Swizzle Switch: A Self-Arbitrating High-Radix Crossbar for NoC Systems

Ronald Dreslinski, Korey Sewell, Thomas Manville, Sudhir Satpathy, Nathaniel Pinckney, Geoff Blake, Michael Cieslak, Reetuparna Das, Thomas Wenisch, Dennis Sylvester, David Blaauw, and Trevor Mudge
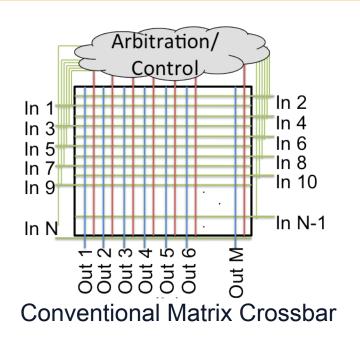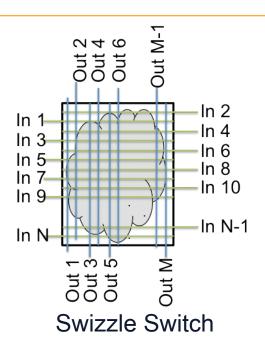
University of Michigan

Outline

- Swizzle Switch—Circuit & Microarchitecture
  - Overview
  - Arbitration
  - Prototype
- Swizzle Switch—Cache Coherent Manycore Interconnect
  - Motivation & Existing Interconnects
  - Swizzle Switch Interconnect
  - Evaluation

Swizzle Switch

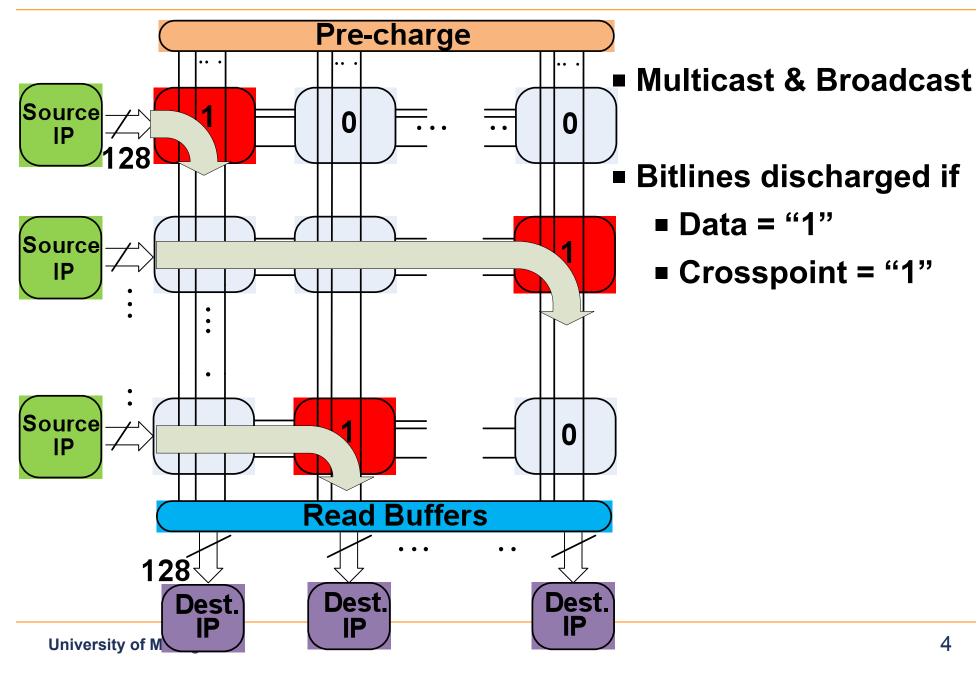Conventional Matrix Crossbar

Swizzle Switch

- Embeds arbitration within crossbar—single cycle arbitration

- Re-use input/output data buses for arbitration

- SRAM-like layout with priority bits at cross-points

- Low-power optimizations

- Excellent scalability
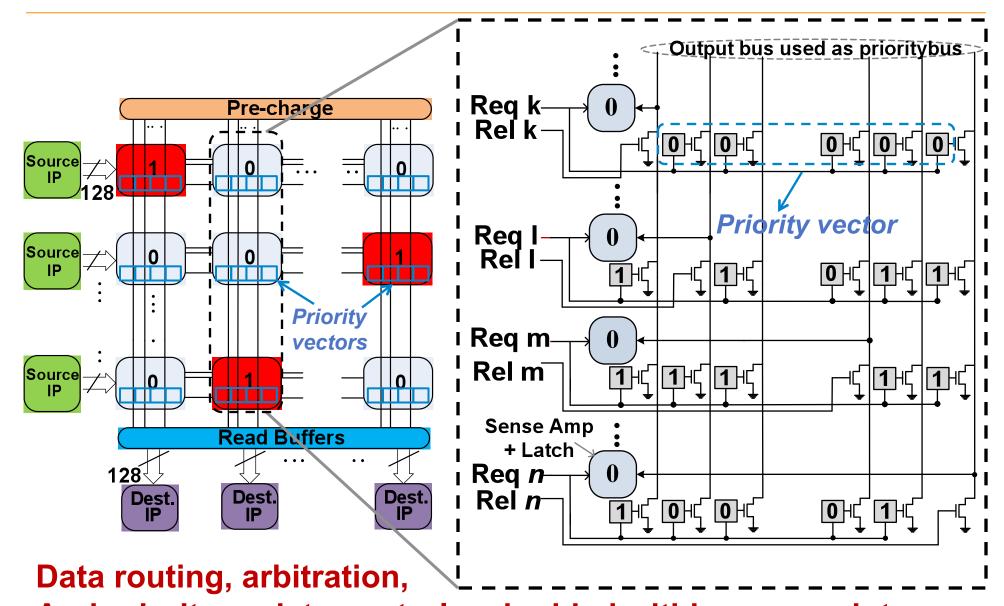
Data Routing

- **Multicast & Broadcast**

- **Bitlines discharged if**
  - **Data = "1"**
  - **Crosspoint = "1"**

Swizzle Switch Architecture

Data routing, arbitration,
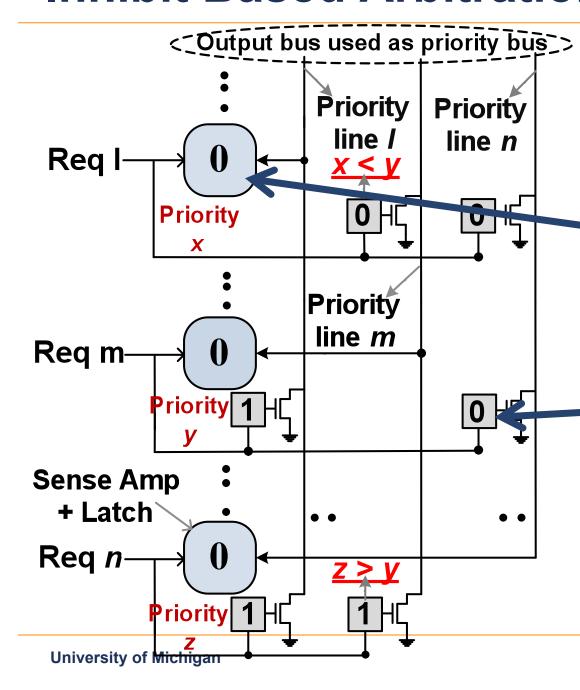And priority update control embedded within crosspoints

Outline

- Swizzle Switch—Circuit & Microarchitecture
  - Overview
  - Arbitration
  - Prototype
- Swizzle Switch—Cache Coherent Manycore Interconnect
  - Motivation & Existing Interconnects
  - Swizzle Switch Interconnect
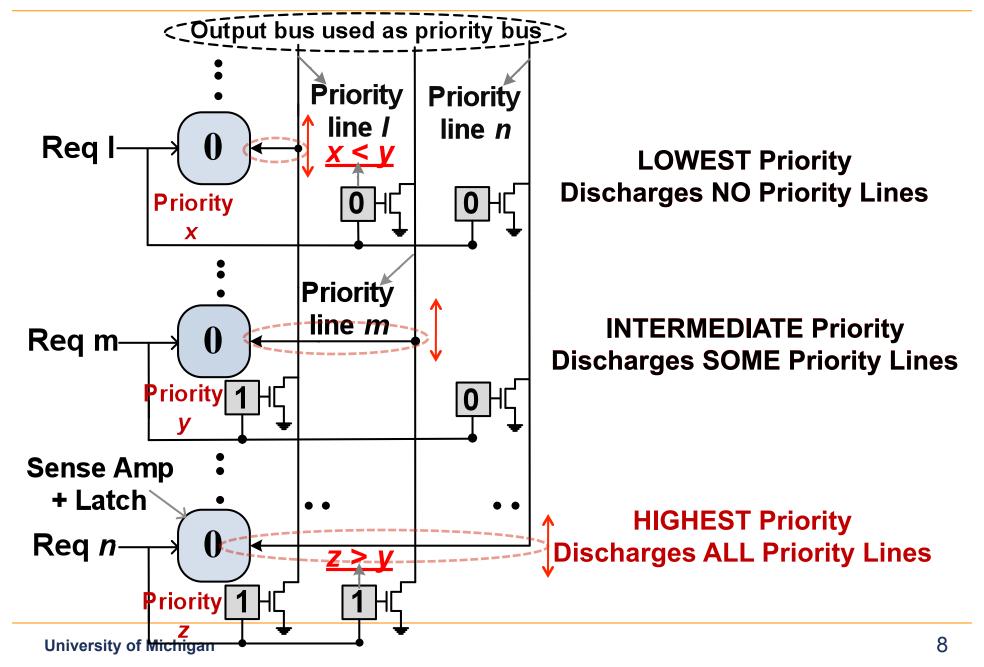 - Evaluation

Inhibit Based Arbitration

This diagram is a single column in the Swizzle-Switch (output), each output arbitrates/transfers data *independently*

Each Crosspoint has a sense amp/latch to indicate connectivity. Each input samples a unique bit of the output bus to determine if it has been granted the channel
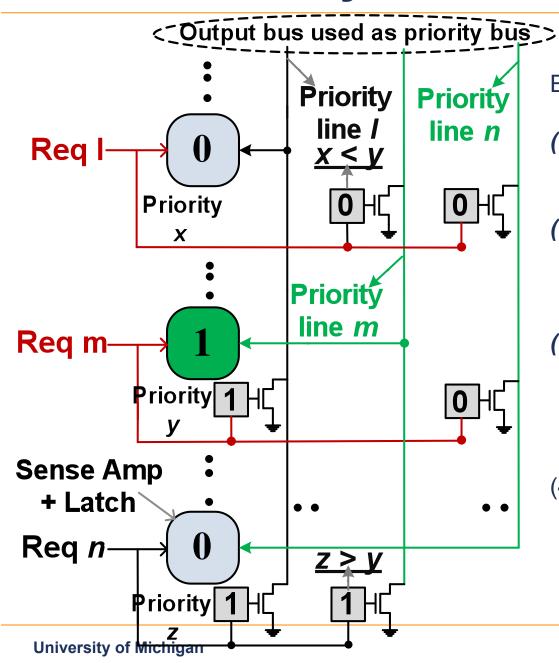
Priority vectors are stored and when a request is issued they discharge bits along the output columns to *INHIBIT* lower priority requests

Finally, the priority vectors are updated when the data transfer completes.

Least Recently Granted(LRG)

Output bus used as priority bus

Priority line *l*
$x < y$

Priority line *n*

Req *l* → 0

Priority *x*

LOWEST Priority
Discharges NO Priority Lines

Priority line *m*

Req *m* → 0

Priority *y* 1

INTERMEDIATE Priority
Discharges SOME Priority Lines

Sense Amp + Latch

Req *n* → 0

$z > y$

Priority *z* 1 1

HIGHEST Priority
Discharges ALL Priority Lines

Least Recently Granted(LRG)

Output bus used as priority bus

Priority line *l* $x < y$

Priority line *n*

Req **l** → 0

Priority *x*

0 | 0

Priority line *m*

Req **m** → 1

Priority *y* 1 | 0

Sense Amp + Latch

Req *n* → 0
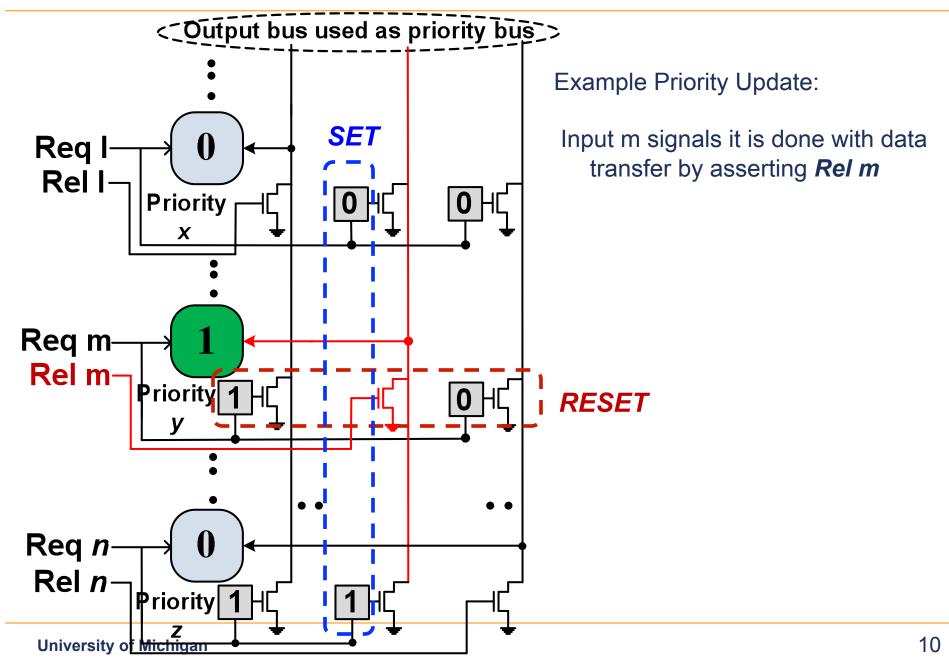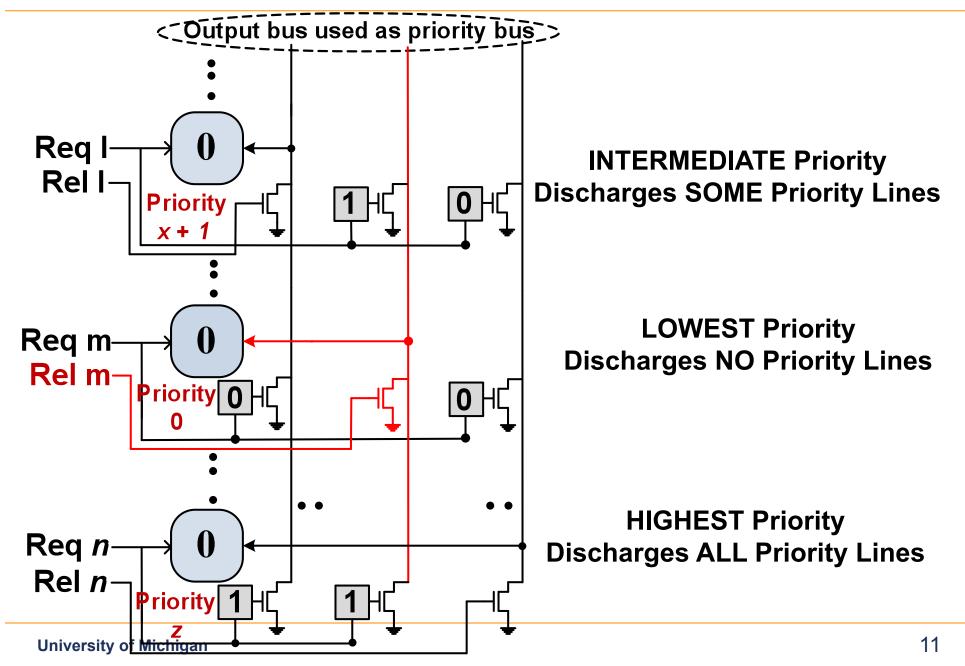
$z > y$

Priority *z* 1 | 1

Example Arbitration:

(1) Req l and *Req m* Request the bus (red lines)

(2) Req m discharges Priority line *l*, priority lines *m* and *n* remain charged (green lines)

(3) Req l senses Priority line *l* and is inhibited (not granted), *Req m* senses Priority line m and is not inhibited

(4) The crosspoint records the connectivity at *input m*

Least Recently Granted(LRG)

Example Priority Update:

Input m signals it is done with data transfer by asserting *Rel m*

Least Recently Granted(LRG)

Least Recently Granted(LRG)

Output bus used as priority bus

Req l
Rel l

0

Priority
x + 1

1   **0**

Increasing Priority

Req m
Rel m

0

Priority
0

0   **0**

Req *n*
Rel *n*

0

Priority
z

1   **1**

LRG Algorithm

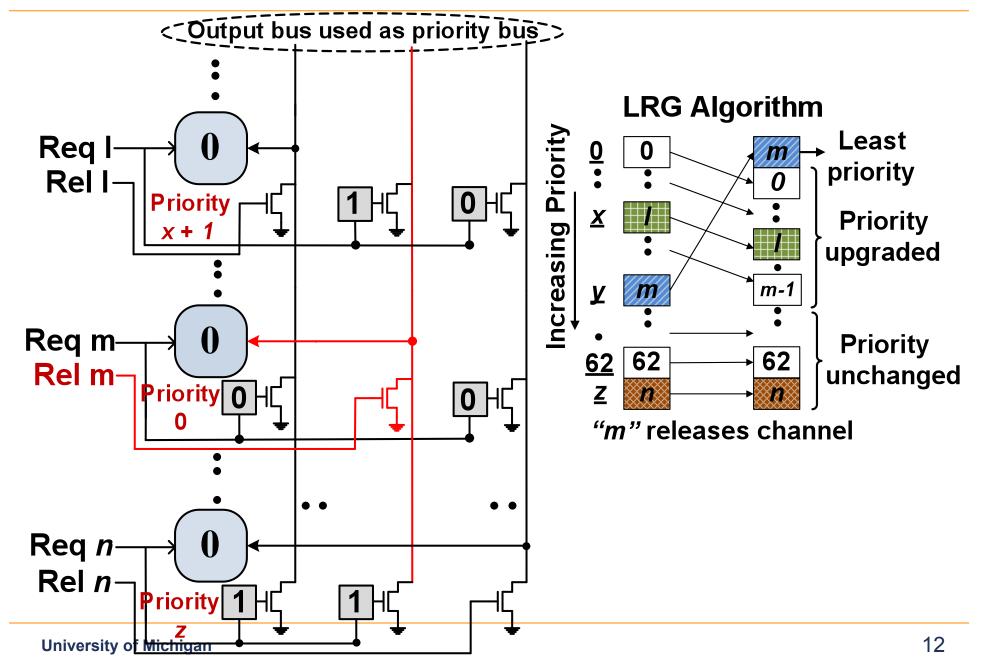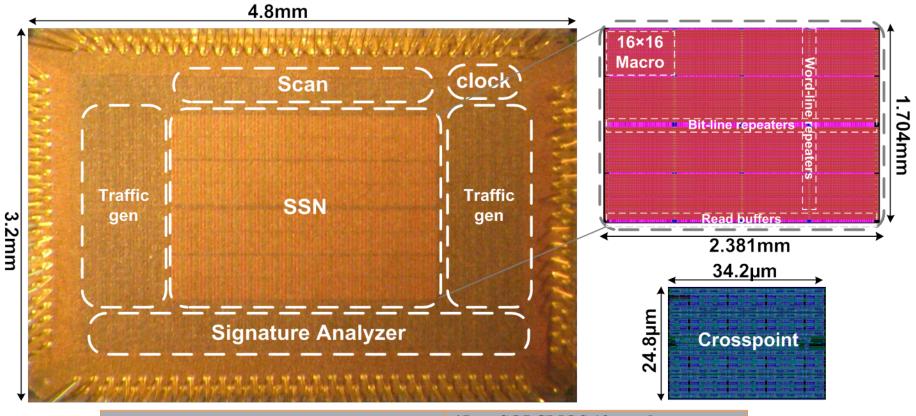|		
0	0	**m** → Least priority
		0
x	*l*	
		l Priority upgraded
y	**m**	*m-1*
62	62	62 Priority unchanged
z	*n*	*n*

"*m*" releases channel

Outline

- Swizzle Switch—Circuit & Microarchitecture
  - Overview
  - Arbitration
  - Prototype
- Swizzle Switch—Cache Coherent Manycore Interconnect
  - Motivation & Existing Interconnects
  - Swizzle Switch Interconnect
  - Evaluation

64x64 Prototype

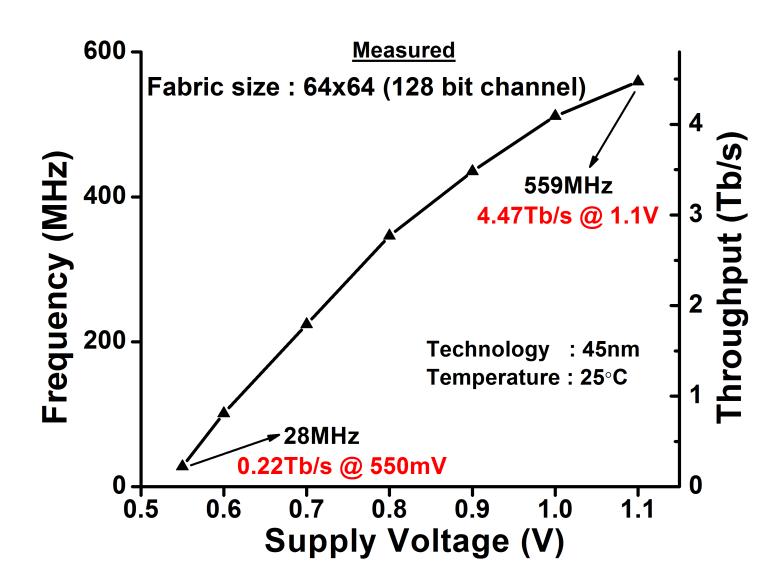
Process	45nm SOI CMOS 12metal interconnect
Die area	15.6mm^2
Fabric area, Transistor count, # Data wires	4.06mm^2, 6.95M, 8192
Throughput, Frequency	4.47Tb/s @ 1.1V, 559MHz, 25°C
Energy Efficiency at peak throughput	3.4Tb/s/W
Peak energy efficiency	7.4Tb/s/W @ 0.6V

Measurement Results

Measured

Fabric size : 64x64 (128 bit channel)

559MHz
4.47Tb/s @ 1.1V

Technology   : 45nm
Temperature : 25°C

28MHz
0.22Tb/s @ 550mV

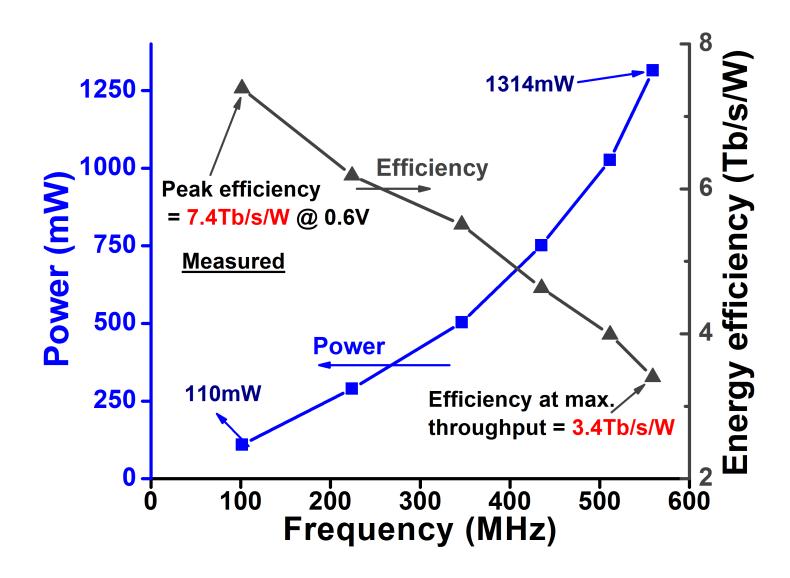# Measurement Results

Outline

- Swizzle Switch—Circuit & Microarchitecture
    - Overview
    - Arbitration
    - Prototype
- Swizzle Switch—Cache Coherent Manycore Interconnect
    - Motivation & Existing Interconnects
    - Swizzle Switch Interconnect
    - Evaluation
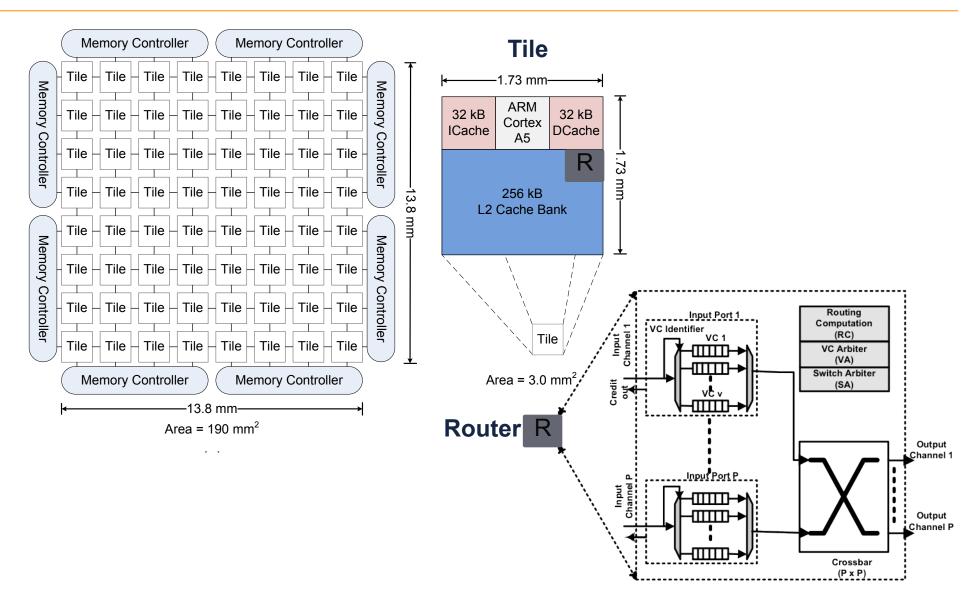
Scaling Interconnect for Many-Cores

- Existing interconnects—Buses, Crossbars, Rings
    - Limited to ~16 cores

- Other's Interconnect proposals for Many-Cores
    - Packet-switched, multi-hop, network-on-chip (NoC)
    - Grid of routers—meshes, tori and flattened butterfly

- **Our Proposal**
    - **Swizzle Switch Networks**
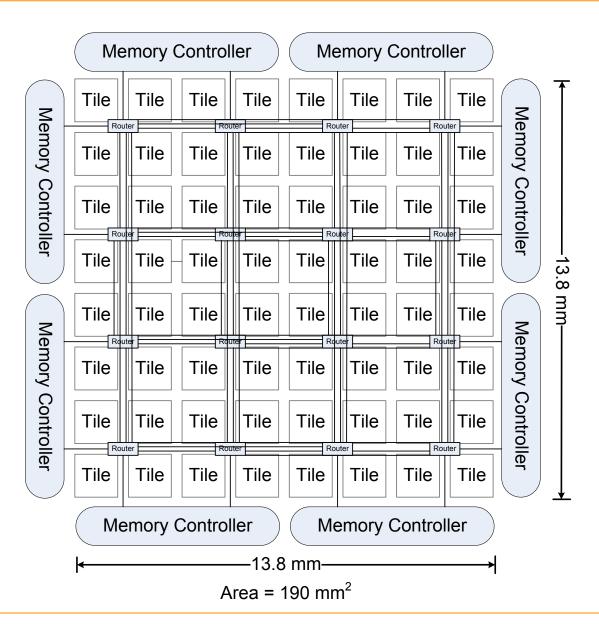   - Flat single-stage, one-hop, crossbar++ interconnect

Mesh Network-on-Chip

Memory Controller Memory Controller

Tile grid (8 × 8)

Memory Controller
Memory Controller
Memory Controller
Memory Controller

Memory Controller Memory Controller

13.8 mm

13.8 mm

Area = 190 mm²

Tile

1.73 mm

| 32 kB ICache | ARM Cortex A5 | 32 kB DCache |

R

256 kB L2 Cache Bank

1.73 mm

Tile

Area = 3.0 mm²

Router R

Input Channel 1

Credit out

VC Identifier

Input Port 1

VC 1

VC v

Routing Computation (RC)

VC Arbiter (VA)

Switch Arbiter (SA)

Input Channel P

Input Port P

Crossbar (P x P)

Output Channel 1

Output Channel P

Flattened Butterfly Network-on-Chip
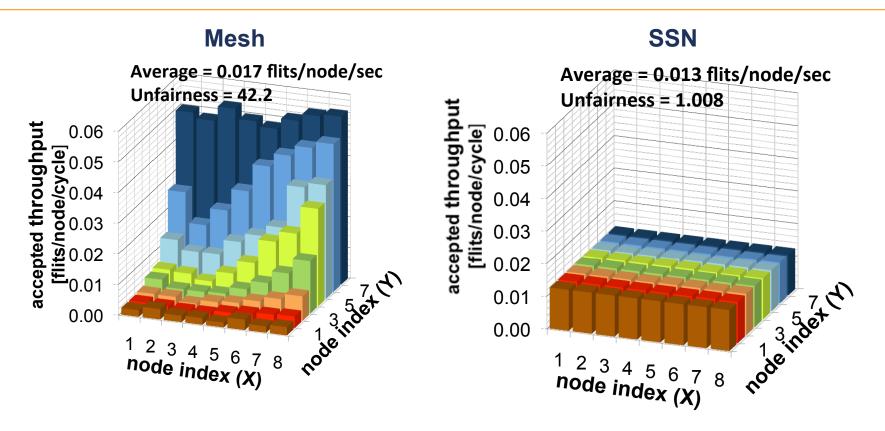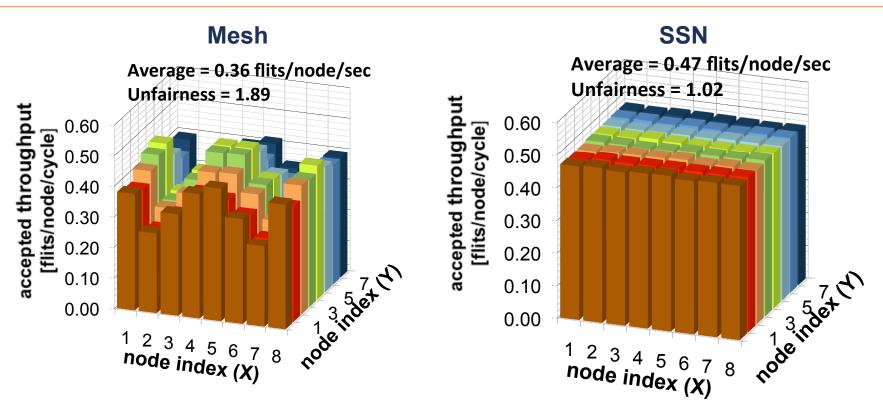
Motivating Swizzle Switch Networks

- ## Uniform access latency
  - Ease of programming, data placement, thread placement,...

- ## Low Power

- ## Simplicity
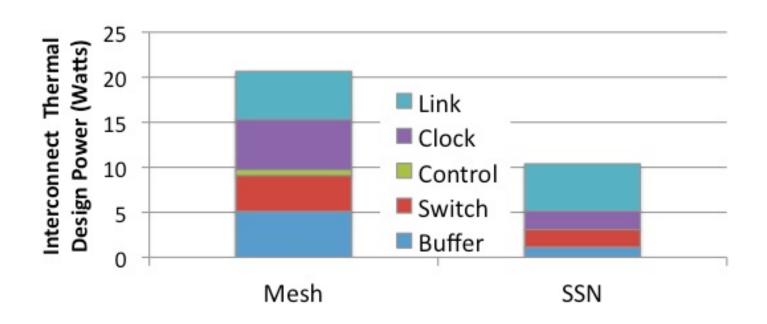 - Packet-switched NoCs need routing, congestion management, flow control, wormhole switching,...

Motivating Swizzle Switch Networks

Mesh

Average = 0.017 flits/node/sec
Unfairness = 42.2

SSN

Average = 0.013 flits/node/sec
Unfairness = 1.008

- Unfairness = $Node_{highest\_throughput}$ / $Node_{lowest\_throughput}$

- Hotspot Traffic = All nodes sending data to $node_{8,8}$

  - Under Hotspot traffic, the Crossbar has a slightly less throughput than the Mesh but is _40x more fair_.

Motivating Swizzle Switch Networks

- In the Mesh, nodes closest to the center receive the highest throughput

- Under Uniform Random traffic, the Crossbar has more throughput than the Mesh and is *87% more fair*.

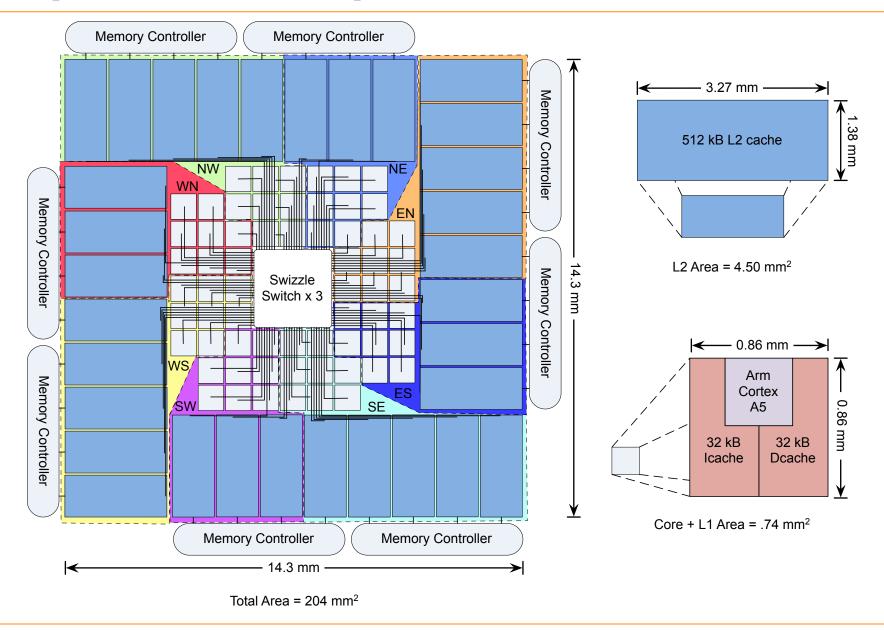# Motivating Swizzle Switch Networks

Outline

- Swizzle Switch—Circuit & Microarchitecture
  - Overview
  - Arbitration
  - Prototype
- Swizzle Switch—Cache Coherent Manycore Interconnect
  - Motivation & Existing Interconnects
  - Swizzle Switch Interconnect
  - Evaluation

Top-Level Floorplan

Memory Controller

Memory Controller

Memory Controller

Memory Controller

Memory Controller

Memory Controller

Memory Controller

Memory Controller

NW

WN

NE

EN

Swizzle
Switch x 3

WS

SW

ES

SE

14.3 mm

14.3 mm

Total Area = 204 mm²

3.27 mm

1.38 mm

512 kB L2 cache

L2 Area = 4.50 mm²

0.86 mm

0.86 mm

Arm
Cortex
A5

32 kB
Icache

32 kB
Dcache

Core + L1 Area = .74 mm²

Outline

- Swizzle Switch—Circuit & Microarchitecture
  - Overview
  - Arbitration
  - Prototype
- Swizzle Switch—Cache Coherent Manycore Interconnect
  - Motivation & Existing Interconnects
  - Swizzle Switch Interconnect
  - Evaluation
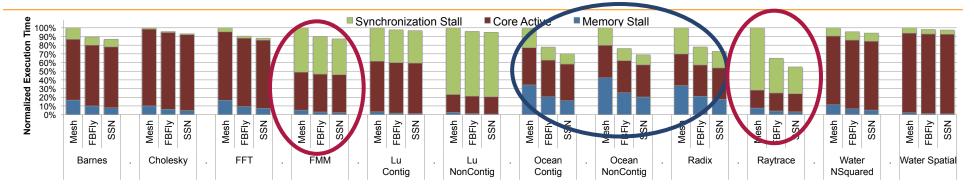
Evaluation

- ## Simulation Parameters

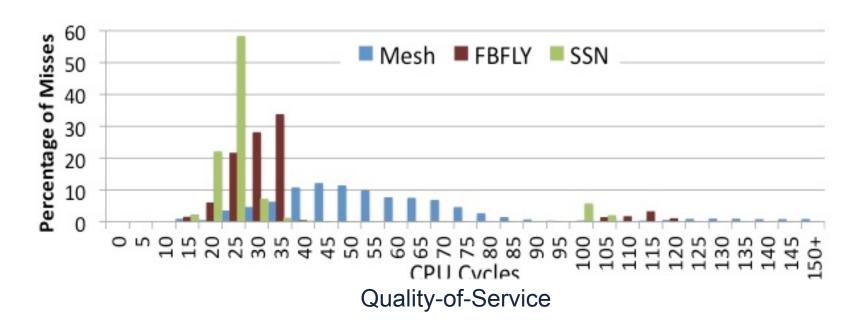| Feature	NoC (Mesh/FBFly)	SSN
Processors	64 in-order cores, 1 IPC, 1.5 GHz	
L1 Cache	32kB I/D Caches, 4-way associative, 64-byte line size, 1 cycle latency	
L2 Cache	Shared L2, 16 MB, 64-way banked, 8-way associative, 64-byte line size, 10 cycle latency	Shared L2, 16MB, 32-way banked, 16-way associative, 64-byte line size, 11 cycle latency
Interconnect	3.0 GHz, 128-bit, 4-stage Routers, 3 virt. networks w/ 3 virt. channels	1.5 GHz, 64x32x128bit Swizzle Switch Network
Main Memory	4096MB, 50 cycle latency	

- ## Benchmarks
  - SPLASH 2 : Scientific parallel application suite

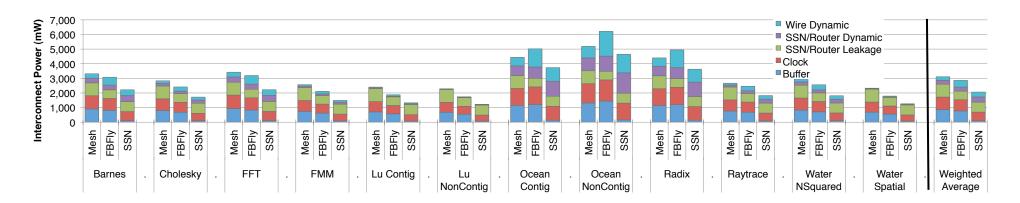# Results—Performance & QoS

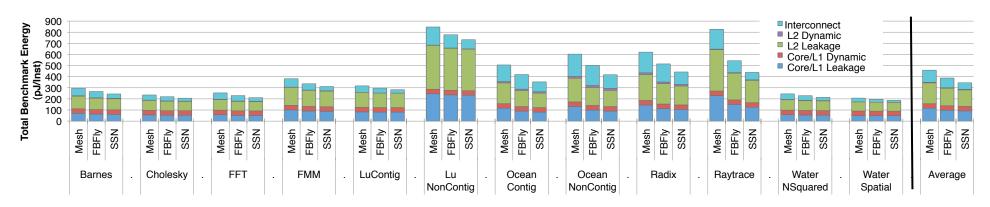
Overall Performance

Quality-of-Service

Results—Power

On average the SSN uses **28%** less power in the interconnect compared to a flattened butterfly

Which results in an average reduction in total system energy to complete the task of **11%**
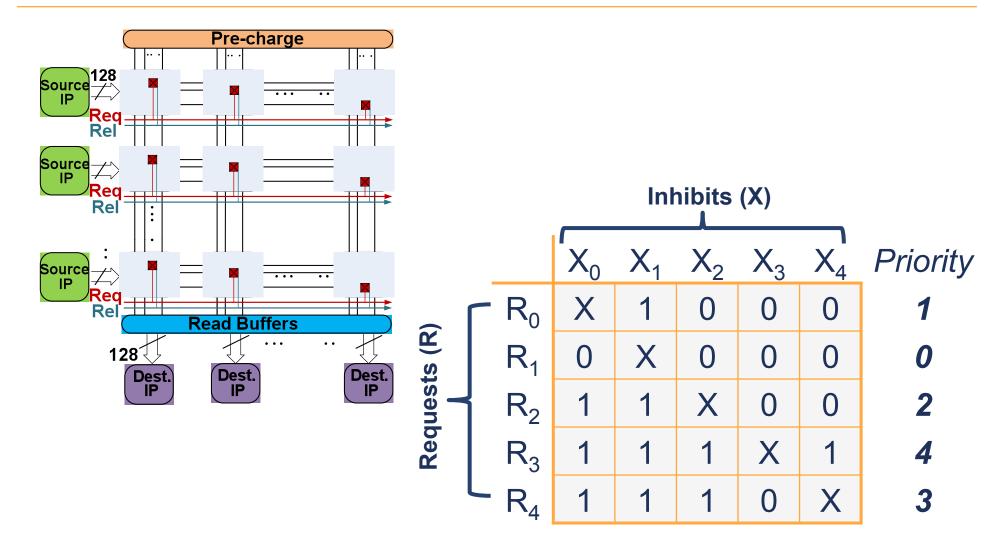
Summary

- ## Swizzle Switch Prototype (45nm)

  - 64x64 Crossbar with 128-bit busses

  - Embedded LRG priority arbitration

  - Achieved 4.4 Tbps @ ~600MHz consuming only 1.3W of power

- ## Swizzle Switch Network Evaluation

  - Improved performance by 21%

  - Reduced power by 28%

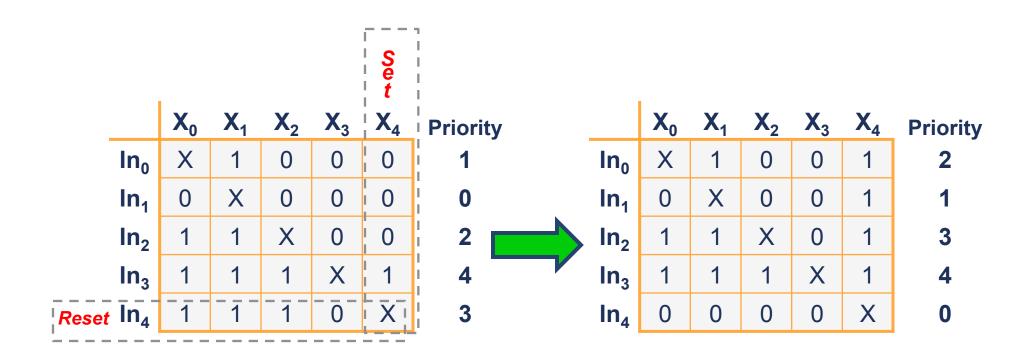 - Reduced latency variability by 3x

Additional Detailed Slides
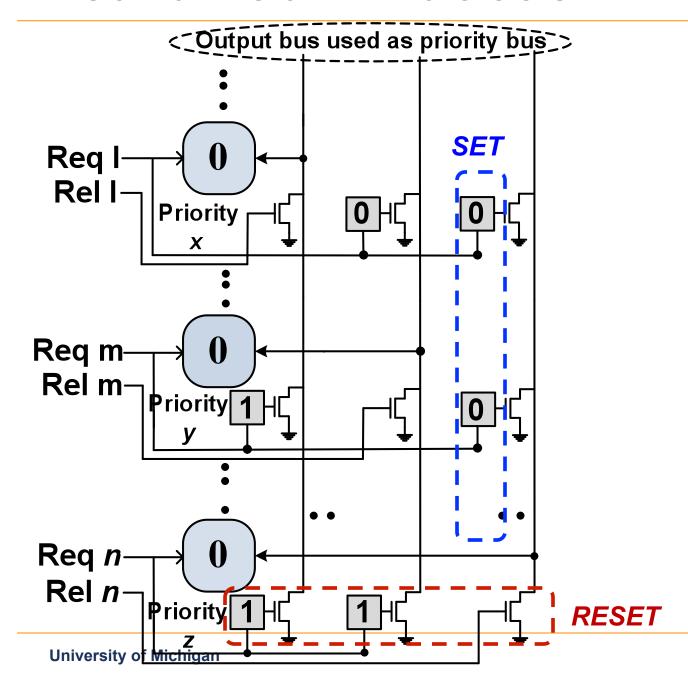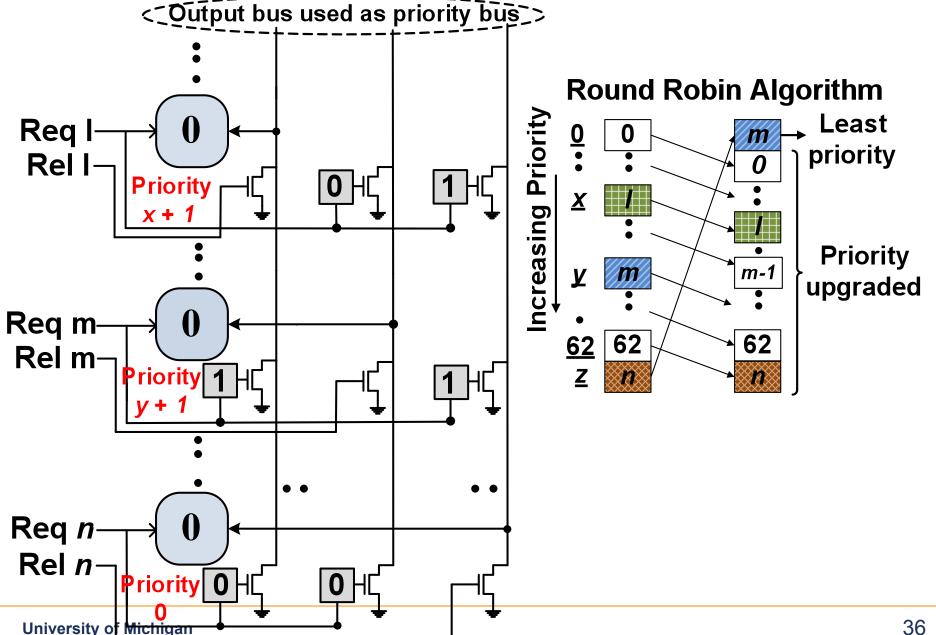
Arbitration Mechanism (Matrix View)

	Inhibits (X)					Priority
	X_0	X_1	X_2	X_3	X_4	
R_0	X	1	0	0	0	1
R_1	0	X	0	0	0	0
R_2	1	1	X	0	0	2
R_3	1	1	1	X	1	4
R_4	1	1	1	0	X	3

Requests (R)

Least Recently Granted (LRG)

	X_0	X_1	X_2	X_3	X_4	Priority
In_0	X	1	0	0	0	1
In_1	0	X	0	0	0	0
In_2	1	1	X	0	0	2
In_3	1	1	1	X	1	4
In_4	1	1	1	0	X	3

Set (column X_4)
Reset (row In_4)

⮕

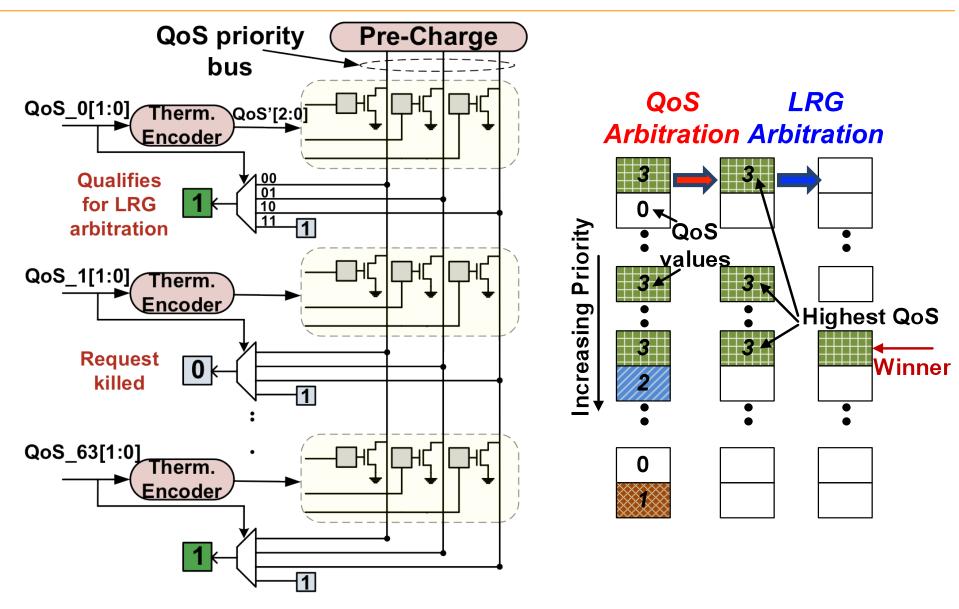|	X_0	X_1	X_2	X_3	X_4	Priority
In_0	X	1	0	0	1	2
In_1	0	X	0	0	1	1
In_2	1	1	X	0	1	3
In_3	1	1	1	X	1	4
In_4	0	0	0	0	X	0

Round Robin Arbitration

Round Robin Arbitration
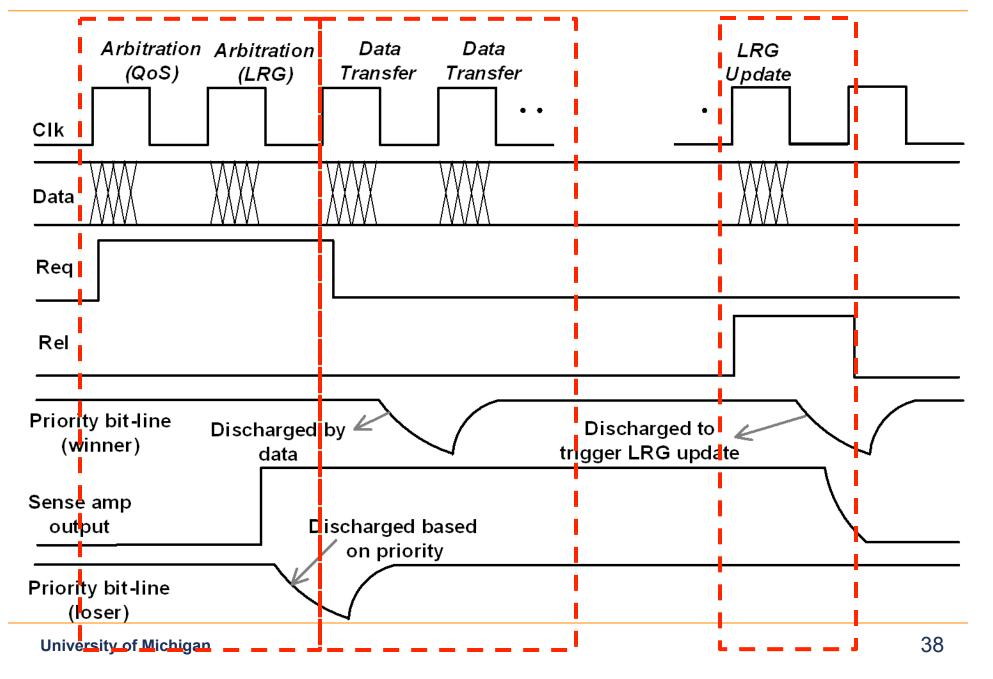
QoS Arbitration
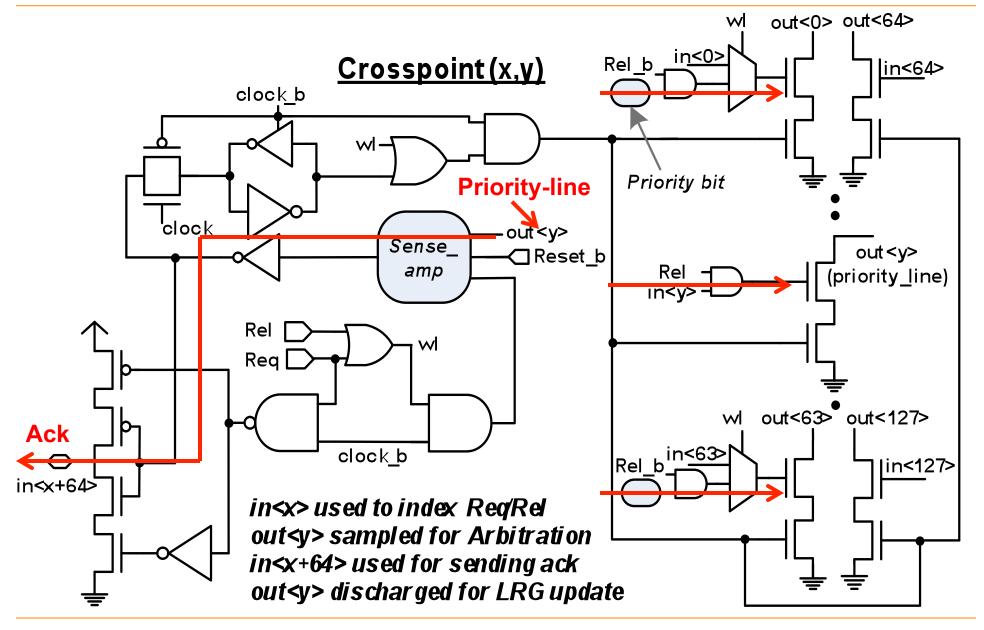
Timing Diagram

Crosspoint Circuit

Crosspoint(x,y)

Priority-line

Priority bit

out<y>
(priority_line)

Ack

in<x+64>

in<x> used to index Req/Rel
out<y> sampled for Arbitration
in<x+64> used for sending ack
out<y> discharged for LRG update
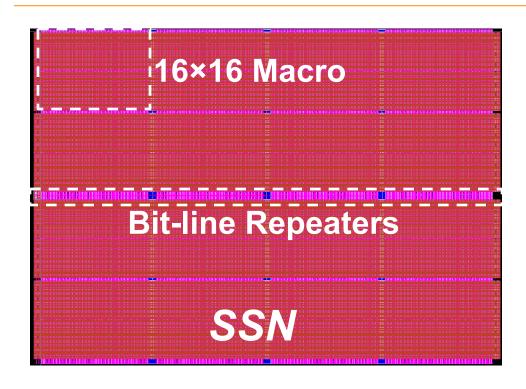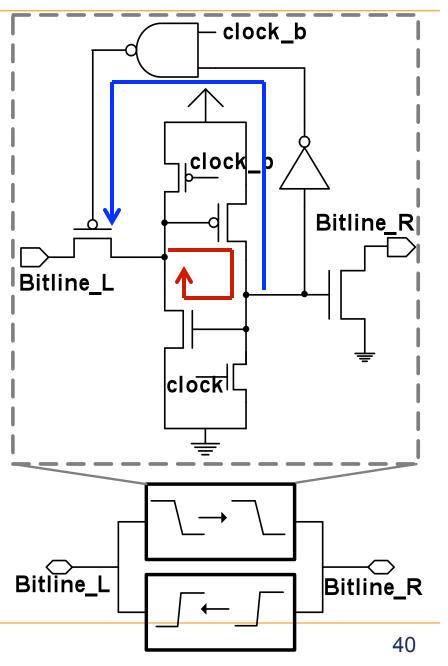
Regenerative Bit-line Repeater

16×16 Macro

Bit-line Repeaters

SSN
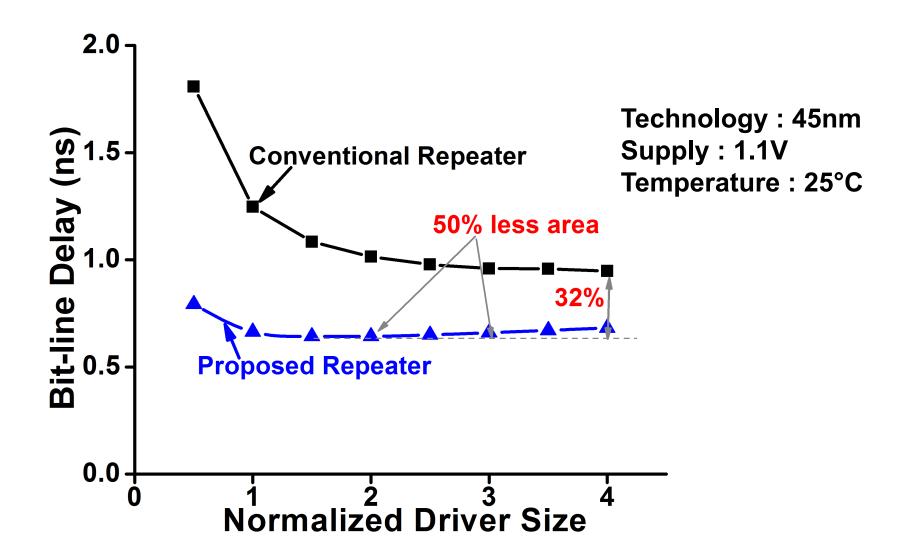
clock_b

clock_b

Bitline_R

Bitline_L

clock

Bitline_L

Bitline_R

Regeneration and **Decoupling** improves speed

Simulated bit-line delay improvement

Technology : 45nm
Supply : 1.1V
Temperature : 25°C

Conventional Repeater
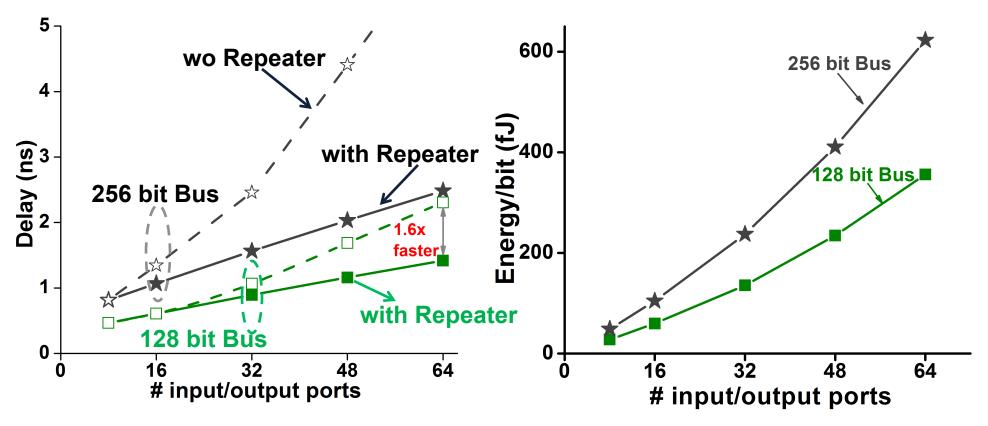
50% less area

32%

Proposed Repeater

SSN Scaling: Simulation

Technology : 45nm
Supply : 1.1V
Temperature : 25°C
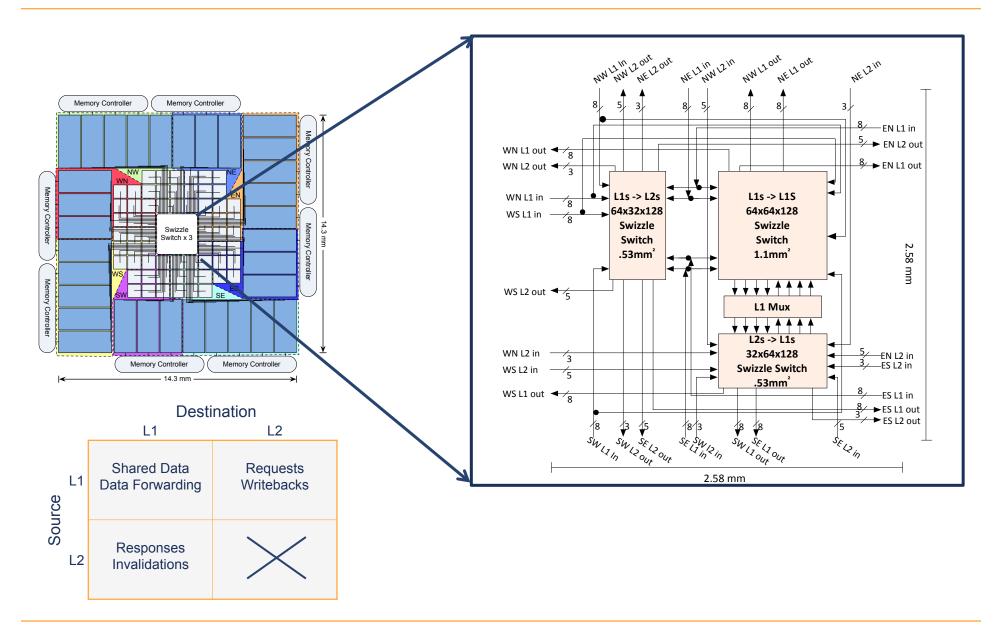
Regenerative repeaters improve SSN scalability

Swizzle Switch Network-on-Chip

	Destination	
	L1	L2
Source L1	Shared Data Data Forwarding	Requests Writebacks
L2	Responses Invalidations	

Results—64-core with A9 O3 cores

Comparison of 64 Core Design (Cortex A9's)