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

Gabriele Tombesi[1], Joseph Zuckerman[1], Paolo Mantovani[1], Davide Giri[1], Maico Cassel dos Santos[1],
Tianyu Jia[2], David Brooks[2], Gu-Yeon Wei[2], and Luca P. Carloni[1]

Columbia University[1], Harvard University[2]

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

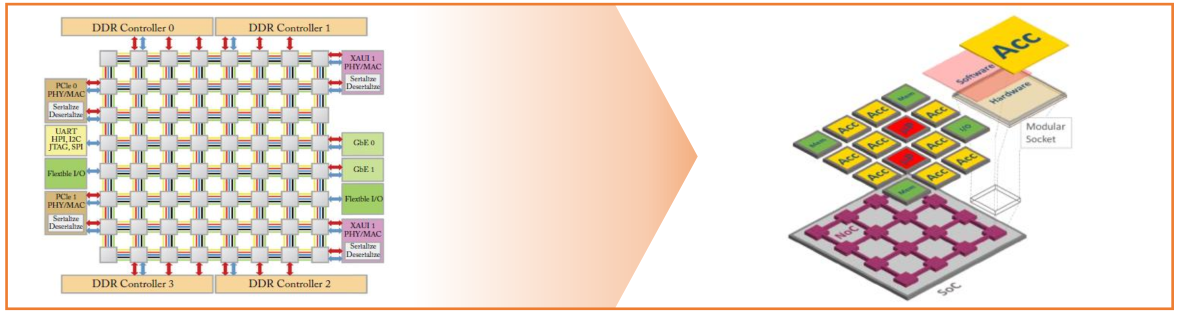
CS@CU

Background

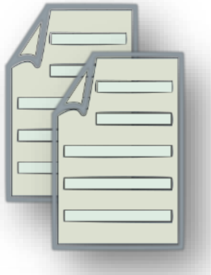
Paradigm shift from multicore to **heterogeneous computing** inherently increases **design complexity**

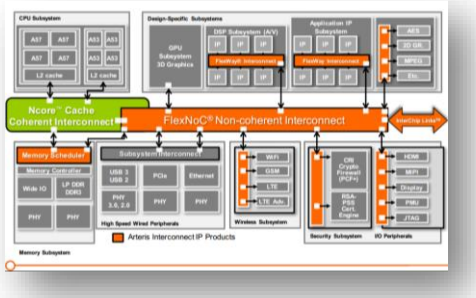
- More lines of the **RTL** code
- **Complex interactions** among components of different nature
- More **physical design** runs
- Increased **verification** effort

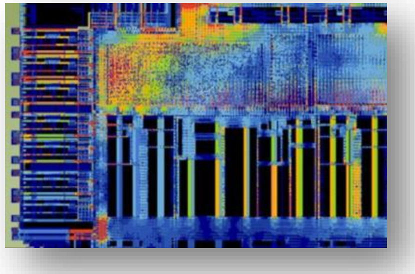
Increased likelihood of faults in the manufactured chips

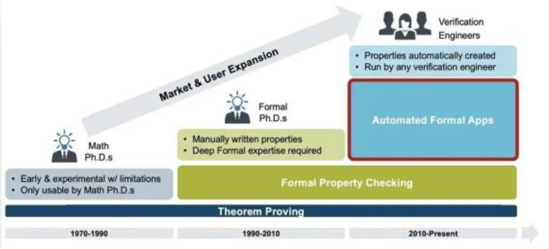


.v .vhdl







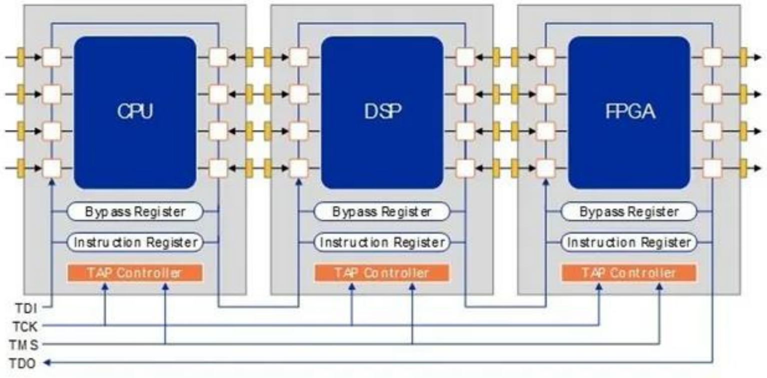

Verification Engineers
Properties automatically created
Run by any verification engineer
Automated Formal Apps
Market & User Expansion
Formal Ph.D.s
Manually written properties
Deep Formal expertise required
Math Ph.D.s
Early & experimental w/ limitations
Only usable by Math Ph.D.s
Formal Property Checking
Theorem Proving
1970-1990
1990-2010
2010-Present

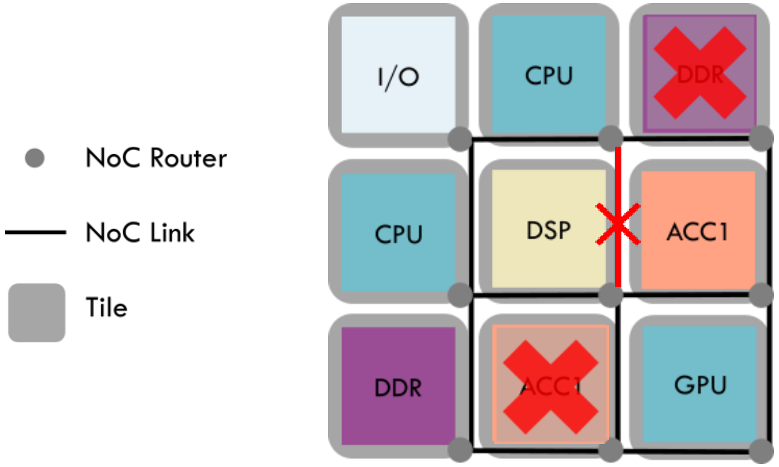
Motivation

- 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**



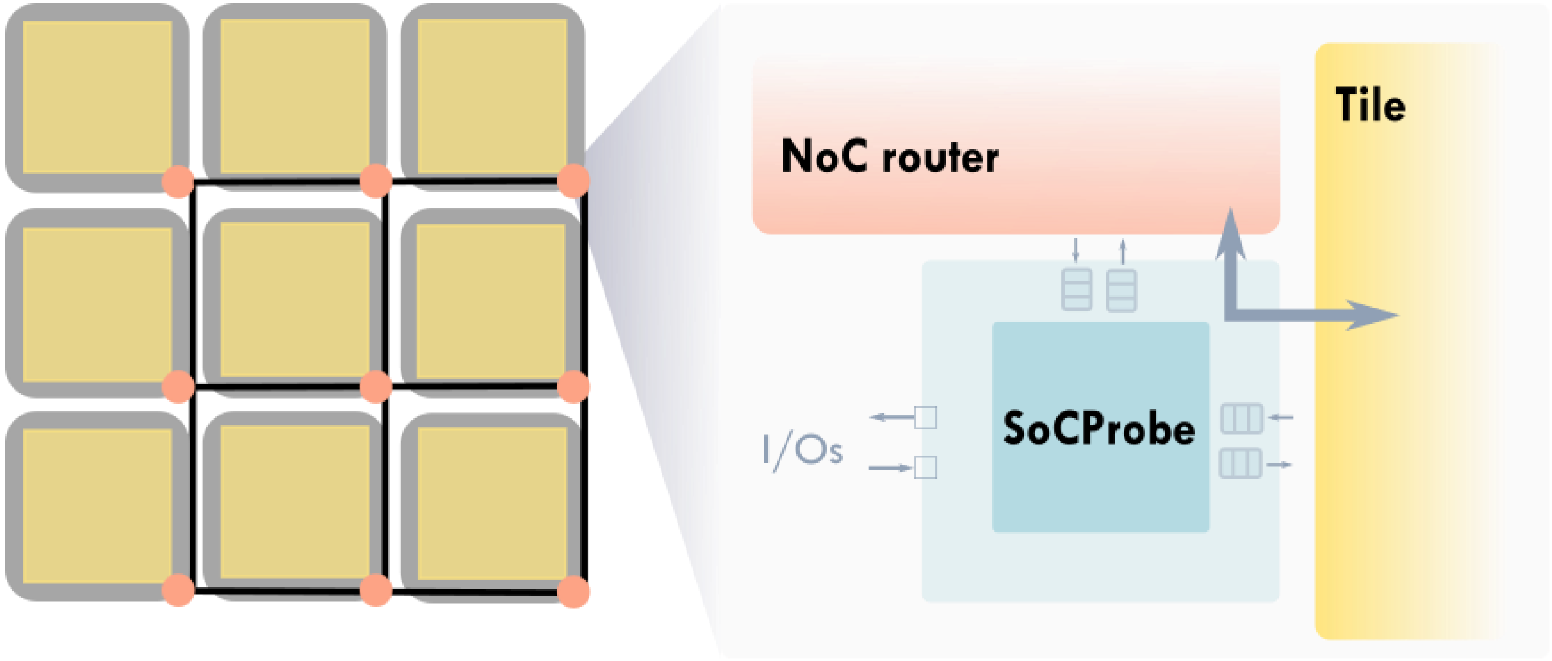


https://www.corelis.com/education/tutorials/jtag-tutorial/jtag-technical-primer/

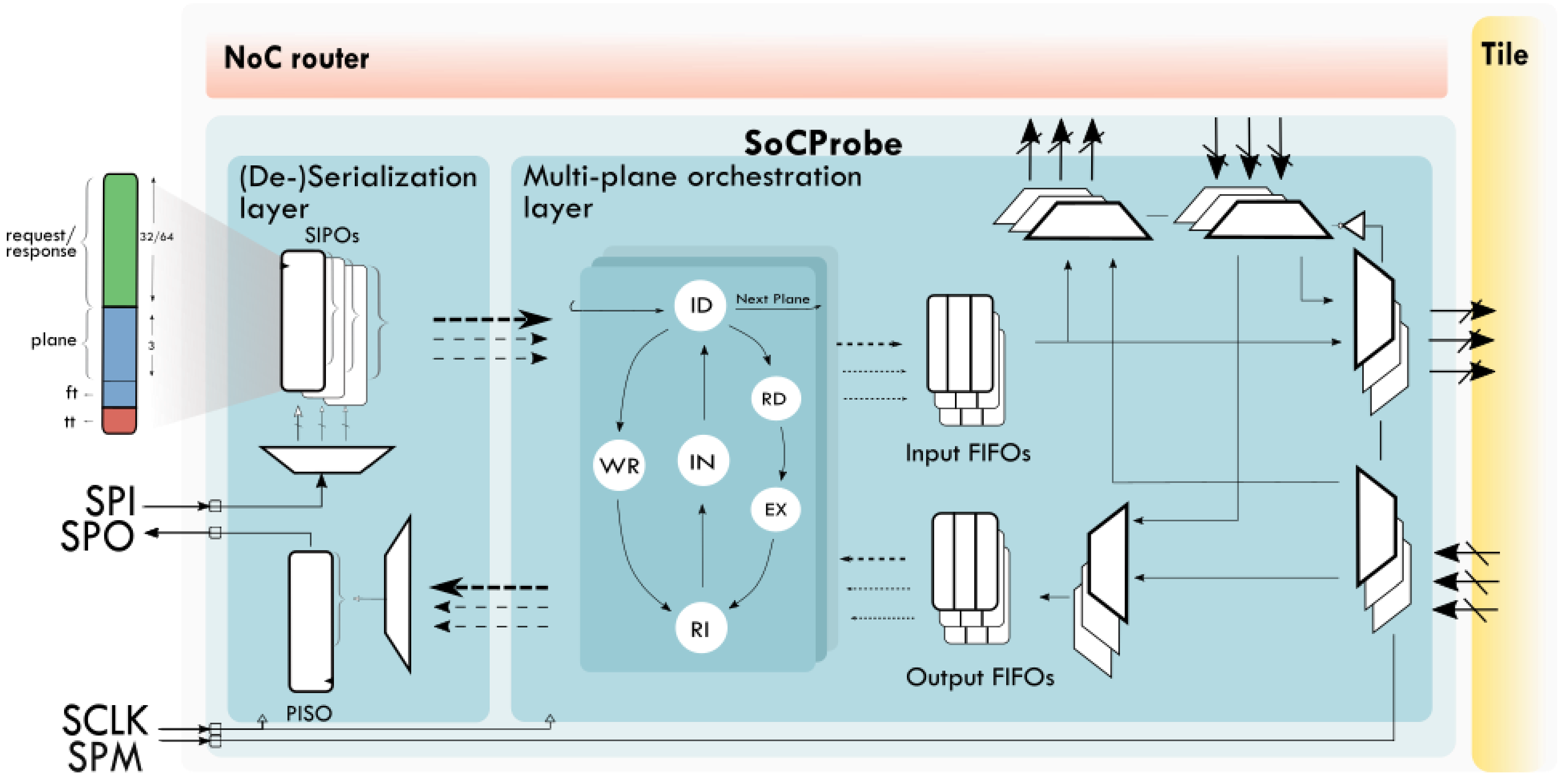
The SoCProbe approach

- 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***

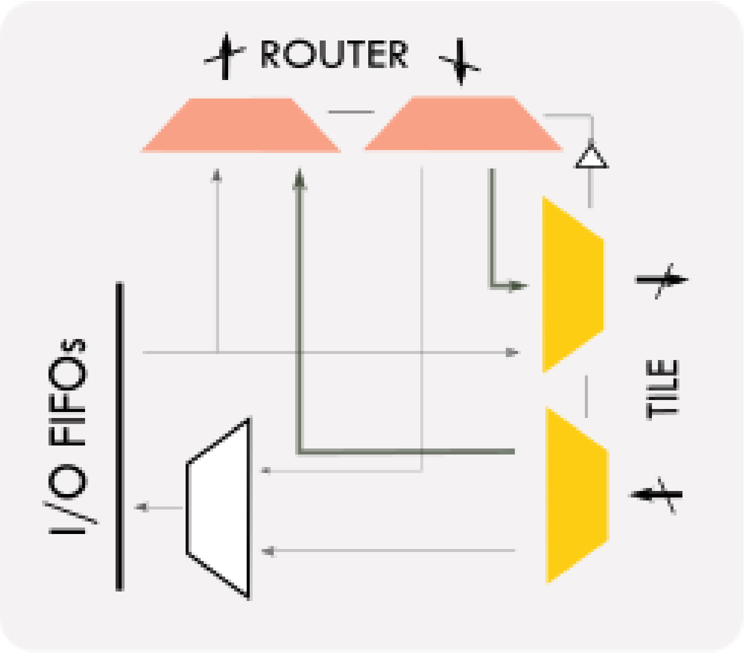


SoCProbe microarchitecture



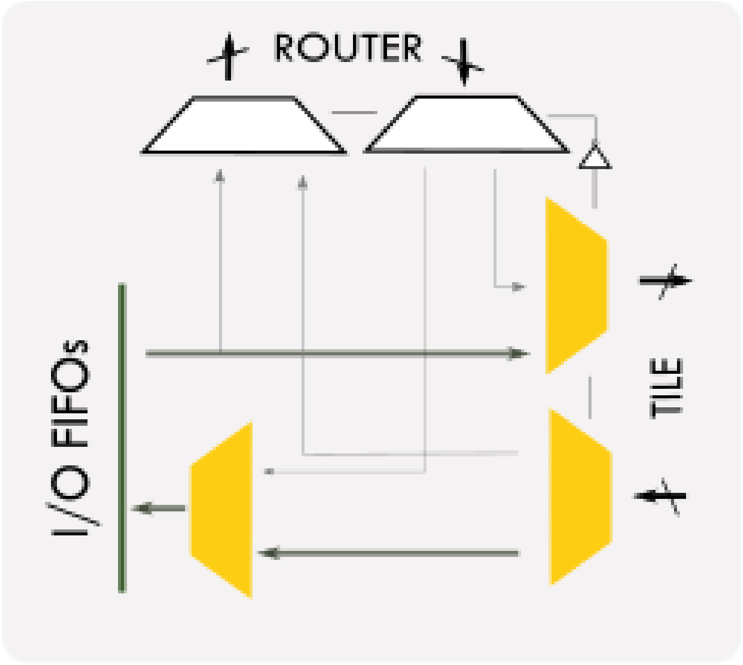
- ➢ ***4-pin IO** interface to meet **scalability** requirements:*
  ***SPI**: shift-in reqs*
  ***SPO**: shift-out reqs*
  ***SPM**: toggle op. mode*
  ***SCLK**: provide clock domain*
- ➢ ***Deserializaiton/serialization layer** to convert between serial IO and NoC bitwidth*
- ➢ ***Multi-plane orchestration layer** to manage the **communication** between SoCProbe and the EUT:*
  - *Async-fifos*
  - *Mux/demux logic*
  - *Control FSM*

SoCProbe Operating modes



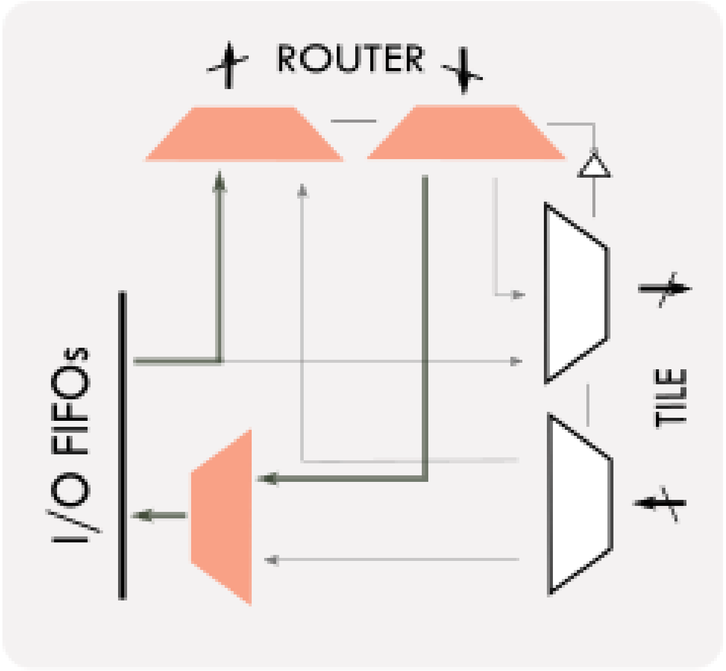
NORMAL MODE (SPM=0)

- Standard tile-to-router connection
- SoCProbe logic inactive



TILE-TEST MODE (SPM=1, tt=1)

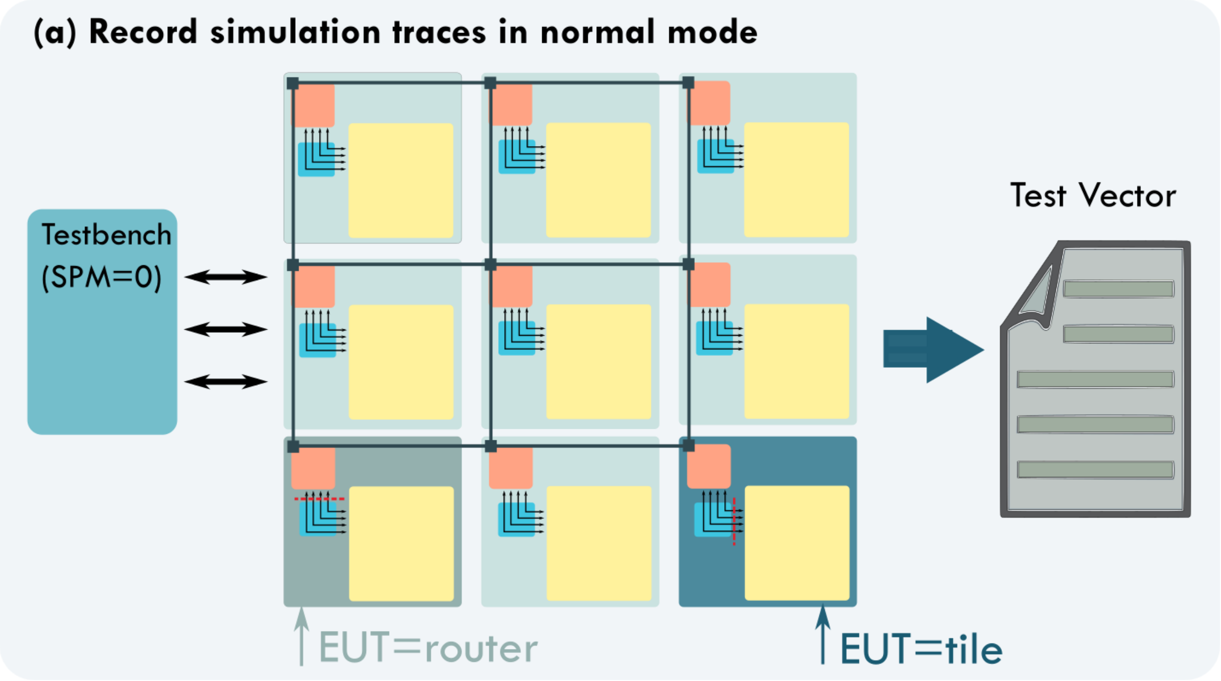
- The router is disconnected
- SoCProbe provides test access to the tile under test

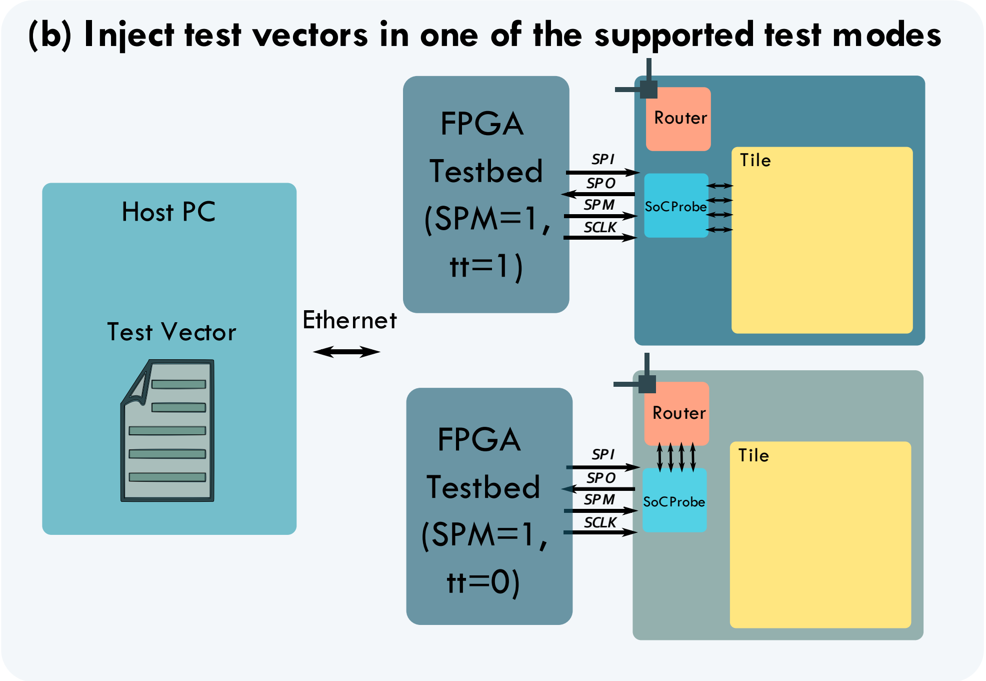


NOC-TEST MODE (SPM=1, tt=0)

- The tile is disconnected
- SoCProbe provides test access to the router under test

SoCProbe Validation Flow


(a) Record simulation traces in normal mode


(b) Inject test vectors in one of the supported test modes

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: ESP

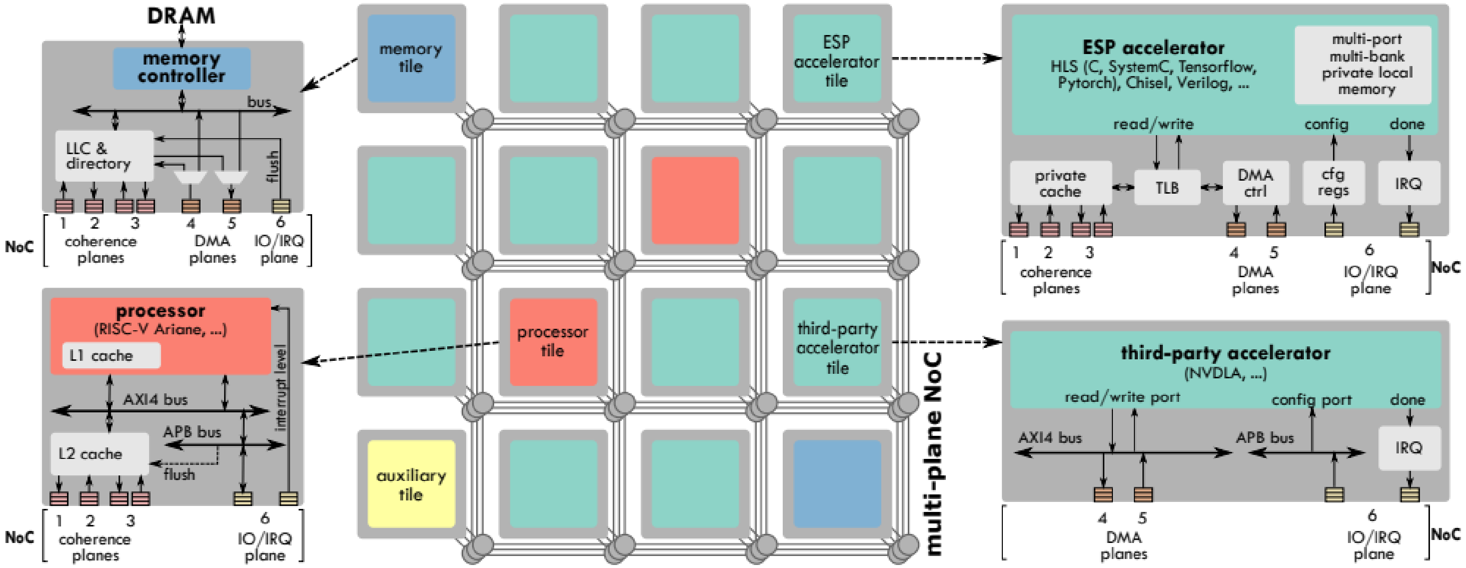
✓ **2-D-mesh NoC** with 6 physical planes

✓ **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

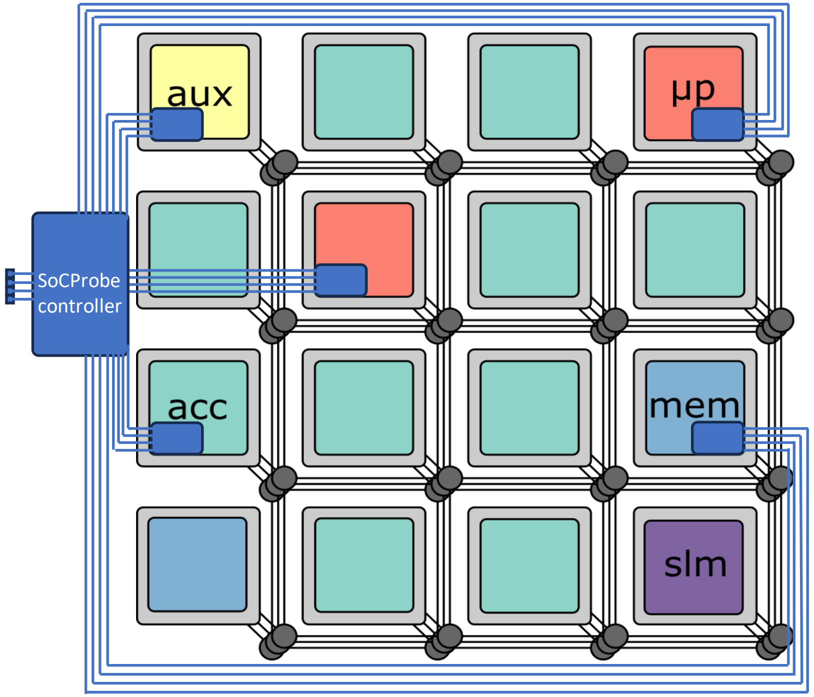


https://esp.cs.columbia.edu

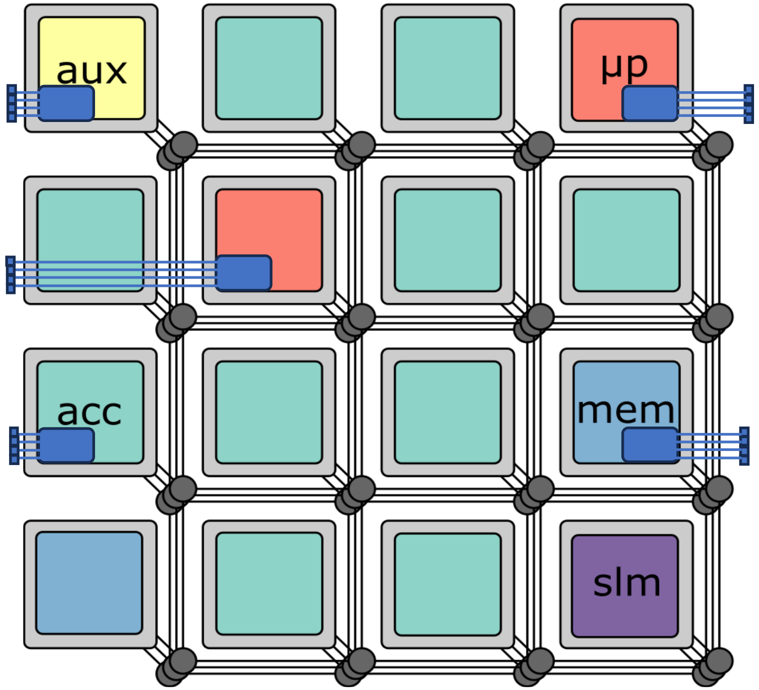
SoC integration

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

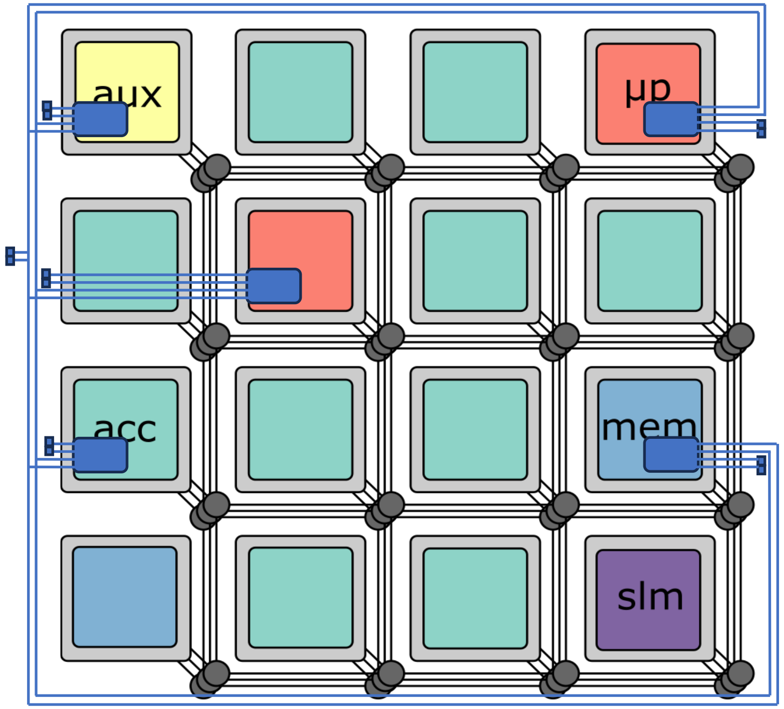
a) ✓ **Scalable**
✗ **Complex top-level**



b) ✓ **Simple**
✗ **Not Scalable**

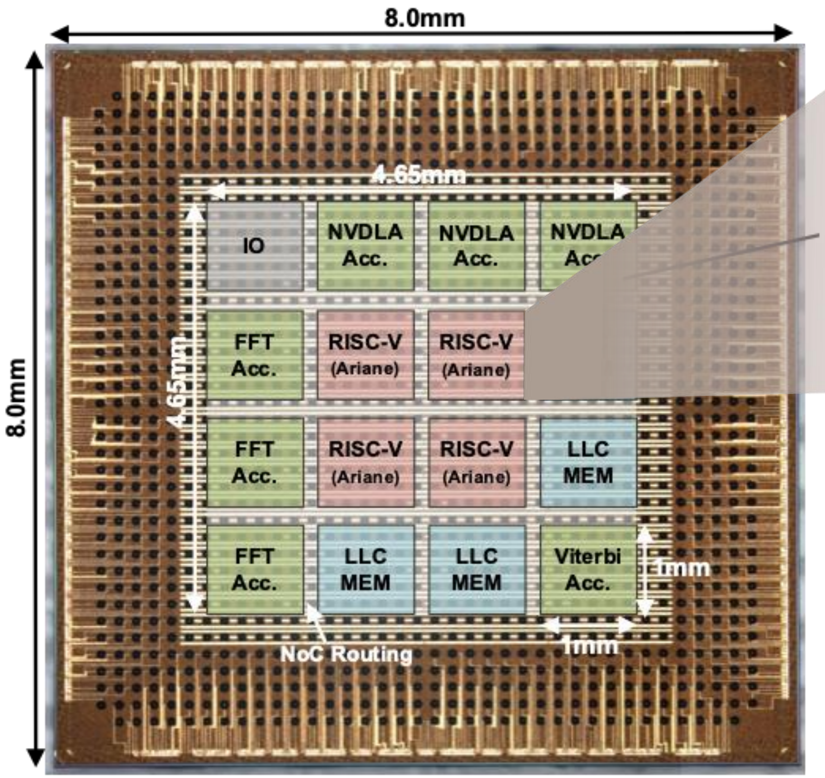


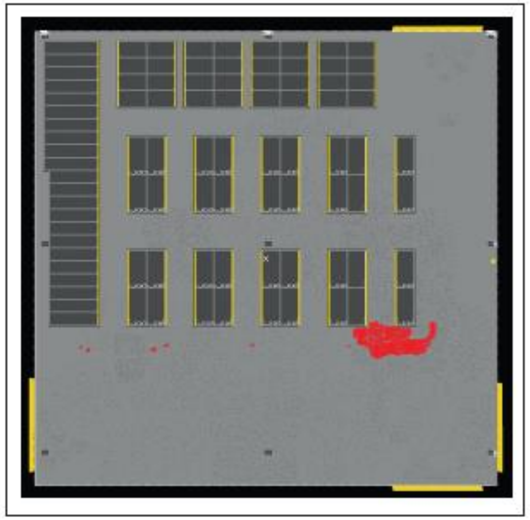
c) ✓ **Our tradeoff**



Fabricated chip

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

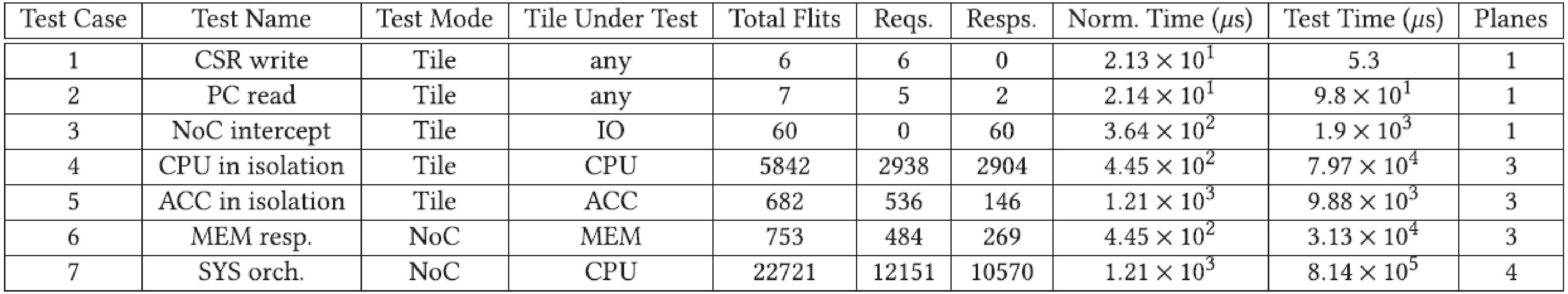




- ▪ **Ariane tile** area:374,44 µm$_2$
- ▪ SoCProbe area (in red): 3,370 µm$_2$,
=> 0.9%

Jia et al. ESSCIRC’22

Test Cases and results



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

1) Writing a configuration register

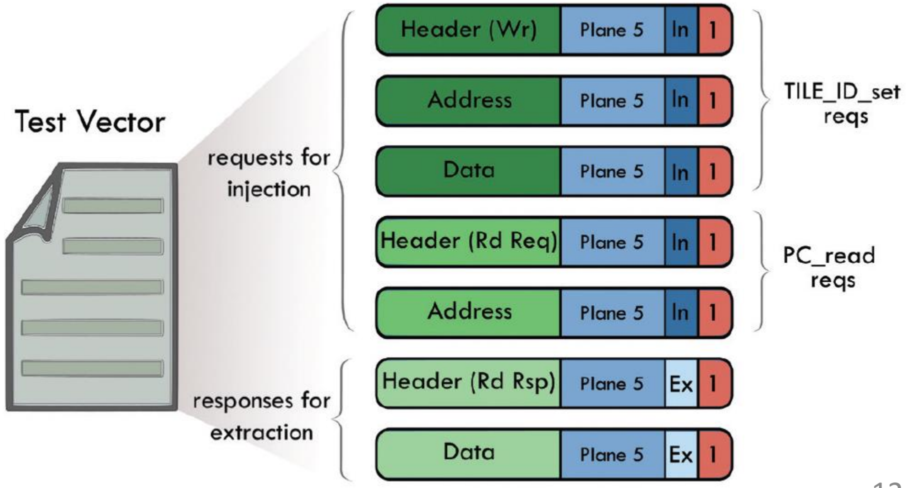
2) Reading performance counters

3) Intercepting NoC messages.

4/5) Functional validation of tiles in isolation.

6) Issuing memory responses.

7) Orchestrating full-system tests.



Test Vector
requests for injection
responses for extraction
Header (Wr) | Plane 5 | In | 1
Address | Plane 5 | In | 1
Data | Plane 5 | In | 1
Header (Rd Req) | Plane 5 | In | 1
Address | Plane 5 | In | 1
Header (Rd Rsp) | Plane 5 | Ex | 1
Data | Plane 5 | Ex | 1
TILE_ID_set reqs
PC_read reqs

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.