SCORPIO: 36-Core Shared-Memory Processor Demonstrating Snoopy Coherence on a Mesh Interconnect

Bhavya K. Daya, Chia-Hsin Owen Chen, Suvinay Subramanian, Woo-Cheol Kwon, Sunghyun Park, Tushar Krishna, Jim Holt, Anantha P. Chandrakasan, and Li-Shiuan Peh

Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA

1. Introduction

Existing shared-memory multicore processors utilizing snoopy coherence, explicitly require ordering of requests to maintain memory consistency semantics. Unfortunately ordered interconnects scale poorly; The Achilles heel for buses lies in the limited bandwidth, for rings it is the delay, and for crossbars it is the area. Network-on- Chip (NoC) fabrics such as meshes provide scalable bandwidth but doesn't natively support snoopy coherence protocols.

Although there is a plethora of research on low-power and low-latency mesh routers, none identify a practical ordered mesh network interconnect. Namely, few academic proposals embed ordering support in NoCs [1, 2], but are primarily evaluated through simulations, prompting concerns over correctness and practicality.

We present SCORPIO, a 36-core processor that is arranged in a 2D mesh of 6×6 tiles, as seen in Figure 1. Within each tile is an in-order core, split L1 I/D caches, private L2 cache with destination filtering, and network interface and router equipped with global ordering support. Two Cadence DDR2 memory controllers attach to four unique routers along the chip edge, with the Cadence IP complying with the AMBA AXI interface, interfacing with Cadence PHY to off-chip DIMM modules.

2. Memory Hierarchy and Coherence

The Freescale e200z760 core assumes a bus is connected to the AMBA AHB data and instruction ports, cleanly isolating the core from the details of the network and snoopy coherence support. Between the network and the processor core IP is the self-designed L2 cache with AMBA AHB processor-side and AMBA ACE network-side interfaces.

Since the core IP doesn't support hardware coherency, the L1 and L2 caches are inclusive. With the L1 cache operating in write-through mode, the L2 cache informs the L1 during invalidations and evictions of a line through the core's invalidation port. All L2 caches and the memory controllers receives all requests. A 128 KB cache at each memory controller keeps track of whether the cacheline owner is on-chip or not. The memory controller responds to a request if the cacheline is off-chip.

Although the ordered SCORPIO NoC can plug-and-play with existing ACE controllers, we designed a MOSI protocol along with two performance optimizations. To reduce the memory writeback frequency, the coherence protocol contains an O_D state to permit sharing of dirty data. It differs from the Modified state which indicates exclusive access to a dirty cacheline. Also, to reduce network backpressure, the request queue is not stalled if a snoop request is waiting until updated data arrives. Rather a list of source IDs is maintained such that subsequent snoop request can proceed and when the data arrives, it may be forwarded to all logged snoop request sources.

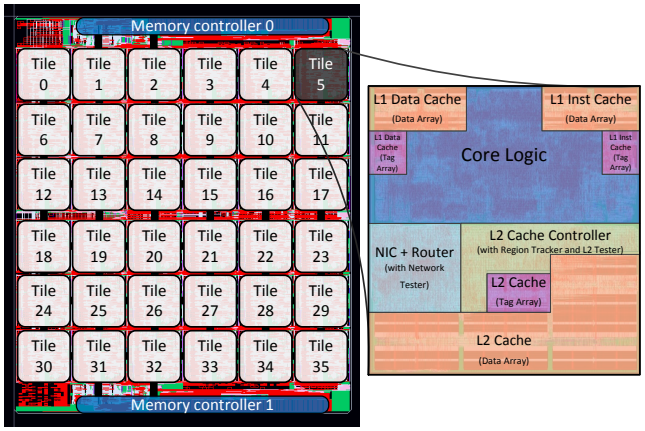


Figure 1: SCORPIO Chip Layout and Tile Floorplan

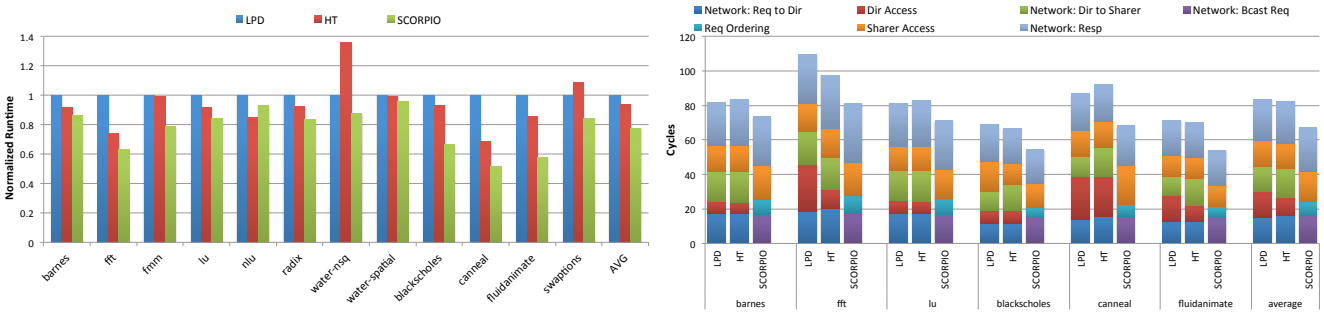
3. On Chip Network

Traditionally, global message ordering on interconnects relies on a centralized ordering point, which imposes greater *indirection*[1] and *serialization latency*[2] as the number of network nodes increases. We eliminate the dependence on the centralized ordering point by decoupling message ordering from message delivery using two physical networks:

Main network. The main network is an unordered network and is responsible for broadcasting coherence requests to all other nodes and delivering the responses to the requesting nodes. Since the network is unordered, the broadcast coherence requests from different source nodes may arrive at the network interface controllers (NIC) of each node in any order. The NICs are responsible for forwarding requests in global order to the cache controller. Ordering maintained within the network achieves sequential consistency, but different message classes or virtual networks are used to avoid protocol deadlocks. The *Globally Ordered Request (GO-REQ)* provides global ordering and hardware broadcast support primarily for coherence and *msync* requests. The *Unordered Response (UO-RESP)* supports data responses and *msync* ACKs. The separation of the responses from the corresponding requests prevents protocol-

[1] Network latency of a message from the source node to ordering point.

[2] Latency of a message waiting at the ordering point before it is ordered and forwarded to other nodes.



(a) Normalized runtime

(b) Served by other caches

Figure 2: Normalized Runtime and Latency Breakdown

level deadlock.

Notification network. For every coherence request sent on the main network, a notification message encoding the source node's ID (SID) is broadcast on the notification network to notify all nodes that a coherence request from this source node is in-flight and needs to be ordered. Essentially, it is a bit vector where each bit corresponds to a request from a source node, so broadcasts can be merged by OR-ing the bit vectors in a contention-less manner. The notification network thus has a fixed maximum network latency bound. Accordingly, we maintain synchronized time windows, greater than the latency bound, at each node in the system. We synchronize and send notification messages only at the beginning of each time window, thus guaranteeing that all nodes received the same set of notification messages at the end of that time window. By processing the received notification messages in accordance with a *consistent* ordering rule, all network interface controllers (NIC) determine *locally* the *global* order for the actual coherence requests in the main network. As a result, even though the coherence requests can arrive at each NIC in any order, they are serviced at all nodes in the same order.

4. Architecture Analysis

For full-system architectural simulations of SCORPIO, we use the GEMS simulator [3] and the GARNET network model. The SCORPIO and baseline architectural parameters are faithfully mimicked within the limits of the simulation environment. SCORPIO is compared with two baseline directory protocols, *Limited-pointer directory* (LPD) and *HyperTransport* (HT).

We evaluate LPD, HT, and SCORPIO with a distributed directory cache to equalize this directory access latency and specifically isolate the effects of indirection and storage overhead. All architectures share the same coherence protocol and run on the same NoC (minus the ordered virtual network and notification network).

4.1. Performance

Figure 2 shows the normalized full-system application runtime for SPLASH-2 and PARSEC benchmarks simulated on GEMS. On average, SCORPIO shows 24.1% better performance over LPD and 12.9% over HT across all benchmarks. Diving in, we realize that SCORPIO experiences average L2 service latency of 78 cycles, which is lower than that of LPD (94 cycles) and HT (91 cycles). When a request is served by other caches, SCORPIO's average latency is 67 cycles, which is 19.4% and 18.3% lower than LPD and HT, respectively. Since we equalize the directory cache size for all configurations, the LPD caches fewer lines compared to SCORPIO and HT, leading to a higher directory access latency which includes off-chip latency. SCORPIO provides the most latency benefit for data transfers from other caches on-chip by avoiding the indirection latency.

As for requests served by the directory, HT performs better than LPD due to the lower directory cache miss rate. Also, because the directory protocols need not forward the requests to other caches and can directly serve received requests, the ordering latency overhead makes the SCORPIO delivery latency slightly higher than the HT protocol. Since the directory only serves 10% of the requests, SCORPIO still shows 17% and 14% improvement in average request delivery latency over LPD and HT, respectively, leading to the overall runtime improvement.

4.2. Power, Area and Frequency

The overall aggregated power consumption of SCORPIO is around 28.8 W. The power consumption of a core with L1 caches is around 62% of the tile power, whereas the L2 cache consumes 18% and the NIC and router 19% of tile power. A notification router costs only a few OR gates; as a result, it consumes less than 1% of the tile power.

The dimension of the fabricated SCORPIO is $11 \times 13\, mm^2$. Each memory controller and each memory interface controller occupies around $5.7\, mm^2$ and $0.5\, mm^2$ respectively. Within a tile, L1 and L2 caches are the major area contributors, taking 46% of the tile area and the network interface controller together with router occupying 10% of the tile area.

SCORPIO was implemented using ARM standard cells, and achieves a post-synthesis frequency of 1 GHz (833 MHz post-layout).

References

[1] N. Agarwal, L.-S. Peh, and N. K. Jha, "In-Network Snoop Ordering (INSO): Snoopy Coherence on Unordered Interconnects," in *HPCA*, 2009.

[2] M. M. Martin, M. D. Hill, and D. A. Wood, "Timestamp snooping: An approach for extending smps," in *ASPLOS*, 2000.

[3] M. M. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (gems) toolset," *Computer Architecture News*, 2005.