ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE

PAIR: Periodically Alternate the Identity of Routers to Ensure Deadlock Freedom in NoC

Zifeng Zhao, Xinghao Zhu, Jiyuan Bai, Gengsheng Chen

State Key Laboratory of ASIC & System, Fudan University, China

Jan.23, 2024

Outline

Introduction

Introduction: Network-on-Chip (NoC)

Strong Demand for NoC

a) General-purpose Computing: Multi/Many-Core CPU

b) Domain-specific Arch: Scalable DNN Accelerator

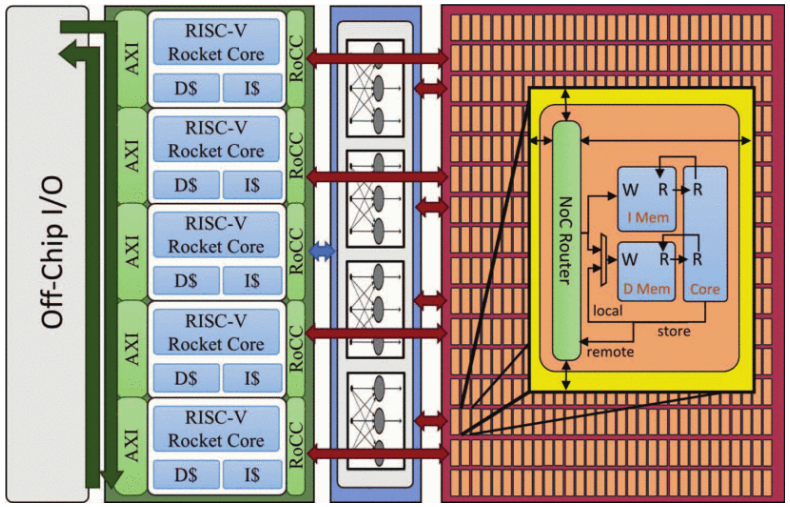


Fig1. Celerity block diagram[1].

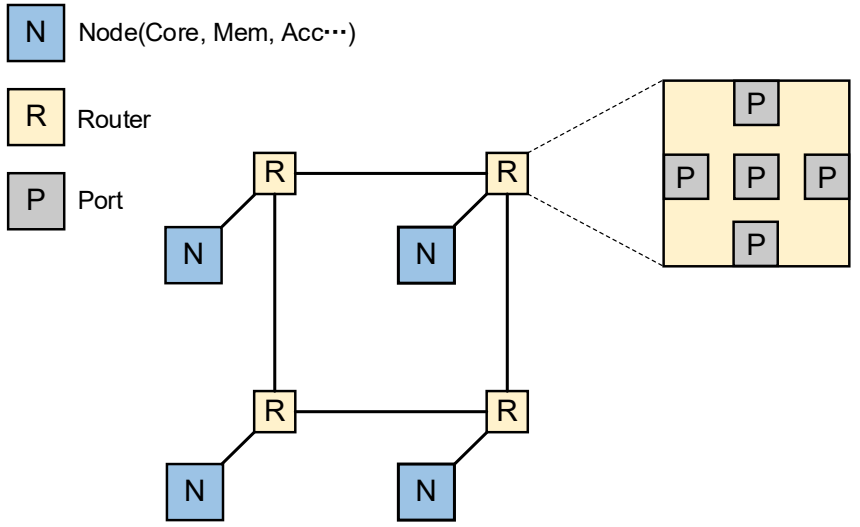


Fig2. Typical NoC Architecture.

[1]S. Davidson et al., "The Celerity Open-Source 511-Core RISC-V Tiered Accelerator Fabric: Fast Architectures and Design Methodologies for Fast Chips," in IEEE Micro, vol. 38, no. 2, pp. 30-41.

Introduction: Deadlock

Deadlock Issues in NoC

a) Deadlock[2]: Circular Resource Dependency Preventing Packet Movement

b) NoC Design Must be Deadlock-free

Deadlock Avoidance

Avoiding deadlock to occur, e.g., XY

✓ Simple to implement

✗ Low performance due to lack of Path Diversity

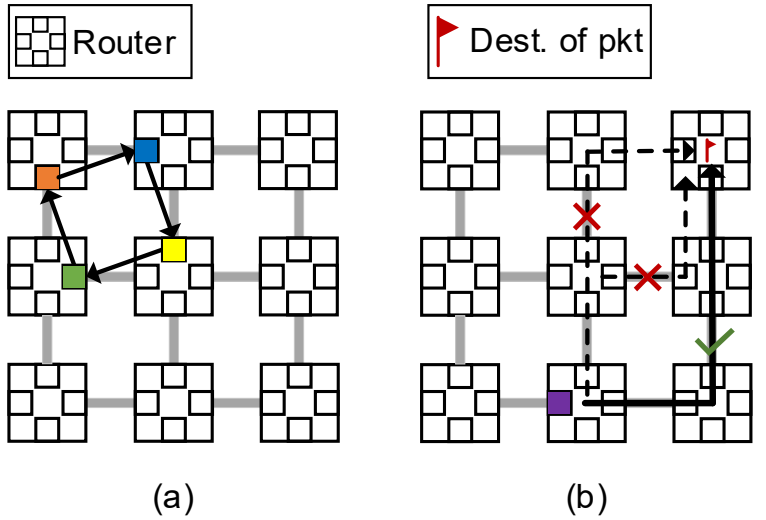


Fig3. (a) Deadlock Example. (b) deterministic XY Routing

[2]Another type of deadlock is protocol deadlock, but in this work, our focus is solely on network deadlock.

Introduction: Recent Solution

Deadlock Recovery

a) Allow deadlock exists

b) Proactively detect and clean, e.g., SPIN, Static Bubble

c) Routing-Agnostic: Any routing algorithm can be used, e.g., Adaptive Routing

✓ High performance due to full Path Diversity

✗ Complex logic and high hardware overhead

Introduction: Recent Solution

Subactive

a) **Allow deadlock exists**

b) **Periodically set up Express Path to break deadlocks by forcing one blocked packet to move, e.g., SWAP, FastPass**

Express Path:

A NoC region where specialized flow control is triggered to forward blocked packets.

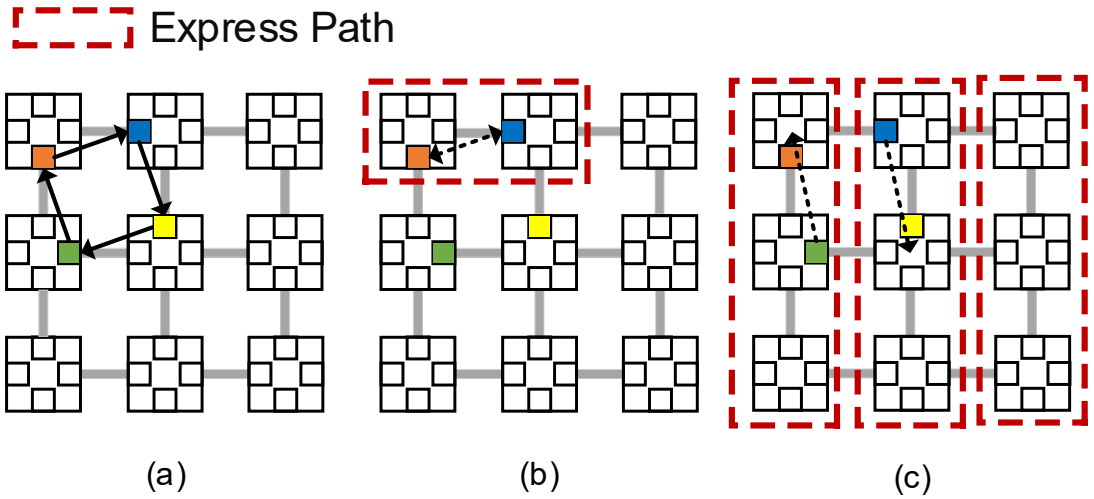


Fig4. (a) Deadlock. Express Path in (b) SWAP (c) FastPass.

Introduction: Path Partition Strategy

Motivation – The issue with Express Path Partition Strategy

Limited Number of Express Paths

a) The Express Path is established using **Inter-Routers** strategy, i.e., R2R path

b) Inefficient utilization of network resource (ports, links)

✓ **No Detection → Low hardware overhead**

✗ **Limited Express Paths → Reduced deadlock resolution efficiency**

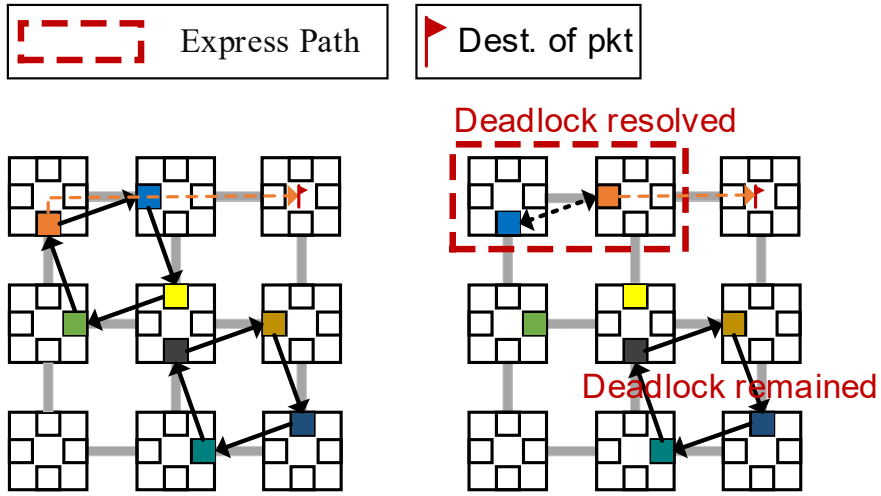


Fig5. (a) Two deadlocks exist in the network.
(b) SWAP exhibits inefficiencies in resolving deadlocks.

PAIR Scheme

PAIR: Overview

Term	Definition
Forward Packet	A packet that makes forward progress
Back Packet	A packet that moves backward
Up Router	Select and propel Forward Packet
Down Router	Select and move the chosen Back Packet backward
PairPeriod	PAIR scheme operates periodically every PairPeriod cycles

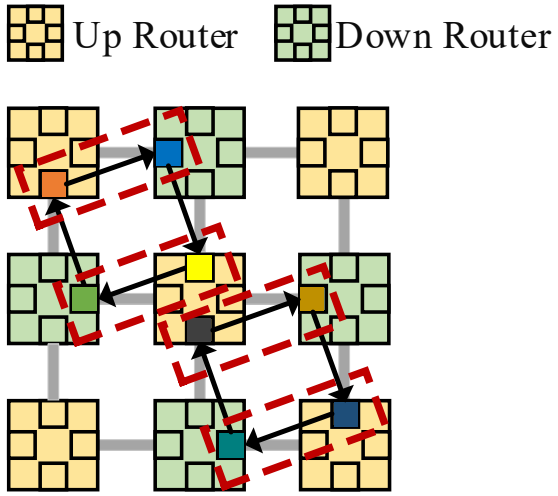


Fig6. “Inter-Ports” Path Partition in PAIR.

Inter-Ports Partition

a) **Set up “Port-to-Port” Express Path across adjacent Up & Down Routers**

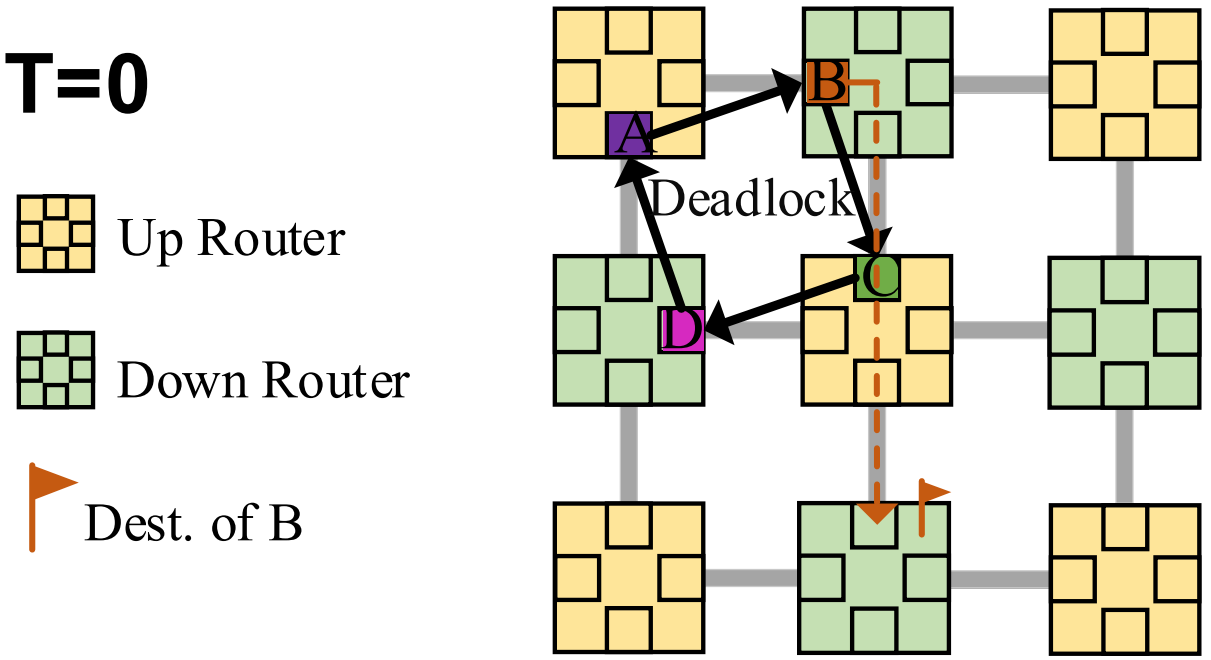
b) **maximize resource utilization**

TABLE I: Different express path partitioning strategies in a $N \times N$ mesh in different solutions

Solutions	Express Path Partitions	Num. Express Paths
XY	NULL	0
BBR	Inter-Routers	1
BINDU	Inter-Routers	1
SWAP	Inter-Routers	1
Pitstop	Inter-Routers	1
SEEC	Inter-Routers	N
FastPass	Inter-Routers	N
PAIR	Inter-Ports	$(N^2/2) \times num\_ports - 1$

PAIR: Walk-through Example

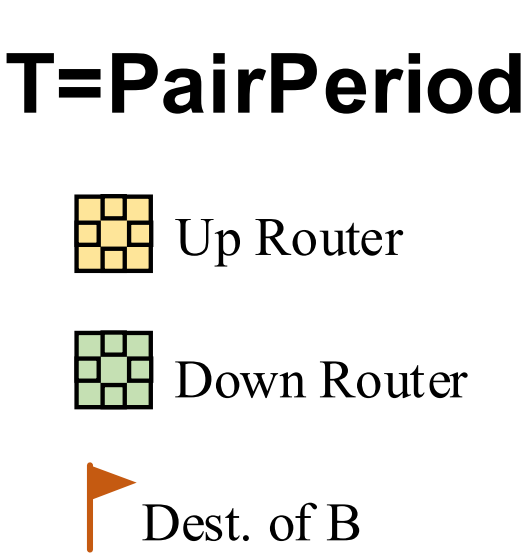
- Deadlock: A→ B→C→D→A
- All Up & Down Routers are staggered in the network

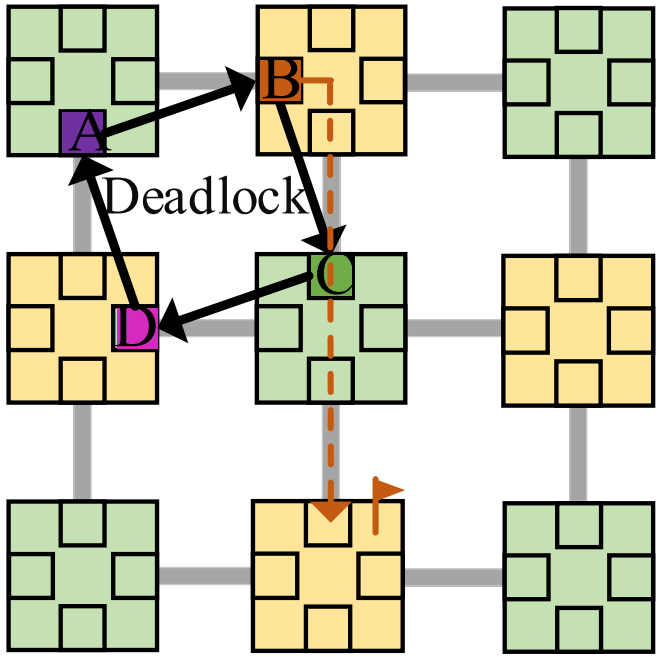


PAIR: Walk-through Example

Step 1:

- Activate PAIR scheme
- Exchange Identity: Up ←→ Down[1]



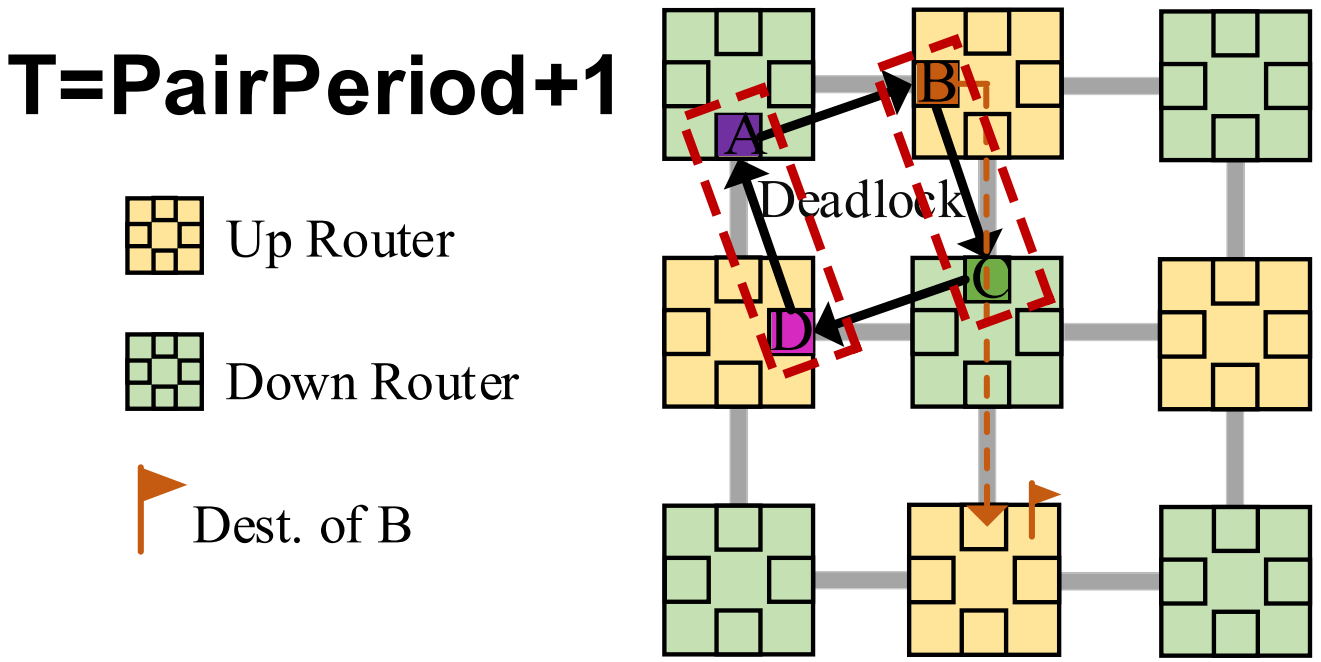


[1]SEnsures every router gets a chance to be upgraded to an Up Router.

PAIR: Walk-through Example

Step 2:

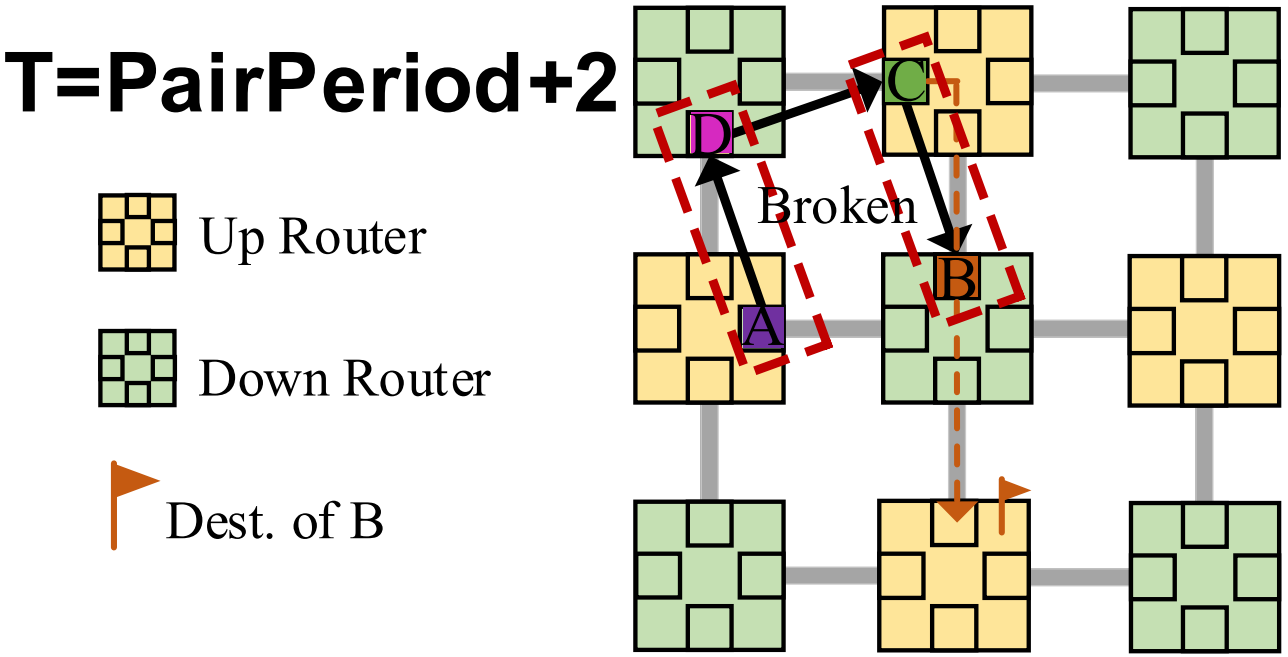
- Build Port-to-Port Express Paths



PAIR: Walk-through Example

Step 3:

- **Swap each pair of Forward & Back Packet (A & D, B & C)**
- **Deadlock Broken**



B & D forwarded, at the cost of misrouted A & C

PAIR: Detailed Implementation

Run for 4 cycles after PAIR is initiated

Cycle 1: Alternate router's Identity(Up/Down); Up Router performs 2-stage arbitration[1] to select the Forward Packet

Cycle 2: PCU generates and sends *pair_req* to adjacent Down Routers

Cycle 3: Down Router setups express path if received *pair_req*

Cycle 4: Swap[2] Forward & Back Packets in the express path

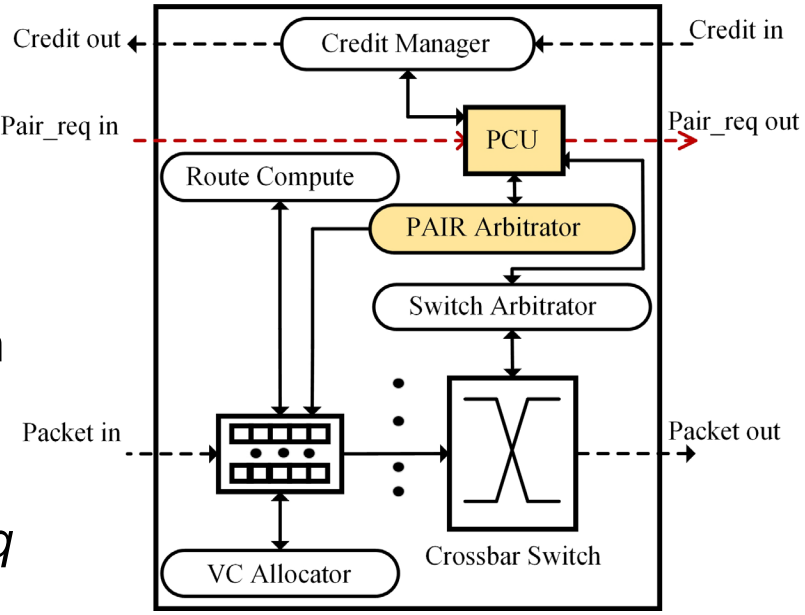


Fig7. PAIR Router Microarchitecture. For clarity, the diagram shows only one input port.

Extend the typical router

a) PAIR Control Unit: initiate PAIR, manage PAIR Arbitrator, send/receive *pair_req* signals

b) PAIR Arbitrator: handle the 2-stage arbitration in Cycle 1

[1]A detailed illustration is attached in the appendix.

[2]For a n-flits packet, it takes n cycles to transfer all flits.

Experiments

Experiments: Configurations

- **We implement and evaluate PAIR using gem5 Garnet3.0 NoC model**
- **Baselines: XY, BBR, SWAP, SEEC, FastPass**
- **Traffic Patterns: Bit Rotation, Shuffle, Transpose**

Network Parameters	
Topologies	8*8, 16*16 mesh
Routing Algorithm	Fully Adaptive Random Minimal Routing (except for XY)
Packet Size	Single Flit
Router	One Cycle
Virtual Channels(VCs) per port	1, 2, 4
Buffer Organization	Virtual Cut Through(VCT); packet per VC

Experiments: Results and Analysis

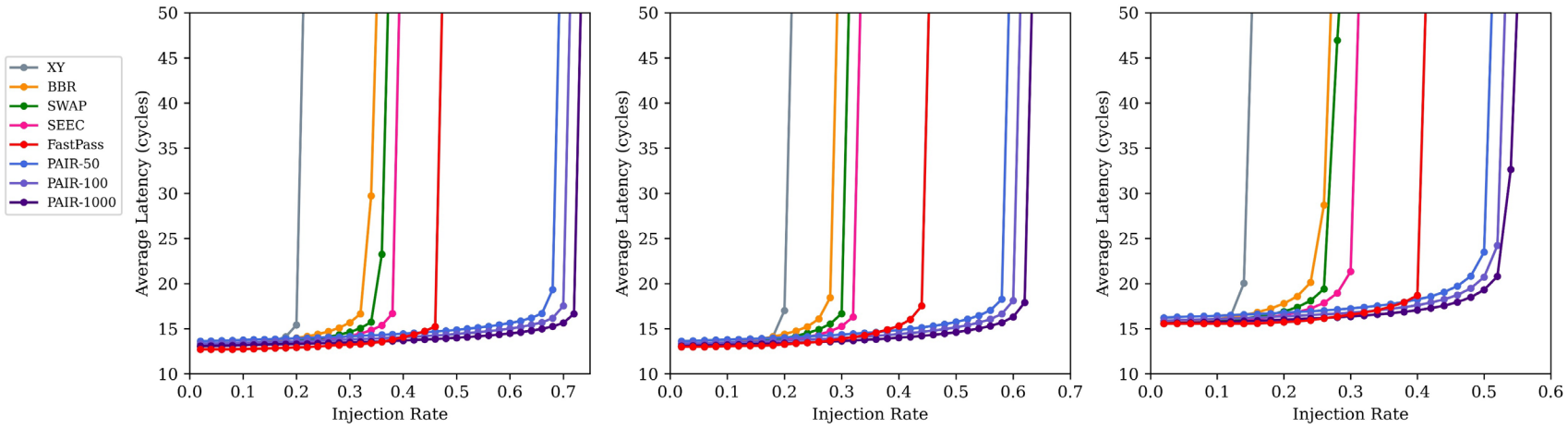


Fig8. Results of 8*8 mesh topology for different traffic patterns

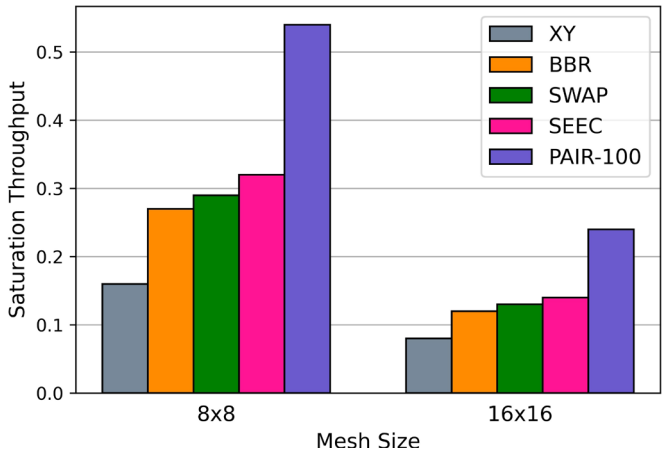


Fig9. Saturation throughput as network size increases for transpose.

Network Throughput Improvement

- In the 8*8 mesh, PAIR shows a 37.78% improvement over FastPass and an 80.36% improvement over all schemes respectively.
- PAIR achieves the highest saturation throughput as network size expands.

Limitation on Average Latency

Compared to other schemes, PAIR leads to higher latency due to frequent misroutes.

Conclusion

Conclusion and Future Work

- A fine-grained “Inter-Ports” partition strategy is proposed to improve buffer & link resource utilization.
- Experimental results shows PAIR outperforms the latest deadlock-free scheme by 37.78%.
- Future work for exploration: More NoC architectures need be explored (NoC Topology, Packet Size, Wormhole Flow Control).

THANK YOU!

Appendix

Two-stage Arbitration:

Stage 1(line 12-15): Multiple blocked packets in an Up Router may request for an input port. Select one as the winner at each input port.

Stage 2(line 18-21): Multiple winners may request the same output port. Select the final winner for each output port.

Algorithm 1: Cycle 1: two-stage arbitration

```
   input : Router Identity PairID
   output: Forward Packets List F2
1  /* Lists initialization */;
2  for op in router.output_ports do
3      F2[op] = NULL;
4      F1_blkd[op] = NULL;
5      F1_nblkd[op] = NULL;
6  if PairID == UP then
7      /* First stage arbitration */;
8      for inport in router.input_ports do
9          if inport.has_packets == True then
10             pkt = R-R(inport.packet);
11             op = pkt.target_output_port;
12             if pkt.blocked == True then
13                 F1_blkd[op].append(pkt);
14             else
15                 F1_nblkd[op].append(pkt);
16     /* Second stage arbitration */;
17     for op in router.output_ports do
18         if F1_blkd[op] ≠ NULL then
19             F2[op] = R-R(F1_blkd[op]);
20         else if F1_nblkd[op] ≠ NULL then
21             F2[op] = R-R(F1_nblkd[op]);
22     return F2;
```