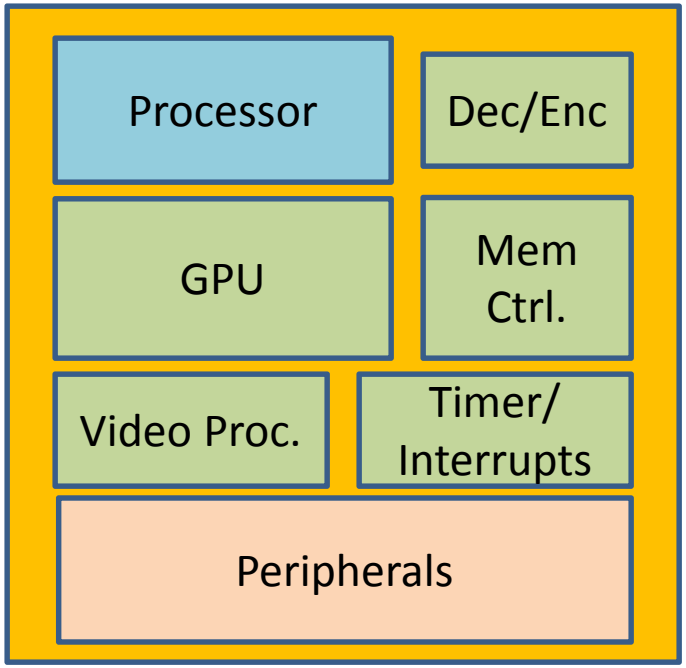
Quality-of-Service for a High-Radix Switch

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



System-on-Chip (SoC)

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

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

Quality-of-Service in Networks

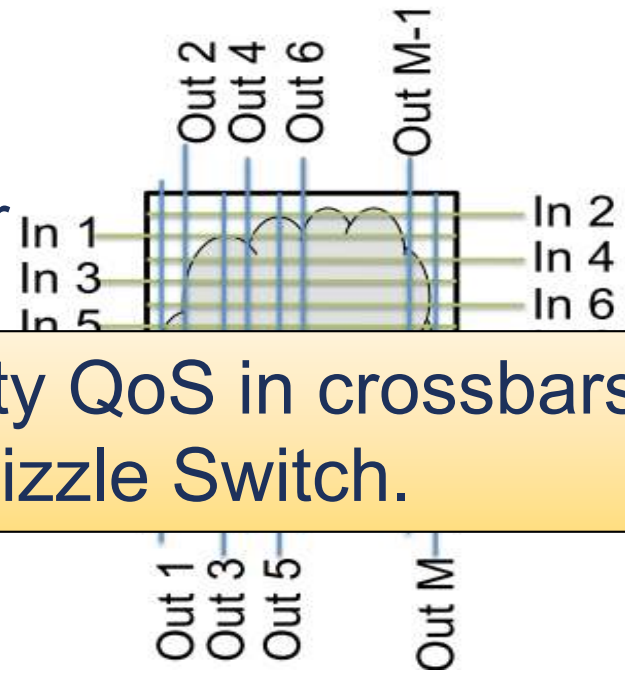
- QoS is characterized by
  - Bandwidth
  - Latency
- QoS is provided by
  - Grouping traffic flows into classes with different service levels
  - Service levels enforced by QoS algorithms

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

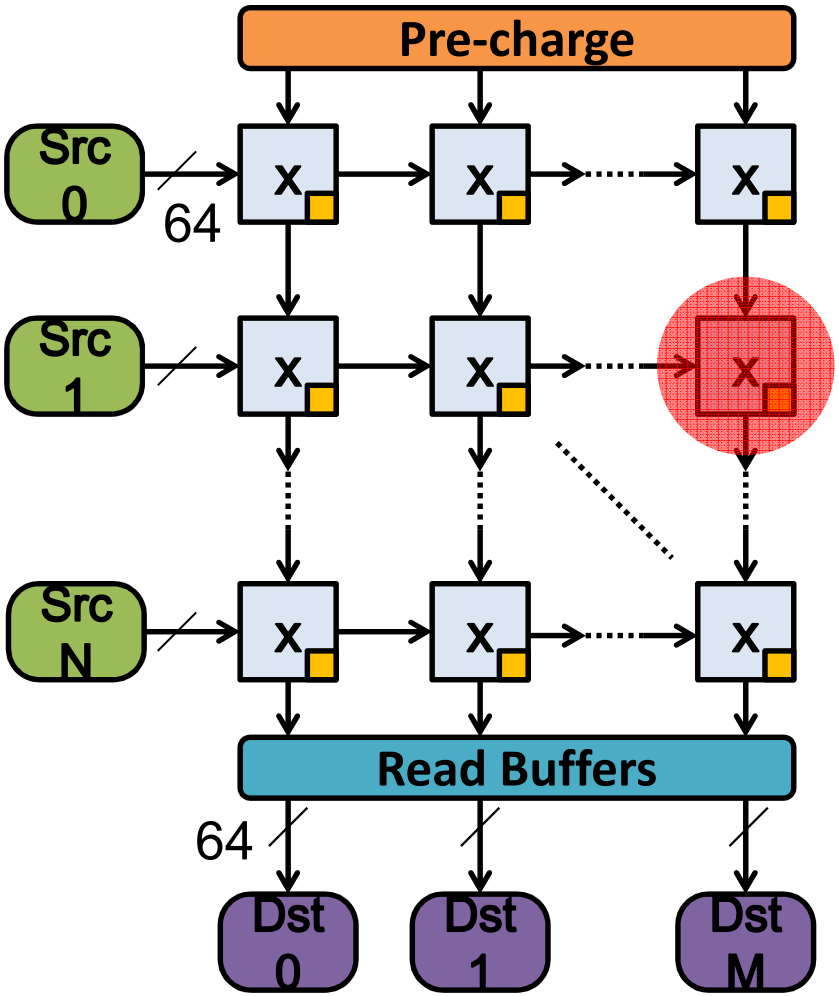


Combine simple, high-quality QoS in crossbars with scalability of Swizzle Switch.

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

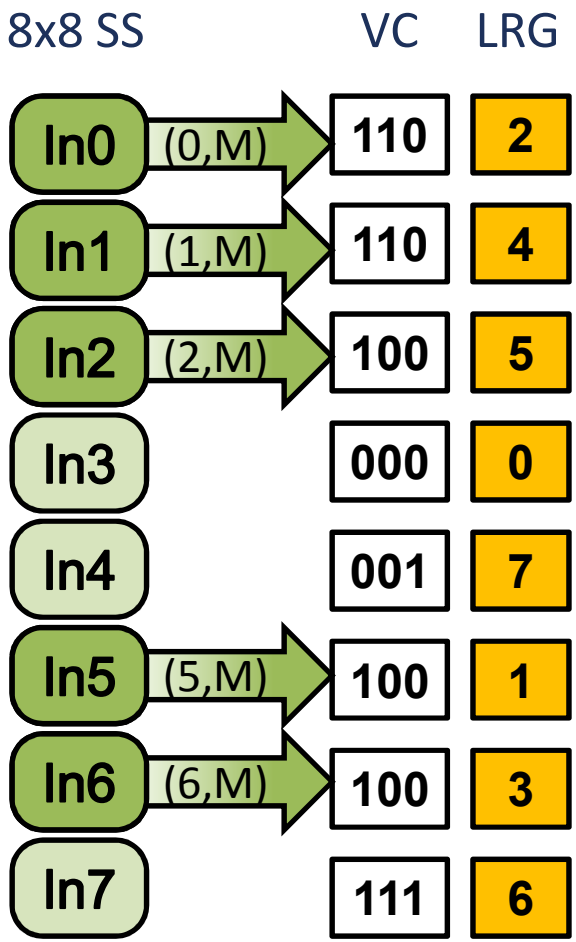
Swizzle Switch

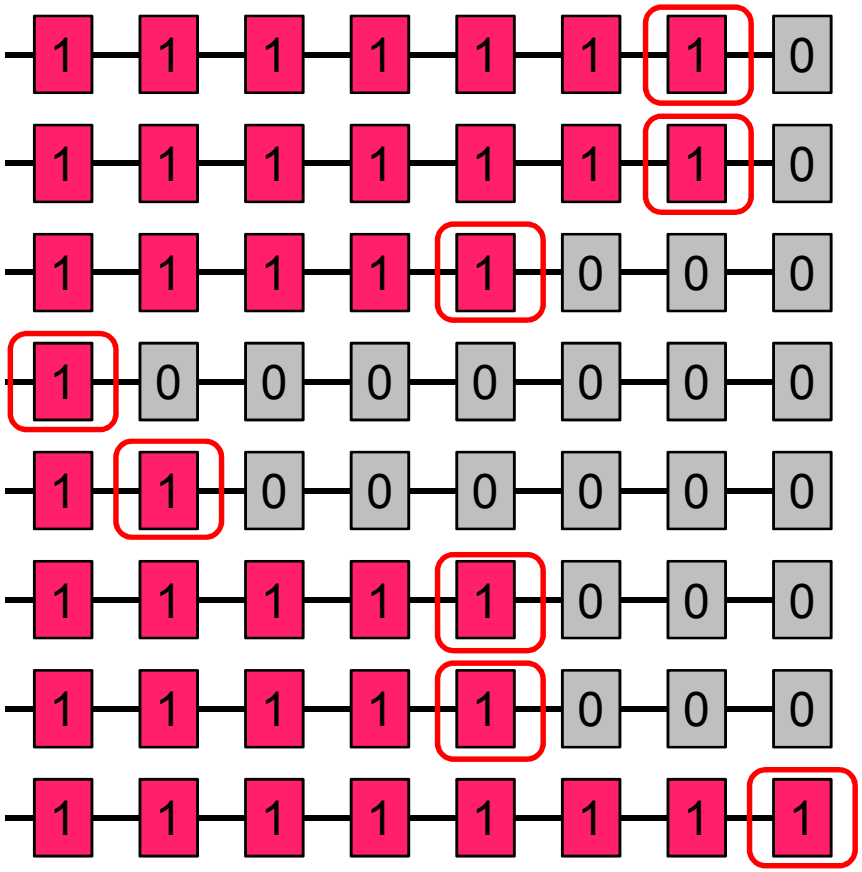


- At each crosspoint
  - Least-Recently Granted (LRG) arbitration logic
  - (N-1)-bit LRG priority value
- Guaranteed Bandwidth (GB) Traffic Class
  - *Swizzle Switch-Virtual Clock* (*SSVC*)
  - Add to each crosspoint
    - VC counter (QoS priority)
    - QoS arbitration logic

Example

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



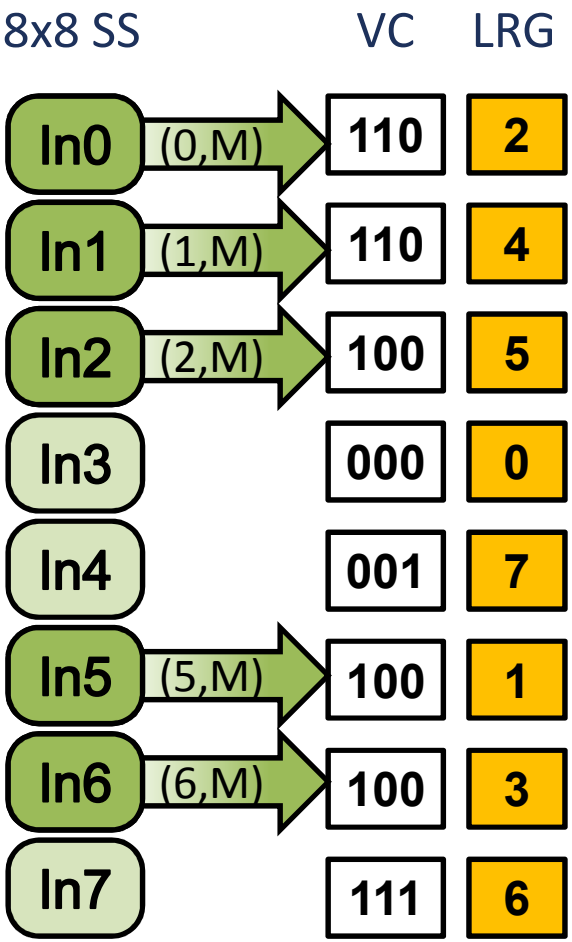


Two sets of priorities at each crosspoint:
1. Primary QoS priority (VC counter value)
2. Secondary LRG priority (break ties)

Example

Create priorities grid:

1. Expand VC priority
2. Place LRG priority @ 1→0 transitions
   a) Replace all 1's with *min* priority



1	1	1	1	1	1	2	0
1	1	1	1	1	1	4	0
1	1	1	1	5	0	0	0
0	0	0	0	0	0	0	0
1	7	0	0	0	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	3	0	0	0
1	1	1	1	1	1	1	6

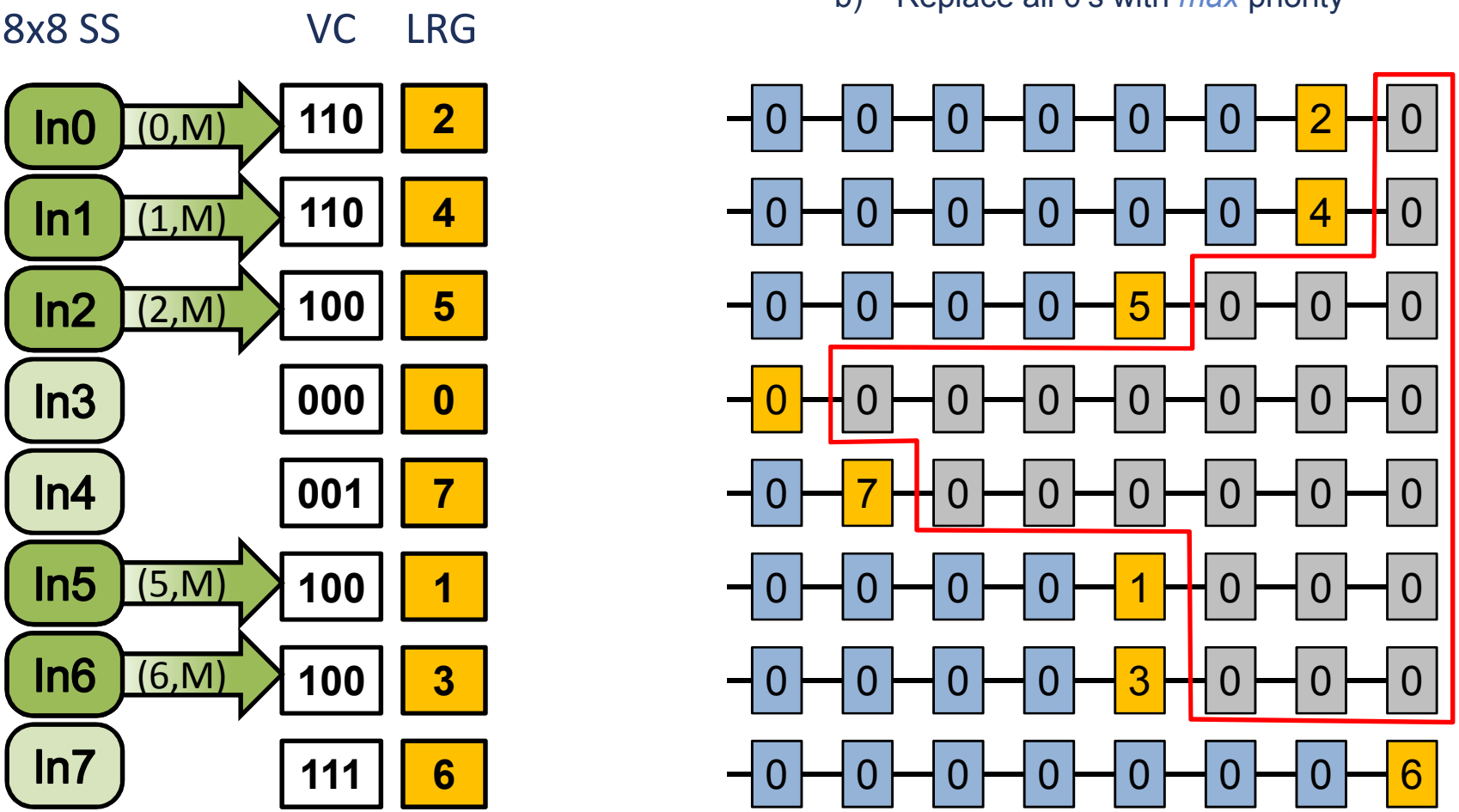
Two sets of priorities at each crosspoint:

1. Primary QoS priority (VC counter value)
2. Secondary LRG priority (break ties)

Example

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)

Example

Create priorities grid:

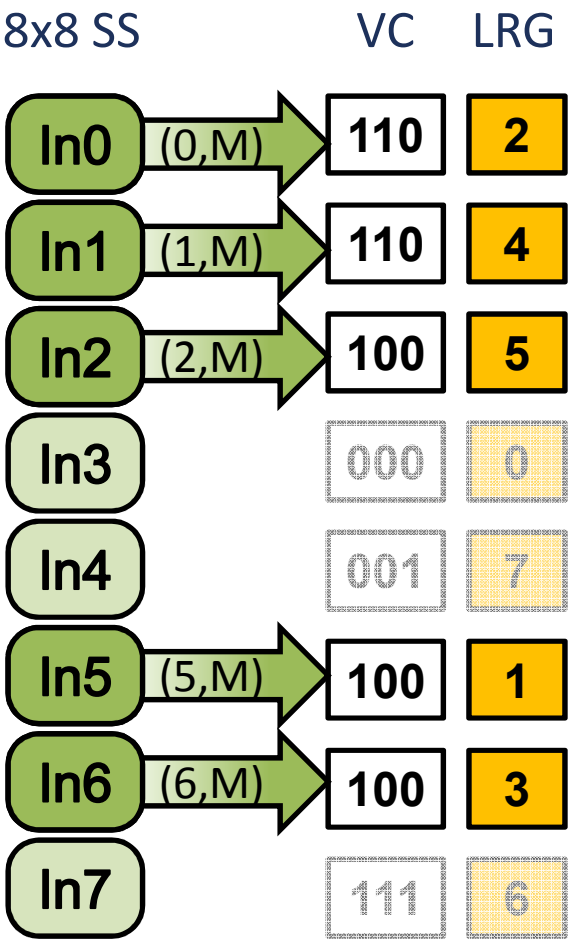
1. Expand VC priority
2. Place LRG priority @ 1→0 transitions
   a) Replace all 1's with *min* priority
   b) Replace all 0's with *max* priority
3. Zero non-requesting inputs

8x8 SS		VC	LRG								
In0	(0,M)	110	2	0	0	0	0	0	0	2	7
In1	(1,M)	110	4	0	0	0	0	0	0	4	7
In2	(2,M)	100	5	0	0	0	0	5	7	7	7
In3		000	0	0	7	7	7	7	7	7	7
In4		001	7	0	7	7	7	7	7	7	7
In5	(5,M)	100	1	0	0	0	0	1	7	7	7
In6	(6,M)	100	3	0	0	0	0	3	7	7	7
In7		111	6	0	0	0	0	0	0	0	6

Two sets of priorities at each crosspoint:

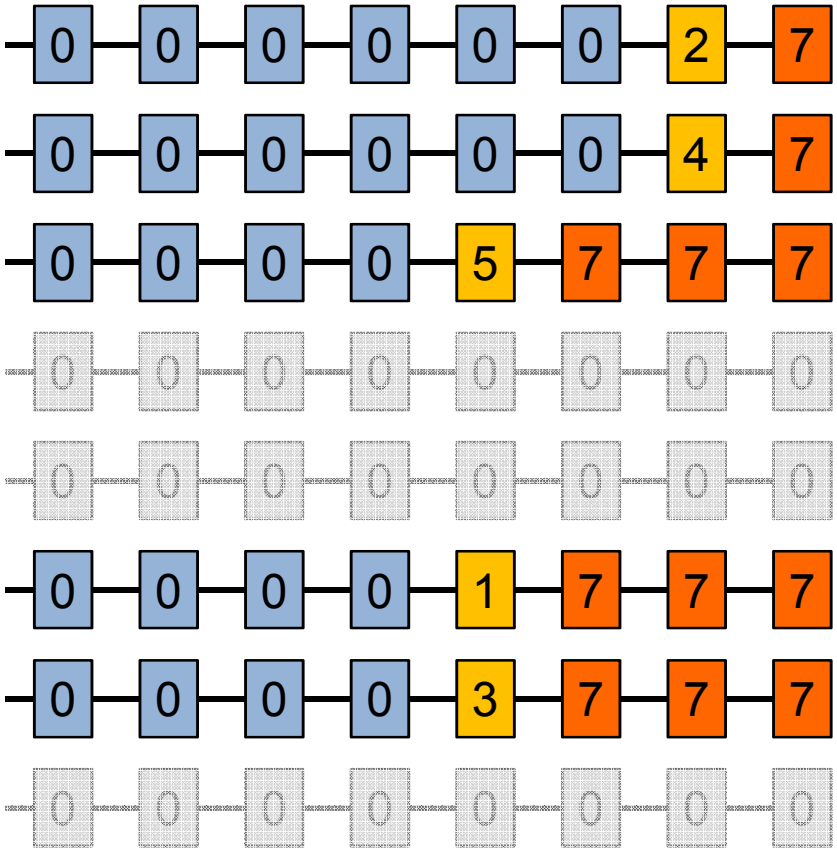
1. Primary QoS priority (VC counter value)
2. Secondary LRG priority (break ties)

Example



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

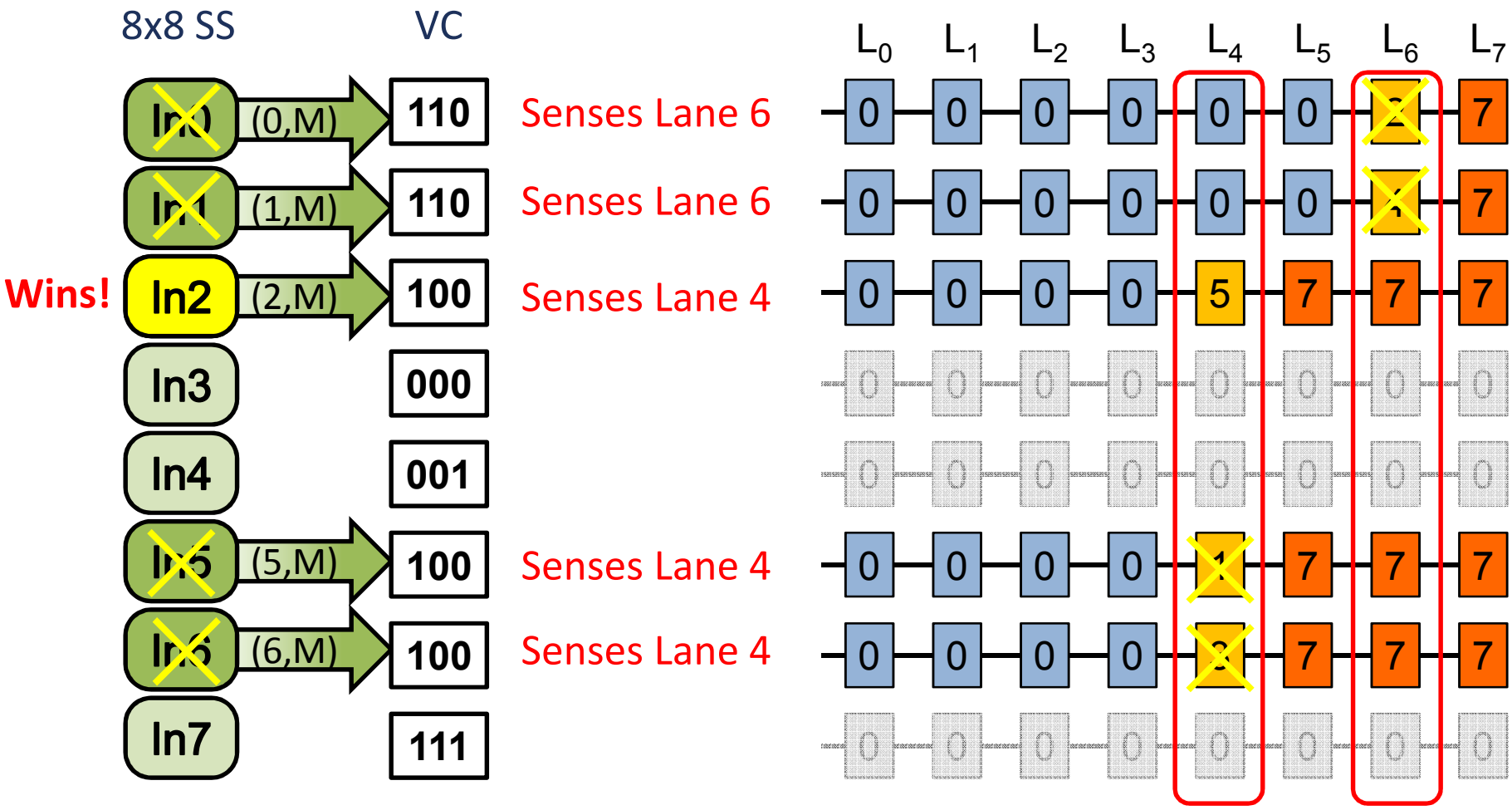


Two sets of priorities at each crosspoint:

1. Primary QoS priority (VC counter value)
2. Secondary LRG priority (break ties)

Example

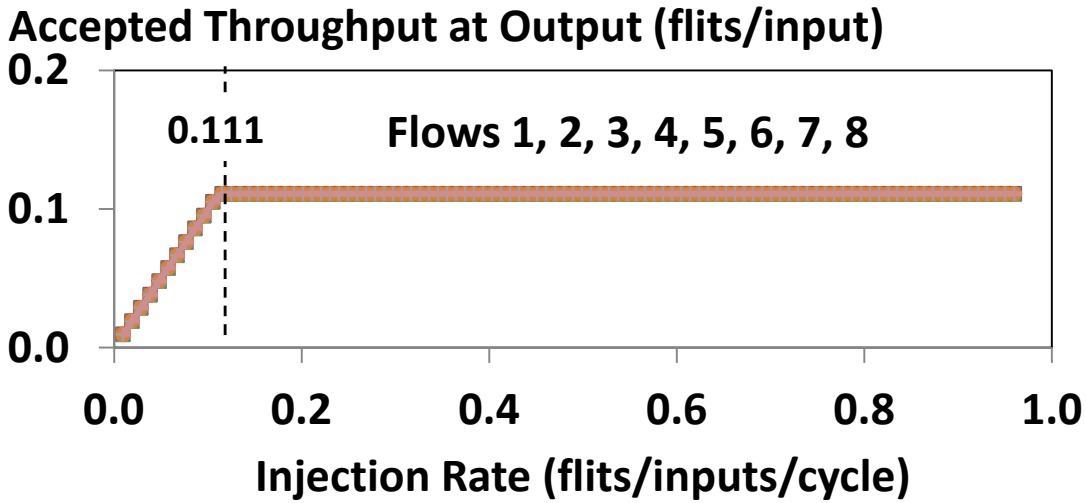
Priorities grid complete!
Perform arbitration!



Priorities grid allows fast, single-cycle arbitration!

Results: GB Class w/ *SSVC*

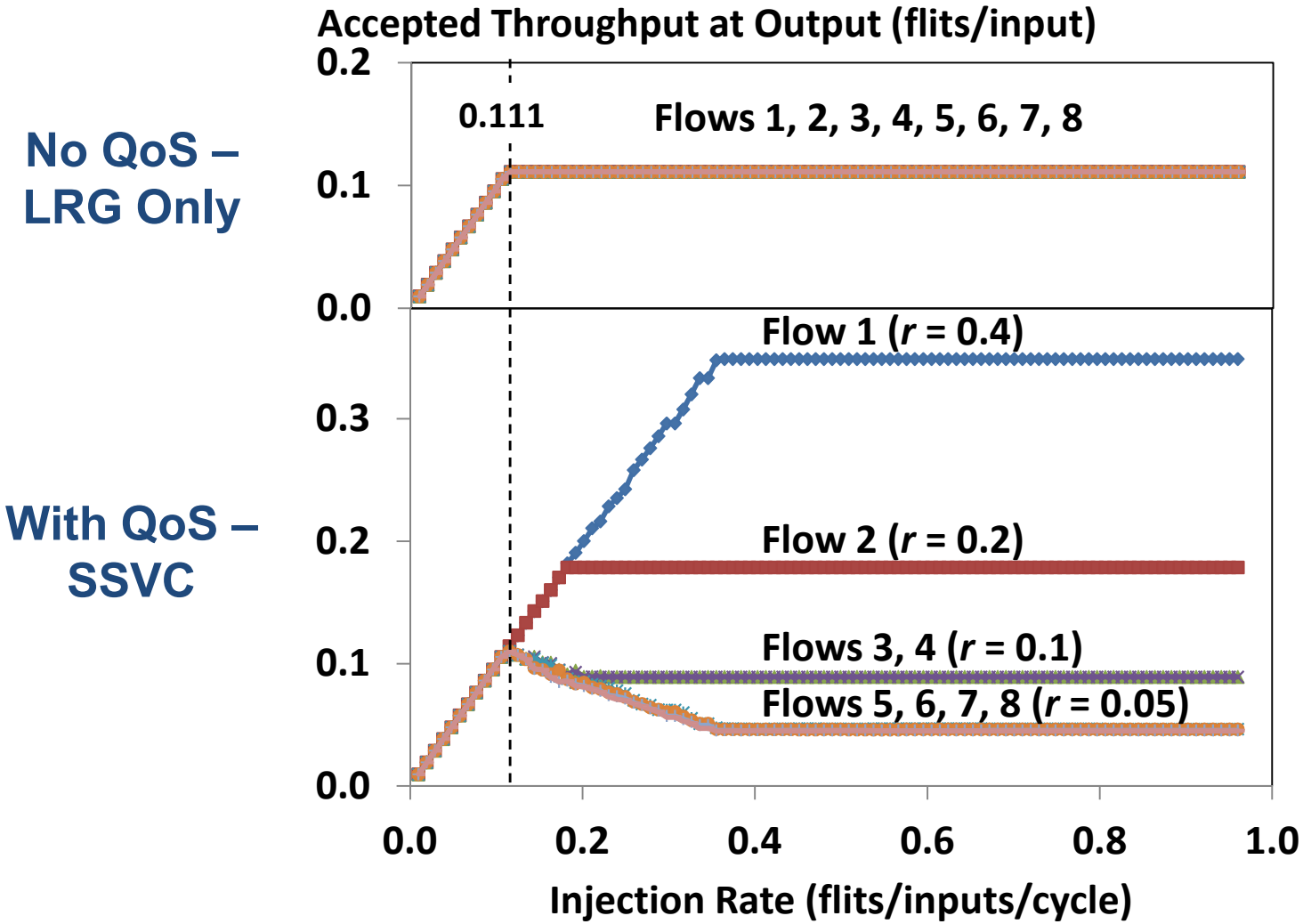
No QoS – LRG Only



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

Without QoS, all flows received equal bandwidth.

Results: GB Class w/ *SSVC*



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!