On-Chip Networks II: Router Microarchitecture & Routing

Mengjia Yan
Computer Science & Artificial Intelligence Lab
M.I.T.

Based on slides from Daniel Sanchez
Reminder: Wormhole Flow Control

- Each router manages buffers in flits
- Each packet is sent through output link as soon as possible (without waiting for all its flits to arrive)
- Router buffers are not large enough to hold full packet → on congestion, packet’s flits often buffered across routers

Problem: On congestion, links assigned to a blocked packet cannot be used by other packets
Reminder: Wormhole Flow Control

- Each router manages buffers in flits
- Each packet is sent through output link as soon as possible (without waiting for all its flits to arrive)
- Router buffers are not large enough to hold full packet → on congestion, *packet’s flits often buffered across routers*
- **Problem**: On congestion, links assigned to a blocked packet cannot be used by other packets

Wormhole

![Wormhole diagram]

April 16, 2020
Virtual-Channel (VC) Flow Control

- When a packet blocks, instead of holding on to channel, hold on to **virtual channel**
- Virtual channel = channel state + flit buffers
- Multiple virtual channels reduce blocking
- Ex: Wormhole (=1 VC/channel) vs 2 VCs/channel

**VC flow control** with 2 VCs/channel

![Diagram of VC flow control with 2 VCs/channel]
Router
Microarchitecture
Ring-based Interconnect
Components in a router
- Buffer
- Switch
- Logic
Components in a router
- Buffer
- Switch
- Logic

Allow input if no traffic on ring
Q: If there is traffic on ring, should traffic on ring or new input get priority?

Components in a router
- Buffer
- Switch
- Logic

Allow input if no traffic on ring
Ring Flow Control: Priorities

Rotary Rule – traffic in ring has priority
Ring Flow Control: Bounces

- What if traffic on the ring cannot get delivered, e.g., if output FIFO is full?
Ring Flow Control: Bounces

• What if traffic on the ring cannot get delivered, e.g., if output FIFO is full?

• One alternative: Continue on ring (bounce)
Ring Flow Control: Bounces

- What if traffic on the ring cannot get delivered, e.g., if output FIFO is full?
- One alternative: Continue on ring (bounce)
- What are the consequences of such bounces?
Ring Flow Control: Bounces

- What if traffic on the ring cannot get delivered, e.g., if output FIFO is full?
- One alternative: Continue on ring (bounce)
- *What are the consequences of such bounces?*

Traffic on ring no longer FIFO
General Interconnect
Tilera, Knights Landing...
What’s In A Router?

• It’s a system as well

  – Logic – State machines, Arbiters, Allocators
    • Control data movement through router
    • Idle, Routing, Waiting for resources, Active

  – Memory – Buffers
    • Store flits before forwarding them
    • SRAMs, registers, processor memory

  – Communication – Switches
    • Transfer flits from input to output ports
    • Crossbars, multiple crossbars, fully-connected, bus
Virtual-channel Router

Flit

Router

VC Allocator

Switch Allocator

Input Unit

Switch

Output Unit
Router Pipeline vs. Processor Pipeline

- Logical stages:
  - BW
  - RC
  - VA
  - SA
  - BR
  - ST
  - LT
- Different flits go through different stages
- Different routers have different variants
  - E.g. speculation, lookaheads, bypassing
- Different implementations of each pipeline stage

- Logical stages:
  - IF
  - ID
  - EX
  - MEM
  - WB
- Different instructions go through different stages
- Different processors have different variants
  - E.g. speculation, ISA
- Different implementations of each pipeline stage
Baseline Router Pipeline

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head</td>
<td>BW</td>
<td>RC</td>
<td>VA</td>
<td>SA</td>
<td>ST</td>
<td>LT</td>
<td></td>
</tr>
<tr>
<td>Body 1</td>
<td>BW</td>
<td></td>
<td></td>
<td>SA</td>
<td>ST</td>
<td>LT</td>
<td></td>
</tr>
<tr>
<td>Body 2</td>
<td>BW</td>
<td></td>
<td></td>
<td>SA</td>
<td>ST</td>
<td>LT</td>
<td></td>
</tr>
<tr>
<td>Tail</td>
<td>BW</td>
<td></td>
<td></td>
<td>SA</td>
<td>ST</td>
<td>LT</td>
<td></td>
</tr>
</tbody>
</table>

- Route computation performed once per packet
- Virtual channel allocated once per packet
- Body and tail flits inherit this info from head flit
Allocators In Routers

• VC Allocator
  – Input VCs requesting for a range of output VCs
  – Example: A packet of VC0 arrives at East input port. It’s destined for west output port, and would like to get any of the VCs of that output port.

• Switch Allocator
  – Input VCs of an input port request for different output ports (e.g., One’s going North, another’s going West)

• “Greedy” algorithms used for efficiency
Allocators In Routers

- VC Allocator
  - Input VCs requesting for a range of output VCs
  - Example: A packet of VC0 arrives at East input port. It’s destined for west output port, and would like to get any of the VCs of that output port.

- Switch Allocator
  - Input VCs of an input port request for different output ports (e.g., One’s going North, another’s going West)

- “Greedy” algorithms used for efficiency

- What happens if allocation fails on a given cycle?
VC & Switch Allocation Stalls

Cycle 1 2 3 4 5 6 7 8

Head Flit (packet A) RC VA SA ST

Tail Flit (packet B - holds VC) SA ST

Body Flit (packet A) SA ST
VC & Switch Allocation Stalls

Cycle 1 2 3 4 5 6 7 8
Head Flit (packet A)
Tail Flit (packet B - holds VC)
Body Flit (packet A)

Cycle 1 2 3 4 5 6 7 8
Head Flit
Body Flit 1
Body Flit 2
Body Flit 3
Pipeline Optimizations: Lookahead Routing [Galles, SGI Spider Chip]

- At current router, perform route computation for next router
  
<table>
<thead>
<tr>
<th>BW</th>
<th>VA</th>
<th>NRC</th>
<th>SA</th>
<th>ST</th>
<th>LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Head flit already carries output port for next router
- RC just has to read output → fast, can be overlapped with BW
- Precomputing route allows flits to compete for VCs immediately after BW
- Routing computation for the next hop (NRC) can be computed in parallel with VA

- Or simplify RC (e.g., X-Y routing is very fast)
Pipeline Optimizations: Speculative Switch Allocation [Peh&Dally, 2001]

• Assume that Virtual Channel Allocation stage will be successful
  – Valid under low to moderate loads

• If both successful, VA and SA are done in parallel

• If VA unsuccessful (no virtual channel returned)
  – Must repeat VA/SA in next cycle

• Prioritize non-speculative requests
Routing
Properties of Routing Algorithms

- **Deterministic/Oblivious**
  - route determined by (source, dest),
  - not affected by network state (i.e. traffic)

- **Adaptive**
  - route influenced by traffic along the way

- **Minimal**
  - only selects shortest paths

- **Deadlock-free**
  - no traffic pattern can lead to a situation where no packets move forward
Properties of Routing Algorithms

- **Deterministic/Oblivious**
  - route determined by (source, dest),
  - not affected by network state (i.e. traffic)

- **Adaptive**
  - route influenced by traffic along the way

- **Minimal**
  - only selects shortest paths

- **Deadlock-free**
  - no traffic pattern can lead to a situation where no packets move forward
Network Deadlock

- Flow A holds $u$ and $v$ but cannot make progress until it acquires channel $w$.
- Flow B holds channels $w$ and $x$ but cannot make progress until it acquires channel $u$. 
Network Deadlock

- Flow A holds $u$ and $v$ but cannot make progress until it acquires channel $w$.
- Flow B holds channels $w$ and $x$ but cannot make progress until it acquires channel $u$. 

Circular dependences of resources.
Dimension-Order Routing

XY-order

YX-order

Uses 2 out of 4 turns

Uses 2 out of 4 turns

XY is deadlock free, YX is deadlock free, what about XY+YX?
Dimension-Order Routing

$XY$-order

$YX$-order

Uses 2 out of 4 turns

Uses 2 out of 4 turns

$XY$ is deadlock free, $YX$ is deadlock free, what about $XY+YX$?

No!
DOR – Turns allowed

- One way of looking at whether a routing algorithm is deadlock free is to look at the turns allowed.

- Deadlocks may occur if turns can form a cycle

![Diagrams of XY and YX Models](L17-33)
Allowing more turns

- Allowing more turns may allow adaptive routing, but also **deadlock**

Six turn model
Allowing more turns

- Allowing more turns may allow adaptive routing, but also **deadlock**
Turn Model [Glass and Ni, 1994]

- A systematic way of generating deadlock-free routes with a small number of prohibited turns.

- Deadlock-free if routes conform to at least ONE of the turn models (acyclic channel dependence graph).

West-First Turn Model

North-Last Turn Model
Can create a channel dependency graph (CDG) of the network.

Vertices in the CDG represent network links. Edges in CDG represent allowed route.

Disallowing 180° turns, e.g., AB → BA.
Cycles in CDG

The channel dependency graph D derived from the network topology may contain many cycles.

Flow routed through links AB, BE, EF
Flow routed through links EF, FA, AB
\( \Rightarrow \) Deadlock!
Key Insight

If routes of flows conform to **acyclic** CDG, then there will be no possibility of deadlock!

Disallow/Delete certain edges in CDG

Edges in CDG correspond to turns in network!
Acyclic CDG → Deadlock-free routes

Turns could be prohibited ad-hoc, all the edges in red are deleted.

Ad-hoc Acyclic CDG

April 16, 2020

MIT 6.823 Spring 2020

L17-40
East-first → Deadlock-free routes

Per the East-First prohibited turns, all the edges in red are deleted

East-First Turn Model

East-First Acyclic CDG
Routing deadlocks in wormhole routing result from Structural hazard at router resources, e.g., buffers.

*How can structural hazards be avoided?*
Routing deadlocks in wormhole routing result from Structural hazard at router resources, e.g., buffers.

*How can structural hazards be avoided?*

Adding more resources
Virtual Channels

- Virtual channels can be used to avoid deadlock by restricting VC allocation
CDG and Virtual Channels
Randomized Routing: Valiant

- Route each packet through a randomly chosen intermediate node

A packet, going from node $SA$ to node $DA$, is first routed from $SA$ to a randomly chosen intermediate node $IA$, before going from $IA$ to final destination $DA$.

It helps load-balance the network and has a good worst-case performance at the expense of locality.
To retain locality, choose intermediate node in the \textit{minimal quadrant}

Equivalent to randomly selecting among the various minimal paths from source to destination
Thank you!

Next Lecture: Memory Consistency Models