A Case for Bufferless Routing in On-Chip Networks

Onur Mutlu
Carnegie Mellon University

Thomas Moscibroda
Microsoft Research

ISCA 2009
Austin, Texas, USA

Presented by Jonas Bokstaller
14 November 2018

Executive Summary

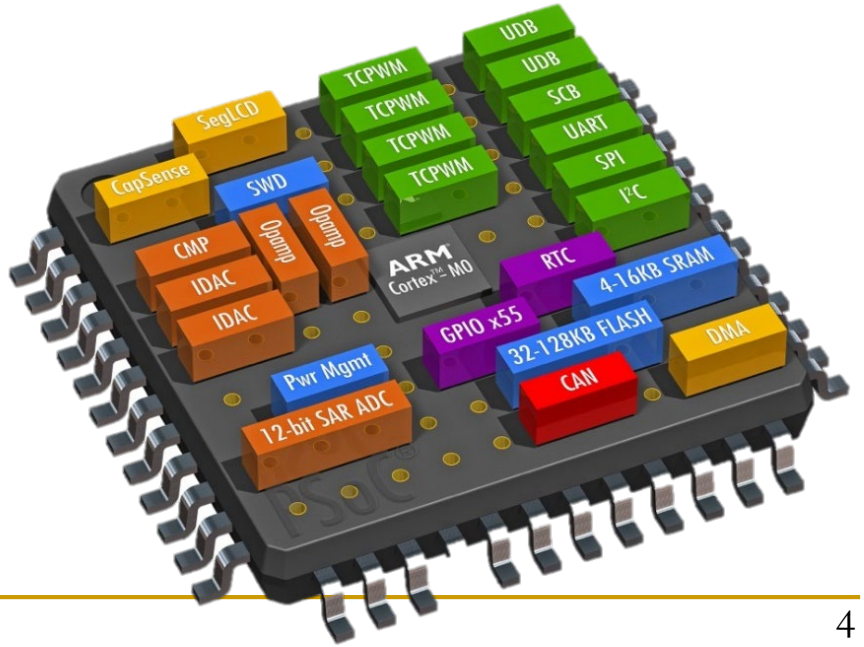
- **Problem:** The on chip networks in system on chips use the most energy/physical area for packet buffers which are used for routing the packets from different components on the chip.
- **Proposal:** We use three completely new routing algorithms "FLIT-Level-Routing", "Bless Wormhole Routing" and "Bless with Buffers" which aims to eliminate/reduce the need for buffers by deflecting packet inside the network.
- **Results:** Most of the time buffers are not needed on NoC
  - Average performance decrease by only 0.5%
  - Worst-case performance decrease by 3.2%
  - Average network energy consumption decrease by 39.4%
  - Area-savings of 60%

Outline

- **Background, Problem & Goal**
- Key Approach and Ideas
- Mechanisms (in some detail)
- Benefits and Limitations
- Key Results: Methodology and Evaluation
- Summary
- Strengths
- Weaknesses
- Takeaways
- Thoughts, Ideas and Discussion starters

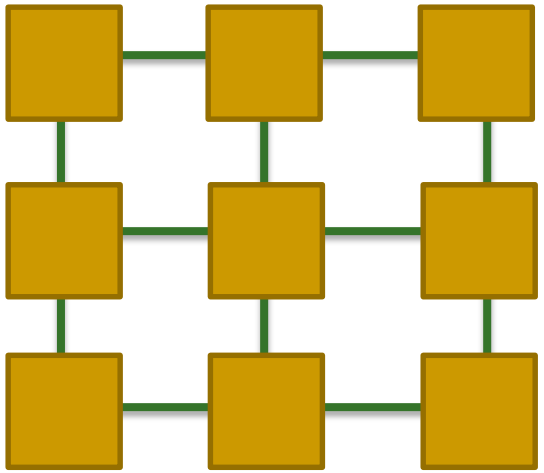
System on Chip

- System on a Chip (=SoC)
  - Every component on the same chip
  - Small footprint
  - Low power consumption
  - Commonly used in Smartphones, Internet of Things, etc…



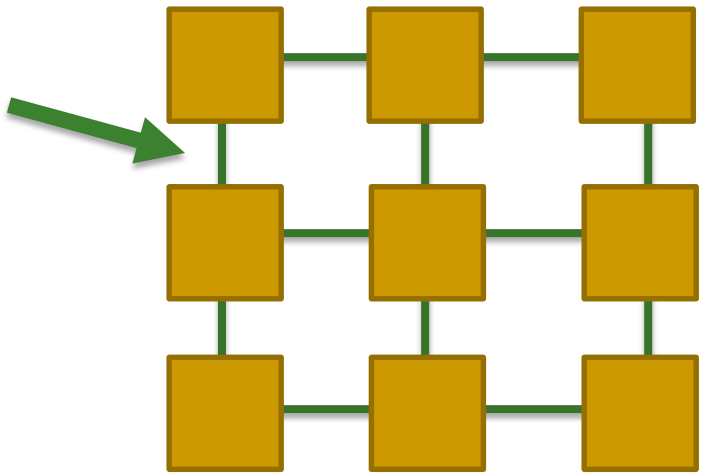
Network on Chip

- Network on Chip (=NoC)
  - Connect components on SoC
    - Cores, caches, etc…
  - Like in typical Computer Network



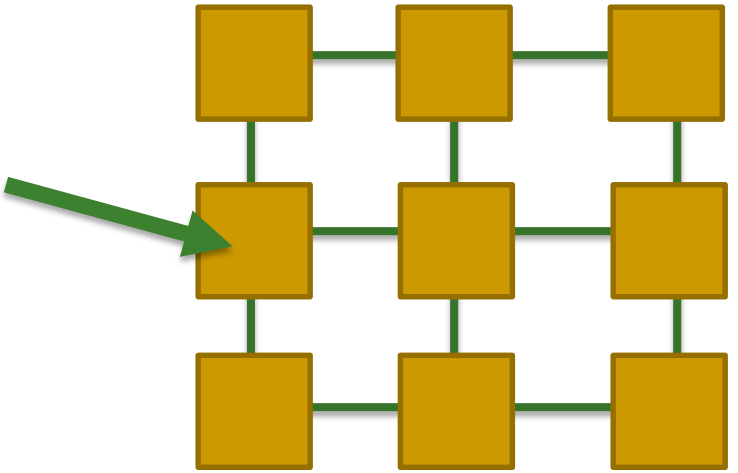
Network on Chip

- Network on Chip (=NoC)
  - Connect components on SoC
    - Cores, caches, etc…
  - Like in typical Computer Network
    - Physical link



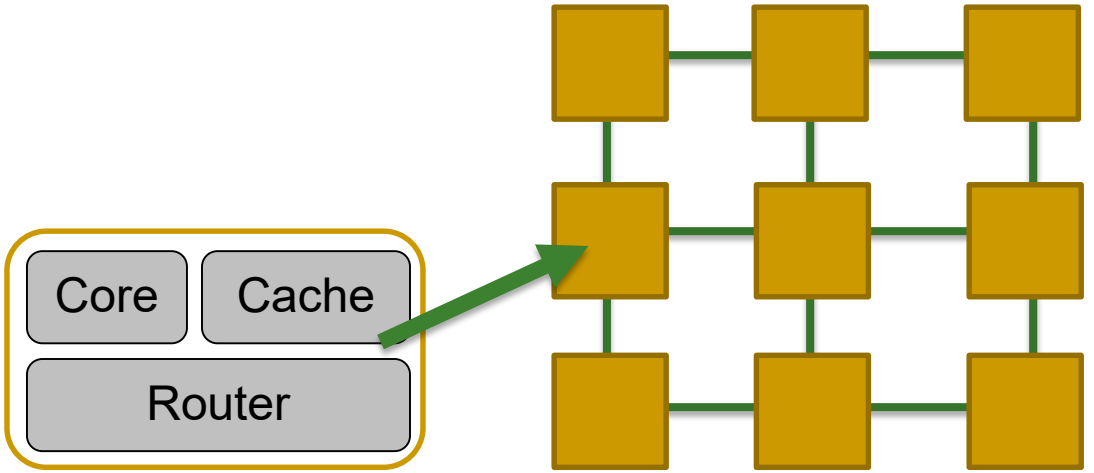
Network on Chip

- Network on Chip (=NoC)
  - Connect components on SoC
    - Cores, caches, etc…
  - Like in typical Computer Network
    - Physical link
    - Components



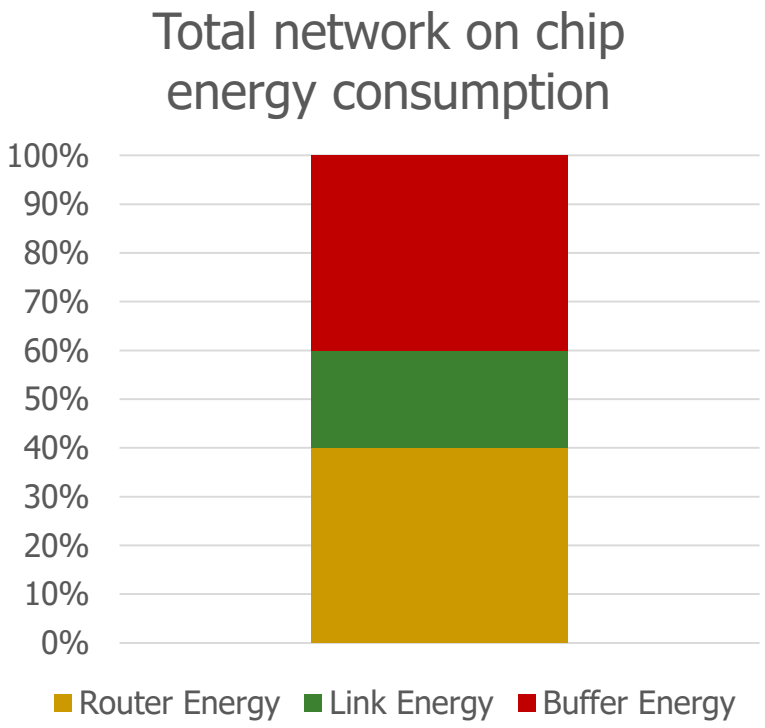
Network on Chip

- Network on Chip (=NoC)
  - Connect components on SoC
    - Cores, caches, etc…
  - Like in typical Computer Network
    - Physical link
    - Components
      - Built in router
      - E.g. CPU core



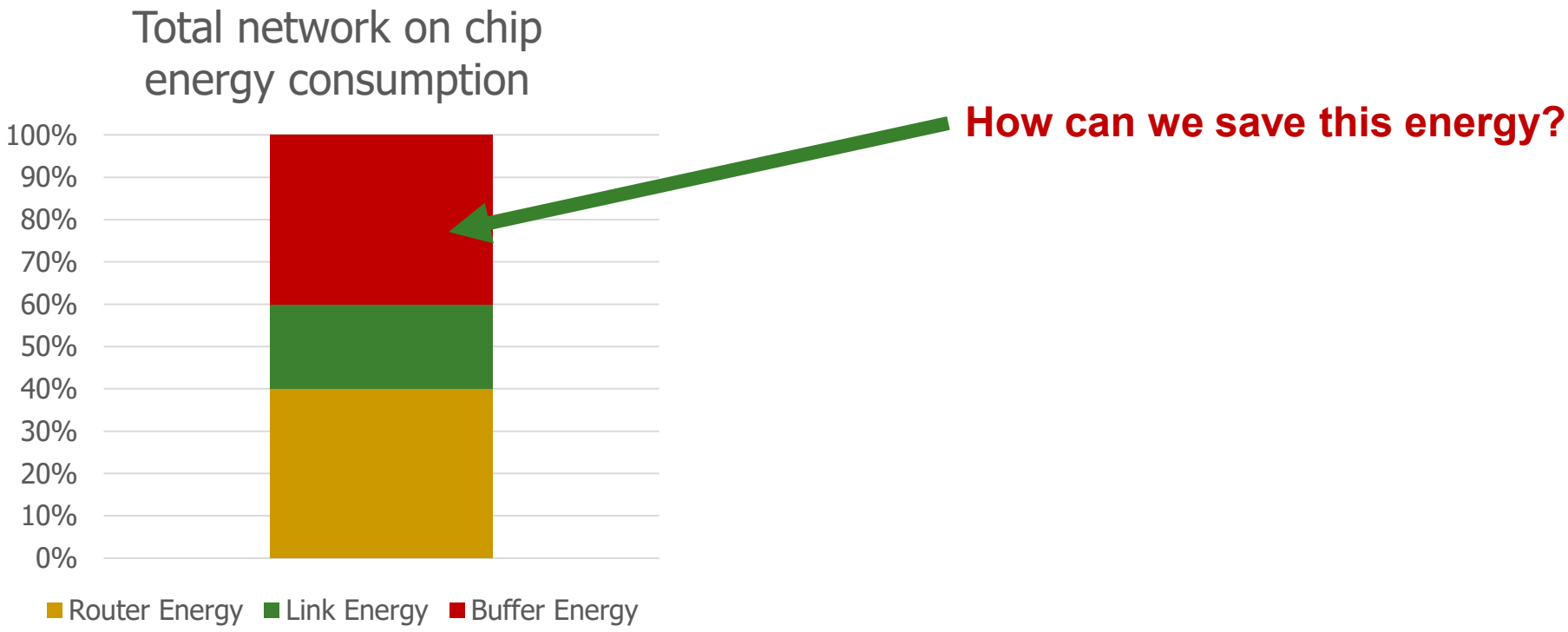
Problem with Buffers

- Energy consumption is too high
- Occupy chip area (≈75% of NoC)
- Increase design complexity
- Current approaches assume every router needs a buffer



Problem with Buffers

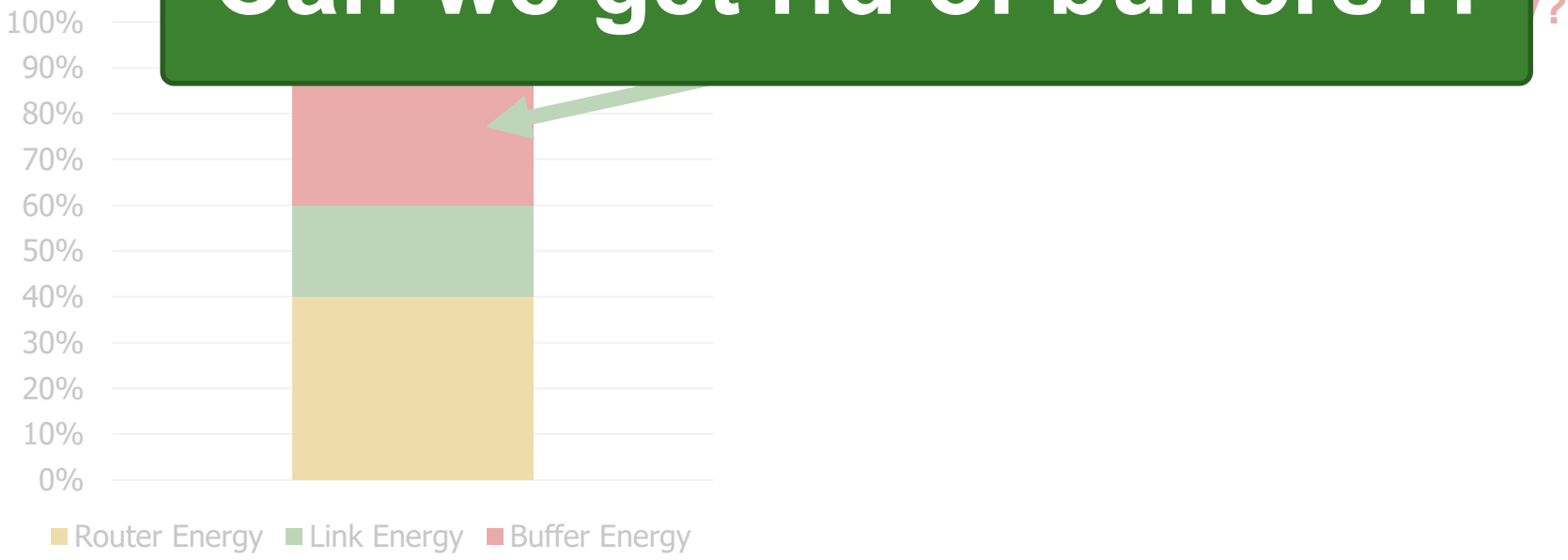
- Energy consumption is too high
- Occupy chip area (≈75% of NoC)
- Increase design complexity
- Current approaches assume every router needs a buffer



Problem with Buffers

- Energy consumption is too high
- Occupy chip area (≈75% of NoC)
- Increase design complexity
- Existing work assumes every router needs a buffer

Can we get rid of buffers?!

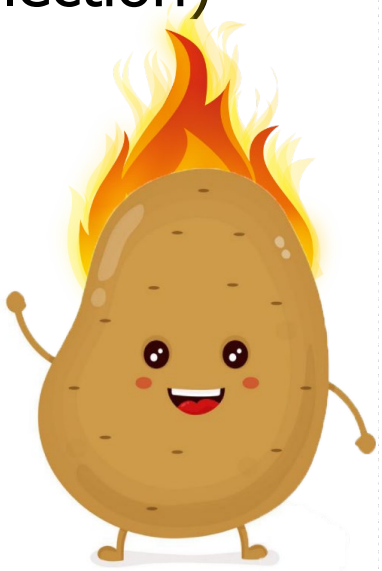


Outline

- Background, Problem & Goal
- **Key Approach and Ideas**
- Mechanisms (in some detail)
- Benefits and Limitations
- Key Results: Methodology and Evaluation
- Summary
- Strengths
- Weaknesses
- Takeaways
- Thoughts, Ideas and Discussion starters

Bufferless Routing

- "Hot potato"-routing
  - Too hot to keep (buffer)
- Always route a packet
- Links act as buffers
- Don't care about the lowest distance → keep packet moving
- Misroute if right output-port isn't available (=deflection)



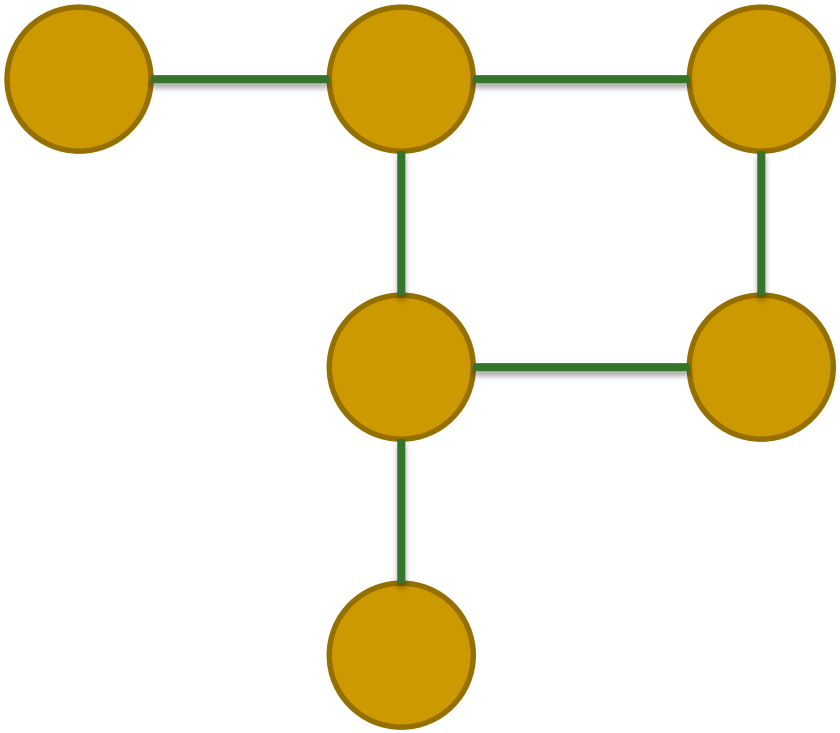
Main Advantages/Disadvantages

Small traffic volumes

Number of collisions is low

Rerouting is low

- Increase of bandwidth
- Decrease of latency
- Decrease of buffer-energy



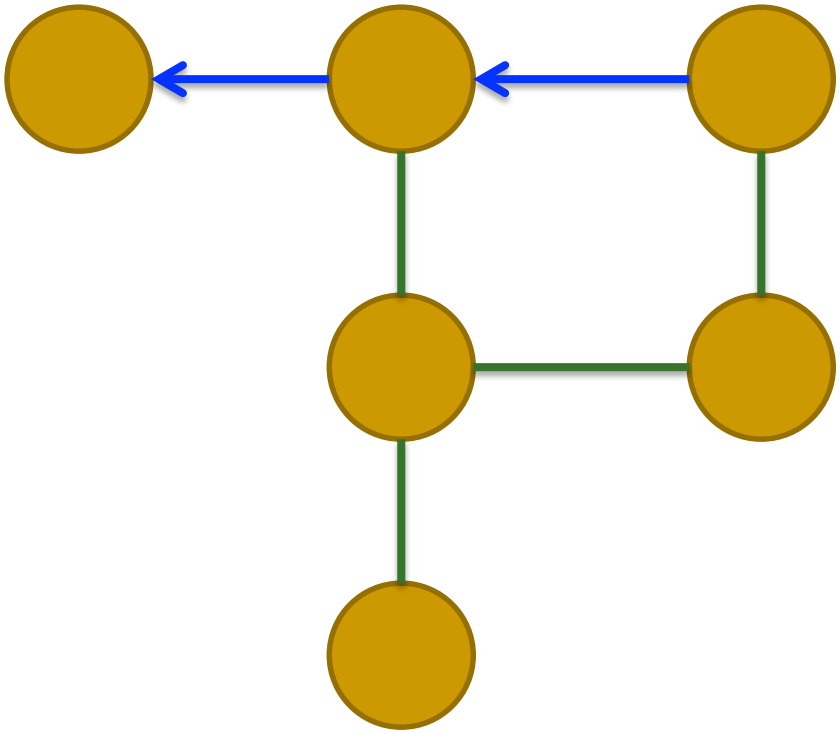
Main Advantages/Disadvantages

Small traffic volumes

Number of collisions is low

Rerouting is low

- Increase of bandwidth
- Decrease of latency
- Decrease of buffer-energy



Main Advantages/Disadvantages

Large traffic volumes

↓

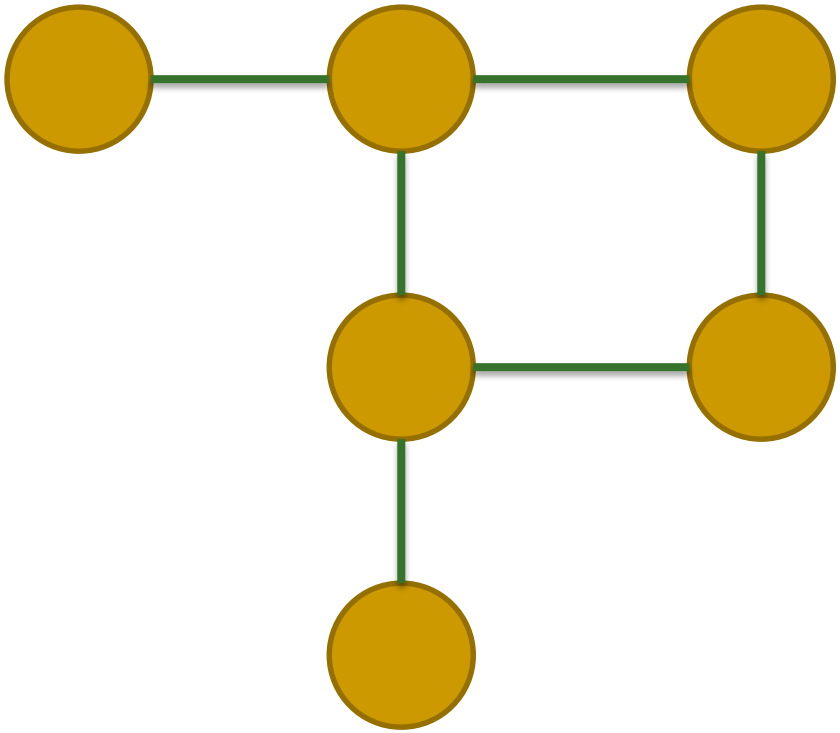
Collisions occur

↓

Packets get rerouted

↓

- Reduction of bandwidth
- Increase of latency
- Increase of link/router-energy



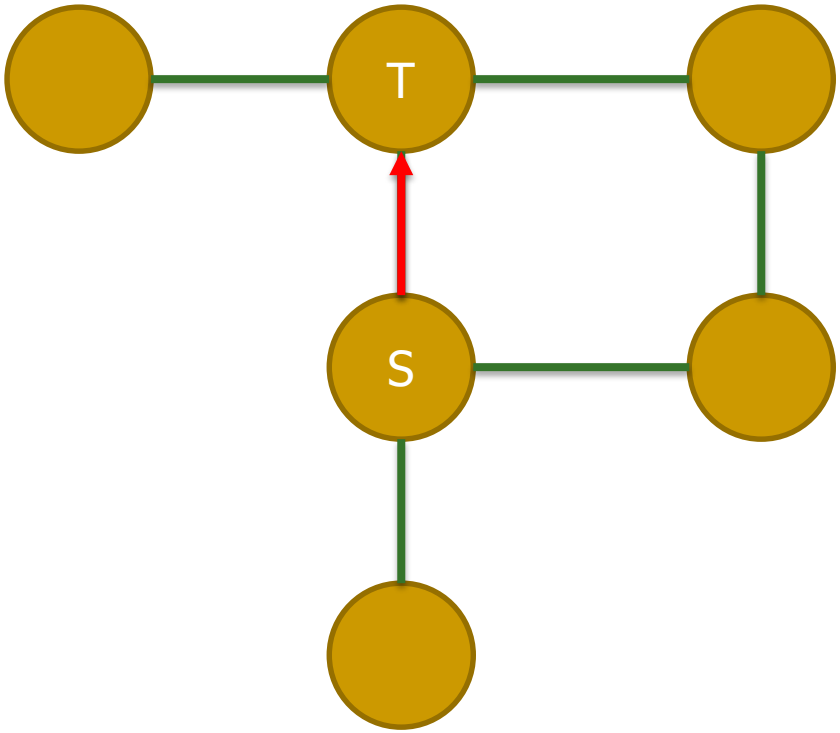
Main Advantages/Disadvantages

Large traffic volumes

Collisions occur

Packets get rerouted

- Reduction of bandwidth
- Increase of latency
- Increase of link/router-energy



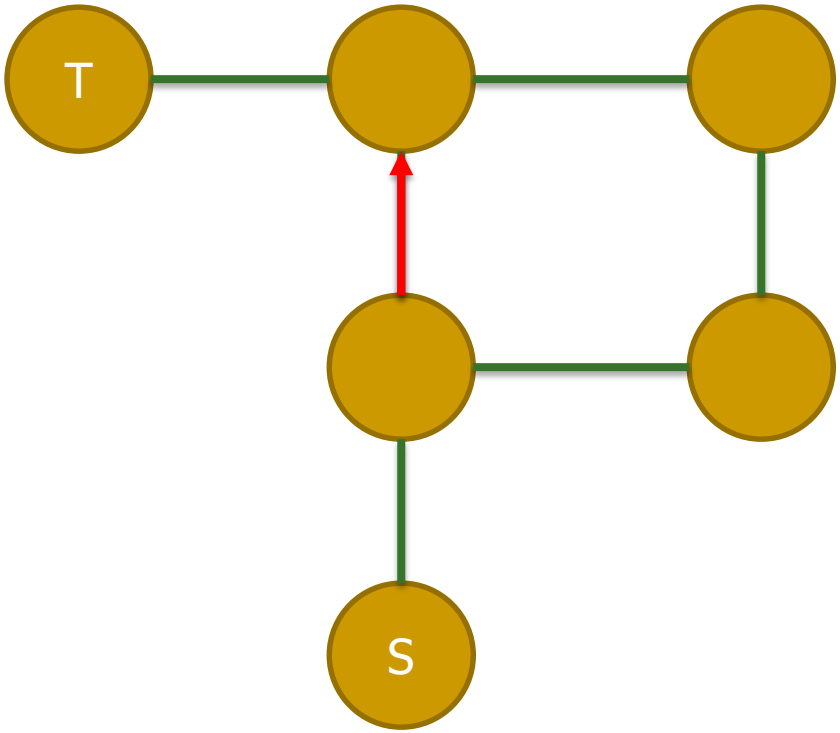
Main Advantages/Disadvantages

Large traffic volumes

Collisions occur

Packets get rerouted

- Reduction of bandwidth
- Increase of latency
- Increase of link/router-energy



Main Advantages/Disadvantages

Large traffic volumes

↓

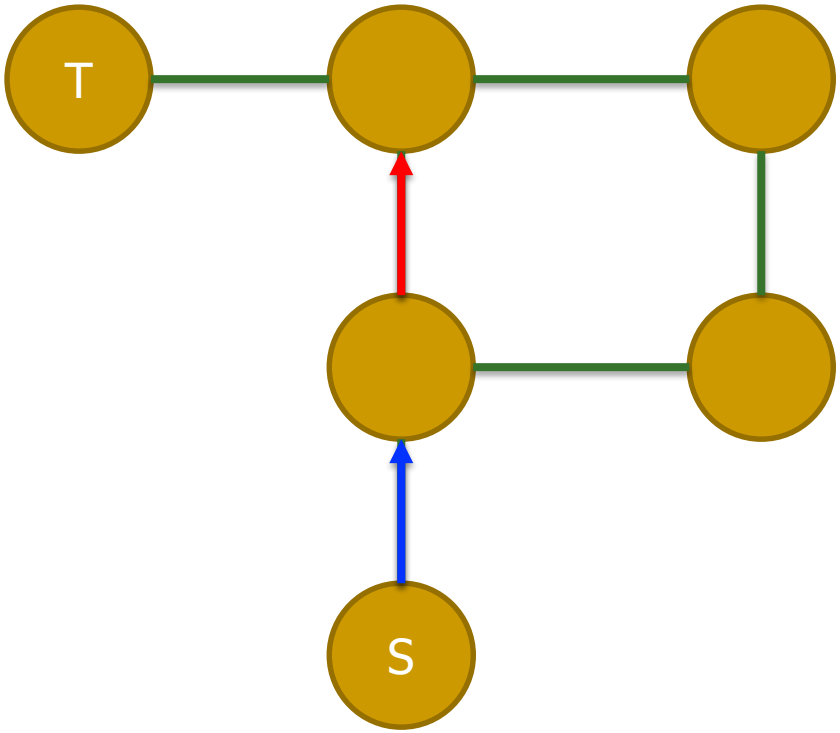
Collisions occur

↓

Packets get rerouted

↓

- Reduction of bandwidth
- Increase of latency
- Increase of link/router-energy



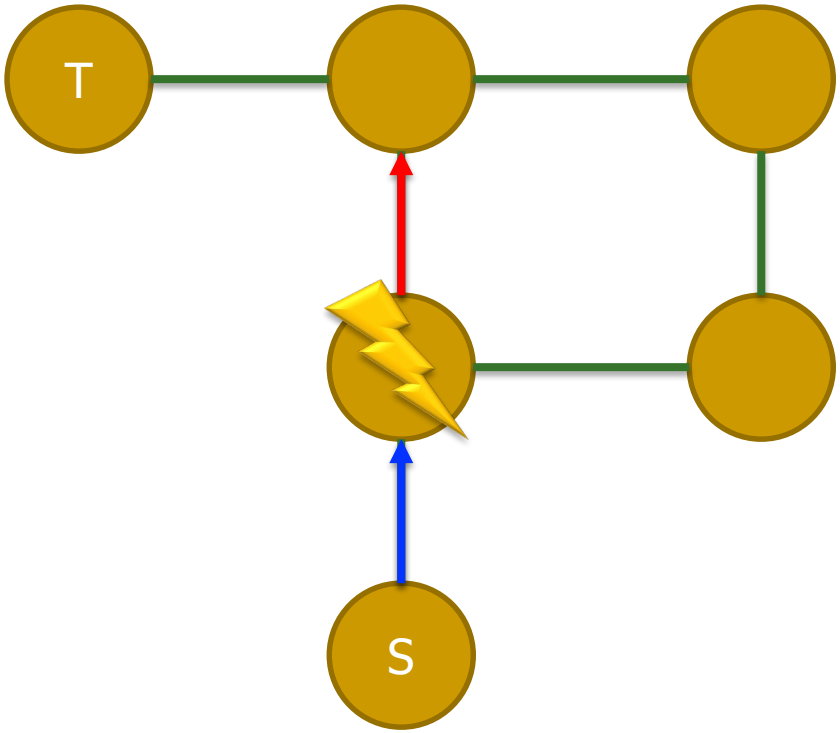
Main Advantages/Disadvantages

Large traffic volumes

Collisions occur

Packets get rerouted

- Reduction of bandwidth
- Increase of latency
- Increase of link/router-energy



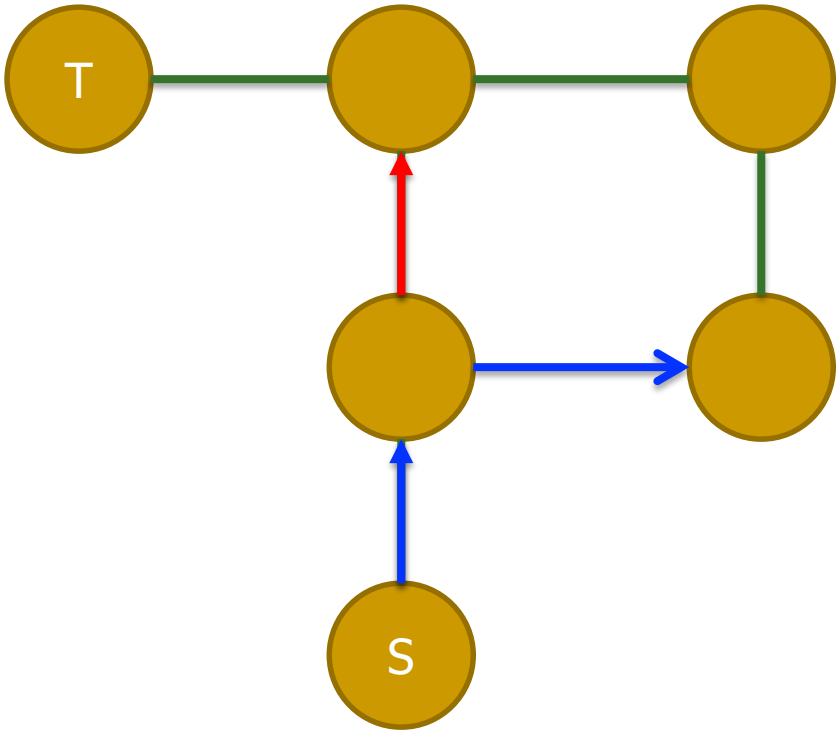
Main Advantages/Disadvantages

Large traffic volumes

Collisions occur

Packets get rerouted

- Reduction of bandwidth
- Increase of latency
- Increase of link/router-energy



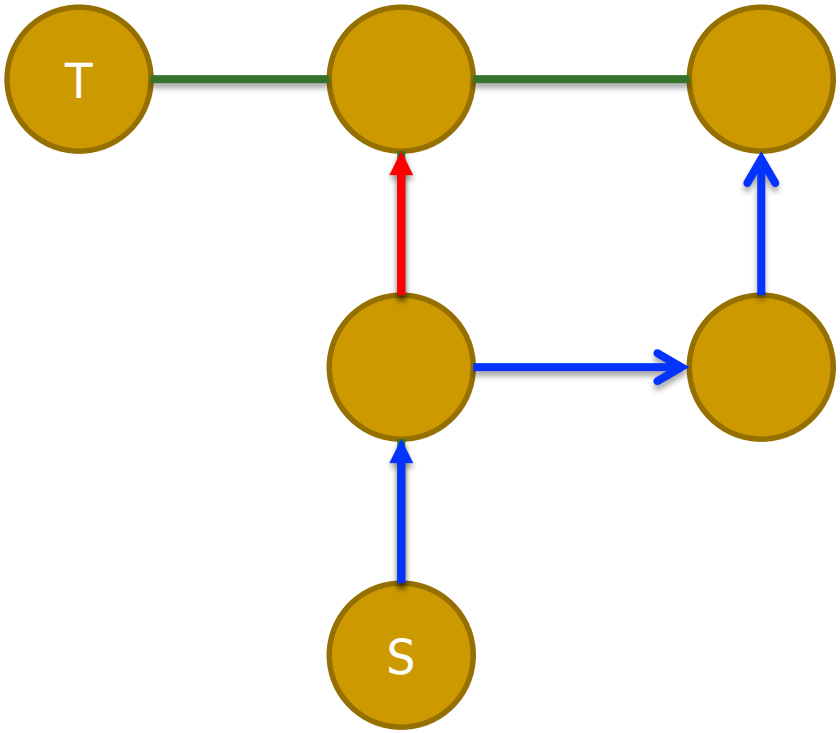
Main Advantages/Disadvantages

Large traffic volumes

Collisions occur

Packets get rerouted

- Reduction of bandwidth
- Increase of latency
- Increase of link/router-energy



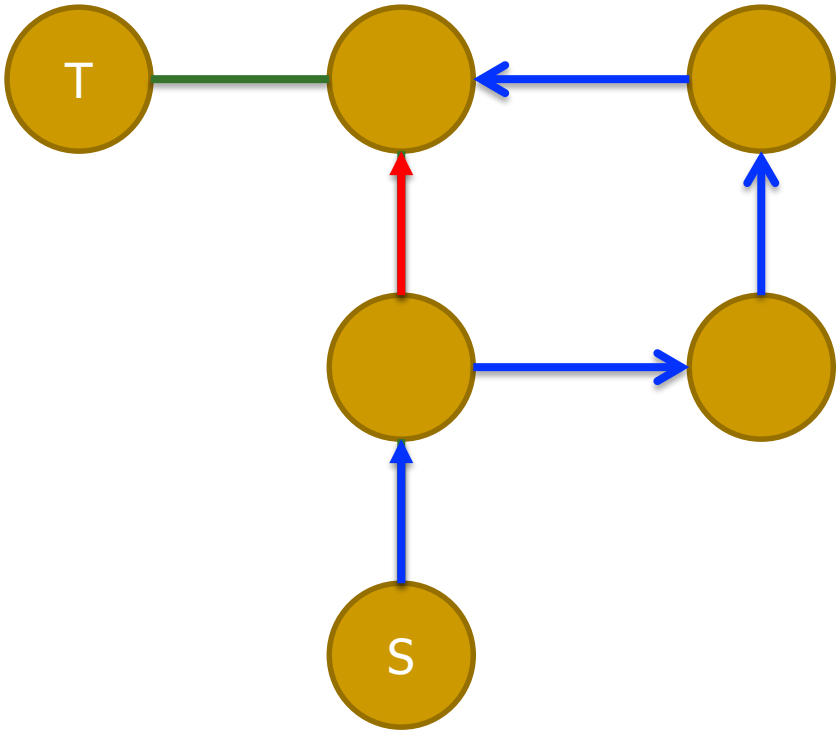
Main Advantages/Disadvantages

Large traffic volumes

Collisions occur

Packets get rerouted

- Reduction of bandwidth
- Increase of latency
- Increase of link/router-energy



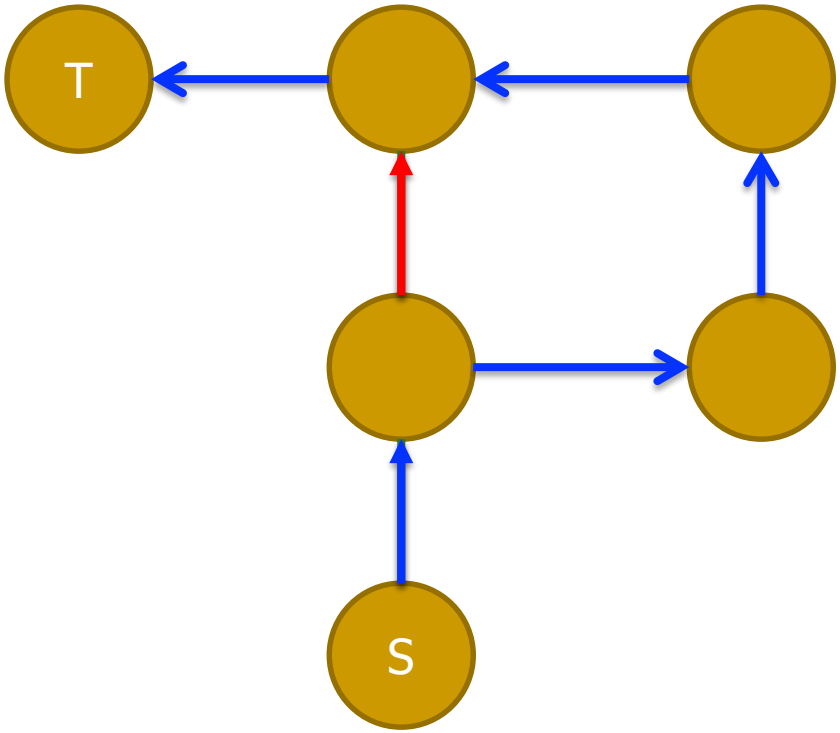
Main Advantages/Disadvantages

Large traffic volumes

Collisions occur

Packets get rerouted

- Reduction of bandwidth
- Increase of latency
- Increase of link/router-energy

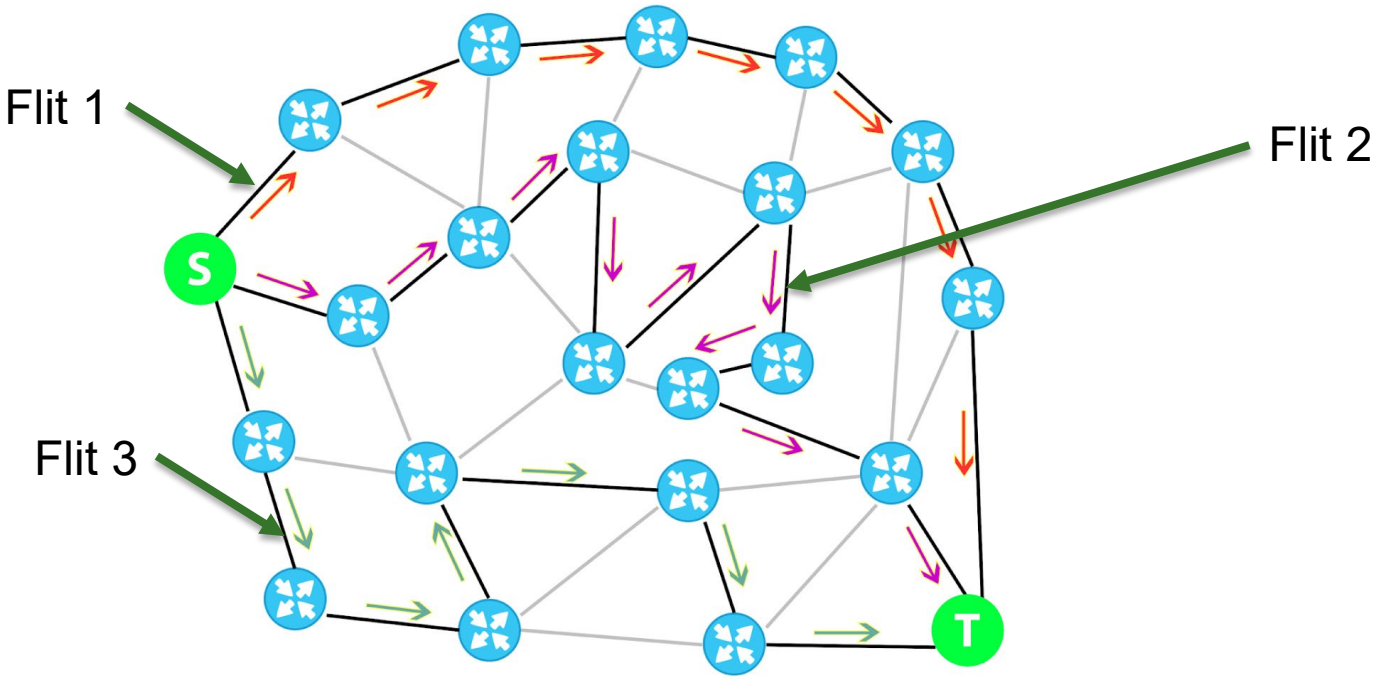


Outline

- Background, Problem & Goal
- Key Approach and Ideas
- **Mechanisms (in some detail)**
- Benefits and Limitations
- Key Results: Methodology and Evaluation
- Summary
- Strengths
- Weaknesses
- Takeaways
- Thoughts, Ideas and Discussion starters

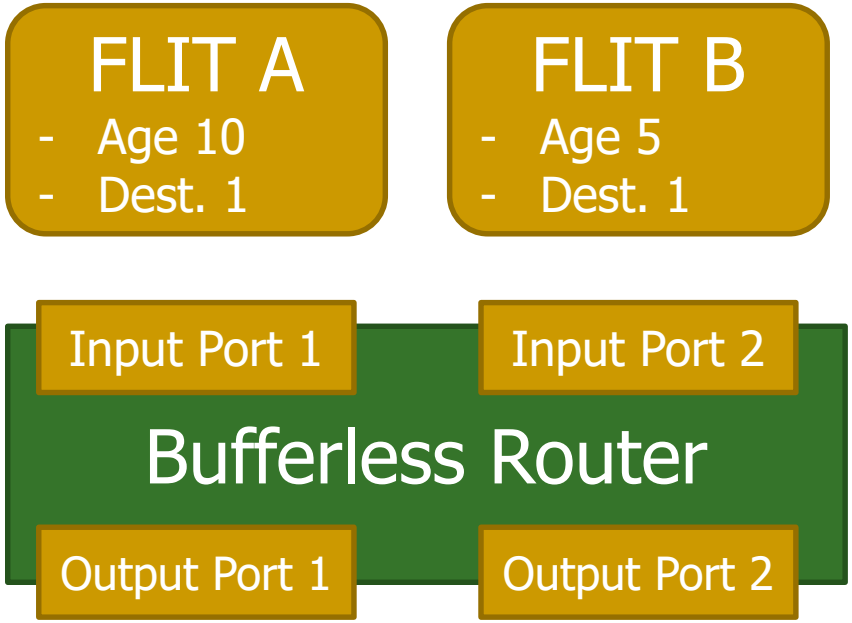
Basic Algorithm: Flit-Level Routing (I)

- Flit
  - **Fl**ow control un**it**s
  - Large network packets broken into smaller pieces
- Each Flit can take a different path but is always forwarded



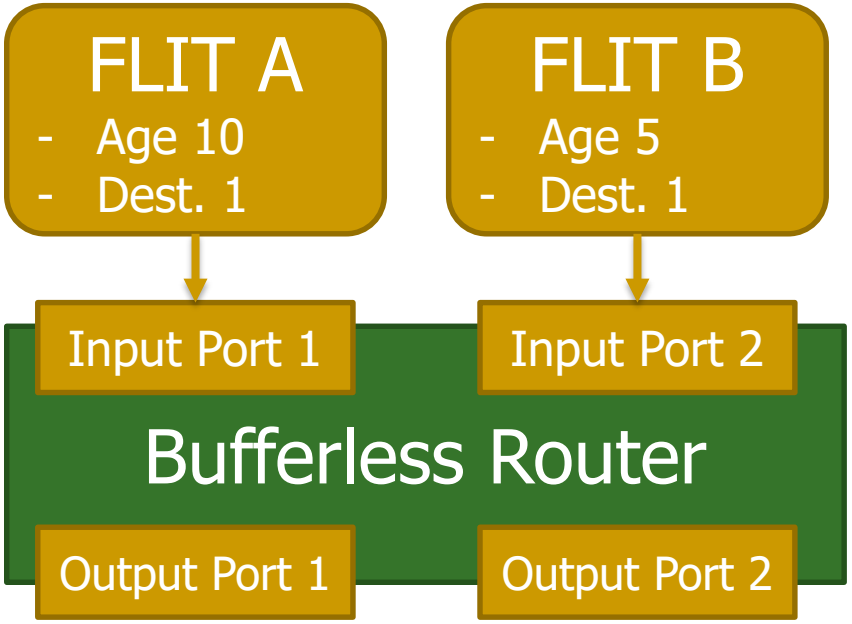
Basic Algorithm: Flit-Level Routing (II)

- If no productive output-port is available, send/deflect flit to a non-productive output-port
- Input ports ≤ output ports
- Routers form a connected graph



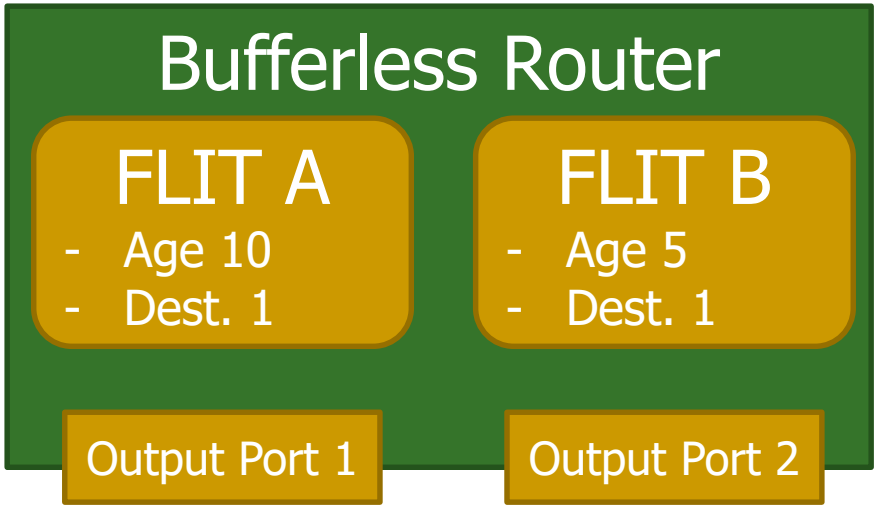
Basic Algorithm: Flit-Level Routing (II)

- If no productive output-port is available, send/deflect flit to a non-productive output-port
- Input ports ≤ output ports
- Routers form a connected graph



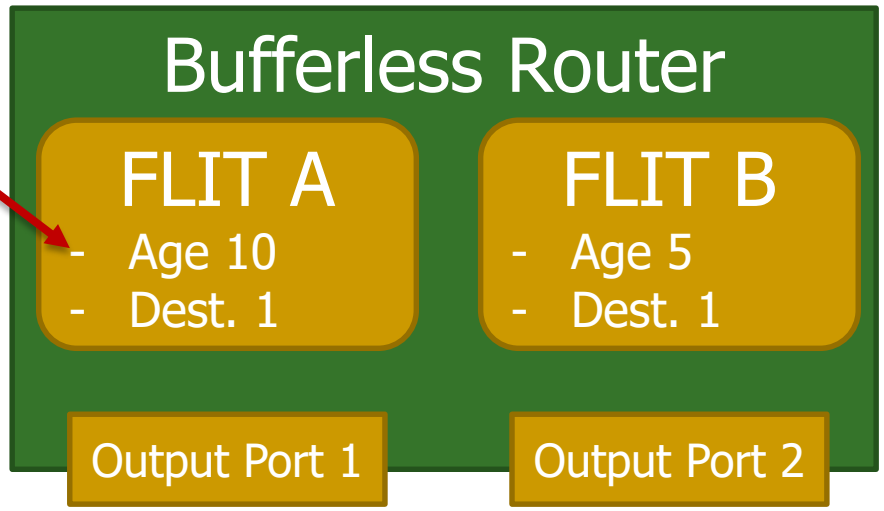
Basic Algorithm: Flit-Level Routing (II)

- If no productive output-port is available, send/deflect flit to a non-productive output-port
- Input ports ≤ output ports
- Routers form a connected graph



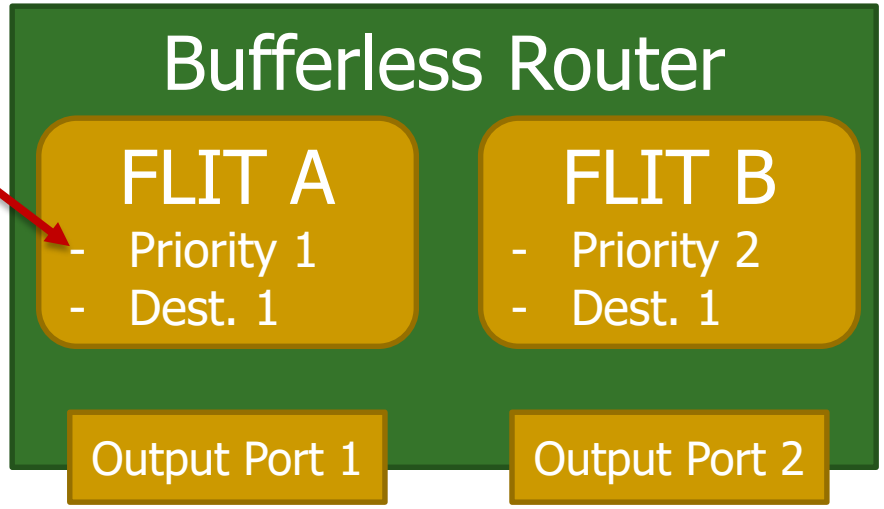
Basic Algorithm: Flit-Level Routing (II)

- If no productive output-port is available, send/deflect flit to a non-productive output-port
- Input ports ≤ output ports
- Routers form a connected graph
- Flit-Ranking
  - Oldest first
  - Avoids Livelocks



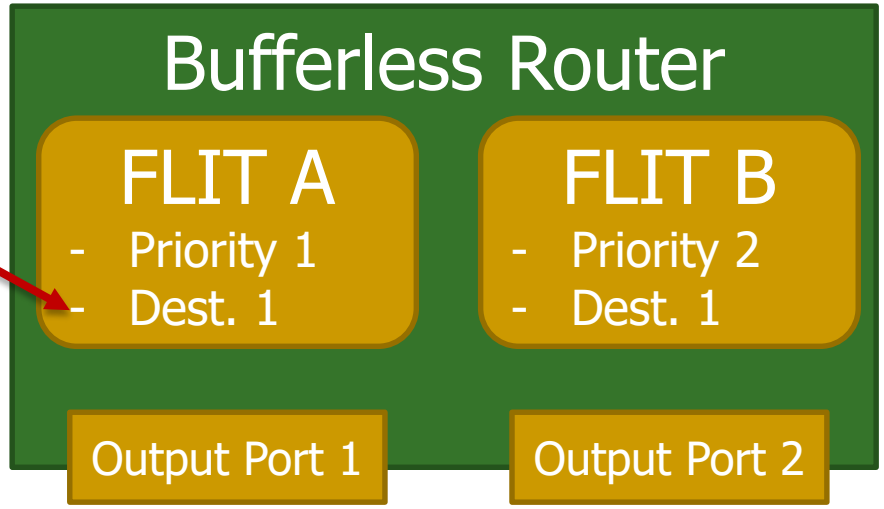
Basic Algorithm: Flit-Level Routing (II)

- If no productive output-port is available, send/deflect flit to a non-productive output-port
- Input ports ≤ output ports
- Routers form a connected graph
- Flit-Ranking
  - Oldest first
  - Avoids Livelocks



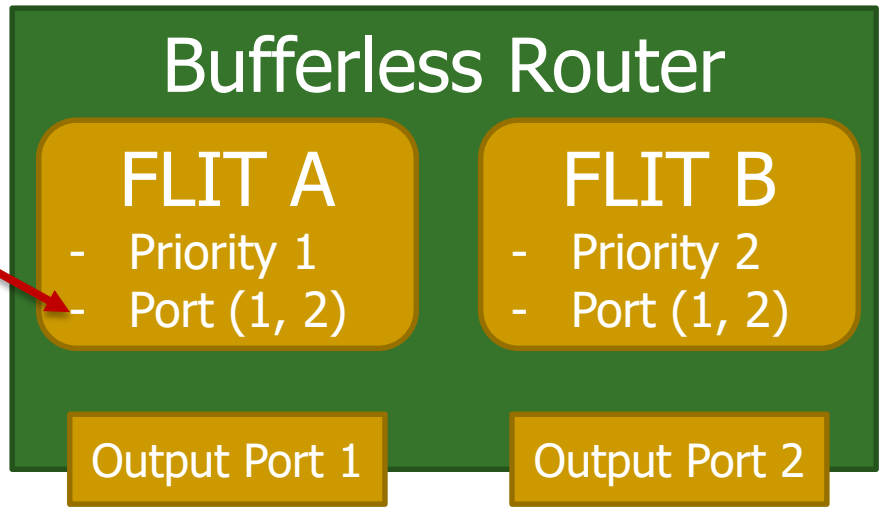
Basic Algorithm: Flit-Level Routing (II)

- If no productive output-port is available, send/deflect flit to a non-productive output-port
- Input ports ≤ output ports
- Routers form a connected graph
- Flit-Ranking
  - Oldest first
  - Avoids Livelocks
- Port-Prioritization
  - Different for every flit
  - Find the best output-ports



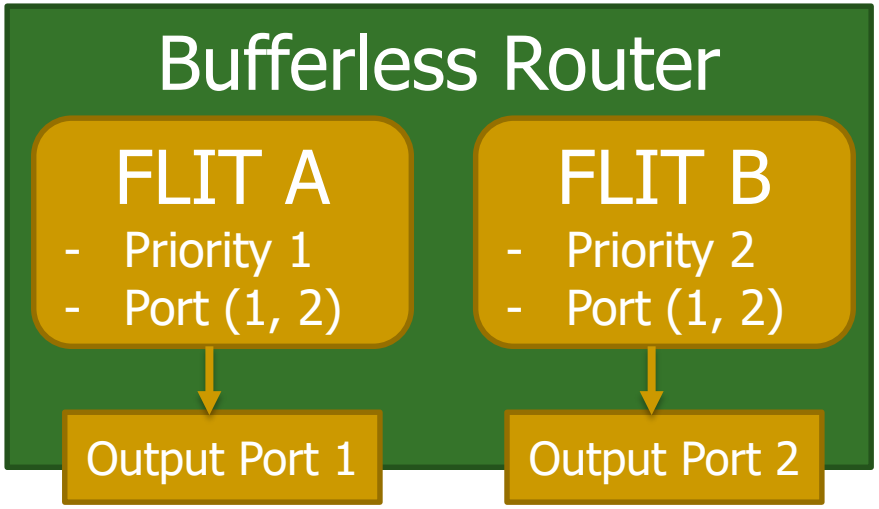
Basic Algorithm: Flit-Level Routing (II)

- If no productive output-port is available, send/deflect flit to a non-productive output-port
- Input ports ≤ output ports
- Routers form a connected graph
- Flit-Ranking
  - Oldest first
  - Avoids Livelocks
- Port-Prioritization
  - Different for every flit
  - Find the best output-ports



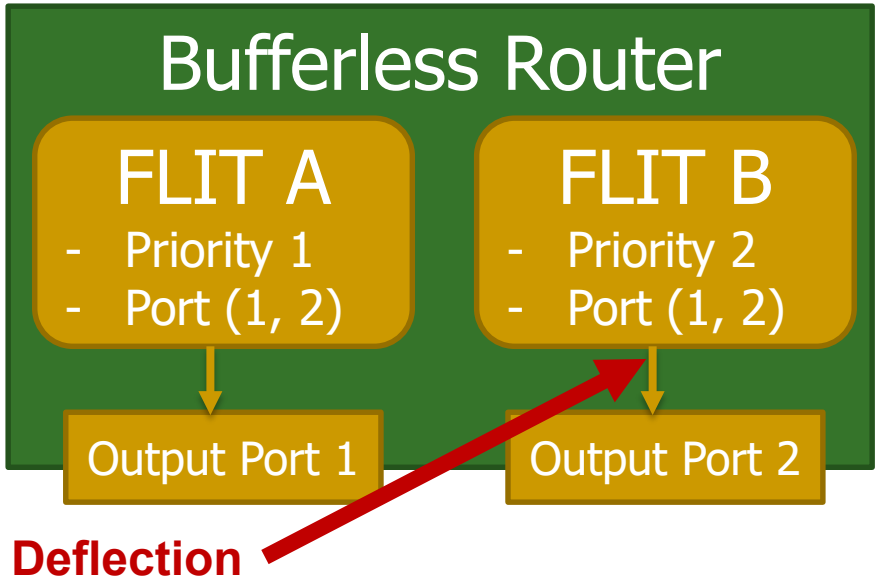
Basic Algorithm: Flit-Level Routing (II)

- If no productive output-port is available, send/deflect flit to a non-productive output-port
- Input ports ≤ output ports
- Routers form a connected graph
- Flit-Ranking
  - Oldest first
  - Avoids Livelocks
- Port-Prioritization
  - Different for every flit
  - Find the best output-ports



Basic Algorithm: Flit-Level Routing (II)

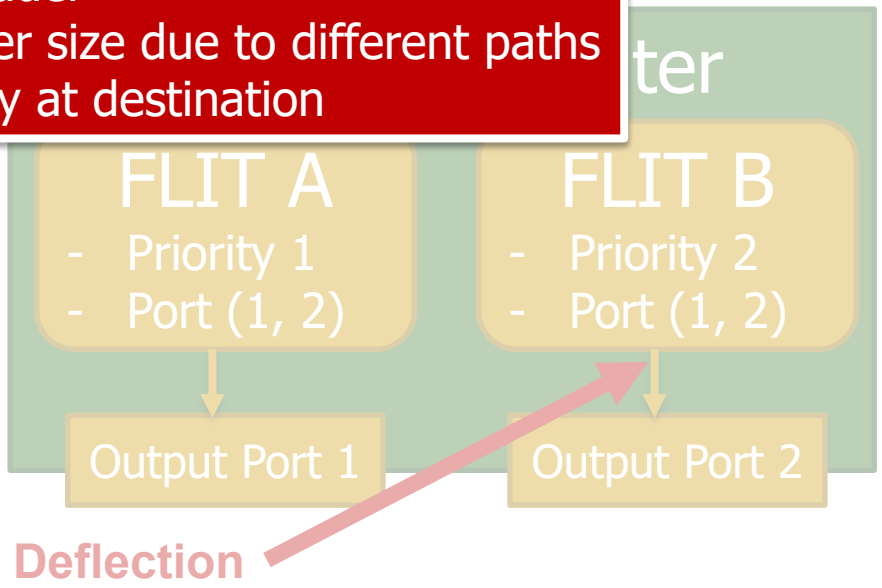
- If no productive output-port is available, send/deflect flit to a non-productive output-port
- Input ports ≤ output ports
- Routers form a connected graph
- Flit-Ranking
  - Oldest first
  - Avoids Livelocks
- Port-Prioritization
  - Different for every flit
  - Find the best output-ports



Basic Algorithm: Flit-Level Routing (II)

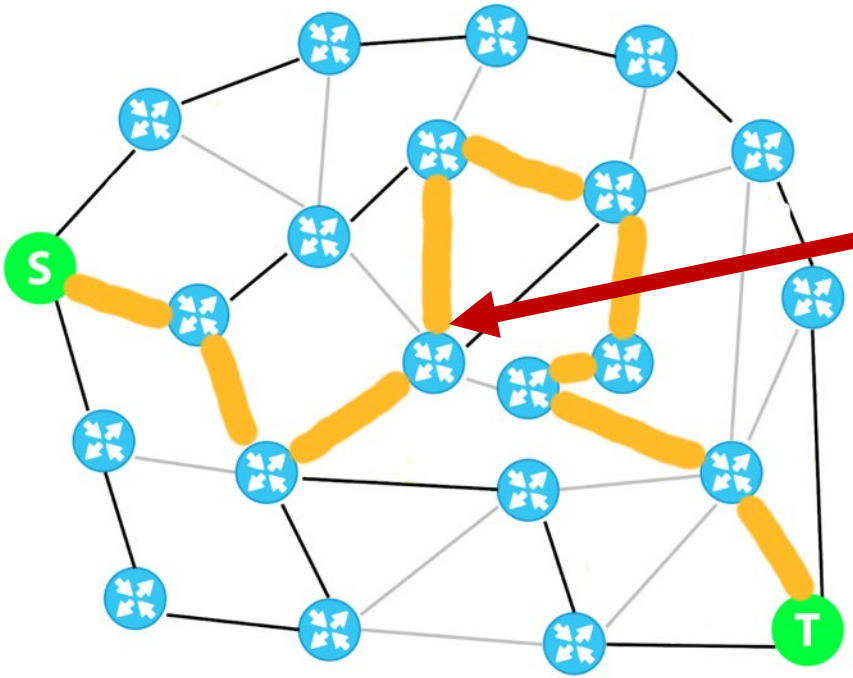
- If no productive output-port is available, send/deflect flit to a non-productive output-port
- Input ports ≤ output ports
- Routers form a connected graph
- Flit-Ranking
  - Oldes
  - Avoid
- Port-Pri
  - Different for every flit
  - Find the best output-ports

Limitations
- Each flit needs larger header
- Increase in receiver buffer size due to different paths
- Extra logic for reassembly at destination



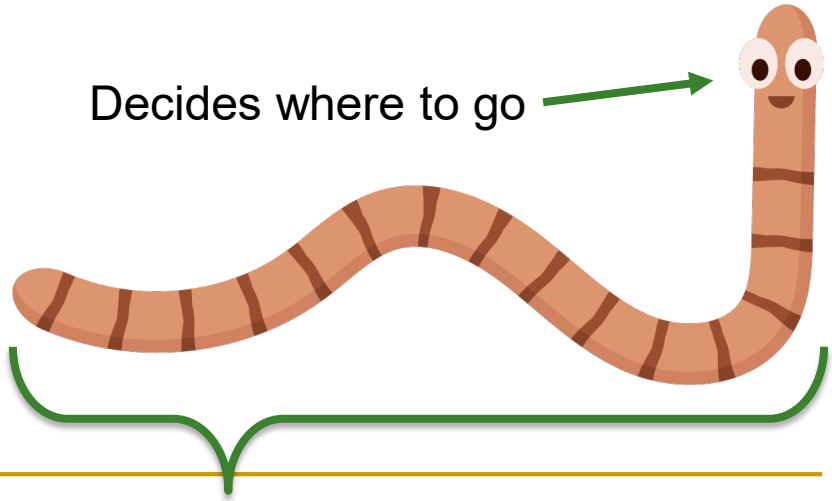
Optimized Version: BLESS Wormhole Routing

- Only the first of each packet/worm contains the header-info
- All other flits of the packet → follow the leading-flit



Deflection

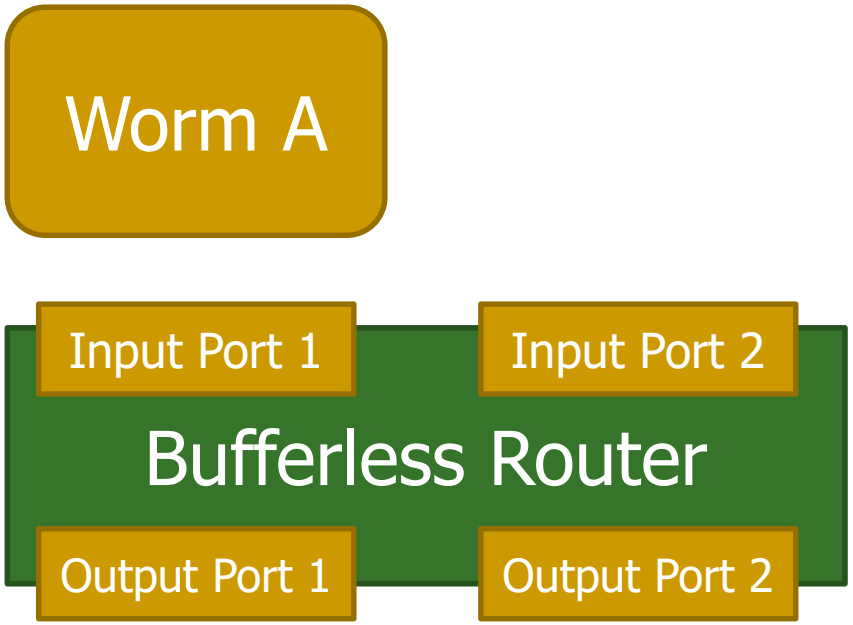
Decides where to go



Follow the “Head of the Worm”

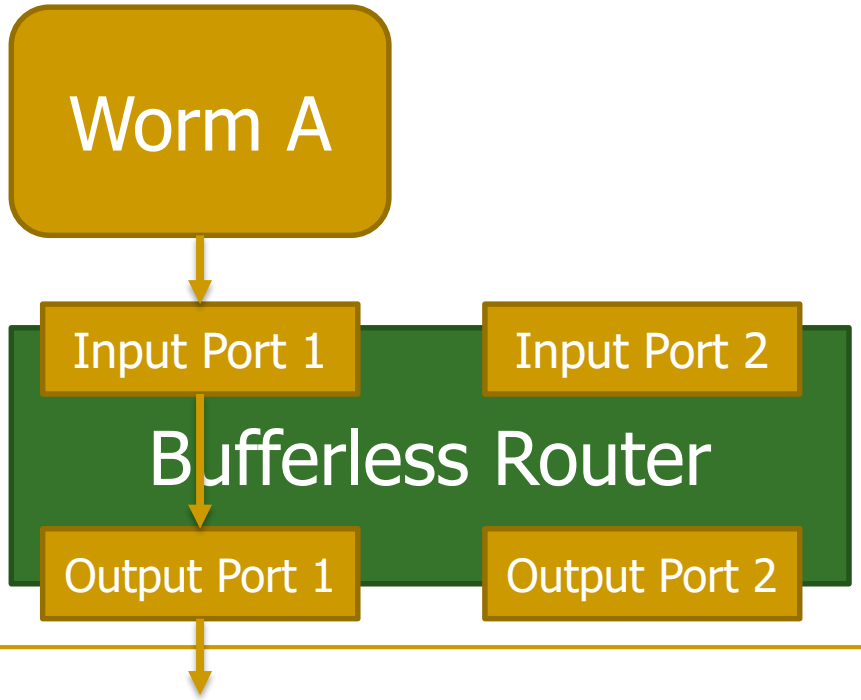
Wormhole Routing: Injection Problem

- **Injection Problem** (when is it safe to inject a new worm)
  - Whenever not all input-ports are busy
  - While inserting all input-ports become busy → truncate worm



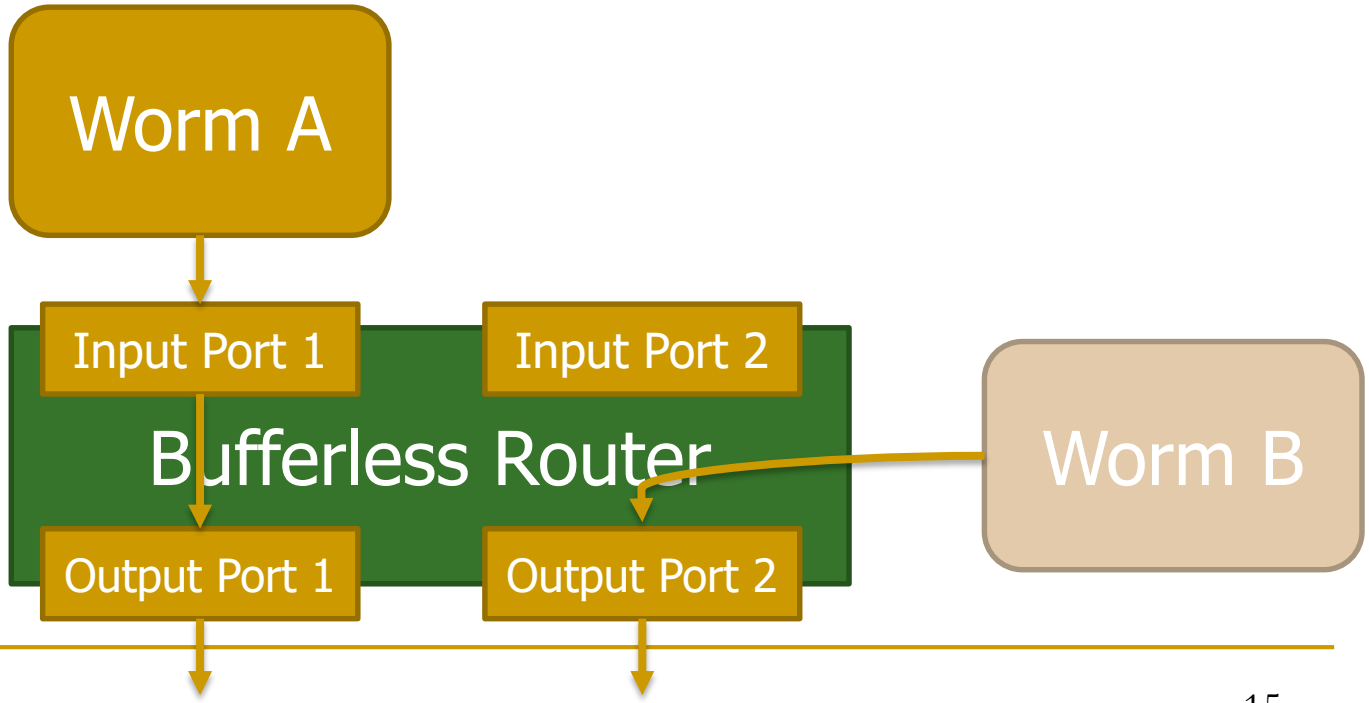
Wormhole Routing: Injection Problem

- **Injection Problem** (when is it safe to inject a new worm)
  - Whenever not all input-ports are busy
  - While inserting all input-ports become busy → truncate worm



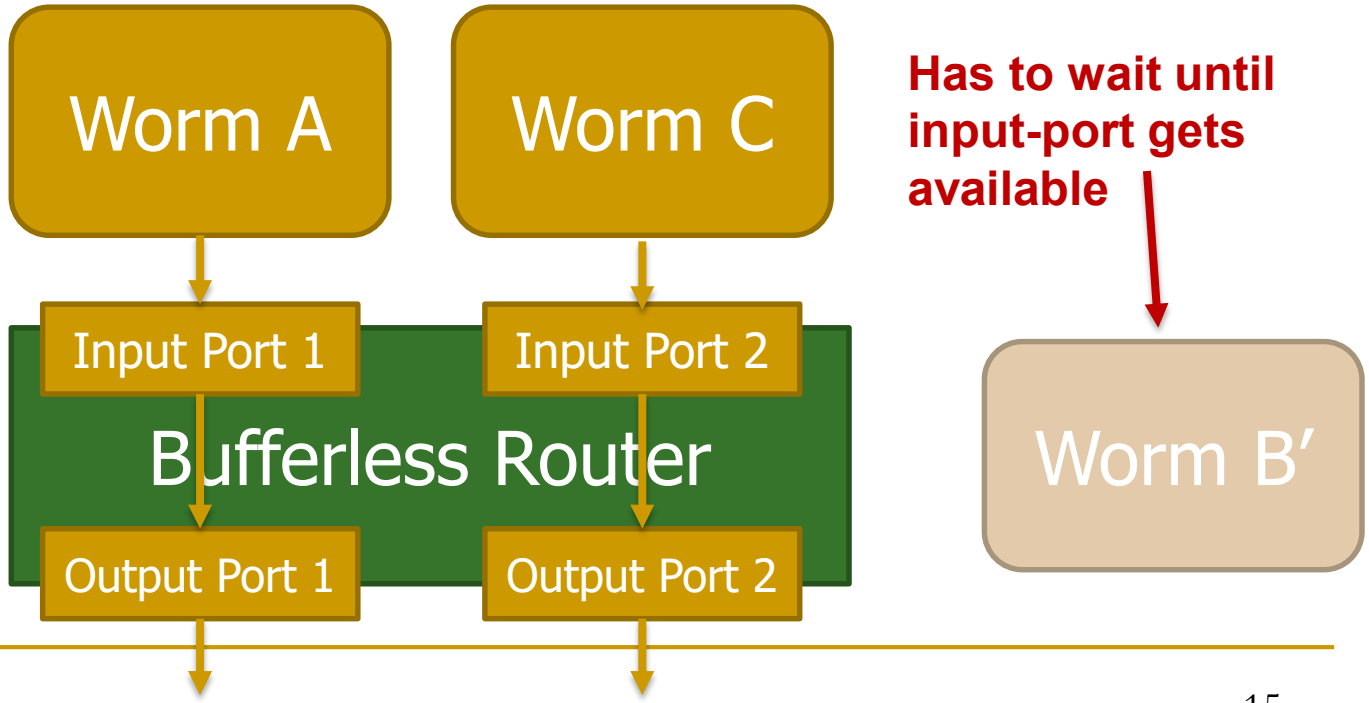
Wormhole Routing: Injection Problem

- **Injection Problem** (when is it safe to inject a new worm)
  - Whenever not all input-ports are busy
  - While inserting all input-ports become busy → truncate worm



Wormhole Routing: Injection Problem

- **Injection Problem** (when is it safe to inject a new worm)
  - Whenever not all input-ports are busy
  - While inserting all input-ports become busy → truncate worm



Wormhole Routing: Livelock Problem

- **Livelock Problem** (packets can be deflected forever)
  - Head-Flit
    - New output port must be allocated
    1. Unallocated, productive port → worm makes progress
    2. Allocated, productive port → other worm gets truncated
    3. Unallocated, non-productive port → worm is deflected
    4. Allocated, non-productive port → other worm gets truncated
  - Non-head-Flit
    - Flit is routed to same output-port as head-flit

Combined Version: BLESS with Buffers

- If good performance at high bandwidth rates is desired
- Implement Buffers into FLIT-BLESS or WORM-BLESS
- Buffers reduce probability of misrouting
- **If productive port isn't available → Buffer it**
- Whenever an input-buffer is full, the oldest flit in the buffer becomes "must-schedule-flit"
  - Must-schedule-flit must be send out in the next cycle
  - Mechanism to avoid buffer-overflow

Outline

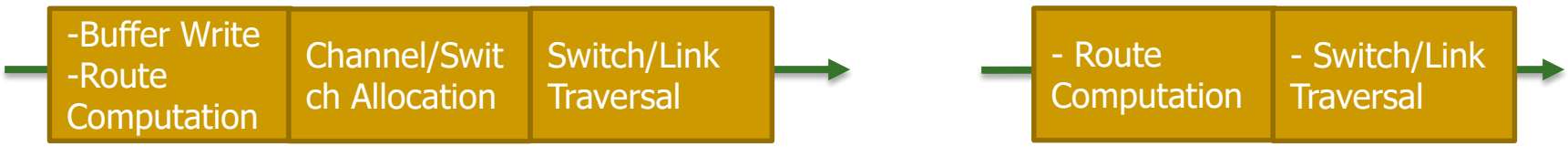
- Background, Problem & Goal
- Key Approach and Ideas
- Mechanisms (in some detail)
- **Benefits and Limitations**
- Key Results: Methodology and Evaluation
- Summary
- Strengths
- Weaknesses
- Takeaways
- Thoughts, Ideas and Discussion starters

Benefits

- No buffers
- Simpler/cheaper chip design
- Area savings
- Absence of Deadlocks
  - # Input ports ≤ # Output ports → packet will leave router
- Absence of Livelocks
  - Oldest-first flit-ranking and port prioritization
- Router latency reduction

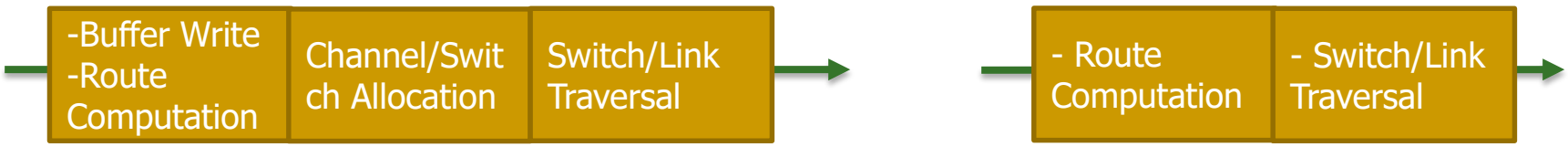
Benefits

- No buffers
- Simpler/cheaper chip design
- Area savings
- Absence of Deadlocks
  - # Input ports ≤ # Output ports → packet will leave router
- Absence of Livelocks
  - Oldest-first flit-ranking and port prioritization
- Router latency reduction



Benefits

- No buffers
- Simpler/cheaper chip design
- Area savings
- Absence of Deadlocks
  - # Input ports ≤ # Output ports → packet will leave router
- Absence of Livelocks
  - Oldest-first flit-ranking and port prioritization
- Router latency reduction

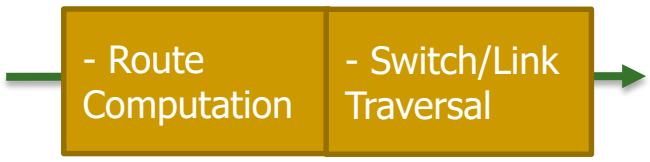


Router with input-buffers
Latency = 3

Benefits

- No buffers
- Simpler/cheaper chip design
- Area savings
- Absence of Deadlocks
  - # Input ports ≤ # Output ports → packet will leave router
- Absence of Livelocks
  - Oldest-first flit-ranking and port prioritization
- Router latency reduction

Router with input-buffers
Latency = 3



BLESS bufferless Router
Latency = 2

Limitations

- At high network utilization, deflections happen more often which causes unnecessary link/router traversals
  - Reduces network throughput
  - Increases latency
  - Increases link/routing energy consumption

Outline

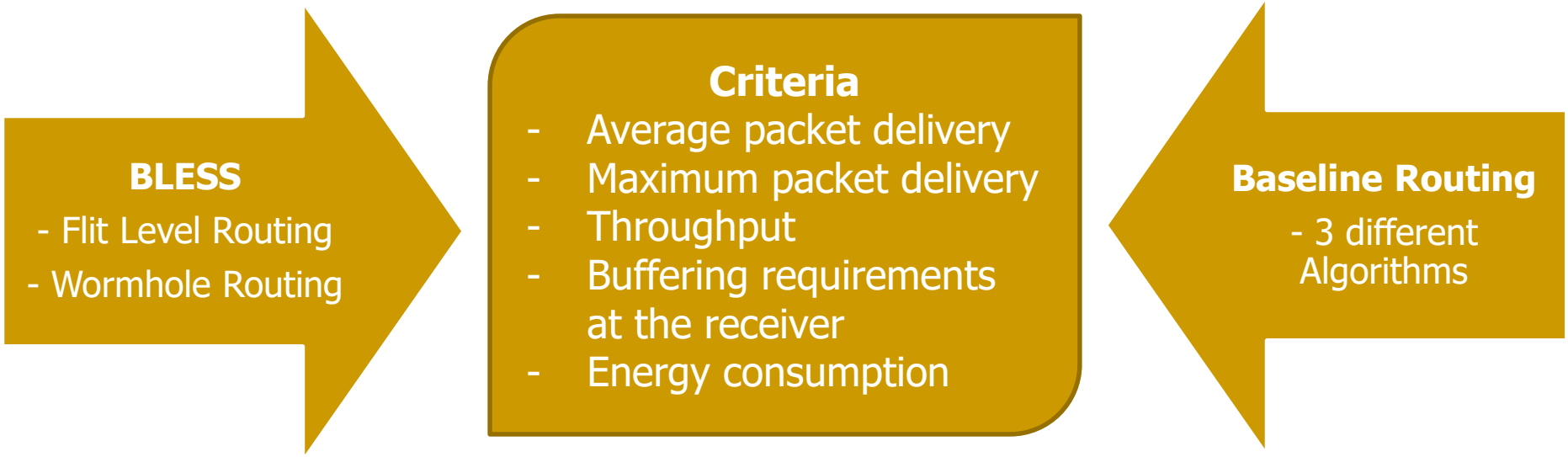
- Background, Problem & Goal
- Key Approach and Ideas
- Mechanisms (in some detail)
- Benefits and Limitations
- **Key Results: Methodology and Evaluation**
- Summary
- Strengths
- Weaknesses
- Takeaways
- Thoughts, Ideas and Discussion starters

Evaluation Methodology

- Cycle-accurate interconnection network simulator
- 5 input/output ports
- 1 Packet = 4 Flits
- Request generation: real world application
  - **Matlab** (most network intense)
  - Milc (=physical benchmark)
  - H264ref (=video encoder benchmark)

Evaluation Methodology

- Cycle-accurate interconnection network simulator
- 5 input/output ports
- 1 Packet = 4 Flits
- Request generation: real world application (e.g. Matlab)

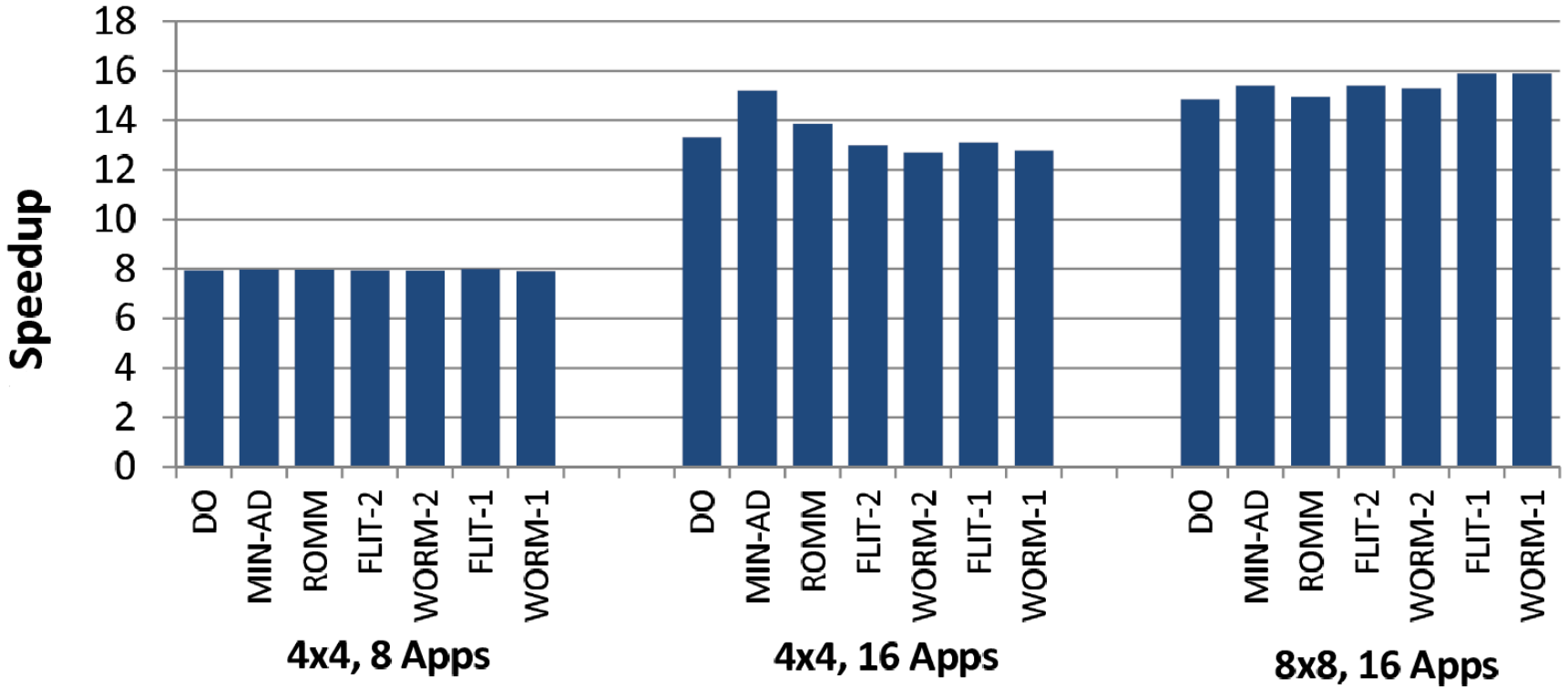


Results for homogenous Case: Matlab (I)

- Performance decrease without buffers relatively small
- Injection rates of real applications relatively low → Not many L1 misses

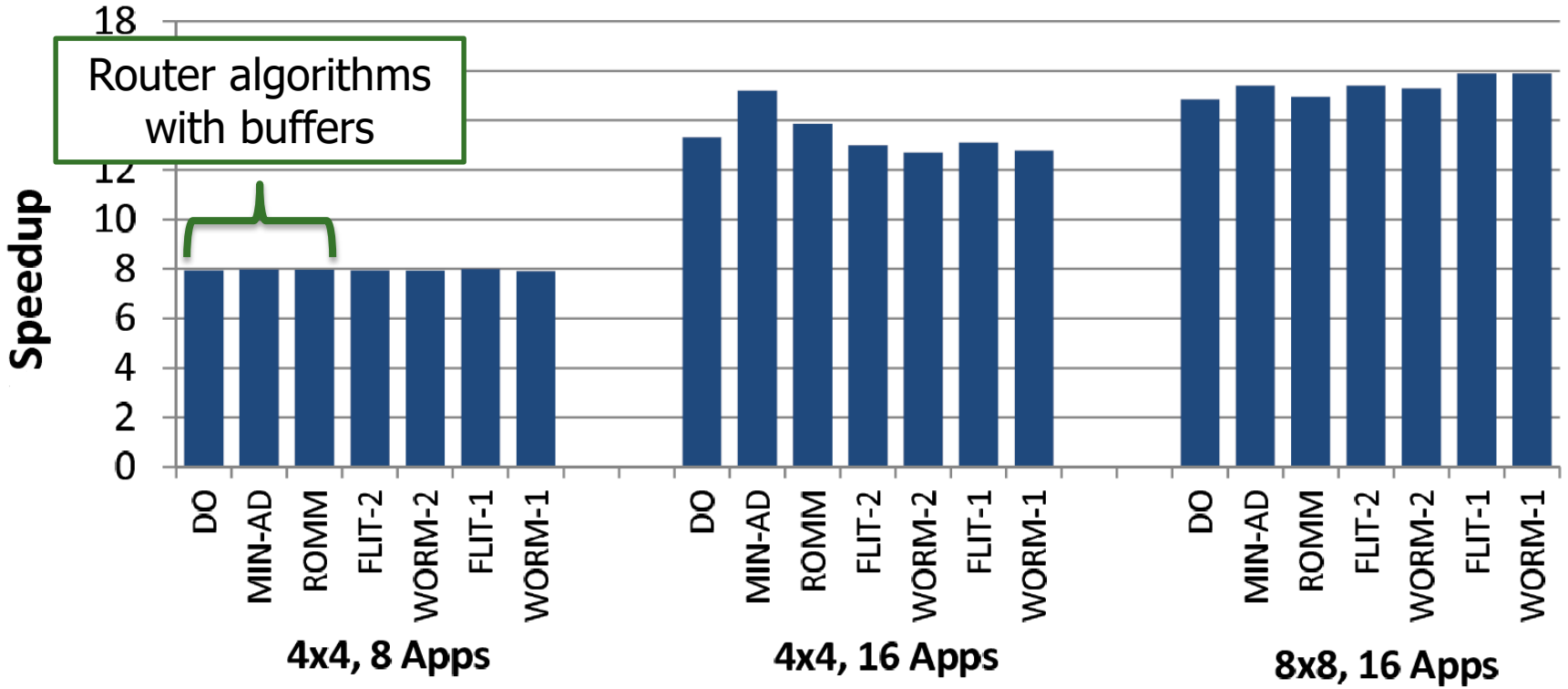
Results for homogenous Case: Matlab (I)

- Performance decrease without buffers relatively small
- Injection rates of real applications relatively low → Not many L1 misses



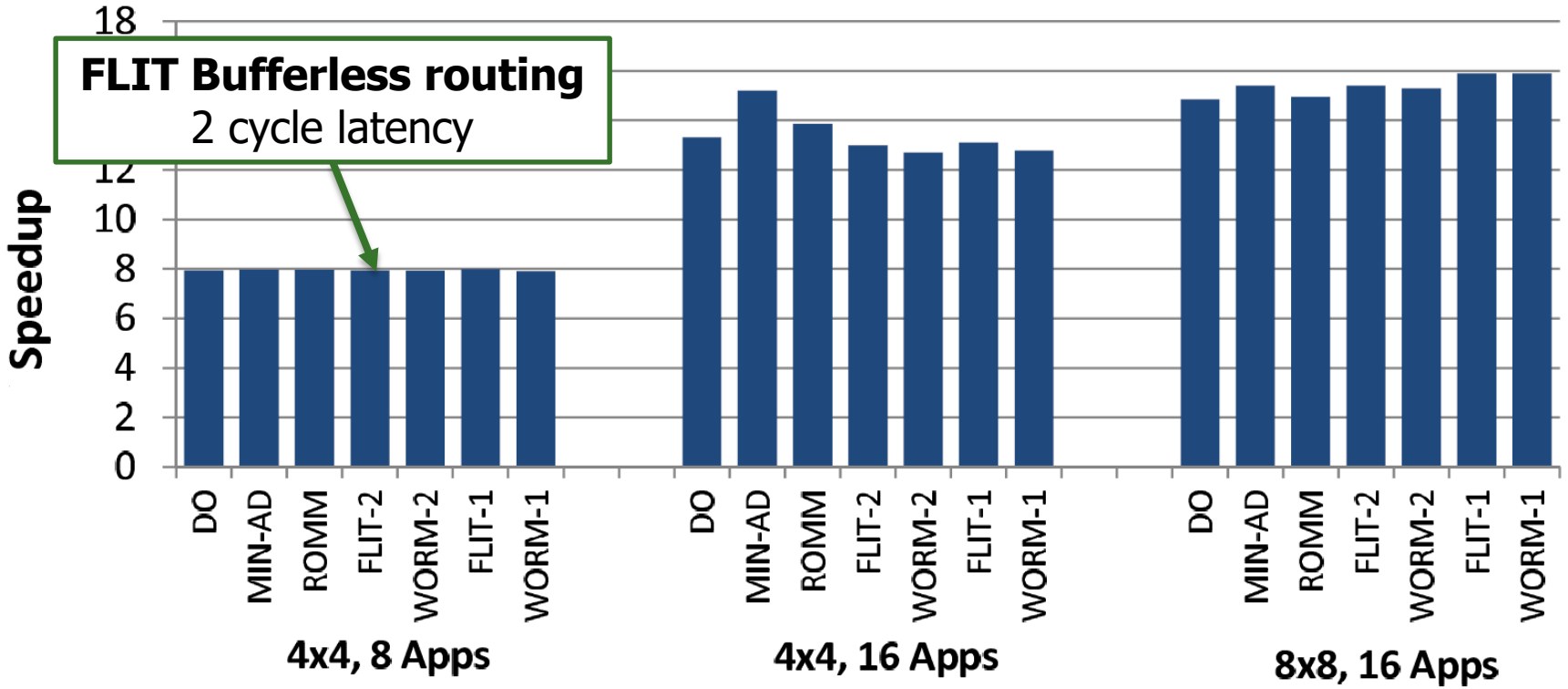
Results for homogenous Case: Matlab (I)

- Performance decrease without buffers relatively small
- Injection rates of real applications relatively low → Not many L1 misses



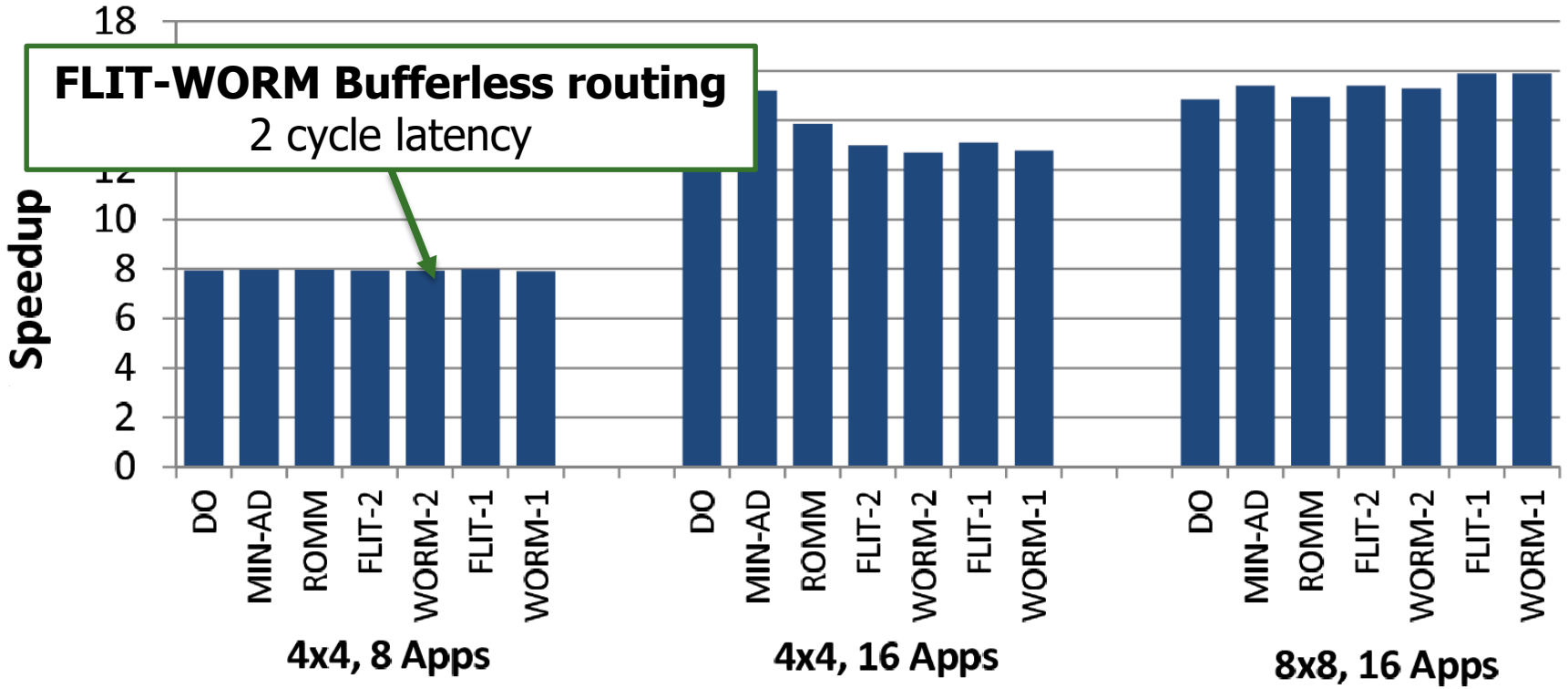
Results for homogenous Case: Matlab (I)

- Performance decrease without buffers relatively small
- Injection rates of real applications relatively low → Not many L1 misses



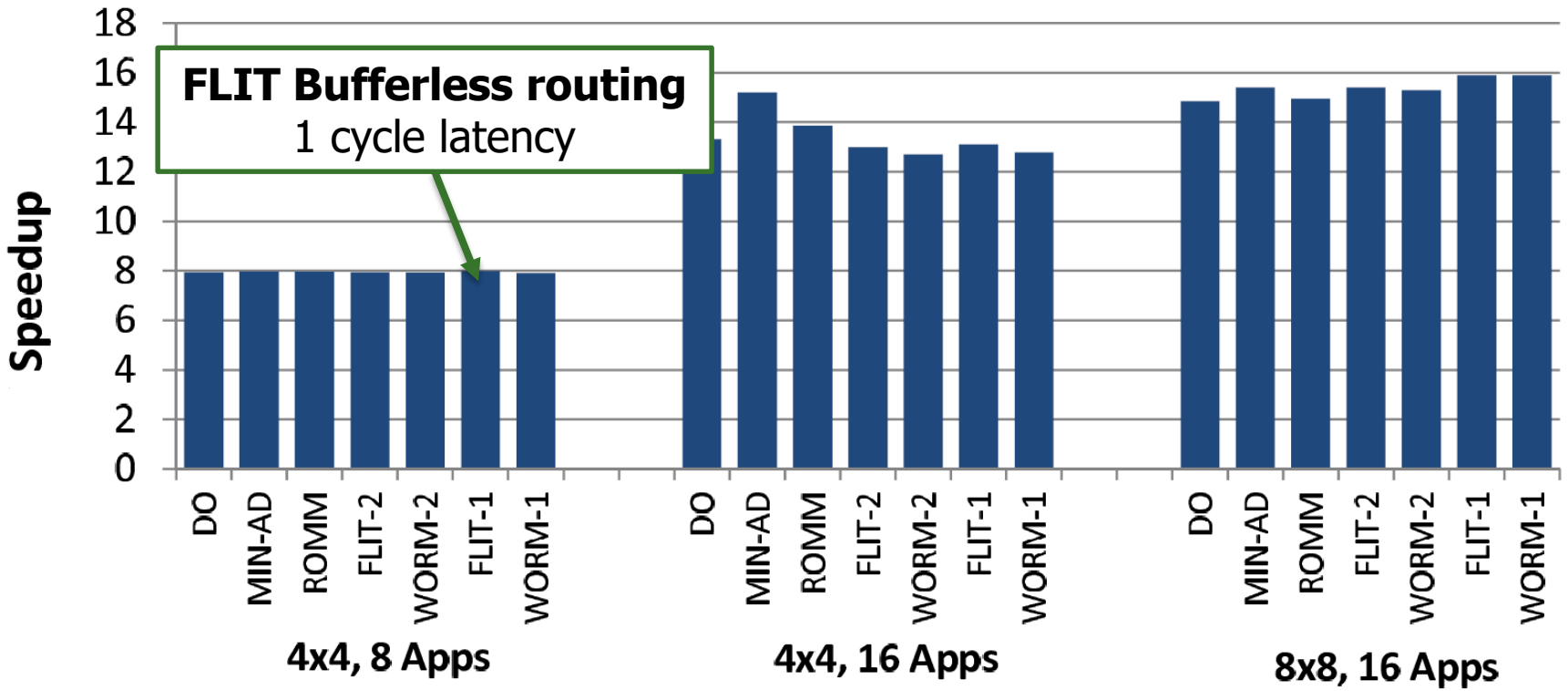
Results for homogenous Case: Matlab (I)

- Performance decrease without buffers relatively small
- Injection rates of real applications relatively low → Not many L1 misses



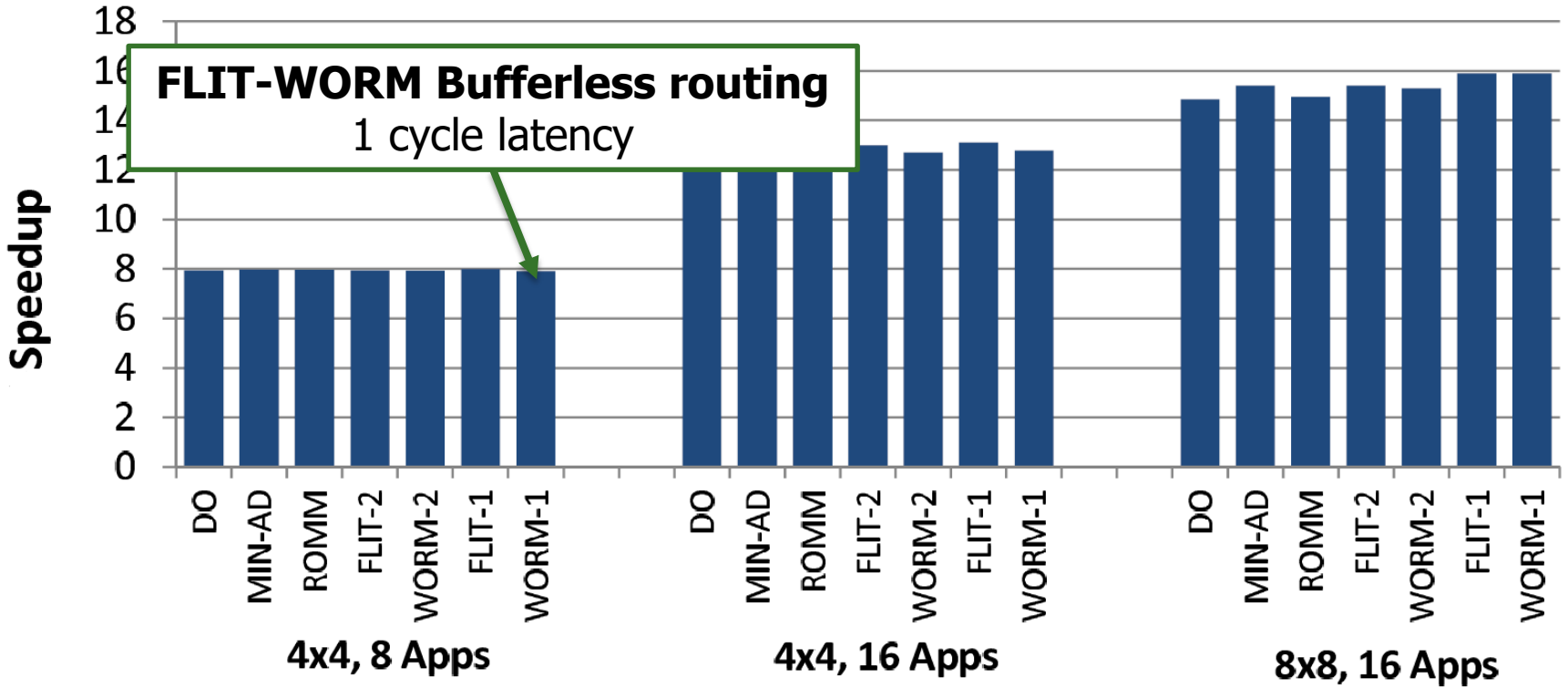
Results for homogenous Case: Matlab (I)

- Performance decrease without buffers relatively small
- Injection rates of real applications relatively low → Not many L1 misses



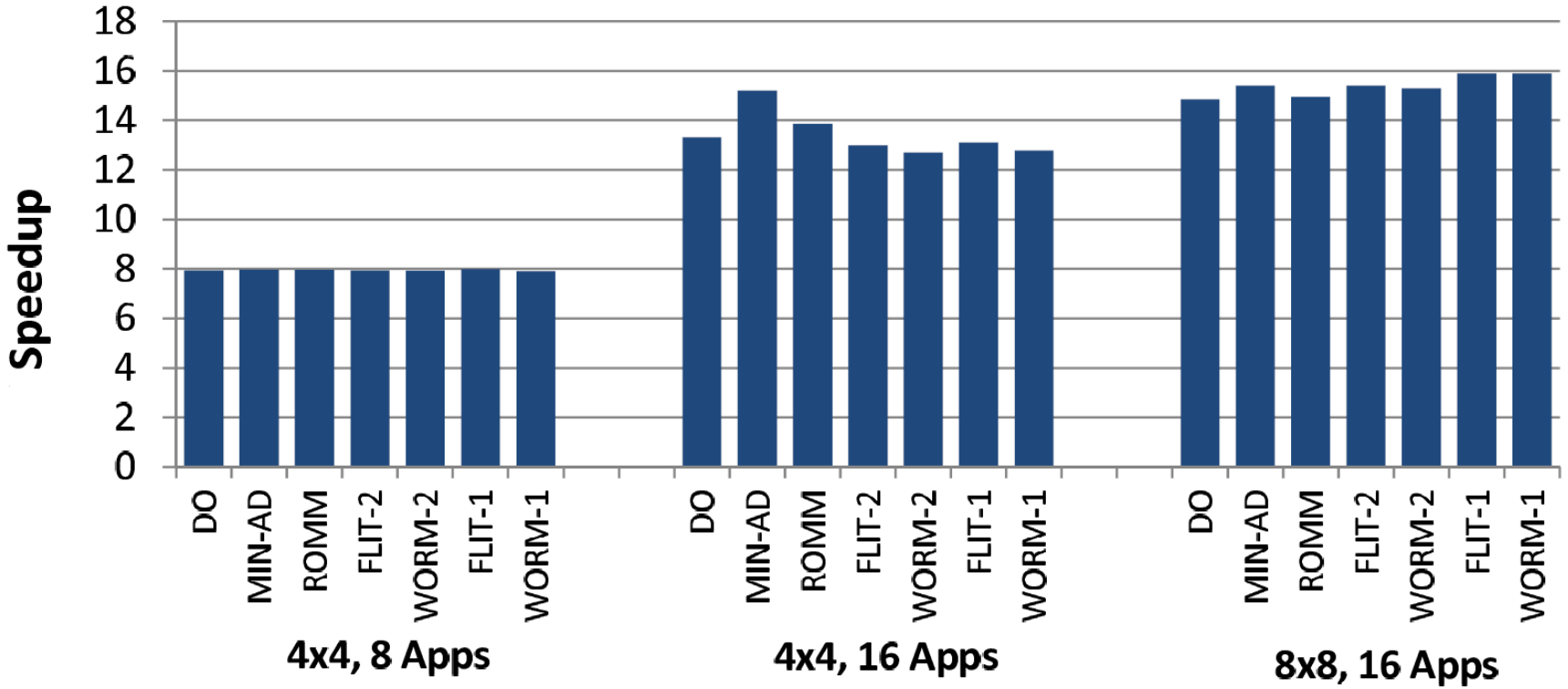
Results for homogenous Case: Matlab (I)

- Performance decrease without buffers relatively small
- Injection rates of real applications relatively low → Not many L1 misses



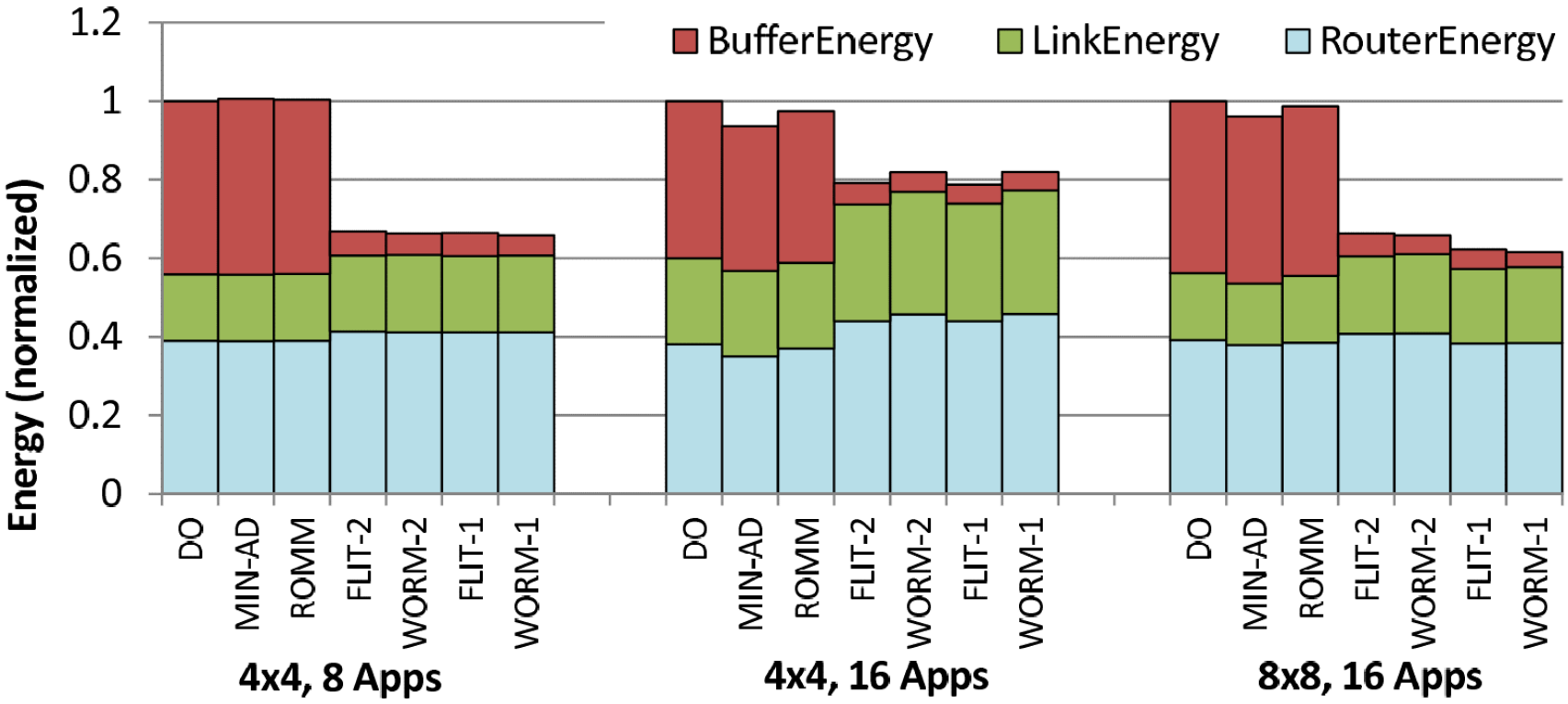
Results for homogenous Case: Matlab (I)

- Performance decrease without buffers relatively small
- Injection rates of real applications relatively low → Not many L1 misses



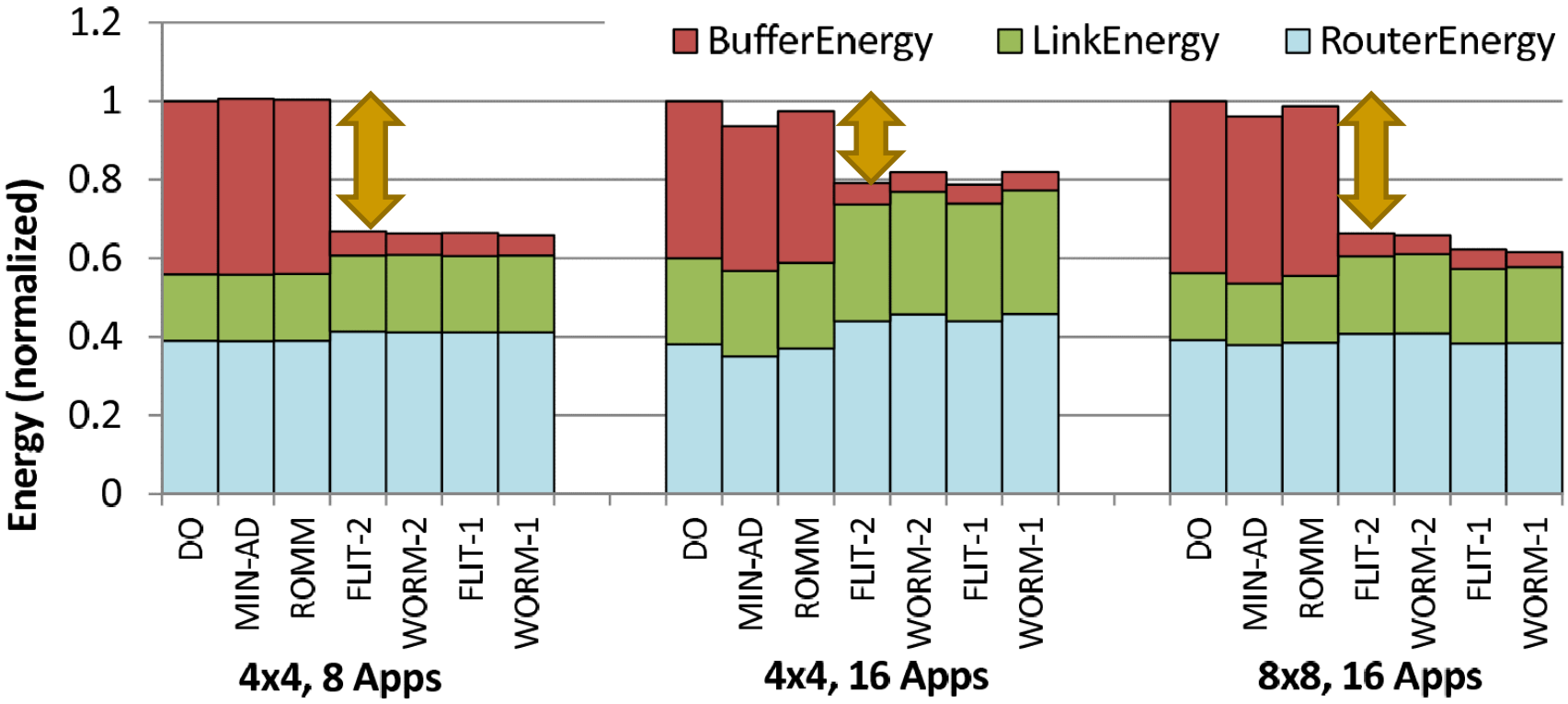
Results for homogenous Case: Matlab (II)

- BLESS significantly reduces energy consumption



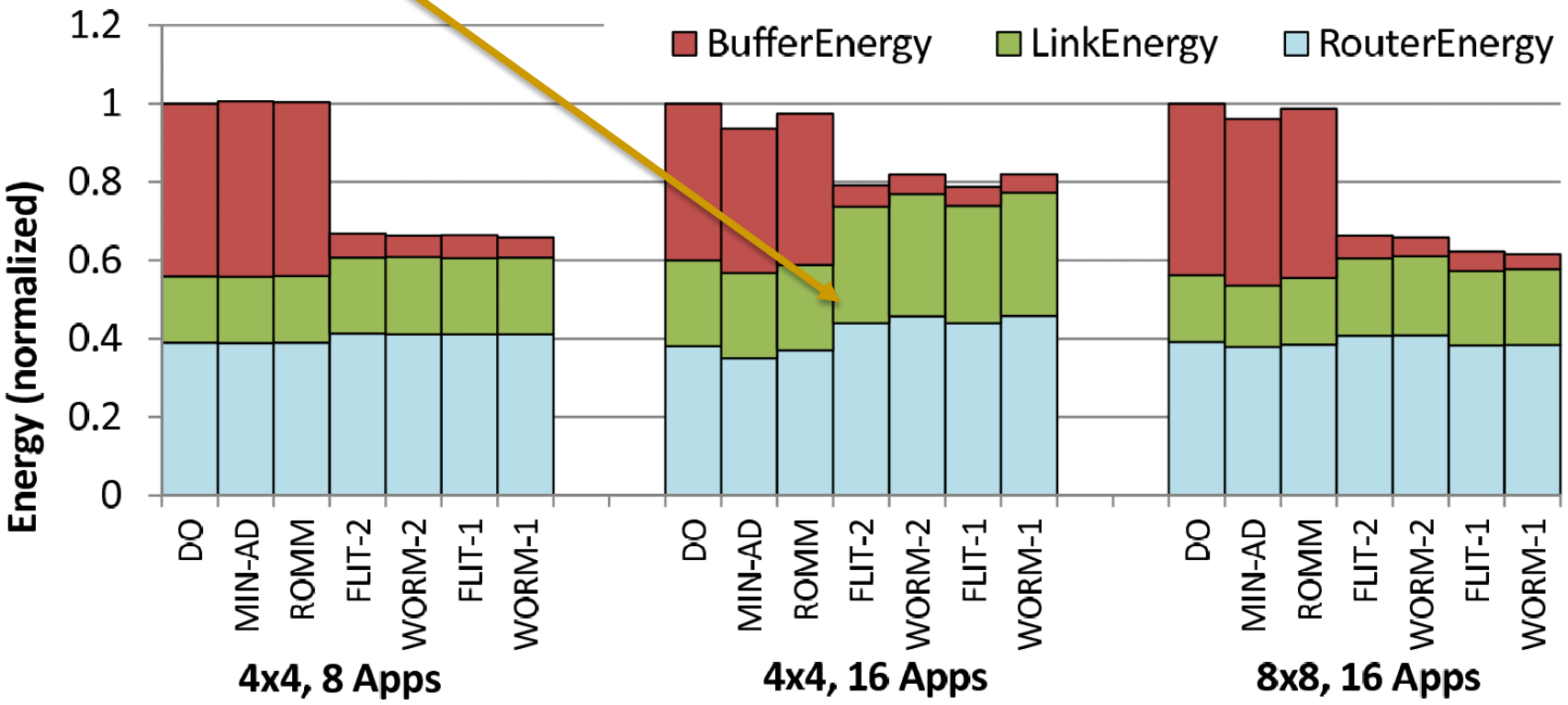
Results for homogenous Case: Matlab (II)

- BLESS significantly reduces energy consumption



Results for homogenous Case: Matlab (II)

- BLESS significantly reduces energy consumption
- Link/Router energy slightly higher due to deflections



Outline

- Background, Problem & Goal
- Key Approach and Ideas
- Mechanisms (in some detail)
- Benefits and Limitations
- Key Results: Methodology and Evaluation
- **Summary**
- Strengths
- Weaknesses
- Takeaways
- Thoughts, Ideas and Discussion starters

Summary

- **Problem:** The on chip networks in system on chips use the most energy/physical area for packet buffers which are used for routing the packets from different components on the chip.
- **Proposal:** We use three completely new routing algorithms "FLIT-Level-Routing", "Bless Wormhole Routing" and "Bless with Buffers" which aims to eliminate/reduce the need for buffers by deflecting packet inside the network.
- **Results:** Most of the time buffers are not needed on NoC
  - Average performance decrease by only 0.5%
  - Worst-case performance decrease by 3.2%
  - Average network energy consumption decrease by 39.4%
  - Area-savings of 60%

→ BLESS achieves significant energy savings at low performance loss

Outline

- Background, Problem & Goal
- Key Approach and Ideas
- Mechanisms (in some detail)
- Benefits and Limitations
- Key Results: Methodology and Evaluation
- Summary
- **Strengths**
- Weaknesses
- Takeaways
- Thoughts, Ideas and Discussion starters

Strengths

- Does not only use computer generated workload for evaluation
  - Video benchmark encoder
  - 3D fluid benchmark
- Had an impact on current bufferless research
  - Cited 377 times in other papers (last citation 29. October 2018)
- First paper which proposes variety of bufferless algorithms
- Buffers are everywhere: idea can be transferred to other areas
- Early evaluation of a problem that is more important than ever → Smartphones & Internet of things
- Good foundation for further research

Outline

- Background, Problem & Goal
- Key Approach and Ideas
- Mechanisms (in some detail)
- Benefits and Limitations
- Key Results: Methodology and Evaluation
- Summary
- Strengths
- **Weaknesses**
- Takeaways
- Thoughts, Ideas and Discussion starters

Weaknesses

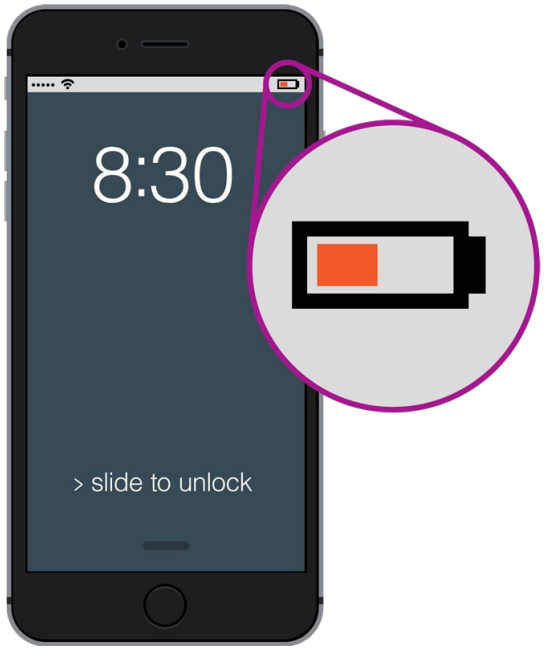
- No explanation why certain programs for evaluations were chosen
  - Matlab on SoC not typical
  - What does Matlab compute?
- Always speaks of bufferless routing but there need to be more buffers at the receiver side → How to reassembly packet with receiver buffer not covered
- Some critical features are not implemented
  - Manual priorities for different packets
  - Congestion control
    - "Next generation on-chip networks: what kind of congestion control do we need?" by Onur Mutlu in 2010
- Assumes no faulty routers/links

Outline

- Background, Problem & Goal
- Key Approach and Ideas
- Mechanisms (in some detail)
- Benefits and Limitations
- Key Results: Methodology and Evaluation
- Summary
- Strengths
- Weaknesses
- **Takeaways**
- Thoughts, Ideas and Discussion starters

Takeaways

- Very important topic, especially today
- Research about bufferless routing is still in progress
  - Latest research paper published on 11th of June 2018
  - "High-performance 3D NoC bufferless router with approximate priority comparison" by Konstantinos Tatas
- BLESS is going into the right direction but it lacks some needed functions
  - Built foundation for further research



Outline

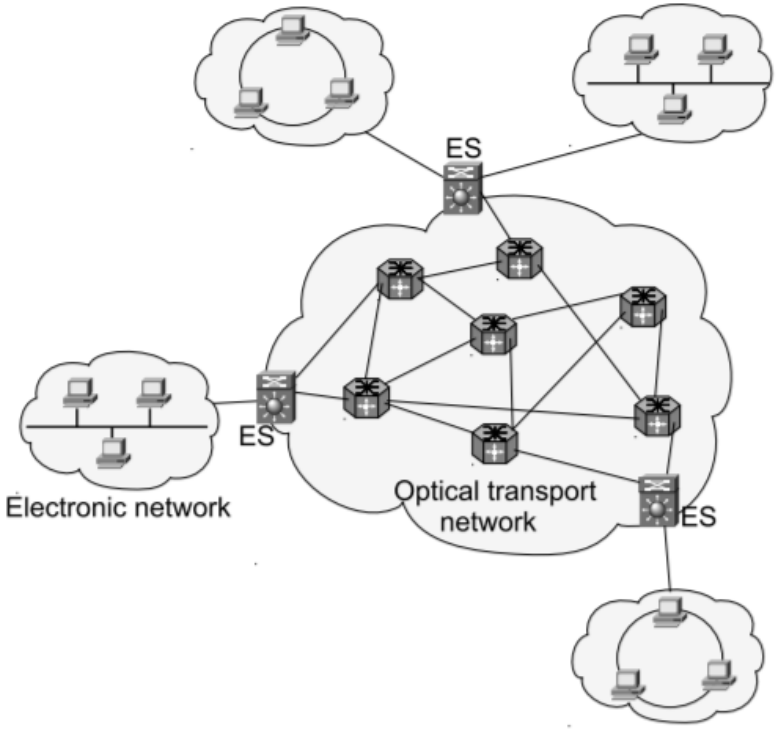
- Background, Problem & Goal
- Key Approach and Ideas
- Mechanisms (in some detail)
- Benefits and Limitations
- Key Results: Methodology and Evaluation
- Summary
- Strengths
- Weaknesses
- Takeaways
- **Thoughts, Ideas and Discussion starters**

Thoughts, Ideas and Discussion starters

Are there any questions?

Thoughts, Ideas and Discussion starters

- **In what other areas could bufferless routing be used?**
  - "Deflection routing in IP optical networks", Guido Maier 2011
  - Optical data transfer is much faster than buffers
  - Deflection routing as an alternative in an optical network without using buffers
  - Today, optical networks use only a small fraction of the large capacity since switching, processing and storage technologies aren't that fast

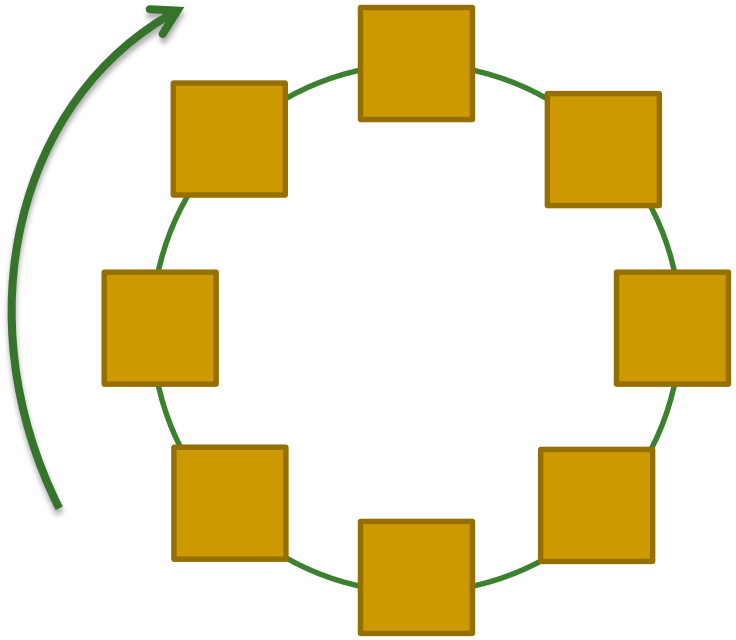


Thoughts, Ideas and Discussion starters

- **Other ideas to eliminate buffers without deflections?**
  - "Scarab: A single cycle adaptive routing an bufferless network", M. Hayenga, Micro-42, 2009
  - Drop based bufferless routing
  - Just drop packages when the router is congested
  - Establish circuit-switched backend for requesting retransmits
  - Requires extra links for the retransmit-requests

Thoughts, Ideas and Discussion starters

- **Other ideas to eliminate buffers without deflections?**
  - Ring based interconnect
  - No routing is needed at all, just forward the packet inside the ring until it reaches the desired node
  - Not suitable for large networks



Thoughts, Ideas and Discussion starters

- **Is switching between bufferless routing and routing with buffers a good idea (=Hybrid Routing)?**
  - "Adaptive flow control for robust performance and energy", Jafri et al, Micro-43, 2010
  - Energy savings but no area savings
  - Switch between bufferless deflection routing and buffered operation depending on the needed bandwidth

A Case for Bufferless Routing in On-Chip Networks

Onur Mutlu
Carnegie Mellon University

Thomas Moscibroda
Microsoft Research

36th International Symposium on Computer Architecture
June 22, 2009
Austin, Texas, USA

Presented by Jonas Bokstaller
14 November 2018