Computer Architecture
Lecture 22: On-Chip Networks

Prof. Onur Mutlu
ETH Zürich
Fall 2021
10 December 2021
Buffered Flow Control
Review: Buffered Flow Control

Store and Forward

Wormhole

Any other issues?

Head-of-Line Blocking

Use Virtual Channels

Buffer full: blue cannot proceed

Blocked by other packets
Communicating Buffer Availability

- **Credit-based flow control**
  - Upstream knows how many buffers are downstream
  - Downstream passes back credits to upstream
  - Significant upstream signaling (esp. for small flits)

- **On/Off (XON/XOFF) flow control**
  - Downstream has on/off signal to upstream

- **ACK/NACK flow control**
  - Upstream optimistically sends downstream
  - Buffer cannot be deallocated until ACK/NACK received
  - Inefficiently utilizes buffer space
**Credit-based Flow Control**

- **Round-trip credit delay:**
  - Time between when buffer empties and when next flit can be processed from that buffer entry
- Significant throughput degradation if there are few buffers
- Important to size buffers to tolerate credit turn-around
On/Off (XON/XOFF) Flow Control

- Downstream has on/off signal to upstream

F_{off} set to prevent flits arriving before t4 from overflowing

F_{on} set so that Node 2 does not run out of flits between t5 and t8
Interconnection Network Performance
Interconnection Network Performance

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

Injection rate into network

Throughput given by flow control

Throughput given by routing

Throughput 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
  - The distance between two points measured along axes at right angles.
- \( v = \) propagation velocity
- \( L = \) packet size
- \( b = \) channel bandwidth
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
- \( v \) = propagation velocity
- \( L \) = packet size
- \( b \) = channel bandwidth
- \( H \) = hops
- \( T_{\text{router}} \) = router latency
- \( T_c \) = latency due to contention
Latency and Throughput Curve

- **Latency (cycles)**
- **Injected load (fraction of capacity)**

**Graph Details:**
- **Ideal** line remains constant.
- **On-chip Network** line shows an exponential increase in latency as the injected load increases.
Network Performance Metrics

- Packet latency
- Round trip latency
- Saturation throughput
- Application-level performance: system performance
  - Affected by interference among threads/applications
Buffering and Flow Control in On-Chip Networks
On-Chip Networks

- Connect cores, caches, memory controllers, etc
  - Buses and crossbars are not scalable
- Packet switched
- 2D mesh: Most commonly used topology
- Primarily serve cache misses and memory requests

Router

Processing Element
(Cores, L2 Banks, Memory Controllers, etc)
On-chip Networks

- Router
- PE (Processing Element: Cores, L2 Banks, Memory Controllers etc.)
- VC Identifier
- Input Port with Buffers
- Control Logic
  - Routing Unit (RQ)
  - VC Allocator (VA)
  - Switch Allocator (SA)
- Crossbar (5x5)
On-Chip vs. Off-Chip Interconnects

- **On-chip advantages**
  - Low latency between cores
  - No pin constraints
  - Rich wiring resources
  - Very high bandwidth
  - Simpler coordination

- **On-chip constraints/disadvantages**
  - 2D substrate limits implementable topologies
  - Energy/power consumption a key concern
  - Complex algorithms undesirable
  - Logic area constrain use of wiring resources
On-Chip vs. Off-Chip Interconnects (II)

- **Cost**
  - Off-chip: Channels, pins, connectors, cables
  - On-chip: Cost is storage and switches (wires are plentiful)
  - Leads to networks with many wide channels, few buffers

- **Channel characteristics**
  - On chip short distance $\rightarrow$ low latency
  - On chip RC lines $\rightarrow$ need repeaters every 1-2mm
    - Can put logic in repeaters

- **Workloads**
  - Multi-core cache traffic vs. supercomputer interconnect traffic
On-Chip vs. Off-Chip Tradeoffs

George Nychis, Chris Fallin, Thomas Moscibroda, Onur Mutlu, and Srinivasan Seshan,
"On-Chip Networks from a Networking Perspective: Congestion and Scalability in Many-core Interconnects"
Proceedings of the 2012 ACM SIGCOMM Conference (SIGCOMM), Helsinki, Finland, August 2012. Slides (pptx)

On-Chip Networks from a Networking Perspective: Congestion and Scalability in Many-Core Interconnects

George Nychis†, Chris Fallin†, Thomas Moscibroda§, Onur Mutlu†, Srinivasan Seshan†
† Carnegie Mellon University
{gnychis, cfallin, onur, srini}@cmu.edu
§ Microsoft Research Asia
moscitho@microsoft.com
Buffers are necessary for high network throughput

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

- Buffers consume significant energy/power
  - Dynamic energy when 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!
**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]

![Diagram of BLESS routing](image)
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
Each flit is routed independently.

**Oldest-first arbitration** (other policies evaluated in paper)

1. Oldest-first ranking
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
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
• Oldest-first arbitration complex
• QoS becomes difficult

Impact on energy...?
Evaluation – Synthetic Traces

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

• 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!

• Future research:
  • Support for quality of service, different traffic classes, energy-management, etc…
Bufferless Deflection Routing in NoCs


A Case for Bufferless Routing in On-Chip Networks

Thomas Moscibroda  
Microsoft Research  
moscitho@microsoft.com

Onur Mutlu  
Carnegie Mellon University  
onur@cmu.edu
Issues In Bufferless Deflection Routing

- Livelock
- Resulting Router Complexity
- Performance & Congestion at High Loads
- Quality of Service and Fairness

Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, and Onur Mutlu,
"Bufferless and Minimally-Buffered Deflection Routing"
Low-Complexity Bufferless Routing


CHIPPER: A Low-complexity Bufferless Deflection Router

Chris Fallin
cfallin@cmu.edu

Chris Craik
craik@cmu.edu

Onur Mutlu
onur@cmu.edu

Computer Architecture Lab (CALCM)
Carnegie Mellon University
CHIPPER: A Low-complexity Bufferless Deflection Router

Chris Fallin, Chris Craik, and Onur Mutlu,
"CHIPPER: A Low-Complexity Bufferless Deflection Router"
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

Reassembly Buffers

Eject

Crossbar

Deflection Routing Logic

Router

Local Node
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
Livelock 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

New traffic is lowest priority

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

<table>
<thead>
<tr>
<th>Flit</th>
<th>Direction</th>
<th>Action</th>
<th>Port Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>East</td>
<td>GRANT</td>
<td>{N,S,W}</td>
</tr>
<tr>
<td>2</td>
<td>East</td>
<td>DEFLECT</td>
<td>{S,W}</td>
</tr>
<tr>
<td>3</td>
<td>South</td>
<td>GRANT</td>
<td>{W}</td>
</tr>
<tr>
<td>4</td>
<td>South</td>
<td>DEFLECT</td>
<td>{W}</td>
</tr>
</tbody>
</table>

Age-Ordered Flits
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!

Flit age forms total order
Partial ordering is sufficient!

“Golden Flit”
Which Packet is Golden?

- We select the **Golden Packet** so that:
  1. a given packet stays golden long enough to ensure arrival → maximum no-contention latency
  2. the selection rotates through all possible packet IDs → static rotation schedule for simplicity

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Source</th>
<th>Dest</th>
<th>Request ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>Src 3</td>
<td></td>
<td>Req 1</td>
</tr>
</tbody>
</table>

Packet Header:
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

![Diagram showing no need for Priority Sort and Port Allocator](image)
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 is always routed toward its destination
Golden Flit Routing with Four Inputs

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

![Diagram of routing with four inputs](image)
Permutation Network Operation

- **Golden:**
  - **wins → swap!**

- **Port Allocator:**
  - **Priority Sort**

Diagram:
- Connections marked with red and green arrows indicating swaps or no swaps.
- Directions: N (North), E (East), S (South), W (West).
Permutation Network-based Pipeline
Problem 2: Packet Reassembly
Reassembly Buffers are Large

- **Worst case**: every node sends a packet to one receiver
- Why can't we make reassembly buffers smaller?

\[
\text{N sending nodes} \quad \xrightarrow{\text{one packet in flight per node}} \quad \text{O(N) space!}
\]
Small Reassembly Buffers Cause Deadlock

What happens when reassembly buffer is too small?

Many Senders

Network full

Remaining flits must be injected for forward progress

cannot inject new traffic

One Receiver

cannot eject: reassembly buffer full
Reserve Space to Avoid Deadlock?

- What if every sender asks permission from the receiver before it sends?

  ➔ adds additional delay to every request

1. Reserve Slot
2. ACK
3. Send Packet

Sender

Reserve Slot?

Receiver
reassemble buffers

Reserved

Sender

Reserve Slot?

Receiver
reassemble buffers

Reserved
Escaping Deadlock with Retransmissions

- Sender is optimistic instead: assume buffer is free
  - If not, receiver drops and NACKs; sender retransmits

→ no additional delay in best case
→ transmit buffering overhead for all packets
→ potentially many retransmits

1. Send (2 flits)
2. Drop, NACK
3. Other packet completes retransmit packet
4. Other packet completes
5. ACK
6. Sender frees data
Solution: Retransmitting Only Once

- **Key Idea:** Retransmit only *when space becomes available.*
  - Receiver *drops packet* if full; *notes* which packet it drops
  - When space frees up, receiver *reserves space* so retransmit is successful
  - Receiver notifies sender to retransmit
Use MSHRs as Reassembly Buffers

Outstanding Cache Misses

Miss Status Handling Register (MSHR)

Pending | Block 0x3C |  
--- | --- | ---
Status | Address | Data Buffer

Reassembly buffering for “free”

⇒ A truly bufferless NoC!
Using MSHRs as Reassembly Buffers

Using miss buffers for reassembly makes this a truly bufferless network.
CHIPPER: Cheap Interconnect Partially-Permuting Router

Baseline Bufferless Deflection Router

Long critical path:
1. Sort by age
2. Allocate ports sequentially

→ Golden Packet
→ Permutation Network

Large buffers for worst case

→ Retransmit-Once
→ Cache miss buffers

Eject

Reassembly Buffers
CHIPPER: Cheap Interconnect Partially-Permuting Router

Inject/Eject

Miss Buffers (MSHRs)

Inject Eject
EVALUATION
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

SAFARI
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**
Results: Area and Critical Path Reduction

- **Normalized Router Area**
  - CHIPPER maintains area savings of BLESS
  - Critical path becomes competitive to buffered

- **Normalized Critical Path**
  - -36.2% (CHIPPER)
  - +1.1% (BUFFERED)
  - -29.1% (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
More on CHIPPER

- Chris Fallin, Chris Craik, and Onur Mutlu, "CHIPPER: A Low-Complexity Bufferless Deflection Router"

CHIPPER: A Low-complexity Bufferless Deflection Router

Chris Fallin  
cfallin@cmu.edu

Chris Craik  
craik@cmu.edu

Onur Mutlu  
onur@cmu.edu

Computer Architecture Lab (CALCM)  
Carnegie Mellon University
Minimally-Buffered Deflection Routing

- 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
Minimally-Buffered Deflection Routing


MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect

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

Carnegie Mellon University
{cfallin,gnazario,kevincha,rachata,onur}@cmu.edu

†Tsinghua University & Carnegie Mellon University
yxythu@gmail.com
MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect

Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, and Onur Mutlu,
"MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect"
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\(^1\): Golden Packet
  - Globally prioritize one packet until delivered

- **Correctness**: Reassemble packets without **deadlock**
  - CHIPPER\(^1\): 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 $\rightarrow$ any link contention causes a deflection
   $\rightarrow$ use side buffers

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

3. **Deflection arbitration**: practical (fast) deflection arbiters deflect unnecessarily
   $\rightarrow$ 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

DEFLECTED

Baseline Router

Destination

Destination

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

SAFARI
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
Unnecessary Deflections in Fast Arbiters

- 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

```
[Diagram: A network diagram showing the path of flits from source to destination with unnecessary deflections highlighted.]
```

all flits have equal priority

unnecessary deflection!
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

- **Bufferless-buffered hybrid** router: AFC
  - 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
Reduced Deflections & Improved Perf.

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

3. Overall, \textbf{5.8\%} over baseline, \textbf{2.7\%} over dual-eject by reducing deflections \textbf{64\%} / \textbf{54\%}

<table>
<thead>
<tr>
<th>Deflection Rate</th>
<th>28%</th>
<th>17%</th>
<th>22%</th>
<th>27%</th>
<th>11%</th>
<th>10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E (Side Buffer)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D (Dual-Eject)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S (Silver Flits)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B+D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B+S+D (MinBD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overall Performance Results

- **Weighted Speedup**
  - **Injection Rate**
  - **Buffered (8,8)**
  - **Buffered (4,4)**
  - **Buffered (4,1)**
  - **CHIPPER**
  - **AFC (4,4)**
  - **MinBD-4**

- **Improves 2.7%** over CHIPPER (8.1% at high load)
- **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.
- There are significant reductions in power in Baseline routers.
- 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
Minimally-Buffered Deflection Routing

- Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, and Onur Mutlu,
"MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect"
One of the five papers nominated for the Best Paper Award by the Program Committee.

MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect

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

Carnegie Mellon University
{cfallin,gnazario,kevincha,rachata,onur}@cmu.edu

†Tsinghua University & Carnegie Mellon University
yxythu@gmail.com
HAT: Heterogeneous Adaptive Throttling for On-Chip Networks

Kevin Chang, Rachata Ausavarungnirun, Chris Fallin, and Onur Mutlu,
"HAT: Heterogeneous Adaptive Throttling for On-Chip Networks"
Executive Summary

• **Problem:** Packets contend in on-chip networks (NoCs), causing congestion, thus reducing performance

• **Observations:**
  1) Some applications are more sensitive to network latency than others
  2) Applications must be throttled differently to achieve peak performance

• **Key Idea:** Heterogeneous Adaptive Throttling (HAT)
  1) Application-aware source throttling
  2) Network-load-aware throttling rate adjustment

• **Result:** Improves performance and energy efficiency over state-of-the-art source throttling policies
Source Throttling in Bufferless NoCs


HAT: Heterogeneous Adaptive Throttling for On-Chip Networks

Kevin Kai-Wei Chang, Rachata Ausavarungnirun, Chris Fallin, Onur Mutlu
Carnegie Mellon University
{kevincha,rachata,cfallin,onur}@cmu.edu
“Bufferless” Hierarchical Rings


- Describes the design and implementation of a mostly-bufferless hierarchical ring

Design and Evaluation of Hierarchical Rings with Deflection Routing

Rachata Ausavarungrunirun  Chris Fallin  Xiangyao Yu†  Kevin Kai-Wei Chang  Greg Nazario  Reetuparna Das‡  Gabriel H. Loh‡  Onur Mutlu

Carnegie Mellon University  §University of Michigan  †MIT  ‡Advanced Micro Devices, Inc.
“Bufferless” Hierarchical Rings (II)


Achieving both High Energy Efficiency and High Performance in On-Chip Communication using Hierarchical Rings with Deflection Routing

Rachata Ausavarungnirun  Chris Fallin  Xiangyao Yu†  Kevin Kai-Wei Chang
Greg Nazario  Reetuparna Das§  Gabriel H. Loh‡  Onur Mutlu
Carnegie Mellon University  §University of Michigan  †MIT  ‡AMD
Chapter 1
Bufferless and Minimally-Buffered Deflection Routing

Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, Onur Mutlu
Summary of Eight Years of Research

Energy-Efficient Deflection-based On-chip Networks: Topology, Routing, Flow Control

Rachata Ausavarungnirun\textsuperscript{b}, Onur Mutlu\textsuperscript{a}

\textit{SAFARI Research Group}

\textsuperscript{a}ETH Zürich
\textsuperscript{b}King Mongkut's University of Technology North Bangkok

\section*{Abstract}

As the number of cores scales to tens and hundreds, the energy consumption of routers across various types of on-chip networks in chip multiprocessors (CMPs) increases significantly. A major source of this energy consumption comes from the input buffers inside Network-on-Chip (NoC) routers, which are traditionally designed to maximize performance. To mitigate this high energy cost, many works propose bufferless router designs that utilize deflection routing to resolve port contention. While this approach is able to maintain high performance relative to its buffered counterparts at low network traffic, the bufferless router design suffers performance degradation under high network load.

In order to maintain high performance and energy efficiency under both low and high network loads, this chapter discusses critical drawbacks of traditional bufferless designs and describes recent research works focusing on two major modifications to improve the overall performance of the traditional bufferless network-on-chip design. The first modification is a minimally-buffered design that introduces limited buffering inside critical parts of the on-chip network in order to reduce the number of deflections. The second modification is a hierarchical bufferless interconnect design that aims to further improve performance by limiting the number of hops each packet needs to travel while in the network. In both approaches, we discuss design tradeoffs and provide evaluation results based on common CMP configurations with various network topologies to show the effectiveness of each proposal.

\textit{Keywords}: network-on-chip, deflection routing, topology, bufferless router, energy efficiency, high-performance computing, computer architecture, emerging technologies, latency, low-latency computing

\url{https://arxiv.org/pdf/2112.02516.pdf}
More Readings

- Studies of congestion and congestion control in on-chip vs. internet-like networks


- George Nychis, Chris Fallin, Thomas Moscibroda, and Onur Mutlu, "Next Generation On-Chip Networks: What Kind of Congestion Control Do We Need?". Proceedings of the 9th ACM Workshop on Hot Topics in Networks (HOTNETS), Monterey, CA, October 2010. Slides (ppt) (key)
On-Chip vs. Off-Chip Tradeoffs


On-Chip Networks from a Networking Perspective: Congestion and Scalability in Many-Core Interconnects

George Nychis†, Chris Fallin†, Thomas Moscibroda§, Onur Mutlu†, Srinivasan Seshan†

† Carnegie Mellon University
§ Microsoft Research Asia
{gnychis,cfallin,onur,srini}@cmu.edu
moscitho@microsoft.com
Packet Scheduling
Packet Scheduling

- **Which packet to choose for a given output port?**
  - Router needs to prioritize between competing flits
  - Which input port?
  - Which virtual channel?
  - Which application’s packet?

- **Common strategies**
  - Round robin across virtual channels
  - Oldest packet first (or an approximation)
  - Prioritize some virtual channels over others

- **Better policies in a multi-core environment**
  - Use application characteristics
Application-Aware Packet Scheduling

The Problem: Packet Scheduling

Network-on-Chip is a critical resource shared by multiple applications
The Problem: Packet Scheduling

Routers

Processing Element
(Cores, L2 Banks, Memory Controllers etc)

Input Port with Buffers

VC Identifier
VC 0
VC 1
VC 2

From East
From West
From North
From South
From PE

Control Logic
Routing Unit (RC)
VC Allocator (VA)
Switch Allocator (SA)

Crossbar (5 x 5)

Crossbar

To East
To West
To North
To South
To PE
The Problem: Packet Scheduling
The Problem: Packet Scheduling

- From East: VC0, VC1, VC2
- From West: VC0, VC1, VC2
- From North: VC0, VC1, VC2
- From South: VC0, VC1, VC2
- From PE: VC0, VC1, VC2

Routing Unit (RC)
VC Allocator (VA)
Switch Allocator (SA)
The Problem: Packet Scheduling

Which packet to choose?
The Problem: Packet Scheduling

- Existing scheduling policies
  - Round Robin
  - Age

- Problem 1: Local to a router
  - Lead to contradictory decision making between routers: packets from one application may be prioritized at one router, to be delayed at next.

- Problem 2: Application oblivious
  - Treat all applications packets equally
  - But applications are heterogeneous

- Solution: Application-aware global scheduling policies.
STC Scheduling Example

Batching interval length = 3 cycles

Ranking order =

Packet Injection Order at Processor
STC Scheduling Example

<table>
<thead>
<tr>
<th>STALL CYCLES</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>8.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Age</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>STC</th>
</tr>
</thead>
</table>
STC Scheduling Example

**STALL CYCLES**

<table>
<thead>
<tr>
<th></th>
<th>RR</th>
<th>Age</th>
<th>STC</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>8</td>
<td>6</td>
<td>11</td>
<td>8.3</td>
</tr>
<tr>
<td>Age</td>
<td>4</td>
<td>6</td>
<td>11</td>
<td>7.0</td>
</tr>
</tbody>
</table>

**Round Robin**

5 4 3 1 2 2 3 2

**Age**

4 3 5 4 6 7 8

**STC**

- Router: 5, 8, 4, 3, 7, 1, 6, 2, 3, 2
- Scheduler: 5, 4, 3, 1, 2, 2, 3, 2
- Time: 7, 6

**Diagram Description**

- STC Scheduling Example
- Router and Scheduler
- Round Robin and Age
- Table showing STALL CYCLES and Avg
STC Scheduling Example

Router

Scheduler

Round Robin

STC

STALL CYCLES | Avg
---|---
RR | 8 | 6 | 11 | 8.3
Age | 4 | 6 | 11 | 7.0
STC | 1 | 3 | 11 | 5.0
Application-Aware Prioritization in NoCs


Application-Aware Prioritization Mechanisms for On-Chip Networks

Reetuparna Das§ Onur Mutlu† Thomas Moscibroda‡ Chita R. Das§
§Pennsylvania State University {rdas,das}@cse.psu.edu †Carnegie Mellon University onur@cmu.edu ‡Microsoft Research moscitho@microsoft.com
Slack-Based Packet Scheduling


One of the 11 computer architecture papers of 2010 selected as Top Picks by IEEE Micro.

Aérgia: Exploiting Packet Latency Slack in On-Chip Networks

Reetuparna Das§  Onur Mutlu†  Thomas Moscibroda‡  Chita R. Das§
§Pennsylvania State University  †Carnegie Mellon University  ‡Microsoft Research
{rdas,das}@cse.psu.edu  onur@cmu.edu  moscitho@microsoft.com
Low-Cost QoS in On-Chip Networks (I)

- Boris Grot, Stephen W. Keckler, and Onur Mutlu, "Preemptive Virtual Clock: A Flexible, Efficient, and Cost-effective QoS Scheme for Networks-on-Chip"

Preemptive Virtual Clock: A Flexible, Efficient, and Cost-effective QoS Scheme for Networks-on-Chip

Boris Grot
Department of Computer Sciences
The University of Texas at Austin
{bgrot, skeckler}@cs.utexas.edu

Stephen W. Keckler

Onur Mutlu†
†Computer Architecture Laboratory (CALCM)
Carnegie Mellon University
onur@cmu.edu
Boris Grot, Joel Hestness, Stephen W. Keckler, and Onur Mutlu, "Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for Scalability and Service Guarantees"
Proceedings of the 38th International Symposium on Computer Architecture (ISCA), San Jose, CA, June 2011. Slides (pptx)
One of the 12 computer architecture papers of 2011 selected as Top Picks by IEEE Micro.

Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for Scalability and Service Guarantees

Boris Grot\textsuperscript{1} 
bgrot@cs.utexas.edu

Joel Hestness\textsuperscript{1} 
hestness@cs.utexas.edu

Stephen W. Keckler\textsuperscript{1,2} 
skeckler@nvidia.com

Onur Mutlu\textsuperscript{3} 
onur@cmu.edu

\textsuperscript{1}The University of Texas at Austin 
Austin, TX

\textsuperscript{2}NVIDIA 
Santa Clara, CA

\textsuperscript{3}Carnegie Mellon University 
Pittsburgh, PA
Low-Cost QoS in On-Chip Networks (III)


A QoS-Enabled On-Die Interconnect Fabric for Kilo-Node Chips

To meet rapidly growing performance demands and energy constraints, future chips will likely feature thousands of on-die resources. Existing network-on-chip solutions weren’t designed for scalability and will be unable to meet future interconnect demands. A hybrid network-on-chip architecture called Kilo-NOC co-optimizes topology, flow control, and quality of service to achieve significant gains in efficiency.
Boris Grot, Joel Hestness, Stephen W. Keckler, and Onur Mutlu,
"Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for
Scalability and Service Guarantees"
Proceedings of the 38th International Symposium on Computer
Architecture (ISCA), San Jose, CA, June 2011. Slides (pptx)
Motivation

- Extreme-scale chip-level integration
  - Cores
  - Cache banks
  - Accelerators
  - I/O logic
  - Network-on-chip (NOC)
- 10-100 cores today
- 1000+ agents in the near future
Kilo-NOC requirements

- High efficiency
  - Area
  - Energy
- Good performance
- Strong service guarantees (QoS)
Topology-Aware QoS

- Problem: QoS support in each router is expensive (in terms of buffering, arbitration, bookkeeping)

- Goal: Provide QoS guarantees at low area and power cost

- Idea:
  - Isolate shared resources in a region of the network, support QoS within that area
  - Design the topology so that applications can access the region without interference
Baseline QOS-enabled CMP

Multiple VMs sharing a die

- Shared resources (e.g., memory controllers)
- VM-private resources (cores, caches)
- QOS-enabled router
Conventional NOC QOS

Contention scenarios:

- **Shared resources**
  - memory access
- **Intra-VM traffic**
  - shared cache access
- **Inter-VM traffic**
  - VM page sharing
Conventional NOC QOS

Contention scenarios:
- Shared resources
  - memory access
- Intra-VM traffic
  - shared cache access
- Inter-VM traffic
  - VM page sharing

Network-wide guarantees *without* network-wide QOS support
Kilo-NOC QoS

- **Insight: leverage rich network connectivity**
  - Naturally reduce interference among flows
  - Limit the extent of hardware QOS support

- **Requires a low-diameter topology**
  - This work: Multidrop Express Channels (MECS)

---

*Grot et al., HPCA 2009*
- Dedicated, QOS-enabled regions
  - Rest of die: QOS-free
- Richly-connected topology
  - Traffic isolation
- Special routing rules
  - Manage interference
Topology-Aware QOS

- Dedicated, QOS-enabled regions
  - Rest of die: QOS-free
- Richly-connected topology
  - Traffic isolation
- Special routing rules
  - Manage interference
Toplogy-Aware QOS

- Dedicated, QOS-enabled regions
  - Rest of die: QOS-free
- Richly-connected topology
  - Traffic isolation
- Special routing rules
  - Manage interference
Topology-Aware QOS

- Dedicated, QOS-enabled regions
  - Rest of die: QOS-free
- Richly-connected topology
  - Traffic isolation
- Special routing rules
  - Manage interference
- **Topology-aware QOS support**
  - Limit QOS complexity to a fraction of the die

- **Optimized flow control**
  - Reduce buffer requirements in QOS-free regions

Kilo-NOC view
## Parameter | Value
---|---
Technology | 15 nm
Vdd | 0.7 V
System | 1024 tiles: 256 concentrated nodes (64 shared resources)

### Networks:

| MECS+PVC | VC flow control, QOS support (PVC) at each node |
| MECS+TAQ | VC flow control, QOS support only in shared regions |
| MECS+TAQ+EB | EB flow control outside of SRs, Separate Request and Reply networks |
| K-MECS | Proposed organization: TAQ + hybrid flow control |
Area comparison

Area (mm²)

<table>
<thead>
<tr>
<th></th>
<th>MECS+PVC</th>
<th>MECS+TAQ</th>
<th>MECS+TAQ+EB</th>
<th>K-MECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR Routers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Routers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Link EBs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Links</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Energy comparison

Network energy/packet (pJ)

- MECS+PVC
- MECS+TAQ
- MECS+EB+TAQ
- K-MECS

Legend:
- SR Routers
- Routers
- Link EBs
- Links
Energy comparison

![Graph showing energy comparison]

- **MECS**
- **MECS EB**
- **MECS hybrid**

- **SR Routers**
- **Routers**
- **Link EBs**
- **Links**

**Average packet latency (cycles)**

**Load (%)**

- **MECS+PVC**
- **MECS+TAQ**
- **MECS+EB+TAQ**
- **K-MECS**
Kilo-NOC: a heterogeneous NOC architecture for kilo-node substrates

- **Topology-aware QOS**
  - Limits QOS support to a fraction of the die
  - Leverages low-diameter topologies
  - Improves NOC area- and energy-efficiency
  - Provides strong guarantees
More on Kilo-NoC (I)

- Boris Grot, Joel Hestness, Stephen W. Keckler, and Onur Mutlu, "Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for Scalability and Service Guarantees". 
  Proceedings of the 38th International Symposium on Computer Architecture (ISCA), San Jose, CA, June 2011. Slides (pptx)

One of the 12 computer architecture papers of 2011 selected as Top Picks by IEEE Micro.

Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for Scalability and Service Guarantees

Boris Grot\(^1\) bgrot@cs.utexas.edu

Joel Hestness\(^1\) hestness@cs.utexas.edu

Stephen W. Keckler\(^{1,2}\) skeckler@nvidia.com

Onur Mutlu\(^3\) onur@cmu.edu

\(^{1}\)The University of Texas at Austin, Austin, TX

\(^{2}\)NVIDIA, Santa Clara, CA

\(^{3}\)Carnegie Mellon University, Pittsburgh, PA

A QoS-Enabled On-Die Interconnect Fabric for Kilo-Node Chips

To meet rapidly growing performance demands and energy constraints, future chips will likely feature thousands of on-die resources. Existing network-on-chip solutions weren’t designed for scalability and will be unable to meet future interconnect demands. A hybrid network-on-chip architecture called Kilo-NoC co-optimizes topology, flow control, and quality of service to achieve significant gains in efficiency.
Computer Architecture
Lecture 22: On-Chip Networks

Prof. Onur Mutlu
ETH Zürich
Fall 2021
10 December 2021