Interconnection Networks:

Router Microarchitecture

Prof. Natalie Enright Jerger
Introduction

• Topology: connectivity
• Routing: paths
• Flow control: resource allocation

• Router microarchitecture: implementation of routing, flow control and router pipeline
  – Impacts per-hop delay and energy
Router Microarchitecture Overview

• Focus on microarchitecture of Virtual Channel (VC) router
  – Router complexity increase with bandwidth demands
  – Simple routers built when high throughput is not needed
    • Wormhole flow control, unpipelined, limited buffers
Virtual Channel Router

Route Computation

VC Allocator

Switch Allocator

Credits In

Credits Out

Input buffers

VC 1
VC 2
VC 3
VC 4

Input 1

Output 1

Input 5

Output 5

Fall 2014
ECE 3/451, Interconnection Networks (Enright-Jerger)
Router Components

• Input buffers, route computation logic, virtual channel allocator, switch allocator, crossbar switch

• Most OCN routers are input buffered
  – Use single-ported memories

• Buffer store flits for duration in router
  – Contrast with processor pipeline that latches between stages
Baseline Router Pipeline

- Logical stages
  - Fit into physical stages depending on frequency

- Canonical 5-stage pipeline
  - BW: Buffer Write
  - RC: Routing computation
  - VA: Virtual Channel Allocation
  - SA: Switch Allocation
  - ST: Switch Traversal
  - LT: Link Traversal
Baseline Router Pipeline (2)

- Routing computation performed once per packet
- Virtual channel allocated once per packet
- Body and tail flits inherit this info from head flit
Router Pipeline Performance

• Baseline (no load) delay

\[ = (5 \text{cycles} + \text{link delay}) \times \text{hops} + t_{\text{serialization}} \]

• Ideally, only pay link delay

• Techniques to reduce pipeline stages
Pipeline Optimizations: Lookahead Routing

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

- Dependence between output of one module and input of another
  - Determine critical path through router
  - Cannot bid for switch port until routing performed
Atomic Modules

• Some components of router cannot be easily pipelined

• Example: pipeline VC allocation
  – Grants might not be correctly reflected before next allocation

• Separable allocator: many wires connecting input/output stages
  – Require latches if pipelined
Pipeline Optimizations: Speculation

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

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

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

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

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

1a. Lookahead Routing Computation
2a. Virtual Channel Allocation

1. Inject

2. No buffered flits when A arrives
Speculation

Virtual Channel Allocation
- Switch Allocation

Lookahead Routing Computation

Inject

Eject

Port conflict detected

A succeeds in VA but fails in SA, retry SA

Fall 2014
ECE 1749H: Interconnection Networks (Enright Jerger)
Buffer Organization

- Single buffer per input
- Multiple fixed length queues per physical channel
Buffer Organization

- Multiple variable length queues
  - Multiple VCs share a large buffer
  - Each VC must have minimum 1 flit buffer
    - Prevent deadlock
  - More complex circuitry
Buffer Organization

• Many shallow VCs?
• Few deep VCs?

• More VCs ease HOL blocking
  – More complex VC allocator

• Light traffic
  – Many shallow VCs – underutilized
• Heavy traffic
  – Few deep VCs – less efficient, packets blocked due to lack of VCs
Switch Organization

• Heart of datapath
  – Switches bits from input to output
• High frequency crossbar designs challenging
• Crossbar composed for many multiplexers
  – Common in low-frequency router designs
• Long wires: repeater insertion

Fall 2014

ECE 1749H: Interconnection Networks (Enright Jerger)
Switch Organization: Crosspoint

- Area and power scale at $O((pw)^2)$
  - $p$: number of ports (function of topology)
  - $w$: port width in bits (determines phit/flit size and impacts packet energy and delay)
Crossbar speedup

- Increase internal switch bandwidth

- Simplifies allocation or gives better performance with a simple allocator
  - More inputs to select from \( \rightarrow \) higher probability each output port will be matched (used) each cycle
Switch Microarchitecture: No Speedup

Routing Control and Arbitration Unit

Physical channel

Link Control

Input buffers

MUX

Demux

Physical channel

CrossBar

Physical channel

Input buffers

MUX

Demux

Physical channel

Routing Control and Arbitration Unit

Physical channel
Switch Microarchitecture: Speedup

Switch input speedup

Routing Control and Arbitration Unit
Switch Microarchitecture: Speedup

Switch input & output speedup
Crossbar Dimension Slicing

• Crossbar area and power grow with $O((pw)^2)$

• Replace 1 5x5 crossbar with 2 3x3 crossbars
• Suited to DOR  
  – Traffic mostly stays within 1 dimension
Arbiters and Allocators

• Allocator matches N requests to M resources

• Arbiter matches N requests to 1 resource

• Resources are VCs (for virtual channel routers) and crossbar switch ports.
Arbiters and Allocators (2)

- Virtual-channel allocator (VA)
  - Resolves contention for output virtual channels
  - Grants them to input virtual channels

- Switch allocator (SA) that grants crossbar switch ports to input virtual channels

- Allocator/arbiter that delivers high matching probability translates to higher network throughput.
  - Must also be fast and/or able to be pipelined

- VC allocation typically determines cycle time
Fairness

• Intuitively, a fair arbiter is one that provides equal service to different requesters

• Weak fairness: Every request is eventually served

• Strong fairness: Requesters will be served equally often

• FIFO Fairness: Requesters are served in the order they make their requests
Locally Fair Example

- R3 receives 4 times the bandwidth as r0, even though individual arbiters provide strong fairness.
Round Robin Arbiter

• Last request serviced given lowest priority

• Generate the next priority vector from current grant vector

• If no requests, priority is unchanged

• Exhibits fairness
• $G_i$ granted, next cycle $P_{i+1}$ high
Weighted Round-Robin Arbiter

- Disable request if input has already reached its quota
Matrix Arbiter

• Least recently served priority scheme

• Triangular array of state bits $w_{ij}$ for $j < i$
  – Bit $w_{ij}$ indicates request $i$ takes priority over $j$
  – Each time request $k$ granted, clears all bits in row $k$
    and sets all bits in column $k$

• Good for small number of inputs

• Fast, inexpensive and provides strong fairness
Matrix Arbiter (2)
• When a request is asserted
  – AND-ed with the state bits in its row to disable any lower priority requests
  – Outputs of AND gates in column are OR-ed together to generate disable signal
Matrix Arbiter (4)

- If Request 2 granted
  - Clear row 2
  - Set column 2
Matrix Arbiter Example

Request 0

Request 1

Request 2

Disable 0

Disable 1

Disable 2

Grant 0

Grant 1

Grant 2

Bit \([1,0] = 1\), Bit \([2,0] = 1\) \(\rightarrow\) 1 and 2 have priority over 0

Bit \([2,1] = 1\) \(\rightarrow\) 2 has priority over 1

\(C_1\) (Req 2) granted
Matrix Arbiter Example (2)

Request 0

Request 1

Request 2

Disable 0

Disable 1

Disable 2

Grant 0

Grant 1

Grant 2

Set column 2, clear row 2
Bit [1,0] = 1, Bit [1,2] = 1 \(\rightarrow\) Req 1 has priority over 0 and 2
Grant \(B_1\) (Req 1)
Matrix Arbiter Example (3)

Set column 1, clear row 1
Bit [0,1] = 1, Bit [0,2] = 1 $\Rightarrow$ Req 0 has priority over 1 and 2
Grant $A_1$ (Req 0)
Matrix Arbiter Example (4)

Request 0

Request 1

Request 2

Disable 0

Disable 1

Disable 2

Grant 0

Grant 1

Grant 2

Set column 0, clear row 0

Bit \([2,0] = 1\), Bit \([2,1] = 1\) \(\rightarrow\) Req 2 has priority over 0 and 1

Grant \(C_2\) (Req 2)

Fall 2014
Matrix Arbiter Example (5)

Request 0

Request 1

Request 2

Disable 0

Disable 1

Disable 2

Grant 0

Grant 1

Grant 2

Set column 2, clear row 2
Grant Request A_2
Allocators

• Arbiter assigns a single resource to one of a group of requesters
• Allocator performs a matching between a group of resources and a group of requestors
  – Each of which may request one or more resources
• 3 rules
  – A grant can be asserted only if the corresponding request is asserted
  – At most one grant for each input (requester) may be asserted
  – At most one grant for each output (resource) can be asserted
Allocation Example

• Request Matrix, \( R = \)

\[
\begin{pmatrix}
1 & 1 & 1 \\
1 & 1 & 0 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{pmatrix}
\]

G1 =

G2 =

• Both G1 and G2 satisfy rules but G2 is more desirable
  – All three resources assigned to inputs
  – Maximum matching: solution containing maximum possible number of assignments
Exact Algorithms

• Allocation problem can be represented as a bipartite graph
• Exact algorithms not feasible in time budget of router
• Useful to compare a new heuristic against
Augmenting Paths Algorithm

• Start with sub-optimal matching of the bipartite graph

• Construct a directed residual graph
  – If an edge is in the current matching, M, it points from its output to input

• Now augmenting path is found
  – Any directed path through the residual graph from an unmatched input to an unmatched output
  – Matching is updated using this path
Augmenting Paths Example

R =
\[
\begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 \\
\end{pmatrix}
\]
Augmenting Paths Example

$R = \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 \\
\end{pmatrix}$
Augmenting Paths Example

R =

\[
\begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 \\
\end{pmatrix}
\]
Augmenting Paths Example

\[ R = \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 \\
\end{pmatrix} \]
Augmenting Paths Example

\[ R = \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0
\end{pmatrix} \]
Augmenting Paths Example

\[ R = \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 
\end{pmatrix} \]
Augmenting Paths Example

R =
\[
\begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0
\end{pmatrix}
\]
Wavefront Allocator

• Arbitrates among requests for inputs and outputs simultaneously

• Row and column tokens granted to diagonal group of cells

• If a cell is requesting a resource, it will consume row and column tokens
  – Request is granted

• Cells that cannot use tokens
  – Pass row tokens to right and column tokens down
Wavefront Allocator Example

Tokens inserted at P0

A requesting Resources 0, 1, 2

B requesting Resources 0, 1

C requesting Resource 0

D requesting Resources 0, 2

Entry [0,0] receives grant, consumes token

Remaining tokens pass down and right

[3,2] receives 2 tokens and is granted
Wavefront Allocator Example

[1,1] receives 2 tokens and granted

All wavefronts propagated
Separable Allocator

• Need for pipelineable allocators

• Allocator composed of arbiters
  – Arbiter chooses one out of N requests to a single resource

• Separable switch allocator
  – First stage: select single request at each input port
  – Second stage: selects single request for each output port
A 3:4 allocator
First stage: 3:1 – ensures only one grant for each input
Second stage: 4:1 – only one grant asserted for each output
Separable Allocator Example

- Input-first allocator
- 4 requestors, 3 resources
- Arbitrate locally among requests
  - Local winners passed to second stage

Requestor 1 wins A
Requestor 4 wins C
Adaptive Routing & Allocator Design

• Deterministic routing
  – Single output port
  – Switch allocator bids for output port

• Adaptive routing
  – Returns multiple candidate output ports
    • Switch allocator can bid for all ports
    • Granted port must match VC granted
  – Return single output port
    • Reroute if packet fails VC allocation
Speculative VC Router

• Non-speculative switch requests must have higher priority than speculative ones
  – Two parallel switch allocators
    • 1 for speculative
    • 1 for non-speculative
    • From output, choose non-speculative over speculative

  – Possible for flit to succeed in speculative switch allocation but fail in virtual channel allocation
    • Done in parallel
    • Speculation incorrect
      – Switch reservation is wasted

  – Body and Tail flits: non-speculative switch requests
    • Do not perform VC allocation → inherit VC from head flit
Microarchitecture Summary

- Ties together topological, routing and flow control design decisions

- Pipelined for fast cycle times

- Area and power constraints important in NoC design space