Replica: A Wireless Manycore for Communication-Intensive and Approximate Data

Vimuth Fernando[1], Antonio Franques[1], Sergi Abadal[2], Sasa Misailovic[1], Josep Torrellas[1]

[1]University of Illinois at Urbana-Champaign

[2] Universitat Politècnica de Catalunya

CCF-1629431
CCF-1703637

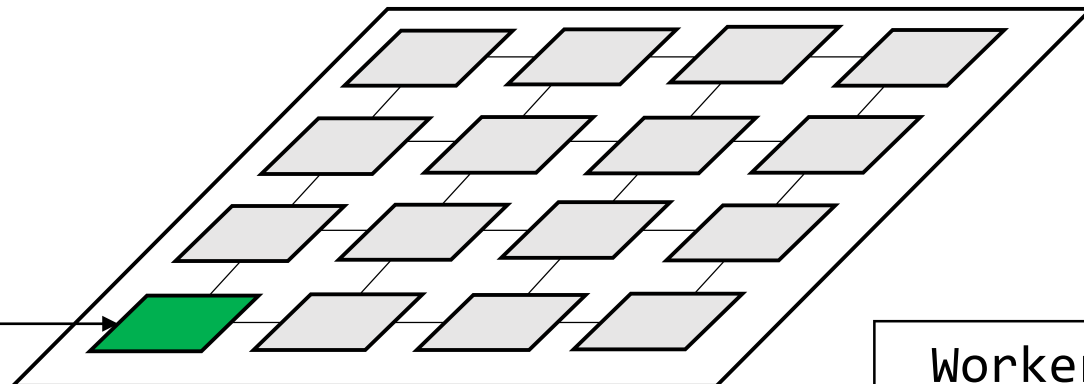
Motivation

Computations with broadcast and fine-grained data sharing do not scale well in shared-memory multiprocessor architectures

```
Master Thread

counter++;

barrier_wait(b)
```

```
Worker Threads

barrier_wait(b)

x = counter;
```

Manycore with a Network on Chip



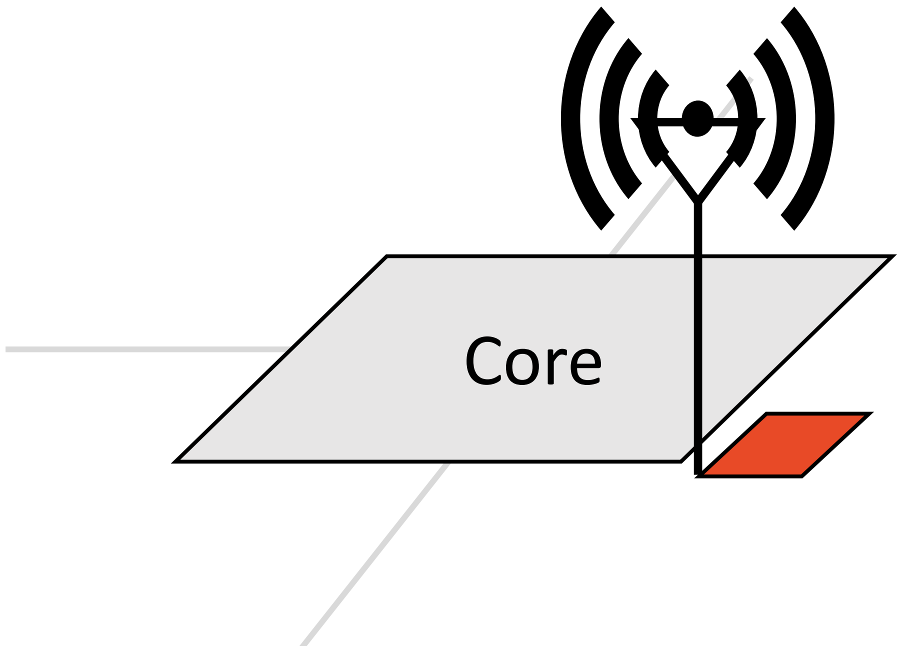
Master Thread

```
counter++;

barrier_wait(b)
```

Worker Threads

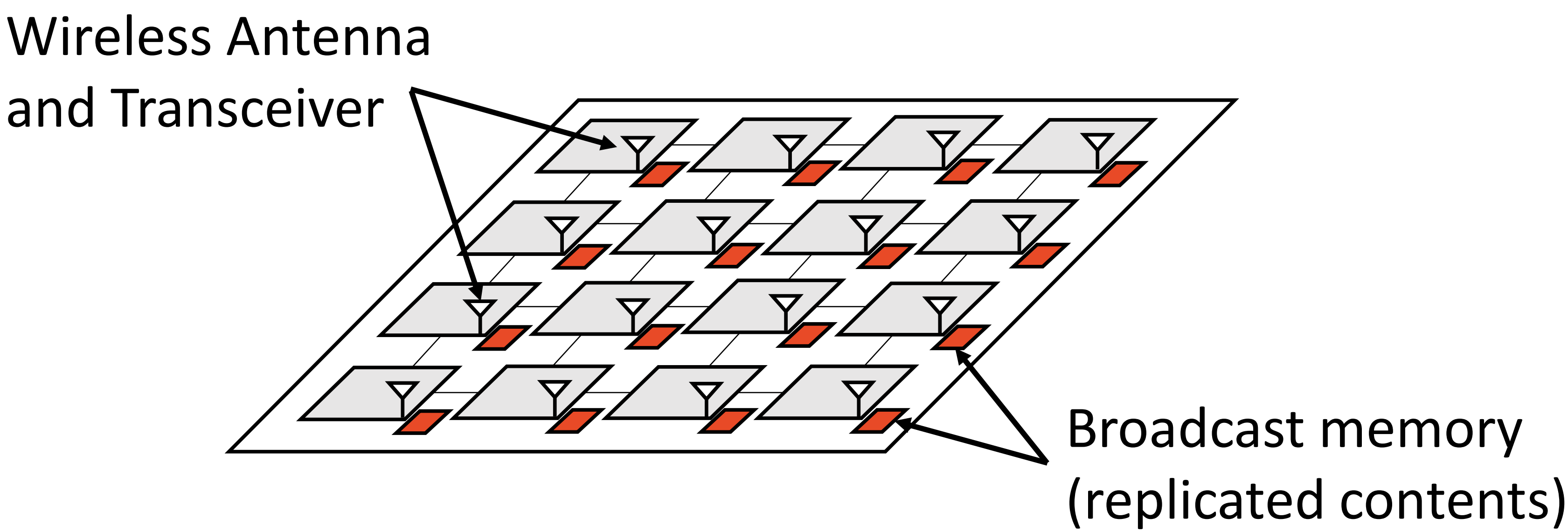
```
barrier_wait(b)

x = counter;
```

WiSync: On-chip Wireless Communication for Synchronization



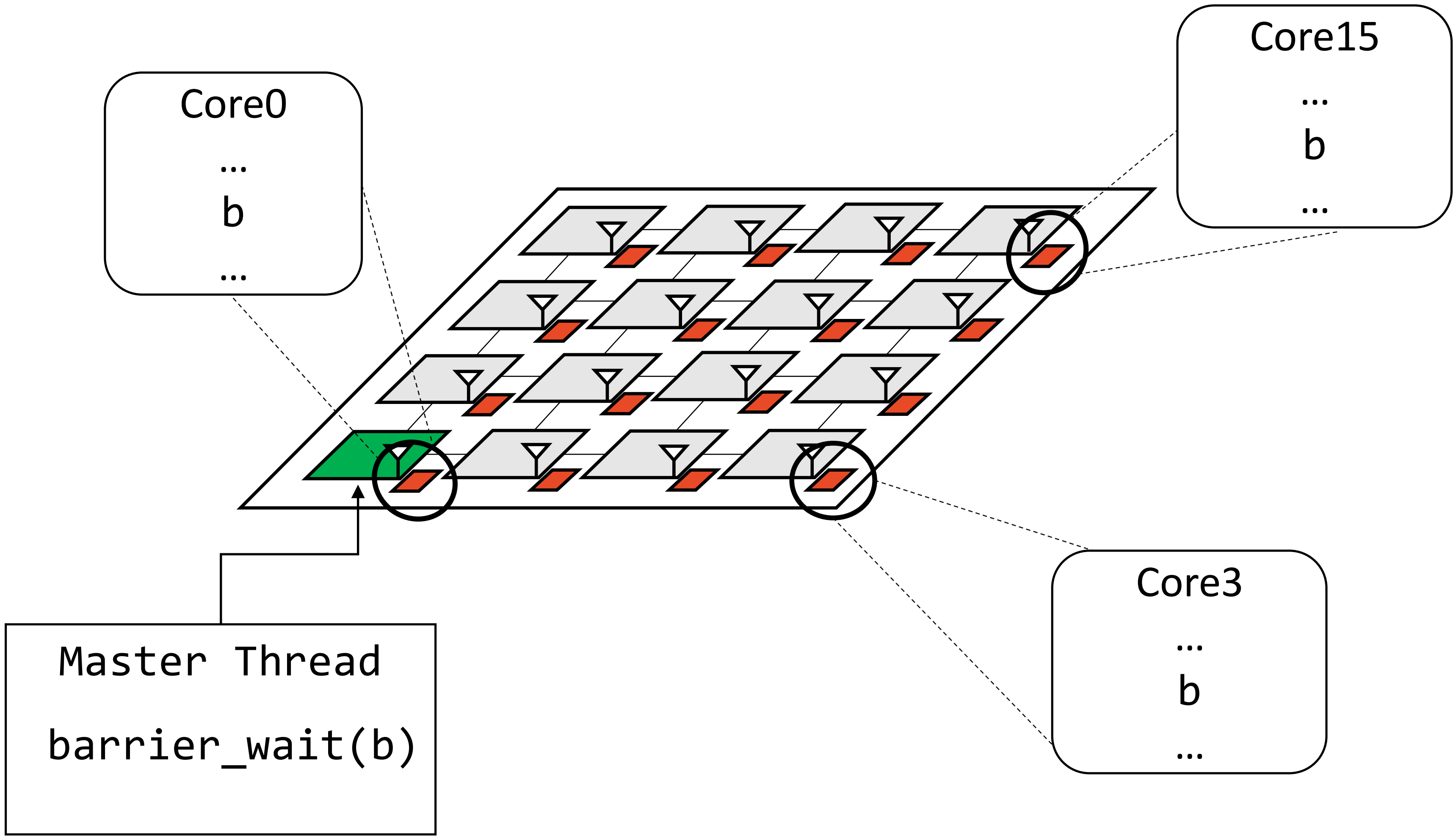
Abadal et al. "WiSync: an architecture for fast synchronization through on-chip wireless communication." ASPLOS 2016

WiSync: On-chip Wireless Communication for Synchronization

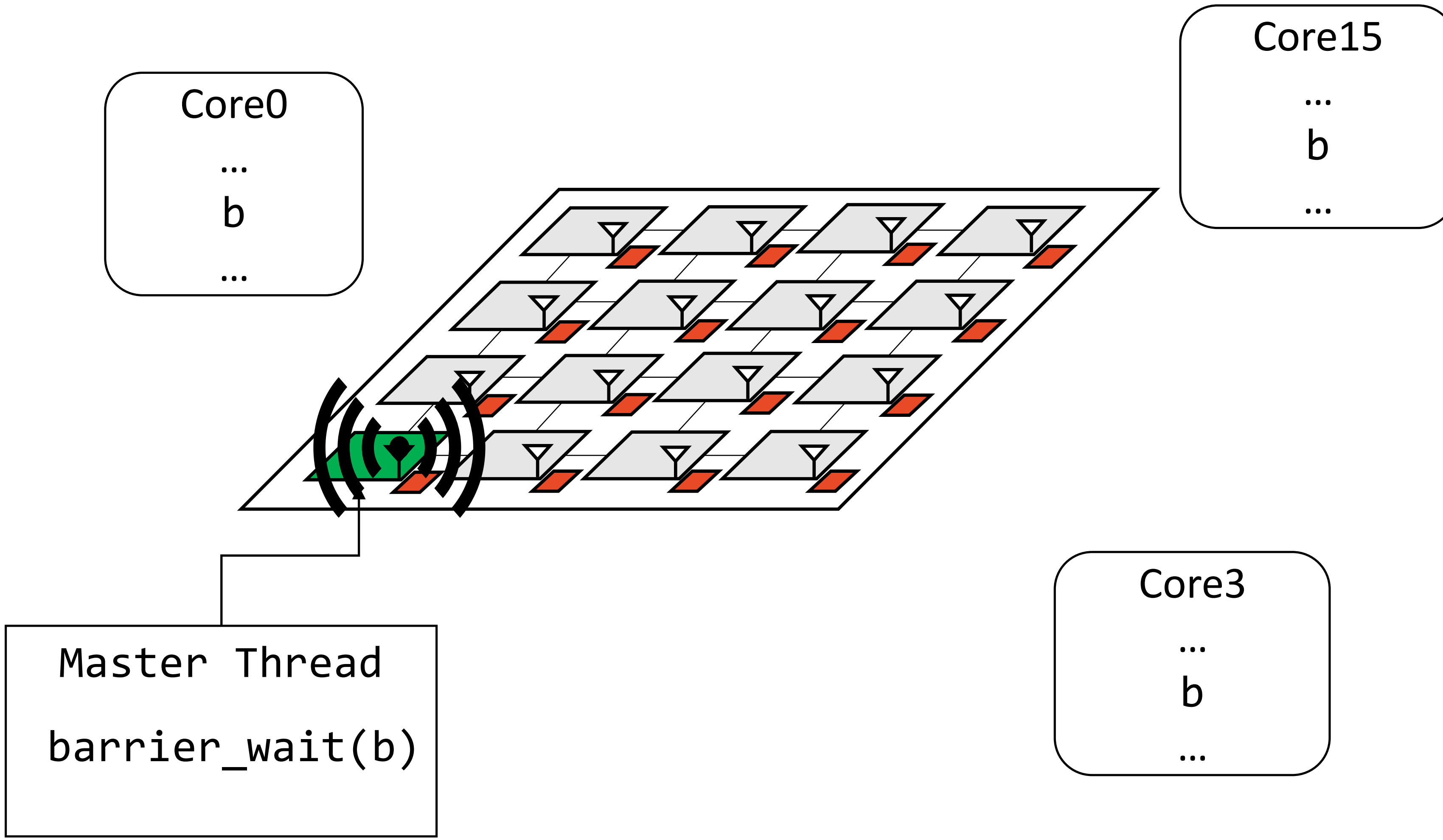


Abadal et al. "WiSync: an architecture for fast synchronization through on-chip wireless communication." ASPLOS 2016

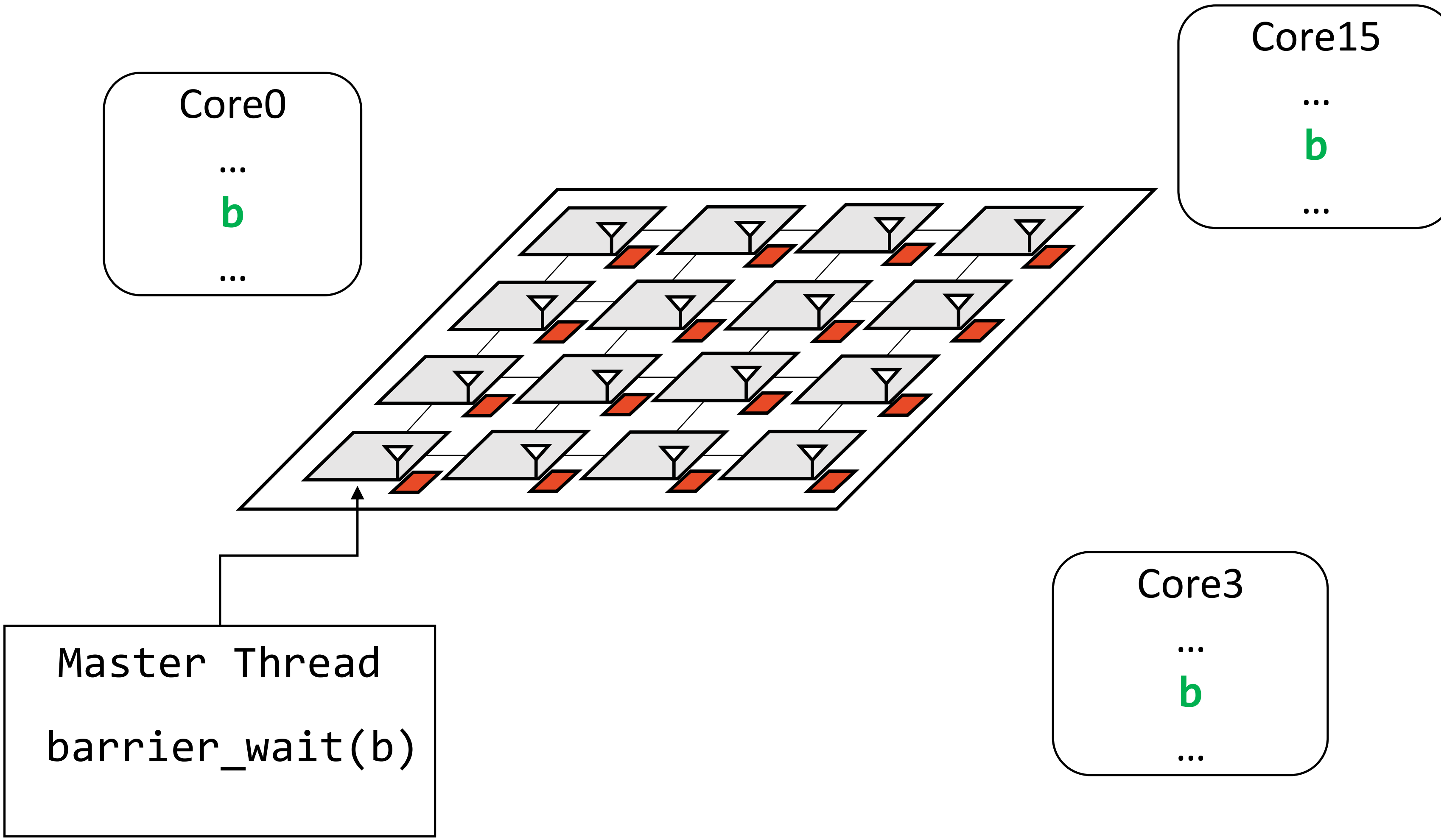
WiSync: On-chip Wireless Communication for Synchronization



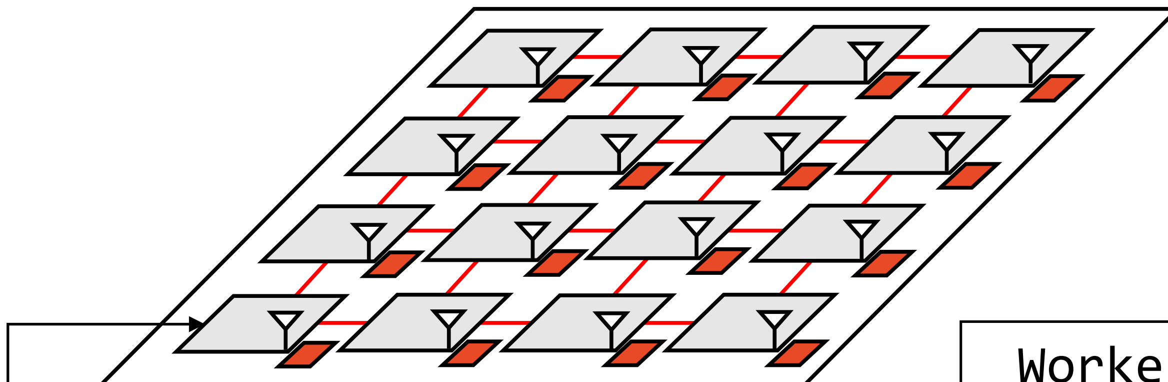
WiSync: On-chip Wireless Communication for Synchronization



WiSync: On-chip Wireless Communication for Synchronization



In WiSync, ordinary data uses the wired network



Master Thread

```
counter++;
```

Worker Threads

```
x = counter;
```

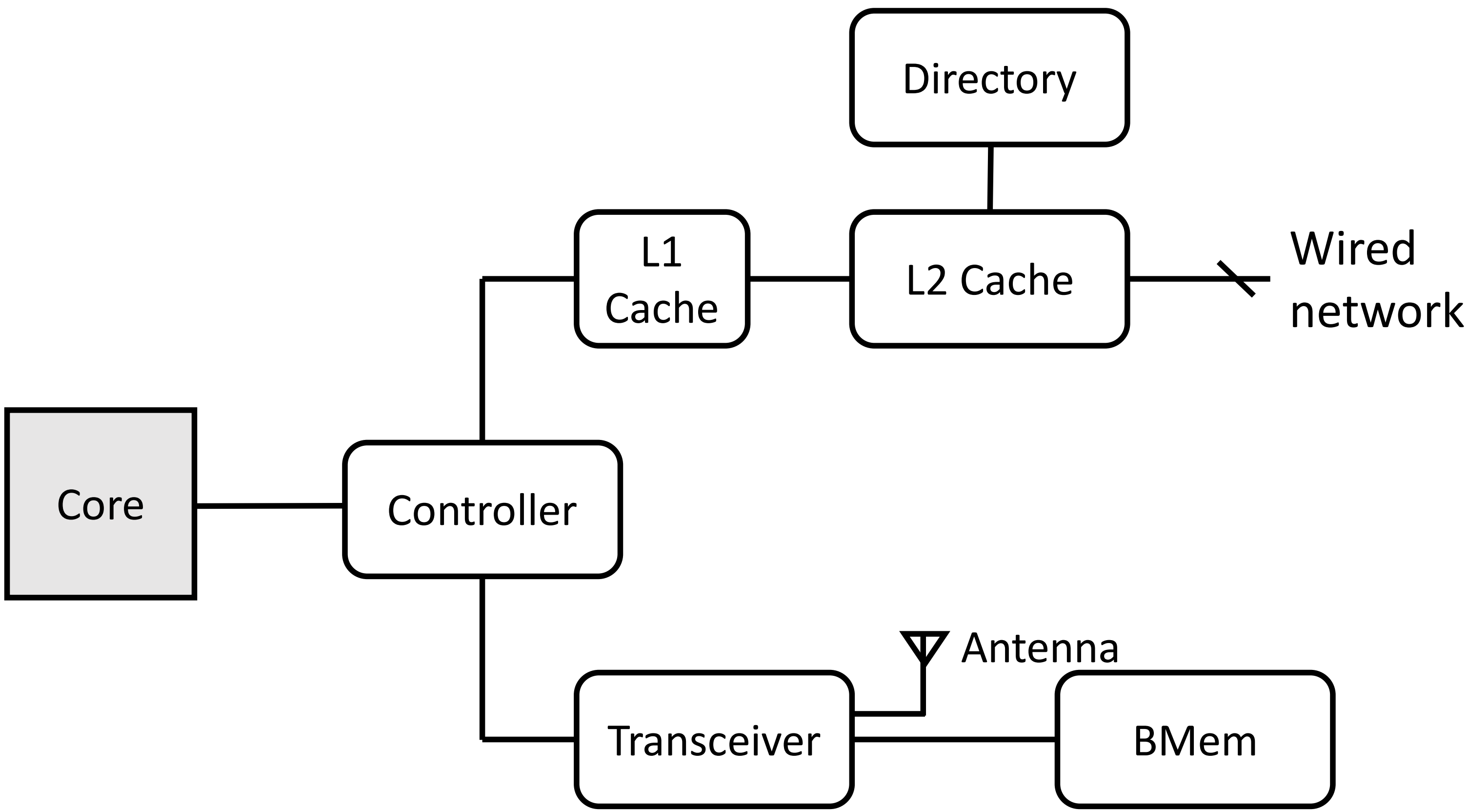
Key Question

Can we leverage wireless communication to speed-up transfers of ordinary shared data?

Contributions: Replica

- A manycore architecture and software interface for wireless communication (sync and ordinary data)
- Hardware innovations
  - Adaptive wireless protocol
  - Selective packet dropping
- Software innovations
  - Transformations and tools to adapt applications to wireless
  - Optimizations for approximate computing
- For 64 core execution: speedup applications by 1.89x over a conventional multicore

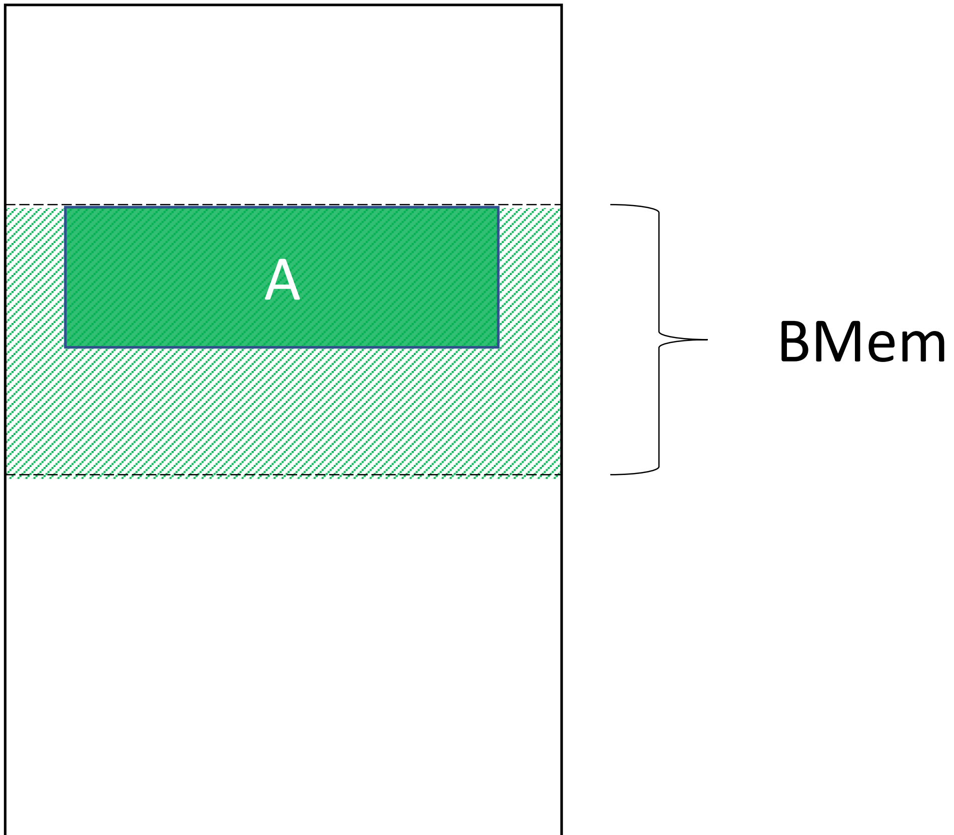
Replica Architecture



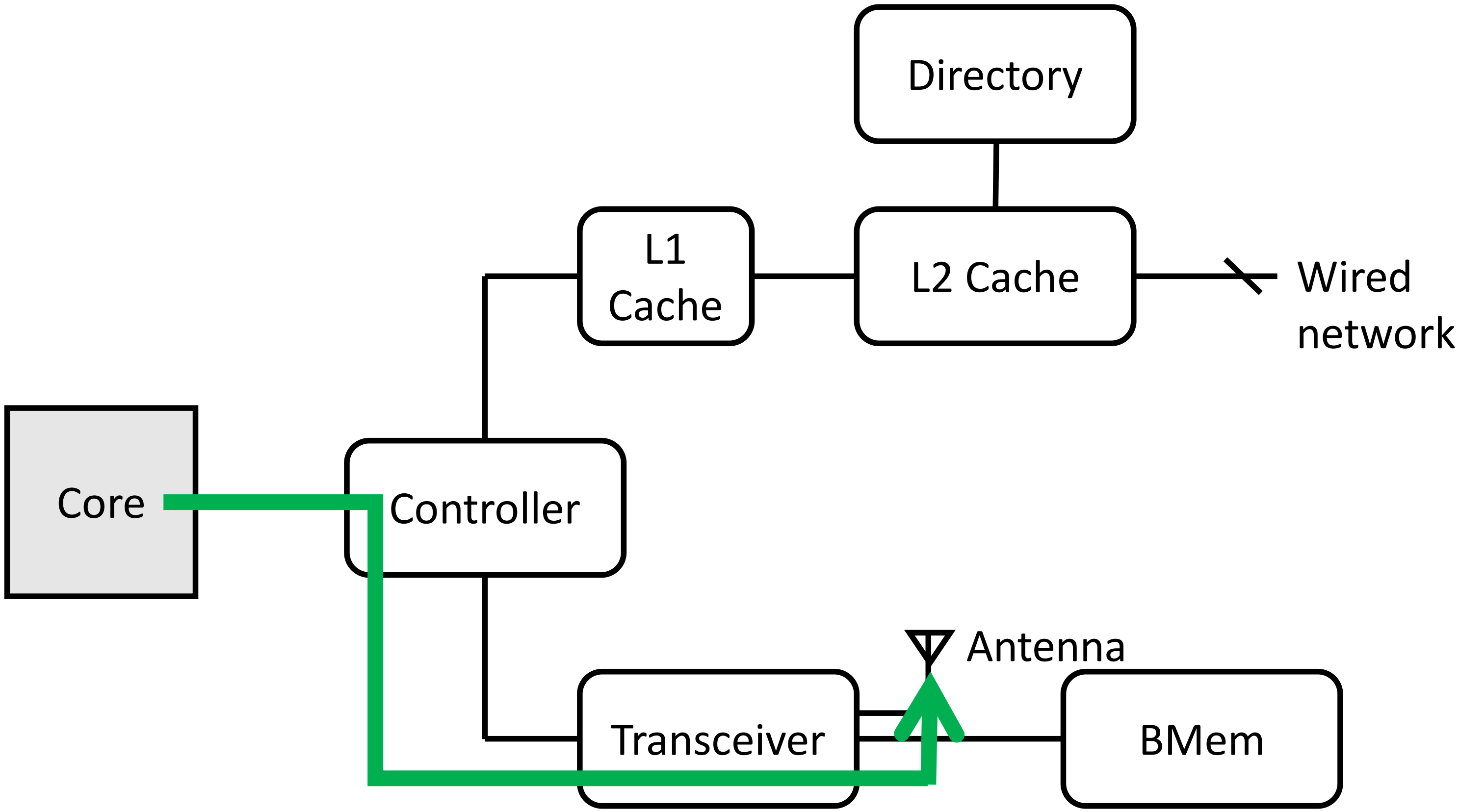
32-512 KB

Example

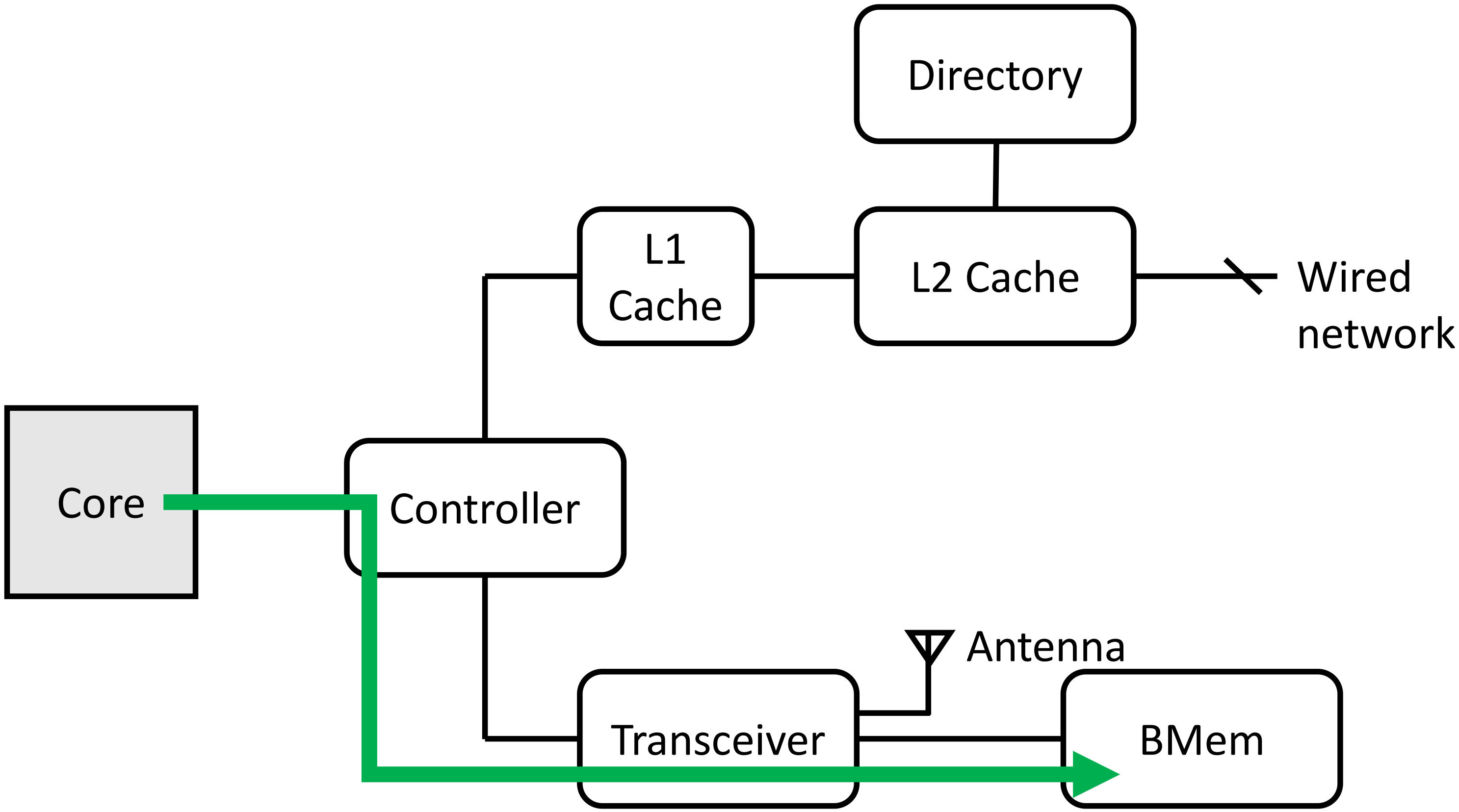
```
int* A = (int*) wireless_malloc(size)
```



Write

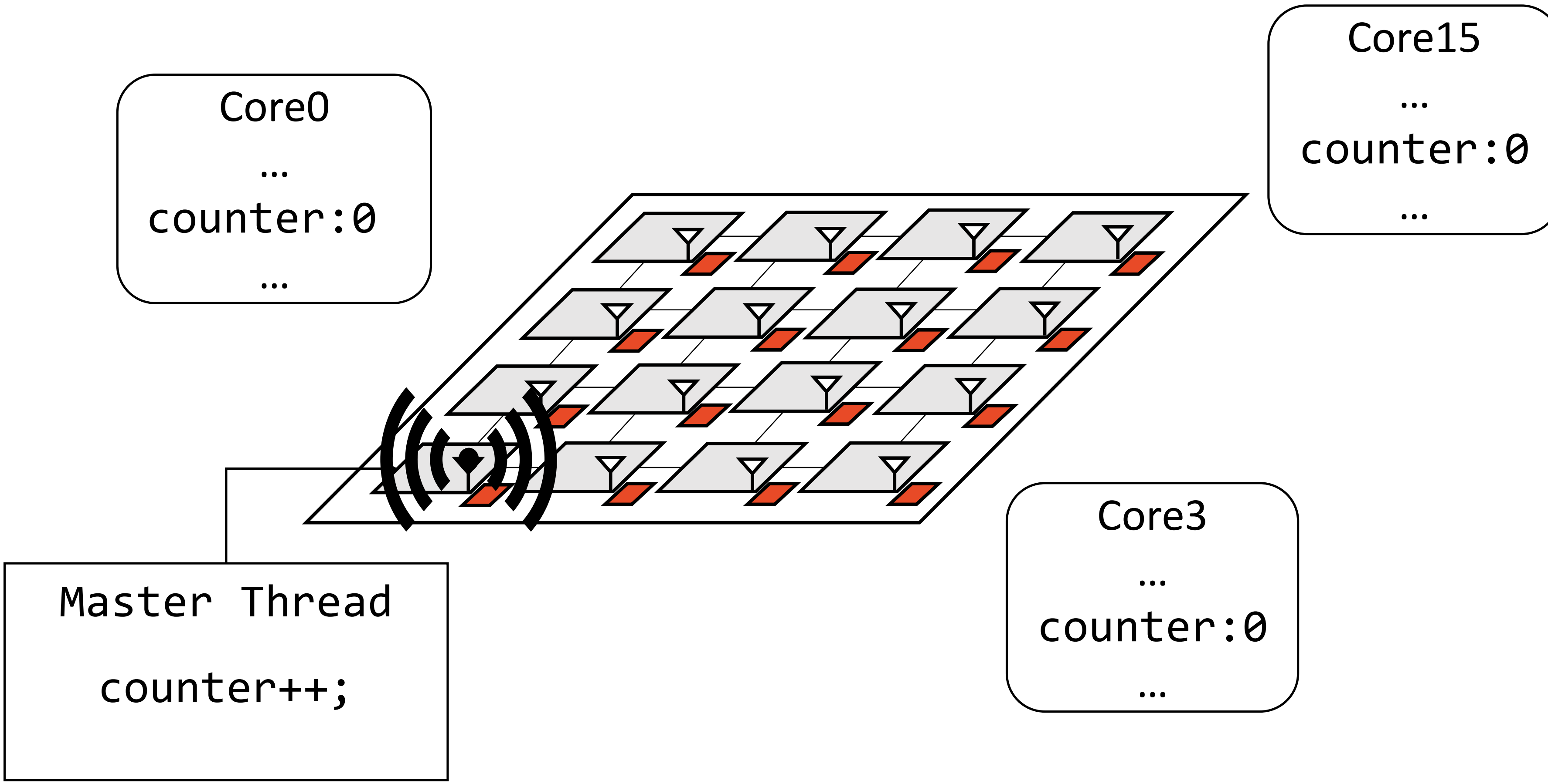

Directory
L1
Cache
L2 Cache
Wired
network
Core
Controller
Antenna
Transceiver
BMem

Write

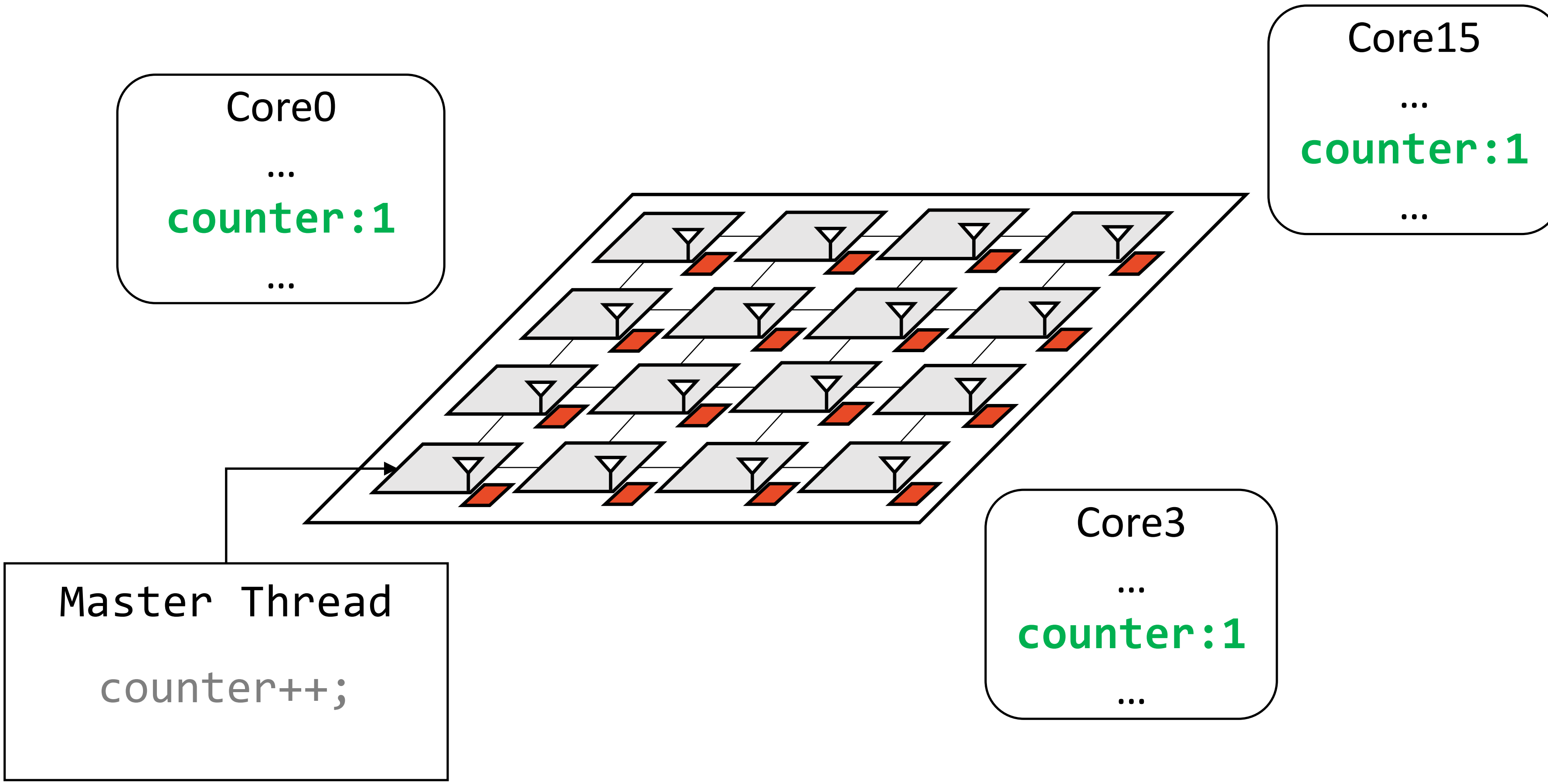


Atomic update of local and all remote BMems

Broadcast Memory for ordinary data

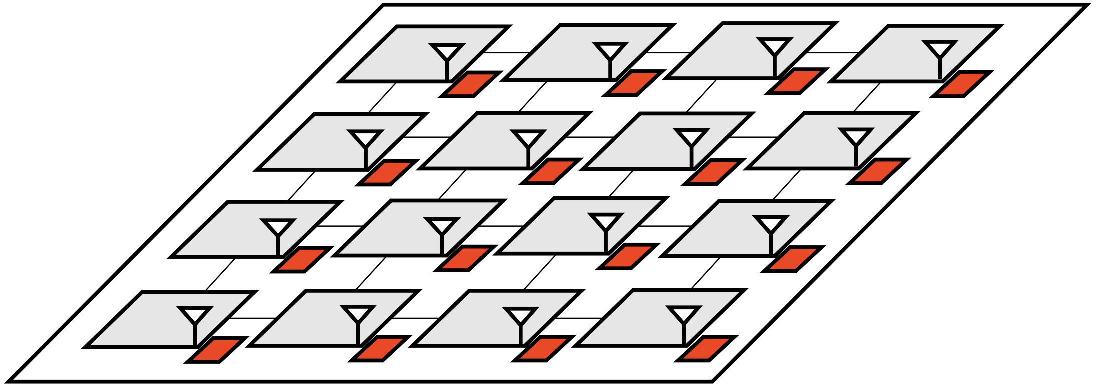


Broadcast Memory for ordinary data

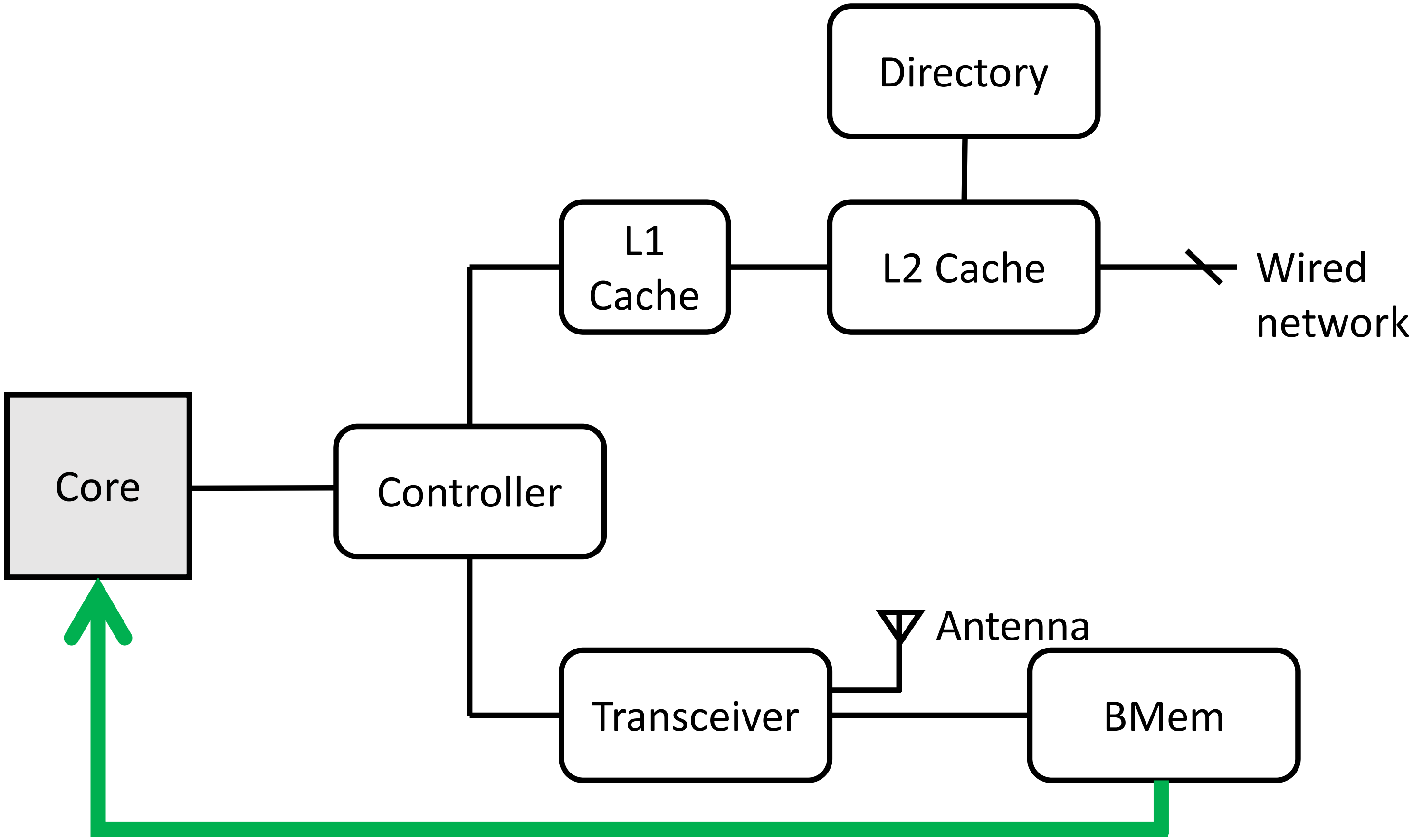


Replica: Wireless channel

- One channel shared by all the cores
- Everyone receives what one core transmits
- Only one core can transmit at a given time
  - ensures the same order of updates across all BMems



Reads



Read: Local access

Challenges

- Limited wireless bandwidth: Only one core can transmit at a time

- Bounded size of the BMem: Arbitrary data structures will not fit

Solutions

- Limited wireless bandwidth: Only one core can transmit at a time
  - Adaptive wireless protocol
  - Selective message dropping
  - Approximate transformations to use less bandwidth
- Bounded size of the BMem: Arbitrary data structures will not fit

Solutions

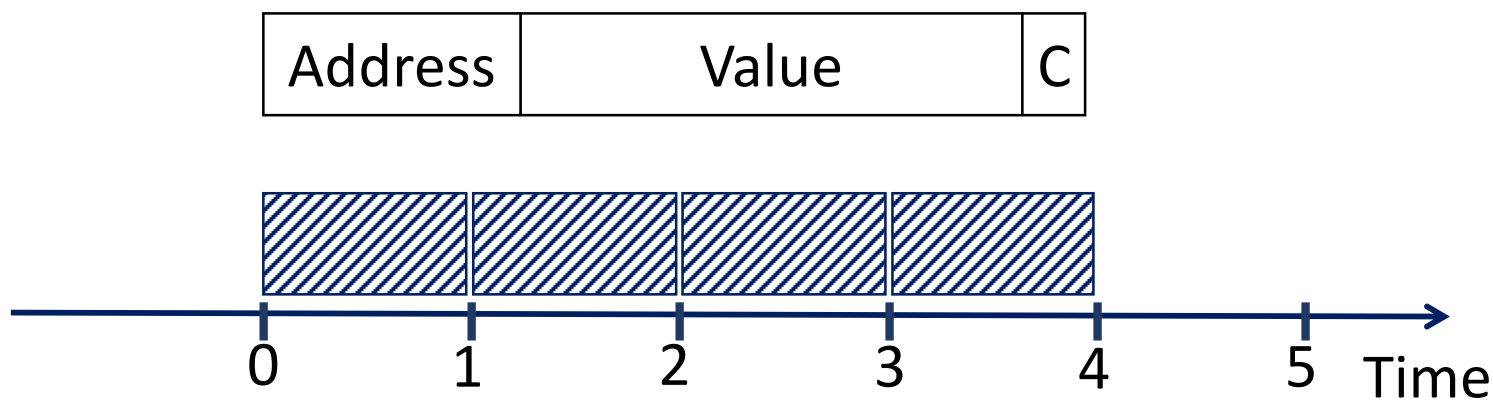
- Limited wireless bandwidth: Only one core can transmit at a time
    - Adaptive wireless protocol
    - Selective message dropping
    - Approximate transformations to use less bandwidth

- Bounded size of the BMem: Arbitrary data structures will not fit
    - Software transformations to fit most important structures in BMem
    - Approximate transformations to use BMem effectively
    - Tools to identify/autotune highly-shared data structures

Wireless Protocol

- Wireless protocol organizes the accesses to the wireless network
- Two wireless protocols can be used based on application behavior
  - Broadcast Reliability Sensing protocol (BRS)
  - Token Ring protocol

Wireless Message

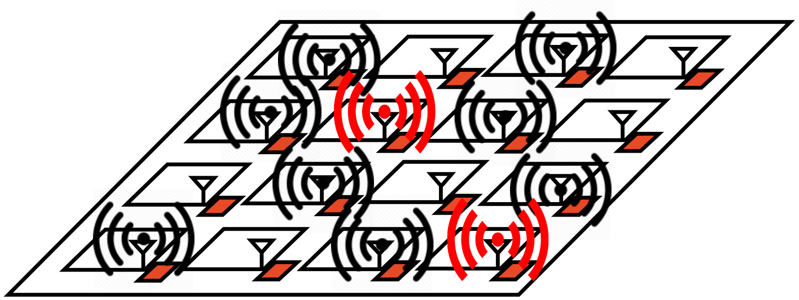


4 cycles at 20Gb/s*

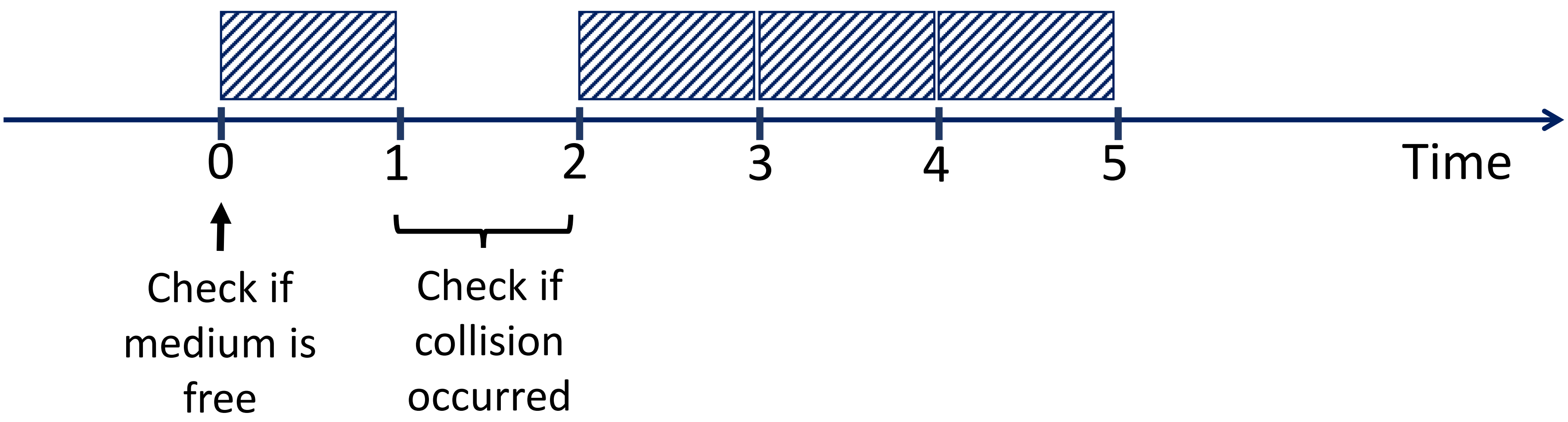
* Yu, et al. “Architecture and Design of Mul5-Channel Millimeter-Wave Wireless Network-on-Chip,” IEEE Design & Test, 2014 (scaled)

Broadcast **R**eliability **S**ensing Protocol (BRS)

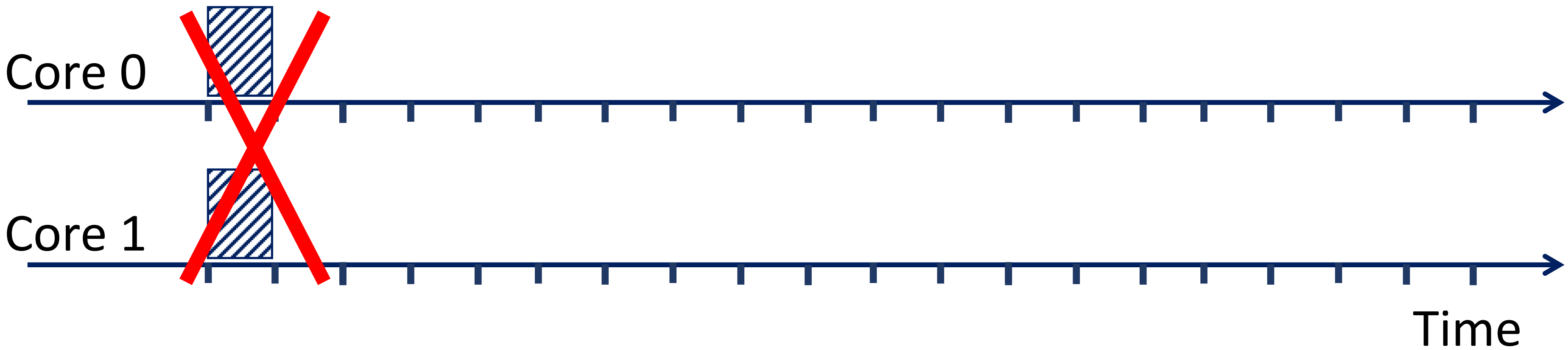
- Start sending message if the medium is free
- Two cores starting at the same time results in a collision



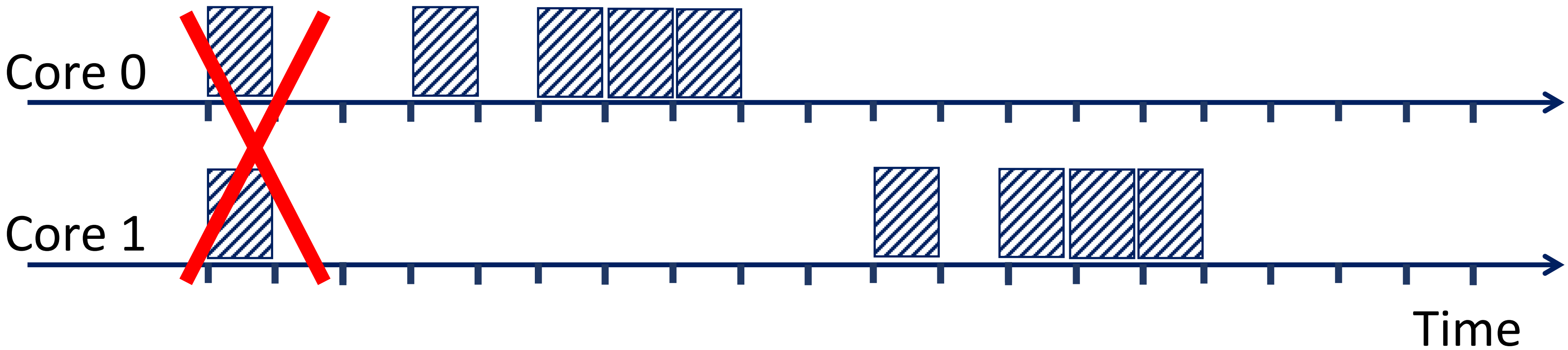
Broadcast **R**eliability **S**ensing Protocol (BRS)



Broadcast **R**eliability **S**ensing Protocol (BRS)



Broadcast **R**eliability **S**ensing Protocol (BRS)

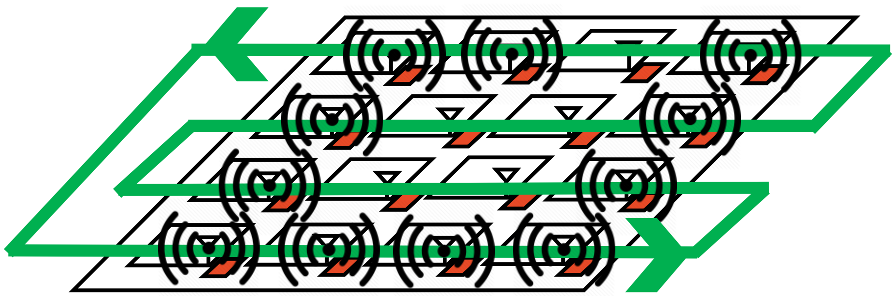


\+ No wasted cycles if low contention

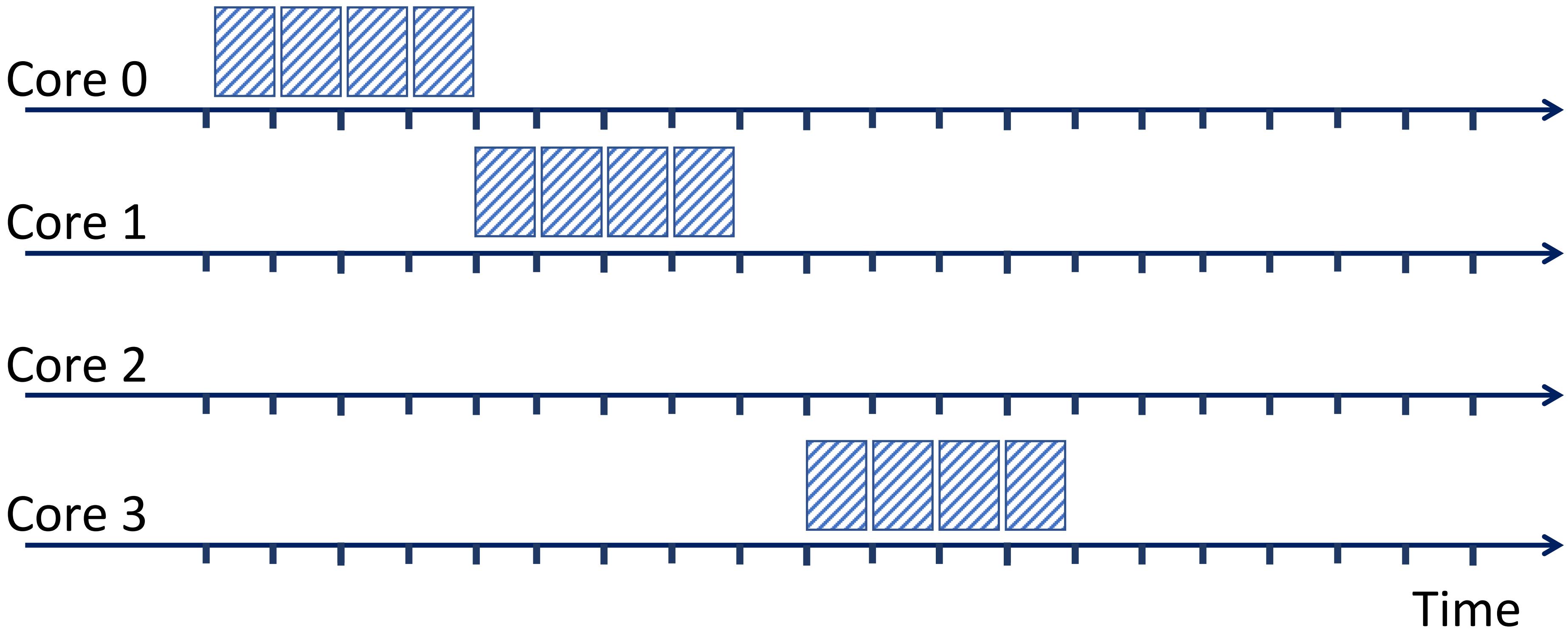
\- Lot of collisions if high contention

Token Ring Protocol

- Pass conceptual token among cores
- Can send wireless message only if the core *owns* the token



Token Ring Protocol



+ No wasted cycles if high contention

− Unnecessary delays if low contention

Adaptive Wireless Protocol

- In Replica, the utilization of the wireless network vary across applications and within an application
  - Sparse traffic – BRS
  - Bursty traffic – Token Ring

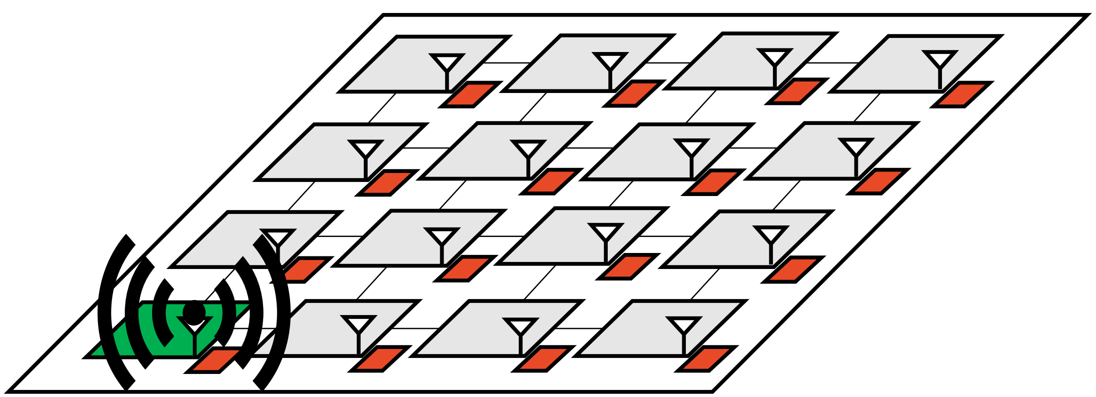
- Replica uses an adaptive dynamic protocol that switches between the two by observing communication behavior
  - Number of collisions
  - Number of skipped token slots

Approximate transformations to use less bandwidth

- Every write to data in the BMem results in a message being broadcasted
- We can reduce the pressure on the network by skipping some of the writes
  - Reducing communication at the cost of accuracy
- Many programs have shared data structures that are amenable to approximations

Opportunity in Replica: Dropping Messages

- All cores see the contention in the wireless network
- Can drop messages while maintaining the same state across all cores

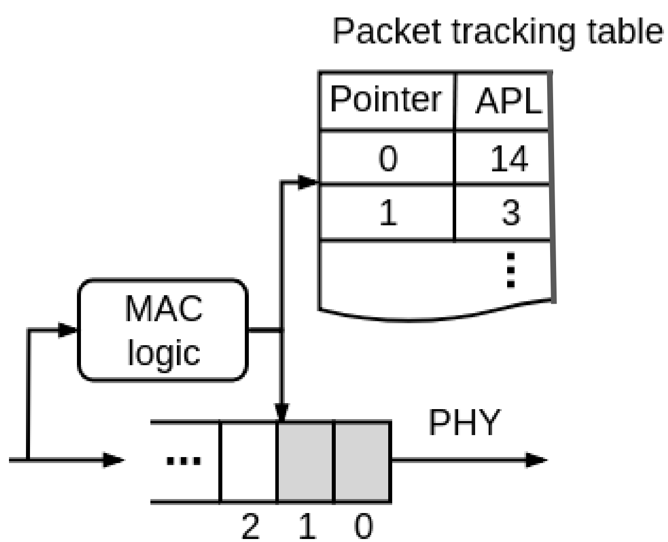


Approximate stores

- Developers indicate approximable data structures

  `approx_wireless_malloc(size)`

- Stores to approximable variables are dropped if they cannot access the wireless network before a given threshold

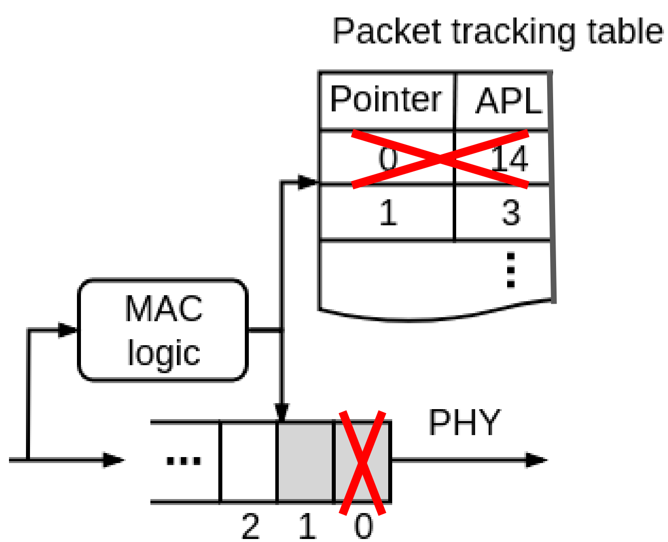


Approximate stores

- Developers indicate approximable data structures

  `approx_wireless_malloc(size)`

- Stores to approximable variables are dropped if they cannot access the wireless network before a given threshold



Approximate transformations to use less bandwidth

- We used the approximate stores to implement primitives such as Approximate Locks
  - Spin lock that gives up trying to acquire a lock after some time

- Existing approximate techniques that reduce communication more useful in this resource constrained setting
  - Example: Skipping negligible updates to shared data

Addressing Bounded size of the BMem

- Software transformations to fit most important structures in BMem
- Approximate transformations to use BMem effectively
  - Example: Numerical precision reduction, Cyclic collection update
- Tools to identify highly-shared data and tune the application

See the paper for more details

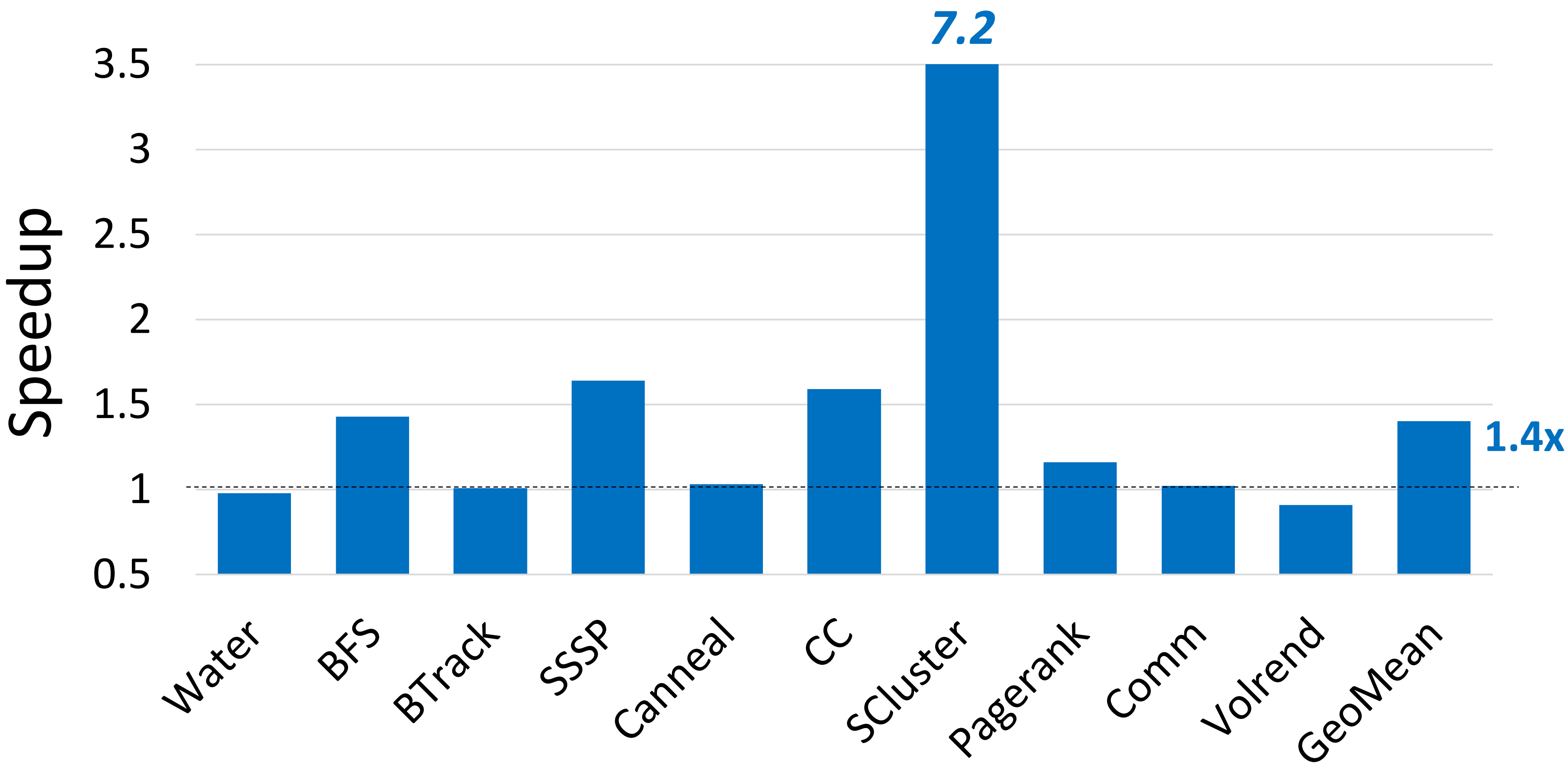
Evaluation

- Cycle-level architectural simulations using Multi2sim
  - 64 core chip
  - 32-512 KB BMem
  - 2D Mesh wired network
- Applications
  - 10 benchmarks from PARSEC and CRONO
  - Multiple domain: Scientific simulations, computer vision, and graph applications

Benchmarks: Communication Patterns

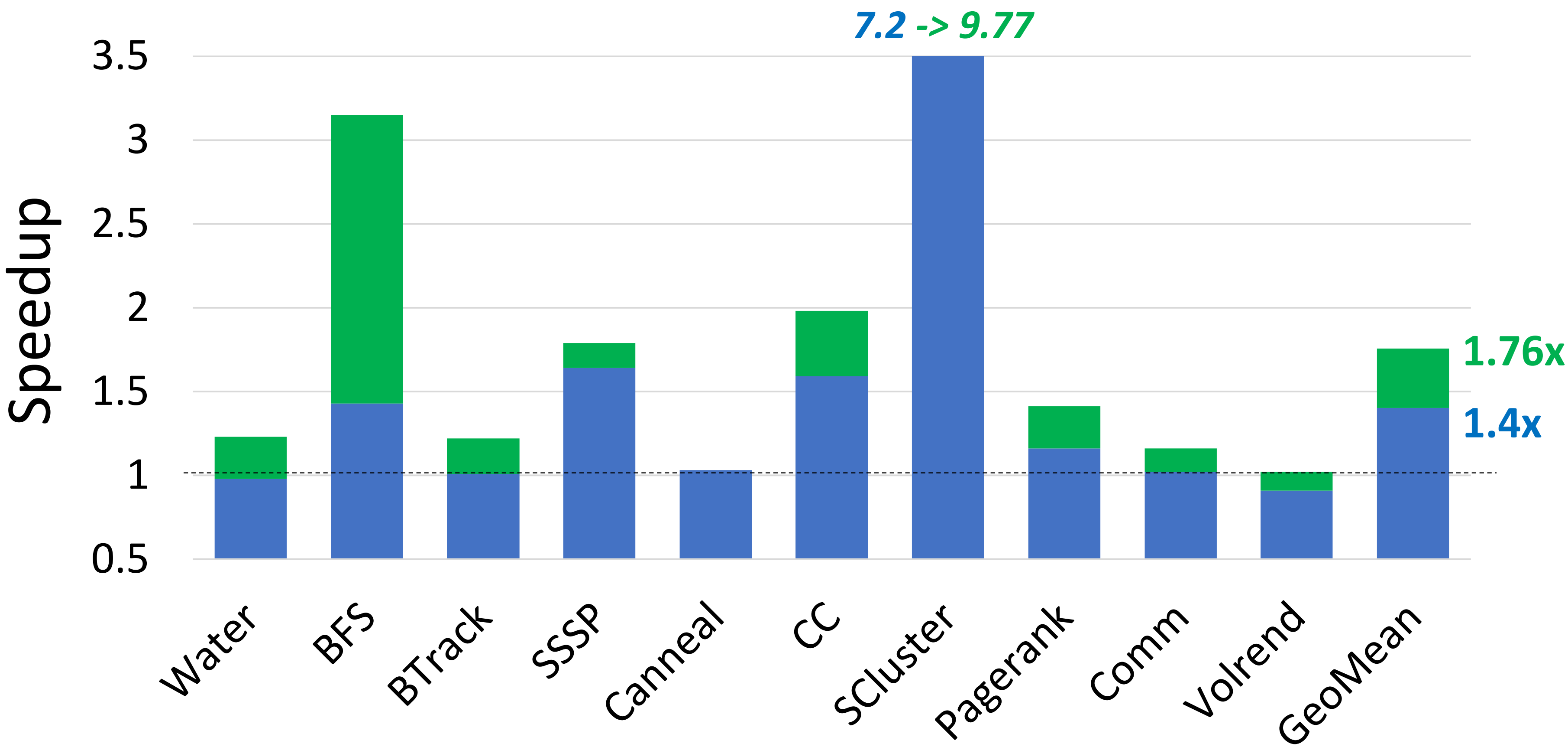
Benchmark	Sharing Pattern
Water	Broadcast
BFS	Irregular: many-to-many
Bodytrack	One-to-many
SSSP	Irregular: many-to-many
Canneal	Irregular
CC	Irregular: many-to-many
Streamcluster	One-to-many, reduction
Pagerank	Irregular: many-to-many
Community	Irregular: many-to-many
Volrend	One-to-many

BMem for sync variables (WiSync)



1.4x speed up over conventional wired multicore (Geometric Mean)

BMem for shared data

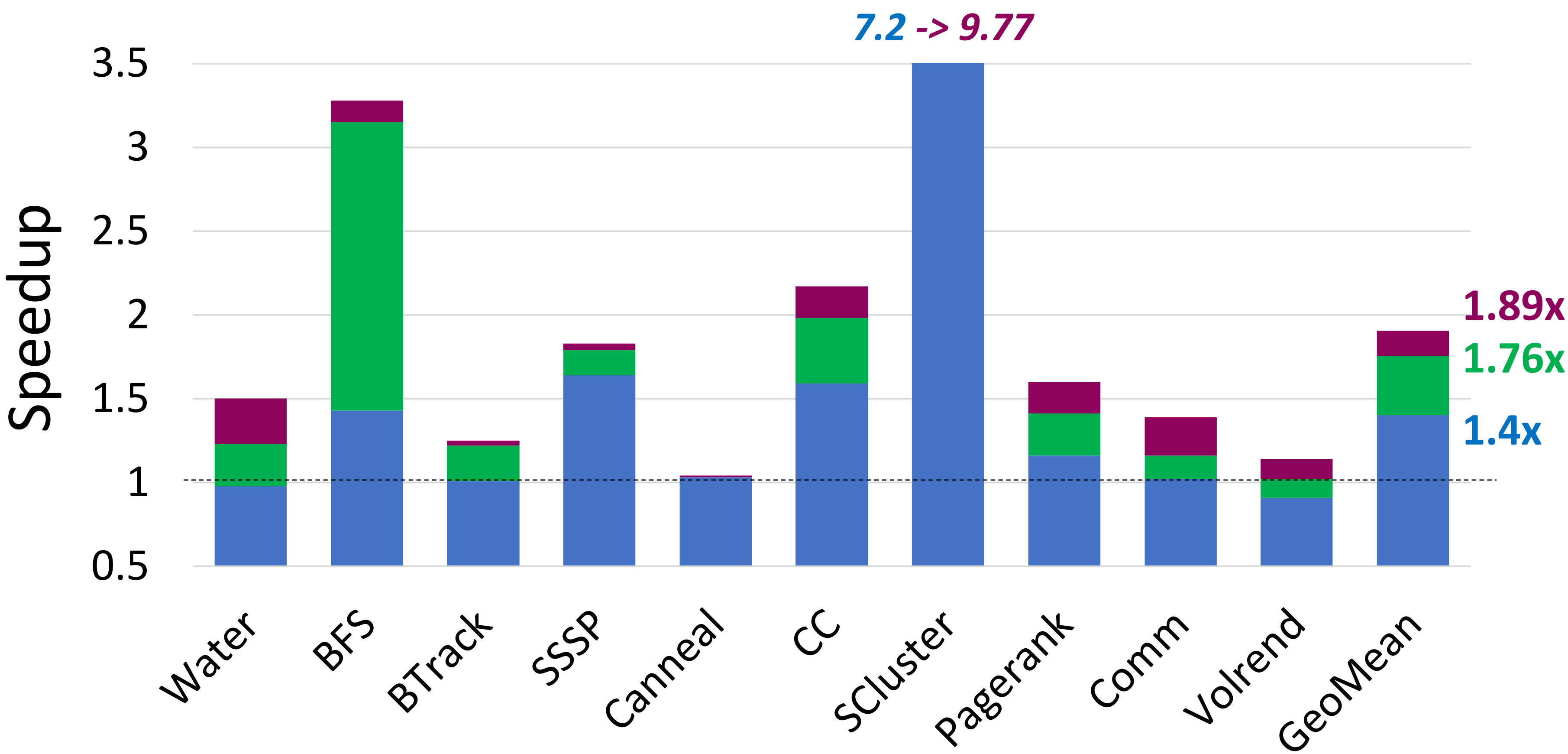


1.76x speed up (Geometric Mean)

Benchmarks: Approximation

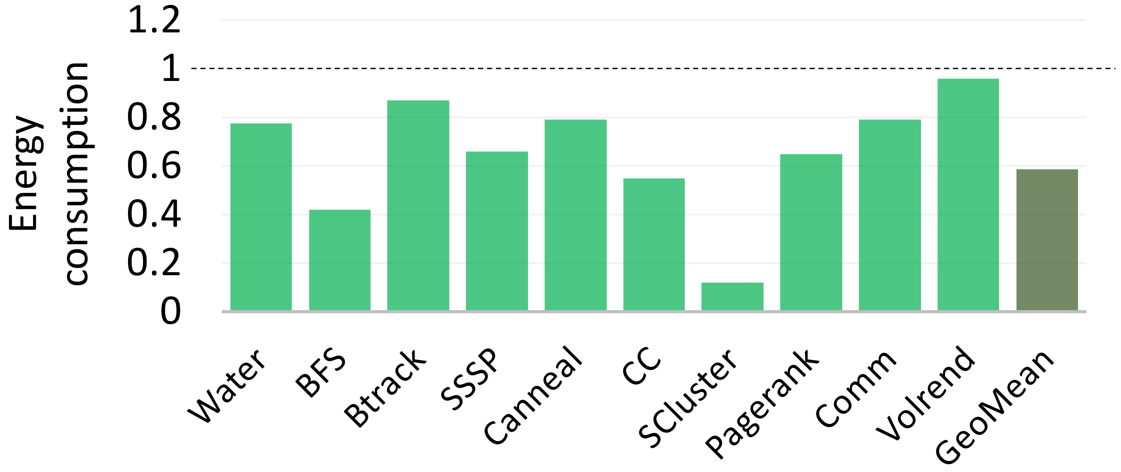
Benchmark	**Sharing Pattern**	**Approximations**
Water	Broadcast	Precision reduction and Approximate Locks
BFS	Irregular: many-to-many	Approximate Stores
Bodytrack	One-to-many	Approximate Stores
SSSP	Irregular: many-to-many	Approximate Stores
Canneal	Irregular	Approximate Locks
CC	Irregular: many-to-many	Approximate Stores
Streamcluster	One-to-many, reduction	Cyclic collection updates
Pagerank	Irregular: many-to-many	Skipping negligible updates
Community	Irregular: many-to-many	Approximate Stores
Volrend	One-to-many	Approximate Stores

BMem for shared data + approximations



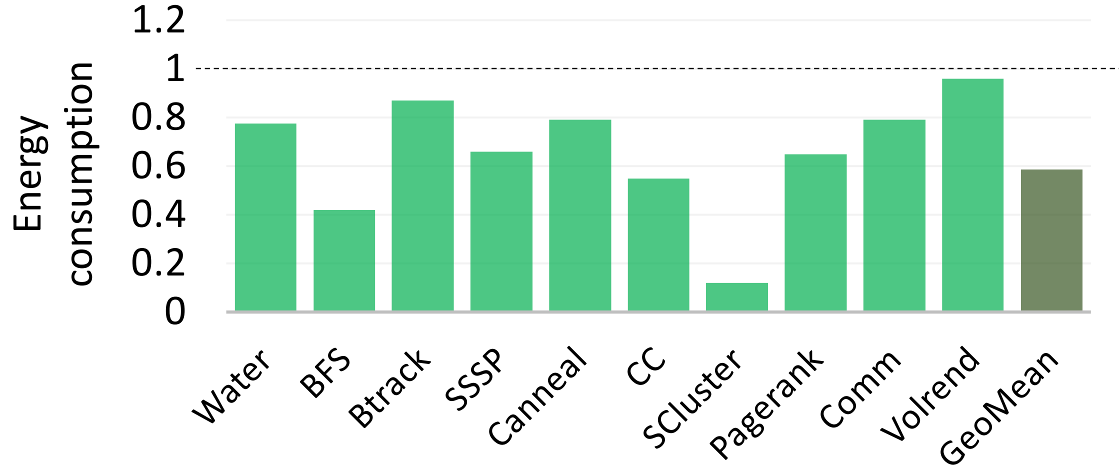
On average 1.89x speed up

Energy and area



- Since faster execution: 33% energy reduction
- Replica components: 9% of total energy consumed

Energy and area



- Since faster execution: 33% energy reduction
- Replica components: 9% of total energy consumed

- 15% increase in the area
  - 11% from the BMem + 4% from the transceiver/antenna
  - Using the same area to increase the L2 cache has little impact on performance (1.04x speedup)

Also in the paper

- Scalability analysis
- Power evaluation
- Area consumption
- Architecture sensitivity analysis
- Effectiveness of profiler and autotuner
- Statistics on developer effort to adapt programs

Conclusions

- Replica: a manycore that uses a wireless NoC to communicate ordinary data
- Hardware and Software innovations
  - Adaptive wireless protocol
  - Selective packet dropping
  - Software techniques to identify and allocate shared data in BMem
  - Software transformations for approximate computing
- Effectively supports communication-intensive computations
- Average speedup of 1.89x over conventional machines