</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>ACC in isolation</td>
<td>Tile</td>
<td>ACC</td>
<td>682</td>
<td>536</td>
<td>146</td>
<td>$1.21 \times 10^3$</td>
<td>$9.88 \times 10^3$</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>MEM resp.</td>
<td>NoC</td>
<td>MEM</td>
<td>753</td>
<td>484</td>
<td>269</td>
<td>$4.45 \times 10^2$</td>
<td>$3.13 \times 10^4$</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>SYS orch.</td>
<td>NoC</td>
<td>CPU</td>
<td>22721</td>
<td>12151</td>
<td>10570</td>
<td>$1.21 \times 10^3$</td>
<td>$8.14 \times 10^5$</td>
<td>4</td>
</tr>
</tbody>
</table>

1) Writing a configuration register  
2) Reading performance counters  
3) Intercepting NoC messages.  
4/5) Functional validation of tiles in isolation.  
6) Issuing memory responses.  
7) Orchestrating full-system tests.
Conclusions

**SoCProbe**: solution for *compositional validation* of tiled SoC architectures connected by an NoC.

The **SoCProbe approach** permits:

- Direct access to individual tiles for validation
- Diagnostic
- Debugging

- **Reusable** across the different tiles of a heterogeneous SoC
- Supports a **Seamless flow** for collecting test vectors for a variety of applications.

- Integration in ESP and evaluation of its efficacy and overhead in a 12-nm chip prototype.
- We have released SoCProbe’s implementation, test-flow scripts, and FPGA test-bed design available as open-source artifacts.