On-Chip Networks II: Router Microarchitecture & Routing

Tushar Krishna
Associate Professor @ Georgia Tech
Visiting Professor @ MIT EECS and CSAIL
Interconnection Network Architecture

- **Topology**: How to connect the nodes up? (processors, memories, router line cards, ...)

- **Routing**: Which path should a message take?

- **Flow control**: How is the message actually forwarded from source to destination?

- **Router microarchitecture**: How to build the routers?

- **Link microarchitecture**: How to build the links?
Recap: Modern on-chip networks

Core will not be shown explicitly in the rest of the slides. Only the routers will be.
Recap: Topology

- **Bus**
- **Ring**
- **Crossbar**
- **Switch**
- **Mesh**
- **Torus**
- **Hierarchical Rings**
- **Concentrated Mesh**
Today’s Agenda

- **Topology**: How to connect the nodes up? (processors, memories, router line cards, ...)

- **Routing**: Which path should a message take?

- **Flow control**: How is the message actually forwarded from source to destination?

- **Router microarchitecture**: How to build the routers?

- **Link microarchitecture**: How to build the links?
Routing
Routing

- Once topology is fixed, routing determines exact path from source to destination
- Analogous to the series of road segments from source to destination
Routing Algorithms

• Property
  – Minimal or Non-Minimal
    • Minimal: only select shortest paths
    • Non Minimal: need not select shortest paths

  – Oblivious or Adaptive
    • Oblivious: routing decisions do not depend on network state
      (i.e., traffic), only depends on (src, dest)
      – Deterministic is a subset where is always chosen
    • Adaptive: uses different routes depending on traffic

• Design Considerations
  – Deadlock Freedom
    • traffic pattern should not lead to a situation where no packets
      move forward

  – Implementation
    • Table-based or combination circuit
Dimension Ordered Routing

**XY: Always go X first, then Y**

Cons of this approach?
- Eliminates any path diversity provided by topology
- Poor load balancing

And yet ... This is the most common approach!
Valiant’s Routing Algorithm

- To route from $s$ to $d$
  - Randomly choose intermediate node $d'$
  - Route* from $s$ to $d'$ (Phase I), and $d'$ to $d$ (Phase II)

- Pros
  - Randomizes any traffic pattern
    - All patterns appear uniform random
  - Balances network-load
    - Higher throughput

- Cons
  - Non-minimal
    - Higher latency and energy
  - Destroys locality

Non-Minimal and *Oblivious

*can also be Adaptive
ROMM: Randomized, Oblivious Multi-phase Minimal Routing

- Confine intermediate node to be within minimal quadrant
- Retain locality + some load-balancing
- This approach essentially translates to randomly selecting between all minimal paths from source to destination
Challenges with Minimal + Oblivious

What happens if you use both simultaneously?
Suppose we toss a coin and send either XY or YX

Benefits?
Challenge?
Network Deadlock

- Flow A holds u and wants v
- Flow B holds v and wants w
- Flow C holds w and wants x
- Flow D holds x and wants u
Turn Model (Glass and Ni 1994)

- 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
  - Removing some turns can make algorithm deadlock free

XY Model

YX Model
Deadlock-free Routing Algorithms

West-First Turn Model

North-Last Turn Model

Negative-First Turn Model
Can we eliminate any 2 turns?

Six turn model

Deadlock!
Channel Dependency Graph (CDG)

- Vertices represent network links (channels)
- Edges represent turns
  - $180^\circ$ turns not allowed, e.g., $AB \rightarrow BA$
Cycles in the 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
Deadlock!

Edges in CDG = Turns in Network
→ Disallow/Delete certain edges in CDG
Acyclic CDG

This is the West-first turn model!

Disable certain edges

Cyclic CDG

Acyclic CDG
Path Diversity vs Deadlock

- Path diversity required for higher throughput
- Path restrictions because of deadlock-free routing requirement

- Can we allow all turns and still get deadlock freedom?
Why do deadlocks occur?

Resource conflicts! (i.e., structural hazard!)

- Flow A holds buffer in 1 and wants buffer in 2
- Flow B holds buffer in 2 and wants buffer in 3
- Flow C holds buffer in 3 and wants buffer in 0
- Flow D holds buffer in 0 and wants buffer in 1

Add more buffers and partition!
Virtual Channels

- Same physical link/channel between routers
  - additional buffers in each router to avoid deadlocks – called “virtual” channels
Example 1

- Policy: XY in VC0, YX in VC1
Example 2

- Policy: Start in VC0, after Dateline jump to VC1
Escape Virtual Channels

- **Policy:**
  - Allow any turns across all VCs except one
    - "Escape" VC → deadlock-free route
  - If there is a deadlock, can jump into escape VC which is guaranteed to drain
Router
Microarchitecture
Example

- Suppose we have a Ring network
What does each “router” look like?

1. Who should use output link?

2. What to do with the other flit (from ring/core)

Have you seen this same situation in real life on a road network?

Note: only showing anti-clockwise ring for illustration
Link Arbitration

1. Who should use output link?

Traffic already on ring has priority

2. What to do with the other flit (from ring/core)

Wait
Arbitration Protocol

This is known as “arbitration”
The control structure is called an “arbiter”

3. What should a flit do if its output is blocked?

![Diagram of arbitration protocol with input and output lanes, new flit, and full state]

Send input if no traffic on ring
Buffer Management

• What should a flit do if its output is blocked?
  – **Option 1:** Drop!
    • Send a NACK back for dropped packet or have a timeout
      – Source retransmits
      – Implicit congestion control
    • Flow control protocol on the Internet
  • **Advantage: can be bufferless!**
  • Challenges?
    – Latency and energy overhead of re-transmitting more than that of buffering so not preferred on-chip
Buffer Management

• What should a flit do if its output is blocked?
  – **Option 2: Misroute!**
    • As long as N input ports and N output ports, can send flit out of some other output port
      – called “bouncing” on a ring
    • **Advantage: can be bufferless!**
  • Challenges
    – Energy
      » Routes become non-minimal – more energy consumption at router latches and on links
    – Performance
      » Non-minimal routes – can lead to longer delays
    – Correctness
      » Livelock! – cannot *guarantee* forward progress
        » Not the same as deadlock
        » *Need to restrict number of misroutes of same packet*
Buffer Management

- What should a flit do if its output is blocked?
  - **Option 3**: Wait!
  - Signal to previous router to not send any more flits till the input at this router can be drained
  - **Backpressure** techniques
    - On/Off: one bit to signal if next router can receive or not
      » Challenge: Delay of on/off signal
    - Credit-based: A count of how many flits can be sent to the next node?
      » When should credit be decremented?
        Whenever a flit is sent to the next router
      » When should credit be incremented?
        Whenever a flit leaves the next router
More general topology
What’s Inside 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

VC Allocator

SW Allocator

Input Buffers

VC 1
VC 2
VC n

VC Allocator

SW Allocator

Input Buffers

VC 1
VC 2
VC n

Crossbar Switch

BW: Buffer Write
RC: Route Compute
VA: VC Allocation
Input VCs arbitrate for “output” VCs (Input VCs at next router)
SA: Switch Allocation
Input ports arbitrate for output ports
BR: Buffer Read
ST: Switch Traversal
LT: Link Traversal

FLIT
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

**Key Points:***
- Route computation performed once per packet
- Virtual channel allocated once per packet
- Body and tail flits inherit this info from head flit

![Baseline Router Pipeline Diagram](image)

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

Head Flit (packet A)

Tail Flit (packet B - holds VC)

Body Flit (packet A)

Cycle

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>SA</th>
<th>ST</th>
<th>LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td>NRC</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

<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 SA requests

Today: 1-2 cycles per router
Additional Self Reading: Evaluating NoCs
Network Latency

\[ T_N = (t_r + t_l) \times H + T_C + T_S \]

**Which of these is static?**

\[ t_r \quad t_w \quad T_s \]

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

\[ H \quad T_c \]
Evaluating NoCs

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

Offered Traffic (bits/sec)
Open Research questions in NoCs

• “Best” *on-chip* topology
  - Uniform vs Hierarchical
  - Few routers with more ports (“High-Radix”) or more routers with few ports (“Low-Radix”)

• NoCs with unconventional interconnects
  - Photonic, RF, wireless

• Resilient NoCs
  - How to deal with run-time failures of links and routers

• NoCs for heterogeneous SoCs
  - Smartphones, IoT

• NoCs for Accelerators
  - NoCs for FPGAs
  - NoCs for deep learning accelerators
  - NoCs for database accelerators
  - NoCs for graph processing accelerators

Surge of research in last few years
Thank you!

Next Lecture: VLIW