NoCAlert: An On-Line and Real-Time Fault Detection Mechanism for Network-on-Chip Architectures

Andreas Prodromou, Andreas Panteli,
Chrysostomos Nicopoulos, Yiannakis Sazeides

Presenters: Babak Zamirai, Jiecao Yu

EECS 573

Sep 28th, 2015

Outline

- Necessity of Networks-on-Chip (NoCs)
- Reliability and NoCs
- The NoCAlert Approach: Invariance Checking
- Identifying Invariances and Examples
- Evaluation
- Results
- Conclusion

Outline

- Necessity of Networks-on-Chip (NoCs)
- Reliability and NoCs
- The NoCAlert Approach: Invariance Checking
- Identifying Invariances and Examples
- Evaluation
- Results
- Conclusion

Core Number Increases

~~Increase clock frequency~~

~~Instruction-Level Parallelism (ILP)~~

→ More cores to take advantage of specified parallelism

Intel 4004

4-bit

Intel Pentium 4

1 Core

…

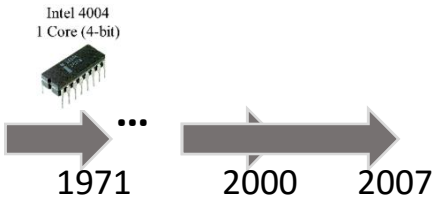
1971 2000

*Pictures from author

Core Number Increases

Intel Core 2 Duo

Intel Pentium 4



*Pictures from author

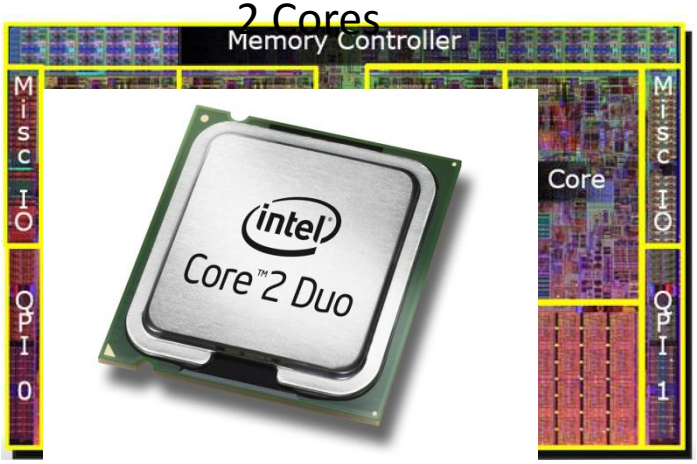
Core Number Increases

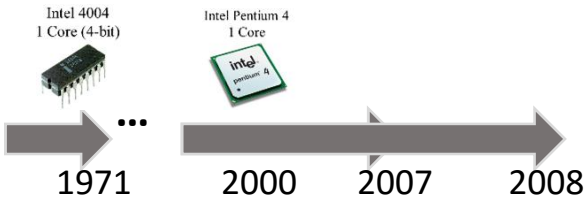
Intel Core i7 (Nehalem)

Intel Core 2 Duo

4 Cores

2 Cores

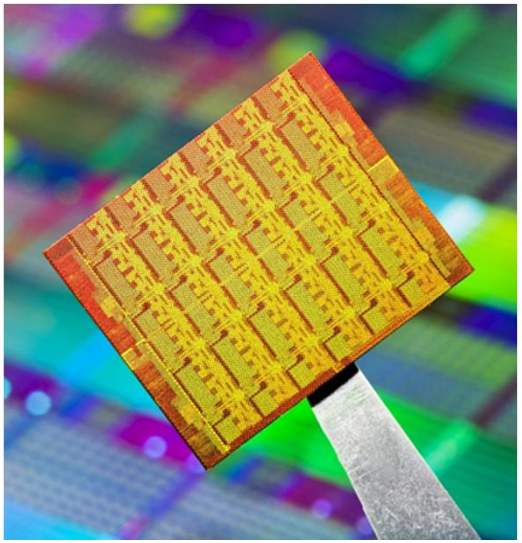




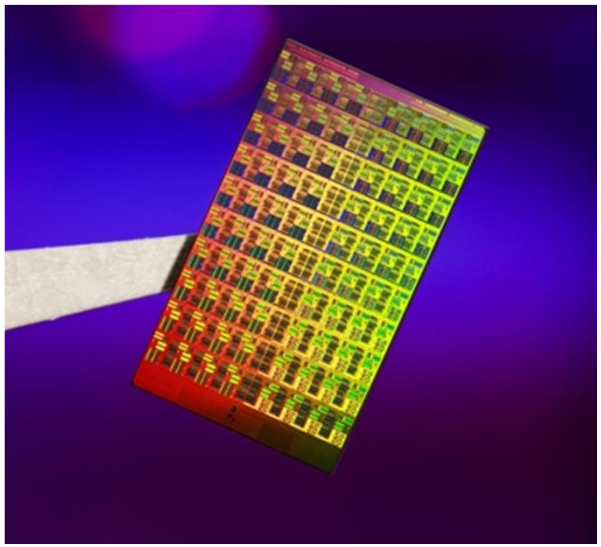
*Pictures from author

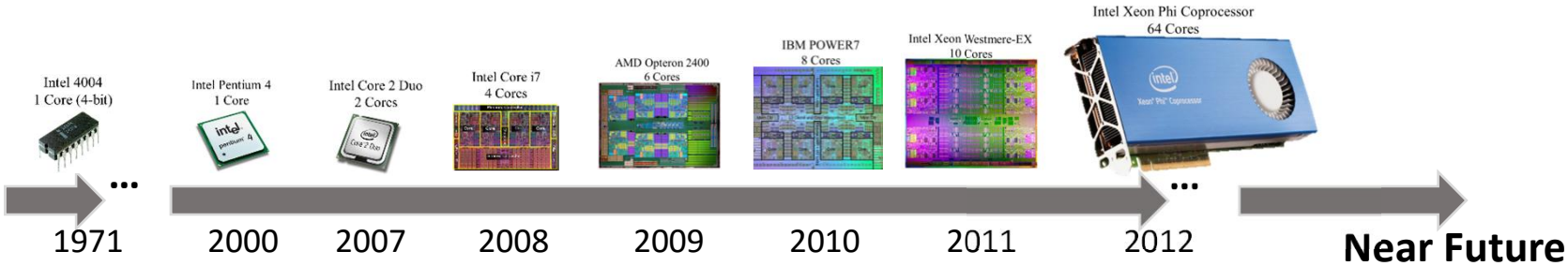
Core Number Increases

Intel Single-Chip Cloud Computer
48 Cores



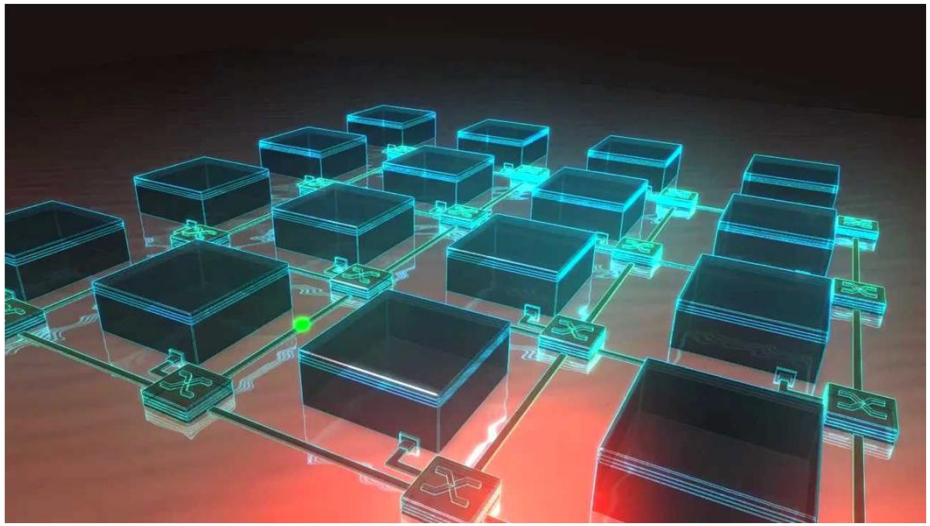
Intel Polaris Chip
80 Cores





*Pictures from author

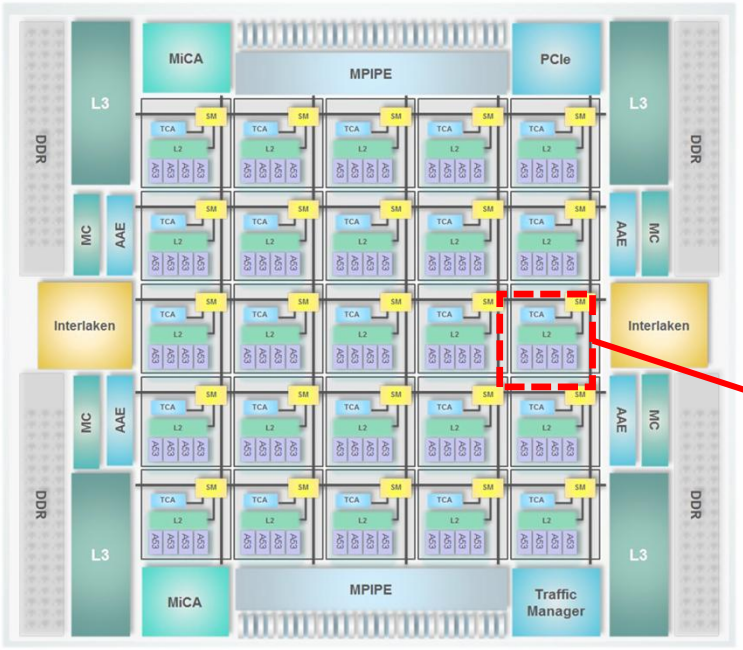
Network-on-Chip (NoC)



- On-chip interconnection network to connect all nodes
  - Packet-based communication
- Node: router + processing element
- Modular design - structured interconnect layout
- *Scalable* and *efficient*

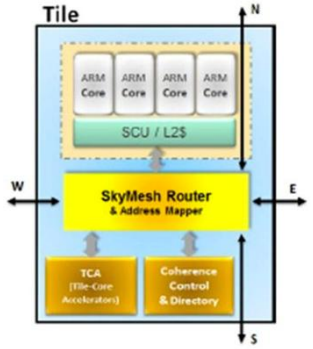
It's already happening!

- NoCs are becoming necessary
- Router is becoming part of the core design



Tilera-MX™– 100 Cores

- 2D mesh NoC comprising
- 25Tbps of aggregate bandwidth
  - SkyMesh protocol



Outline

- Necessity of Networks-on-Chip (NoCs)
- Reliability and NoCs
- The NoCAlert Approach: Invariance Checking
- Identifying Invariances and Examples
- Evaluation
- Results
- Conclusion

NoC Reliability

- A single fault in the NoC can cause:
  - Network disconnections
  - Deadlocks (Network and Protocol-level)
  - Lost packets
  - Degraded performance

  → A *single fault* can disable the entire system (CMP)

- Protecting the NoC is of prime importance

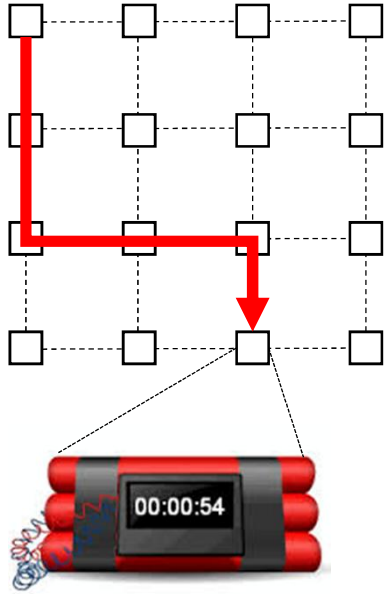
NoC Protection

- Fault recovery
  - ECC: inter-router faults
  - Bulletproof: online repair and recovery
- Fault detection
  - Test vectors / BIST
  - Boot-up only or interrupt at running time
- ForEVeR framework
  - Checker network
  - Counter in destination node reaches zero at least once

  ✗ False positive in fault-free environment

  ✗ Epoch duration sensitive to traffic

  ✗ Delayed fault detection



Outline

- Necessity of Networks-on-Chip (NoCs)
- Reliability and NoCs
- The NoCAlert Approach: Invariance Checking
- Identifying Invariances and Examples
- Evaluation
- Results
- Conclusion

NoCAlert: The Big Picture

- Distributed invariance checkers
  - Invariance violation
- Network's operation never interrupted
  - Online checking
- Almost instantaneous fault detection
- Against faults in the control logic
  - Intra- / Inter- router faults
  - Packet/flit contents are protected with ECC

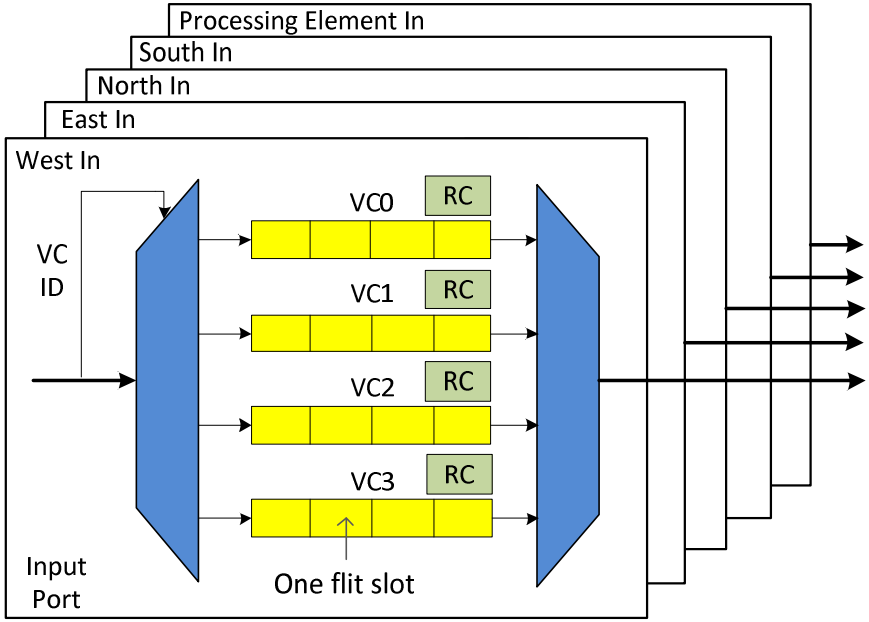
Invariance Checking

- Invariance: fundamental functional rules within the context of a component's operation
- Checks for legality, not correctness
  - Legality: illegal is an output that is impossible to occur
  - Erroneous but legal module outputs are always benign
- Emulates assertions used in software
  - assert(X!=5)
  - In hardware this would be achieved with a comparison unit that raises a flag

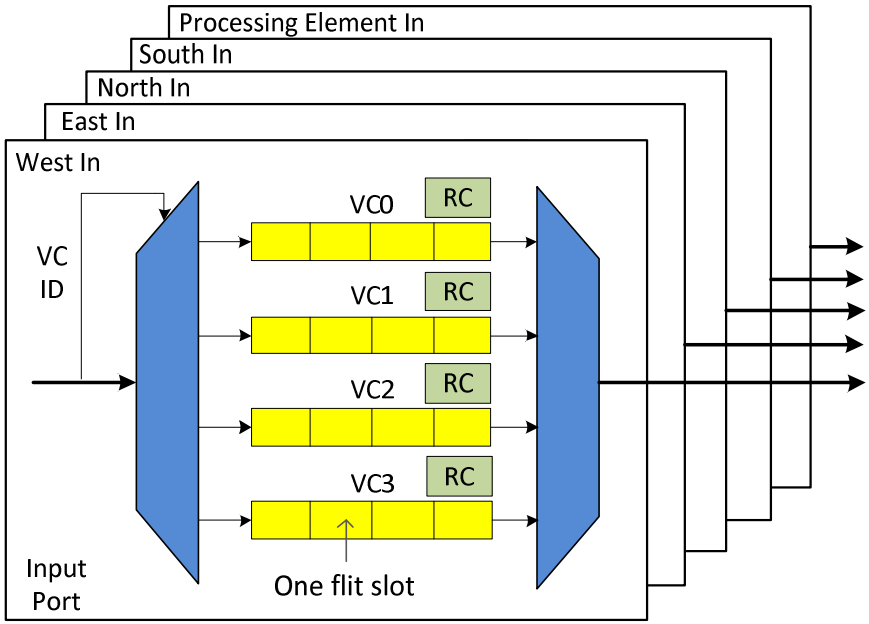
Outline

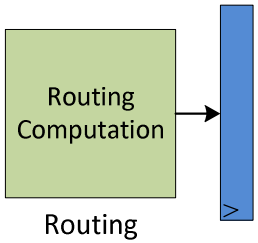
- Necessity of Networks-on-Chip (NoCs)
- Reliability and NoCs
- The NoCAlert Approach: Invariance Checking
- Identifying Invariances and Examples
- Evaluation
- Results
- Conclusion

Typical NoC Router Micro-Architecture



Typical NoC Router Micro-Architecture

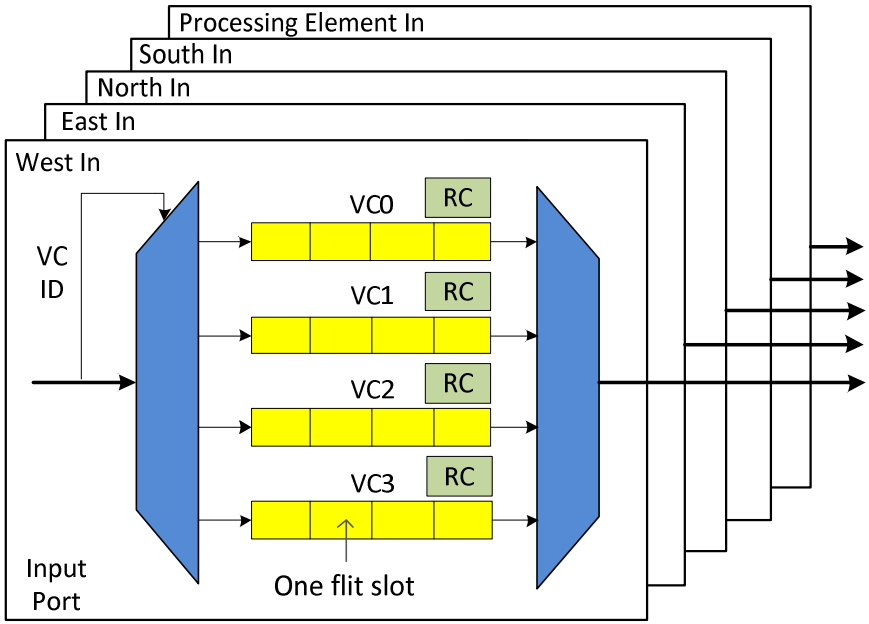


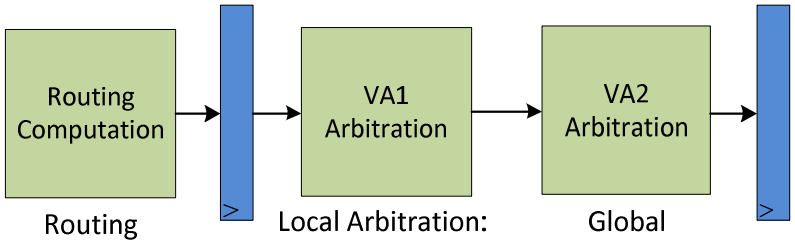


Routing Computation: **Next-hop direction**

Router Pipeline

Typical NoC Router Micro-Architecture





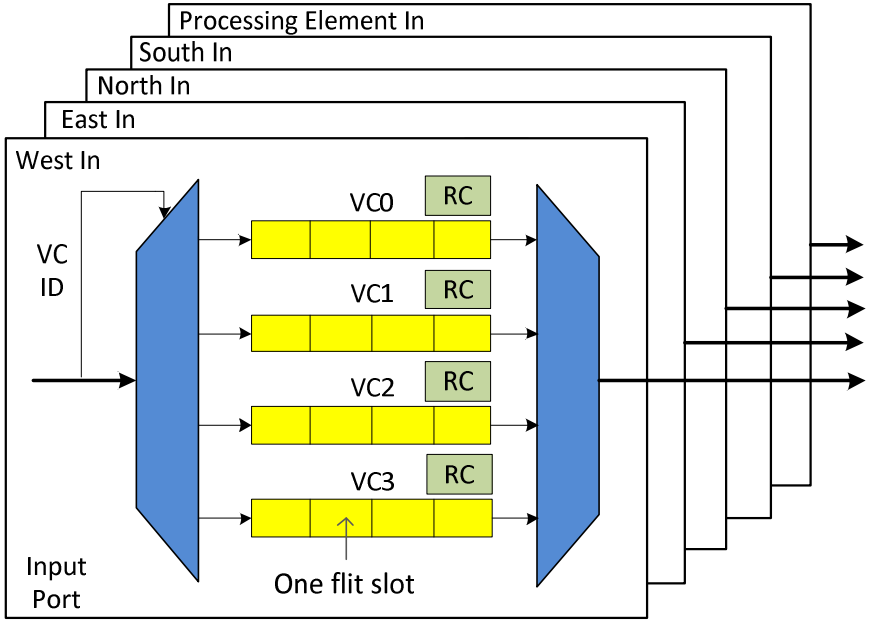
Routing Computation: **Next-hop direction**

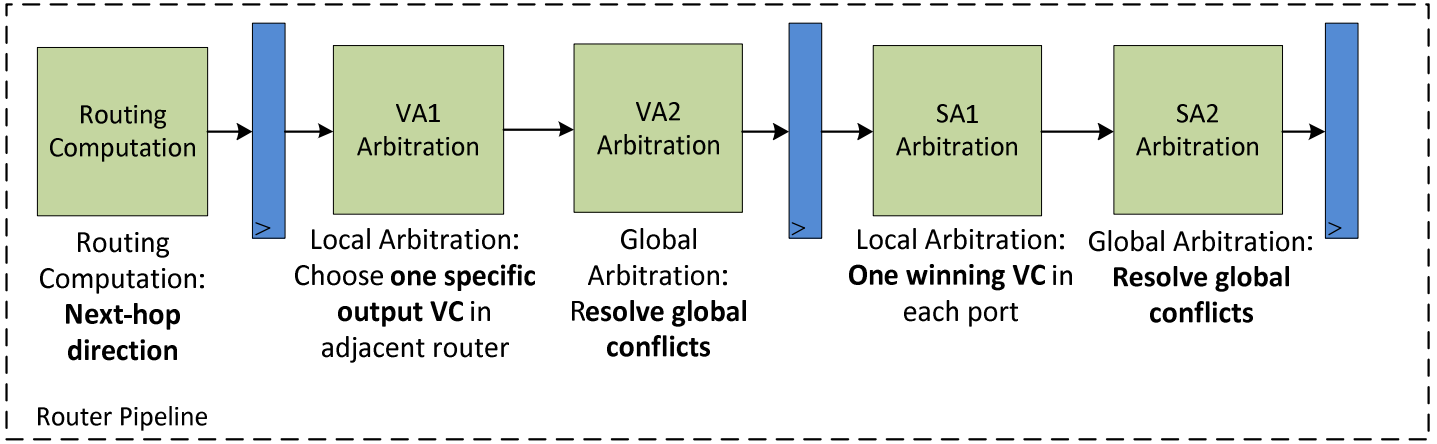
Local Arbitration: Choose **one specific output VC** in adjacent router

Global Arbitration: **Resolve global conflicts**

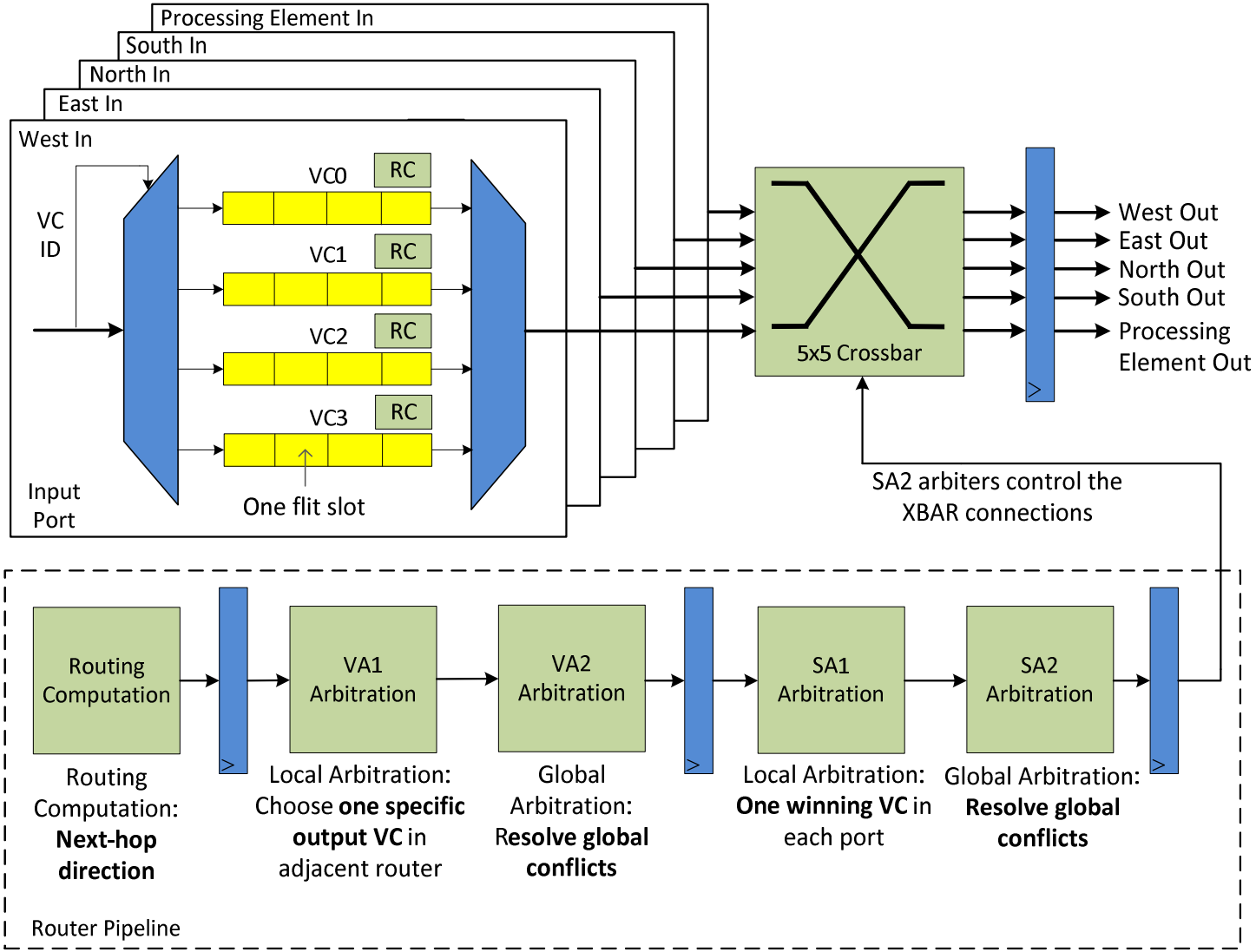
Router Pipeline

Typical NoC Router Micro-Architecture





Typical NoC Router Micro-Architecture



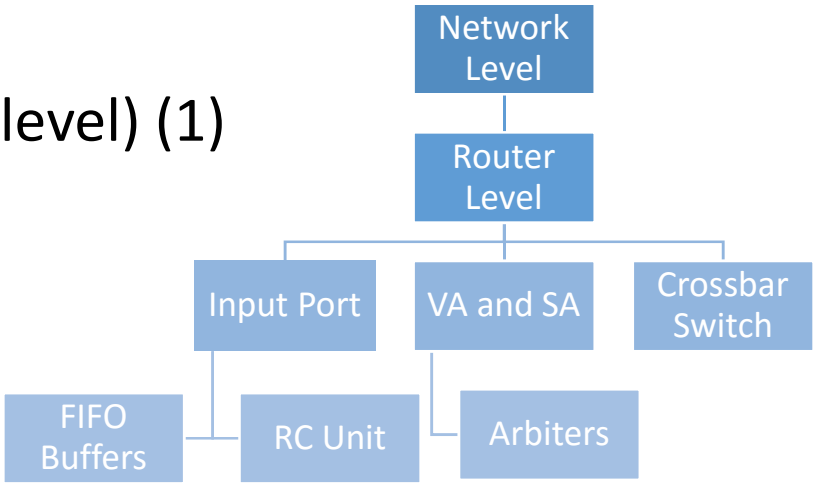
Identifying Invariances

- Modularity and hierarchy of the NoC Router
- Bottom-up approach
  - Identification of all the functional rules
  - Identification of all the functionally illegal outputs

- End-to-end invariances at the network-level

Invariance Categorization

- 32 invariances categorized based on the router module they are associated with
  - Routing Computation unit (3)
  - Arbiters (10)
  - Crossbar (3)
  - Buffer State (12)
  - Port-Level (3)
  - End-to-End (network-level) (1)

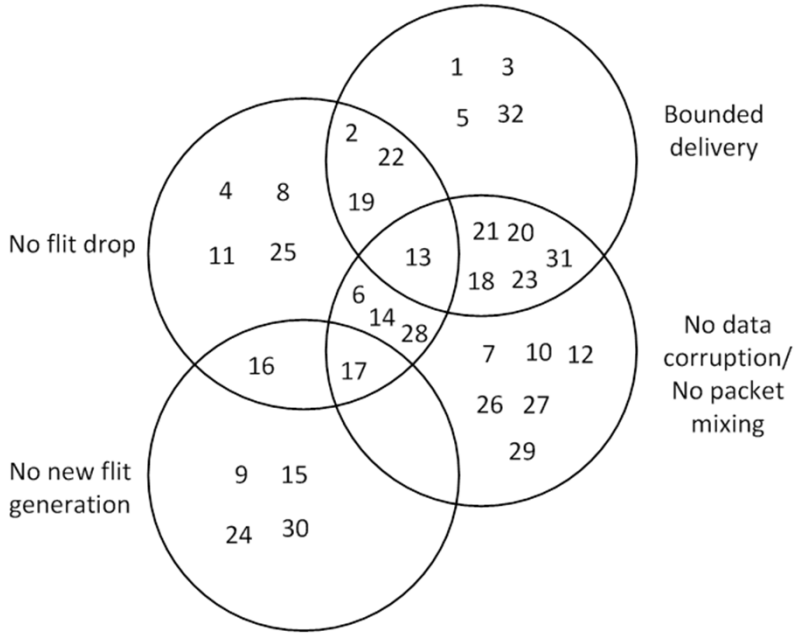


Ensuring Network Correctness

- Four main conditions that *ensure* functional correctness within the network
  - No packets are dropped
  - Delivery time is bounded
  - No data corruption occurs
  - No new packet is generated within the network

- Additional requirement:
  Intra-packet flit ordering

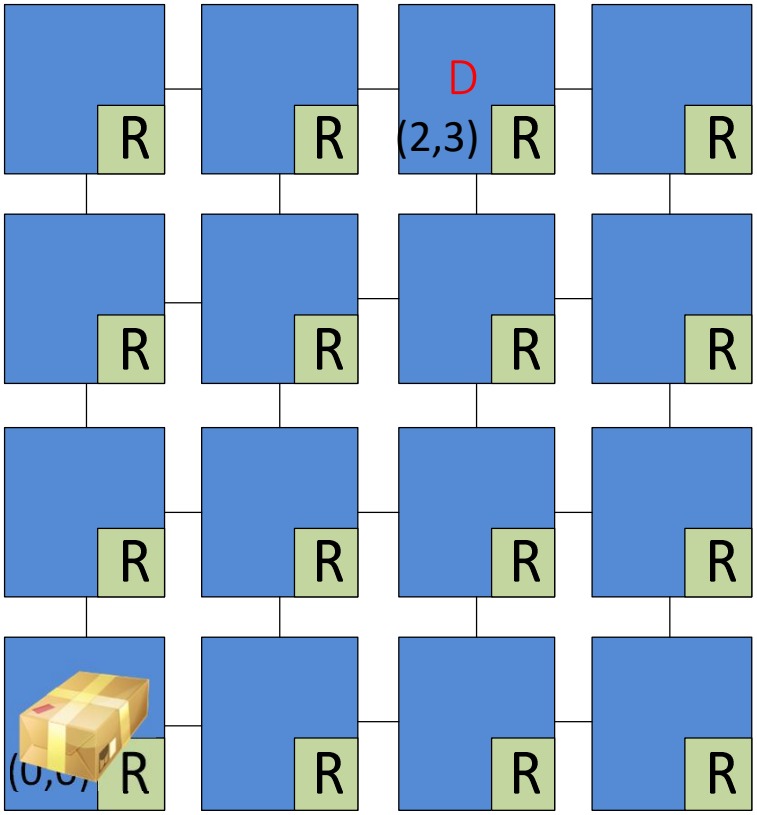
*How the 32 NoCAlert invariance checkers satisfy the **four fundamental network correctness rules***



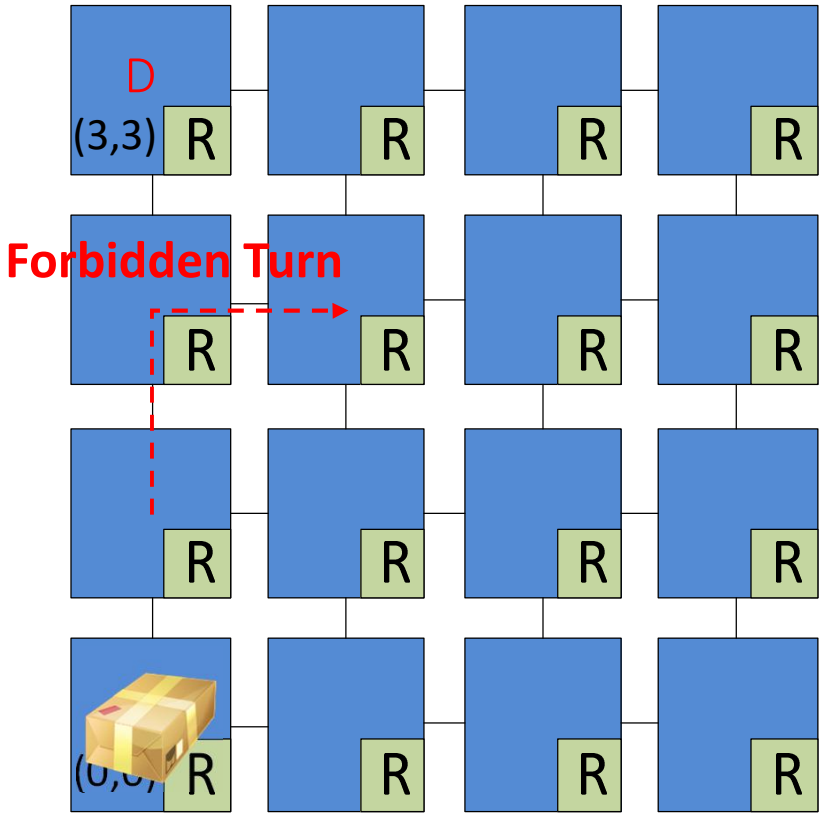
Invariance Examples

Routing Algorithm

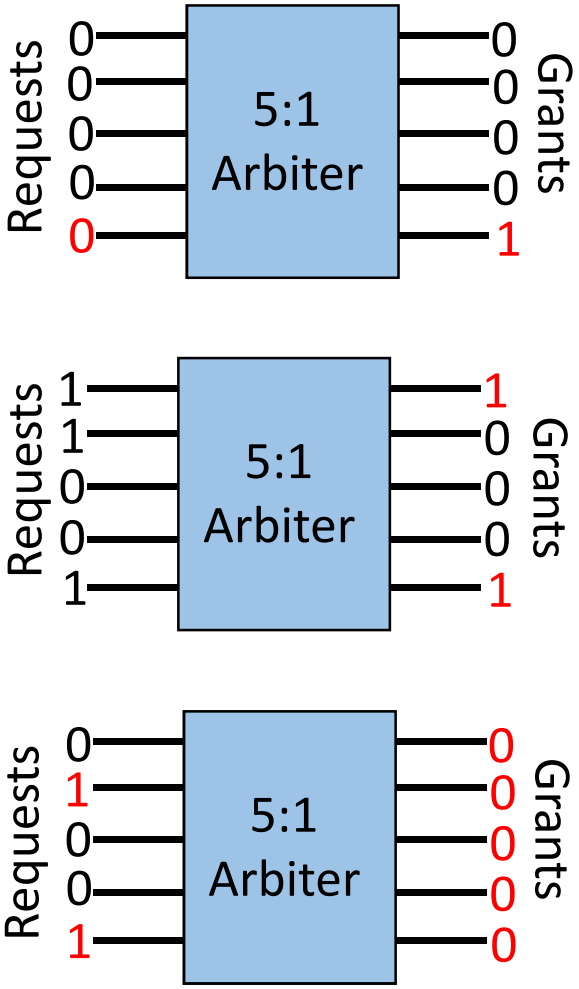
- Routing algorithms forbid some turns to avoid deadlocks and livelocks in the network
- E.g., Dimension-order XY routing



Invariance Example – Routing Algorithm



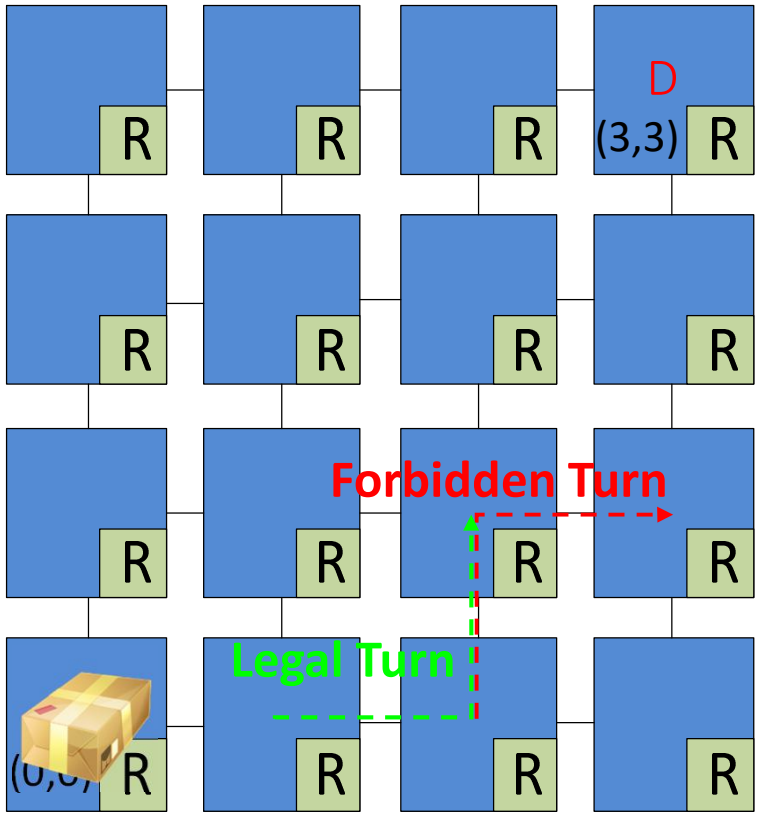
Invariance Example - Arbiters



- Grant without corresponding request
- Arbiter's output must *always* be 1-hot
- The arbiter *must* grant one of the contestants

Routing Algorithm

- Invariance checking only detects illegal outputs
- Does not necessarily detect incorrect outputs



Faults that do Not Cause Invariance Violations

- Two elemental questions arising by this kind of faults:

1. **Will the fault be caught by subsequent NoCAlert checkers?**

2. **Do they end up affecting the overall network correctness?**

Outline

- Necessity of Networks-on-Chip (NoCs)
- Reliability and NoCs
- The NoCAlert Approach: Invariance Checking
- Identifying Invariances and Examples
- Evaluation
- Results
- Conclusion

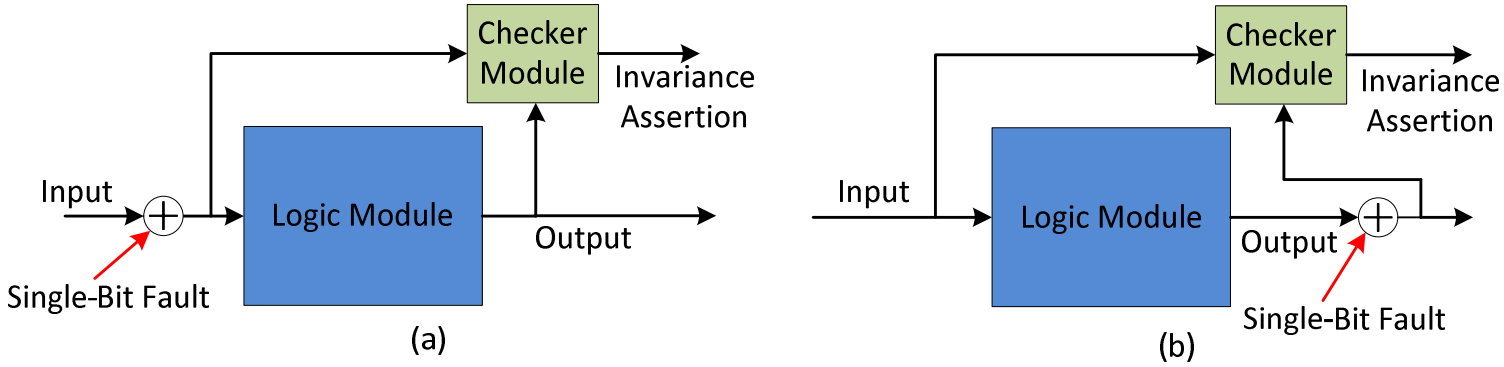
Evaluation Framework

- Tools used:
  - GARNET
    - Cycle-accurate NoC simulator
    - Extensive experimentation under fault presence
  - Synopsys Design Compiler
    - Verilog HDL
    - 65 nm commercial standard-cell libraries
    - Hardware overhead

- Compared against ForEVeR, the current state-of-the-art

Fault-Injection Framework

- Fault model: Single-bit, single-event transient faults
- At the *inputs* and *outputs* of every control module of a router
  - One fault injected in each experiment
- Total number of fault locations:
  - 11,808 for 8x8 2D mesh network



Golden Reference

- A log of the entire network's output under a *fault-free* run
- "Contaminated" Logs are compared against the GR
  - All flits were delivered *correctly* (Four rules)
  - Intra-packet order was maintained
  - Global order of packets *is allowed* to change

Network's State Affects Fault Detection

- Faults in an empty network are less likely to be masked
  - Warmed-up networks might "hide" faults
- Need for testing at different *states*
  - 7 different traffic injection ratios (10-40% in 5% increments)
  - 3 different fault injection instances
    - Cycle 0 (*empty* network)
    - Cycle 32 K
    - Cycle 64 K (*warmed-up* network)
  - 248 K simulations

Classification of NoCAlert's Detection Outcomes

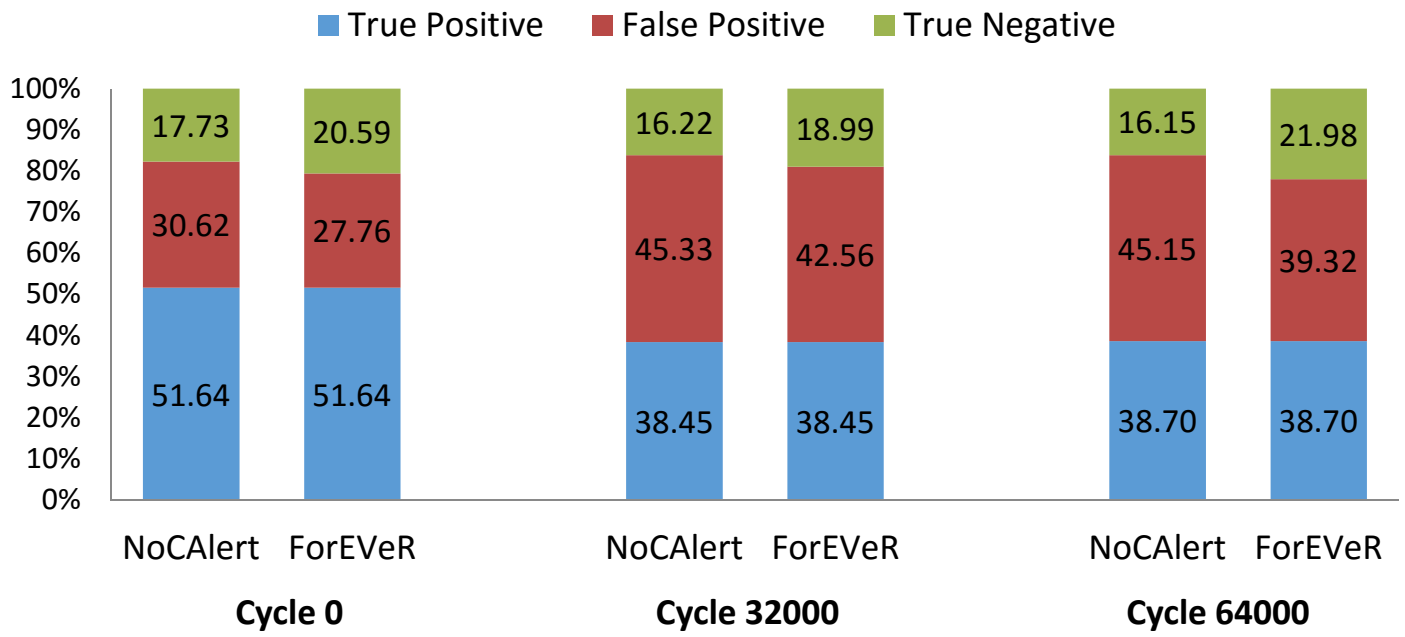
- True positive
- True negative
- False positive
  - Unnecessary fault recovery triggering
- False negative
  - Worst case
  - Ideally, this should be ZERO

	True	False
Positive	✓	✓
Negative	✓	✗

Outline

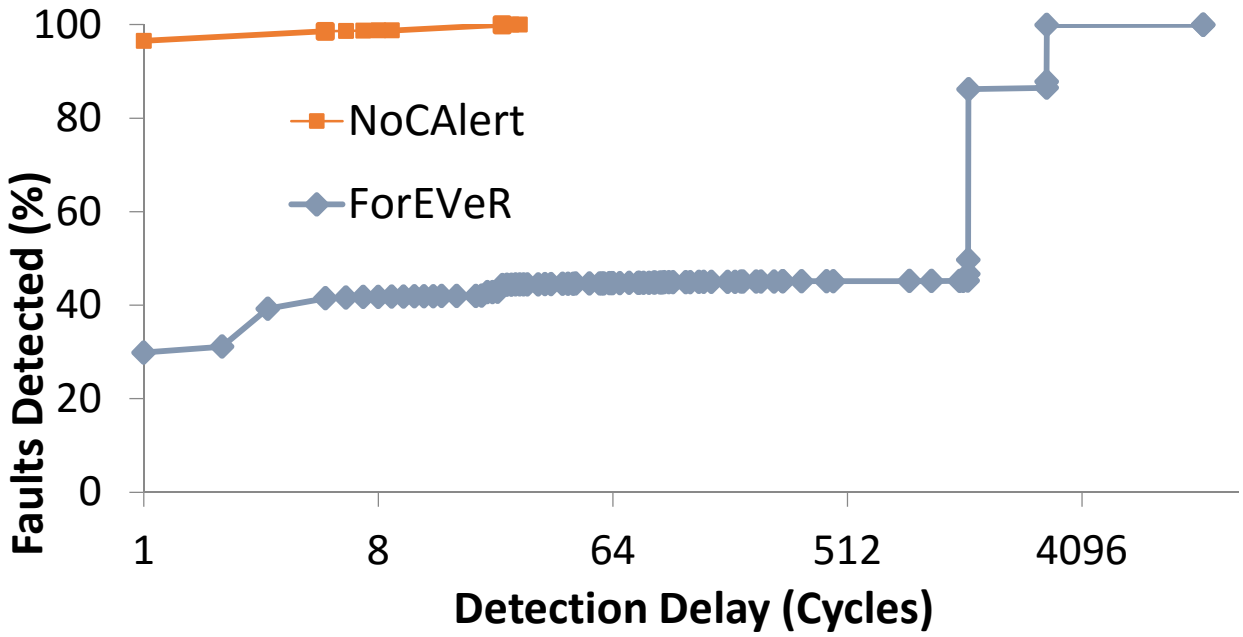
- Necessity of Networks-on-Chip (NoCs)
- Reliability and NoCs
- The NoCAlert Approach: Invariance Checking
- Identifying Invariances and Examples
- Evaluation
- Results
- Conclusion

Fault Coverage Breakdown



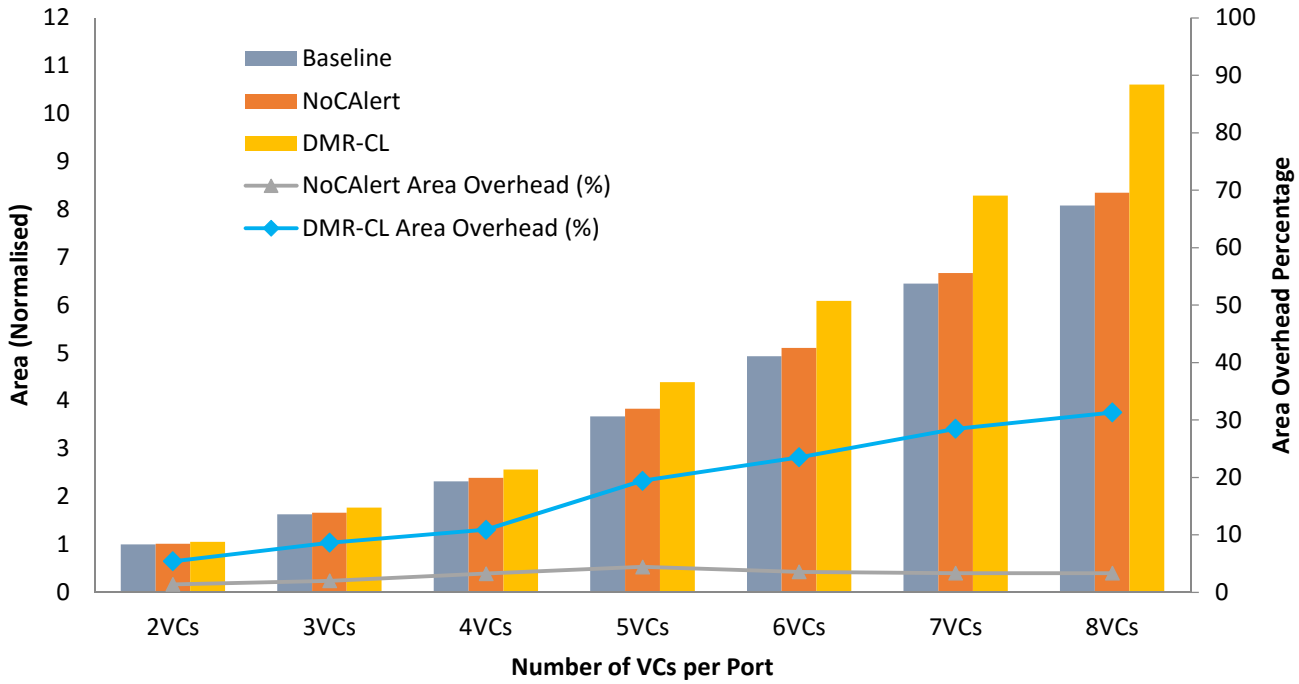
- 0% false negatives
- Higher False-positive in a warmed-up network
  - More faults are masked
- Slightly worse than ForEVeR (false positives)
  - Some faults vanish by end of epoch

Fault Detection Latency



- 97% of fault detections are *instantaneous*
- *Up to 100x* fault detection latency improvement

Hardware Overhead



- Area overhead: 3% on average
- Power overhead : 0.7% on average
- Critical path overhead: 1% on average

Outline

- Necessity of Networks-on-Chip (NoCs)
- Reliability and NoCs
- The NoCAlert Approach: Invariance Checking
- Identifying Invariances and Examples
- Evaluation
- Results
- Conclusion

Conclusion

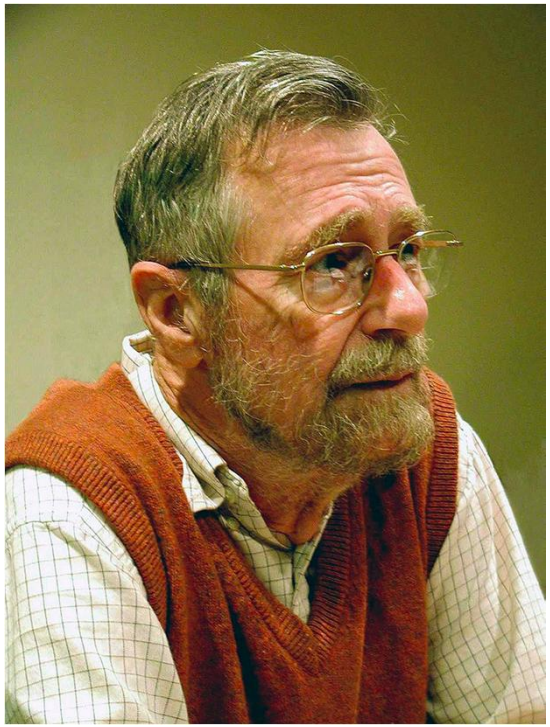
- *On-line* and *real-time* fault detection mechanism
- 0% false negatives
- *Invariance checking*
  - Distributed checkers throughout the router's control logic modules
  - Real-time hardware assertions
- Tremendous improvement in detection delay
- Extremely lightweight

Thank You!

Questions?

Discussion Points

- Does simulation expose bugs?
  - Fault model: Single-bit, single-event transient faults
  - ***At the inputs and outputs*** of every control module of a router
  - One fault injected in each experiment



Dijkstra (1969):

Testing shows the presence, not the absence of bugs.

Discussion Points

- Is NoCAlert clearly better than ForEVeR?
  - Higher false positives
  - Lower delay

ForEVeR:

- Epoch-based on-line fault detection mechanism
  - Additional 100% reliable lightweight checker network
  - Run-time checks for arbitration stages and End-to-End coverage
- Counter-based scheme that uses notification packets
- Fault assessment occurs at the end of each epoch
  - In-flight data delivered to the destination via the checker network

Discussion Points

- Is NoCAlert practical when we do not know anything about the microarchitecture of the chip?
  - Usually companies do not release too much detail