Readings: Interconnection Networks

Required


Recommended

Last Lecture

- Interconnection Networks
  - Introduction & Terminology
  - Topology
  - Buffering and Flow control
Today

- Review (Topology & Flow Control)
- More on interconnection networks
  - Routing
  - Router design
  - Network performance metrics
  - On-chip vs. off-chip differences

- Research on NoC Router Design
  - BLESS: bufferless deflection routing
  - CHIPPER: cheaper bufferless deflection routing
  - MinBD: adding small buffers to recover some performance

- Research on Congestion Control
  - HAT: Heterogeneous Adaptive Throttling
Today

- **Review (Topology & Flow Control)**
- More on interconnection networks
  - Routing
  - Router design
  - Network performance metrics
  - On-chip vs. off-chip differences

- Research on NoC Router Design
  - BLESS: bufferless deflection routing
  - CHIPPER: cheaper bufferless deflection routing
  - MinBD: adding small buffers to recover some performance

- Research on Congestion Control
  - HAT: Heterogeneous Adaptive Throttling
### Review: Topologies

<table>
<thead>
<tr>
<th>Topology</th>
<th>Crossbar</th>
<th>Multistage Logarithm.</th>
<th>Mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct/Indirect</td>
<td>Indirect</td>
<td>Indirect</td>
<td>Direct</td>
</tr>
<tr>
<td>Blocking/Non-blocking</td>
<td>Non-blocking</td>
<td>Blocking</td>
<td>Blocking</td>
</tr>
<tr>
<td>Cost</td>
<td>$O(N^2)$</td>
<td>$O(N \log N)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>Latency</td>
<td>$O(1)$</td>
<td>$O(\log N)$</td>
<td>$O(\sqrt{N})$</td>
</tr>
</tbody>
</table>
Review: Flow Control

Store and Forward

Any other issues?

Head-of-Line Blocking

Use Virtual Channels
Review: Flow Control

Store and Forward

Cut Through / Wormhole

Shrink Buffers
Reduce latency

Any other issues?
Head-of-Line Blocking
Use Virtual Channels

Buffer full: blue cannot proceed
Blocked by other packets
Today

- Review (Topology & Flow Control)

- More on interconnection networks
  - **Routing**
  - Router design
  - Network performance metrics
  - On-chip vs. off-chip differences

- Research on NoC Router Design
  - BLESS: bufferless deflection routing
  - CHIPPER: cheaper bufferless deflection routing
  - MinBD: adding small buffers to recover some performance

- Research on Congestion Control
  - HAT: Heterogeneous Adaptive Throttling
Routing Mechanism

■ Arithmetic
  - Simple arithmetic to determine route in regular topologies
  - Dimension order routing in meshes/tori

■ Source Based
  - Source specifies output port for each switch in route
    + Simple switches
      - no control state: strip output port off header
    - Large header

■ Table Lookup Based
  - Index into table for output port
    + Small header
    - More complex switches
Routing Algorithm

- **Types**
  - **Deterministic**: always choose the same path
  - **Oblivious**: do not consider network state (e.g., random)
  - **Adaptive**: adapt to state of the network

- **How to adapt**
  - Local/global feedback
  - Minimal or non-minimal paths
Deterministic Routing

- All packets between the same (source, dest) pair take the same path

- **Dimension-order routing**
  - E.g., XY routing (used in Cray T3D, and many on-chip networks)
  - First traverse dimension X, then traverse dimension Y

  + Simple
  + Deadlock freedom (no cycles in resource allocation)
  - Could lead to high contention
  - Does not exploit path diversity
Deadlock

- No forward progress
- Caused by circular dependencies on resources
- Each packet waits for a buffer occupied by another packet downstream
Handling Deadlock

- Avoid cycles in routing
  - Dimension order routing
    - Cannot build a circular dependency
  - Restrict the “turns” each packet can take

- Avoid deadlock by adding virtual channels
  - Separate VC pool per distance

- Detect and break deadlock
  - Preemption of buffers
Turn Model to Avoid Deadlock

- **Idea**
  - Analyze directions in which packets can turn in the network
  - Determine the cycles that such turns can form
  - Prohibit just enough turns to break possible cycles


![Diagrams](image-url)
Valiant’s Algorithm

- An example of oblivious algorithm
- Goal: Balance network load
- Idea: Randomly choose an intermediate destination, route to it first, then route from there to destination
  - Between source-intermediate and intermediate-dest, can use dimension order routing

- Randomizes/balances network load
- Non minimal (packet latency can increase)

Optimizations:
- Do this on high load
- Restrict the intermediate node to be close (in the same quadrant)
Adaptive Routing

- **Minimal adaptive**
  - Router uses network state (e.g., downstream buffer occupancy) to pick which “productive” output port to send a packet to
  - Productive output port: port that gets the packet closer to its destination
  - + Aware of local congestion
  - - Minimality restricts achievable link utilization (load balance)

- **Non-minimal (fully) adaptive**
  - “Misroute” packets to non-productive output ports based on network state
  - + Can achieve better network utilization and load balance
  - - Need to guarantee livelock freedom
More on Adaptive Routing

- Can avoid faulty links/routers

- Idea: Route around faults

  + Deterministic routing cannot handle faulty components
  - Need to change the routing table to disable faulty routes
    - Assuming the faulty link/router is detected
Today

- Review (Topology & Flow Control)
- More on interconnection networks
  - Routing
  - **Router design**
  - Network performance metrics
  - On-chip vs. off-chip differences

- Research on NoC Router Design
  - BLESS: bufferless deflection routing
  - CHIPPER: cheaper bufferless deflection routing
  - MinBD: adding small buffers to recover some performance

- Research on Congestion Control
  - HAT: Heterogeneous Adaptive Throttling
On-chip Networks

- Router
- Processing Element (Cores, L2 Banks, Memory Controllers etc.)

Input Port with Buffers
- VC Identifier
- VC 0
- VC 1
- VC 2

Control Logic
- Routing Unit (RQ)
- VC Allocator (VA)
- Switch Allocator (SA)

Crossbar (5 x 5)
- To East
- To West
- To North
- To South
- To PE
Router Design: Functions of a Router

- Buffering (of flits)

- Route computation

- Arbitration of flits (i.e. prioritization) when contention
  - Called packet scheduling

- Switching
  - From input port to output port

- Power management
  - Scale link/router frequency
Router Pipeline

- Five logical stages
  - BW: Buffer Write
  - RC: Route computation
  - VA: Virtual Channel Allocation
  - SA: Switch Allocation
  - ST: Switch Traversal
  - LT: Link Traversal
Wormhole Router Timeline

- Route computation performed once per packet
- Virtual channel allocated once per packet
- Body and tail flits inherit this information from head flit
Dependencies in a Router

- Dependence between output of one module and input of another
  - Determine critical path through router
  - Cannot bid for switch port until routing performed
Pipeline Optimizations: Lookahead Routing

- At current router perform routing computation for next router
  - Overlap with BW
    - Precomputing route allows flits to compete for VCs immediately after BW
    - RC decodes route header
    - Routing computation needed at next hop
      - Can be computed in parallel with VA

Pipeline Optimizations: Speculation

- Assume that Virtual Channel Allocation stage will be successful
  - Valid under low to moderate loads
- Entire VA and SA in parallel

<table>
<thead>
<tr>
<th>BW</th>
<th>VA</th>
<th>ST</th>
<th>LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- If VA unsuccessful (no virtual channel returned)
  - Must repeat VA/SA in next cycle
- Prioritize non-speculative requests
Pipeline Optimizations: Bypassing

- When no flits in input buffer
  - Speculatively enter ST
  - On port conflict, speculation aborted

- In the first stage, a free VC is allocated, next routing is performed and the crossbar is setup
Today

- Review (Topology & Flow Control)
- **More on interconnection networks**
  - Routing
  - Router design
  - Network performance metrics
  - On-chip vs. off-chip differences

- Research on NoC Router Design
  - BLESS: bufferless deflection routing
  - CHIPPER: cheaper bufferless deflection routing
  - MinBD: adding small buffers to recover some performance

- **Research on Congestion Control**
  - HAT: Heterogeneous Adaptive Throttling
Interconnection Network Performance

- **Latency**
  - **Zero load latency** (topology+routing+flow control)
  - **Min latency** given by routing algorithm
  - **Min latency** given by topology

- **Throughput**
  - Given by flow control
  - Given by routing
  - Given by topology
Ideal Latency

- Ideal latency
  - Solely due to wire delay between source and destination

\[ T_{ideal} = \frac{D}{v} + \frac{L}{b} \]

- \( D \) = Manhattan distance
- \( L \) = packet size
- \( b \) = channel bandwidth
- \( v \) = propagation velocity
Actual Latency

- Dedicated wiring impractical
  - Long wires segmented with insertion of routers

\[ T_{\text{actual}} = \frac{D}{v} + \frac{L}{b} + H \cdot T_{\text{router}} + T_c \]

- \( D \) = Manhattan distance
- \( L \) = packet size
- \( b \) = channel bandwidth
- \( v \) = propagation velocity
- \( H \) = hops
- \( T_{\text{router}} \) = router latency
- \( T_c \) = latency due to contention
Network Performance Metrics

- Packet latency
- Round trip latency
- Saturation throughput
- Application-level performance: system performance
  - Affected by interference among threads/applications
Today

- Review (Topology & Flow Control)
- **More on interconnection networks**
  - Routing
  - Router design
  - Network performance metrics
  - **On-chip vs. off-chip differences**

- Research on NoC Router Design
  - BLESS: bufferless deflection routing
  - CHIPPER: cheaper bufferless deflection routing
  - MinBD: adding small buffers to recover some performance

- Research on Congestion Control
  - HAT: Heterogeneous Adaptive Throttling
On-Chip vs. Off-Chip Differences

Advantages of on-chip

- **Wires are “free”**
  - Can build highly connected networks with wide buses

- **Low latency**
  - Can cross entire network in few clock cycles

- **High Reliability**
  - Packets are not dropped and links rarely fail

Disadvantages of on-chip

- **Sharing resources with rest of components on chip**
  - Area
  - Power

- **Limited buffering available**

- **Not all topologies map well to 2D plane**
Today

- Review (Topology & Flow Control)
- More on interconnection networks
  - Routing
  - Router design
  - Network performance metrics
  - On-chip vs. off-chip differences

- **Research on NoC Router Design**
  - BLESS: bufferless deflection routing
  - CHIPPER: cheaper bufferless deflection routing
  - MinBD: adding small buffers to recover some performance

- **Research on Congestion Control**
  - HAT: Heterogeneous Adaptive Throttling
A Case for Bufferless Routing in On-Chip Networks

Thomas Moscibroda
Microsoft Research

Onur Mutlu
CMU
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 chip
- Design goals in NoC design:
  - High throughput
  - Fairness between cores
  - Low complexity
  - Power, low energy consumption

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!

Thomas Moscibroda, Microsoft Research
A Typical Router

Buffers are integral part of existing NoC Routers

Thomas Moscibroda, Microsoft Research
Buffers in NoC Routers

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

Graph showing the relationship between average packet latency and injection rate for small, medium, and large buffers.
Buffers in NoC Routers

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

- Buffers consume significant energy:
  - Dynamic energy during read/write
  - Static energy even when not occupied

- Buffers add complexity and latency:
  - Logic for buffer management
  - Virtual channel allocation
  - Credit-based flow control

- Buffers 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!

Thomas Moscibroda, Microsoft Research
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]

Thomas Moscibroda, Microsoft Research
BLESS: Bufferless Routing

1. Create a ranking over all incoming flits

2. For a given flit in this ranking, find the best free output-port
   Apply to each flit in order of ranking

Thomas Moscibroda, Microsoft Research
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

Thomas Moscibroda, Microsoft Research
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**…?
  - Problems:
    - **Injection Problem**
      (not known when it is safe to inject)
    - **Livelock Problem**
      (packets can be deflected forever)

[Dally, Seitz’86]
WORM-BLESS: Wormhole Routing

1. Oldest-first ranking
2. If flit is head-flit
   a) assign flit to unallocated, productive port
   b) assign flit to allocated, productive port
   c) assign flit to unallocated, non-productive port
   d) assign flit to allocated, non-productive port
else,
   a) assign flit to port that is allocated to worm

Deflect worms if necessary!
Truncate worms if necessary!

At low congestion, packets travel routed as worms

Body-flit turns into head-flit

Head-flit: West
 allocated to North
 allocated to West

This worm is truncated! & deflected!

See paper for details...

Thomas Moscibroda, Microsoft Research
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, its 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

Baseline Router (speculative)

BLESS Router (standard)

BLESS Router (optimized)

Thomas Moscibroda, Microsoft Research

**Baselines**
- **Router 1**
  - **RC**
  - **ST**
  - **LT**

**BLESS Router**

**Router 1**
- **RC**
- **ST**
- **LA LT**

**Router 2**
- **RC**
- **ST**

**BLESS Router**

**Router 1**
- **RC**
- **ST**

**Router 2**
- **RC**
- **ST**

**Router Latency**

- **Router Latency = 3**
- **Router Latency = 2**
- **Router Latency = 1**

**Notations**
- **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]

Can be improved to 2.
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…?

Extensive evaluations in the paper!
Evaluation Methodology

• 2D mesh network, router latency is 2 cycles
  o 4x4, 8 core, 8 L2 cache banks (each node is a core or an L2 bank)
  o 4x4, 16 core, 16 L2 cache banks (each node is a core and an L2 bank)
  o 8x8, 16 core, 64 L2 cache banks (each node is L2 bank and may be a core)
  o 128-bit wide links, 4-flit data packets, 1-flit address packets
  o For baseline configuration: 4 VCs per physical input port

• Benchmarks
  o Multiprogrammed SPEC CPU2006 and Windows Desktop applications
  o Heterogeneous and homogenous application mixes
  o Synthetic traffic patterns: UR, Transpose, Tornado, Bit Complement

• x86 processor model based on Intel Pentium M
  o 2 GHz processor, 128-entry instruction window
  o 64Kbyte private L1 caches
  o Total 16Mbyte shared L2 caches; 16 MSHRs per bank
  o DRAM model based on Micron DDR2-800

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

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

Thomas Moscibroda, Microsoft Research
**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

• 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%)

See paper for details…

Thomas Moscibroda, Microsoft Research
Evaluation – Further Results

- 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!

*See paper for details...*
Evaluation – Aggregate Results

- Aggregate results over all 29 applications

<table>
<thead>
<tr>
<th>Sparse Network</th>
<th>Perfect L2</th>
<th>Realistic L2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Worst-Case</td>
</tr>
<tr>
<td>Δ Network Energy</td>
<td>-39.4%</td>
<td>-28.1%</td>
</tr>
<tr>
<td>Δ System Performance</td>
<td>-0.5%</td>
<td>-3.2%</td>
</tr>
</tbody>
</table>

Thomas Moscibroda, Microsoft Research
Evaluation – Aggregate Results

- Aggregate results over all 29 applications

<table>
<thead>
<tr>
<th>Sparse Network</th>
<th>Perfect L2</th>
<th>Realistic L2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Worst-Case</td>
</tr>
<tr>
<td>Δ Network Energy</td>
<td>-39.4%</td>
<td>-28.1%</td>
</tr>
<tr>
<td>Δ System Performance</td>
<td>-0.5%</td>
<td>-3.2%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dense Network</th>
<th>Perfect L2</th>
<th>Realistic L2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Worst-Case</td>
</tr>
<tr>
<td>Δ Network Energy</td>
<td>-32.8%</td>
<td>-14.0%</td>
</tr>
<tr>
<td>Δ System Performance</td>
<td>-3.6%</td>
<td>-17.1%</td>
</tr>
</tbody>
</table>
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…
CHIPPER: A Low-complexity Bufferless Deflection Router

Chris Fallin
Chris Craik
Onur Mutlu
Motivation

- Recent work has proposed **bufferless deflection routing** (BLESS [Moscibroda, ISCA 2009])
  - **Energy savings**: ~40% in total NoC energy
  - **Area reduction**: ~40% in total NoC area
  - **Minimal performance loss**: ~4% on average

- Unfortunately: unaddressed complexities in router
  - long critical path, large reassembly buffers

- **Goal**: obtain these benefits while simplifying the router in order to **make bufferless NoCs practical.**
Problems that Bufferless Routers Must Solve

1. Must provide **livelock freedom**

   ➞ A packet should not be deflected forever

2. Must **reassemble packets** upon arrival

**Flit**: atomic routing unit

**Packet**: one or multiple flits

0 1 2 3
A Bufferless Router: A High-Level View

Problem 1: Livelock Freedom

Problem 2: Packet Reassembly

Deflection Routing Logic

Crossbar

Reassembly Buffers

Eject
Complexity in Bufferless Deflection Routers

1. Must provide livelock freedom

   Flits are sorted by age, then assigned in age order to output ports

   ➔ 43% longer critical path than buffered router

2. Must reassemble packets upon arrival

   Reassembly buffers must be sized for worst case

   ➔ 4KB per node
   (8x8, 64-byte cache block)
Problem 1: Livelock Freedom

SAFARI
Livellok Freedom in Previous Work

- What stops a flit from deflecting forever?
- All flits are **timestamped**
- **Oldest flits** are assigned their desired ports
- Total order among flits

```
< < < < < <
Flit age forms total order
```

**Guaranteed progress!**

- But what is the **cost** of this?
Age-Based Priorities are Expensive: Sorting

- Router must sort flits by age: long-latency sort network
  - Three comparator stages for 4 flits
Age-Based Priorities Are Expensive: Allocation

- After sorting, flits assigned to output ports in priority order.
- Port assignment of younger flits depends on that of older flits.
  - **Sequential dependence** in the port allocator.

**East?**

1. **East?**
   - Age-Ordered Flits: 1
   - Grant: Flit 1 → East
   - {N, S, W}

2. **East?**
   - Age-Ordered Flits: 2
   - Deflect: Flit 2 → North
   - {S, W}

3. **South?**
   - Age-Ordered Flits: 3
   - Grant: Flit 3 → South
   - {W}

4. **South?**
   - Age-Ordered Flits: 4
   - Deflect: Flit 4 → West

**SAFARI**
Age-Based Priorities Are Expensive

- Overall, deflection routing logic based on Oldest-First has a **43% longer critical path** than a buffered router.

- Question: is there a cheaper way to route while guaranteeing livelock-freedom?
Solution: Golden Packet for Livelock Freedom

- What is *really necessary* for livelock freedom?

  **Key Insight:** No total order. It is enough to:
  1. Pick one flit to prioritize until arrival
  2. Ensure any flit is *eventually* picked

New traffic is lowest-priority

Guaranteed progress!

“Golden Flit”

Flit age forms total order; partial ordering is sufficient!
What Does Golden Flit Routing Require?

- Only **need** to properly route the Golden Flit

- **First Insight:** no need for full sort

- **Second Insight:** no need for sequential allocation
Golden Flit Routing With Two Inputs

- Let’s route the Golden Flit in a two-input router first

- **Step 1**: pick a “winning” flit: Golden Flit, else random

- **Step 2**: steer the winning flit to its desired output and deflect other flit

  ➔ Golden Flit always routes toward destination
Golden Flit Routing with Four Inputs

- Each block makes decisions **independently**!
- Deflection is a distributed decision

![Diagram showing Golden Flit Routing with Four Inputs](image-url)
Permutation Network Operation

wins → swap!

wins → no swap!

wins → no swap!

wins → no swap!

deflected

Golden:
CHIPPER: **Cheap Interconnect Partially-Permuting Router**

![Diagram showing Inject/Eject and Miss Buffers (MSHRs)]
Methodology

- **Multiprogrammed** workloads: CPU2006, server, desktop
  - 8x8 (64 cores), 39 homogeneous and 10 mixed sets

- **Multithreaded** workloads: SPLASH-2, 16 threads
  - 4x4 (16 cores), 5 applications

- **System configuration**
  - **Buffered** baseline: 2-cycle router, 4 VCs/channel, 8 flits/VC
  - **Bufferless** baseline: 2-cycle latency, FLIT-BLESS

  - Instruction-trace driven, closed-loop, 128-entry OoO window
  - 64KB L1, **perfect L2** (stresses interconnect), XOR mapping
Methodology

- **Hardware modeling**
  - Verilog models for CHIPPER, BLESS, buffered logic
    - Synthesized with commercial 65nm library
  - ORION for crossbar, buffers and links

- **Power**
  - Static and dynamic power from hardware models
  - Based on event counts in cycle-accurate simulations
Results: Performance Degradation

Minimal loss for low-to-medium-intensity workloads

C Minimal loss for low-to-medium-intensity workloads

5.6 %

49.8 %
Results: Power Reduction

**Multiprogrammed (subset of 49 total)**

- **Buffered**
- **BLESS**
- **CHIPPER**

**Multithreaded**

- **54.9%**
- **73.4%**

- **Removing buffers ➔ majority of power savings**
- **Slight savings from BLESS to CHIPPER**
CHIPPER maintains area savings of BLESS

Critical path becomes competitive to buffered
Conclusions

- Two key issues in bufferless deflection routing
  - livelock freedom and packet reassembly

- Bufferless deflection routers were high-complexity and impractical
  - Oldest-first prioritization → long critical path in router
  - No end-to-end flow control for reassembly → prone to deadlock with reasonably-sized reassembly buffers

- CHIPPER is a new, practical bufferless deflection router
  - Golden packet prioritization → short critical path in router
  - Retransmit-once protocol → deadlock-free packet reassembly
  - Cache miss buffers as reassembly buffers → truly bufferless network

- CHIPPER frequency comparable to buffered routers at much lower area and power cost, and minimal performance loss
MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect

Chris Fallin, Greg Nazario, Xiangyao Yu*, Kevin Chang, Rachata Ausavarungnirun, Onur Mutlu

Carnegie Mellon University
*CMU and Tsinghua University
Bufferless Deflection Routing

- **Key idea**: Packets are never buffered in the network. When two packets contend for the same link, one is *deflected*.

- Removing **buffers** yields significant benefits
  - Reduces **power** (CHIPPER: reduces NoC power by 55%)
  - Reduces **die area** (CHIPPER: reduces NoC area by 36%)

- But, at **high network utilization** (load), bufferless deflection routing causes **unnecessary link & router traversals**
  - Reduces network throughput and application performance
  - Increases dynamic power

- **Goal**: Improve high-load performance of low-cost deflection networks by reducing the deflection rate.
Outline: This Talk

- Motivation

- **Background**: Bufferless Deflection Routing

- **MinBD**: Reducing Deflections
  - Addressing Link Contention
  - Addressing the Ejection Bottleneck
  - Improving Deflection Arbitration

- Results

- Conclusions
Outline: This Talk

- Motivation

- **Background**: Bufferless Deflection Routing

- **MinBD**: Reducing Deflections
  - Addressing Link Contention
  - Addressing the Ejection Bottleneck
  - Improving Deflection Arbitration

- Results

- Conclusions
Issues in Bufferless Deflection Routing

- **Correctness**: Deliver all packets without *livelock*
  - **CHIPPER**: Golden Packet
    - Globally prioritize one packet until delivered
  - **CHIPPER**: Retransmit-Once

- **Correctness**: Reassemble packets without *deadlock*
  - CHIPPER: Retransmit-Once

- **Performance**: Avoid performance degradation at *high load*
  - **MinBD**

---

1 Fallin et al., “CHIPPER: A Low-complexity Bufferless Deflection Router”, HPCA
Key Performance Issues

1. **Link contention**: no buffers to hold traffic → any link contention causes a deflection
   → use side buffers

2. **Ejection bottleneck**: only one flit can eject per router per cycle → simultaneous arrival causes deflection
   → eject up to 2 flits/cycle

3. **Deflection arbitration**: practical (fast) deflection arbiters deflect unnecessarily
   → new priority scheme (silver flit)
Outline: This Talk

- Motivation

- **Background**: Bufferless Deflection Routing

- **MinBD**: Reducing Deflections
  - Addressing Link Contention
  - Addressing the Ejection Bottleneck
  - Improving Deflection Arbitration

- Results

- Conclusions
Outline: This Talk

- Motivation

- **Background:** Bufferless Deflection Routing

- **MinBD:** Reducing Deflections
  - Addressing Link Contention
  - Addressing the Ejection Bottleneck
  - Improving Deflection Arbitration

- Results
- Conclusions
Addressing Link Contention

- **Problem 1**: Any link contention causes a deflection

- **Buffering** a flit can avoid deflection on contention

- But, **input buffers** are expensive:
  - All flits are buffered on every hop → *high dynamic energy*
  - Large buffers necessary → *high static energy* and *large area*

- **Key Idea 1**: add a **small buffer** to a bufferless deflection router to buffer **only** flits that **would have been deflected**
How to Buffer Deflected Flits

Baseline Router

Eject Inject

Destination Destination

DEFLECTED

Baseline Router

How to Buffer Deflected Flits

Step 1. Remove up to one deflected flit per cycle from the outputs.

Step 2. Buffer this flit in a small FIFO "side buffer."

Step 3. Re-inject this flit into pipeline when a slot is available.
Why Could A Side Buffer Work Well?

- Buffer some flits and deflect other flits at per-flit level

  - Relative to bufferless routers, deflection rate reduces (need not deflect all contending flits)
    - 4-flit buffer reduces deflection rate by 39%

  - Relative to buffered routers, buffer is more efficiently used (need not buffer all flits)
    - similar performance with 25% of buffer space
Outline: This Talk

- Motivation

- **Background**: Bufferless Deflection Routing

- **MinBD**: Reducing Deflections
  - Addressing Link Contention
  - Addressing the Ejection Bottleneck
  - Improving Deflection Arbitration

- Results

- Conclusions


Addressing the Ejection Bottleneck

- **Problem 2**: Flits deflect unnecessarily because only one flit can *eject* per router per cycle.

- In 20% of all ejections, $\geq 2$ flits could have ejected
  $\rightarrow$ all but one flit must *deflect* and try again
  $\rightarrow$ these deflected flits cause additional contention

- Ejection width of 2 flits/cycle reduces *deflection rate* 21%

- **Key idea 2**: Reduce deflections due to a single-flit ejection port by allowing *two flits* to eject per cycle.
Addressing the Ejection Bottleneck

Single-Width Ejection

DEFLECTED
Addressing the Ejection Bottleneck

For fair comparison, **baseline routers** have dual-width ejection for perf. (not power/area)

Eject  Inject

Dual-Width Ejection
Outline: This Talk

- Motivation

- Background: Bufferless Deflection Routing

- **MinBD:** Reducing Deflections
  - Addressing Link Contention
  - Addressing the Ejection Bottleneck
  - Improving Deflection Arbitration

- Results
- Conclusions
Improving Deflection Arbitration

- **Problem 3**: Deflections occur unnecessarily because fast arbiters must use simple priority schemes

- Age-based priorities (several past works): full priority order gives fewer deflections, but requires slow arbiters

- State-of-the-art deflection arbitration (Golden Packet & two-stage permutation network)
  - Prioritize one packet globally (ensure forward progress)
  - Arbitrate other flits randomly (fast critical path)

- Random common case leads to uncoordinated arbitration
Fast Deflection Routing Implementation

- Let’s route in a two-input router first:

  - **Step 1**: pick a “winning” flit (Golden Packet, else random)
  - **Step 2**: steer the winning flit to its desired output and deflect other flit

  ➔ Highest-priority flit always routes to destination
Fast Deflection Routing with Four Inputs

- Each block makes decisions **independently**
- Deflection is a distributed decision
How does lack of coordination cause unnecessary deflections?
1. No flit is golden (pseudorandom arbitration)
2. Red flit wins at first stage
3. Green flit loses at first stage (must be deflected now)
4. Red flit loses at second stage; Red and Green are deflected

All flits have equal priority

unnecessary deflection!
Improving Deflection Arbitration

Key idea 3: Add a priority level and prioritize one flit to ensure at least one flit is not deflected in each cycle.

- Highest priority: one Golden Packet in network
  - Chosen in static round-robin schedule
  - Ensures correctness

- Next-highest priority: one silver flit per router per cycle
  - Chosen pseudo-randomly & local to one router
  - Enhances performance
Adding A Silver Flit

- Randomly picking a silver flit ensures **one flit is not deflected**
  1. No flit is golden but **Red** flit is silver
  2. **Red** flit wins at first stage (silver)
  3. **Green** flit is deflected at first stage
  4. **Red** flit wins at second stage (silver); not deflected
Minimally-Buffered Deflection Router

Problem 1: Link Contention
Solution 1: Side Buffer

Problem 2: Ejection Bottleneck
Solution 2: Dual-Width Ejection

Problem 3: Unnecessary Deflections
Solution 3: Two-level priority scheme
Outline: This Talk

- **Motivation**

- **Background**: Bufferless Deflection Routing

- **MinBD**: Reducing Deflections
  - Addressing Link Contention
  - Addressing the Ejection Bottleneck
  - Improving Deflection Arbitration
Outline: This Talk

- Motivation

- **Background**: Bufferless Deflection Routing

- **MinBD**: Reducing Deflections
  - Addressing Link Contention
  - Addressing the Ejection Bottleneck
  - Improving Deflection Arbitration

- Results

- Conclusions
Methodology: Simulated System

- **Chip Multiprocessor Simulation**
  - **64-core** and **16-core** models
  - **Closed-loop** core/cache/NoC cycle-level model
  - Directory cache coherence protocol (SGI Origin-based)
  - 64KB L1, perfect L2 (stresses interconnect), XOR-mapping
  - Performance metric: **Weighted Speedup**
    (similar conclusions from network-level latency)
  - Workloads: multiprogrammed SPEC CPU2006
    - 75 randomly-chosen workloads
    - Binned into network-load categories by average injection rate
Methodology: Routers and Network

- **Input-buffered** virtual-channel router
  - 8 VCs, 8 flits/VC [*Buffered*(8,8)]: large buffered router
  - 4 VCs, 4 flits/VC [*Buffered*(4,4)]: typical buffered router
  - 4 VCs, 1 flit/VC [*Buffered*(4,1)]: smallest deadlock-free router
  - All power-of-2 buffer sizes up to (8, 8) for perf/power sweep

- **Bufferless deflection** router: **CHIPPER**¹

- **Bufferless-buffered hybrid** router: **AFC**²
  - Has input buffers and deflection routing logic
  - Performs coarse-grained (multi-cycle) mode switching

- **Common parameters**
  - 2-cycle router latency, 1-cycle link latency
  - 2D-mesh topology (16-node: 4x4; 64-node: 8x8)
  - Dual ejection assumed for baseline routers (for perf. only)

Methodology: Power, Die Area, Crit. Path

- **Hardware modeling**
  - Verilog models for CHIPPER, MinBD, buffered control logic
    - Synthesized with commercial 65nm library
  - ORION 2.0 for datapath: crossbar, muxes, buffers and links

- **Power**
  - Static and dynamic power from hardware models
  - Based on event counts in cycle-accurate simulations
  - Broken down into buffer, link, other
1. All mechanisms individually reduce deflections.

2. Side buffer alone is not sufficient for performance (ejection bottleneck remains).

3. Overall, **5.8%** over baseline, **2.7%** over dual-eject by reducing deflections **64%** / **54%**.
Overall Performance Results

- Similar perf. to Buffered (4,1) @ 25% of buffering space
- Within 2.7% of Buffered (4,4) (8.3% at high load)
Overall Power Results

- Dynamic power increases with deflection routing
- Buffers are significant fraction of power in baseline routers
- Dynamic power reduces in MinBD relative to CHIPPER
- Dynamic power reduces in MinBD relative to CHIPPER
Performance-Power Spectrum

• Most **energy-efficient** (perf/watt) of any evaluated network router design
Die Area and Critical Path

- Only **3%** area increase over CHIPPER (4-flit buffer)
- Increases by **7%** over CHIPPER, **8%** over Buffered (4,4)
Conclusions

- Bufferless deflection routing offers reduced power & area
- But, high deflection rate hurts performance at high load

- **MinBD** (Minimally-Buffered Deflection Router) introduces:
  - Side buffer to hold only flits that would have been deflected
  - Dual-width ejection to address ejection bottleneck
  - Two-level prioritization to avoid unnecessary deflections

- MinBD yields reduced power (31%) & reduced area (36%) relative to buffered routers
- MinBD yields improved performance (8.1% at high load) relative to bufferless routers → closes half of perf. gap

- MinBD has the best energy efficiency of all evaluated designs with competitive performance