



UNIVERSITY OF TURKU

A Low-Latency and Memory-Efficient On-chip Network

Masoud Daneshtalab, Masoumeh Ebrahimi, Pasi Liljeberg, Juha Prosila,
Hannu Tenhunen

Computer Systems Lab.
University of Turku



Outline

- Introduction
- Backgrounds
- Network Interface Architectures
- Order-Sensitive Memory Scheduler
- Experimental Results



Introduction

- ❑ Network-on-Chip has emerged as a solution to address the communication demands of many/multi core architectures due to its reusability, scalability, and parallelism in communication infrastructure.
- ❑ NoC Structures are composed of routers connecting Processing Elements (PE), to deliver the data (packets) from one place to another , and Network Interfaces (NI).



The fundamental function of network interfaces is to provide communication between PEs and the network infrastructure.



To translate the language between the PE and router based on a standard communication protocol such as AXI and OCP.

- ❑ In-order delivery should be handled when obtaining memory access parallelization by sending requests from a master IP core to multiple slave memories, or when exploiting an out-of-order memory access scheduler in memory controller to reorder memory requests.
- ❑ In this work, we introduce an efficient network interface architecture where the key ideas are threefold:
 - 1) Deal with out-of-order handling.
 - 2) Improve resource utilization due to limitation of hardware resources (e.g. buffers).
 - 3) Present a brilliant memory controller integrated into the network interface, which helps to improve memory utilization and reduce both memory and network latencies.

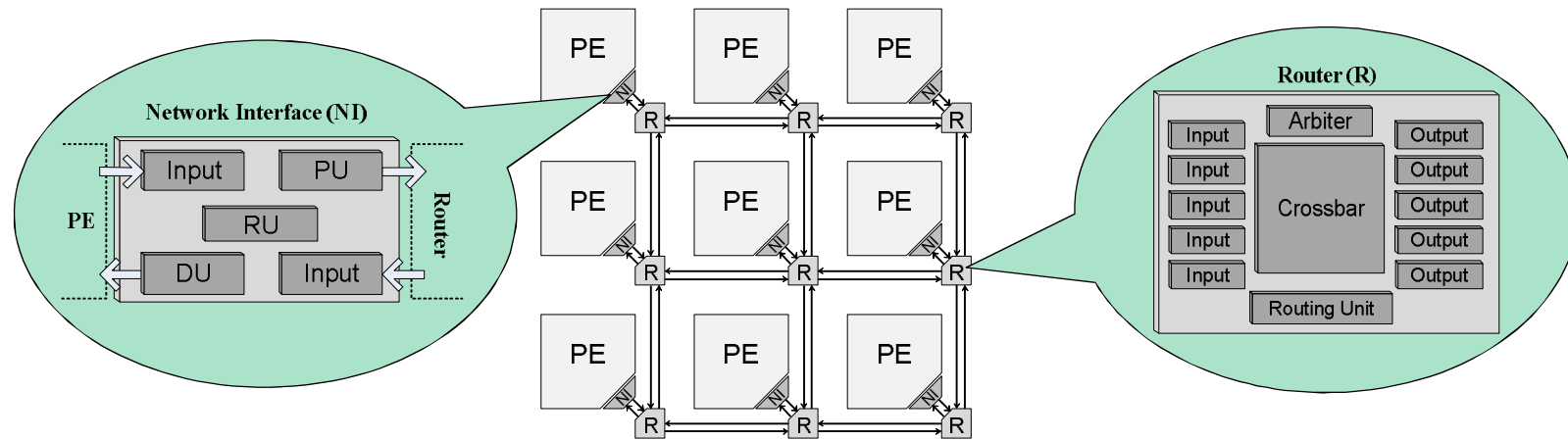


Outline

- Introduction
- Backgrounds
- Network Interface Architectures
- Order-Sensitive Memory Scheduler
- Experimental Results



Background



- ❑ 2D-mesh NoC consists of Routers (R), Processing Elements (PE), and Network Interfaces (NI).
- ❑ PEs may be intellectual property (IP) blocks or embedded memories.
- ❑ To be compatible with existing transaction-based IP-cores, we use the AMBA AXI protocol.



The key features of AMBA AXI:

- Ability to issue multiple outstanding addresses.
- Burst-based transactions with only start address issued.
- Separate read and write data channels.

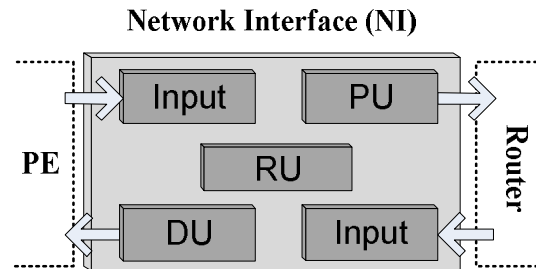


Such advanced functions have motivated us to implement AXI on NoCs as an interface protocol between each PE and router to avoid the structural limitations in SoCs due to the bus architecture.



Background(cont.)

- ❑ In the AXI transaction-based model, IP cores are classified as **master** and **slave**.
- ❑ Master IPs initiate transactions by issuing read and write requests and one or more slaves (memories) receive and execute each request.
- ❑ A response issued by a slave can be either an acknowledgment (corresponding to the write request) or data (corresponding to the read request).
- ❑ Transactions from the same master IP core, but with different IDs have no ordering restriction while transactions with the same ID must be completed in order. Thus, a reordering mechanism in the network interface is needed to afford this ordering requirement.

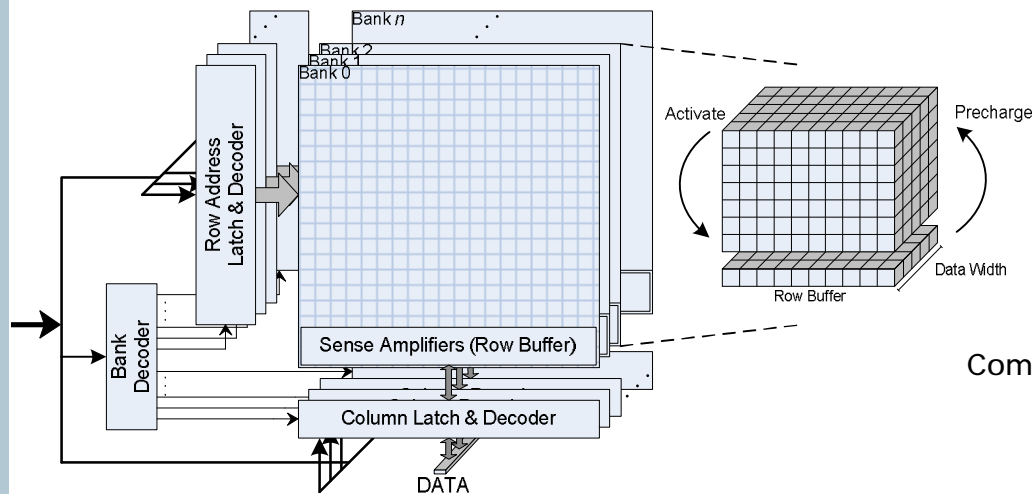


- ❑ The network interface consists of input buffers (forward and reverse directions), a Packetizer Unit (PU), a Depacketizer Unit (DU), and a Reorder Unit (RU).
- ❑ PU is configured to packetize the burst data stored in the input buffer and transfer the packet to the router.
- ❑ DU is configured to restore original data format, required for the PE, from the packet provided by the router.
- ❑ The RU performs a packet reordering to meet the in-order requirement of each PE.



Background(cont.)

- ❑ SDRAM is composed of multiple independent memory banks such that memory requests to different banks can be serviced in parallel.



Commands to different banks can be pipelined.

- ❑ A complete SDRAM access may require three commands:
 - ❑ **Bank precharge:** charges and prepares the bank. (t_{RP})
 - ❑ **Row activation:** is used to copy all data in the selected row into the row buffer, i.e. sense amplifier. The row buffer serves as a cache to reduce the latency of subsequent accesses to that row. (t_{RCD})
 - ❑ **Column access:** Once a row is in the row buffer, then column commands (read/write) can be issued to read/write data from/into the memory addresses (columns) contained in the row. (t_{CL})
- ❑ The **memory controller** lies between processors and the SDRAM to generate the required commands for each request and schedules them on the SDRAM buses.
- ❑ a memory request could be:
 - ❑ **row hit:** a request is accessing the row currently in the row buffer. (t_{CL})
 - ❑ **row conflict:** the access is to a row different from the one currently in the row buffer. ($t_{RP} + t_{RCD} + t_{CL}$)
 - ❑ **row empty:** If the bank is closed (precharged) or there is no row in the row buffer then a row empty occurs. ($t_{RCD} + t_{CL}$)

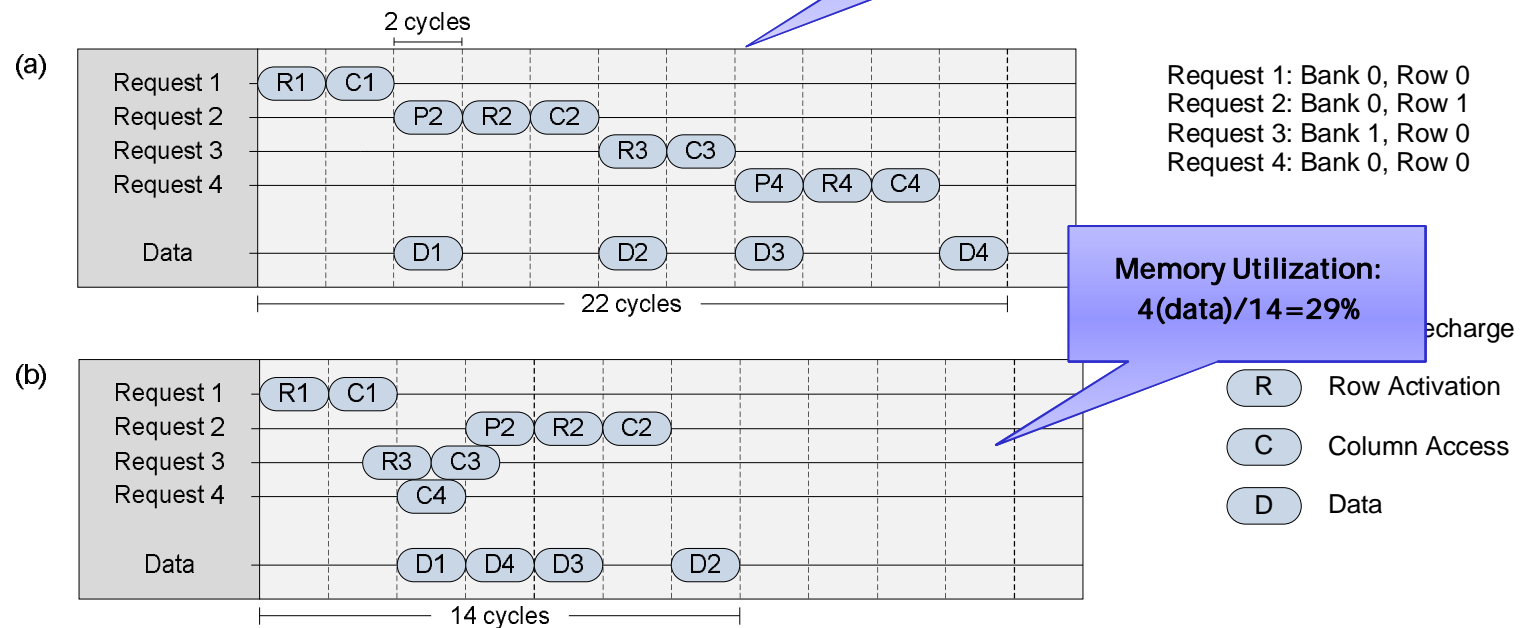


Background (cont.)

- The memory controller consists of:
 - **Request table:** to store the state of each memory request, e.g. valid, address, read/write, header pointer to the data buffer and any additional state necessary for memory scheduling.
 - **Request buffer(s):** The data of outstanding requests are stored in write buffers. (**link list structure**)
 - **Memory access scheduler:** Among all pending memory requests, based on the state of the DRAM banks and the timing constraints of the DRAM, the memory scheduler decides which DRAM command should be issued.

The memory scheduler effects the memory access latency and memory bandwidth.

Memory Utilization:
 $4(\text{data})/22=18\%$





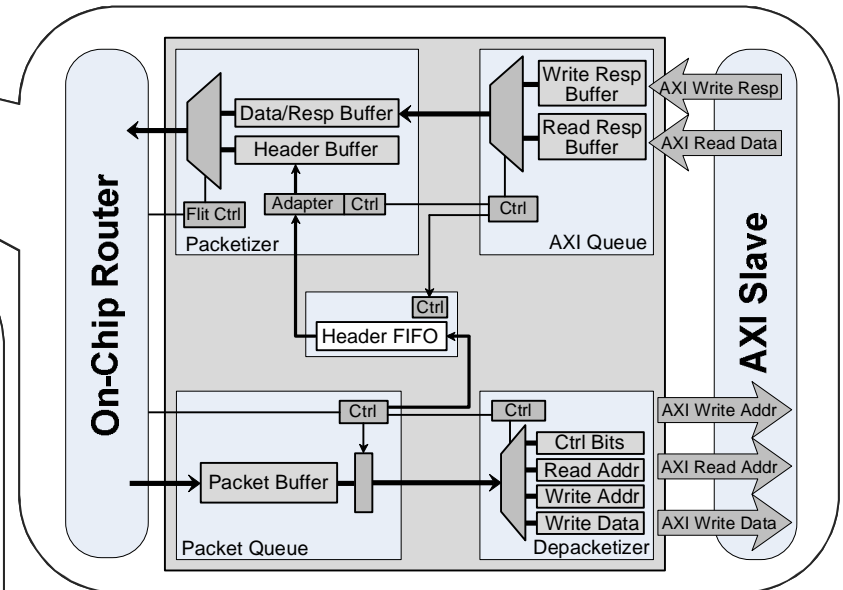
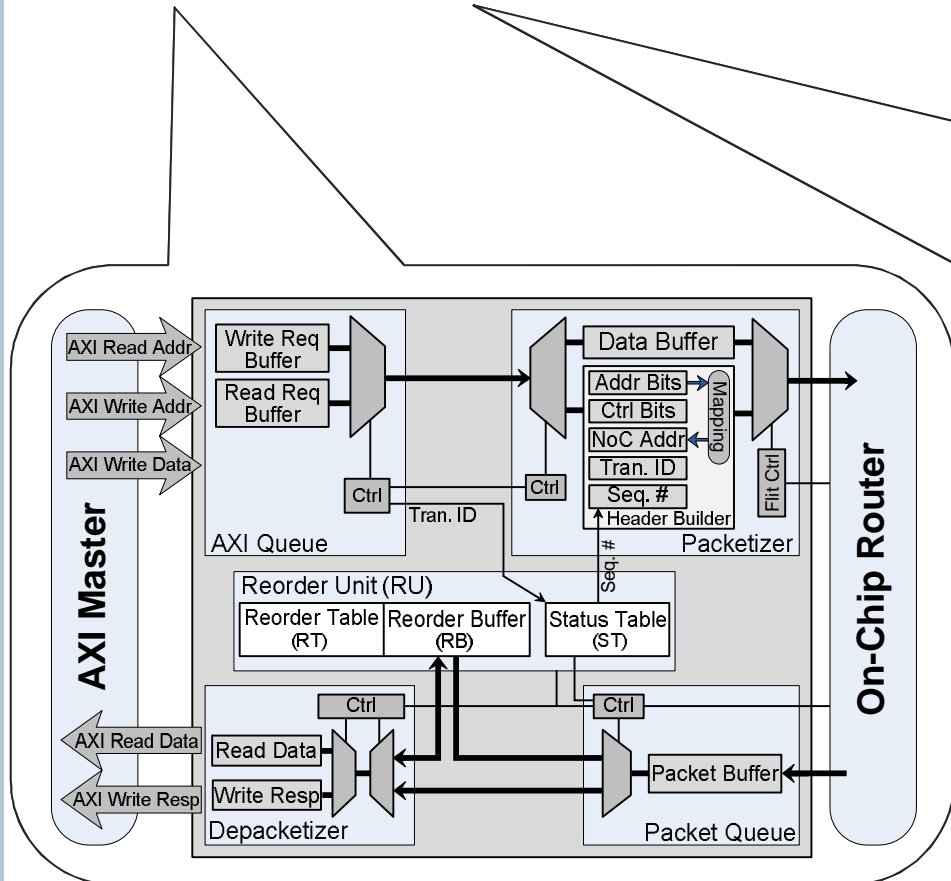
Outline

- Introduction
- Backgrounds
- Network Interface Architectures
- Order-Sensitive Memory Scheduler
- Experimental Results



NI Architectures

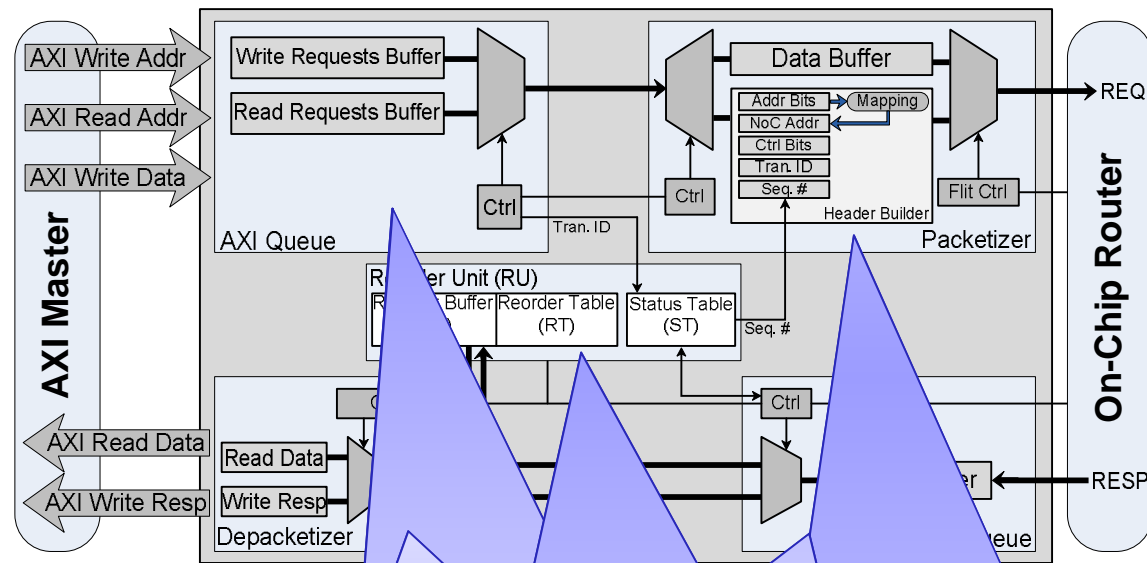
- Since IP cores are classified into masters and slaves, the network interface is also divided into the **master-NI** and **slave NI**.



- Both network interfaces are partitioned into two paths: forward and reverse.
- The forward path transmits the AXI transactions received from an IP core to a router; and the reverse path receives the packets from the router and converts them back to AXI transactions.



Master NI



- ❑ AXI-Queue read transaction write or read
- ❑ The request unit if address
- ❑ A sequence prepared

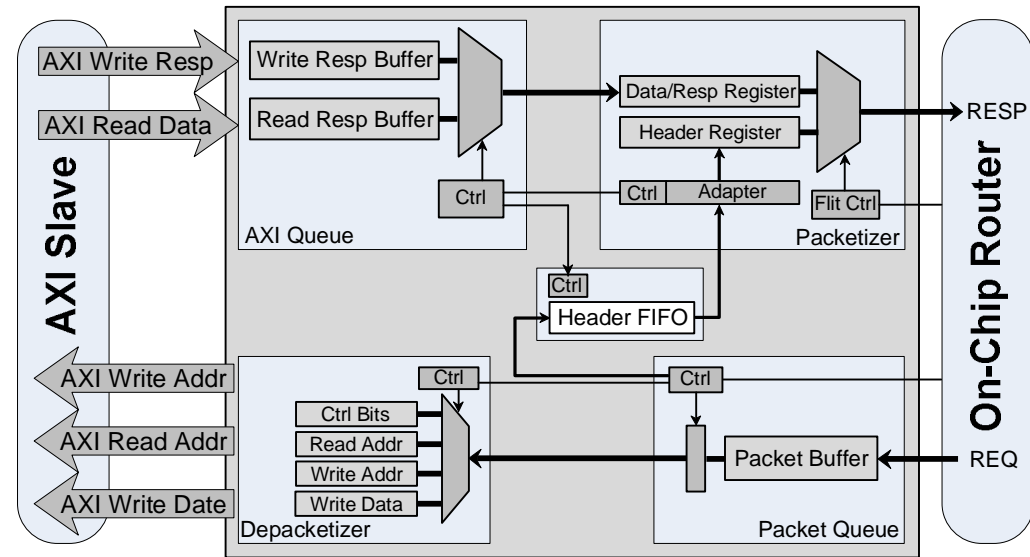
- ❑ This according is determined when transaction unit. Based on packet is out of buffer, and of depacketizer unit

- ❑ the rest unit the

- ❑ The most influential part of the network interface.
- ❑ Including a Status-Register, a Status-Table, a Reorder Buffer, and a Reorder-Table. **(Link-list structure)**
- ❑ In the forward path, prepares the sequence number for corresponding transaction ID, and avoiding overflow of the reorder buffer by the admittance mechanism.
- ❑ in the reverse path, this unit determines where the outstanding packets from the packet queue should be transmitted (reorder buffer or depacketizer), and when the packets in the reorder buffer could be released to the depacketizer unit.



Slave NI



- ❑ A slave IP core cannot operate independently.
- ❑ It receives requests from master cores and responds to them.
- ❑ To avoid losing the order of header information (transaction ID, sequence number, and etc) carried by arriving requests, a FIFO has been considered.
- ❑ After processing a request in a slave core, the response packet should be created by the packetizer.
- ❑ The components of the slave-side NI in both forward and reverse paths are almost similar to the master-side interface components, *except the reorder unit*.

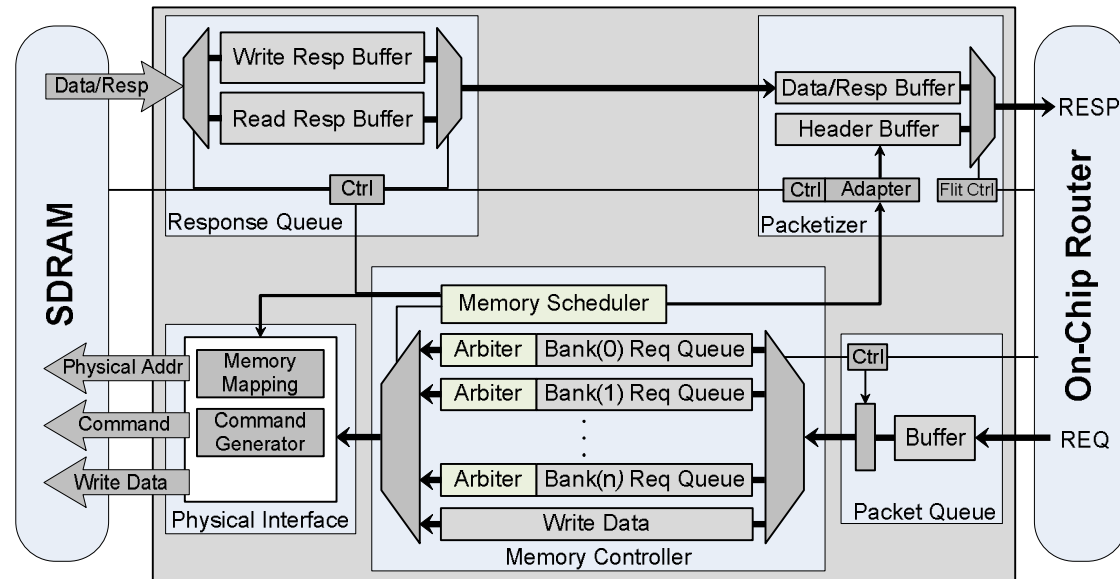


Outline

- Introduction
- Backgrounds
- Network Interface Architectures
- Order-Sensitive Memory Scheduler
- Experimental Results



Order-Sensitive Memory Scheduler



- ❑ The proposed memory controller, **Order-Sensitive (OS)**, is integrated in the slave-side NI.
- ❑ Requests after arriving to the network interface on the edge of the network are stored in the respective queues based on their target banks.
- ❑ The data associated with write requests are stored in the write queue (Linked-list structure).
- ❑ The 2's complement of received request sequence number is assigned as a priority value for this request.
- ❑ To prevent starvation, the priority values of existing requests in the queue at every *input queue* event will be increased.
- ❑ Each bank arbiter selects a request from the queue with the highest priority value based on the **row-first** policy as the first level of scheduling procedure.
- ❑ If there are not any row hits, the bank arbiter selects the highest priority request.
- ❑ In the second level of the scheduling procedure, at each memory cycle the memory scheduler decides which request from all bank arbiters should be issued.
- ❑ To simplify the hardware implementation and provide the *bank interleaving*, round robin mechanism is utilized by the memory scheduler.



Order-Sensitive Memory Scheduler

Request 1: Bank 0, Row 0, Seq # 3
Request 2: Bank 0, Row 1, Seq # 4
Request 3: Bank 1, Row 0, Seq # 3
Request 4: Bank 1, Row 1, Seq # 7
Request 5: Bank 0, Row 2, Seq # 3
Request 6: Bank 1, Row 2, Seq # 3
Request 7: Bank 0, Row 0, Seq # 3
Request 8: Bank 1, Row 0, Seq # 3

RR
Second-level Scheduling

Bank 0								
R1	-3							
R1	-2	R2	-4					Prior Row: 0
		R2	-3	R5	-3			
		R2	-2	R5	-2	R7	-3	Prior Row: 0

Bank 1								
R3	-3							
R3	-2	R4	-7					Prior Row: 1
R3	-1			R6	-3			
R3	0			R6	-2	R8	-3	Prior Row: 1



Outline

- Introduction
- Backgrounds
- Network Interface Architectures
- Order-Sensitive Memory Scheduler
- Experimental Results

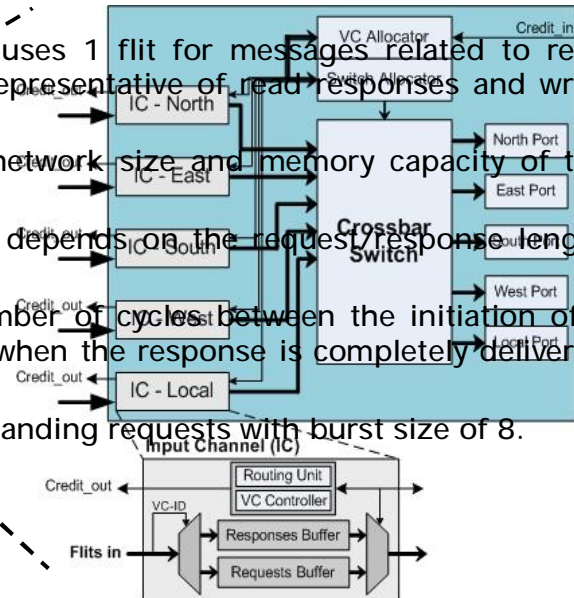
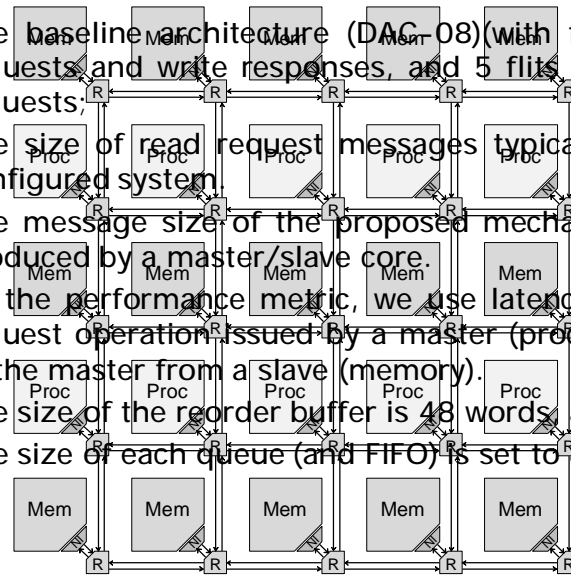


Experimental Results

□ System Configuration

- 5x5 2D-mesh on-chip network configuration for the entire architecture.
- 10 processors (master cores, connected by master NIs)
- 15 memories (slave cores, connected by slave NIs).
- MS (Master/Slave NI) and MS-OS (Slave-NI uses OS) are used to represent the proposed architectures.
- The processors are 32b AXI and the memories are DDR2-512MB ($t_{RP}-t_{RCD}-t_{CL}=2-2-2$, 32b, 4 banks) from Micron Technology.

- The baseline architecture (DAC-O8) (with fixed packet length) uses 1 flit for messages related to read requests and write responses, and 5 flits for data messages, representative of read responses and write requests.
- The size of read request messages typically depends on the network size and memory capacity of the configured system.
- The message size of the proposed mechanism is variable and depends on the request/response length produced by a master/slave core.
- As the performance metric, we use latency defined as the number of cycles between the initiation of a request operation issued by a master (processor) and the time when the response is completely delivered to the master from a slave (memory).
- The size of the reorder buffer is 48 words, able to embed 6 outstanding requests with burst size of 8.
- The size of each queue (and FIFO) is set to 8×32 bits.



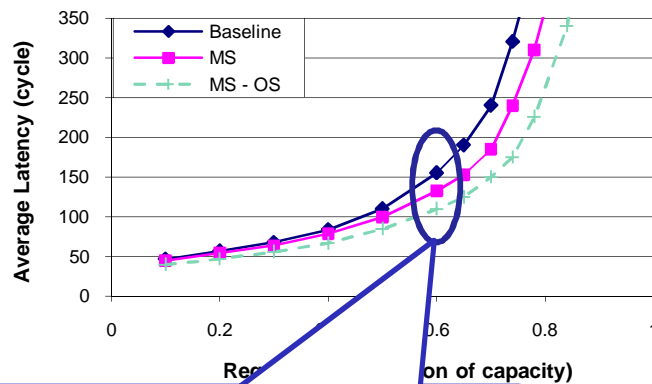
- The on-chip network considered for experiment is formed by a typical state-of-the-art router structure.
- Each input port of the router has 2 VCs. Packets of different message types (request and response) are assigned to corresponding VCs to avoid message dependency deadlock.
- The routers adopt the XY routing and utilize wormhole switching.
- For all routers, the data width (flit) is set to 32 bits, and the buffer depth of each VC is 5 flits.



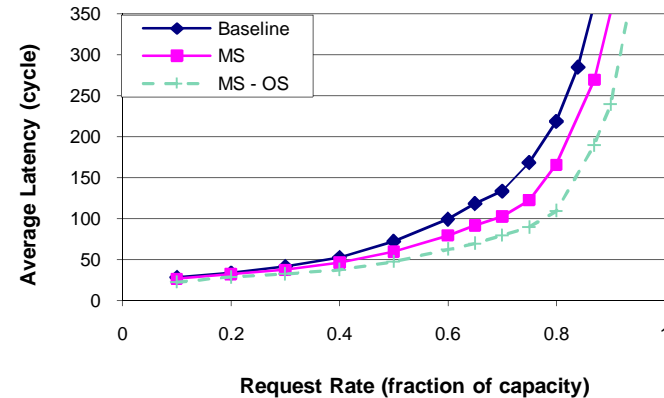
Experimental Results

Performance Evaluation

- ❑ To evaluate the performance of the proposed schemes, the uniform and non-uniform synthetic traffic patterns have been considered separately for the specified configuration.
- ❑ Each processor sends in-order read/write requests to memories with the uniform probability.
- ❑ The memories and request type (read or write) are selected randomly.
- ❑ Eight burst sizes, among 1 to 8, are stochastically chosen regarding the data length of the request.
- ❑ In the non-uniform mode, the traffic consists of 70% local requests, where the destination memory is one hop away from the master core, and the rest 30% traffic is uniformly distributed to the non-local memories.



The average utilization of memories is improved by 22%.
The average network latency is reduced by 19%.



Non-uniform

Hardware implementation of Nis using 90nm UMC

NI	Area		Power (mW)
	Gates (#)	(μm^2)	
Slave-side	18218	42848	0.274
Master-side	32125	75559	0.441
Slave-side with OS	33218	80848	0.486