Combining Arm & RISC-V in Heterogeneous Designs

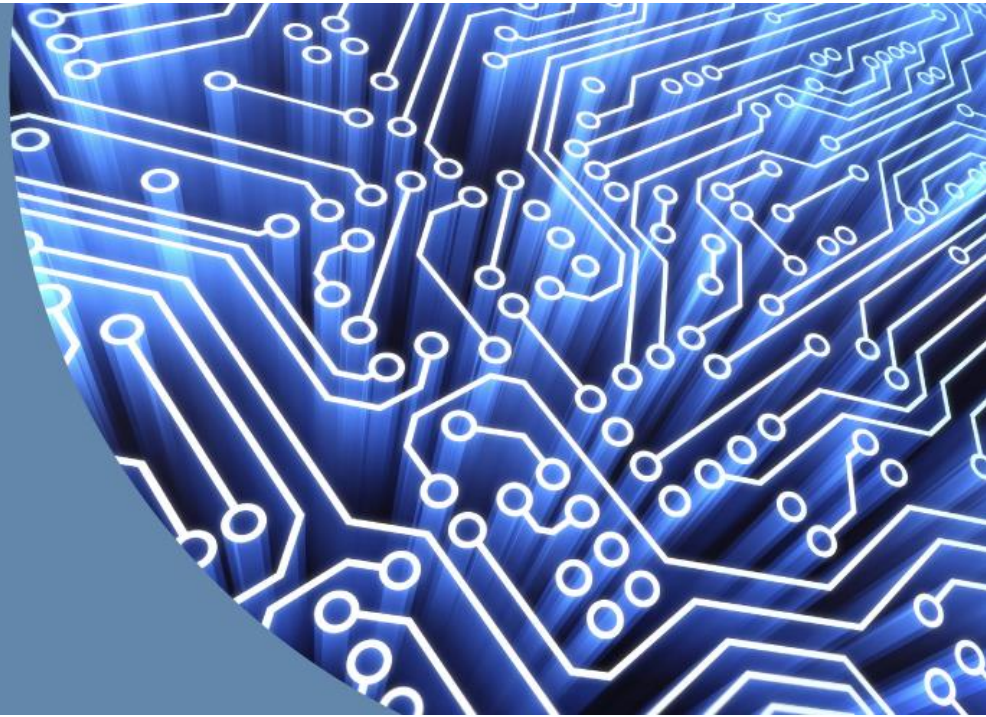
Gajinder Panesar, CTO, UltraSoC

gajinder.panesar@ultrasoc.com

RISC-V Summit

3 – 5 December 2018 – Santa Clara, USA

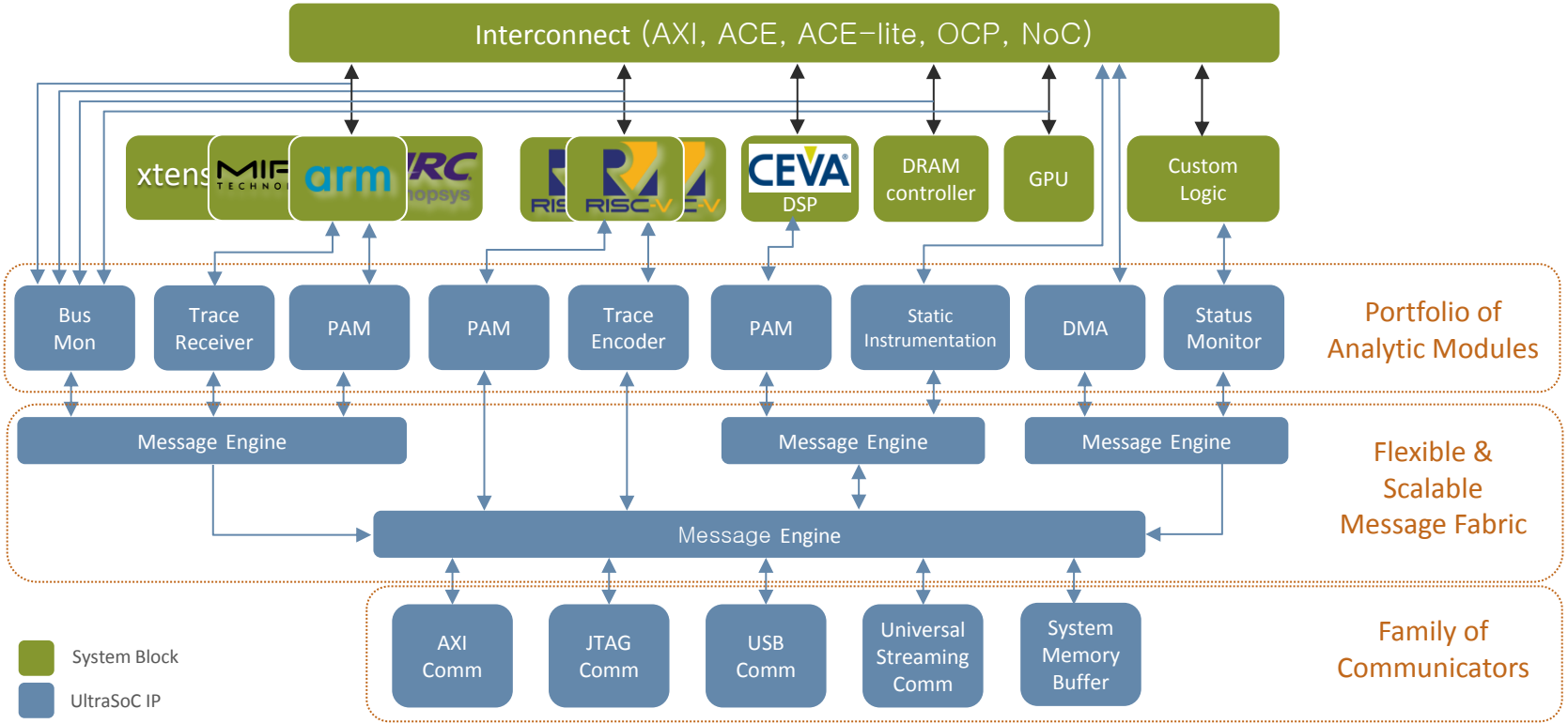
- Problem statement
- Deterministic multi-core
- Example scenarios
- In-field analysis/ML
- Summary



Problem statements

- It is not about the ISA(s)
- It is not about the core(s)
  - Compute is largely ‘solved’
- The challenge today is systemic complexity, for example:
  - Ad-hoc programming paradigms
  - Processor-processor interactions
  - HW/SW interactions
  - Interconnect, NoC & deadlock
  - System not architected

Advanced Debug/Monitoring for the Whole SoC



UltraSoC

- A coherent architecture to debug, monitor and provide rich data for run-time analytics
  - Silicon IP is highly parameterizable - allows customers to trade hardware resources and thus silicon area
  - Hardware resources are configurable at runtime
  - Allows reuse of hardware resources for different scenarios and different algorithms
  - Help with security and safety of systems
  - Hardware provides data so CPU load is small

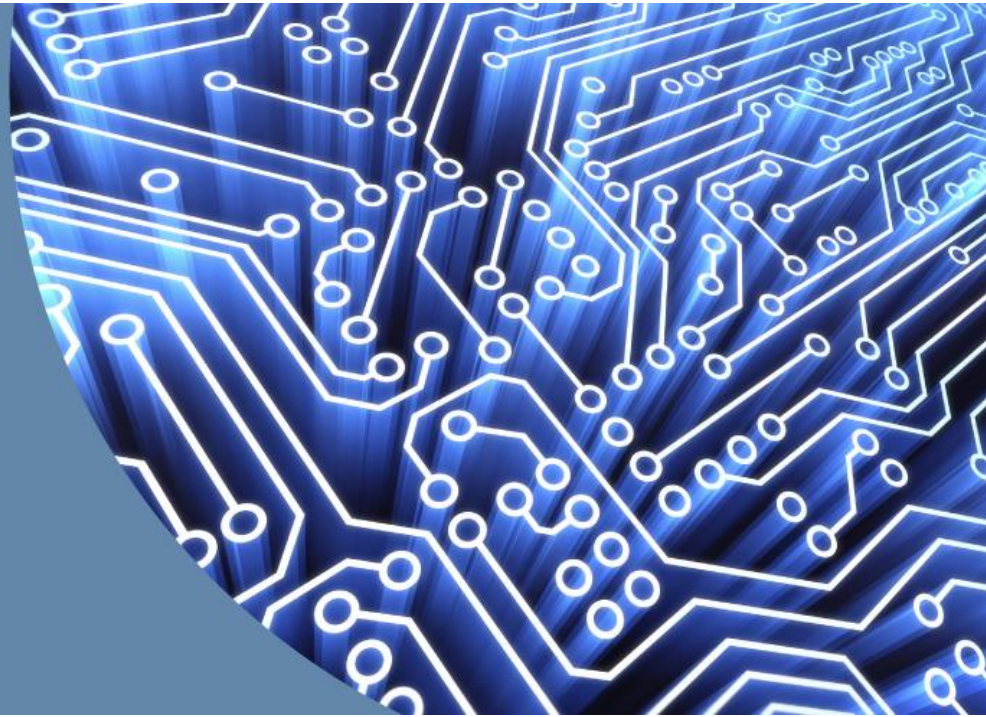
Problem statement

Deterministic multi-core

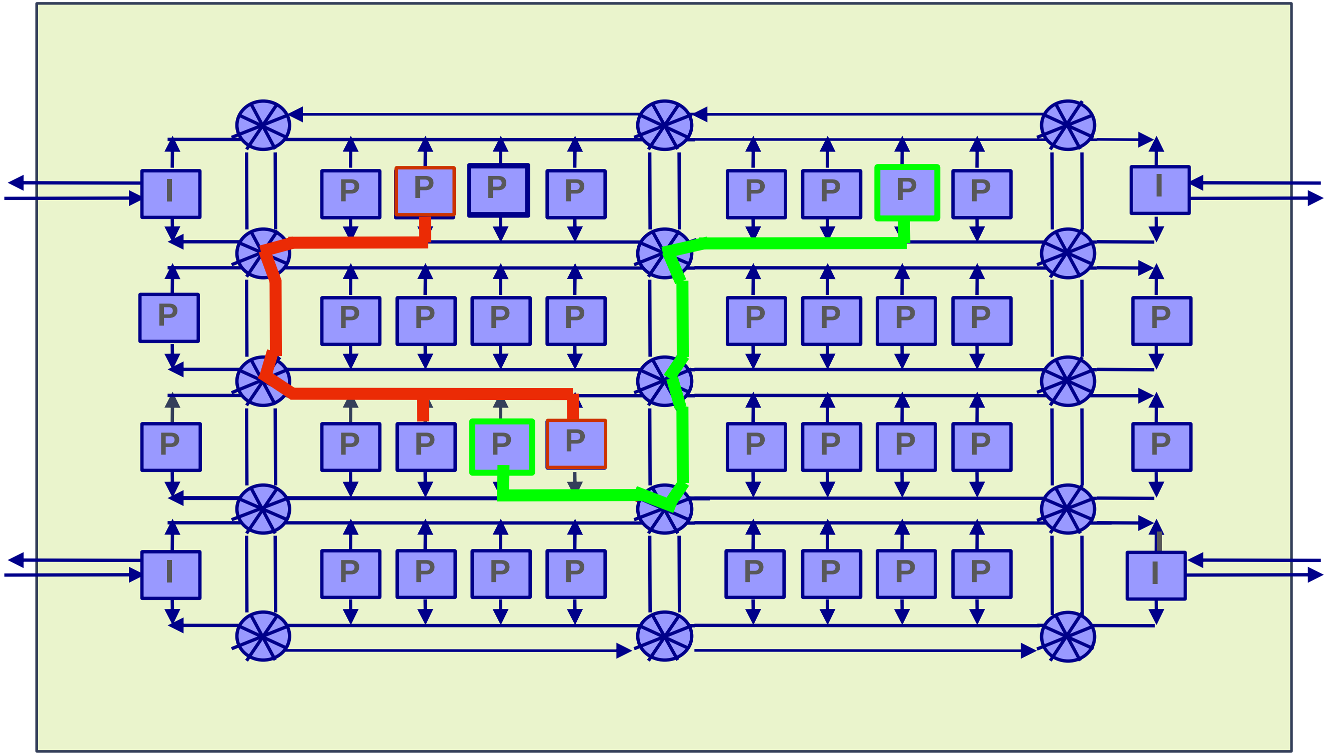
Example scenarios

In-field analysis/ML

Summary

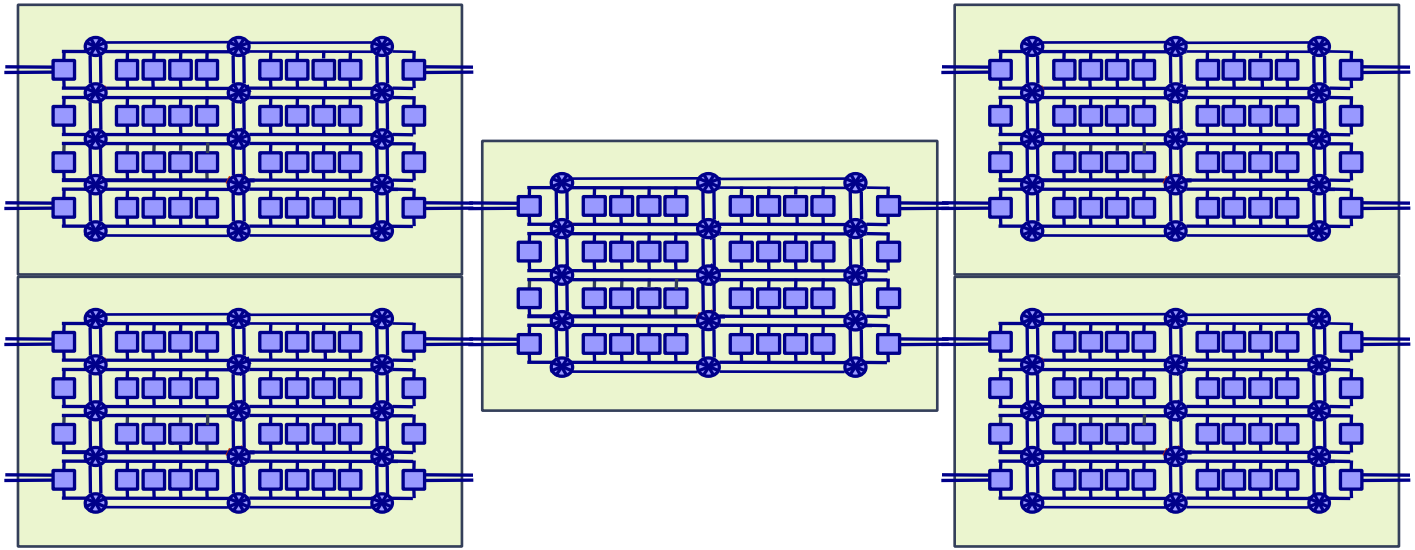


Deterministic multi-core system



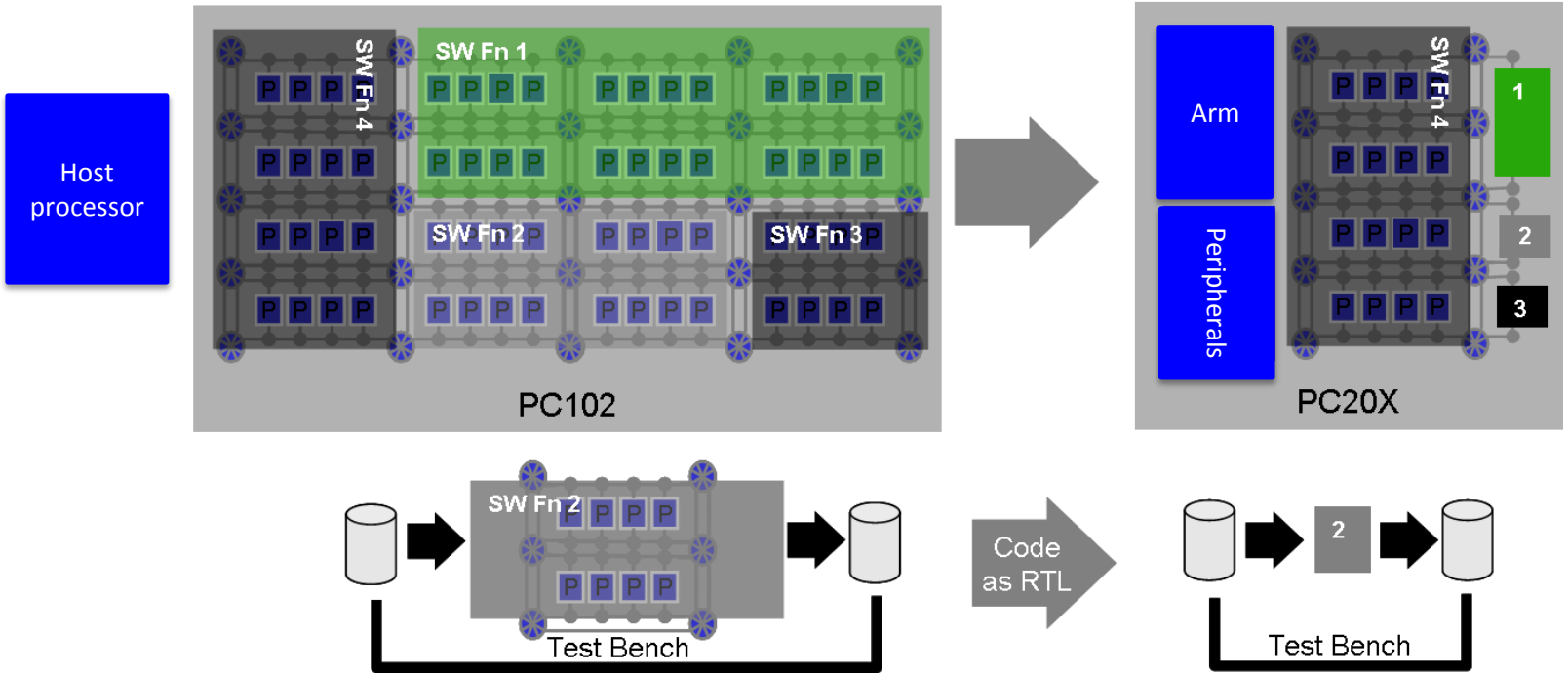
picoArray concept, circa 2000

Deterministic multi-die, multi-core system



picoArray concept, circa 2000

Hardware accelerator flow



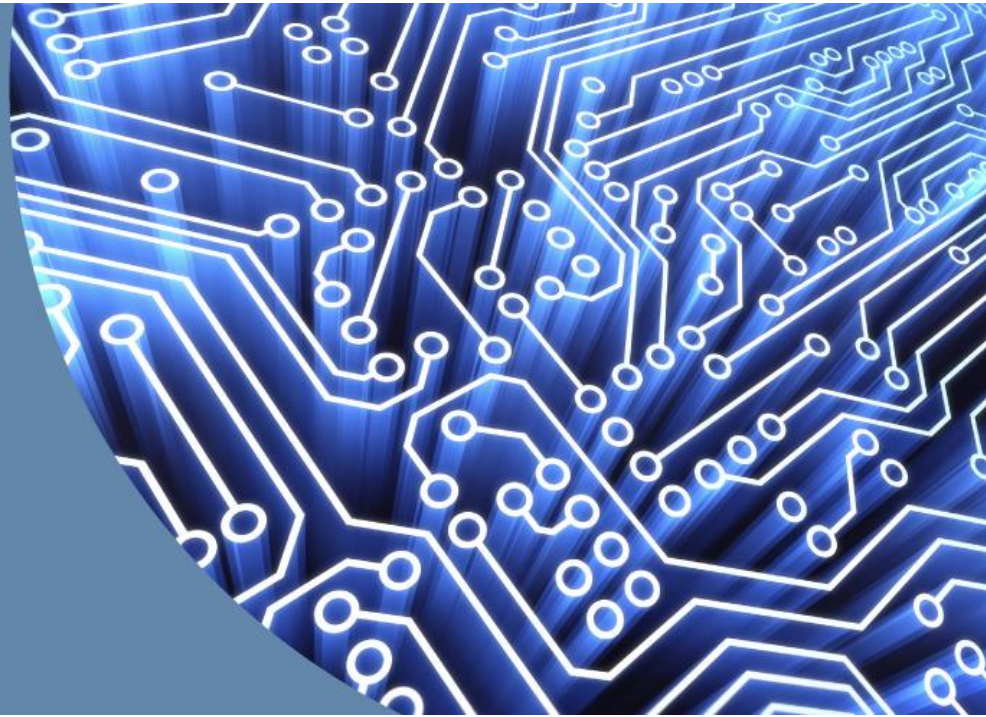
Problem statement

Deterministic multi-core

Example scenarios

In-field analysis/ML

Summary

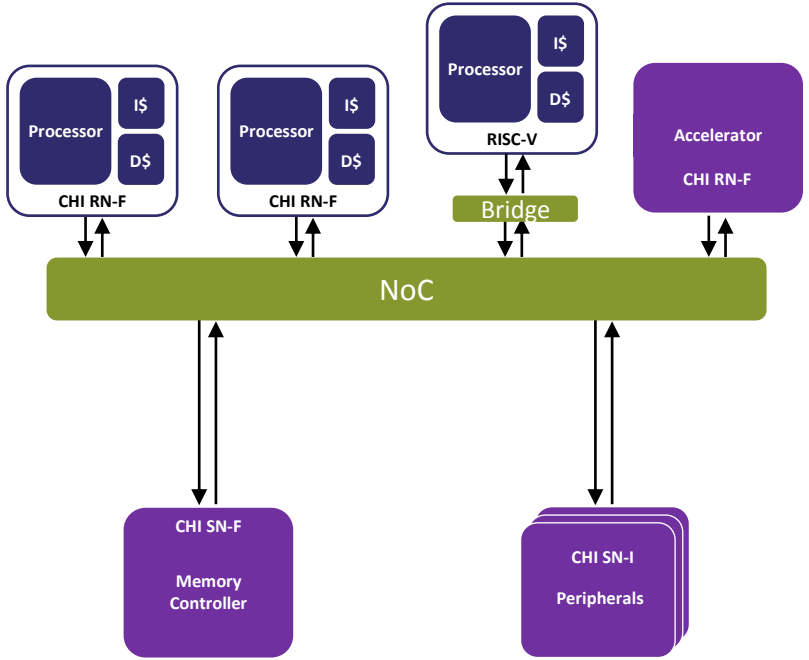


Requirements on tools

- There is a need for heterogeneous architectural and modelling exploration systems
  - Be able to feed in run-time system data to close the loop
- There is a need for true heterogeneous core tool chain
- This is especially true for debugging tools
  - Open source tools such as GDB and OpenOCD need to handle this in an efficient manner
  - Strangely, not everyone likes or wants open source tools – it is True!
  - Need one cockpit for the different cores in a system
- Need to have tools that help with run-time visibility
  - These need to have open APIs
- As complexity continues to increase, need means of autonomous analysis

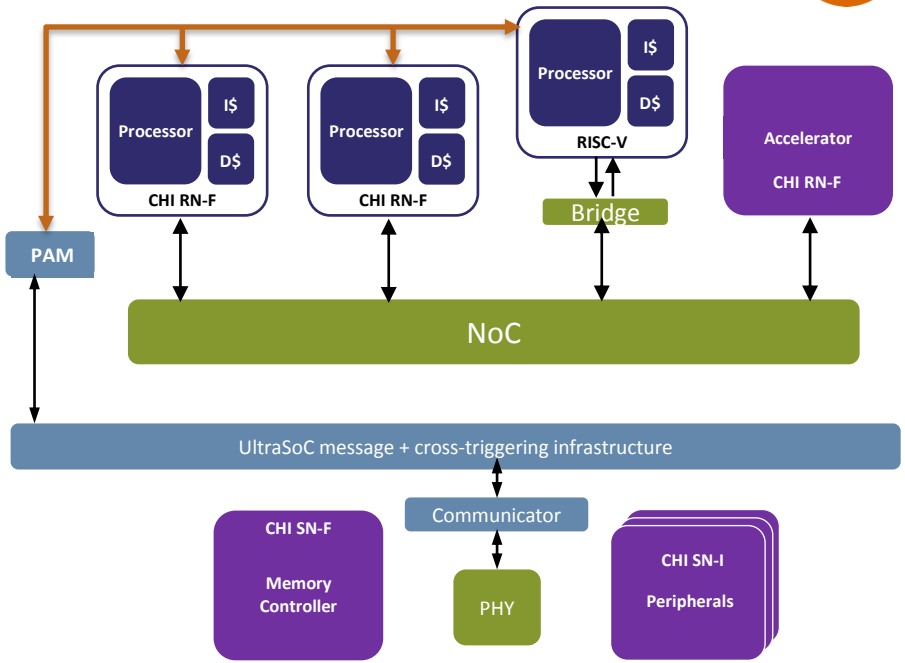
Typical SoC with network on chip

- Multiple processors with fully coherent caches
- I/O coherent accelerator
- Shared memory controller
- NoC could be ring, mesh or crossbar

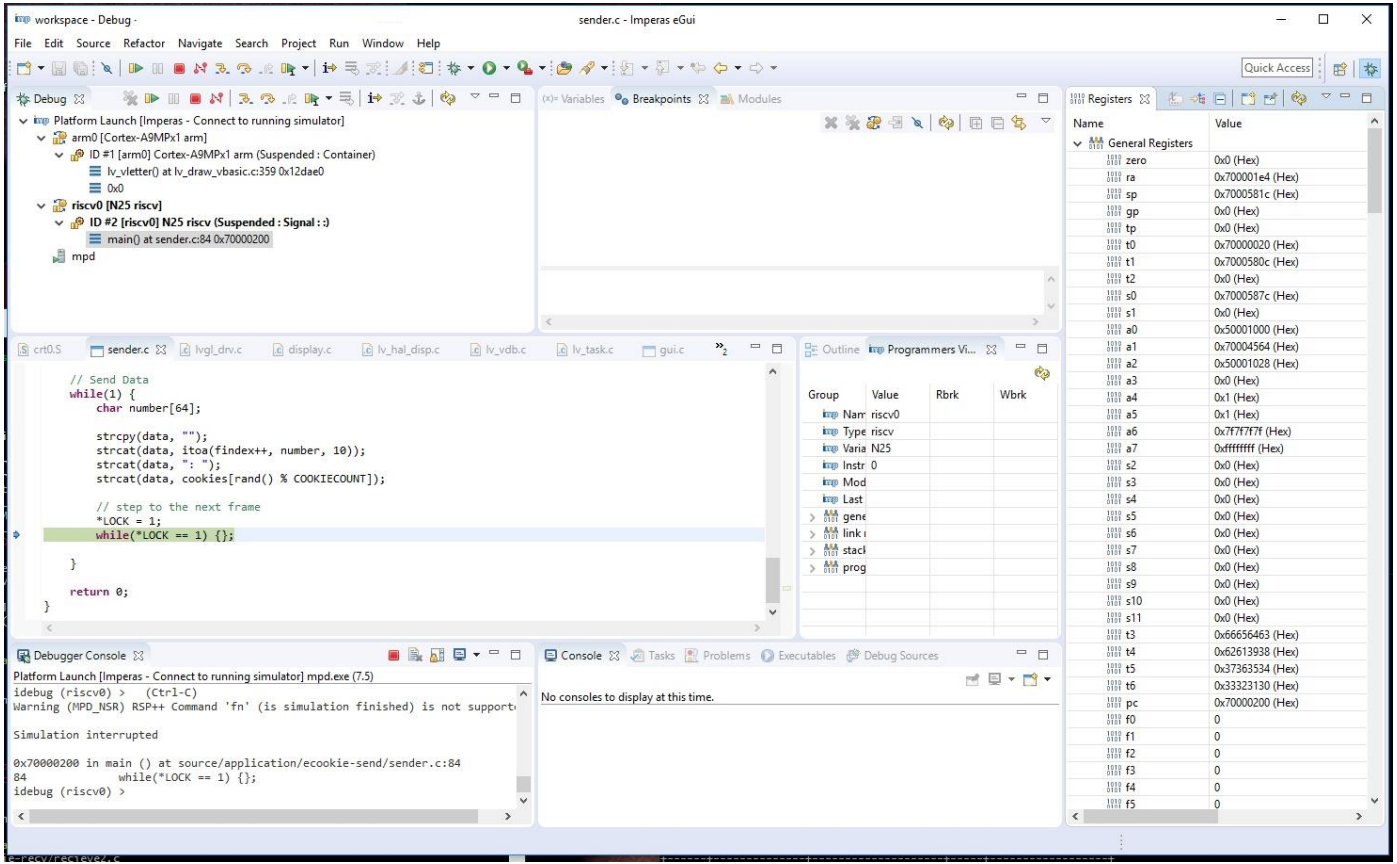


Unified run-control with UltraSoC interconnect

- Separate from system interconnect
- Message-based
- Used for both configuration (in) and diagnostic reporting (out)
- Integrated real-time event broadcast for cross-triggering

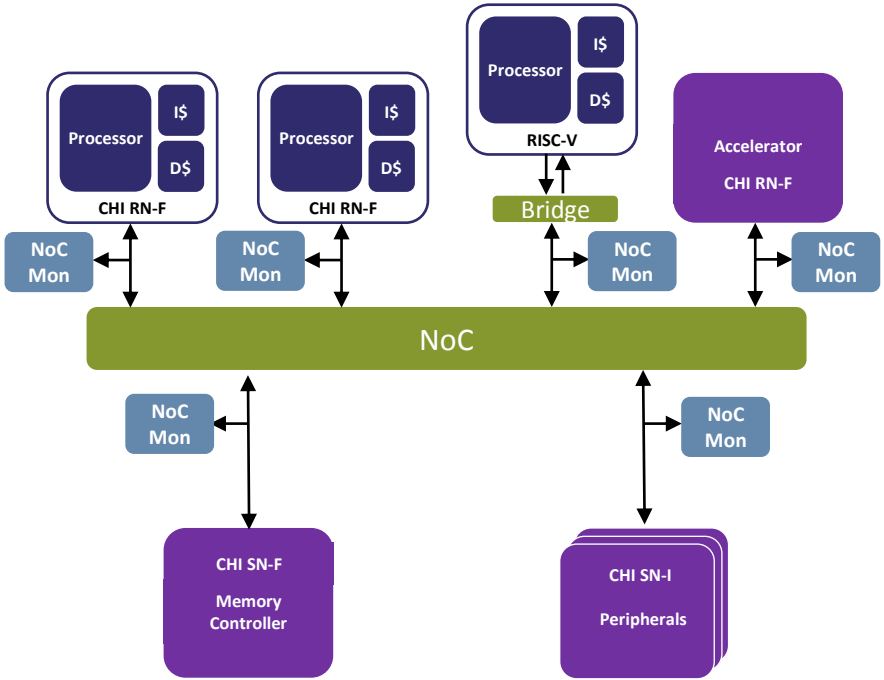


Heterogeneous run-control



NoC boundary monitoring

- UltraSoC NoC monitors on connections to the NoC
- Monitors are fully transaction aware



Example: “deadlock detection”

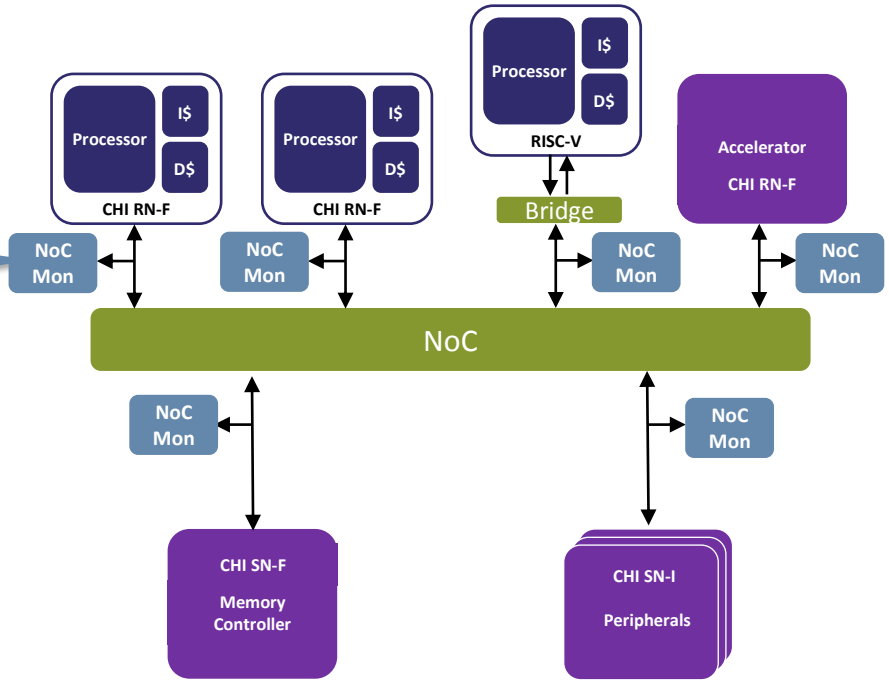
- Automatically detect deadlock on the NoC

Trace all traffic into circular buffer within NoC monitor

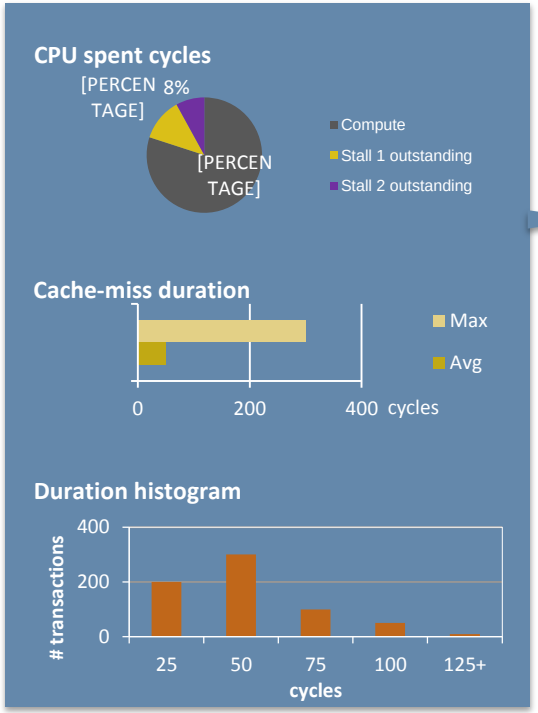
Trigger trace if transaction duration exceeds threshold (e.g. 5k cycles)

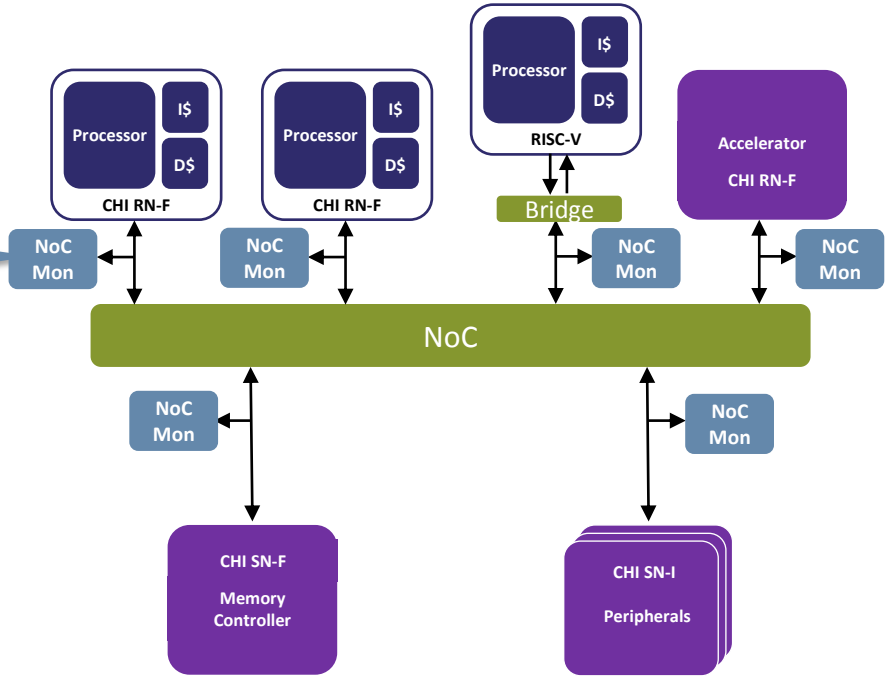
Stop tracing and output full details of deadlocked transaction and those immediately preceding it

Nothing sent off-chip until deadlock occurs

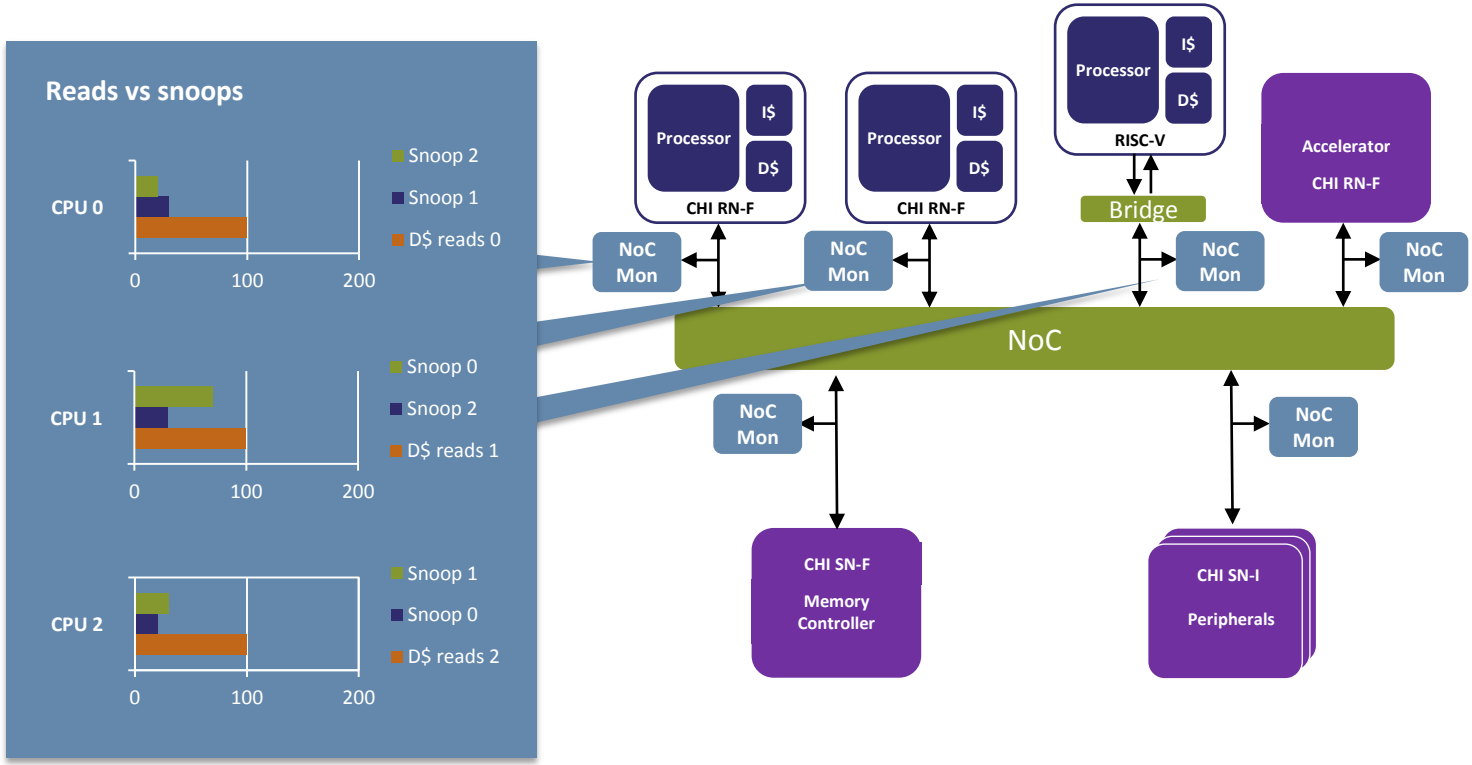


Example: “where have my MIPs gone?”



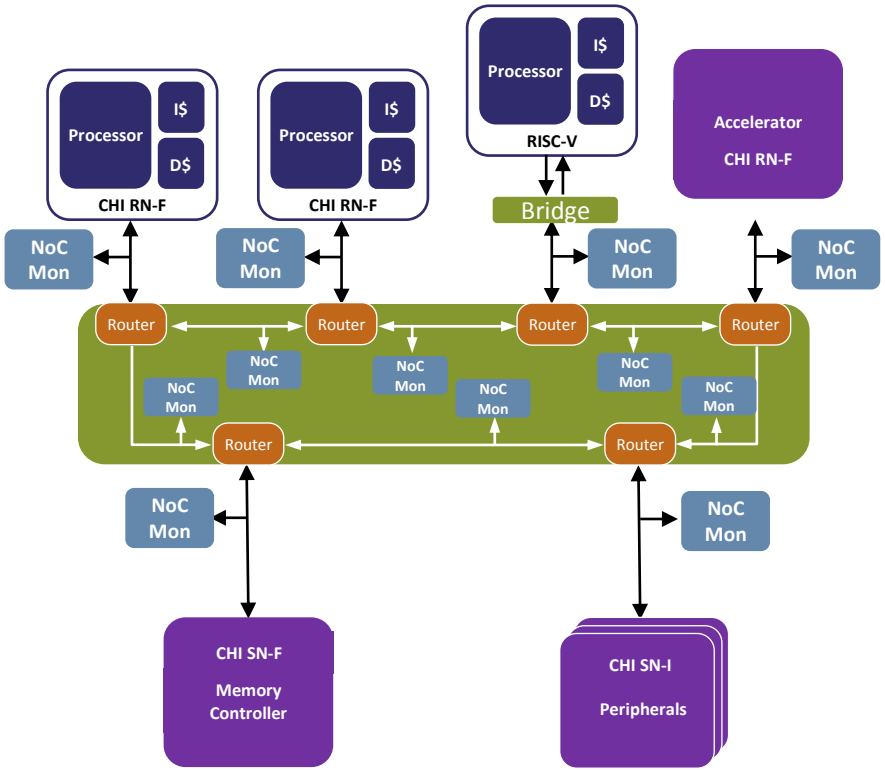


Example: “how effectively is data shared?”



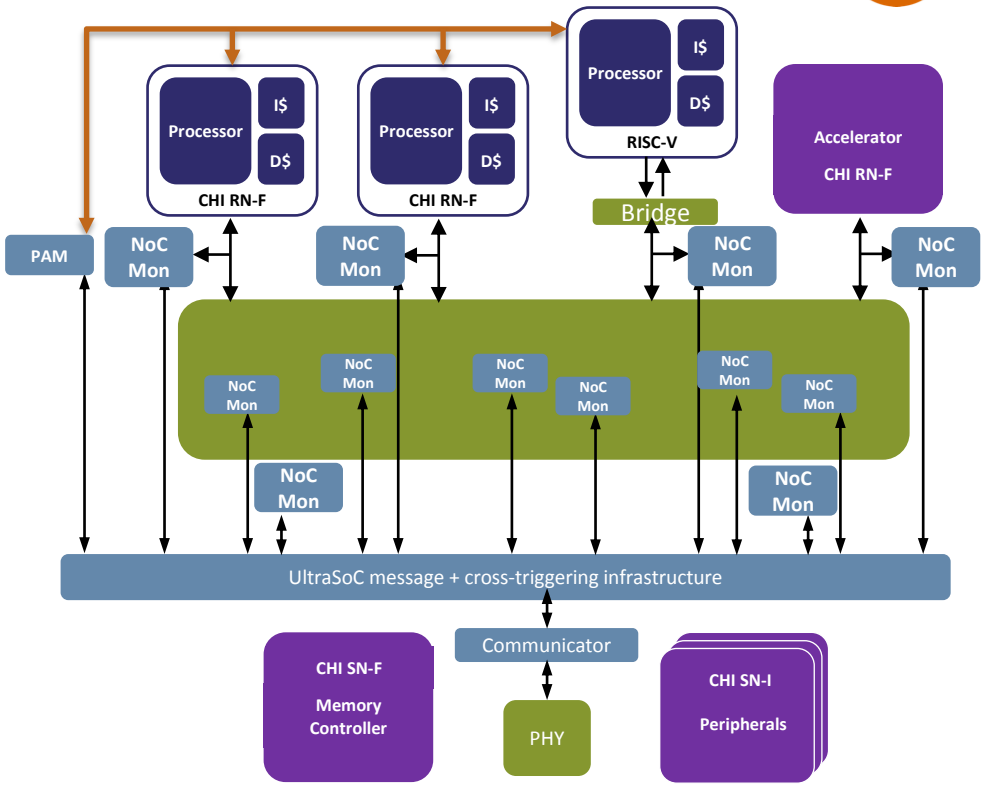
Internal NoC monitoring

- Example shows ring topology
- Intra-router monitors are typically not transaction aware
  - Monitor individual channels only
  - Simpler

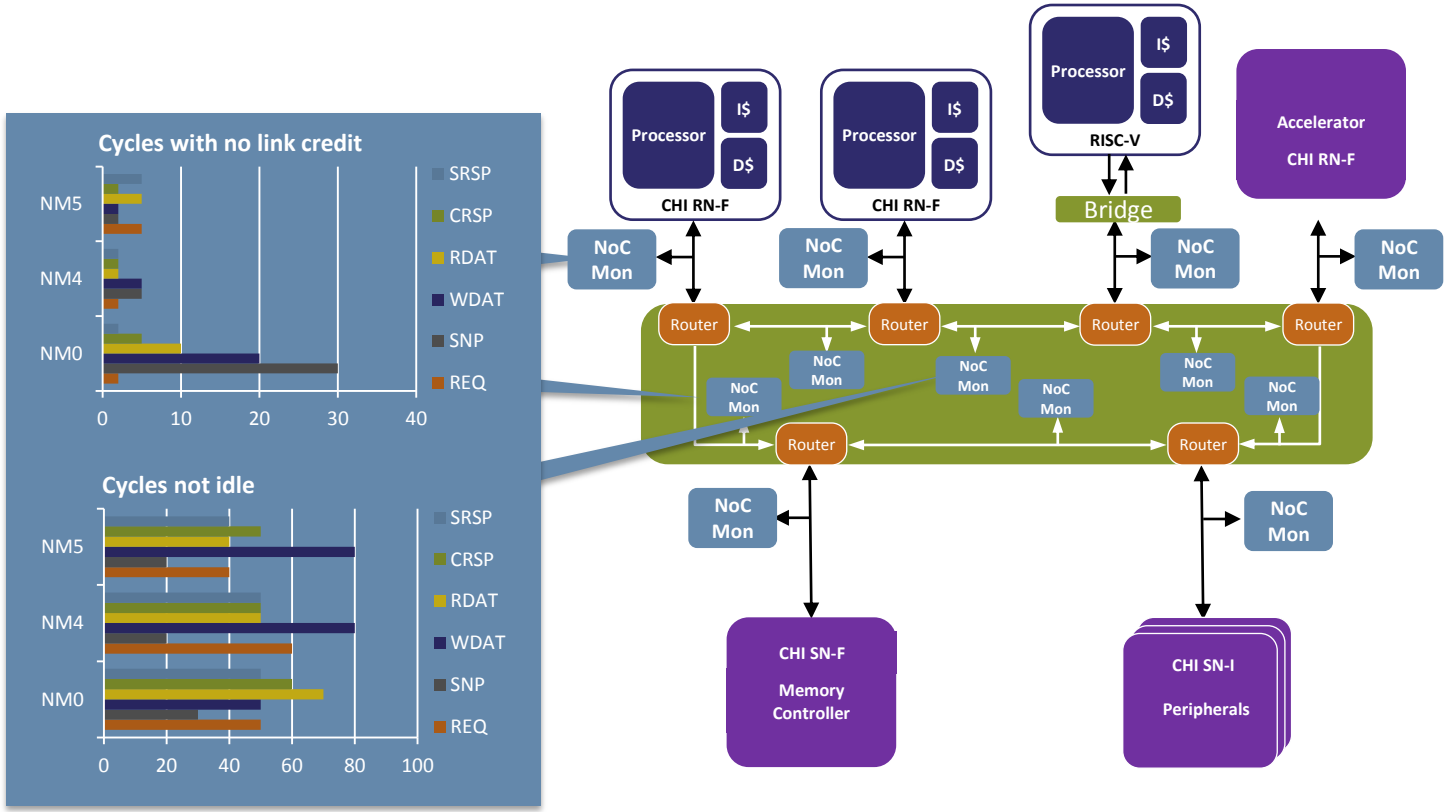


Independent, orthogonal UltraSoC interconnect

- Separate from system interconnect
- Message-based
- Used for both configuration (in) and diagnostic reporting (out)
- Integrated real-time event broadcast for cross-triggering



Example: “is the NoC oversubscribed or unbalanced?”



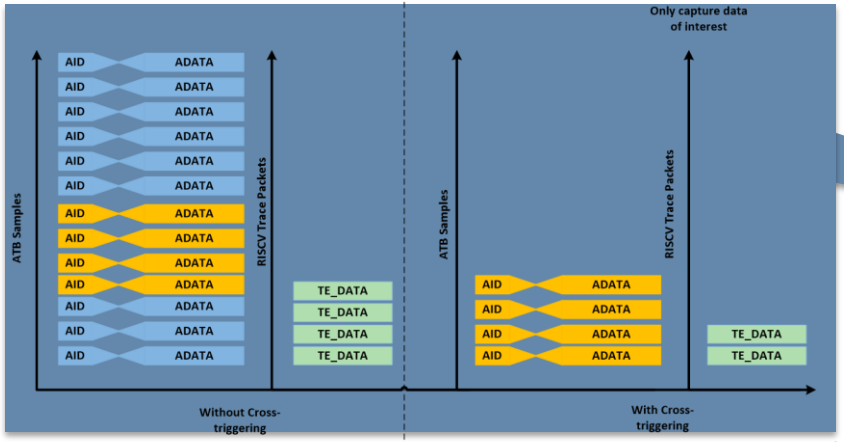
Example: “deadlock – with cross-triggering”

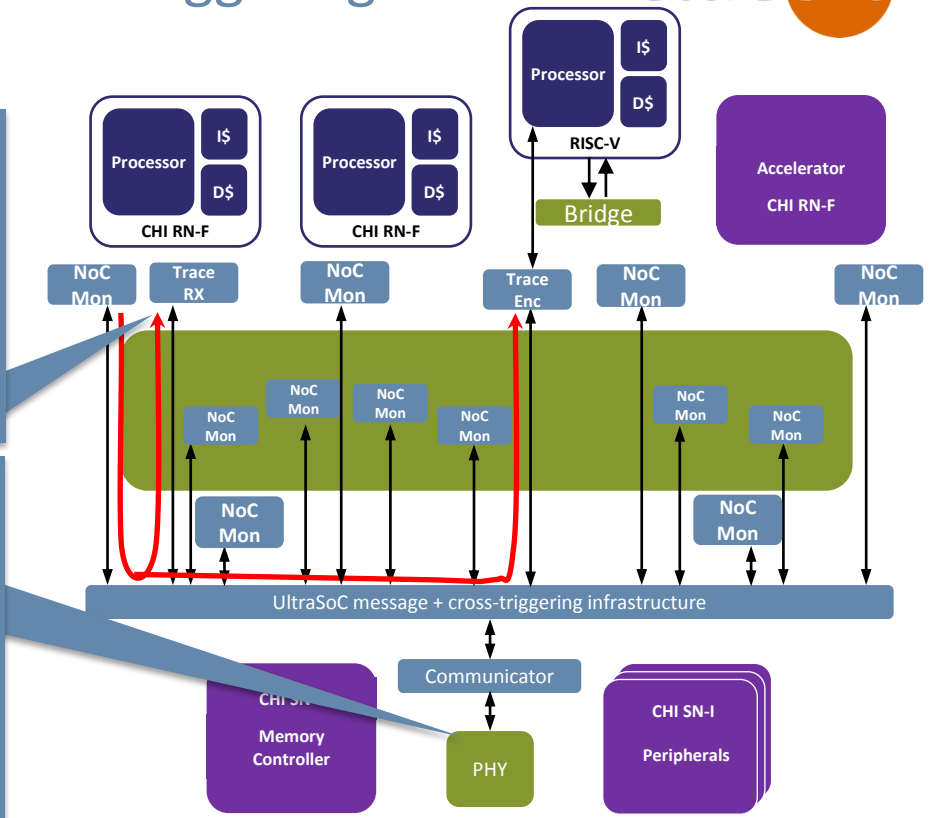
Trace Receiver module

Captures processor trace (e.g. from ARM ETM) into circular buffer

RISC-V trace captured in circular buffer

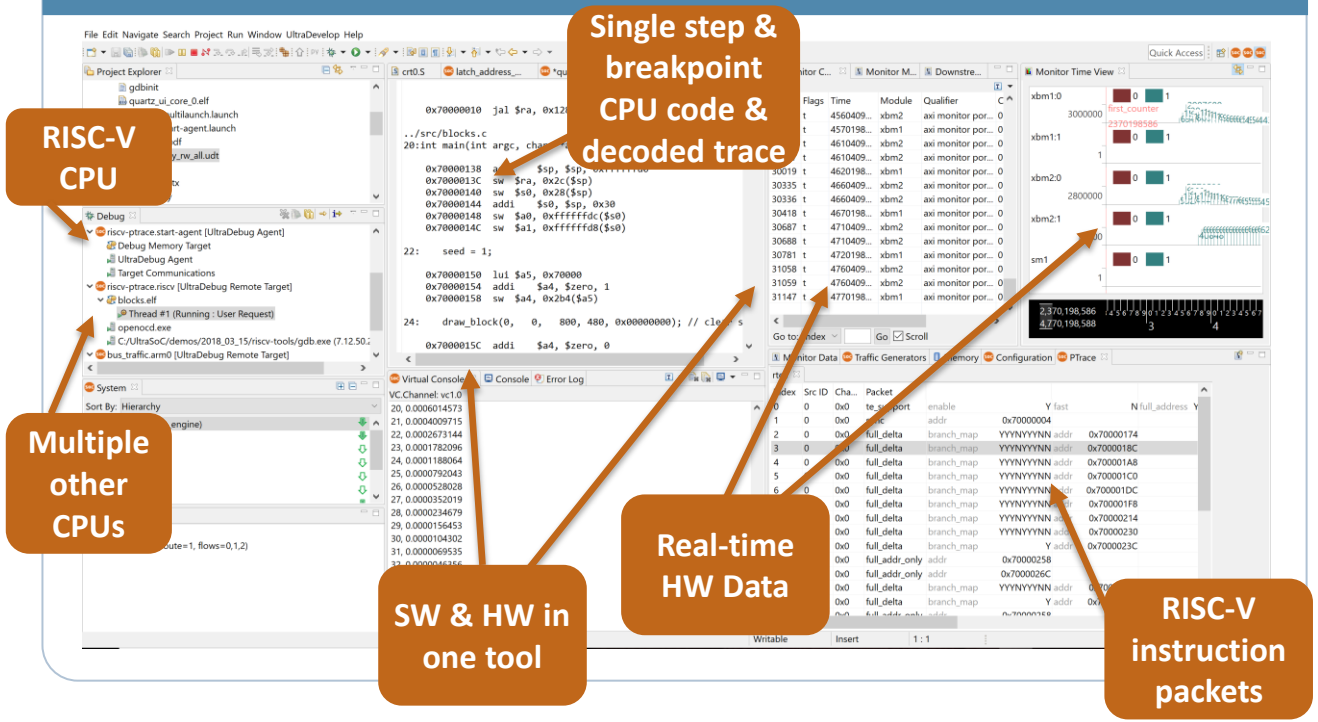
NoC monitor trace trigger also cross-triggers Trace Receiver and RISC-V Trace Encoder



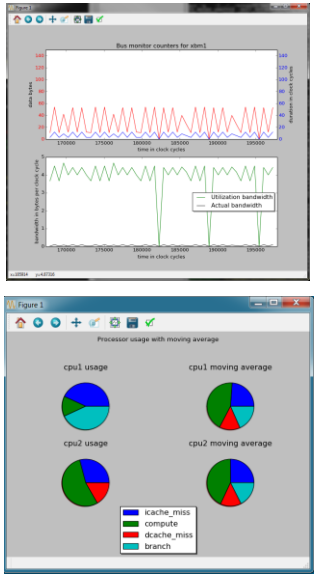


Software tools for data-driven insights

Eclipse based UltraDevelop IDE



Script based



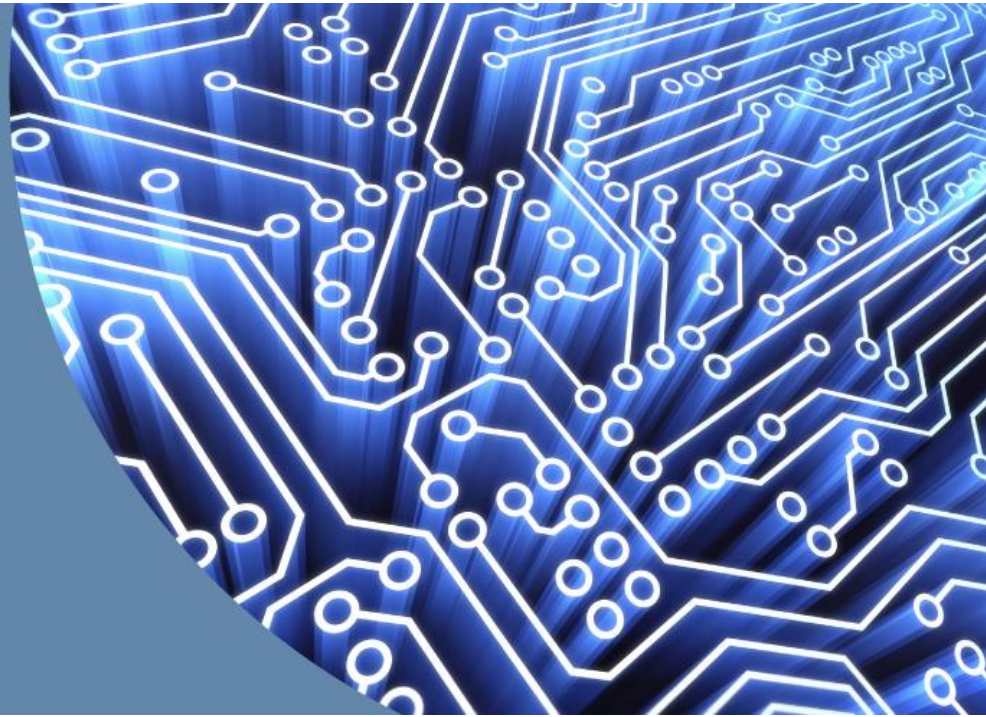
Problem Statement

Deterministic multi-core

Example scenarios

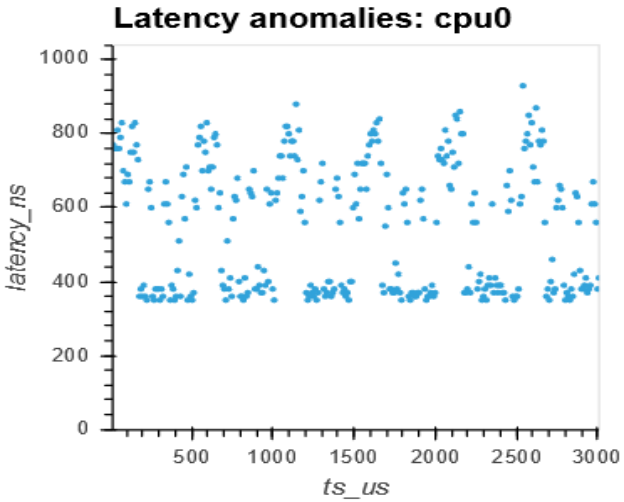
In-field analysis/ML

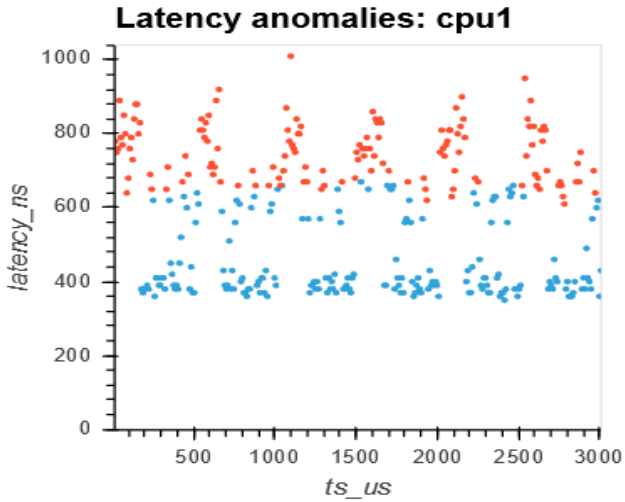
Summary

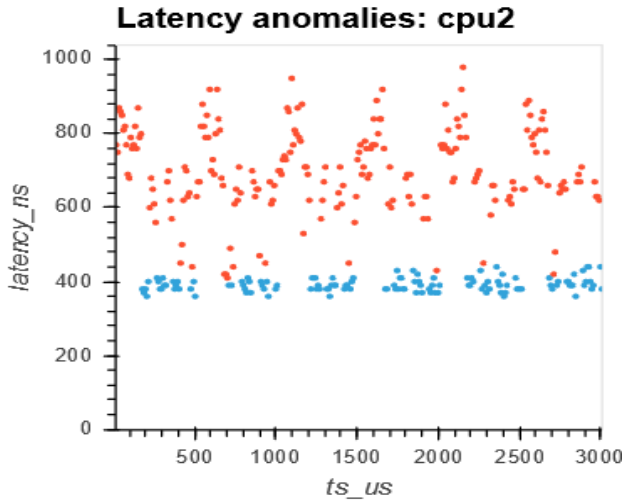


Non intrusive anomaly detection

- Three CPU plots below show CPU cache-like traffic for 3 CPUs configured with different miss rates
- Excessive (anomalous) latencies are shown in red

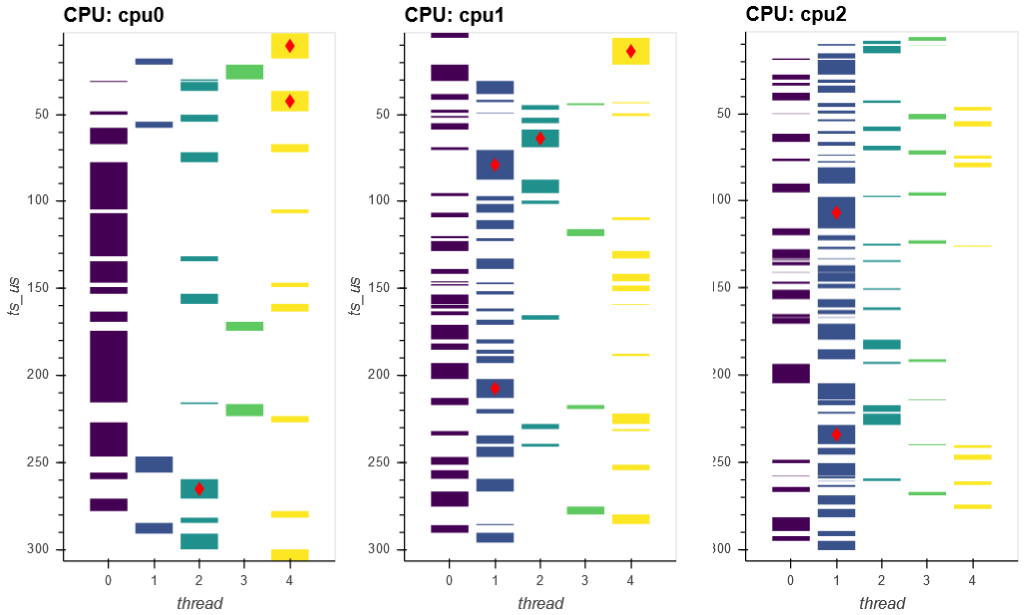






Non-intrusive profiling with anomaly detection

- Traditional profilers are inadequate:
  - Sampling = miss subtle or fast events (Nyquist)
  - Performance impact/intrusive
  - “Heisenbugs”
- UltraSoC is non-intrusive
- UltraSoC is wirespeed (100% coverage)
- Analytics and automated anomaly detection to make engineer more efficient



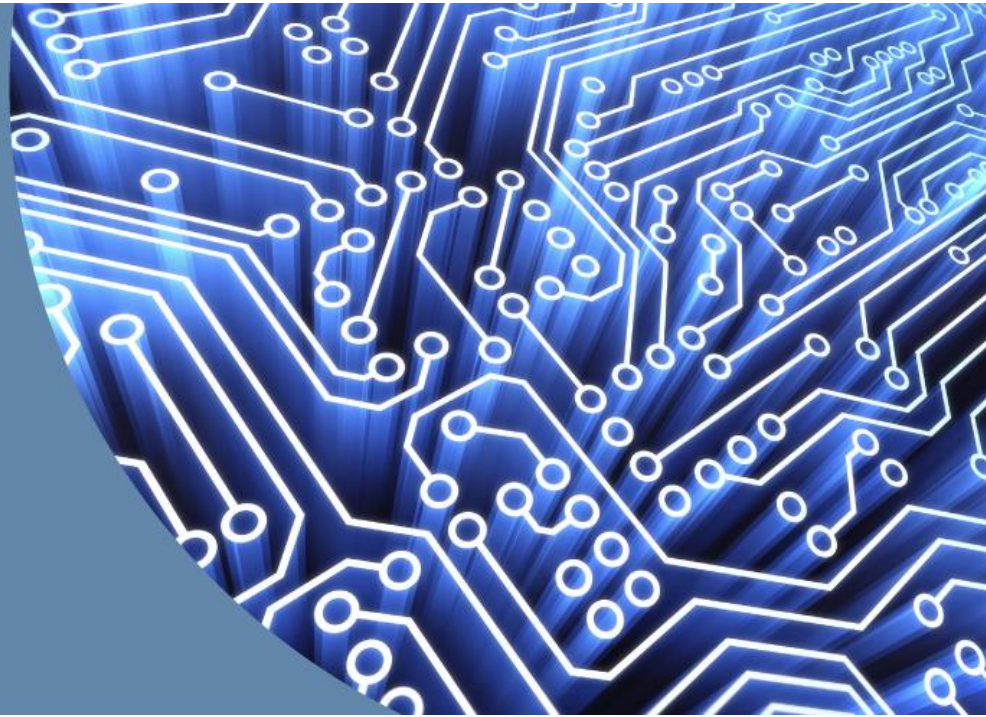
Problem Statement

Deterministic multi-core

Example scenarios

In-field analysis/ML

Summary



Summary

- The challenge today is systemic complexity
  - Architectural and modelling is needed but not enough
- Need tools that support true heterogenous systems
  - Both open source and commercial
- In addition to run-control, need non-intrusive monitoring
- More complex systems will require autonomous analytics and causality detection
- UltraSoC provides all or is working on all these