Quality-of-Service for a High-Radix Switch

Nilmini Abeyratne, Supreet Jeloka, Yiping Kang, David Blaauw, Ronald G. Dreslinski, Reetuparna Das, and Trevor Mudge

University of Michigan

51st DAC
06/05/2014
Motivation

- Network traffic from modules can have different bandwidth and latency requirements.

**Quality-of-Service mechanisms regulate network access such that all applications/modules meet their communication needs without degrading the performance of others.**
Quality-of-Service in Networks

- QoS is characterized by
  - Bandwidth
  - Latency

- QoS is provided by
  - Grouping traffic flows into classes with different service levels
  - Service levels enforced by QoS algorithms
Virtual Clock QoS Algorithm

- Emulates Time Division Multiplexing (TDM)
  - Each flow has its own virtual time space
  - Packets are serviced at virtual time slots

- Advantages
  - Fixed bandwidth guarantees
  - Redistributes unused bandwidth
  - Supports variable-length packets
  - No global synchronization across modules necessary

- Disadvantages
  - Per-flow queuing and status registers required
QoS for a High-Radix Switch

- More modules → multi-hop network-on-chips
- High-quality QoS in crossbars is simple
  - Per-flow state is only stored at a single crosspoint
- Swizzle Switch crossbar – scalable alternative
  - ISSCC 2012, JETCAS 2012
  - SRAM-style design
  - Arbitration logic inside crossbar
  - Scales to radix-64, 1.5GHz

Combine simple, high-quality QoS in crossbars with scalability of Swizzle Switch.
In the Paper

- Guaranteed Bandwidth (GB) traffic class
  - Swizzle Switch-Virtual Clock (SSVC)
  - Latency fairness vs. bandwidth rates
- Guaranteed Latency (GL) traffic class
- Details of circuit implementation
- Evaluation methodology and results
Swizzle Switch

- At each crosspoint
  - Least-Recently Granted (LRG) arbitration logic
  - (N-1)-bit LRG priority value

- Guaranteed Bandwidth (GB) Traffic Class
  - Swizzle Switch-Virtual Clock (SSVC)
  - Add to each crosspoint
    - VC counter (QoS priority)
    - QoS arbitration logic
Example

<table>
<thead>
<tr>
<th>8x8 SS</th>
<th>VC</th>
<th>LRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>In0</td>
<td>110</td>
<td>2</td>
</tr>
<tr>
<td>In1</td>
<td>110</td>
<td>4</td>
</tr>
<tr>
<td>In2</td>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td>In3</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>In4</td>
<td>001</td>
<td>7</td>
</tr>
<tr>
<td>In5</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>In6</td>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>In7</td>
<td>111</td>
<td>6</td>
</tr>
</tbody>
</table>

Create priorities grid:
1. Expand VC priority
2. Place LRG priority @ 1→0 transitions

Two sets of priorities at each crosspoint:
1. Primary QoS priority (VC counter value)
2. Secondary LRG priority (break ties)
Example

Two sets of priorities at each crosspoint:
1. Primary QoS priority (VC counter value)
2. Secondary LRG priority (break ties)

Create priorities grid:
1. Expand VC priority
2. Place LRG priority @ 1→0 transitions
   a) Replace all 1’s with min priority
Create priorities grid:
1. Expand VC priority
2. Place LRG priority @ 1→0 transitions
   a) Replace all 1’s with min priority
   b) Replace all 0’s with max priority

Two sets of priorities at each crosspoint:
1. Primary QoS priority (VC counter value)
2. Secondary LRG priority (break ties)
Create priorities grid:
1. Expand VC priority
2. Place LRG priority @ 1→0 transitions
   a) Replace all 1’s with min priority
   b) Replace all 0’s with max priority
3. Zero non-requesting inputs

Two sets of priorities at each crosspoint:
1. Primary QoS priority (VC counter value)
2. Secondary LRG priority (break ties)
Two sets of priorities at each crosspoint:
1. Primary QoS priority (VC counter value)
2. Secondary LRG priority (break ties)

Create priorities grid:
1. Expand VC priority
2. Place LRG priority @ 1→0 transitions
   a) Replace all 1’s with min priority
   b) Replace all 0’s with max priority
3. Zero non-requesting inputs
Example

8x8 SS  VC

L0  L1  L2  L3  L4  L5  L6  L7

0  0  0  0  0  0  0  7

0  0  0  0  0  1  7  7

0  0  0  0  5  7  7  7

Priorities grid allows fast, single-cycle arbitration!

In0  (0,M)  110  Senses Lane 6
In1  (1,M)  110  Senses Lane 6
In2  (2,M)  100  Senses Lane 4
In3    000
In4    001
In5  (5,M)  100  Senses Lane 4
In6  (6,M)  100  Senses Lane 4
In7    111

Wins!

Priorities grid complete! Perform arbitration!
Results: GB Class w/ SSVC

**No QoS – LRG Only**

**Assumption:** all packets are 8-flits
Max. throughput = 0.89 flits/cycle

Without QoS, all flows received equal bandwidth.
Results: GB Class w/ SSVC

No QoS – LRG Only

With QoS – SSVC

Accepted Throughput at Output (flits/input)

Flows 1, 2, 3, 4, 5, 6, 7, 8

Flow 1 (r = 0.4)

Flow 2 (r = 0.2)

Flows 3, 4 (r = 0.1)

Flows 5, 6, 7, 8 (r = 0.05)
Results: Area and Delay

- Evaluated several switch configurations
  - Radices: 8, 16, 32, 64
  - Bus widths: 128, 256, 512

- Worst-case overhead
  - Storage [radix-64]: 1MB
  - Switch area increase [128-bit bus]: 2%
  - Switch frequency decrease [radix-8]: 8.4%
Summary

- QoS manages bandwidth & latency requirements of network traffic flows
- Virtual Clock provides bandwidth guarantees with flexibility
- SSVC: combines simplicity of QoS in crossbars with scalability of Swizzle Switch

Come to the poster session!
THANK YOU!