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

- 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”
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

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)

- **LOWEST Priority**: Discharges NO Priority Lines
- **INTERMEDIATE Priority**: Discharges SOME Priority Lines
- **HIGHEST Priority**: Discharges ALL Priority Lines
Least Recently Granted (LRG)

Example Arbitration:

(1) \textit{Req} \textit{l} \quad \text{and} \quad \textit{Req} \textit{m} \quad \text{Request the bus (red lines)}

(2) \textit{Req} \textit{m} \text{ discharges Priority line} \textit{l}, \text{ priority lines} \textit{m} \text{ and} \textit{n} \text{ remain charged (green lines)}

(3) \textit{Req} \textit{l} \text{ senses Priority line} \textit{l} \text{ and is inhibited (not granted),} \textit{Req} \textit{m} \text{ senses Priority line} \textit{m} \text{ and is not inhibited}

(4) The crosspoint records the connectivity at \textit{input m}
Least Recently Granted (LRG)

Output bus used as priority bus

Example Priority Update:

Input m signals it is done with data transfer by asserting \text{Rel m}
Least Recently Granted (LRG)

- **HIGHEST Priority**
  - Discharges ALL Priority Lines

- **INTERMEDIATE Priority**
  - Discharges SOME Priority Lines

- **LOWEST Priority**
  - Discharges NO Priority Lines

**Diagram:**

- Output bus used as priority bus
- Req I, Rel I
  - Priority $x + 1$
- Req m, Rel m
  - Priority 0
- Req n, Rel n
  - Priority 1
Least Recently Granted (LRG)

Output bus used as priority bus

LRG Algorithm

Least priority
Priority upgraded
Priority unchanged

"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

<table>
<thead>
<tr>
<th>Process</th>
<th>45nm SOI CMOS 12metal interconnect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die area</td>
<td>15.6mm²</td>
</tr>
<tr>
<td>Fabric area, Transistor count, # Data wires</td>
<td>4.06mm², 6.95M, 8192</td>
</tr>
<tr>
<td>Throughput, Frequency</td>
<td>4.47Tb/s @ 1.1V, 559MHz, 25°C</td>
</tr>
<tr>
<td>Energy Efficiency at peak throughput</td>
<td>3.4Tb/s/W</td>
</tr>
<tr>
<td>Peak energy efficiency</td>
<td>7.4Tb/s/W @ 0.6V</td>
</tr>
</tbody>
</table>
Measurement Results

Fabric size: 64x64 (128 bit channel)

Technology: 45nm
Temperature: 25°C

559MHz
4.47Tb/s @ 1.1V

28MHz
0.22Tb/s @ 550mV
Measurement Results

![Graph showing measurement results](image)

- Peak efficiency: $7.4\text{Tb/s/W} @ 0.6\text{V}$
- Efficiency at max. throughput: $3.4\text{Tb/s/W}$
- Measured power: $110\text{mW}$
- Efficiency: $1314\text{mW}$
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

Tile

Area = 3.0 mm²

Router

Area = 190 mm²
Flattened Butterfly Network-on-Chip

Area = 190 mm²
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

Unfairness = Node\text{highest_throughput} / Node\text{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 \textit{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

![Bar graph showing interconnect thermal design power comparison between Mesh and SSN.]
Outline

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

Total Area = 204 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

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

### Benchmarks

- SPLASH 2 : Scientific parallel application suite
Results—Performance & QoS

Overall Performance

Quality-of-Service

Table 2 shows the speedup for each benchmark and Figure 12 shows execution time breakdowns comprising three categories: core active cycles, memory stall cycles, and synchronization stall cycles. We observe three different performance impact scenarios in the results. The first arises for benchmarks with high L2 miss rates and substantial sensitivity to L2 access stalls. OceanContig, OceanNonContig, and Radix all have high L2 Misses Per KiloInstruction L2 MPKI as shown in Table 2 and also spend a substantial fraction of execution time on memory stalls as shown in Figure 12. The Swizzle-Switch based topologies substantially accelerate these workloads due to the improved average L2 access latency.

The second class of workloads, including Raytrace and FMM, spend a large fraction of time in synchronization stalls. These particular benchmarks have locks that are sensitive to miss latency. As average miss latency improves, synchronization time is also reduced, yielding significant speedups. When synchronization stalls arise due to load imbalance, as in LuNonContig, there is no significant speedup since improving memory latency does not resolve the load imbalance.

The last scenario arises for benchmarks with a low L2 MPKI. Such benchmarks are insensitive to L2 latency as their working sets fit in L1 and thus achieve only minimal performance gains from the faster interconnects.
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)

![Arbitration Mechanism Diagram]

Table:

<table>
<thead>
<tr>
<th>Requests (R)</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>R₀</td>
<td>X 1 0 0 0 0</td>
</tr>
<tr>
<td>R₁</td>
<td>0 X 0 0 0 0</td>
</tr>
<tr>
<td>R₂</td>
<td>1 1 X 0 0 0</td>
</tr>
<tr>
<td>R₃</td>
<td>1 1 1 X 1 1</td>
</tr>
<tr>
<td>R₄</td>
<td>1 1 1 0 X</td>
</tr>
</tbody>
</table>

Inhibits (X)
Least Recently Granted (LRG)

<table>
<thead>
<tr>
<th>In0</th>
<th>X</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>Priority</th>
<th>In0</th>
<th>X</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>In1</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>In1</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>In2</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>In2</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>In3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>1</td>
<td>4</td>
<td>In3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>In4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>3</td>
<td>In4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

Set

Reset
Round Robin Arbitration

Output bus used as priority bus

Req l
Rel l
Priority x

SET

Req m
Rel m
Priority y

RESET

Req n
Rel n
Priority z
Round Robin Arbitration

Output bus used as priority bus

Round Robin Algorithm

Increasing Priority

Least priority

Priority upgraded

University of Michigan
QoS Arbitration

QoS priority bus

Pre-Charge

QoS[2:0]

QoS_0[1:0]

Therm. Encoder

QoS_1[1:0]

Therm. Encoder

QoS_63[1:0]

Therm. Encoder

Qualifies for LRG arbitration

Request killed

LRG Arbitration

QoS Arbitration

Increasing Priority

Winner

Highest QoS

QoS values

0

1
Timing Diagram

Arbitration (QoS)  Arbitration (LRG)  Data Transfer  Data Transfer  LRG Update

Clk

Data

Req

Rel

Priority bit-line (winner)

Discharged by data

Discharged based on priority

Priority bit-line (loser)

Discharged to trigger LRG update
Crosspoint Circuit

Crosspoint (x,y)

Priority-line

In<x> used to index ReqRel

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

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
were determined using wire models from the design kit using SPICE analysis including repeaters. That leaves two nearest memory controller in order to minimize interconnect congestion. The design uses eight interleaved memory controllers. Each L1 address range is assigned to the compiler estimates and SPICE simulations. We select total cache size to target a frequency of 52 GHz and occupies based on published characteristics of an ARM Cortex-A, [.] in 96nm. The 96nm A, achieves a frequency used to improve a Flattened Butterfly interconnect. Figure 9: High-level architecture diagram (a) of a 64-core system built with a Swizzle-Switch Network-on-Chip. Common Components

Figure 10: a: Classification of communication messages required for coherence. b: Floor plan and wiring diagram for combining three Swizzle-Switches.

Our target 96nm process provides a 591-layer metallization stack. In this metallization stack there are many metal layers and one of the metal layers are reserved for 128bit crossbar. The wires are labeled by the quadrant to which they connect. Each wire in the diagram can be a 3( or 5( or 8 busses, where each bus is 128 bits. The overall area of the Crossbar is

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>L1</td>
</tr>
<tr>
<td></td>
<td>Shared Data</td>
</tr>
<tr>
<td>L1</td>
<td>Data</td>
</tr>
<tr>
<td></td>
<td>Forwarding</td>
</tr>
<tr>
<td>L2</td>
<td>Requests</td>
</tr>
<tr>
<td></td>
<td>Writebacks</td>
</tr>
<tr>
<td>L2</td>
<td>Responses</td>
</tr>
<tr>
<td></td>
<td>Invalidations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1X</th>
<th>2X</th>
<th>4X</th>
<th>8X</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS</td>
<td>SW</td>
<td>NW</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L1</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>Mux</td>
</tr>
<tr>
<td></td>
<td>L1</td>
</tr>
<tr>
<td></td>
<td>L5</td>
</tr>
<tr>
<td></td>
<td>L2</td>
</tr>
</tbody>
</table>

Figure 10: a: Classification of communication messages required for coherence. b: Floor plan and wiring diagram for combining three Swizzle-Switches.
Results—64-core with A9 O3 cores