A Case for Bufferless Routing in On-Chip Networks

Thomas Moscibroda
Microsoft Research

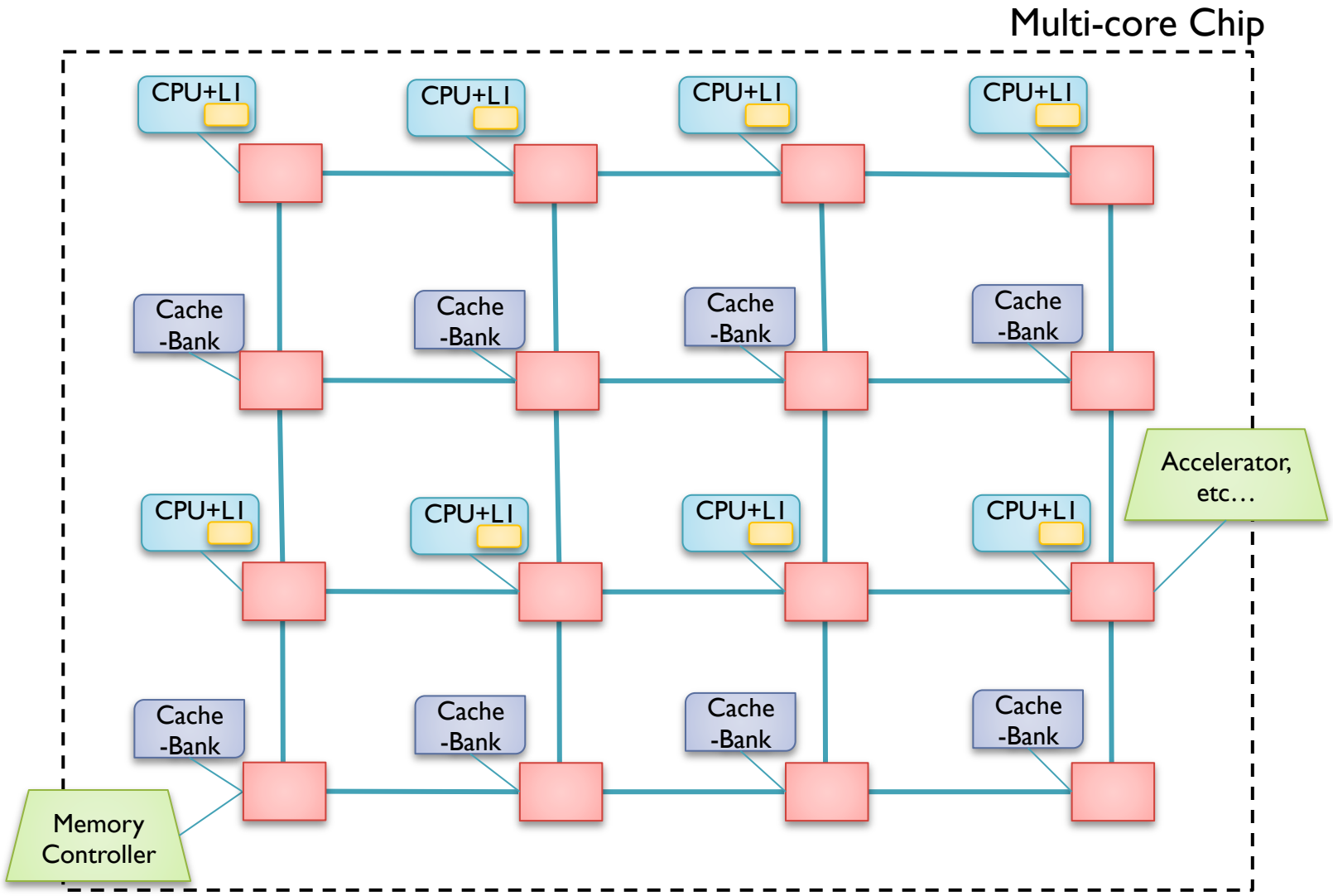
Onur Mutlu
CMU

Microsoft® Research

Carnegie Mellon

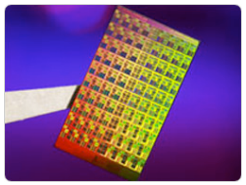
TexPoint fonts used in EMF

On-Chip Networks (NoC)



On-Chip Networks (NoC)

- Connect cores, caches, memory controllers, etc…
- Examples:
  - Intel 80-core Terascale chip
  - MIT RAW chip



- Design goals in NoC design:
  - High throughput, low latency
  - Fairness between cores, QoS, …
  - Low complexity, low cost
  - Power, low energy consumption

On-Chip Networks (NoC)

- Connect cores, caches, memory controllers, etc…
- Examples:
  - Intel 80-core Terascale chip
  - MIT RAW
- Design goals in
  - High throu
  - Fairness be
  - Low comp
  - Power, low

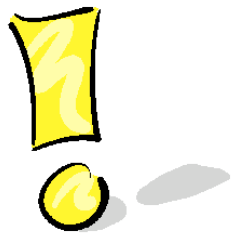
Energy/Power in On-Chip Networks

- Power is a key constraint in the design of high-performance processors
- NoCs consume substantial portion of system power
  - ~30% in Intel 80-core Terascale [IEEE Micro'07]
  - ~40% in MIT RAW Chip [ISCA'04]
- NoCs estimated to consume 100s of Watts [Borkar, DAC'07]

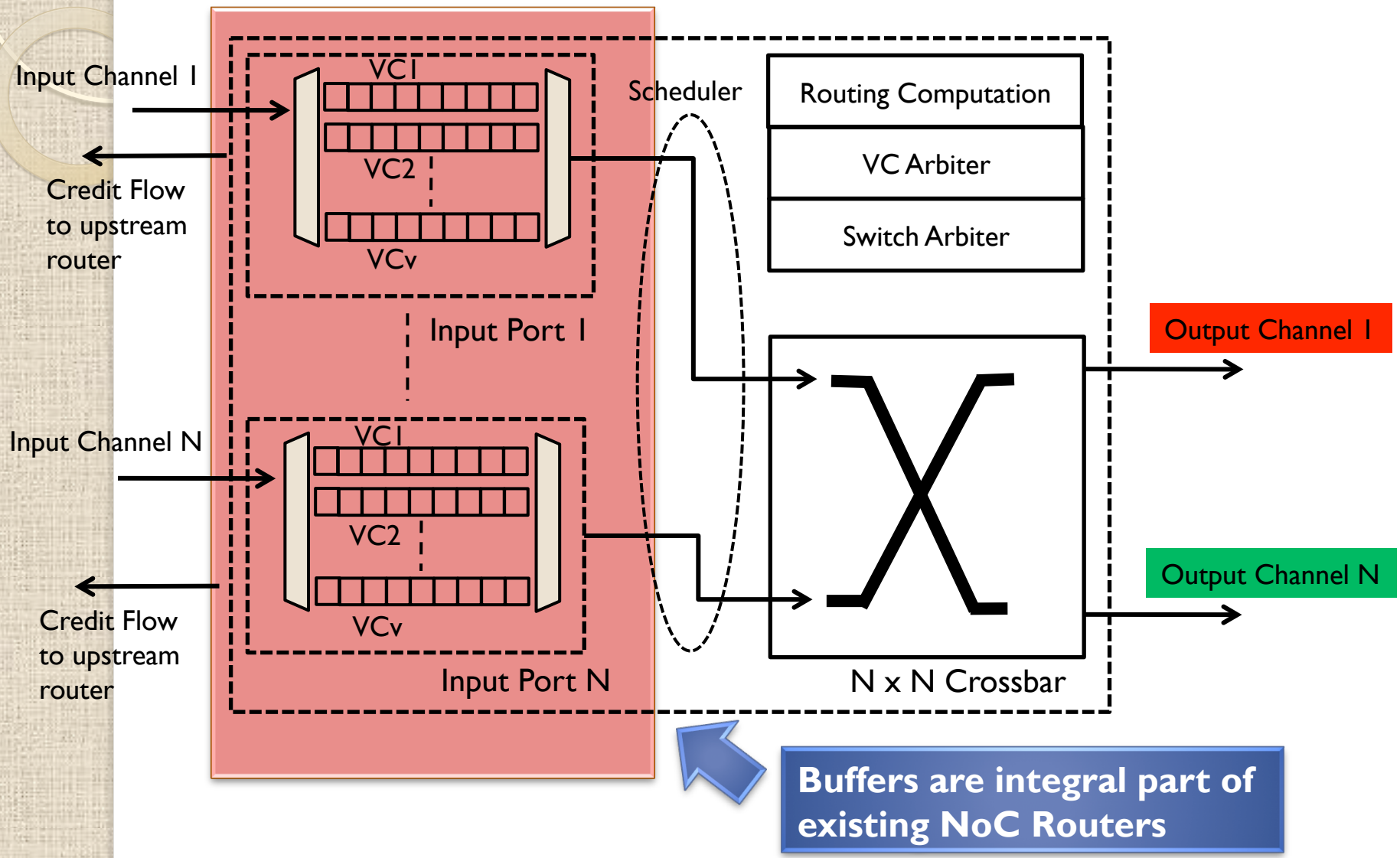
Current NoC Approaches

- Existing approaches differ in numerous ways:
  - Network topology [Kim et al, ISCA'07, Kim et al, ISCA'08 etc]
  - Flow control [Michelogiannakis et al, HPCA'09, Kumar et al, MICRO'08, etc]
  - Virtual Channels [Nicopoulos et al, MICRO'06, etc]
  - QoS & fairness mechanisms [Lee et al, ISCA'08, etc]
  - Routing algorithms [Singh et al, CAL'04]
  - Router architecture [Park et al, ISCA'08]
  - Broadcast, Multicast [Jerger et al, ISCA'08, Rodrigo et al, MICRO'08]

Existing work assumes existence of buffers in routers!

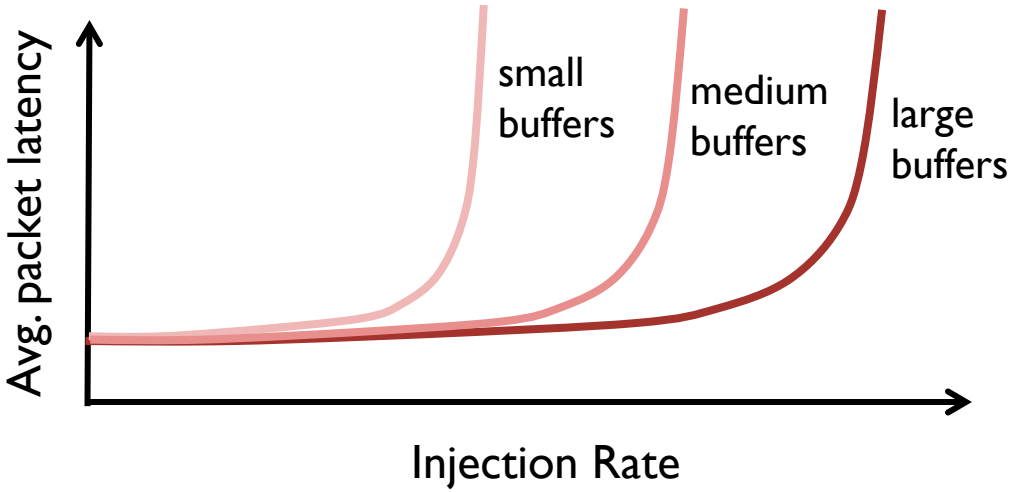


A Typical Router



Buffers in NoC Routers

- Buffers are necessary for high network throughput
  → buffers increase total available bandwidth in network



Buffers in NoC Routers

- Buffers are necessary for high networ
  → buffers increase total availab

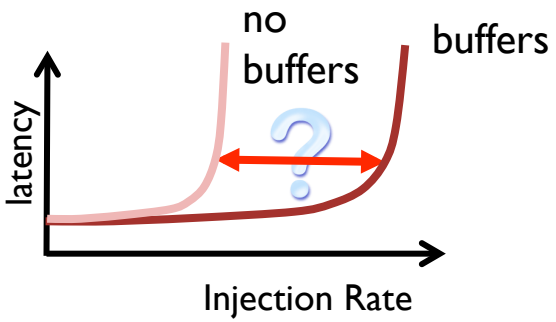
- Buffers consume sig
  - Dynamic e
  - Stati
- Buff
  - ency
  - ment
  - cation
  - flow control
- require significant chip area
  - e.g., in TRIPS prototype chip, input buffers occupy 75% of total on-chip network area [Gratz et al, ICCD'06]

Can we get rid of buffers…?!




Going Bufferless…?

- How much throughput do we lose?
  → How is latency affected?



- Up to what injection rates can we use bufferless routing?
  → Are there realistic scenarios in which NoC is operated at injection rates below the threshold?

- Can we achieve energy reduction?
  → If so, how much…?

- Can we reduce area, complexity, etc…?

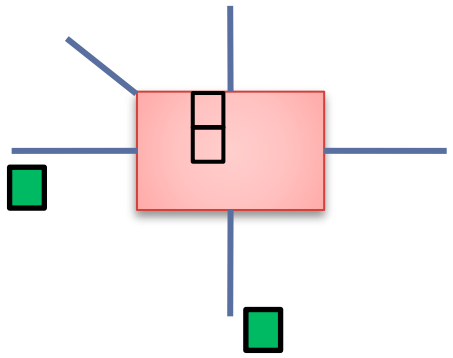
Answers in our paper!

Overview

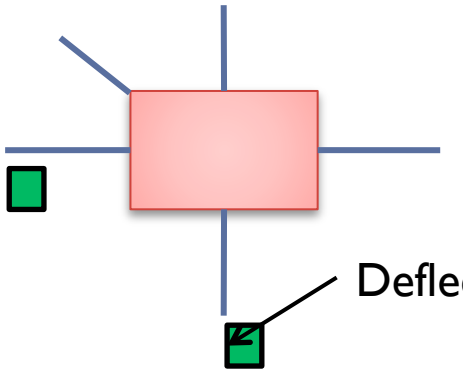
- Introduction and Background
- Bufferless Routing (BLESS)
  - FLIT-BLESS
  - WORM-BLESS
  - BLESS with buffers
- Advantages and Disadvantages
- Evaluations
- Conclusions

BLESS: Bufferless Routing

- Always forward *all* incoming flits to some output port
- If no productive direction is available, send to another direction
- → packet is deflected
- → Hot-potato routing [Baran'64, etc]

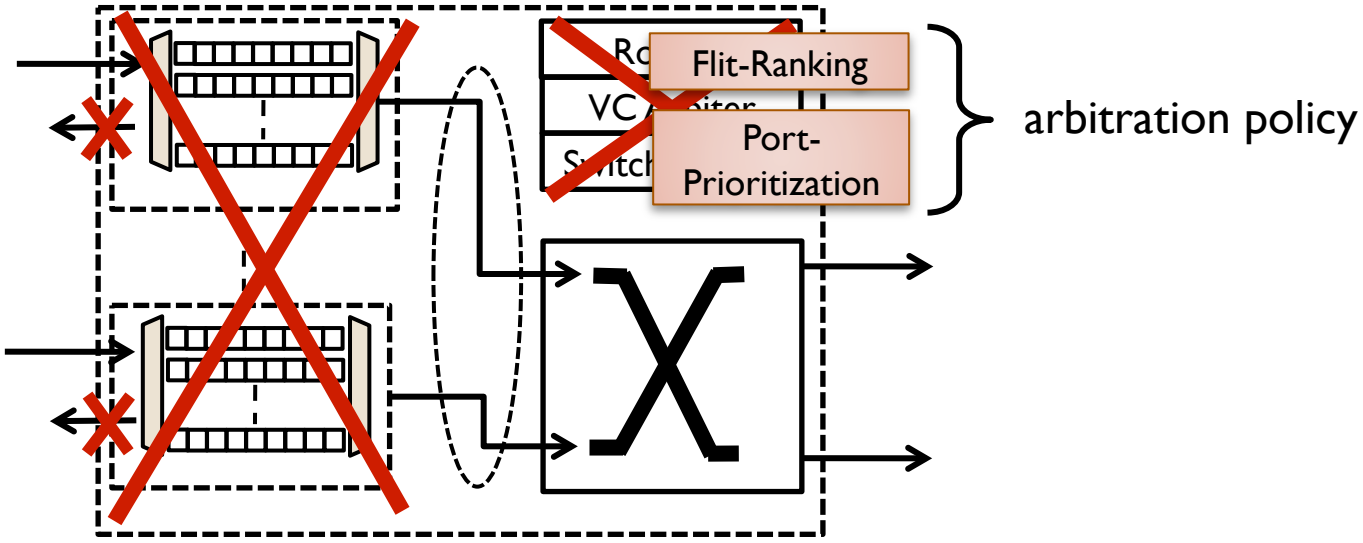


Buffered



BLESS

BLESS: Bufferless Routing



Flit-Ranking

1. Create a ranking over all incoming flits

Port-Prioritization

2. For a given flit in this ranking, find the best free output-port

Apply to each flit in order of ranking

FLIT-BLESS: Flit-Level Routing

- Each flit is routed independently.
- Oldest-first arbitration (other policies evaluated in paper)

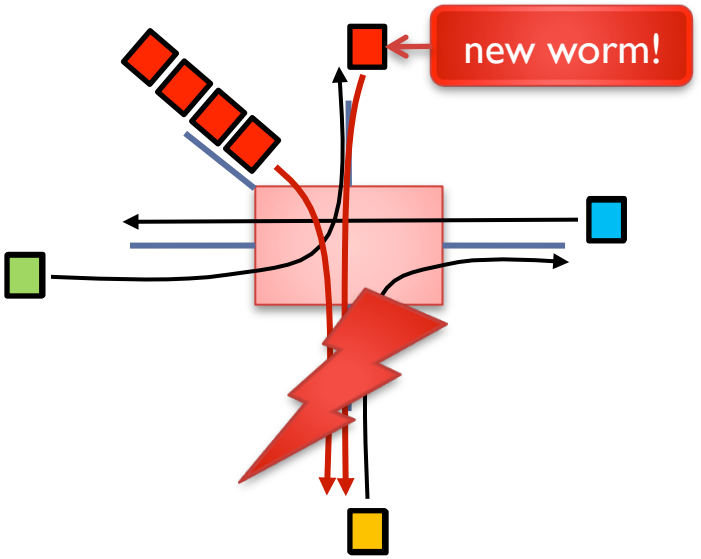
Flit-Ranking	1. Oldest-first ranking
Port-Prioritization	2. Assign flit to productive port, if possible. Otherwise, assign to non-productive port.

- Network Topology:
  → Can be applied to most topologies (Mesh, Torus, Hypercube, Trees, …)
  1) #output ports ¸ #input ports at every router
  2) every router is reachable from every other router
- Flow Control & Injection Policy:
  → Completely local, inject whenever input port is free
- Absence of Deadlocks: every flit is always moving
- Absence of Livelocks: with oldest-first ranking

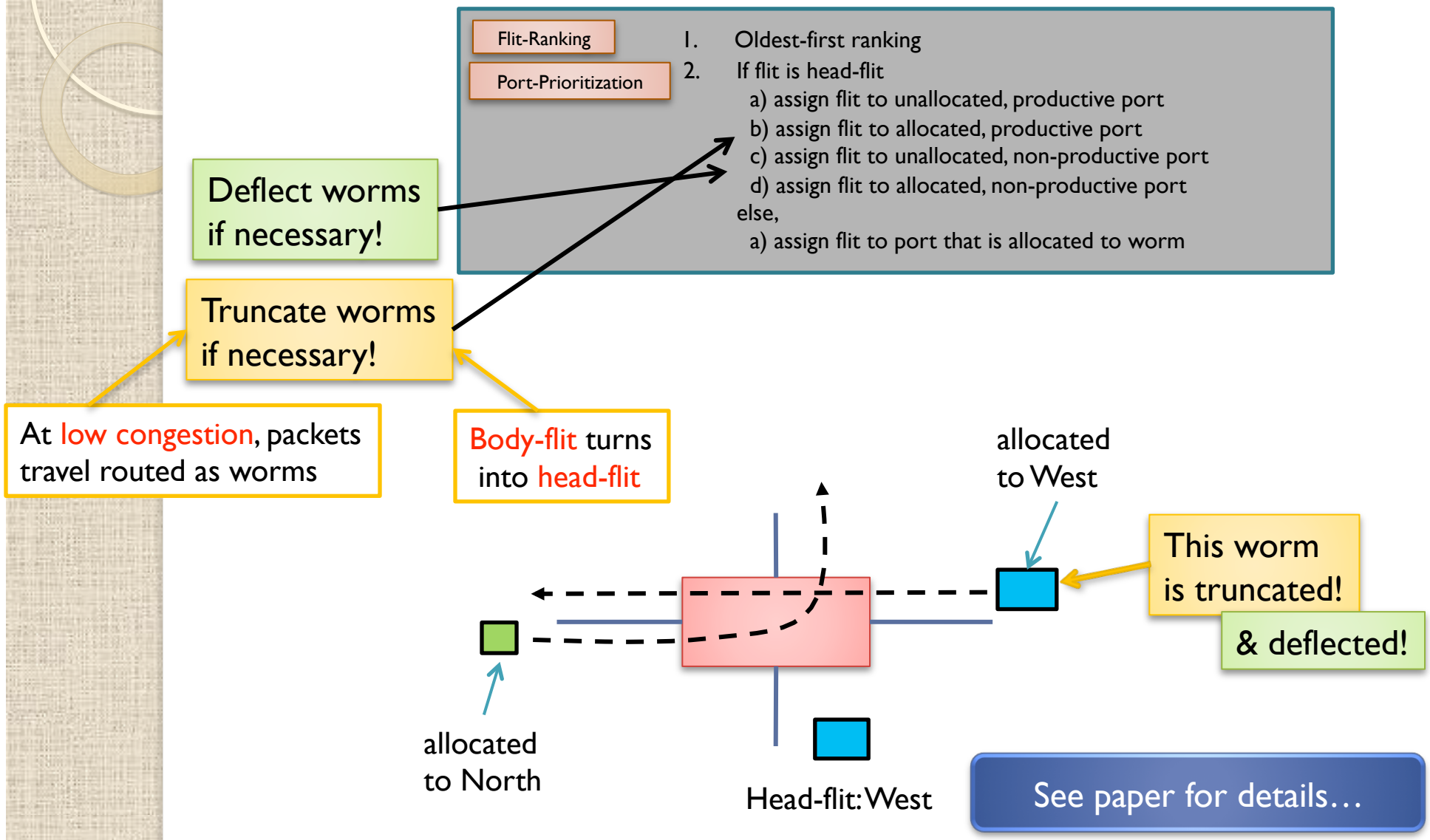
WORM-BLESS: Wormhole Routing

- Potential downsides of FLIT-BLESS
  - Not-energy optimal (each flits needs header information)
  - Increase in latency (different flits take different path)
  - Increase in receive buffer size

- BLESS with wormhole routing…? [Dally, Seitz'86]
- Problems:
  - Injection Problem
    (not known when it is safe to inject)
  - Livelock Problem
    (packets can be deflected forever)



WORM-BLESS: Wormhole Routing



BLESS with Buffers

- BLESS without buffers is extreme end of a continuum
- BLESS can be integrated with buffers
  - FLIT-BLESS with Buffers
  - WORM-BLESS with Buffers
- Whenever a buffer is full, it's first flit becomes must-schedule
- must-schedule flits must be deflected if necessary

See paper for details…

Overview

- Introduction and Background
- Bufferless Routing (BLESS)
  - FLIT-BLESS
  - WORM-BLESS
  - BLESS with buffers
- Advantages and Disadvantages
- Evaluations
- Conclusions

BLESS: Advantages & Disadvantages

Advantages

- No buffers
- Purely local flow control
- Simplicity
  - no credit-flows
  - no virtual channels
  - simplified router design
- No deadlocks, livelocks
- Adaptivity
  - packets are deflected around congested areas!
- Router latency reduction
- Area savings

Disadvantages

- Increased latency
- Reduced bandwidth
- Increased buffering at receiver
- Header information at each flit

Impact on energy…?

Reduction of Router Latency

- BLESS gets rid of input buffers and virtual channels

BW: Buffer Write
RC: Route Computation
VA: Virtual Channel Allocation
SA: Switch Allocation
ST: Switch Traversal
LT: Link Traversal
LA LT: Link Traversal of Lookahead

[Dally, Towles'04]

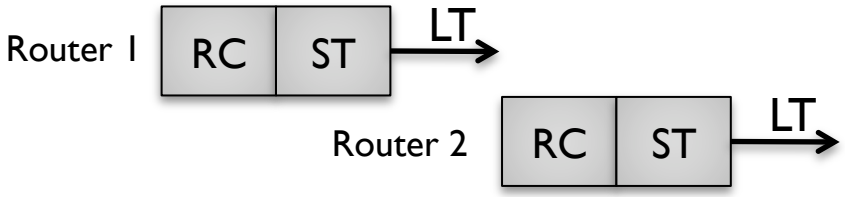
Baseline Router (speculative)

Router Latency = 3

Can be improved to 2.

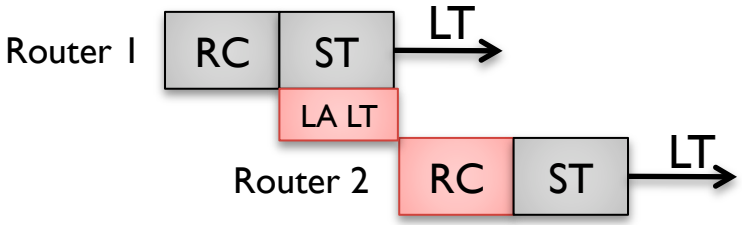
our evaluations

BLESS Router (standard)



Router Latency = 2

BLESS Router (optimized)



Router Latency = 1

BLESS: Advantages & Disadvantages

Advantages

- No buffers
- Purely local flow control
- Simplicity
  - no credit-flows
  - no virtual channels
  - simplified router design
- No deadlocks, livelocks
- Adaptivity
  - packets are deflected around congested areas!
- Router latency reduction
- Area savings

Disadvantages

- Increased latency
- Reduced bandwidth
- Increased buffering at receiver
- Header information at each flit

Extensive evaluations in the paper!

Impact on energy…?

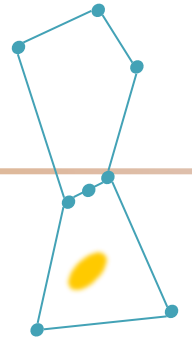
Evaluation Methodology

- 2D mesh network, router latency is 2 cycles
  - 4x4, 8 core, 8 L2 cache banks (each node is a core or an L2 bank)
  - 4x4, 16 core, 16 L2 cache banks (each no
  - 8x8, 16 core, 64 L2 cache banks (each no
  - 128-bit wide links, 4-flit data packets, 1-f
  - For baseline configuration: 4 VCs per phys
- Benchmarks
  - Multiprogrammed SPEC CPU2006 and Windows Desktop applications
  - Heterogeneous and homogenous application mixes
  - Synthetic traffic patterns: UR, Transpose, Tornado, Bit Complement
- x86 processor model based on Intel P
  - 2 GHz processor, 128-entry instruction w
  - 64Kbyte private L1 caches
  - Total 16Mbyte shared L2 caches; 16 MSHRs per bank
  - DRAM model based on Micron DDR2-800

Simulation is cycle-accurate
→ Models stalls in network and processors
→ Self-throttling behavior
→ Aggressive processor model

Most of our evaluations with perfect L2 caches
→ Puts maximal stress on NoC

Evaluation Methodology

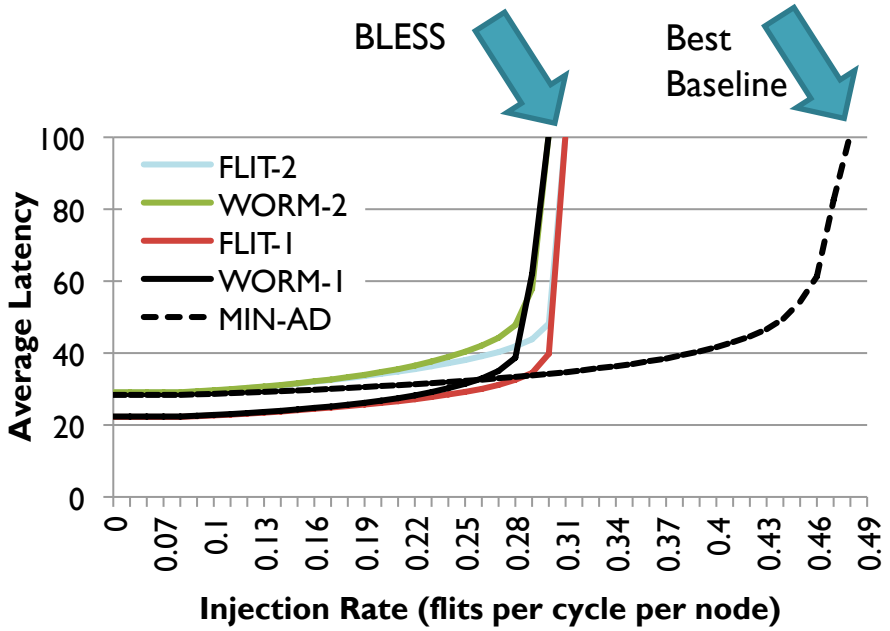


- Energy model provided by Orion simulator [MICRO'02]
  - 70nm technology, 2 GHz routers at 1.0 V_{dd}
- For BLESS, we model
  - Additional energy to transmit header information
  - Additional buffers needed on the receiver side
  - Additional logic to reorder flits of individual packets at receiver
- We partition network energy into buffer energy, router energy, and link energy, each having static and dynamic components.

- Comparisons against non-adaptive and aggressive adaptive buffered routing algorithms (DO, MIN-AD, ROMM)

Evaluation – Synthethic Traces

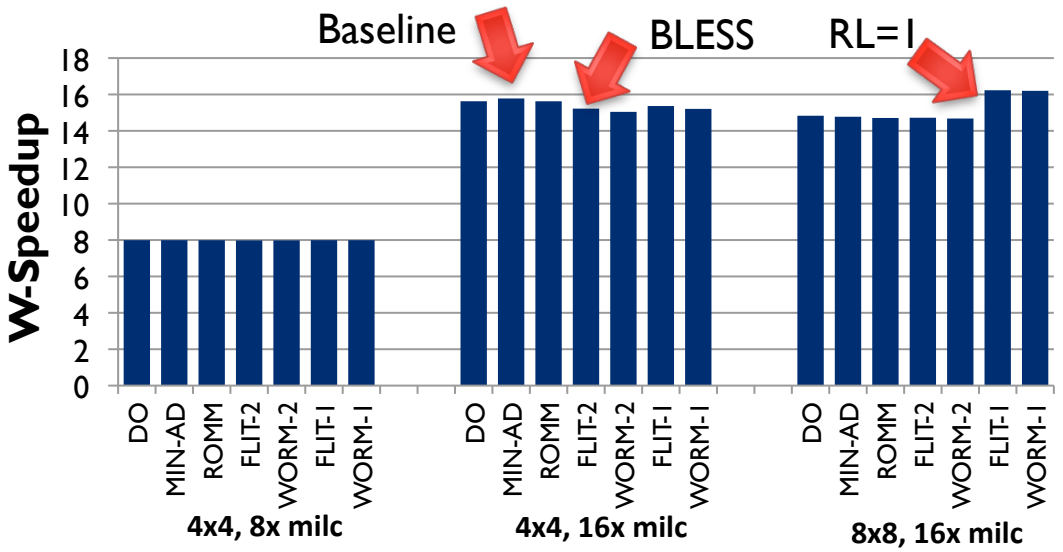
- First, the bad news ☺
- Uniform random injection
- BLESS has significantly lower saturation throughput compared to buffered baseline.

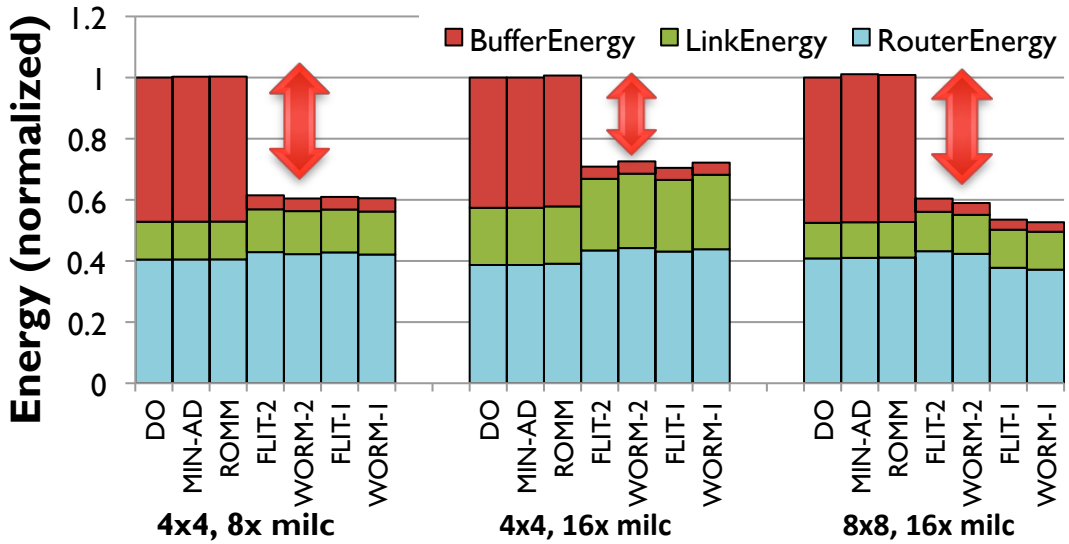


Is BLESS doomed…?

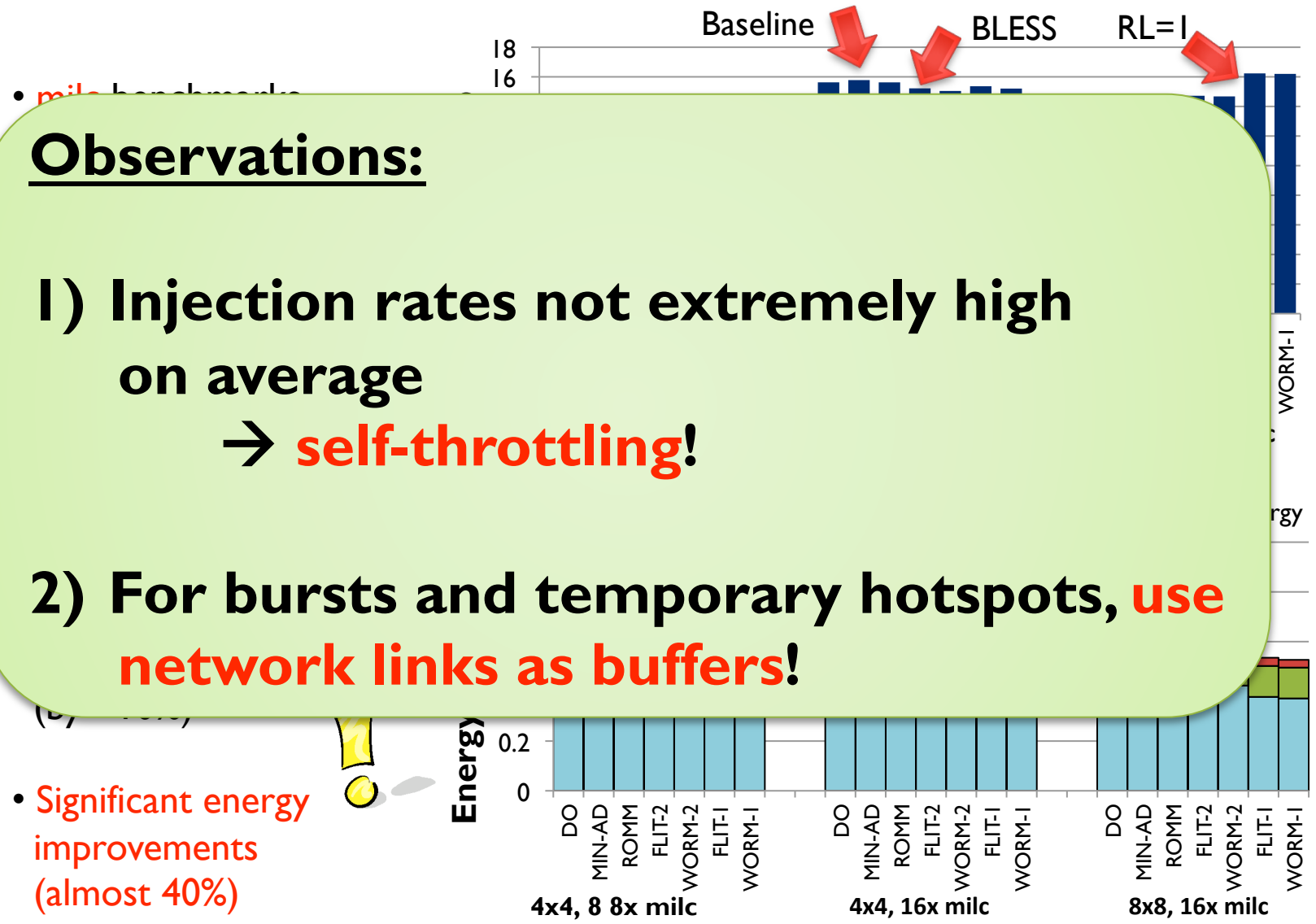
Evaluation – Homogenous Case Study

- milc benchmarks (moderately intensive)
- Perfect caches!
- Very little performance degradation with BLESS (less than 4% in dense network)
- With router latency 1, BLESS can even outperform baseline (by ~10%)
- Significant energy improvements (almost 40%)





Evaluation – Homogenous Case Study



Observations:

1) **Injection rates not extremely high on average**
   → **self-throttling!**

2) **For bursts and temporary hotspots, use network links as buffers!**

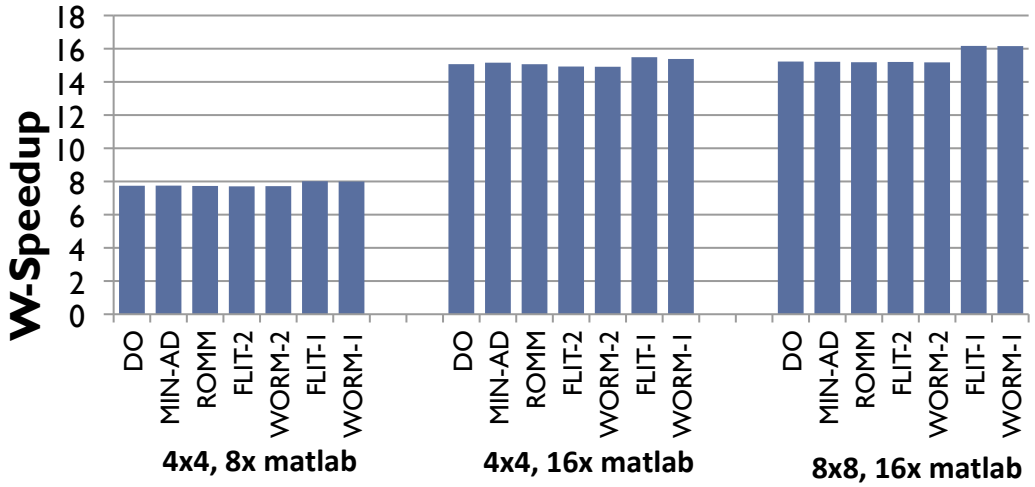
- Significant energy improvements (almost 40%)

Evaluation – Further Results

See paper for details…

- BLESS increases buffer requirement at receiver by at most 2x
  → overall, energy is still reduced

- Impact of memory latency
  → with real caches, very little slowdown! (at most 1.5%)



Evaluation – Further Results

See paper for details…

- BLESS increases buffer requirement at receiver by at most 2x
  → overall, energy is still reduced

- Impact of memory latency
  → with real caches, very little slowdown! (at most 1.5%)

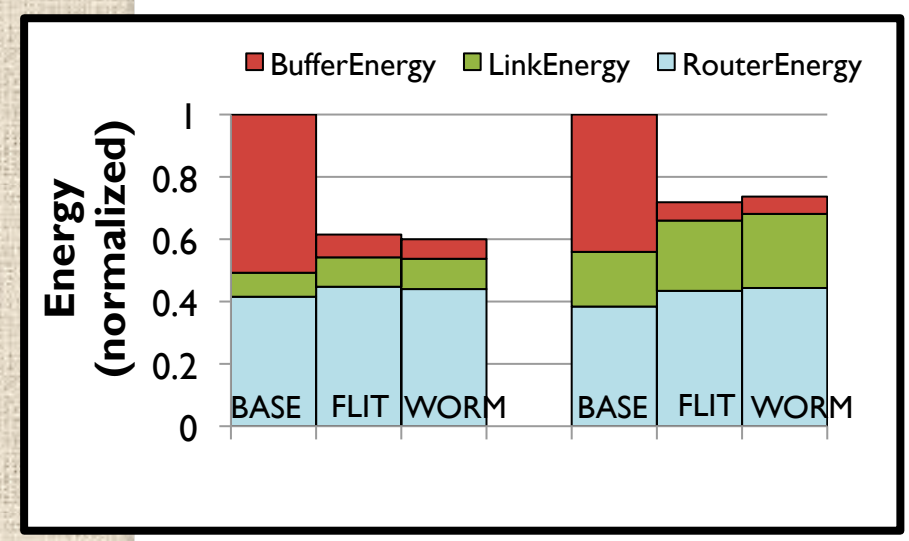
- Heterogeneous application mixes
  (we evaluate several mixes of intensive and non-intensive applications)
  → little performance degradation
  → significant energy savings in all cases
  → no significant increase in unfairness across different applications

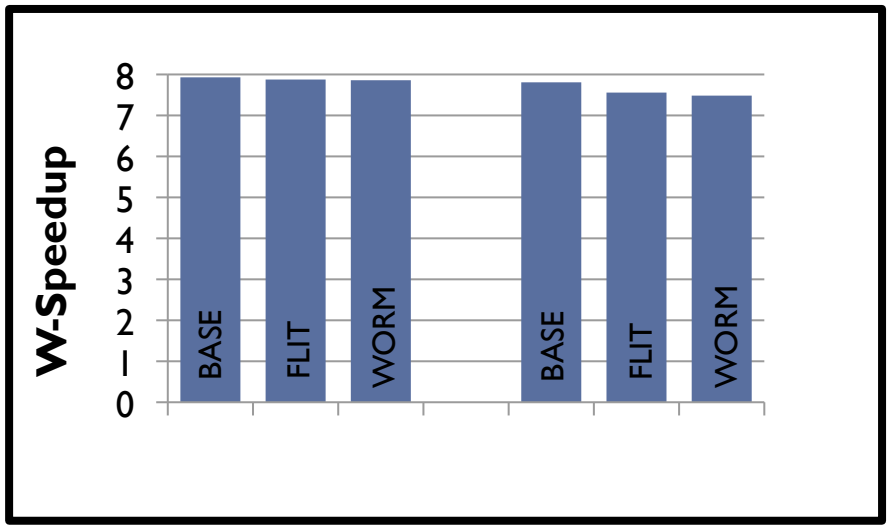
- Area savings: ~60% of network area can be saved!

Evaluation – Aggregate Results

- Aggregate results over all 29 applications

Sparse Network	Perfect L2		Realistic L2	
	Average	Worst-Case	Average	Worst-Case
Δ Network Energy	-39.4%	-28.1%	-46.4%	-41.0%
Δ System Performance	-0.5%	-3.2%	-0.15%	-0.55%





Evaluation – Aggregate Results

- Aggregate results over all 29 applications

Sparse Network	Perfect L2		Realistic L2	
	Average	Worst-Case	Average	Worst-Case
Δ Network Energy	-39.4%	-28.1%	-46.4%	-41.0%
Δ System Performance	-0.5%	-3.2%	-0.15%	-0.55%

Dense Network	Perfect L2		Realistic L2	
	Average	Worst-Case	Average	Worst-Case
Δ Network Energy	-32.8%	-14.0%	-42.5%	-33.7%
Δ System Performance	-3.6%	-17.1%	-0.7%	-1.5%

Conclusion

- For a very wide range of applications and network settings, buffers are not needed in NoC
  - Significant energy savings
    (32% even in dense networks and perfect caches)
  - Area-savings of 60%
  - Simplified router and network design (flow control, etc…)
  - Performance slowdown is minimal (can even increase!)

➢ A strong case for a rethinking of NoC design!

- We are currently working on future research.
  - Support for quality of service, different traffic classes, energy-management, etc…