Lecture 12: SMART NoC

Tushar Krishna

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

tushar@ece.gatech.edu

Network Architecture

- **Topology**
  - How to connect the nodes
  - ~Road Network

- **Routing**
  - Which path should a message take
  - ~Series of road segments from source to destination

- **Flow Control**
  - When does the message have to stop/proceed
  - ~Traffic signals at end of each road segment

- **Router Microarchitecture**
  - How to build the routers
  - ~Design of traffic intersection (number of lanes, algorithm for turning red/green)
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
Express Virtual Channels (ISCA 2007)

- 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)
Modern Pipelines

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

Used by Tilera’s iMesh, Intel’s Ring, NoC prototypes (Park et al., DAC 2012)
Is that the best we can do?

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

Can we remove the dependence of latency on hops?

Stay tuned!
Designing a 1-cycle network

What limits us from designing a 1-cycle network?

Is it the wire delay?

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

Repeater spacing expected to remain constant/decrease slightly with technology scaling.

Metal Layer = M6
Repeater Spacing = 1mm
Wire Width = DRC_{min}
Wire Spacing = 3 \times DRC_{min} (coupling cap \rightarrow 0)

*DSENT (NOCS 2012): Timing-driven NoC Power Estimation Tool
Designing a 1-cycle network

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.
Designing a 1-cycle network

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

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

Fully-connected (Impractical)

Actual Traversal

- Hop → Stop → Hop
- On-chip router to manage sharing of output link every cycle

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!

Dedicated topology impractical*

*unless we design a chip for a specific application

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

Fully-connected (Impractical)

Is it the routers? Yes!

Mesh (Practical)

Routers required to share links

Number of wires = $O(n)$

9-Cycles (Best Case)

(singue-cycle router, no other traffic!)

Hop $\rightarrow$ Stop $\rightarrow$ Hop

Best Case: 1 cycle

Note: this is at every hop

February 22, 2017
Designing a 1-cycle network

What limits us from designing a 1-cycle network?

Is it the wire delay? No!

Is it the routers? Yes!

Dedicated topology impractical*

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

---

**February 22, 2017**
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 showing network architecture](image_url)
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!
Microarchitecture: Data Path

Network packet

Input Links
- Flit
- Core
- North
- South
- East
- West

Input Port
- Bypass Path
- 128-bit

Control

Xbar

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

Traditional Router Microarchitecture

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.

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

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

e.g., HPC\textsubscript{max} = 10-16
@45nm, 1GHz, 1mm hop

Let HPC\textsubscript{max} = 3

SSR for E direction

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

128-bit data path
SMART Flow Control

- Divide the packet into flits
  - head, body, tail

- Divide the route into segments of $HPC_{max}$

- 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 $HPC_{max} = 3$ (max Hops Per Cycle)
Traversing Example 1: R0 → R3

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

 Assumes \( HPC_{\text{max}} = 3 \) (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 $\text{HPC}_{\text{max}} = 3$
(max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)

- SSR$_{R0} = 3$
- Cycle 1 (Ctrl): R0 sends SSR = 3

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

- Assume $\text{HPC}_{\text{max}} = 3$
- (max Hops Per Cycle)
Traversals 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.

Cycle 2 (Data): R0 sends flit to R3

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

A SMART path is simply a combination of buffer, mux and xbar at all the intermediate routers.
Example 2: R0 $\rightarrow$ R3 and R2 $\rightarrow$ 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

**Solution**: Routers prioritize between path requests based on distance

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

Two alternate schemes: 
- $Prio=\text{Local}$
- $Prio=\text{Bypass}$
Example 2: R0 ➝ R3 and R2 ➝ R4

Prio = Local ➞ 0 hop > 1 hop > 2 hop ... > $HPC_{\text{max}}$ hop

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

**Cycle 4+ (Data):** R2 sends blocked flit to R3. (after local arbitration + sending ctrl)
Example 2: R0 → R3 and R2 → R4

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

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

**Cycle 3 (Data):** R2 sends flit to R4.
Achievable Hops Per Cycle (HPC)

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

SMART paths are opportunistic

Achievable HPC depends on link contention

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

**Average HPC**

![Graph showing achievable HPC vs. flit injection rate.](graph.png)

- **Fully Connected (Dream)**
- **SMART**
- **Baseline (Mesh)**

**Average hops in traffic pattern**

**Flit Injection Rate (% of capacity)**

**Achievable HPC** depends on link contention.
Pipeline and Implementation

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

Switch Allocation Global (SA-G)

\[ \min\{T_{ST+LT}, T_{Req} + T_{SA-G}\} \text{ determines critical path (thus } HPC_{max}\)\]

Cycle 0

Cycle 1*

Multi-hop Switch (crossbar) + Link Traversal (ST+LT)
(till it is stopped by some buffer=1)

\[ HPC_{max} = 13 \mid \text{SMART}_1D, \quad HPC_{max} = 9 \mid \text{SMART}_2D \]

SA-G Overhead

Energy: \(~2\% \mid \text{SMART}_1D, \quad ~2-6\% \mid \text{SMART}_2D\)
Area: \(< 1\% \mid \text{SMART}_1D, \quad 1-5\% \mid \text{SMART}_2D\)

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

*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
Managing Distributed Arbiration

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

Can different routers enforce different priorities?

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

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

Prioritize SSR$_{R2}$ over SSR$_{R0}$
 Managing Distributed Arbitration

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

Can different routers enforce different priorities? **No!**

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

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

Prioritize $SSR_{R0}$ over $SSR_{R2}$

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**: $0 \text{ hop} > 1 \text{ hop} > \ldots (HPC_{\text{max}} - 1) \text{ hop} > HPC_{\text{max}} \text{ hop}$
  - **Prio = Bypass**: $HPC_{\text{max}} \text{ hop} > (HPC_{\text{max}} - 1) \text{ hop} > \ldots 1 \text{ hop} > 0 \text{ hop}$

- **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*?

- SSR$_{R1} = X$
- SSR$_{R0} = 3$

**Control for N direction**

**Control for E direction**

<table>
<thead>
<tr>
<th>Buffer</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mux</td>
<td>X</td>
</tr>
<tr>
<td>Xbar</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Buffer</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mux</td>
<td>X</td>
</tr>
<tr>
<td>Xbar</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Buffer</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mux</td>
<td>X</td>
</tr>
<tr>
<td>Xbar</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Buffer</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mux</td>
<td>X</td>
</tr>
<tr>
<td>Xbar</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Buffer</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mux</td>
<td>X</td>
</tr>
<tr>
<td>Xbar</td>
<td>X</td>
</tr>
</tbody>
</table>
Can a router *not receive* a flit that it *does expect*? Yes!

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?

No. (Prio=Local)
Impact of False Negatives

Cycle 1 (Ctrl Req): R0→R2, R1→R4, R3→R4.

Req Priority = Bypass

Cycle 2: Flit: R0→R2.

Forced starvation and throughput loss
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
Bypass at turns

Assume $HPC_{max} = 3$

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

Shortest Path Routing $\rightarrow$ Any router in shaded $HPC_{max}$ 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.
Control Arbitration

**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 and Multiplexer Settings

<table>
<thead>
<tr>
<th>Buffer</th>
<th>Multiplexer</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Local</td>
<td>W→E</td>
</tr>
<tr>
<td>1</td>
<td>Local</td>
<td>W→E</td>
</tr>
<tr>
<td>0</td>
<td>Bypass</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Bypass</td>
<td></td>
</tr>
<tr>
<td>S→?</td>
<td>S→N?</td>
<td></td>
</tr>
</tbody>
</table>

**Prio = Local**

**Red wins over Blue**
Priority at Output Port

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

2-level Ctrl Req Priority
1. **Distance**
   - $0\text{ hop} > 1\text{ hop} > 2\text{ hop} \ldots$ (Prio = Local)

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

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

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

All routers enforce same priority (to guarantee no false positives)
**Priority at Input Port**

2-level Ctrl Req Priority
1. **Distance**
   - 0 hop > 1 hop > 2 hop … (Prio = Local)

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

---

**At A: Purple wins over Red**

**At B: Purple wins over Red**

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

Assume $HPC_{max} = 3$
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
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
R0 → R3, R2 → R4

SSR_{R0} = 3

SSR_{R2} = 2

How do we guarantee R2 has a free buffer / VC for the flit from R0?

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.
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.
Multi-flit Packets (2)

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
Multi-flit Packets (2)

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-<HPC_{max}>_1D**: Always stop at turning router
    - Best case delay: 4 cycles (Req_X → Flit_X → Req_Y → Flit_Y)
  - **SMART-<HPC_{max}>_2D**: Bypass turning router
    - Best case delay: 2 cycles (Req → Flit)
  - **DREAM (T_N=1)**: Contention-less 1-cycle network (fully-connected)
Breaking the latency barrier

**Average flit latency (cycles)**

**Injec4on Rate (flits/node/cycle)**

- **BASELINE**(t_r=1)
- **SMART-8_1D**
- **SMART-8_2D**
- **DREAM**(T_N = 1)

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

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

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

**Shuffle (Avg Hops = 4)**

**Transpose (Avg Hops = 6)**
Impact of $HPC_{\text{max}}$

HPC_{\text{max}}$ of 2 and 4 give 1.8X and 3X reduction in latency

HPC_{\text{max}}$ of 8 gives 5.4X reduction in latency

$HPC_{\text{max}}$ (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 $HPC_{\text{max}}$)

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

$HPC_{\text{max}}$ will go up as technology scales! (smaller cores, similar frequency)
Full-system Runtime

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

SMART reduces runtime by 49-52% for Shared L2
9% off a dream 1-cycle network
Backups
Protocol-level ordering

- 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.
SMART vs. Flattened Butterfly

What if we exploit repeated wires and add explicit 1-cyle 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 [SA-L, SSR+SA-G]</td>
<td>3-4 cycle in router [8-14:1 Arbiter, Multi-stage Xbar]</td>
</tr>
<tr>
<td><strong>Bisection Bandwidth (BB)</strong></td>
<td>128x8 1-flit req 5-flit resp</td>
<td>128x8 [1x] 7-flit req 35-flit resp</td>
</tr>
<tr>
<td><strong>Dyn Power (mW)</strong></td>
<td>24.5</td>
<td>23.1 [~1x]</td>
</tr>
<tr>
<td><strong>Area (mm x mm)</strong></td>
<td>0.27x0.27</td>
<td>0.47x0.47 [3x]</td>
</tr>
</tbody>
</table>
SMART vs. Flattened Butterfly

Assume 1-cycle Flattened Butterfly Router: Highly optimistic assumption

- 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.