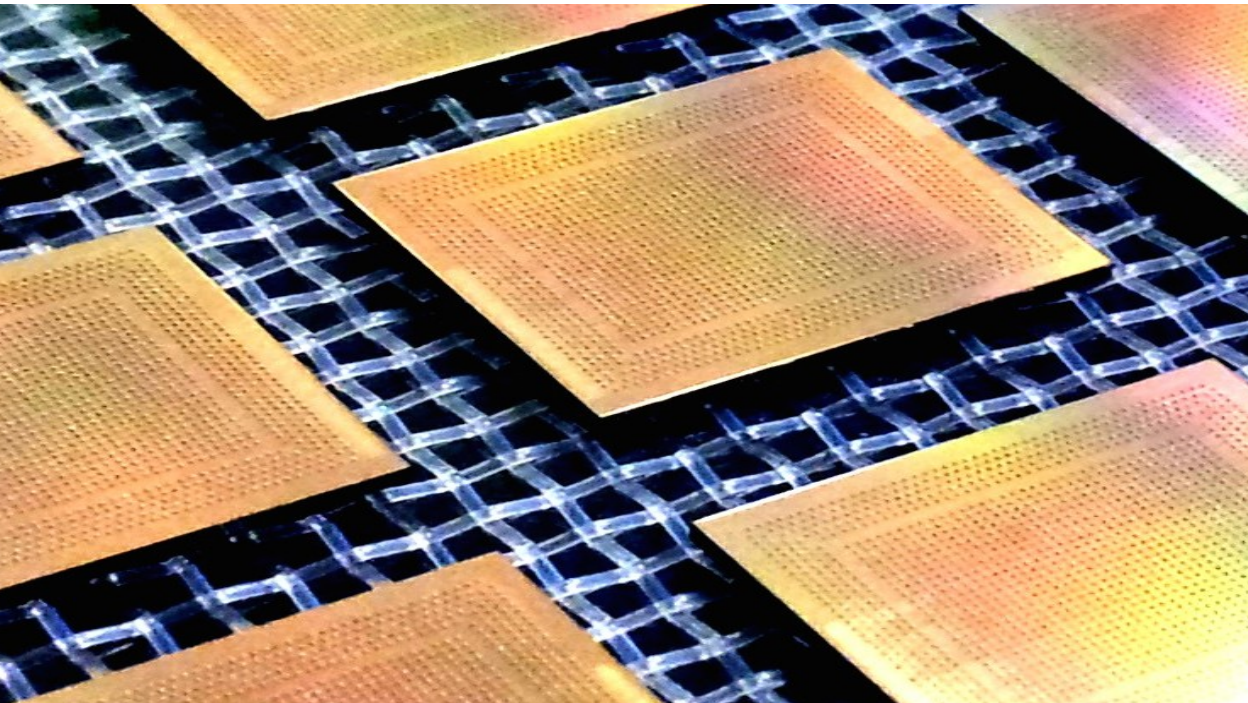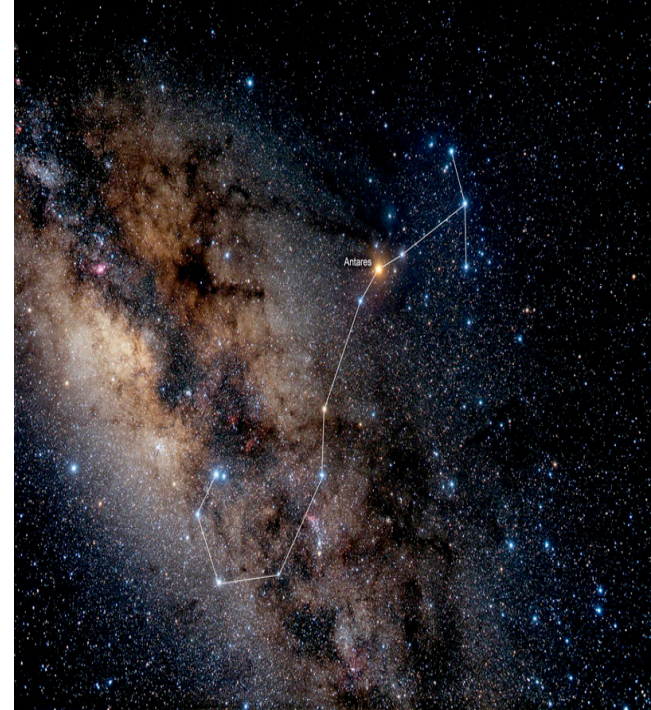# SCORPIO:

## A 36-Core Research Chip Demonstrating Snoopy Coherence on a Scalable Mesh NoC with In-Network Ordering

## Bhavya Daya

**Collaborators:**
**Chia-Hsin Chen**
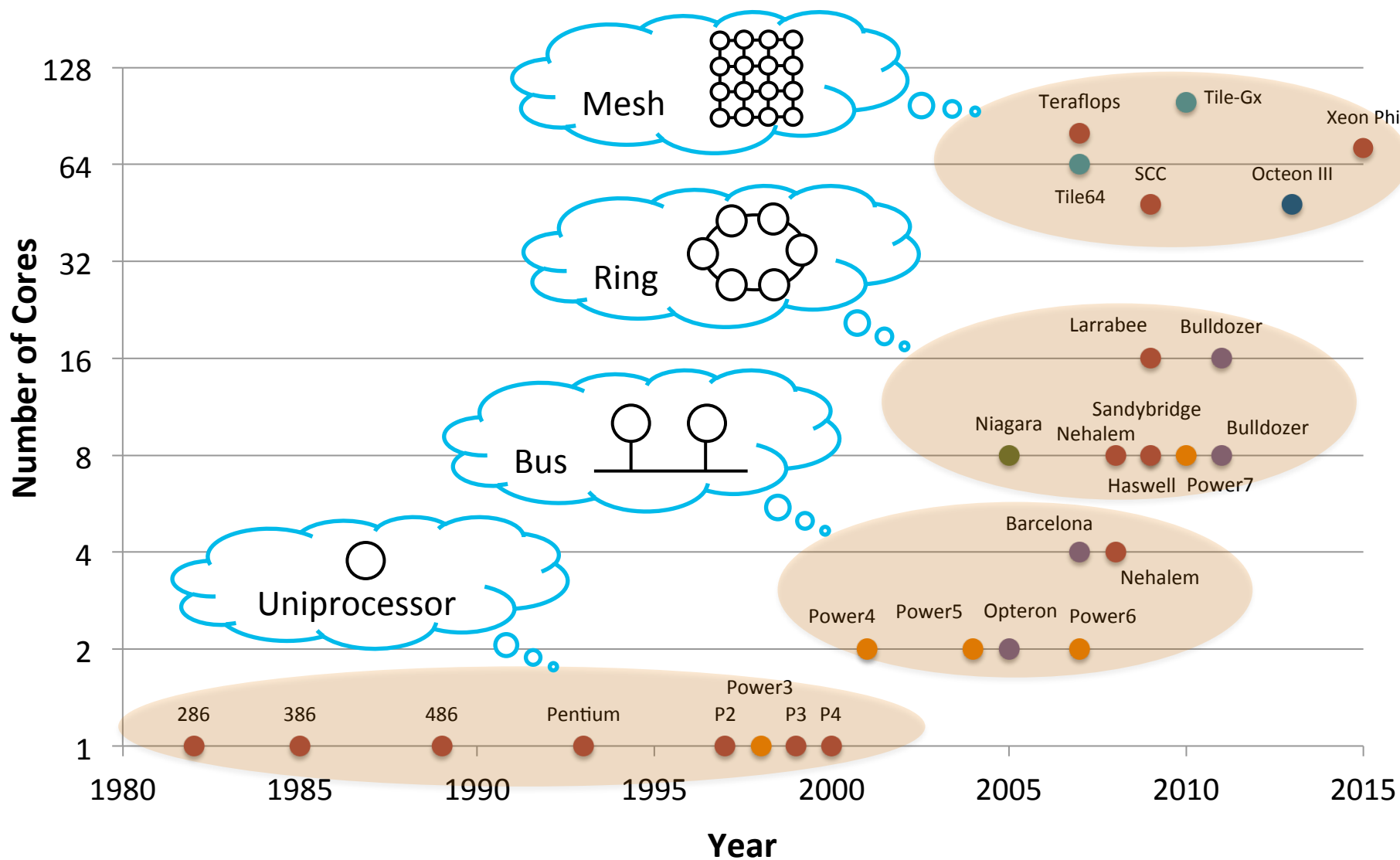**Suvinay Subramanian**
**Woo-Cheol Kwon**
**Sunghyun Park**
**Tushar Krishna**
**Jim Holt**
**Anantha Chandrakasan**
**Li-Shiuan Peh**

# Evolution of On-Chip Networks

# Objectives

**Simple**
- Homogenous/Tiled Architecture
- IP Compatibility
- Snoopy Coherence

**Scalable**
- Low Latency, High Throughput On-Chip Interconnect
- Minimal Storage Overhead and Indirection Latency for Cache Coherence

**Intuitive**
- Sequential Consistency
- Shared-Memory Paradigm
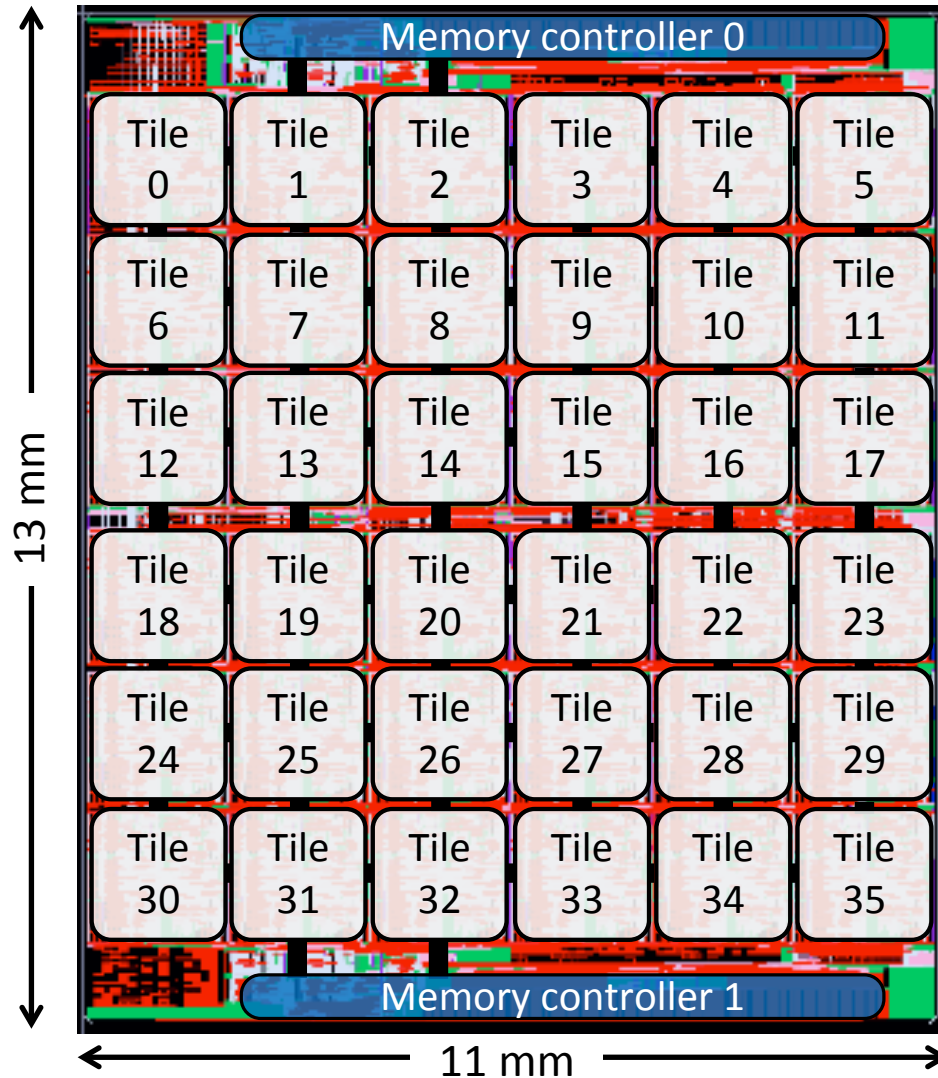- Programmability
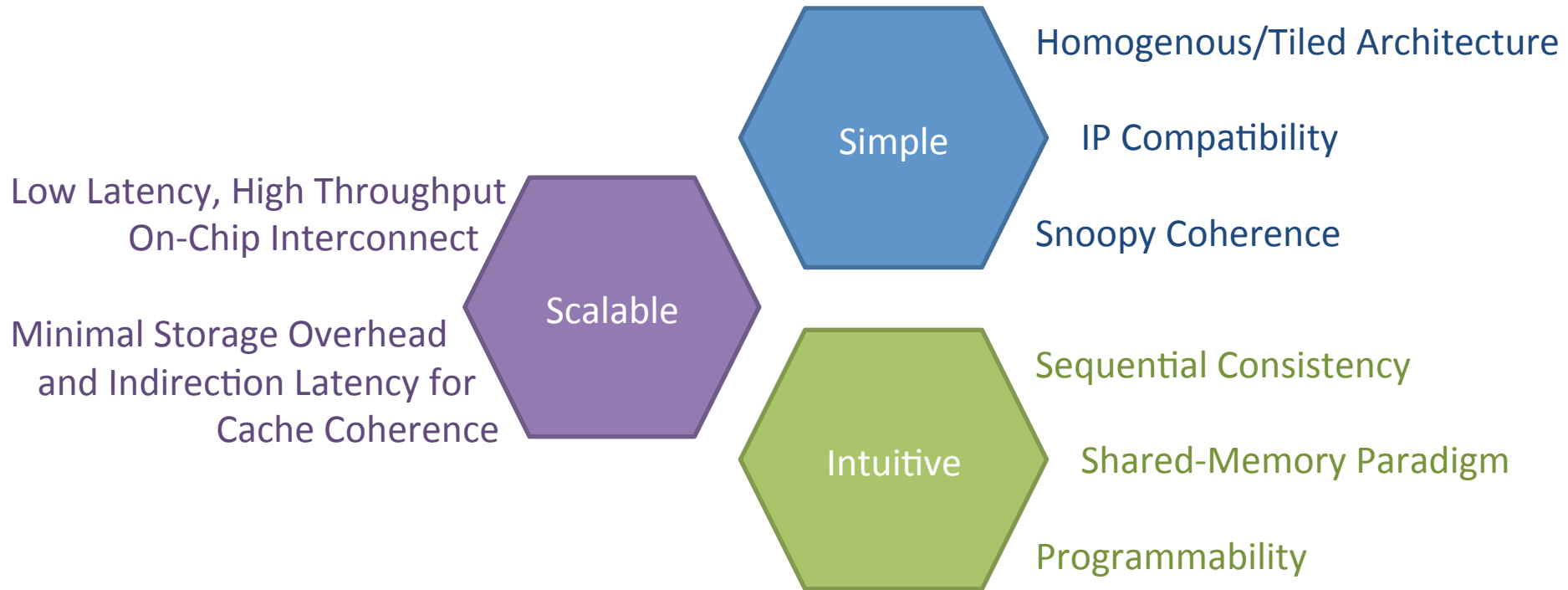
**Develop a simple, scalable, and intuitive multicore processor**

**Focus on enabling snoopy coherence on a mesh interconnect**

# Processor Overview



**IBM 45nm SOI, 143mm$^2$ 600M transistors**

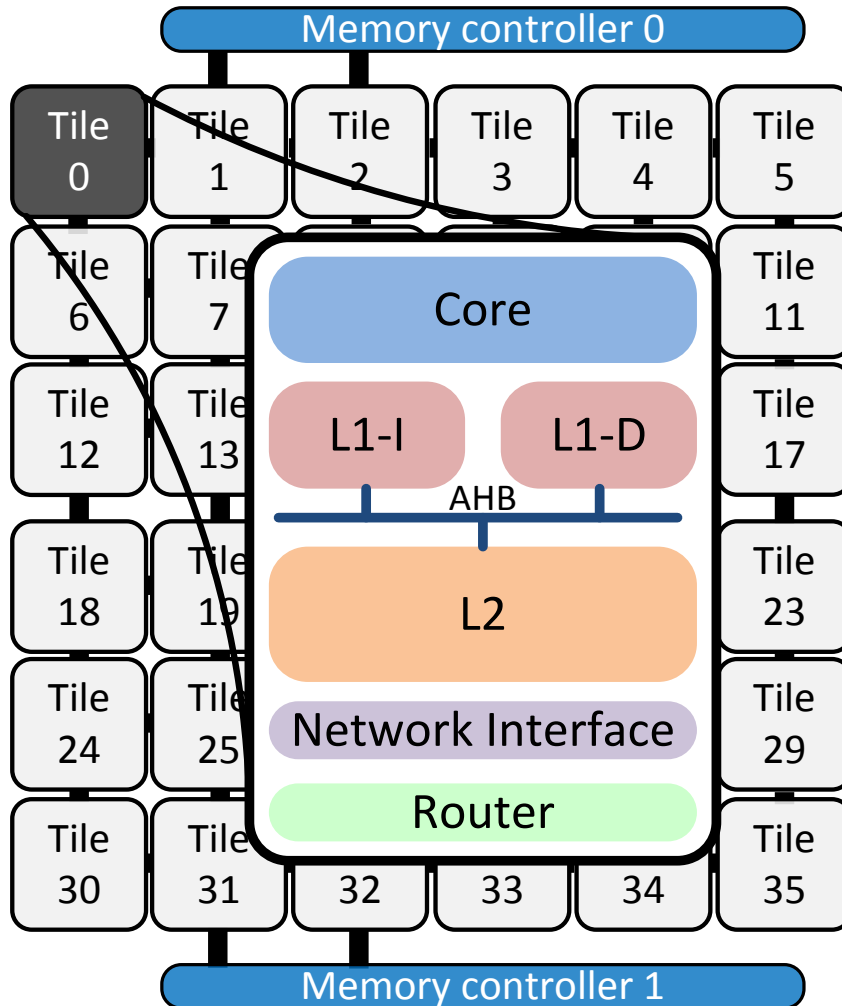**36 cores with total 4.5MB L2**

**6×6 mesh on-chip network supporting snoopy coherence**

**Dual channel DDR2 memory controllers**

# Tile Architecture



**Core**
- **Freescale e200 z760n3**
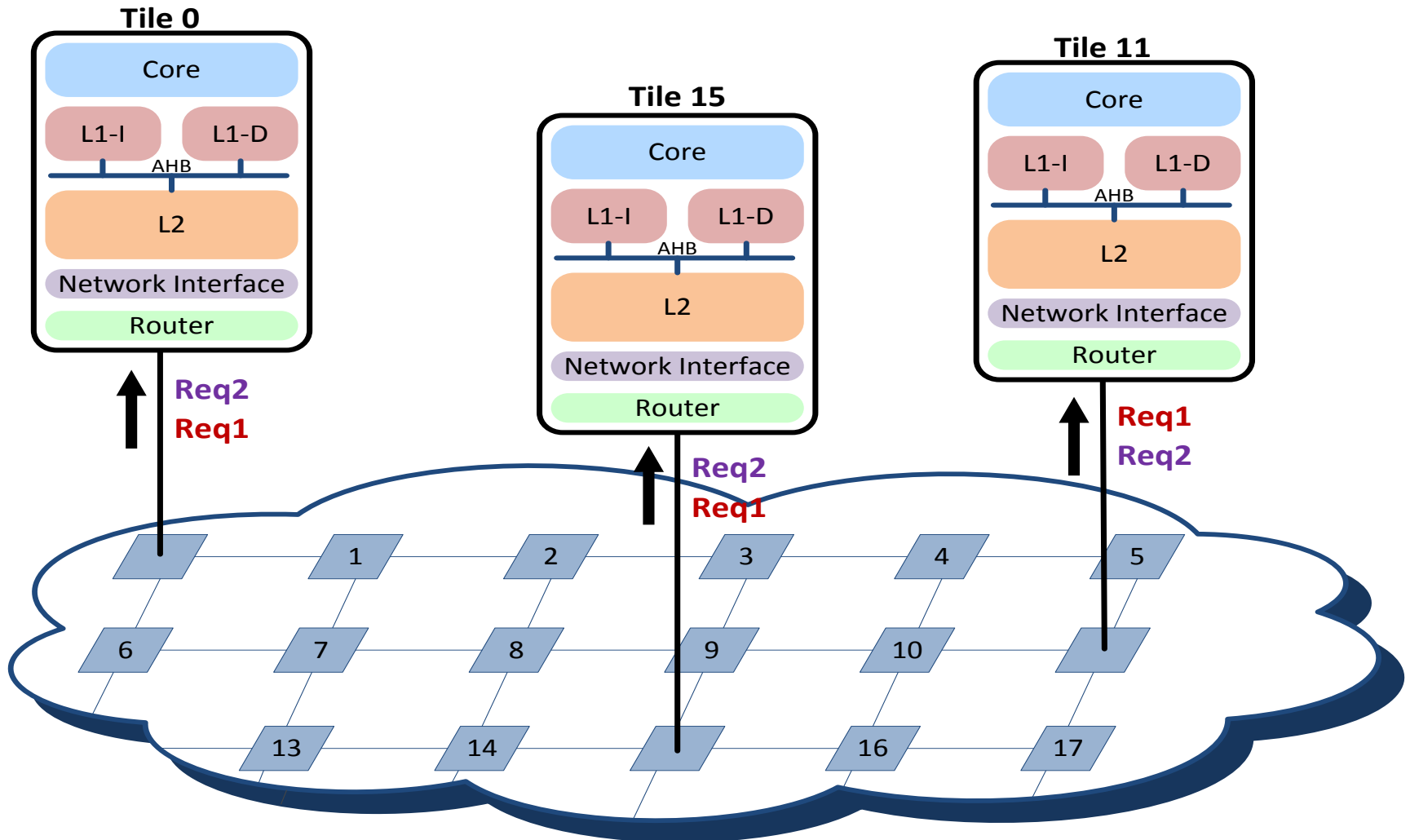- **In-order**
- **Dual-issue**

**Private L1 cache**
- **Split 16KB for Inst / Data**
- **4-way set associative**

Write-through ⬇ ⬆ Back-invalidate
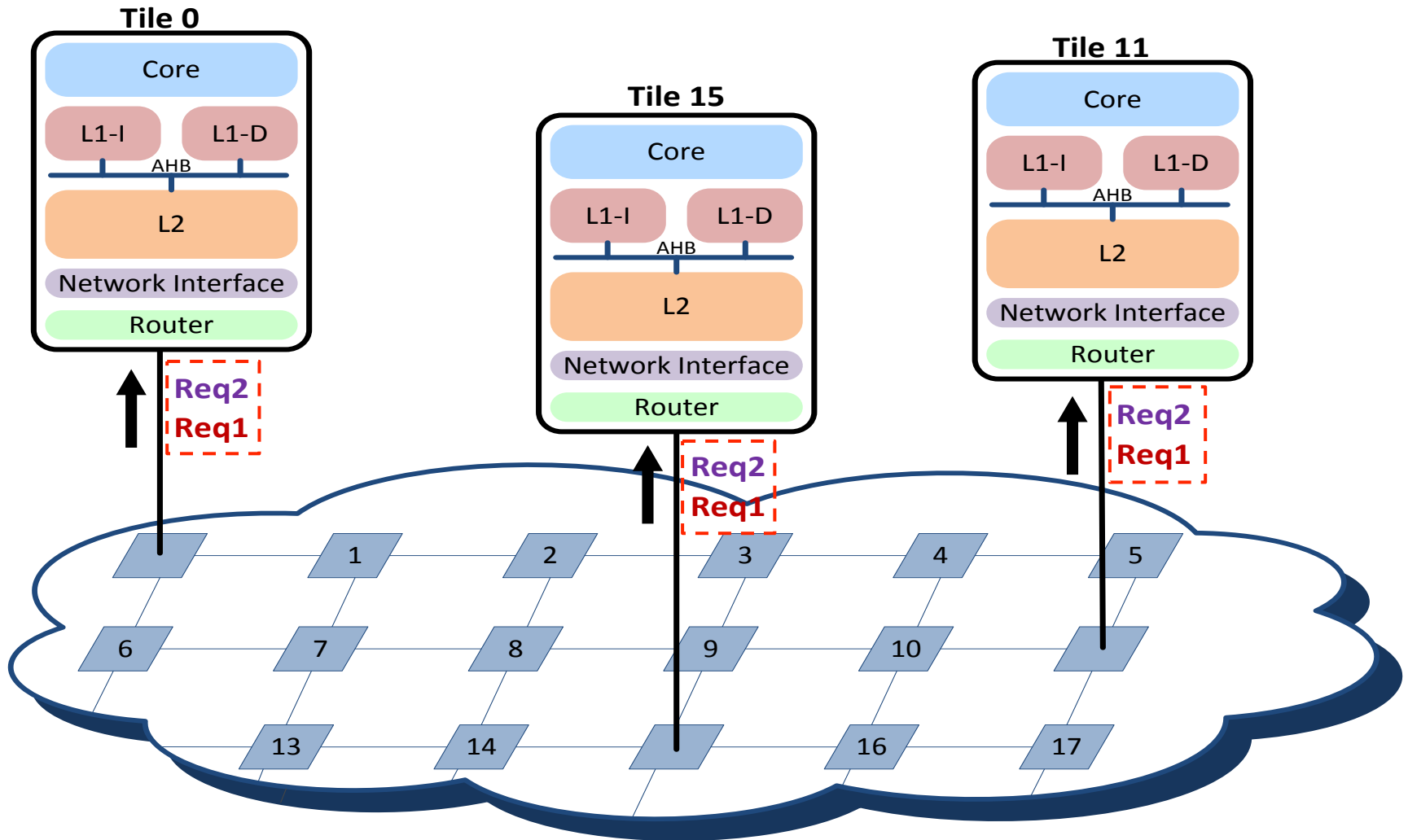
**Private L2 cache**
- **128KB**
- **4-way set associative**
- **Inclusive**
- **MOSI Protocol**

# Globally Ordered Mesh Network

# Globally Ordered Mesh Network

# Decouple Ordering from Message Delivery



**Main network**
- ◇ **Message delivery**

**Notification network**
- ◇ **Message ordering**
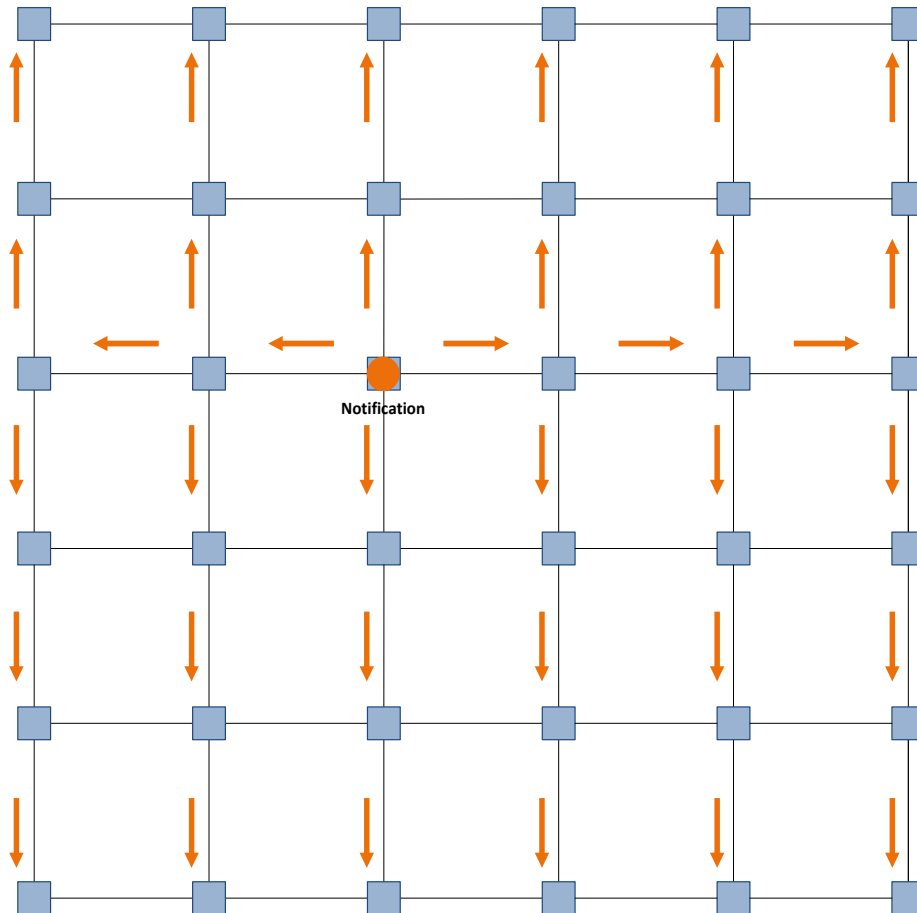
# Notifications



Notification

## Bounded latency ( ≤ 12 cycle )

- **Non-blocking**
- **1 cycle / hop broadcast mesh**

## Notification bit-vector

- **Dedicated 1 bit / tile**
- **Inject notification for each coherent request**

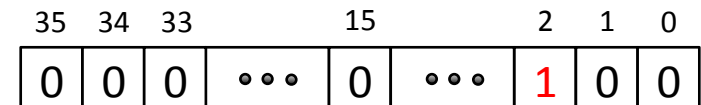| 35 | 34 | 33 | | 15 | | 2 | 1 | 0 |
|----|----|----|-----|----|-----|---|---|---|
| 0 | 0 | 0 | ••• | 0 | ••• | 1 | 0 | 0 |

# Notifications



**Bounded latency ( ≤ 12 cycle )**
- **Non-blocking**
- **1 cycle / hop broadcast mesh**

**Notification bit-vector**
- **Dedicated 1 bit / tile**
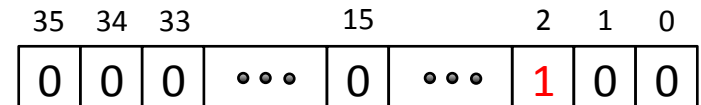- **Inject notification for each coherent request**

| 35 | 34 | 33 | | 15 | | 2 | 1 | 0 |
|----|----|----|-----|----|-----|-----|----|----|
| 0 | 0 | 0 | ••• | 0 | ••• | 1 | 0 | 0 |
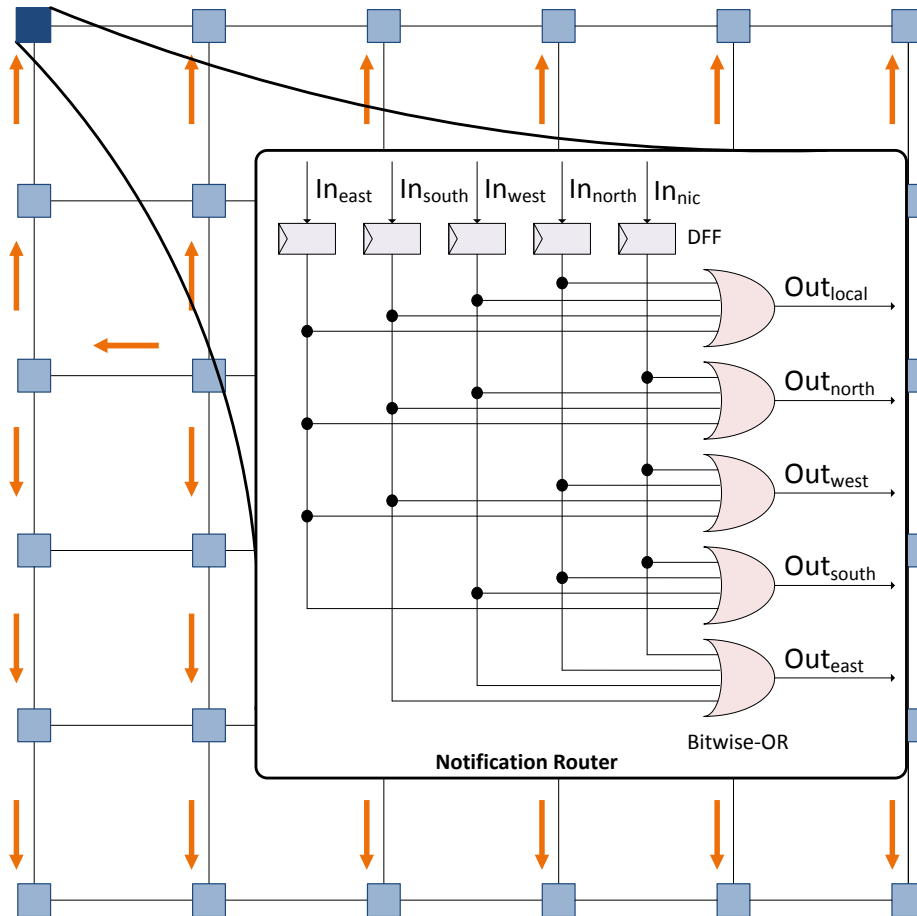
**Low cost**
- **Only DFF + ORs**

# Main Network Router
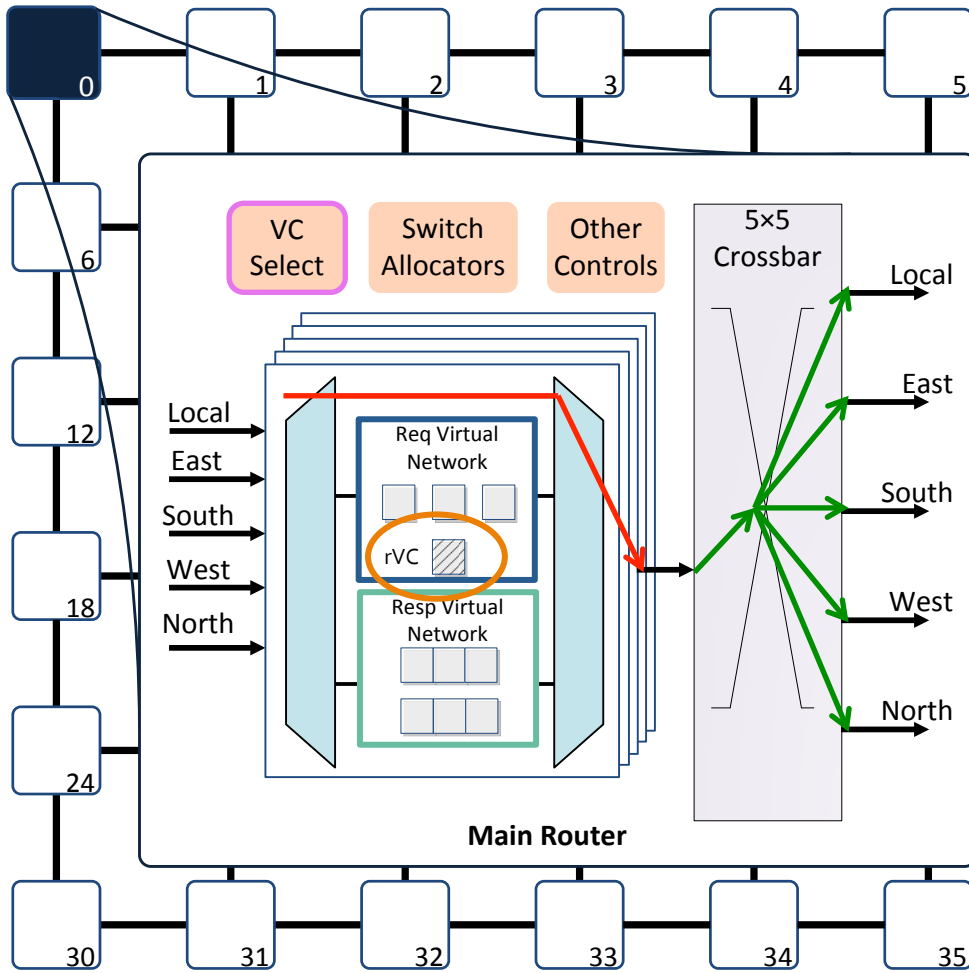


## Two Virtual Networks

- Globally Ordered Request (GO-REQ)
- Unordered Response (UO-RESP)

## Optimizations

- Virtual Bypassing
- Broadcast Support
- VC Select Mechanism

## Deadlock Avoidance

- XY Routing
- Separate virtual networks for requests and responses
- Reserved VC (rVC) for highest priority flit

# Time Window

Broadcast messages on main network

Inject corresponding notifications

All tiles receive the same notifications

Timeline
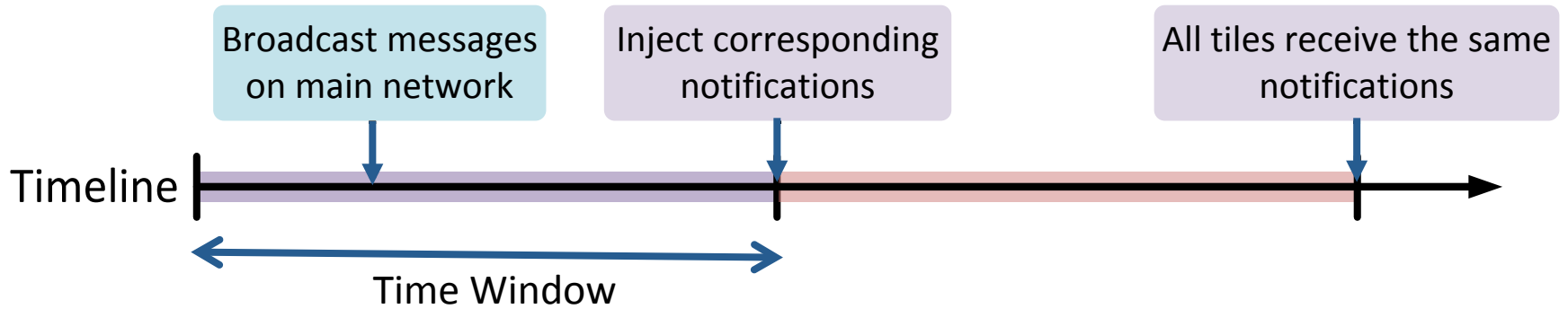
Time Window

| Length | At least equal to the fixed notification network latency bound |

| Fairness | Rotating global order priorities in each time window |

Determine *locally* the *global order* according to consistent ordering rule at all nodes

# Journey of a Coherent Request



T3. Both cores inject notification N1 N2

T1. Core 6 injects M1

Timeline

T2. Core 12 injects M2
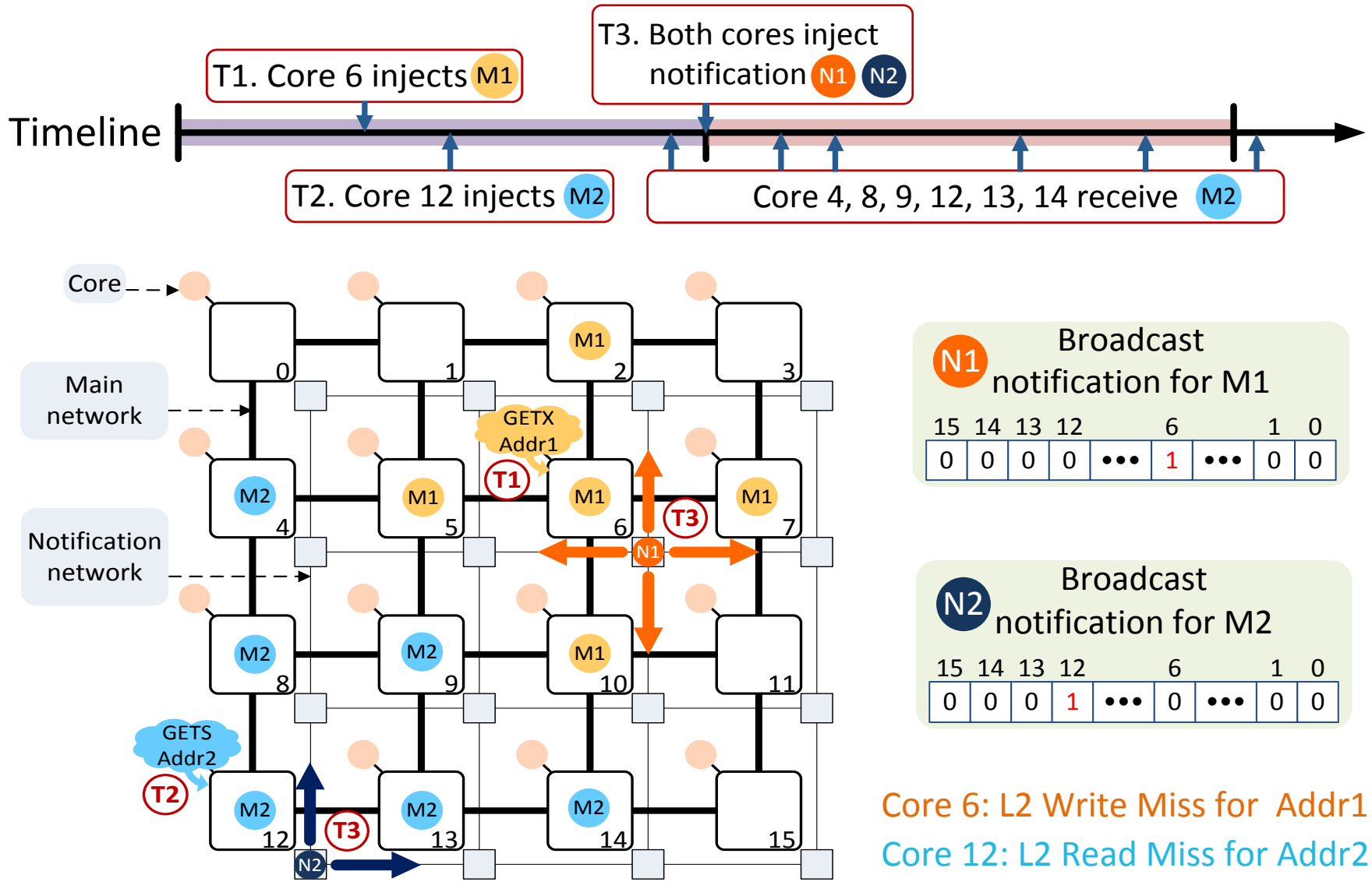
Core 4, 8, 9, 12, 13, 14 receive M2

Core

Main network

Notification network

GETX Addr1

T1

T3

GETS Addr2

T2

T3

N2

N1 Broadcast notification for M1

| 15 | 14 | 13 | 12 | | 6 | | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | ••• | 1 | ••• | 0 | 0 |

N2 Broadcast notification for M2

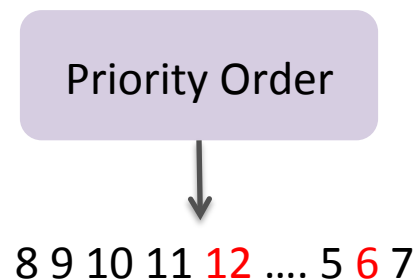| 15 | 14 | 13 | 12 | | 6 | | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | ••• | 0 | ••• | 0 | 0 |

Core 6: L2 Write Miss for Addr1
Core 12: L2 Read Miss for Addr2

# Journey of a Coherent Request



T3. Both cores inject notification N1 N2

T4. Notifications N1 N2 guaranteed to reach all nodes now

Timeline

Core 4, 8, 9, 12, 13, 14 receive M2

T5. Core 4, 8, 9, 12, 13, 14 processed M2

Core

Main network

Notification network

Merged Notification

N1 N2

| 15 | 14 | 13 | 12 | | 6 | | 1 | 0 |
|----|----|----|----|-----|---|-----|---|---|
| 0 | 0 | 0 | 1 | ••• | 1 | ••• | 0 | 0 |

Priority Order

8 9 10 11 12 …. 5 6 7

# Journey of a Coherent Request



Cores receive M1 M2 in any order, and process M2 followed by M1

T7. Core 9, owner of Addr1, responds R1 with data to Core 6

Timeline

T5. Core 4, 8, 9, 12, 13, 14 processed M2

T6. Core 0, owner of Addr2, responds R2 with data to Core 12

Data responses sent on unordered response virtual network in main network

# Architecture Analysis

| Simulator | GEMS + GARNET |
|-----------|---------------|
| Access times | L1 – 1 cycle; L2 – 10 cycles; DRAM 90 cycles |
| LPD-D | Limited Pointer Distributed Directory |
| HT-D | AMD HyperTransport (Distributed) |
| SCORPIO-D | Snoopy Coherence: MOSI (Distributed) |

| | LPD-D | HT-D | SCORPIO-D |
|---|-------|------|-----------|
| What is tracked? | Few sharers | Presence of owner | Presence of owner |
| Ordering point | Directory | Directory | Network |

Isolate

Storage overhead

Indirection latency

# Normalized Runtime



Legend: LPD-D, HT-D, SCORPIO-D

**36 CORES**
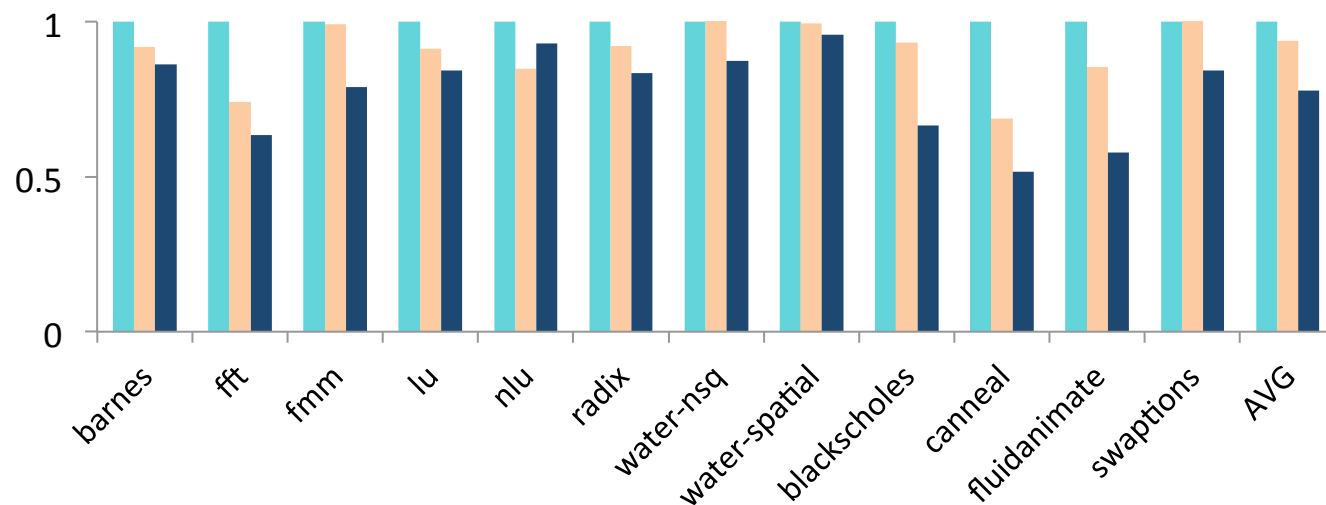
- ➤ 22% lower than LPD-D
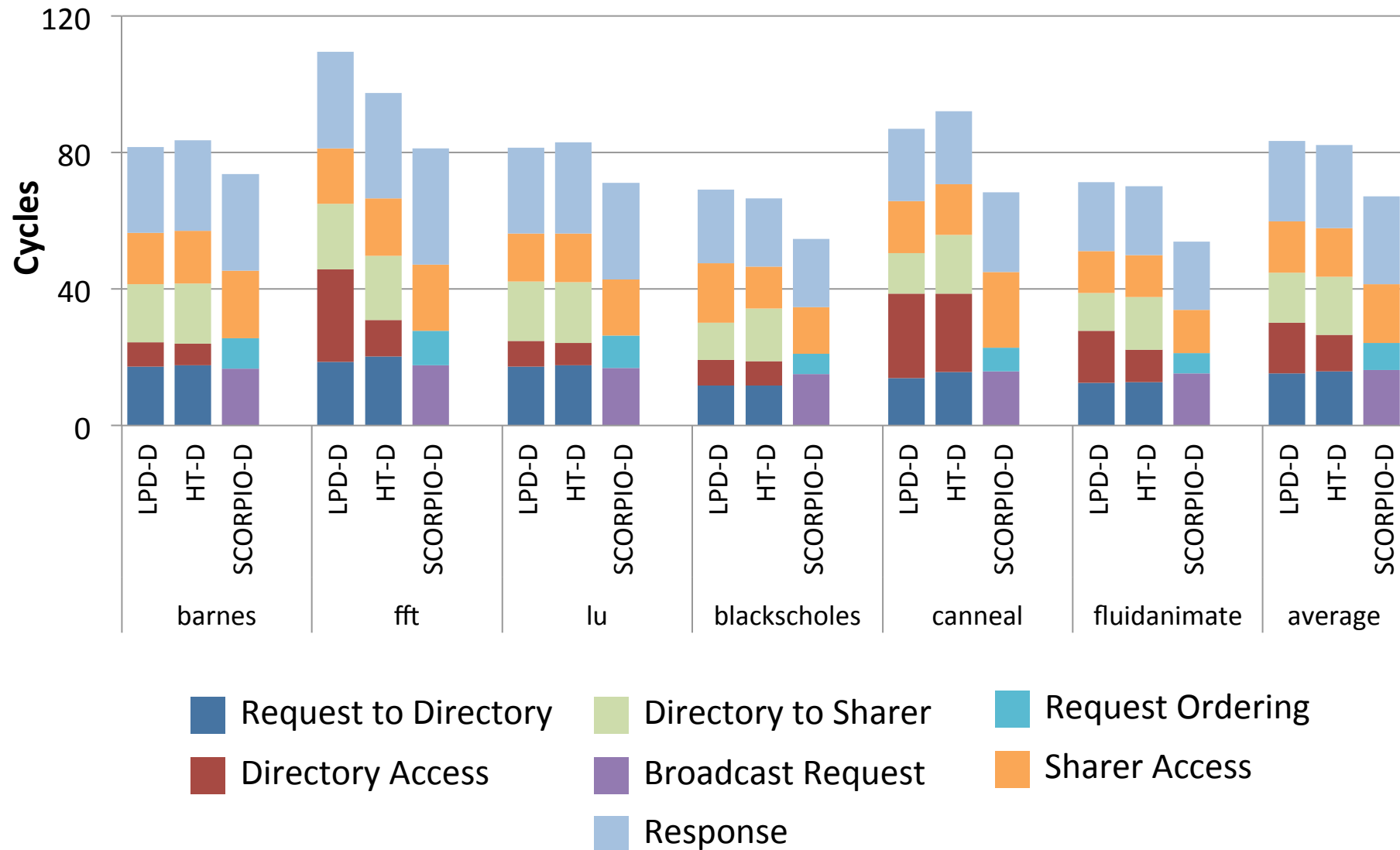- ➤ 17% lower than HT-D

**64 CORES**
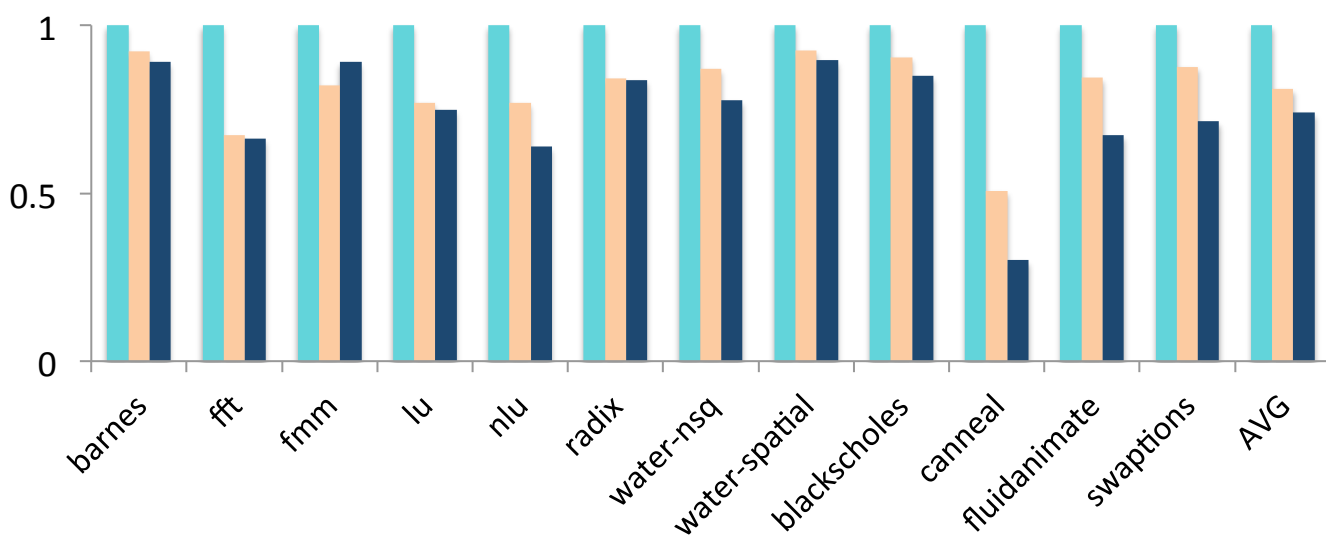
- ➤ 24% lower than LPD-D
- ➤ 13% lower than HT-D
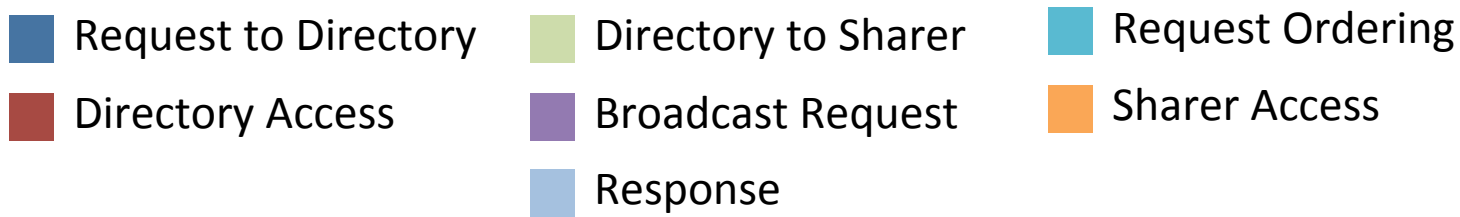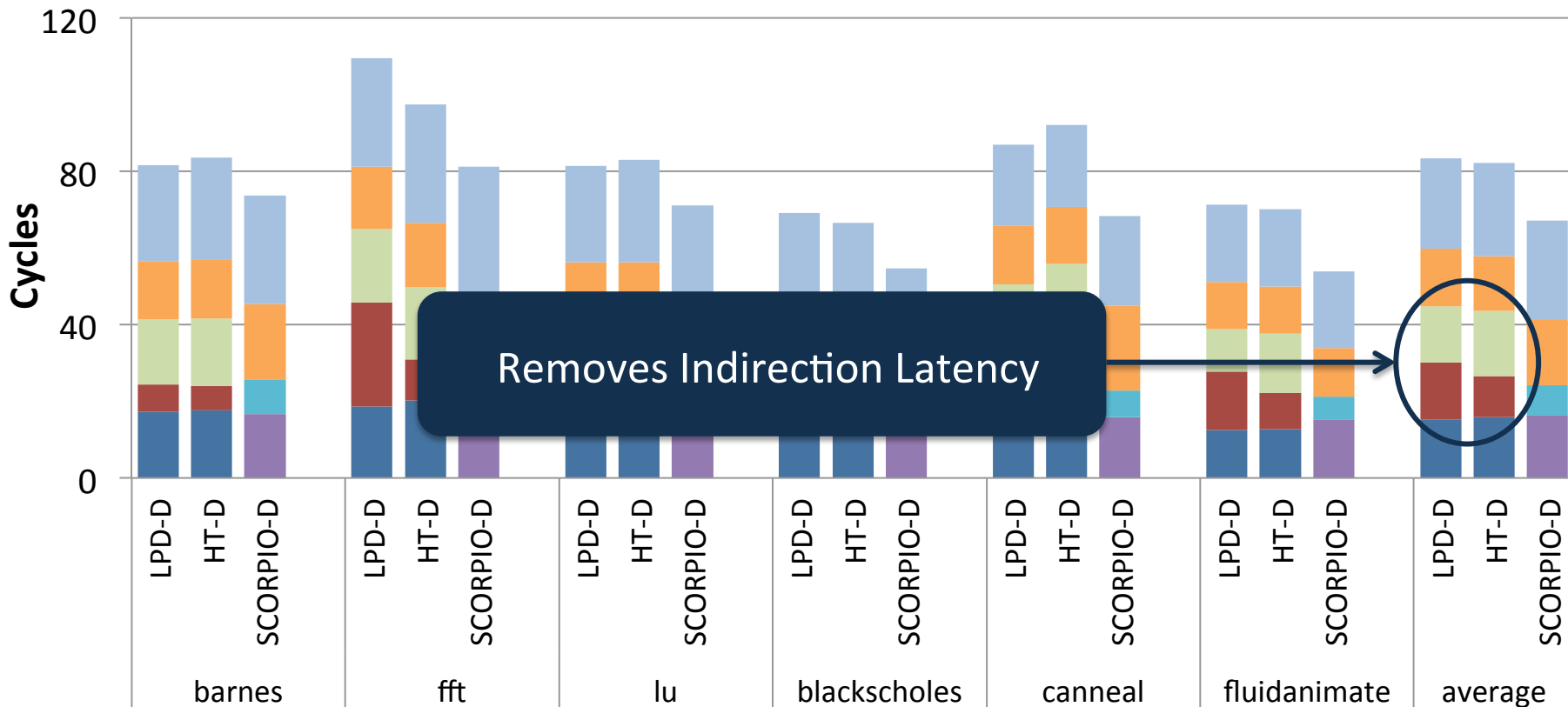
# L2 Service Latency Breakdown



Served by Other Caches On-Chip

# L2 Service Latency Breakdown



**Served by Other Caches On-Chip**

Removes Indirection Latency

Legend:
- Request to Directory
- Directory Access
- Directory to Sharer
- Broadcast Request
- Request Ordering
- Sharer Access
- Response

Categories (x-axis): barnes, fft, lu, blackscholes, canneal, fluidanimate, average — each with LPD-D, HT-D, SCORPIO-D

# L2 Service Latency Breakdown



Served by Off-Chip Memory Access

Legend:
- Request to Directory
- Response
- Directory Access
- Broadcast Request
- Request Ordering

# L2 Service Latency Breakdown



Served by Off-Chip Memory Access

Ordering latency overhead
10% of accesses are off-chip

Effect of limited storage area

Legend:
- Request to Directory
- Response
- Directory Access
- Broadcast Request
- Request Ordering

Categories: barnes, fft, lu, blackscholes, canneal, fluidanimate, average

Each with: LPD-D, HT-D, SCORPIO-D

Y-axis: Cycles (0 to 250)

# Practical Network Design

## Token Coherence (TokenB)

- T tokens for each block of shared memory
- T = # of processors

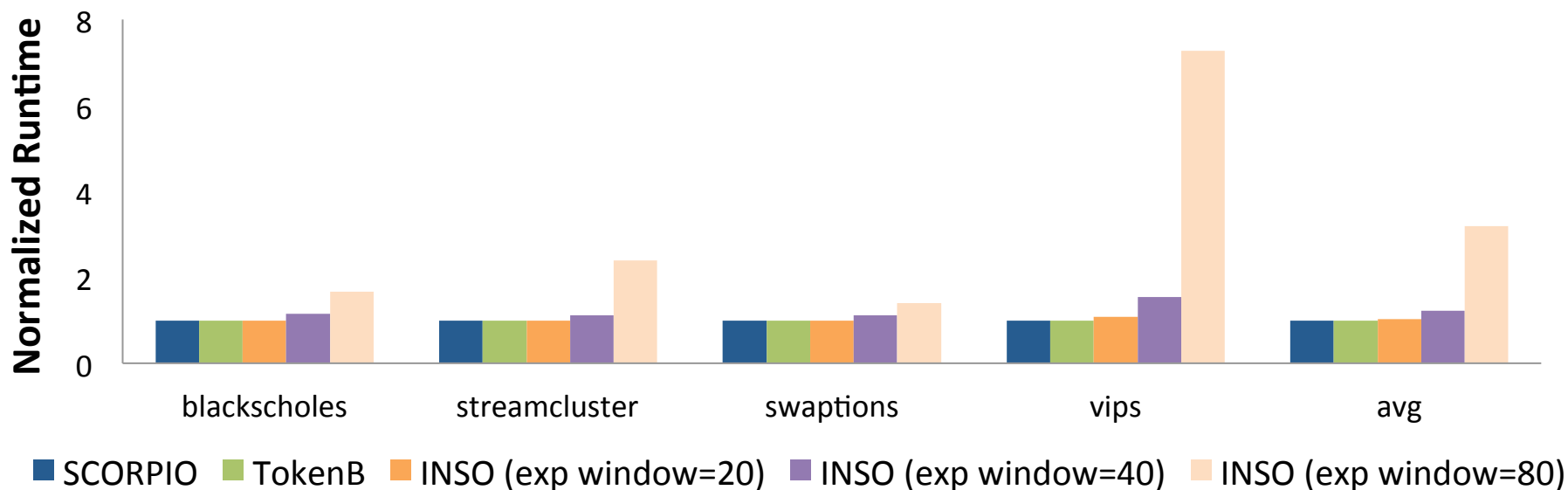*Area Overhead* $\propto$ *Number Cachelines*

## In-Network Snoop Ordering (INSO)

- Periodic expiration required
- Worsening ordering latency and network traffic with small expiration window

## Timestamp Snooping (TS)

- Each request tagged with an ordering time (OT)

*Buffer Area Per Core* $\propto$ *# of Cores*

$\propto$ *Msgs/Core*



Legend: ■ SCORPIO ■ TokenB ■ INSO (exp window=20) ■ INSO (exp window=40) □ INSO (exp window=80)

X-axis: blackscholes, streamcluster, swaptions, vips, avg

Y-axis: Normalized Runtime (0 to 8)

# Practical Network Design

## Token Coherence (TokenB)

- T tokens for each block of sha...
- T = # ...

*Area Overhead* ∝ *Number Cachelines*

## In-Network Snoop Ordering (INSO)

- Periodic expiration

traffic with small expiration window

## Timestamp Snooping (TS)

- Each request tagged ...ng time
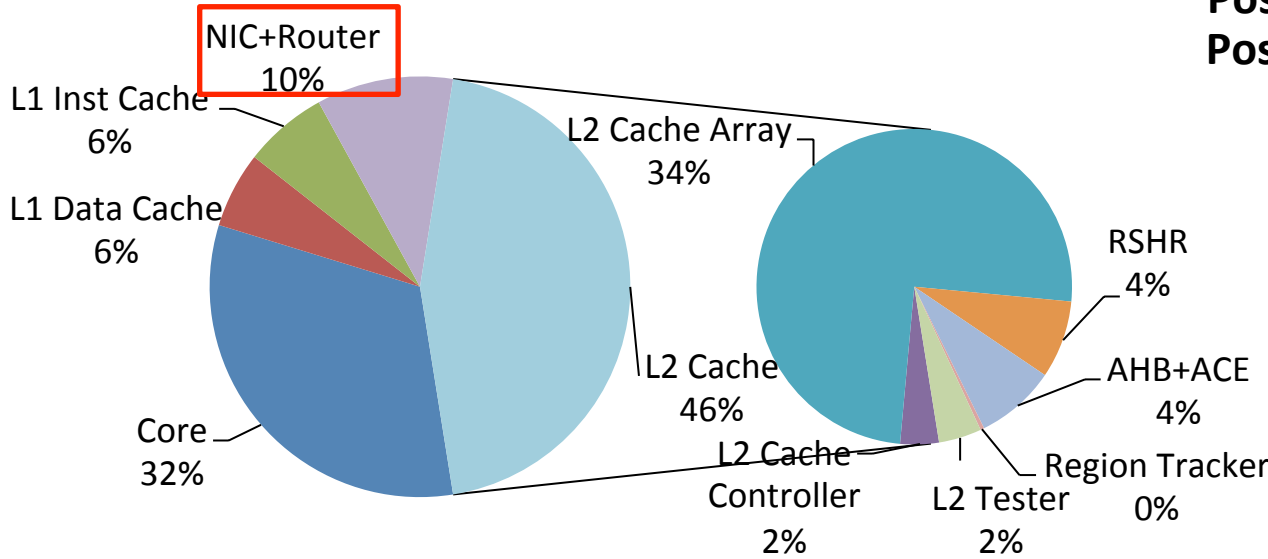
*Buffer Area Per Core* ∝ *# of Cores Msgs/Core*

**SCORPIO Minimum Area Requirement for Correctness:**

**2 buffers at each endpoint**



Performs as well as TokenB (without persistent requests) and INSO (impractical expiration window size)

Legend: ■ SCORPIO ■ TokenB ■ INSO (exp window=20) ■ INSO (exp window=40) ■ INSO (exp window=80)

X-axis: blackscholes, streamcluster, swaptions, vips, avg
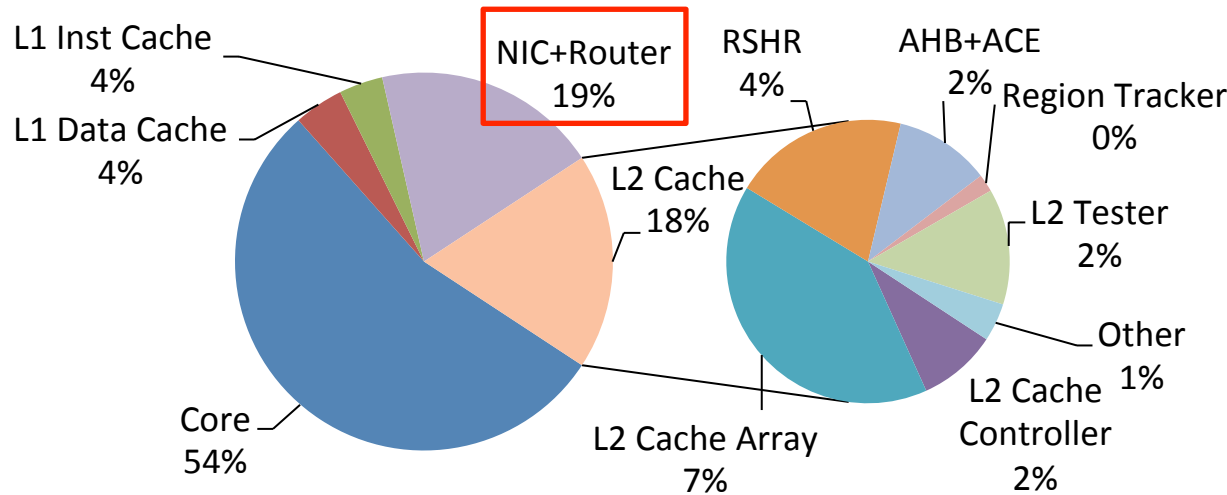Y-axis: Normalized Runtime (0 to 8)
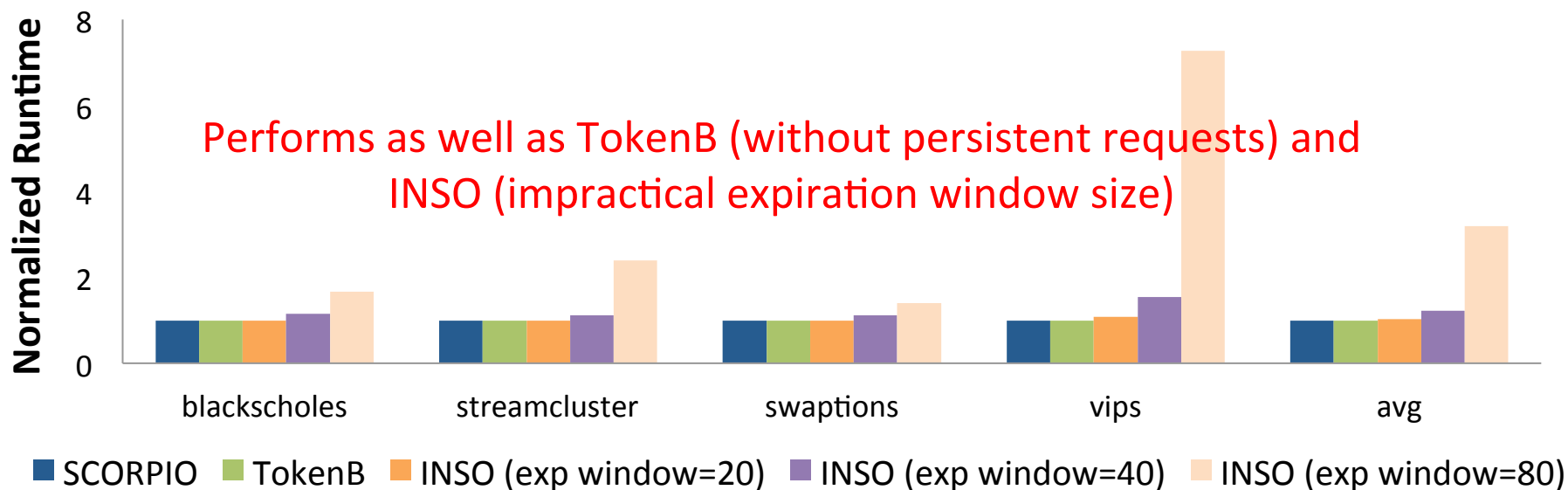
# Network Cost



**Post-synthesis frequency: 1 GHz**
**Post-layout frequency: 833 MHz**

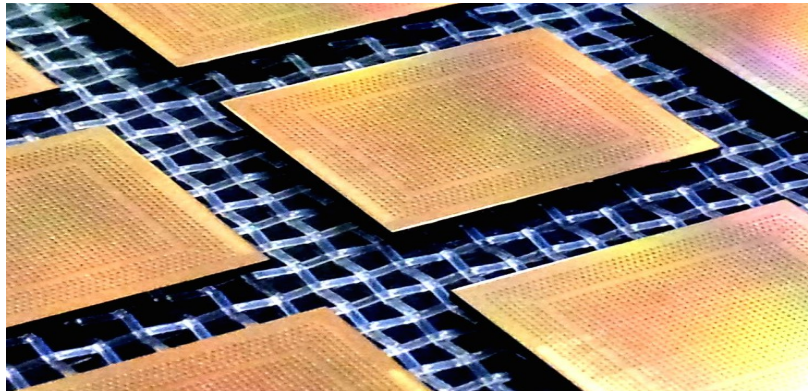Network occupies only 10% of the area
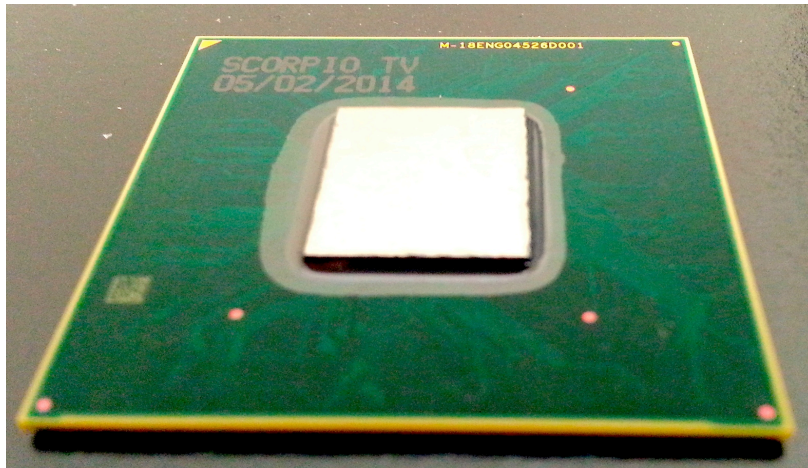
## Area

Network consumes 19% of the power

## Power

# Conclusion



SCORPIO Die



SCORPIO Packaged & Assembled

Mounting SCORPIO Chip to Custom Designed PCB Board

Intend to Run Linux OS and Applications on the Chip

Releasing SCORPIO NoC RTL after Measurements Completed