ECE 6115 / CS 8803 - ICN
Interconnection Networks for High Performance Systems
Spring 2020

SMART NETWORK-ON-CHIP

Tushar Krishna
Assistant Professor
School of Electrical and Computer Engineering
Georgia Institute of Technology

tushar@ece.gatech.edu
**Baseline Router Pipeline**

- **Per Packet**
  - RC, VA → done by Head flit

- **Per Flit**
  - BW, SA, BR, ST, LT
**Why does router delay matter?**

\[ T_N = (t_r + t_w) \times H + T_c + T_S \]

- \( T_N \): Network delay
- \( t_r \): router pipeline delay
- \( t_w \): wire delay per hop
- \( H \): number of hops
- \( T_c \): contention delay
- \( T_S \): serialization delay (for multi-flit packets)

**Which of these is static?**

\( t_r \quad t_w \quad T_s \)

**Which of these is dynamic (traffic-dependent)?**

\( H \quad T_c \)
CASE STUDY: INTEL SCC (ISSCC 2010)

Port 1 to 4
Port 0
24-deep flit FIFO

WEST
NORTH
EAST
SOUTH
Local

STAGE 1
Link Traversal + Buffer Write

STAGE 2
Switch Allocation

STAGE 3
Buffer Read + VC Allocation

STAGE 4
Switch Traversal

Crossbar

Switch Arbitration
The 5-port virtual cut-through router (Fig. 5.7.3) used to create the 2D-mesh network employs a credit-based flow-control protocol. Router ports are packet-switched, have 16-byte data links, and can operate at 2GHz at 1.1V. Each input port has five 24-entry queues, a route pre-computation unit, and a virtual-channel (VC) allocator. Route pre-computation for the output of the next router is done on queued packets. An XY dimension ordered routing algorithm is strictly followed. Deadlock free routing is maintained by allocating 8 virtual channels (VCs) between 2 message classes on all outgoing packets. VC0 through VC5 are kept in a free pool, while VC6 and VC7 are reserved for request classes and response classes, respectively. Input port and output port arbitrations are done concurrently using a wrapped wave front arbiter. Crossbar switch allocation is done in a single clock cycle on a packet granularity. No-load router latency is 4 clock cycles, including link traversal. Individual routers offer 64GB/s interconnect bandwidth, enabling the total network to support 256GB/s of bisection bandwidth.
**COMMON PIPELINE OPTIMIZATIONS**

- **BW + RC in parallel**
  - Lookahead Routing

- **SA + VA in parallel**
  - VC Select (switch output port winner selects VC from pool of free VCs)
  - Speculative VA (if VA takes long, speculatively allocate a VC while flit performs SA) (Peh and Dally, HPCA 2001)
    - If SA and VA both successful, go for ST
    - If SA or VA fails, retry next cycle

- **BR + SA in parallel**
  - The winner of Input Arbitration is read out and sent to the input of the crossbar speculatively

- **Low-load Bypassing**
  - When no flits in input buffer
    - Speculatively enter ST
    - On port conflict, speculation aborted
Analogy – Express Trains and Local Trains

Flits on Express VCs do not get buffered at intermediate routers
  - Send a “lookahead” to ask local flits to wait (i.e., kill switch allocation)
1-cycle for arbitration (tr), 1-cycle for traversal (tw)

Used by Tilera’s iMesh, Intel’s Ring, NoC prototypes (Park et al., DAC 2012)
Output Queues
  - “Virtual” Output Queues (== Virtual Channels)

Centralized Buffers

Rotary Router (in paper presentations)
MODERN PIPELINES

1-cycle for arbitration ($t_r$), 1-cycle for traversal ($t_w$)

*only required for Head flits

Used by Tilera’s iMesh, Intel’s Ring, NoC prototypes (Park et al., DAC 2012)
Can we remove the dependence of latency on hops?

Stay tuned!
What limits us from designing a 1-cycle network?

**Is it the wire delay?**

Repeated global wires can go up to 16mm within 1ns

Repeated global wire delay expected to remain constant/decrease slightly with technology scaling.

- Metal Layer = M6
- Repeater Spacing = 1mm
- Wire Width = $DRC_{\text{min}}$
- Wire Spacing = $3 \times DRC_{\text{min}}$
  
  *(coupling cap $\to 0$)*

*DSENT (NOCS 2012): Timing-driven NoC Power Estimation Tool*
What limits us from designing a 1-cycle network?

Is it the wire delay?

- Global repeated wires can transmit up to 10-16mm within 1ns (1GHz)

- Global repeated wires delay expected to remain fairly constant!

- Chip dimensions expected to remain similar (yield)

- Clock frequency expected to remain similar (power wall)

On-chip wires fast enough to transmit across the chip within 1-2 cycles at 1GHz even as technology scales
What limits us from designing a 1-cycle network?

Is it the wire delay? No!

**Classic scaling challenge with wires**
Wire-delay increases relative to logic delay

But …

**Wire-delay in cycles expected to remain constant.**

**Wires fast enough to transmit across chip in 1-2 cycles today and in future.**
DESIGNING A 1-CYCLE NETWORK

What limits us from designing a 1-cycle network?

Is it the wire delay? No!

Dream Traversal

Dedicated 1-cycle wire

Actual Traversal

Hop → Stop → Hop

on-chip router to manage sharing of output link every cycle

Number of wires = $O(n^2)$

Fully-connected (Impractical)

Number of wires = $O(n)$

Mesh (Practical)

Shared Links!
**DESIGNING A 1-CYCLE NETWORK**

What limits us from designing a 1-cycle network?

**Is it the wire delay? No!**

![1-cycle](image)

- **Number of wires = \( O(n^2) \)** (Fully-connected, impractical)

**Is it the routers? Yes!**

![9-Cycles (Best Case)](image)

- **Number of wires = \( O(n) \)** (Mesh, practical)

**Dedicated topology impractical**

*unless we design a chip for a specific application

- **Best Case: 1 cycle**

- **Note:** this is at every hop
DESIGNING A 1-CYCLE NETWORK

What limits us from designing a 1-cycle network?

Is it the wire delay? No!

Dedicated topology impractical*

Is it the routers? Yes!

Routers required to share links

Can we get both? Yes!

SMART: achieve the performance of dedicated connections over a network of shared links

*unless we design a chip for a specific application

Single-cycle
Multi-hop
Asynchronous
Repeated
Traversal

ICN | Spring 2020 | M07: SMART NoC © Tushar Krishna, School of ECE, Georgia Tech

Feb 17, 2020
A “SMART” NETWORK-ON-CHIP

- **Microarchitecture**
  - *Bypass path with clockless repeater at each router*

- **Flow Control**
  - *Compete for and reserve a sequence of shared links cycle-by-cycle*

Dynamically create repeated links (“SMART paths”) between any two routers

---

**Diagram Notes**:

- Single-cycle traversal
- Single-cycle reconfiguration
- Blue Flow has to stop for Pink flow
- Repeater

---

ICN | Spring 2020 | M07: SMART NoC © Tushar Krishna, School of ECE, Georgia Tech Feb 17, 2020
A “SMART” NETWORK-ON-CHIP

- **Microarchitecture**
  - Bypass path with clockless repeater at each router

- **Flow Control**
  - Compete for and reserve a sequence of shared links cycle-by-cycle

  Dynamically create repeated links ("SMART paths") between any two routers

- **How well does SMART perform?**
  - 88-90% of the performance of an O(n^2) wire fully-connected (dream) topology with an O(n) wire SMART NoC
  - Baseline Mesh needs to be clocked 5.4 times faster to match SMART

(64-core full-system simulation with real applications)

Microarchitecture and Flow Control details next!

---

T. Krishna et al.  
HPCA 2013  
IEEE Computer 2013  
IEEE Micro Top Picks 2014  
NOCS 2014 (Best Paper Award)  
H Kwon et al., ISPASS 2017
**MICOARCHITECTURE: DATA PATH**

**Network packet**

**Flit**

**Control**

**Xbar**

**Input Links**

- Core
- North
- South
- East
- West

**Bypass Path**

**Output Links**

**Repeater**

**Bypass mux select (mux):**
who uses crossbar+link (local or bypass)

**Xbar select (xbar):**
select line for xbar’s output mux

**Buffer enable (buffer):**
latch input flit or not

**SMART Router Microarchitecture**

128-bit

**These signals are setup by the control path**
**MICROARCHITECTURE: CONTROL PATH**

**HPC\textsubscript{max} (max Hops Per Cycle):**
maximum number of “hops” that the underlying wire allows the flit to traverse within a clock cycle

\[ \text{Length} = \text{HPC}\textsubscript{max} \text{ hops} \]

\[ \text{Width} = \log_2(1+\text{HPC}\textsubscript{max}) \text{ bits} \]

*E.g.*, \( \text{HPC}\textsubscript{max} = 10-16 @45\text{nm}, 1\text{GHz}, 1\text{mm hop} \)

Let \( \text{HPC}\textsubscript{max} = 3 \)

SSR for E direction

Dedicated repeated links from every router to help setup a SMART path

128-bit data path

2-bit control path
SMART FLOW CONTROL

- **Request** a path of *desired length (in hops)* over the SSR wires
  - Intermediate routers arbitrate between control requests from various routers and setup *buffer, mux* and *xbar* for the data path
  - No ACK has to be sent back!

- **Send the flit on the data wires**
  - May get partial or full SMART path based on contention that cycle

Assume $HPC_{\text{max}} = 3$ (max Hops Per Cycle)
Cycle 1 (Ctrl): R0 sends SSR = 3 (3-hop path request)

Assume $HPC_{\text{max}} = 3$ (max Hops Per Cycle)
**TRAVERSAL EXAMPLE 1: R0 → R3**

Cycle 1 (Ctrl): R0 sends SSR = 3 (3-hop path request)

All routers set buffer, mux and xbar for this request.

Assume $HPC_{max} = 3$ (max Hops Per Cycle)

---

**Table:**

<table>
<thead>
<tr>
<th></th>
<th>Buffer</th>
<th>Mux</th>
<th>Xbar</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>0</td>
<td>local</td>
<td>W→E</td>
</tr>
<tr>
<td>R1</td>
<td>0</td>
<td>bypass</td>
<td>W→E</td>
</tr>
<tr>
<td>R2</td>
<td>0</td>
<td>bypass</td>
<td>W→E</td>
</tr>
<tr>
<td>R3</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>R4</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
A SMART path is simply a combination of buffer, mux and xbar at all the intermediate routers.
**EXAMPLE 2: R0 → R3 AND R2 → R4**

Cycle 1 (Ctrl): R0 sends SSR = 3. R2 sends SSR = 2.

**Challenge**: only one flit can be sent on shared link between R2 and R3 at a time.

Assume $HPC_{max} = 3$ (max Hops Per Cycle)

Solution: Routers prioritize between path requests based on distance.

Two alternate schemes: $Prio=Local$ and $Prio=Bypass$.
**EXAMPLE 2: R0 → R3 AND R2 → R4**

**Prio = Local → 0 hop > 1 hop > 2 hop … > HPC_{max} hop**

**Cycle 2 (Data):** R2 sends flit to R4. R0’s flit blocked at R2.

<table>
<thead>
<tr>
<th></th>
<th>R0</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>mux</td>
<td>local</td>
<td>bypass</td>
<td>local</td>
<td>bypass</td>
<td>X</td>
</tr>
<tr>
<td>xbar</td>
<td>W→E</td>
<td>W→E</td>
<td>W→E</td>
<td>W→E</td>
<td>X</td>
</tr>
</tbody>
</table>

**Cycle 4+ (Data):** R2 sends blocked flit to R3. *(after local arbitration + sending ctrl)*

<table>
<thead>
<tr>
<th></th>
<th>R0</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>mux</td>
<td>X</td>
<td>X</td>
<td>local</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>xbar</td>
<td>X</td>
<td>X</td>
<td>W→E</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
**Example 2: R0 → R3 AND R2 → R4**

\[ \text{Prio} = \text{Bypass} \Rightarrow HPC_{\text{max}} \text{ hop} > (HPC_{\text{max}} - 1) \text{ hop} > \ldots > 1 \text{ hop} > 0 \text{ hop} \]

**Cycle 2 (Data):** R0 sends flit to R3. R2’s flit waits.

**Cycle 3 (Data):** R2 sends flit to R4.
Achievable HPC depends on link contention

At low loads (best case), as good as dedicated wires

At high loads (worst case), no worse than the baseline

SMART paths are opportunistic

Average hops in traffic pattern

Flit Injection Rate (% of capacity)

Average HPC (hops)

- Fully Connected (Dream)
- SMART
- Baseline (Mesh)
**Pipeline and Implementation**

Switch Allocation (SA-L) (can be bypassed at low loads)

- **Cycle 0**
  - Req + Switch Allocation Local (SA-L)

- **Cycle 1**
  - Req + Switch Allocation Global (SA-G)

**Repeater Overhead**

- **Energy:** Asynchronous repeater consumes 14.3% lower energy than clocked driver

- **Area:** No area overhead since repeaters are embedded in wire-dominated crossbar

**SA-G Overhead**

- **Energy:** \( \sim 2\% |_{\text{SMART}_1D}, \sim 2-6\% |_{\text{SMART}_2D} \)

- **Area:** < 1\% |_{\text{SMART}_1D}, 1-5\% |_{\text{SMART}_2D}

**Multi-hop Switch (crossbar) + Link Traversal (ST+LT)**

(till it is stopped by some buffer=1)

- \(HPC_{\text{max}} = 11\)

*Conventionally (i.e. in baseline), winners of SA-L go for Switch + Link Traversal*
THE DEVIL IS IN THE DETAILS

- Managing Distributed Arbitration
  - Could flits get misrouted?
  - Could flits not arrive when expected?

- SMART_2D
  - How can flits bypass routers at turns?

- Buffer Management
  - How is a flit guaranteed a buffer (and in the correct virtual channel) if it is stopped mid-way?
  - How is buffer availability conveyed?

- Multi-flit packets
  - How does SMART guarantee that flits (head, body, tail) of a packet do not get re-ordered?
  - How do flits know which VC to stop in
THE DEVIL IS IN THE DETAILS

- **Managing Distributed Arbitration**
  - Could flits get misrouted?
  - Could flits not arrive when expected?

- **SMART_2D**
  - How can flits bypass routers at turns?

- **Buffer Management**
  - How is a flit guaranteed a buffer (and in the correct virtual channel) if it is stopped mid-way?
  - How is buffer availability conveyed?

- **Multi-flit packets**
  - How does SMART guarantee that flits (head, body, tail) of a packet do not get re-ordered?
  - How do flits know which VC to stop in
Can different routers enforce different priorities?

Assume $HPC_{max} = 3$ (max Hops Per Cycle)

Cycle 1: R0 sends SSR = 3. R2 sends SSR = 2.

- SSR$_{R0} = 3$
- SSR$_{R2} = 2$

Prioritize SSR$_{R0}$ over SSR$_{R2}$

Prio = Bypass

Prioritize SSR$_{R2}$ over SSR$_{R0}$

Prio = Local
MANAGING DISTRIBUTED ARBITRATION

Cycle 1: R0 sends SSR = 3. R2 sends SSR = 2.

Can different routers enforce different priorities? No!

Assume $HPC_{\text{max}} = 3$ (max Hops Per Cycle)

Prioritize SSR$_{R0}$ over SSR$_{R2}$

R0’s flit incorrectly reaches R4, instead of getting stopped at R3.

Prioritize SSR$_{R2}$ over SSR$_{R0}$
MANAGING DISTRIBUTED ARBITRATION

- **Distributed Consensus**: All routers need to take the *same decision* about *multiple contending flits* in a *distributed manner*

- **Solution**: All routers follow the same *static priority* between the path setup requests that they receive
  - **Prio = Local**: 0 hop > 1 hop > … \((HPC_{\text{max}}-1)\) hop > \(HPC_{\text{max}}\) hop
  - **Prio = Bypass**: \(HPC_{\text{max}}\) hop > \((HPC_{\text{max}}-1)\) hop > … 1 hop > 0 hop

- **Implication**: a router will *not receive* a flit that it *does not expect*

- But can a router *not receive* a flit that it *does expect*?
Can a router *not receive* a flit that it *does expect*?

\begin{equation*}
SSR_{R1} = X
\end{equation*}

Control for N direction

\begin{equation*}
SSR_{R0} = 3
\end{equation*}

Control for E direction

\begin{align*}
\text{buffer} & \quad 0 \\
\text{mux} & \quad X \\
\text{xbar} & \quad X \\
\text{buffer} & \quad 0 \\
\text{mux} & \quad X \\
\text{xbar} & \quad X \\
\text{buffer} & \quad 0 \\
\text{mux} & \quad X \\
\text{xbar} & \quad X \\
\text{buffer} & \quad 0 \\
\text{mux} & \quad X \\
\text{xbar} & \quad X
\end{align*}

Prio = Local
Can a router \textit{not receive} a flit that it \textit{does expect}? Yes!

\textbf{Is there a performance loss?}

The R2→R3 link was granted for this cycle, but went unused. What if some other flit wanted to use it?

\textbf{No.}

(Prio=Local)

<table>
<thead>
<tr>
<th></th>
<th>buffer</th>
<th>mux</th>
<th>xbar</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>0</td>
<td>local</td>
<td>W→E</td>
</tr>
<tr>
<td>R1</td>
<td>1</td>
<td>local</td>
<td>xbar W→N</td>
</tr>
<tr>
<td>R2</td>
<td>0</td>
<td>bypass</td>
<td>xbar W→E</td>
</tr>
<tr>
<td>R3</td>
<td>1</td>
<td>X</td>
<td>xbar X</td>
</tr>
<tr>
<td>R4</td>
<td>0</td>
<td>X</td>
<td>xbar X</td>
</tr>
</tbody>
</table>
**IMPACT OF FALSE NEGATIVES**

**Cycle 1 (Ctrl Req):** R0\(\rightarrow\)R2, R1\(\rightarrow\)R4, R3\(\rightarrow\)R4.

- SSR\(_{R0}\) = 2
- SSR\(_{R1}\) = 3
- SSR\(_{R3}\) = 1

**Req Priority =**

**Cycle 2: Flit: R0\(\rightarrow\)R2.**

- Forced starvation and throughput loss

<table>
<thead>
<tr>
<th>buffer</th>
<th>mux</th>
<th>xbar</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>local</td>
<td>W(\rightarrow)E</td>
</tr>
<tr>
<td>0</td>
<td>bypass</td>
<td>W(\rightarrow)E</td>
</tr>
<tr>
<td>1</td>
<td>local</td>
<td>W(\rightarrow)E</td>
</tr>
<tr>
<td>0</td>
<td>bypass</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
IMPACT OF FALSE NEGATIVES

Prio = Bypass increases false negatives at high-loads

- Prio = Bypass saturates at ~44-48% injection rate
THE DEVIL IS IN THE DETAILS

- **Managing Distributed Arbitration**
  - Could flits get misrouted?
  - Could flits not arrive when expected?

- **SMART_2D**
  - How can flits bypass routers at turns?

- **Buffer Management**
  - How is a flit guaranteed a buffer (and in the correct virtual channel) if it is stopped mid-way?
  - How is buffer availability conveyed?

- **Multi-flit packets**
  - How does SMART guarantee that flits (head, body, tail) of a packet do not get re-ordered?
  - How do flits know which VC to stop in
BYPASS AT TURNS

Dest

Src
Assume $HPC_{\text{max}} = 3$

Any **blue** router ($HPC_{\text{max}}$ neighborhood) can be reached in 1 cycle

Shortest Path Routing → Any router in shaded $HPC_{\text{max}}$ quadrant is potential intermediate destination

Separate ctrl path for every possible route.

One of the ctrl paths chosen during route computation.

Only one of these 5 Reqs will be valid and will request for E/N/S output port.
**Challenge:** All input and output ports at all participating routers should make *consistent* decisions simultaneously.

**Solution:** 2-level priority among Reqs

1. **Distance**
   (i.e. Prio=Local or Prio=Bypass)

2. **Direction**

**How do we choose between Red and Purple?**

---

**buffer** | 0
---|---
**mux** | local
**xbar** | W→E

**buffer** | 1
---|---
**mux** | local
**xbar** | W→E

**buffer** | 0
---|---
**mux** | bypass
**xbar** | W→N? S→N?

**buffer** | 0
---|---
**mux** | bypass
**xbar** | S→? ?→ E

---

Red wins over Blue

Prio = Local
At **A**: Purple wins over Red

2-level Ctrl Req Priority

1. **Distance**
   
   0 hop > 1 hop > 2 hop … (Prio = Local)

2. **Direction**

   *Straight hops > Left hops > Right hops*

Assume \( HPC_{max} = 3 \)

All green routers (sources) can request for the **North output port** at the blue (intermediate) router

All routers enforce same priority (to guarantee no false positives)
**PRIORITY AT INPUT PORT**

At **A**: **Purple** wins over **Red**

At **B**: **Purple** wins over **Red**

2-level Ctrl Req Priority

1. **Distance**
   - 0 hop > 1 hop > 2 hop … (Prio = Local)

2. **Direction**
   - *Straight hops > Left hops > Right hops*

Assume **HPC_{max} = 3**

All **green** routers (sources) can request for the **South input port** at the **blue** (intermediate) router

**Intermediate**

Source Routers
Managing Distributed Arbitration
- Could flits get misrouted?
- Could flits not arrive when expected?

SMART_2D
- How can flits bypass routers at turns?

Buffer Management
- How is a flit guaranteed a buffer (and in the correct virtual channel) if it is stopped mid-way?
- How is buffer availability conveyed?

Multi-flit packets
- How does SMART guarantee that flits (head, body, tail) of a packet do not get re-ordered?
- How do flits know which VC to stop in
THE DEVIL IS IN THE DETAILS

- Managing Distributed Arbitration
  - Could flits get misrouted?
  - Could flits not arrive when expected?

- SMART_2D
  - How can flits bypass routers at turns?

- Buffer Management
  - How is a flit guaranteed a buffer (and in the correct virtual channel) if it is stopped mid-way?
  - How is buffer availability conveyed?

- Multi-flit packets
  - How does SMART guarantee that flits (head, body, tail) of a packet do not get re-ordered?
  - How do flits know which VC to stop in
Every router has free VC information about its neighbor, just like the baseline. R0 sends only if R1 has a free buffer. R1 lets the incoming flit bypass only if R2 has a free buffer, else latches it.

**Corollary:** VCid is allocated after it stops, rather than before starting, since router where it stops is not known.
THE DEVIL IS IN THE DETAILS

- Managing Distributed Arbitration
  - Could flits get misrouted?
  - Could flits not arrive when expected?

- SMART_2D
  - How can flits bypass routers at turns?

- Buffer Management
  - How is a flit guaranteed a buffer (and in the correct virtual channel) if it is stopped mid-way?
  - How is buffer availability conveyed?

- Multi-flit packets
  - How does SMART guarantee that flits (head, body, tail) of a packet do not get re-ordered?
  - How do flits know which VC to stop in
How do we guarantee Body/Tail flits from R0 do not bypass R2 if Head was stopped?

If R2 sees a SSR requesting bypass from a router from where it has a buffered flit, it stops all subsequent flits from that router, and buffers them in the same VC.
How do the Body/Tail flits know which VC to stop in if VC is allocated at the router where Head stops?

Match using source_id of all the flits of the packet
What if 2 flits from different packets from the same source arrive?

Virtual Cut-Through
• enough Buffers to store all flits of a packet
• all flits of one packet should leave before flits of another packet
**EVALUATIONS**

- **Simulation Infrastructure:** GEMS (Full-System) + Garnet (NoC)
- **System:** 64-core (8x8 Mesh)
- **Technology:** 45nm, 1GHz

**Networks being compared**

- **BASELINE** \( (t_r=1) \): Baseline Mesh with 1-cycle router at every hop
- **SMART-**\(<\text{HPC}^{\max}>_{1D}\): Always stop at turning router
  - Best case delay: 4 cycles \((\text{Req}_X \rightarrow \text{Flit}_X \rightarrow \text{Req}_Y \rightarrow \text{Flit}_Y)\)
- **SMART-**\(<\text{HPC}^{\max}>_{2D}\): Bypass turning router
  - Best case delay: 2 cycles \((\text{Req} \rightarrow \text{Flit})\)
- **DREAM** \( (T_N=1) \): Contention-less 1-cycle network (fully-connected)
BREAKING THE LATENCY BARRIER

**Uniform Random (Avg Hops = 5.33)**

**Bit Complement (Avg Hops = 8)**

SMART reduces low-load latency to 2-4 cycles **across all** traffic patterns, **independent** of average hops.

**Shuffle (Avg Hops = 4)**

**Transpose (Avg Hops = 6)**
**IMPACT OF HPC_{max}**

- **HPC_{max} of 2 and 4** give 1.8X and 3X reduction in latency.
- **HPC_{max} of 8** gives 5.4X reduction in latency.

**HPC_{max}** (max Hops Per Cycle): maximum number of “hops” that the underlying wire allows the flit to traverse within a clock cycle.

SMART is a better design choice than a baseline 1-cycle router network even at larger tile size or higher frequency (i.e., lower HPC_{max}).

A baseline mesh network needs to run at 5.4GHz to match the performance of a 1GHz SMART NoC.

HPC_{max} will go up as technology scales! (smaller cores, similar frequency).

---

**Graph Details:**
- **X-axis:** Injection Rate (flits/node/cycle)
- **Y-axis:** Avg Flit Latency (cycles)
- **Legend:**
  - SMART-1_1D
  - SMART-2_1D
  - SMART-4_1D
  - SMART-8_1D
  - SMART-4_2D
  - SMART-8_2D
  - SMART-12_2D

---

**Note:**
- © Tushar Krishna, School of ECE, Georgia Tech
Full-system runtime

Directory Protocol

Private L2/tile
SMART reduces runtime by 26-27% for Private L2
8% off a dream 1-cycle network

Shared L2/tile
SMART reduces runtime by 49-52% for Shared L2
9% off a dream 1-cycle network
PROJECT IDEAS USING SMART

- **SMART NoCs**
  - SMART NoCs vs High-Radix topologies
  - SMART NoCs with non-minimal routes
  - SMART paths for latency guarantees
  - SMART NoC inside GPU
  - Reserve SMART paths using hints from cache/memory-controller/OS
If a virtual network requires protocol-level ordering
  - Only deterministic routing allowed within that virtual network.
  - Priority should be Local > Bypass
    - Guarantees that 2 flits from the same source do not overtake each other at any router.
**What if we exploit repeated wires and add explicit 1-cycle physical express links?**

<table>
<thead>
<tr>
<th></th>
<th>SMART</th>
<th>Flattened Butterfly</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ports</strong></td>
<td>5-port router</td>
<td>15-port router</td>
</tr>
<tr>
<td><strong>Router Delay</strong></td>
<td>1-2 cycle in router</td>
<td>3-4 cycle in router</td>
</tr>
<tr>
<td></td>
<td>[SA-L, SSR+SA-G]</td>
<td>[8-14:1 Arbiter, Multi-stage Xbar]</td>
</tr>
</tbody>
</table>
| **Bisection Bandwidth (BB)** | 128x8  
  1-flit req  
  5-flit resp | 128x8 [1x]  
  7-flit req  
  35-flit resp |
|                      |                        | 448x8 [3.5x]  
  2-flit req  
  10-flit resp |
|                      |                        | 896x8 [7x]  
  1-flit req  
  5-flit resp |
| **Dyn Power (mW)**   | 24.5                   | 23.1 [~1x]                   |
|                      |                        | 37 [1.5x]                    |
|                      |                        | 58 [2.3x]                    |
| **Area (mm x mm)**   | 0.27x0.27              | 0.47x0.47 [3x]               |
|                      |                        | 0.53x0.53 [3.9x]             |
|                      |                        | 0.7x0.7 [6.7x]               |
**SMART vs. Flattened Butterfly**

Assume 1-cycle Flattened Butterfly Router: **Highly optimistic assumption**

![Graph showing latency vs. injection rate comparison between SMART and FBfly.]

- **SMART always beats FBfly in latency**
- **FBfly matches SMART in throughput only with 3.5X more wires**

**Better to use SMART and reconfigure 1-cycle multi-hop paths based on traffic rather than use a **fixed** high-radix topology.**