RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

ONERA
THE FRENCH AEROSPACE LAB

Extracting the Topology of a Hardware Platform Using Hardware Event

Cédric CAZANOVE, Benjamin LESAGE, Frédéric BONIOL, Sandrine MOUYSSET, Jérôme ERMONT

Context

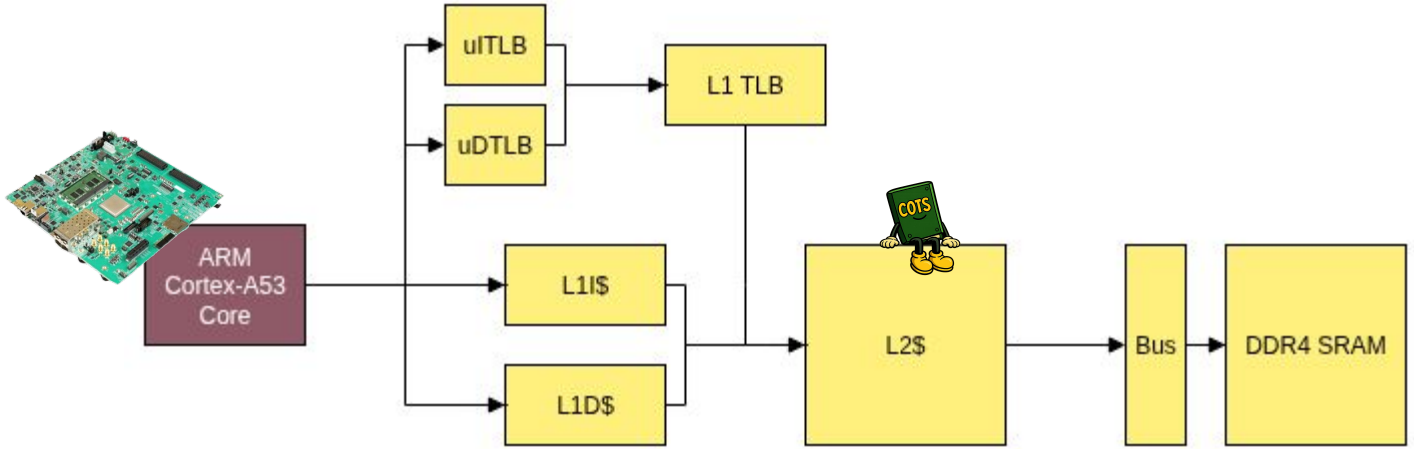


Fig: ARM Cortex-A53 Memory Hierarchy on the Zynq ULTRASCALE+ MPSoC

How to be confident ?

How to be confident ?

Each ARM Cortex-A53 core on the Xilinx ZCU102 includes a PMU able to monitor more than 30 hardware events, such as `L1D_CACHE`, `L2_CACHE`, `BUS_ACCESS`, `LD_RETIRED`, and `ST_RETIRED`, providing observability over memory and instruction behaviors.

How to extract a topology of a Hardware Platform ?

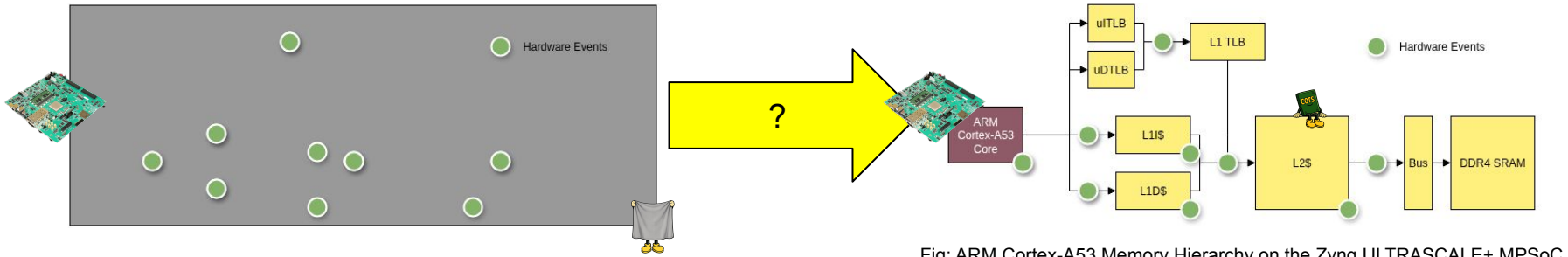


Fig: ARM Cortex-A53 Memory Hierarchy on the Zynq ULTRASCALE+ MPSoC

Observe Behaviours

Targeted populations

Purpose-built benchmarks exercising specific transactions:

- `intAdd` → arithmetic-only behavior
- `readL1` / `writeL1` → L1 cache read/write activity
- `readL2` / `writeL2` → L2 cache read/write activity
- `readMem` / `writeMem` → main memory read/write activity

Each population is designed to isolate a specific resource usage pattern.

A **transaction** is a **specific and controlled interaction** between threads and **targeted COTS components**

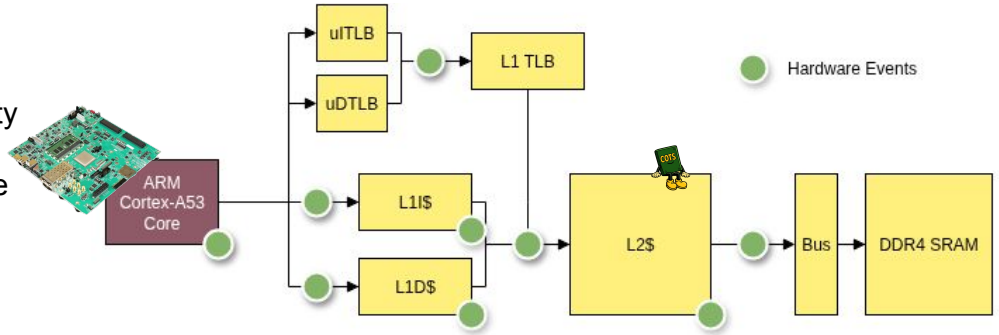
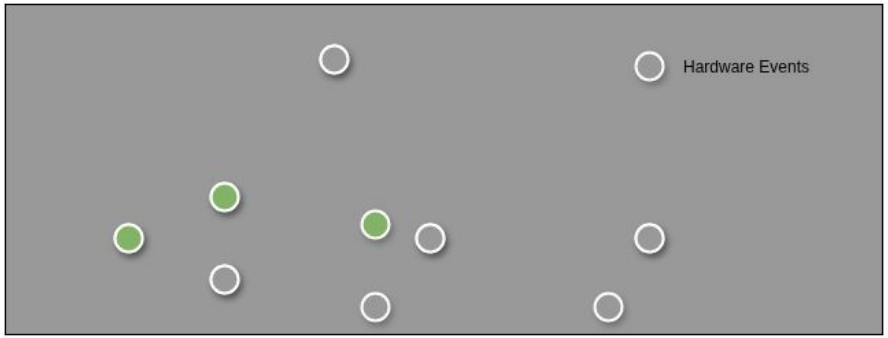


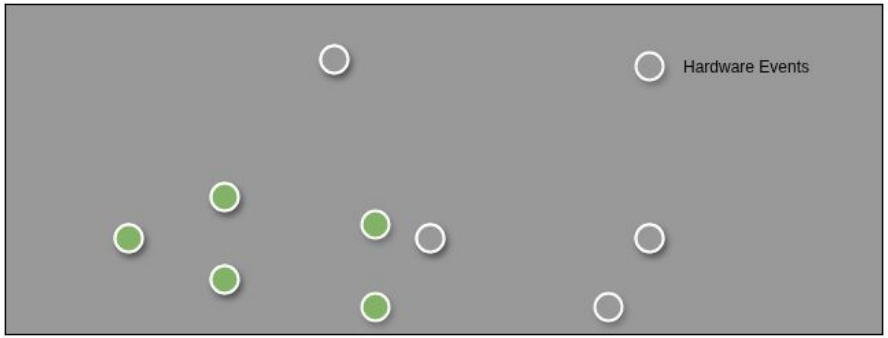
Fig: ARM Cortex-A53 Memory Hierarchy on the Zynq ULTRASCALE+ MPSoC

Triggered Hardware Event

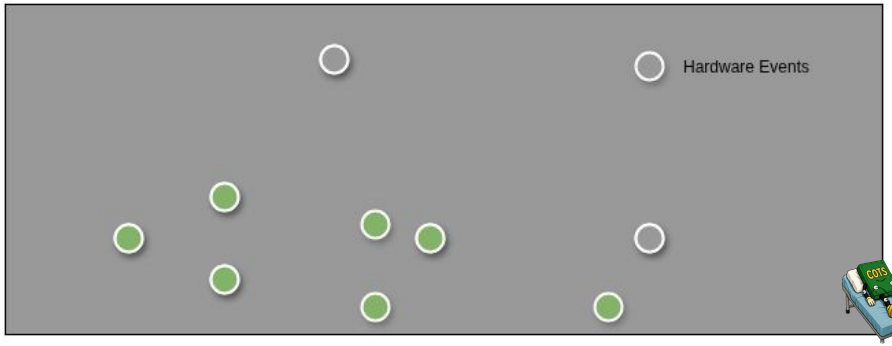
`intAdd`



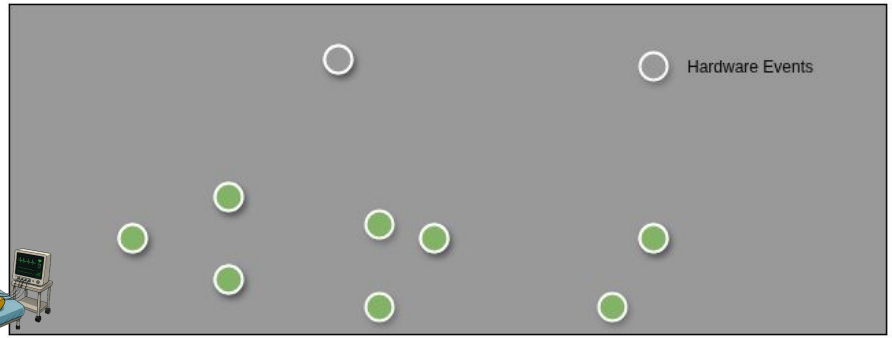
`readL1/writeL1`



`readL2/writeL2`

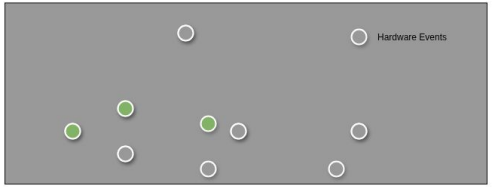


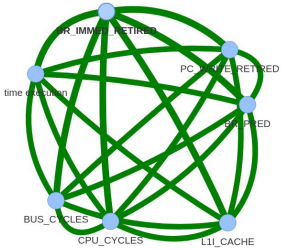
`readMem/writeMem`



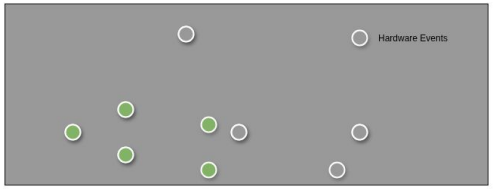
Correlation and MAG (Metric Affinity Graph) Outcome

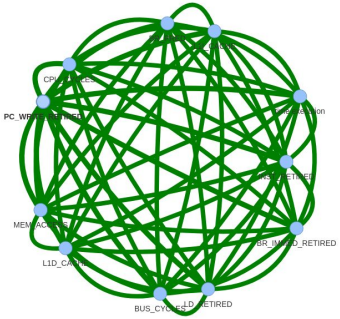
intAdd



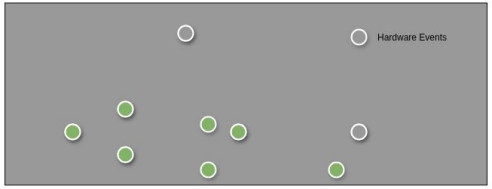


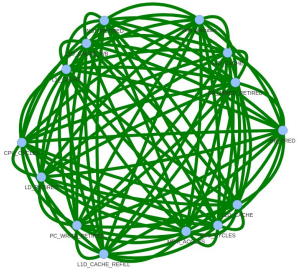
readL1/writeL1



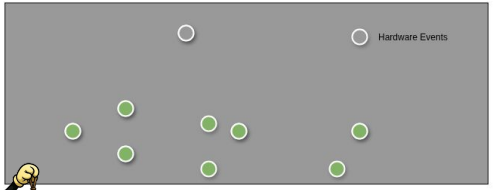


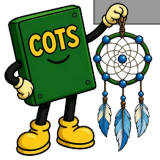
readL2/writeL2

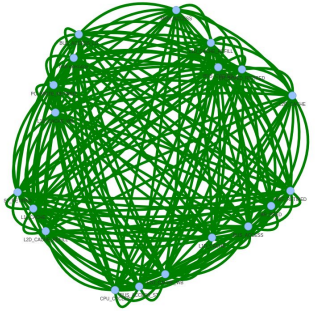




readMem/writeMem

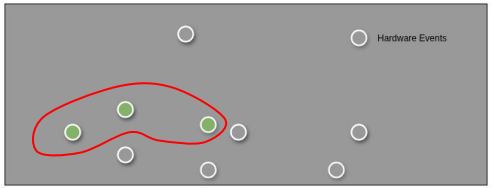


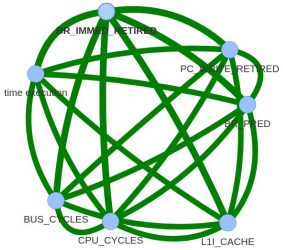




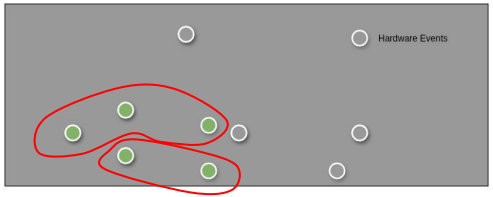
Basic Path Identification

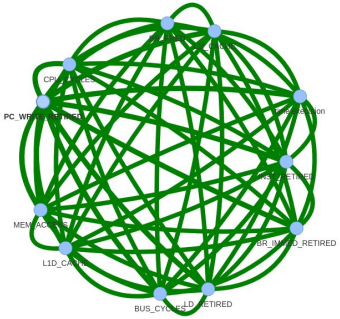
intAdd



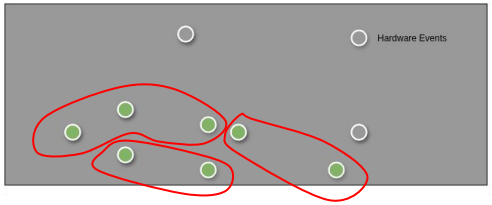


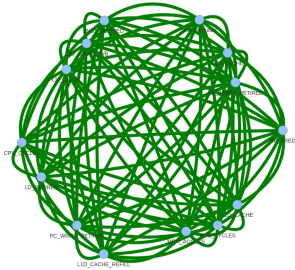
readL1/writeL1



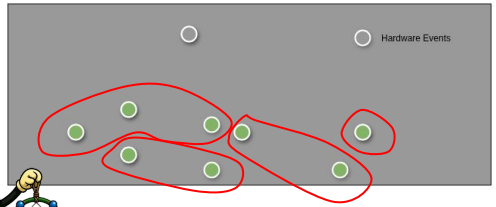


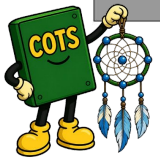
readL2/writeL2

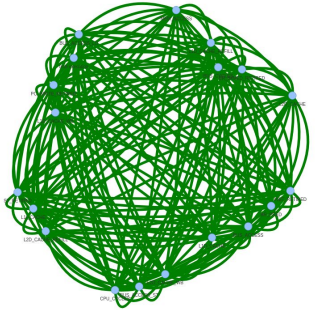




readMem/writeMem

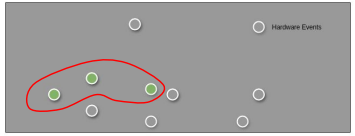


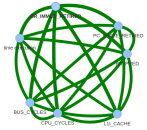




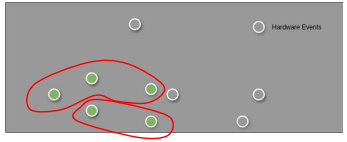
Topology Extraction

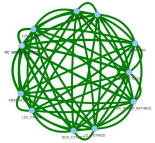
intAdd



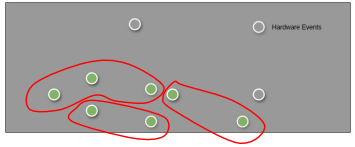


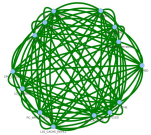
readL1/writeL1



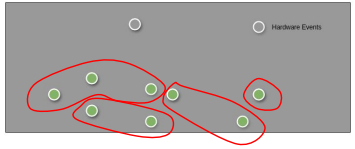


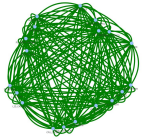
readL2/writeL2

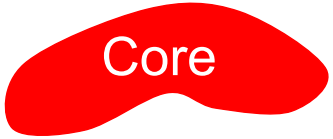




readMem/writeMem

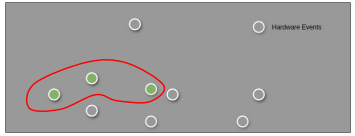


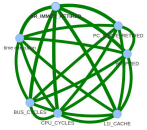




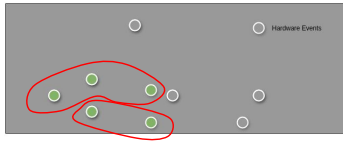
Topology Extraction

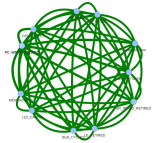
intAdd



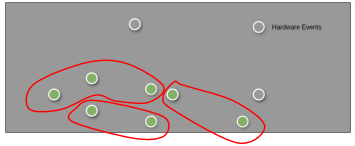


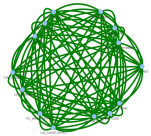
readL1/writeL1



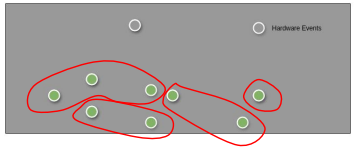


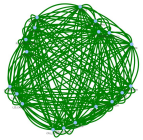
readL2/writeL2

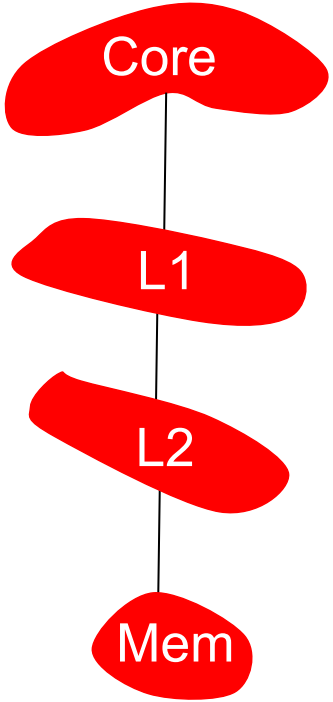




readMem/writeMem

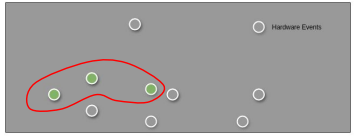


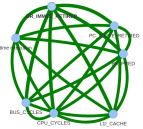




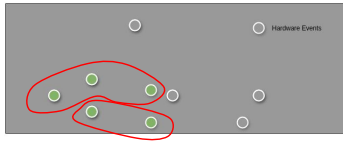
Topology Extraction

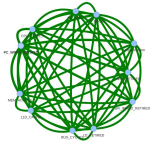
intAdd



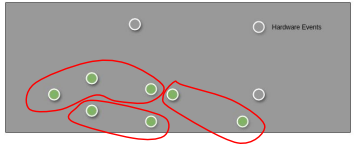


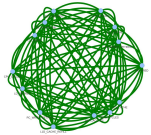
readL1/writeL1



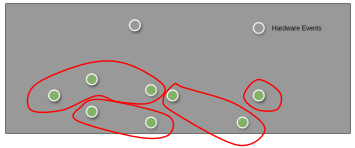


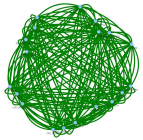
readL2/writeL2

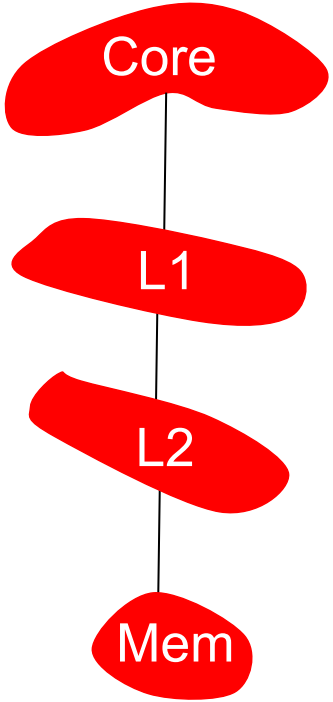


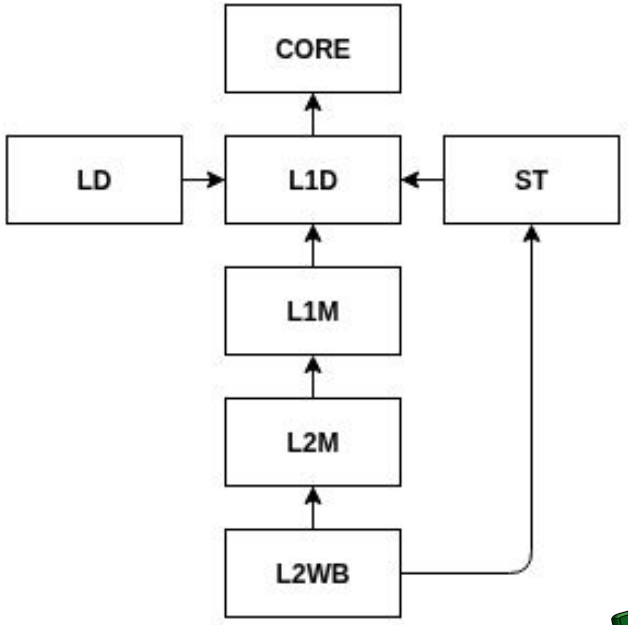


readMem/writeMem



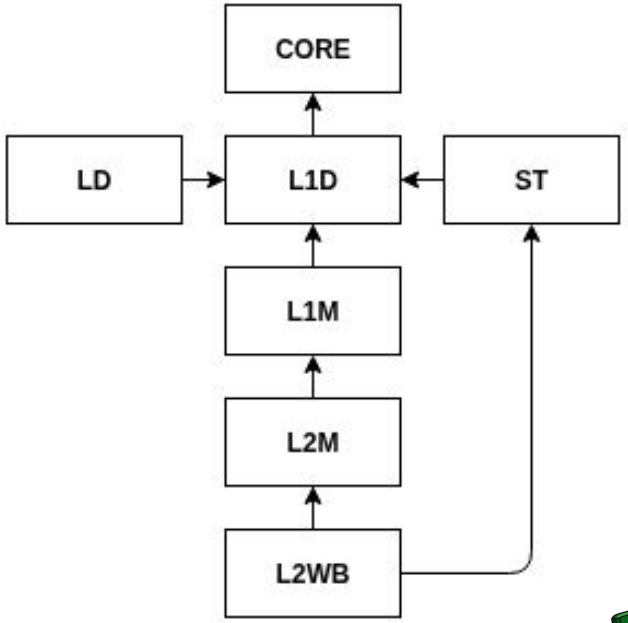






Topology Extraction

Name	Event
L2M	BUS_ACCESS, BUS_ACCESS_LD, L2_CACHE_REFILL
L2WB	BUS_ACCESS_ST, L2_CACHE_WB
LD	LD_RETIRED
L1M	L1D_CACHE_REFILL, L1_CACHE_WB, L2_CACHE
ST	ST_RETIRED
L1D	L1D_CACHE, MEM_ACCESS
CORE	BR_IMMED_RETIRED, BR_PRED, BUS_CYCLES, CPU_CYCLES, L1I_CACHE, PC_WRITE_RETIRED, time execution, INST_RETIRED



Comparison with the Real Platform

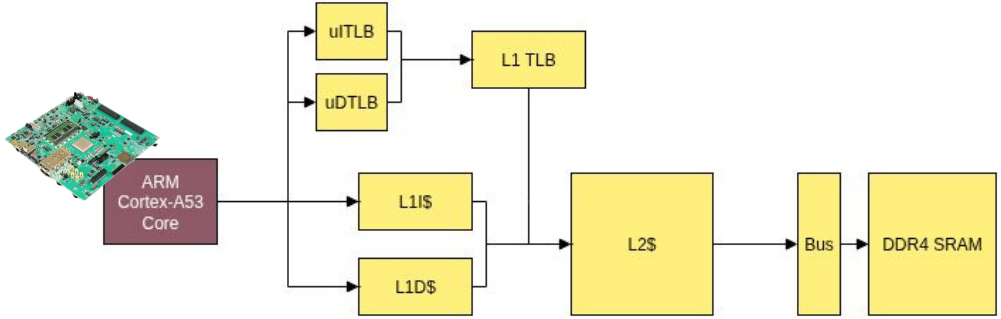


Fig: ARM Cortex-A53 Memory Hierarchy on the Zynq ULTRASCALE+ MPSoC

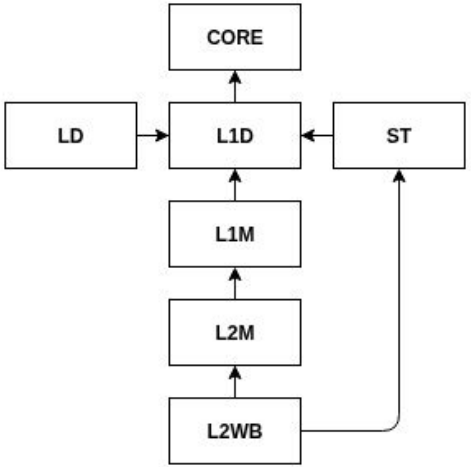


Fig: Topology extracted from the methodology

Observations

- Extracted topology matches known ARM Cortex-A53 memory hierarchy organization
- Shared resources and memory levels are reflected in BP relations
- Dependencies between BPs reveal hierarchical interactions in the platform

Limitations

- Extracted topology depends on the exercised populations
- Untriggered behaviors cannot appear in the resulting topology

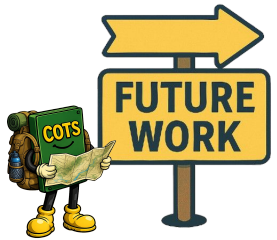
Conclusion & Future Work

Conclusion

- Proposed a methodology to extract an abstract topology of a COTS platform from hardware events
- Used Hardware observations and correlation analysis to identify recurring hardware behaviors
- Reconstructed an interpretable topology coherent with the ARM Cortex-A53 architecture
- Demonstrated that observable hardware events can provide meaningful insights into hidden platform behaviors

Future Work

- Improve confidence and construction of populations
- Extend the methodology to more complex workloads and behaviors
- Apply the approach to larger multicore and heterogeneous platforms



Question(s)

Thank you for listening

Q&A

Context

- Increasing use of **COTS** (Commercial Off-The-Shelf) multicore platforms in critical embedded systems
  - Using HPC capabilities (High Performance Computing) for onboard critical applications
- COTS multicore platforms behave as grey boxes
  - Limited visibility on internal hardware mechanisms
  - Incomplete vendor documentation
- Difficult to fully master and understand platform behavior
  - Critical challenge for timing analysis and certification
- Need for better characterization of COTS platform behavior to **improve confidence** in critical systems

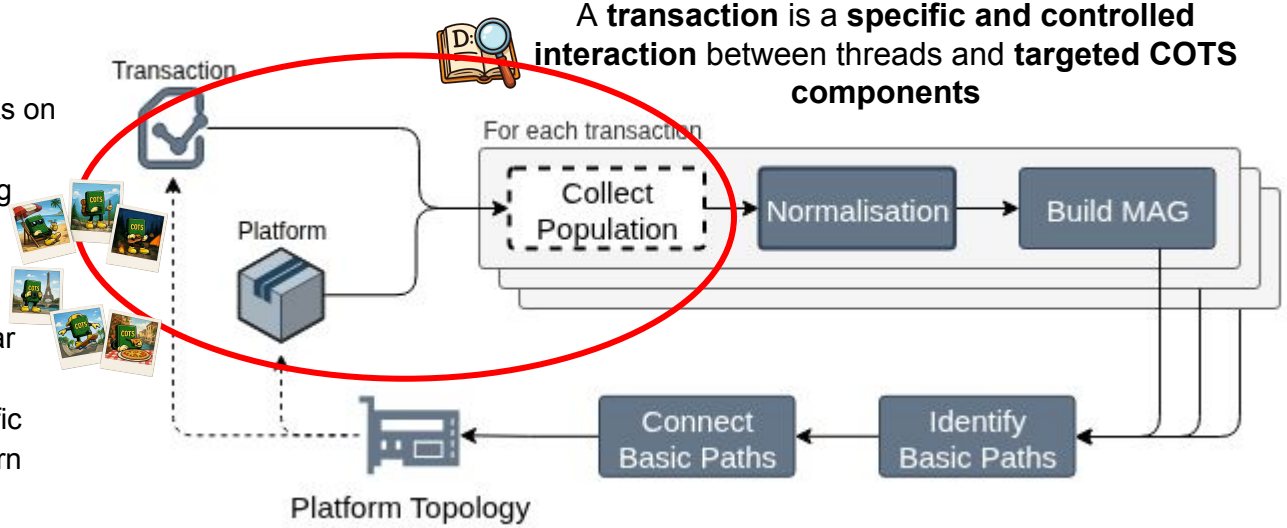
Methodology

1. Observe the platform

- Execute dedicated microbenchmarks on the Xilinx ZCU102
- Collect PMU hardware events during execution

2. Build populations

- Group observations exhibiting similar behaviors
- Each population represents a specific transaction or resource usage pattern



A **transaction** is a **specific and controlled interaction** between threads and **targeted COTS components**

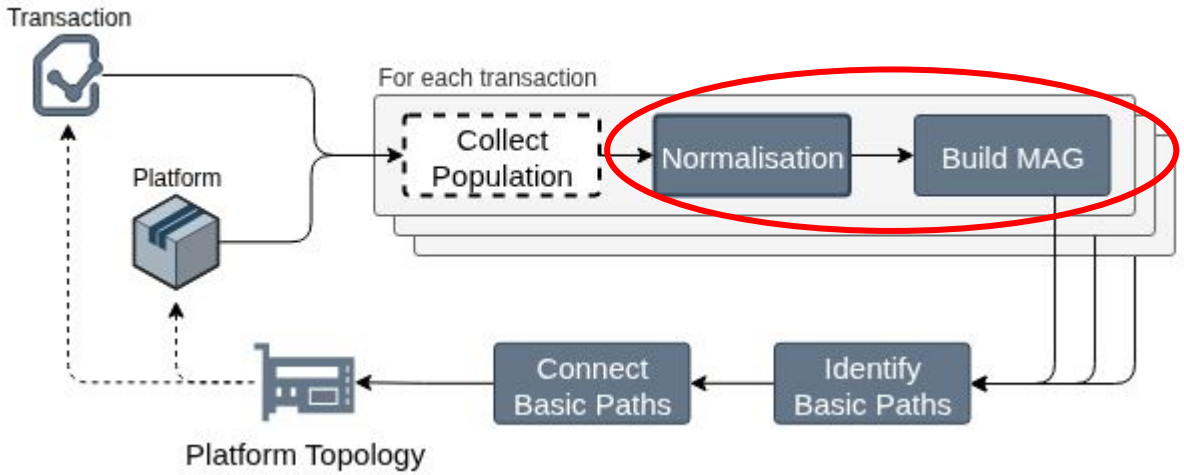
Methodology

3. Correlation analysis

- Compute correlations between hardware events
- Identify strongly related events
- Construct a Metric Affinity Graph (MAG)

Correlation principle:

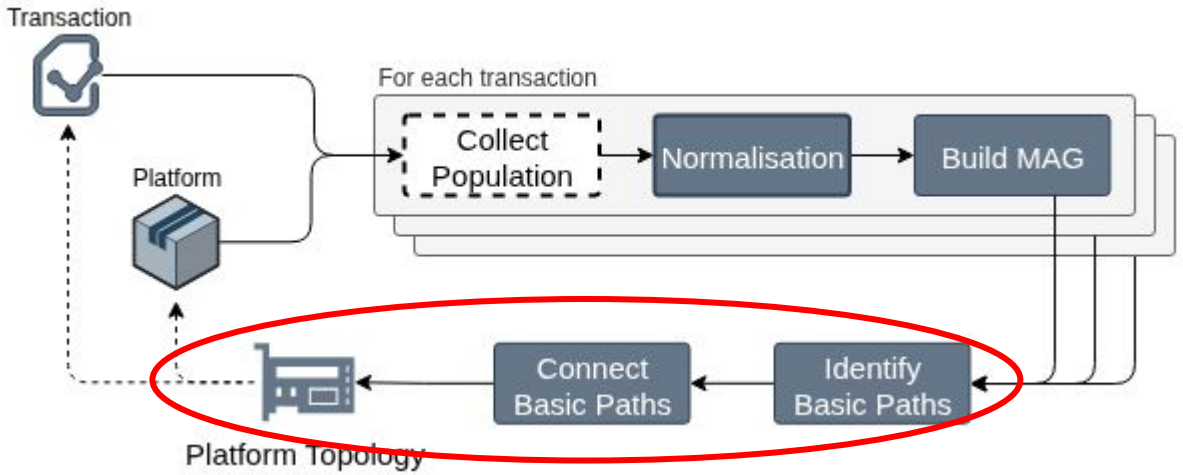
- Strongly correlated events evolve together across executions
- May reveal shared resources or related microarchitectural behaviors



Methodology

4. Build the platform topology

- Extract recurring event groups called Basic Paths (BP)
- Identify relations and implications between BPs
- Construct an abstract topology of the platform behavior



Experimental Setup

Platform under study

- Xilinx ZCU102
- ARM Cortex-A53 core
- **Monothread benchmarks** executed on a single core
- Focus on the memory hierarchy:
  - L1 data cache
  - L2 cache
  - Main memory
- Goal: isolate and observe specific **transactions** through hardware events.

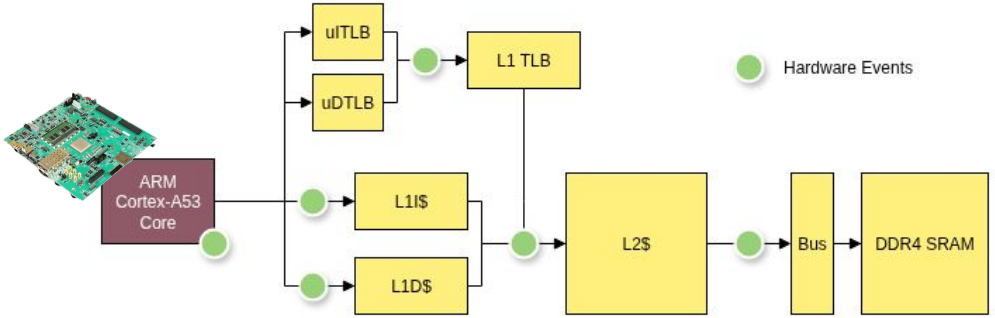


Fig: ARM Cortex-A53 Memory Hierarchy on the Zynq ULTRASCALE+ MPSoC

MAG (Metric Affinity Graph) Outcome

What a MAG represents

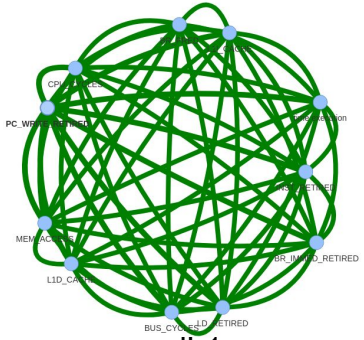
- Nodes represent hardware events
- Edges represent strong correlations between events
- Captures relations between observable platform behaviors

What information can be extracted

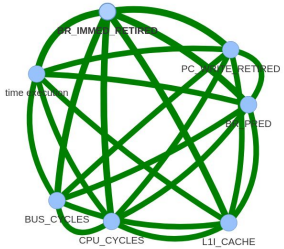
- Identification of groups of related hardware events
- Detection of recurring microarchitectural behaviors
- Insight into resource interactions within the platform

Interest for platform understanding

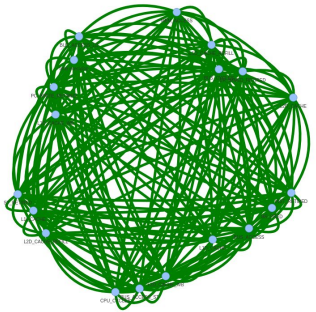
- Provides an observable view of hidden hardware behaviors
- Helps identify which events evolve together



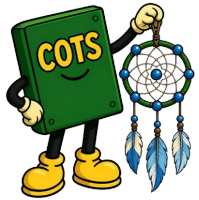
readL1



intAdd



writeMem



Basic Path Construction

Graph intersection

- Intersect MAGs from all populations
- Keep only stable correlations shared across populations

Basic Paths (BP)

- Extract cliques from the intersection graph
- A BP groups strongly related hardware events
- Represents a recurring microarchitectural behavior

Interest of the approach

- Builds stable abstractions of platform behavior

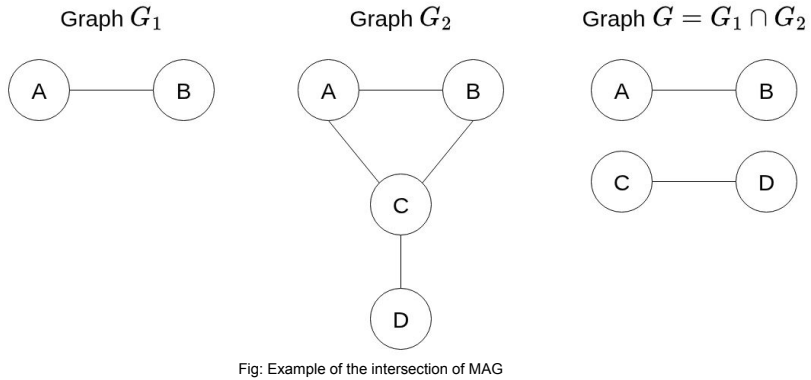


Fig: Example of the intersection of MAG

Basic Paths Identified

- Several stable Basic Paths are identified across populations
- Some BPs are associated with specific memory hierarchy levels
- The extracted BPs provide an interpretable abstraction of recurring platform behaviors
- These BPs constitute the foundation for building the platform topology

Remark

- Not all available hardware events appear in the extracted BPs
- Some events were not triggered by the selected benchmarks and populations under study

Name	Event
L2M	BUS_ACCESS, BUS_ACCESS_LD, L2_CACHE_REFILL
L2WB	BUS_ACCESS_ST, L2_CACHE_WB
LD	LD_RETIRED
L1M	L1D_CACHE_REFILL, L1_CACHE_WB, L2_CACHE
ST	ST_RETIRED
L1D	L1D_CACHE, MEM_ACCESS
CORE	BR_IMMED_RETIRED, BR_PRED, BUS_CYCLES, CPU_CYCLES, L1I_CACHE, PC_WRITE_RETIRED, time execution, INST_RETIRED

Basic Paths Identified in our Populations

Name	Event
L2M	BUS_ACCESS, BUS_ACCESS_LD, L2_CACHE_REFILL
L2WB	BUS_ACCESS_ST, L2_CACHE_WB
LD	LD_RETIRED
L1M	L1D_CACHE_REFILL, L1_CACHE_WB, L2_CACHE
ST	ST_RETIRED
L1D	L1D_CACHE, MEM_ACCESS
CORE	BR_IMMED_RETIRED, BR_PRED, BUS_CYCLES, CPU_CYCLES, L1I_CACHE, PC_WRITE_RETIRED, time execution, INST_RETIRED

Population	Basic Paths
intAdd	CORE
readL1	CORE, LD, L1D
readL2	CORE, LD, L1D, L1M
readMem	CORE, LD, L1D, L1M, L2M
writeL1	CORE, ST, L1D
writeL2	CORE, ST, L1D, L1M
writeMem	CORE, ST, L1D, L1M, L2M, L2WB

Topology Extraction

BP abstraction

- Each population is represented as a set of Basic Paths (BP)

Relation analysis

- Analyze co-occurrences between BPs across populations
- Identify implication relations between BPs:

  A relation BPi -> BPj exists when:
  - BPi always appears with BPj
  - but BPj can appear alone

Topology construction

- Build a directed graph of BP relations
- Apply transitive reduction to simplify the graph

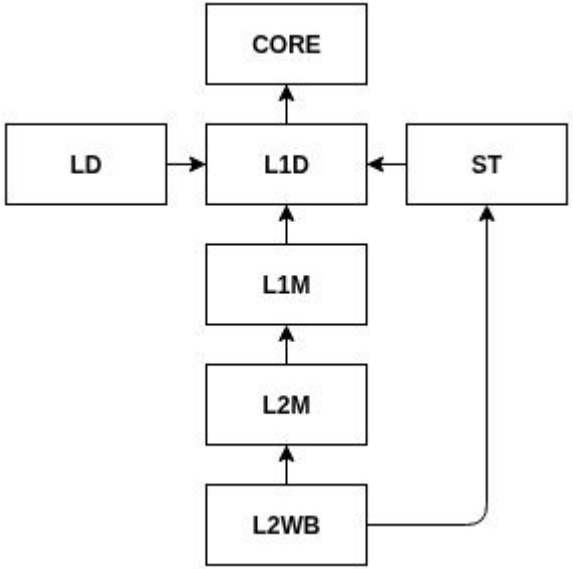


Fig: Topology extracted from the methodology