Towards Unified Mechanisms for Inter-Processor Communication

Manolis Katevenis

Foundation for Research & Technology - Hellas (FORTH)  
Institute of Computer Science (FORTH-ICS),  
and University of Crete  -  Heraklion, Crete, Greece
Communication is at least as important as Computation

- single task: arithmetic, selection and movement of data in memory
- multiple tasks: cooperate via movement of data
- I/O with the rest of the world: movement of data

- Communication architecture has received less attention than what processor architecture received in the past…

M. Katevenis (FORTH-ICS): Towards Unified IPC Mechanisms
Towards Unified IPC Mechanisms: **Outline**

- **Old**: network was far from processor
- **New**: bring the network close to the processor

- Implicit Communication: Coherent Caches & Prefetchers
- Explicit Communication: Local Memories & Remote DMA
  - each one with different advantages
  - each one for different cases

- Configurable SRAM blocks: Cache & Local Memory
- Merge Implicit and Explicit Communication support
- Merge Cache Controller and Network Interface
70’s & 80’s: Bottleneck = Network Links

- Bottleneck: Kilobit to Megabit per second network links
⇒ OK for Network Interface to be far from processor ("I/O"), OK for Networking protocols to be in software
- Over decade-long periods, industry focuses on resolving the currently perceived bottleneck…
90’s: Bottleneck = Networking Protocols

- Gigabit/second Network Links exposed next bottleneck: Networking Protocols and Operating System Calls

⇒ **User-Level** (rather than kernel-mode) access to the (virtualized) network interface
Today: Bottleneck = Latency to access the NI

- Multi-GigaByte/s networking exposes the next bottleneck: while the Network can be as fast as the proc.-mem. “bus”, long-latency access to it forces coarse-grain communication
  ⇒ bring the Network Interface close to the processor, at the level of the cache, as a first-class citizen!
Chip Multiprocessor with Old-Style Network Interfaces

- Processor and/or accelerator Engine
- Higher-Level Cache
- Lower-Level Cache
- buffer space
- parallel
- off-chip
- network
- links
- control registers
- on-chip commun.
- off-chip commun.
- via (directory-based) cache coherence
- (virtualized) (SRAM)
Improvement 1: Network Interf. as First-Class Citizen

Cache Controller

L1 Network Interface

Configurable Cache/Local Memory

parallel

off-chip

network

links

off-chip communication

buffer memories

implicit IPC: cache coherence

explicit IPC: Remote DMA

Implicit Commun. (cache-based)  Explicit Commun. (remote DMA)

Merged Cache Controller + Network Interface
Towards Unified IPC Mechanisms: **Outline**

- Old: network was far from processor
- New: bring the network close to the processor

- **Implicit Communication: Coherent Caches & Prefetchers**
- **Explicit Communication: Local Memories & Remote DMA**
  - each one with different advantages
  - each one for different cases

- Configurable SRAM blocks: Cache & Local Memory
- Merge Implicit and Explicit Communication support
- Merge Cache Controller and Network Interface
Explicit versus Implicit Communication

• Implicit Communication:
  – do not know *in advance* which input data will be needed, or who produced them (hence where they are located) ⇒
  – Cache coherence works best: for sparse and irregular access patterns, will only transfer the dirty cache lines.

• Explicit Communication:
  – Know the *input* data set ahead of time ⇒ can *prefetch*; or
  – know who the *consumers* will be ⇒ *send output* data set.
  – Caches: schedule transfer with programmable prefetcher;
  – Local Memories: schedule transfer with remote DMA;
  – will show: LM & DMA better than Caches & Prefetcher.
  – Recent Advances: programmer only identifies input and output data sets; compiler & runtime schedule the transfers
Remote DMA: In-place Data Delivery

- Global (shared) Address Space
- Allows zero-copy communication, adaptive (multipath) rout’g
- Requires buffer space allocation per producer-consumer pair

M. Katevenis (FORTH-ICS): Towards Unified IPC Mechanisms
Remote Queues: Multi-Party Synchronization

- Remote Queues differ from RDMA as follows:
  - receive buffer space shared among many senders
  - speeds up polling of multiple receive channels
- Atomicity of multiplexing/demultiplexing: Synchronization Primitive
Pull Communication: Prefetch, or Remote Read DMA

Local Memories (LM) & DMA:

3:1 savings in # of packets, hence in energy too, compared to Caches & Prefetchers
Push Communication: Remote Write DMA

Directly send data to consumer (store into remote LM or DMA): 5:1 savings in # of packets, hence in energy too.
Towards Unified IPC Mechanisms: Outline

• Old: network was far from processor
• New: bring the network close to the processor

• Implicit Communication: Coherent Caches & Prefetchers
• Explicit Communication: Local Memories & Remote DMA
  – each one with different advantages
  – each one for different cases

• Configurable SRAM blocks: Cache & Local Memory
• Merge Implicit and Explicit Communication support
• Merge Cache Controller and Network Interface
Undesirable: NI requires Dedicated Memory of its own

- Partitioned memory can provide sufficient throughput, but...
- Promotes data copying
- Underutilizes the total memory space
NI should use the Processor’s Memory

- Space for the NI data structures (at least the large ones) should be dynamically allocated in the processor’s “local” memory
- Sufficient memory throughput provided through bank interleaving
Configurable Local SRAM: Cache + Local Memory

- Run-time Configurable Spaces: adapt to application req’s
- NI Output Com’nd “Registers” (*virtualized*): in local memory
- NI Input Queues (*virtualized*): in local memory
Write-Back’s are like Remote Write DMA

- Global (shared) Address Space
- Cacheable and Non-Cacheable (local memory) subspaces
- Load and store instructions work for any address
Read Misses are like Remote Read DMA’s

- prefetch or read miss
- Proc
  - Load
  - Local SRAM
  - Local mem.
  - Cache Ctrlr
  - NI
  - Remote Read DMA req.
  - fetch req.
  - fetch data
  - RDMA
  - request
  - to/from Network
Hardware-Assisted Software Cache Coherence

- Interesting to consider building (complex) directory-based cache coherence on top of simpler hardware primitives
- Dedicated processors where the OS / runtime system runs
Common Datapath for Cache & LM, CC & NI

- currently under design for FPGA prototyping – SARC project

M. Katevenis (FORTH-ICS): Towards Unified IPC Mechanisms
Acknowledgements

• European Union Funding:
  – SARC
  – HiPEAC
  – UNiSIX

• Georgi Gaydadjiev, Delft
• Alex Ramirez, Barcelona
• Nacho Navarro, Barcelona

Crete:
• Dionisios Pnevmatikatos
• Dimitrios Nikolopoulos
• Angelos Bilas

• Stamatis Kavadias
• Vassilis Papaefstathiou
• George Kalokerinos
• George Nikiforos
Conclusions

• Need High-Speed Interprocessor Communication
• Implicit Communication better served with Caches
• Explicit Com. better served with Local Memories & DMA
• Feasible to make the local SRAM blocks configurable as both cache and local memory
• Similarity of hardware primitives in both cases
  ⇒ Cache Controller – Network Interface Convergence
  ⇒ Merged Implicit & Explicit Communication Support