Cache Hierarchy and Memory Subsystem of the AMD Opteron Processor

THE 12-CORE AMD OPTERON PROCESSOR, CODE-NAMED ''MAGNY COURS,'' COMBINES ADVANCES IN SILICON, PACKAGING, INTERCONNECT, CACHE COHERENCE PROTOCOL, AND SERVER ARCHITECTURE TO INCREASE THE COMPUTE DENSITY OF HIGH-VOLUME COMMODITY 2P/4P BLADE SERVERS WHILE OPERATING WITHIN THE SAME POWER ENVELOPE AS EARLIER-GENERATION AMD OPTERON PROCESSORS. A KEY ENABLING FEATURE, THE PROBE FILTER, REDUCES BOTH THE BANDWIDTH OVERHEAD OF TRADITIONAL BROADCAST-BASED COHERENCE AND MEMORY LATENCY.

Pat Conway
Nathan Kalyanasundharam
Gregg Donley
Kevin Lepak
Bill Hughes
Advanced Micro Devices

Recent trends point to high and growing demand for increased compute density in large-scale data centers. Many popular server workloads exhibit abundant process- and thread-level parallelism, so benefit directly from additional cores. One approach to exploiting thread-level parallelism is to integrate multiple simple, in-order cores, each with multithreading support. Such an approach has achieved high batch throughput on some commercial workloads, but has limitations when response time (latency), distribution of response times (quality of service, or QoS), and user experience are a concern.[1,2] Examples of highly threaded applications requiring both high throughput and bounded latency include real-time trading, server-hosted game play, interactive simulations, and Web search.

Despite focused research efforts to simplify parallel programming,[3] most server applications continue to be single-threaded and latency sensitive, which favors the use of high-performance cores. In fact, a common use of chip multiprocessor (CMP) servers is simply running multiple independent instances of single-threaded applications in multiprogrammed mode. In addition, for workloads with low parallelism, the highest performance will likely be achieved by a CMP built from more complex cores capable of exploiting instruction-level parallelism from the small number of available software threads. Similarly, many high-performance computing applications that operate on large data sets in main memory and use software pipelining to hide memory latency might run best with moderate numbers of high-performance cores.

The AMD Opteron processor has hardware support for virtualization.[4] It lets multiple guest operating systems run on a single system, fully protected from the effects of other guest operating systems, each running

Published by the IEEE Computer Society

its own set of application workloads. A common usage scenario is to dedicate one core per guest operating system to provide hardware context (thread)-based QoS. Data centers are consolidating legacy single-application servers onto high-core-count dual- and quad-processor (2p and 4P) blade and rack servers—the dominant segment of the server market. By doubling compute density, ''Magny Cours'' doubles the number of guest operating systems that can be run per server in this mode. By using larger servers in place of a pool of smaller servers (say, one 4P blade versus four 1P blades), the operating system or hypervisor can flexibly allocate memory and I/O resources across applications and guest operating systems as needed and on demand.

Efficient power management is another first-order consideration in data centers since power budget determines both the data center's maximum scale and its operating cost. Benchmarks such as SPECPower2008 measure power consumption and performance at different hardware utilizations. Such benchmarks reward designs that provide more performance within the same power envelope and conserve power when idle. The consolidation of multiple single-application servers with low hardware utilization levels onto high-core-count blade servers using virtualization results in significant power savings in the data center.

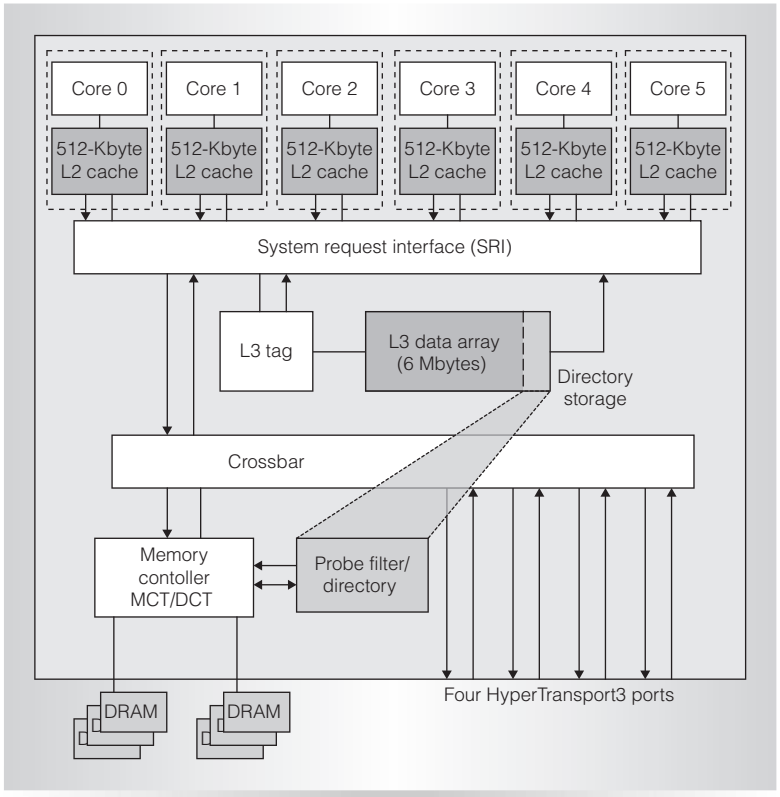


Figure 1. ''Magny Cours'' silicon block diagram. The node integrates six x86-64 cores, a shared L3, two DDR3 memory channels and four HT3 ports.

Processor overview

The basic building block in ''Magny Cours'' is a silicon design, a node that integrates six x86-64 cores, a shared 6-Mbyte level-3 (L3) cache, four HyperTransport3 ports, and two double data rate 3 (DDR3) memory channels (see Figure 1). We built the node using 45-nanometer silicon on insulator (SOI) process technology.

''Magny Cours'' die

Each ''Magny Cours'' processor core is an aggressive out-of-order, three-way superscalar processor. It can fetch and decode up to three x86-64 instructions each cycle from the instruction cache. It turns variable-length x86-64 instructions into fixed-length macro-operations (mops) and dispatches them to two independent schedulers—one for integer and one for floating-point and multimedia operations. These schedulers can dispatch up to nine mops to the following execution resources:

- three integer pipelines, each containing an integer-execution unit and an address-generation unit; and
- three floating-point and multimedia pipelines.

The schedulers also dispatch load-and-store operations to the load/store unit, which can perform two loads or stores each cycle. The processor core can reorder as many as 72 mops. The core has separate instruction and data caches, each 64 Kbytes in size and backed by a large on-chip L2 cache, which is 512 Kbytes. All caches throughout the hierarchy (including the L3 cache) have 64-byte lines. The L1 caches are two-way associative and have a load-to-use latency of three clock cycles; the L2 cache is 16-way

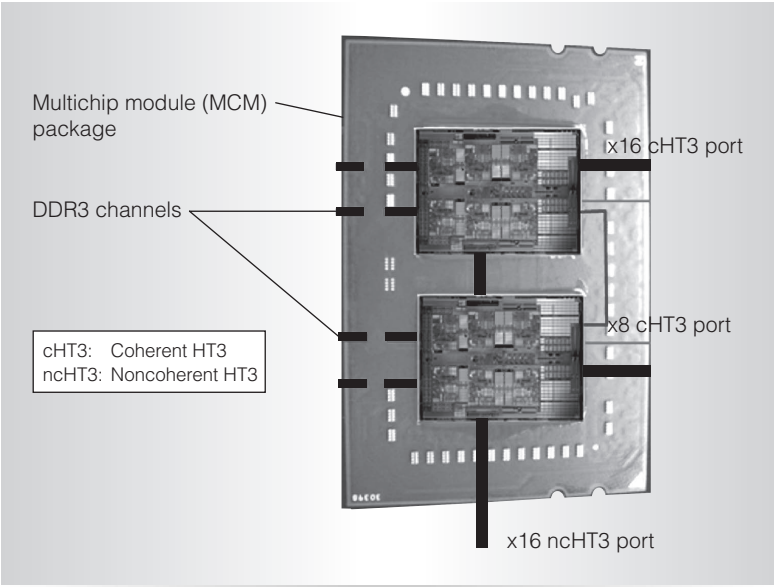


Figure 2. Logical view of the G34 multichip module package. The MCM package has 12 cores, four HyperTransport ports, a 12-Mbyte L3 cache, and four memory channels. Each die has six cores, four HyperTransport ports, a 6-Mbyte L3 cache, and two memory channels.

associative with a best-case load-to-use latency of 12 clock cycles. As in many prior-generation AMD processor cores, the L1 and L2 caches use an exclusive layout—that is, the L2 cache is a victim cache for the L1 instruction and data caches. Fills from outer cache or DRAM layers (system fills) go directly into the appropriate L1 cache and evict any existing L1 entry, which moves into the L2 cache. System fills typically are not placed into the L2 cache directly. Similarly, most common-case CPU core L2 cache hits are invalidated from the L2 cache and placed into the requesting L1 cache.[5,6]

The shared L3 design plays two distinct roles, exploiting the fact that the L3 cache and memory controller are colocated on the same die:

- a traditional cache associated with the processor; and
- storage for a cache directory (probe filter), implemented in fast SRAM, associated with the memory controller.

Most silicon nodes will likely be packaged as uniprocessors for desktop and workstations. As such, we did not want to waste die area on a dedicated on-chip probe filter, which is useful only in multiprocessor configurations. Our implementation allows the probe filter to be enabled in a multiprocessor configuration or disabled in a uniprocessor desktop or workstation configuration.

The probe filter is also known as HyperTransport Assist (HT Assist) at the platform level. This is a reference to one benefit of the probe filter, which is to conserve system bandwidth; a second benefit is to reduce memory latency.

Multichip module package

The processor packages two dies in a tightly coupled multichip module (MCM) configuration to yield a 12-core processor architecture. The processor interfaces to four DDR3 channels and four HT3 technology ports, as Figure 2 shows. The package is a 42.5 × 60-mm organic land grid array (LGA) and has 1,944 pins—735 more than the earlier-generation AMD Opteron L1SP 1,207-pin package. This new socket is known as *G34* (Generation 3, four memory channels). It has 1,132 signal I/O, 341 power, and 471 ground pins organized as a 57 × 40 array of contacts on a 1-mm pitch.

The HT3 ports are ungangable since each ×16 HT3 port can operate as two independent ×8 HT3 ports. This allows us to build highly optimized 2P and 4P blade server topologies by configuring a node as a router with four to eight ports in the network. If pins had not been a constraint, we would have brought out six HT3 ports on the package—three from each node—to allow for total platform-level flexibility. However, pin constraints imposed a limit of four HT3 ports on the package, as Figure 2 shows. An additional design constraint was the requirement that a single high-bandwidth device, such as a GPU, have access to the full ×16 noncoherent HT3 (ncHT3) I/O bandwidth. A single wide ncHT3 link connects to the lower node. Thus, the MCM topology is asymmetric with respect to ncHT3 but symmetric with respect to coherent HT3 (cHT3). The lower node has four wide

×16 HT3 ports, which we allocate as follows:

- one ×16 link for I/O,
- one ×16 plus one ×8 HT3 link to connect to three other sockets in the 4P topology, and
- one ×16 plus one ×8 HT3 link to connect to the other node in the package.

We balance traffic across the pair of on-package HT3 links by routing all ordered traffic (such as I/O direct memory access [DMA] read and write) on the primary ×16 HT3 link, and all unordered traffic (such as cache probes and responses) across the least recently used of the link pair.

2P and 4P blade architectures

Network diameter and *Xfire* (crossfire) bandwidth are two useful metrics for evaluating and comparing topologies. Network diameter is the maximum number of hops to traverse between any pair of nodes in the system. Xfire bandwidth is the maximum coherent memory-read bandwidth in the system when each node accesses its own memory and that of every other node in a round-robin fashion, assuming that the HT3 links are the only limiting factor. The Xfire bandwidth metric is superior to the network-oriented bisection bandwidth metric[7] because it measures useful bandwidth (that is, data bandwidth) and captures the interaction of topology, routing tables, and protocol overhead.

Figure 3a illustrates the recommended 2P blade topology. It has a diameter of 1, or minimum, latency, a key benefit for commercial workloads (hence the name 2P Max Perf). Assuming uniform distribution of traffic (Xfire), the ×8 diagonal links determine the maximum system bandwidth. Using the two horizontal ×16 links reduces the likelihood of hot spotting. We could disable the on-package ×8 HT3 link in this configuration to conserve power without adversely affecting either Xfire bandwidth or average diameter.

Figure 3b shows the recommended 4P topology in platforms with four HT3 I/O channels (4P Max I/O). It consists of two fully connected planes, (P2, P3, P6, P7) and (P0, P1, P4, P5), interconnected by the on-package pair (×16 and ×8) of HT3 links. This topology has a diameter of 2 and an average diameter of 1.25. For a uniform traffic distribution memory access pattern such as Xfire, half of the traffic from each node traverses the on-package HyperTransport link pair and 1/8 of the traffic routes over the ×8 HyperTransport links in each fully connected plane.

Figure 3c shows an alternative 4P topology with two HT3 I/O channels (4P Max Perf). This topology trades I/O connectivity for a more fully connected topology, achieving a reduced average diameter of 1.19. This topology is less susceptible to hot spotting and has lower average latency because of the additional links.

Figure 3d shows yet another 4P topology built with two identical 2P blades. This topology provides a pay-as-you-go upgrade path for customers wanting to start small (2P) and increase the number of processors, memory capacity, and memory bandwidth by adding a second 2P blade. Table 1 shows the Xfire bandwidth and diameter metrics for the four topologies.

Configurable L3 cache

The 6-Mbyte L3 cache is a victim cache, installing lines that are evicted from any of the core L2 caches. Each of its four subcaches contains tag and data macros to form a 1- or 2-Mbyte 16-way associative cache. We built the L3 cache from two 1-Mbyte subcaches and two 2-Mbyte subcaches. We chose this configuration purely for silicon area considerations. The architecture itself allows one, two, or four subcaches, and each can be 1 or 2 Mbytes. This allows flexibility in system-on-chip (SoC) layouts and lets us use different subcache building blocks in different SoCs with reduced effort. For example, some prior-generation processors used two 1-Mbyte subcaches; other processors might use four 2-Mbyte subcaches, with little impact to the overall microarchitecture. We can apply cache probe operations to all subcaches in parallel, or restrict them to a specific subset based on address. The L3 cache allocates lines to an available subcache, accounting for any address-based restrictions and giving preference to any subcache containing

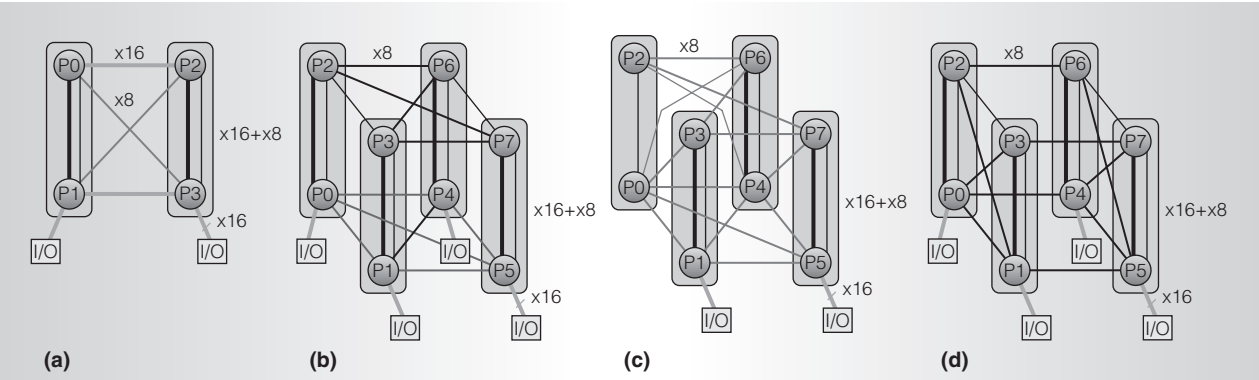


Figure 3. Dual processor (2P) blade and quad processor (4P) rack topologies: 2P Max Perf (a), 4P Max I/O (b), 4P Max Perf (c), and 4P Modular 2+2 (d).

Table 1. Xfire bandwidth and diameter metrics for the topologies in Figure 3.

Topology	Diameter (hops)	Average diameter (hops)	Bandwidth using broadcast (Gbytes/sec)	Bandwidth using the probe filter (Gbytes/sec)	Bandwidth increase using the probe filter (%)	DRAM bandwidth using DDR 1333 (Gbytes/sec)
2P Max Perf	1	0.75	76.5	125.9	64	85.2
4P Max Perf	2	1.19	56.7	143.4	152	170.4
Max I/O	2	1.25	56.7	143.4	152	170.4
Modular 2+2	2	1.25	38.4	71.7	86	170.4

invalid entries in the indexed set. If no invalid entries are present, a weighted round-robin algorithm distributes allocations among the subcaches in proportion to their size. On a valid replacement, the L3 cache chooses the subcache to victimize first (using weighted round-robin), then applies a pseudo least recently used (LRU) algorithm within the 16-way associative subcache for final victim selection.

The L3 cache supports a directed addressing mode in which the L3 cache can be sliced based on address. In this mode, both allocations and probes are directed to half of the cache (a pair of subcaches) based on a hash of the address bits. This reduces power and could reduce latency. Each tag probe only ties up resources in part of the cache, letting the L3 cache controller issue more speculative reads in parallel. However, some applications might be sensitive to the reduced effective associativity in this mode, so the benefit is workload dependent.

Architecting the L3 as a victim cache reduces overlap between the contents of the L3 and L2 caches, allowing more data to be cached. In addition, back invalidation of the L2 caches is not required when L3 lines are victimized. The L3 cache can retain a line after providing a copy to a requesting core when true sharing of a line is detected. The core that victimized a line is stored in a field in the L3 tag entry for the line and is used to detect sharing patterns by the L3 cache controller. If the next read is from the same core that allocated the L3 line, the data is deemed private and the L3 cache does not retain a copy. Instead, it passes responsibility for the line back to the core by providing the line in E or M state and invalidating the line in the L3 cache. Otherwise, if it is consistent with the core

request type, the L3 cache will keep a copy in anticipation of further sharing. This mechanism lets a core evict E and M state lines to the L3 cache and later retrieve them in the same state if they are not shared.

Subcache optimization. The tags in the subcache are physically located near the L3 controller to provide hit/miss results with minimum possible latency. The data array within the subcache is divided into four regions, each providing 128 bits of data. Both the control signals to the data regions and the read data return path are pipelined, causing round-trip latency to each successive region to increase by one clock. As a result, the subcache can provide four consecutive 128-bit data values even though the last value might be located a significant distance from the tags. This provides flexibility in placing the data macros and facilitates efficient floorplanning.

Latency optimization. The L3 cache architecture dynamically optimizes both latency and bandwidth. When the cache is lightly loaded, it operates in a latency-reducing mode in which a processor-initiated tag probe assumes a hit and reserves the necessary data buses and buffers in advance. This allows the read of the data macro to be overlapped with the tag probe, which minimizes latency. However, when the request rate is high and the L3 cache does not contain enough resources, the L3 cache controller sends the request to the tags as a query only, which will determine the cache status without initiating a data transfer. If the line is not present in the cache, it forwards the request to DRAM with minimal latency. Otherwise, it issues an L3 data read to the subcache containing the line once the necessary resources are available.

Bandwidth. The L3 cache controller can issue one processor-initiated tag probe or tag update and one probe-initiated tag probe or tag update per clock. Each subcache provides access ports to the required two tags as well as two read data buses and one write data bus. A dedicated read buffer holds the data from each subcache until it can be returned to the requesting core. Data from any subcache can be returned to any core. One dedicated write buffer per core holds allocation data until it can be written to the cache. Data from any core can store to any subcache. Each L3 data bus can sustain a bandwidth of 16 bytes per clock. The required tag accesses limit the combined L3 data read and write bandwidth to two 64-byte cache line accesses per clock.

Coherency. For requests that miss in the L3 cache, the memory controller is the ordering point between requests from different cores. When data movement is to or from the L3 cache, the L3 controller must ensure that none of the coherency rules are violated. In particular, it must ensure that there are no races between probes to a core and hit data being returned to the same core, or between probes to the L3 cache and victim data from the cores being allocated into the L3 cache. To achieve this goal, the L3 controller performs collision detection of probes against L3 cache data movements before delivering the probes to the core. This is a low-latency operation because it only probes the L3 queues; it performs the L3 cache probe after collision detection and in parallel with the core-cache probe. If a conflict exists, the probe can be delayed if the data movement is guaranteed to complete in a deadlock-free manner. Otherwise, the data movement is delayed and the probe is ordered ahead of the processor operation. When this occurs, the L3 controller applies the probe-state update to the L3 tags as well as conflicting L3 allocations, which have yet to update the tags. Dependency tracking ensures that these conflicting operations are completed in the correct order to maintain coherency.

With multiprogrammed workloads, if a program running on one core has poor caching characteristics, it could negatively affect the cache performance of programs running on other cores. This situation can be detected when a core exhibits a high install rate and a low cache hit rate, which indicates that it is not benefiting much from the L3 cache and is probably polluting the cache for other cores. To address

this situation, the L3 controller implements the *block aggressive neighbors* replacement algorithm. The BAN algorithm computes each core's cache efficiency based on its allocation rate and L3 cache hit rate. It limits the allocations of those programs or cores determined to be getting little benefit from the cache while causing significant pollution into the L3 cache. Rather than promoting an allocated line to the most recently used (MRU) position, the BAN algorithm sets the program lines to either the LRU position or half-way through the LRU stack (position 8). As a result, a poorly behaving program cannot negatively affect the most frequently accessed lines within the L3 cache.

Memory controller and DRAM interface

"Magny Cours" continues the tradition of the AMD Opteron family with an on-die memory controller, but adds DDR3 capability. The DRAM channels are configured for unganged (independent, 72-bit versus combined 144-bit) operation for maximum throughput and DRAM efficiency. The memory controller supports full single-error correction and double-error detection (SECDED) and x4 Chipkill. It does not support ganged (144-bit) DRAM channel operation because the natural DDR3 burst length (8) with a ganged channel leads to 128 bytes of data return, but the AMD Opteron cache line size is 64 bytes, leading to 64 bytes of superfluous data. It does not use DDR3 burst four-chop mode to mitigate this because DRAM performance is significantly reduced in this mode. Maximum capacity configurations support up to three dual-rank, two quad-rank, or two dual-rank plus one quad-rank DDR3 device configurations per channel. At 1.5V, a maximum operating frequency of RDDR3-1333 will be available, subject to platform design for lower capacities (up to two dual-rank registered dual in-line memory module [RDIMM] per channel). Maximum capacity configurations have operating frequencies of either 800 or 1,066 megatransfers per second, depending on platform routing and detailed DIMM characteristics. The memory interface supports 1.35V (DDR3L) operation, leading to lower speeds in some configurations.

The memory controller and DRAM interface have several notable features.

First, they offer memory-controller-based prefetch for CPU and I/O traffic, allowing prefetch chaining with CPU core prefetchers. This includes adaptive throttling when the DRAM interface is heavily used.

In addition, they provide adaptive prefetch of DRAM requests in parallel with local L3 tag accesses to minimize latency for L3 misses to local DRAM along with L3 hit/miss predictors. The L3 hit/miss predictors use a combination of per-page (4-Kbyte region) recent L3 access behavior and per-core local L3 hit/miss ratio to guide prefetch decisions.

A third feature is overlapped DRAM access and directory lookup, which minimizes DRAM latency when HT Assist is enabled. A directory hit to dirty (or potentially dirty) data in another cache in the system will cancel a DRAM data response to the requesting processor, which saves interconnect bandwidth. The memory controller issues probes as early as possible to minimize indirection latency.

The memory controller and DRAM interface also offer optimized DRAM page management and DRAM command bus utilization. Each memory channel has a dedicated DRAM controller that supports timeout-based and predictive page closing utilizing DRAM bank history. The DRAM controller allows aggressive reordering for high DRAM efficiency with out-of-order scheduling of both command (precharge, activate, and so on) phase and data (CAS) phase between multiple DRAM banks simultaneously.

Finally, the memory controller and DRAM interface can collect multiple DRAM write transactions outside the DRAM schedulers until many writes are available to be handled in a burst (*write bursting*) to avoid costly DRAM read-to-write and write-to-read bus turnarounds.

Cache coherence protocol

The cache coherence protocol is an important aspect of CMP systems. Generally, designers favor broadcast-based protocols when either overall protocol simplicity or latency for cache-to-cache transfers is more

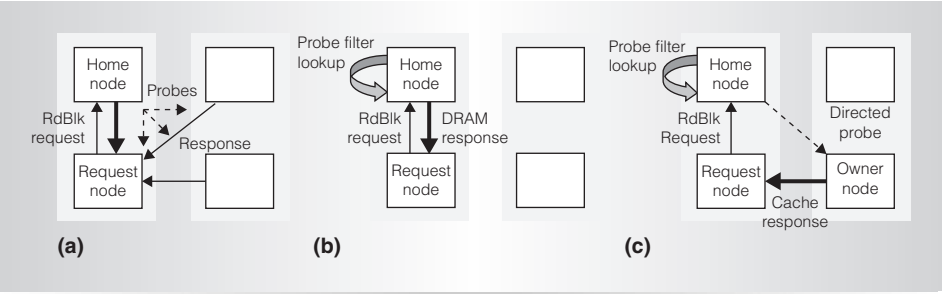


Figure 4. Cache coherence protocol examples: broadcast protocol (a), HT Assist with clean data (b), and HT Assist with dirty data (c).

important than interconnection and probe bandwidth. Directory-based protocols can be easier to scale up to larger systems,[7,8] but they also can be difficult to implement and verify. In addition, in many cases, the directory is stored in DRAM at the home node, which leads to relatively long indirection latencies, more complex protocols, or complex directory-caching mechanisms with performance policies defining which directory entries to cache.

HT Assist is a key innovation in "Magny Cours" and the AMD Opteron processor code-named "Istanbul." The HT Assist directory protocol gains many of the key scalability advantages of classic directories without using DRAM-based directories (for example, by repurposing ECC bits), and also maintains low indirection latencies. It does this by maintaining a cache directory, which fits naturally within the existing AMD Opteron processor broadcast transaction flow.

Review of broadcast protocol

Figure 4a illustrates the broadcast coherence transaction flow used by earlier generations of AMD Opteron processors.[5]

In the broadcast protocol, last-level cache misses go to the home node (where DRAM for the requested cache line resides) and the memory controller determines the ordering for each request for the same cache line. Once a request becomes active, the home node broadcasts cache probe requests to all processors, and the memory controller initiates a DRAM access. All processors send probe responses directly to the requesting processor, and the memory controller (DRAM) sends a separate response when memory data is available. The requesting processor collects all responses, including cache data responses, and determines which data should be used to satisfy the original request. Once it receives all responses, the requesting processor delivers data to the CPU core and sends a transaction (not shown in the figure) to the home-node memory controller indicating that the cache line request is complete and another request for the same cache line can be activated.

HyperTransport Assist directory protocol

HT Assist adds a cache directory. Each home node keeps track of which cache lines from its memory are cached by other processors in the system. The directory includes all cached data in the system. If a cache line is present in any cache, there must be an entry in the home node's directory to indicate that the line is cached within the system. If a directory is full or a mapping conflict occurs, a previous directory entry must be replaced (causing a potential writeback, plus invalidation of the previous entry's data from all caches) to accommodate the new request.

HT Assist's transaction flow is similar to the broadcast protocol. Initial requests travel to the home node, where the memory controller orders and activates them. Once a request is active, instead of broadcasting probes, it begins a DRAM access and a probe filter lookup in parallel to minimize DRAM latency. When the directory lookup is complete, the memory controller generates a broadcast probe, a directed probe (a probe targeting a single processor), or no probe, depending on directory state. The time to access the probe filter and generate a directed probe is the *indirection latency*. The protocol

guarantees that a directed probe never requires a DRAM response, so any DRAM response is canceled. When no probe is generated, the memory sends a data response with an indication that this is the only response to expect and that the request can now be completed. Broadcast probes follow a similar flow to the previous broadcast protocol, except this generation's protocol guarantees that data will be returned from the owner node. Figures 4b and 4c show a simplified version of the transaction flow for cases in which no probes, and a single directed probe, are required.

For most requests, the directory state is immediately updated after the initial directory lookup based on the request type and directory state. For some requests, such as a store to an S-state line in the requestor's cache, the request is always treated as a broadcast, and directory lookup and update are postponed until after the request completes. In all cases, only a single directory lookup and update is required, and these are treated as an atomic read-modify-write action. Designing the protocol in this fashion greatly simplified microarchitecture design and reduced protocol complexity.

The directory protocol must also enable alternate coherence behavior of the CPU-side caches (L1, L2, and L3). Notably, external read probe requests must transition E-state cache lines to O-state, return data to the requester, and send eviction notifications of E-state cache lines to the directory.

Directory storage

The design supports multiple directory sizes through a combination of per-way and per-subcache mappings of L3 space assigned to the directory. However, in its production form, only a subset of sizes and configurations are available for selection.

Because the directory storage is held in the fast L3 SRAM arrays, directory indirection latency is low and we can achieve sufficient bandwidth to the arrays to not limit per-node coherent bandwidth. We used the existing L3 arrays in lieu of dedicated directory storage to maximize processor flexibility, for the discussed latency and bandwidth characteristics, and to avoid additional pressure on the DRAM ECC coding. With existing AMD Opteron cache line sizes, and continual pressure to improve error detection and tolerance surrounding DRAM devices, we did not pursue an in-memory directory strategy. Most directory actions involve read-modify-write accesses that are implemented much more efficiently, and with minimum port occupancy, in SRAM than in DRAM.

Our design's 64-byte cache line holds 16 directory entries, with 4 bytes per entry organized as a four-entry four-way set associative array (see Figure 5a). The tag field tracks normalized addresses, which are computed by subtracting the home node's DRAM base address (or, for node-interleaved addressing, removes appropriate bits of the system's physical address) before storing the resulting address. Using the normalized address reduces the number of tag bits required and lets us size the probe filter tag field based on the maximum DRAM per node, not the total DRAM across all nodes in a system.

By default, the basic input/output system (BIOS) will allocate 1 Mbyte of the 6-Mbyte L3 cache to directory storage. The directory holds 256k directory entries, which can cover 16 Mbytes of cache. This results in a directory coverage ratio of 16 Mbytes/(0.5 Mbytes $\times$ 6 cores + 5-Mbyte L3), or 2.0, which says there are at least twice as many directory entries in a system as cached lines, since a single directory entry can track multiple cached copies of a shared line.

Directory states and transitions

Earlier, we showed the behavior of the directory protocol using some examples. Also important are the protocol's states and transitions, including special consideration for the shared L3 cache and CMP nature of each processor node.

Directory behavior. Figure 5b lists the directory states. The directory supports the full MOESI protocol from all previous AMD Opteron processors. It observes all requests, many of which lead to state updates. To maintain the directory semantics discussed earlier, and to keep the directory up to date, the directory protocol informs the directory of any cache castouts of M, O, and

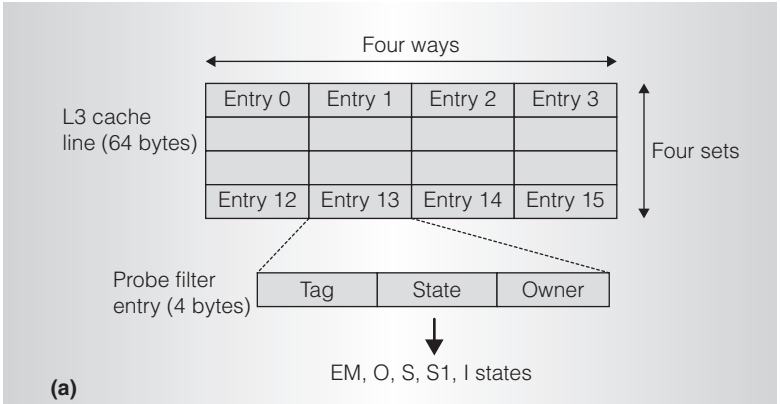
E state lines. Finally, each directory miss might find the directory index full of valid entries, one of which must be evicted to make space for the new entry. We refer to this final case as needing a downgrade probe. Such a probe causes a writeback (if dirty) and invalidates all existing cached copies of the downgraded cache line.

Figure 6 shows some common transaction scenarios. The transactions in the figure have the following semantics:

- *Fetch*—an instruction fetch request; install in S state by default.
- *Load*—a data read request; install in E state by default.
- *Store*—a data store miss request (write-allocate); install in M state.

In each case, the "Directory hit" columns indicate that a request hits a pre-existing directory entry to the same cache line address. The "I," "O," "S," "S1," and "EM" columns indicate the entry's directory state, and the table entries indicate the type of probe generated, if any. The "Directory miss" columns indicate scenarios in which the requested cache line has no pre-existing directory entry, the line is uncached, and no probe is necessary (filtered). In the directory miss case, the directory states indicate the state of the replaced line and the corresponding type of downgrade probe (broadcast downgrade or directed downgrade).

Although a downgrade probe does not block system activity for the demand request that caused it, minimizing downgrades to achieve high probe filtering rates and processor-side cache perturbation is still important. Therefore, informing the directory on M, O, and E state castouts, in combination with directory size and mapping, ensures that most directory miss requests find an available invalid directory location. The protocol does not include notifications of S-state castouts. Downgrade probes reclaim S-state lines in the directory. We omitted S-state evictions for several reasons, due both to performance (a potentially large number of eviction messages) and various microarchitecture-specific implementation complexities. Performance evaluation showed that S-state eviction



(a)

EM	A copy of the cache line is present on a single node, recorded in the owner field. The state of the line in the cache is not known at the directory. It may be clean (Exclusive) or dirty (Modified or Owned with all shared copies on the owner node).
O	The cache line is cached on one node, the Owner node, and possibly multiple Sharing nodes. The line is dirty and is written back to memory when cast out of the node.
S	The cache line may be present on multiple nodes and all copies are clean (Shared).
S1	The cache line is clean and present on a single node, recorded in the owner field.
I	The line is uncached.

(b)

Figure 5. Probe filter entry format (a) and directory states (b), which shows how probe filter entries map into the L3 cache line.

	Directory hit					Directory miss				
	I	O	S	S1	EM	I	O	S	S1	EM
Fetch	–	D	–	–	D	–	B	B	DI	DI
Load	–	D	–	–	D	–	B	B	DI	DI
Store	–	B	B	B	DI	–	B	B	DI	DI

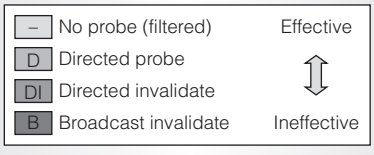


Figure 6. Probe filter transaction scenarios summarizing probe actions as a function of memory access type and probe filter state.

notifications were not critical for achieving acceptable directory performance.

Because the directory is shared with the processor-side L3 cache, AMD Opteron processors with HT Assist enabled must trade off directory size (minimizing downgrades) with processor-side L3 cache performance (eliminating last-level cache misses). We achieved an appropriate balance via extensive modeling and hardware-based evaluations. This tradeoff is particularly noteworthy because the directory's associativity is much lower (four-way) than the net associativity of all CPU-side caches. As we noted earlier, a directory coverage ratio of at least 2.0 provides significant leverage. Additionally, the directory index mapping uses hashing for many common-case scenarios considering the possible x86 page sizes (4 Kbyte, 2 Mbyte, and 1 Gbyte). In addition, interaction with operating system/Hypervisor page-coloring algorithms helps avoid pathological directory mapping collisions in multiprogrammed scenarios. Finally, directory replacement policies attempt to avoid victimizing lines that are cached in many CPUs to reduce CPU-side cache perturbation resulting from directory downgrades.

CMP considerations. Our description of the protocol and states has not specifically mentioned interactions with the various levels of private (L1 and L2) and shared (L3) caches on each processor node. The HT Assist directory treats each set of CPU cores, private caches, and shared L3 cache as a unit (node). All of the transaction flows and state transitions discussed previously are faithful representations of the directory protocol behavior for the AMD Opteron cache hierarchy. The directory is never involved (that is, a message is never sent to the directory) in any internal cache transitions or movement of a cache line between layers in the cache hierarchy within a processor node (for example, movement of a cache line from L3 to L1, or from L2 to L3).

Performance

We evaluated the HT Assist directory both in simulation and by making measurements on preproduction silicon.

The simulation-based performance results presented here use AMD-internal performance models. The model used in this article has a detailed representation of the on-die northbridge (L3 cache, microarchitecture, HT3 links, DRAM controllers/devices, and so on), is designed to be cycle-accurate at the northbridge level, and is correlated against the register transfer level (RTL) design presilicon. The model can use detailed, cycle-accurate, correlated-to-RTL CPU core models (representing AMD's most precise CPU-to-system modeling environment) or abstract CPU core models when simulation efficiency is paramount. For practical reasons, we use the abstract CPU core model. One model input is coherence transaction traces, taken from existing AMD Opteron multiprocessor systems. These traces are essentially lists of L1 cache miss records for fetches, loads, and stores issued from the cores in the system. Each trace record contains metadata that specifies interthread ordering for accesses to shared data so that the simulator can enforce for deterministic execution. The abstract core model consumes these traces when running a simulation. The abstract core model includes a representation of core caches (L1 and L2), CPI, memory-level parallelism, and core frequency. The simulator models the memory traffic from the abstract core model cycle accurately within the northbridge model, using total execution time for a constant set of references from each input trace on each CPU as the performance metric.

The hardware performance measurements used preproduction "Magny Cours" hardware in AMD's performance labs. The CPU and on-die northbridge operational frequencies and other system parameters are representative of final shipping configurations.

Transaction scenario frequencies

Figure 7 shows the probe filter transaction scenario frequencies from early 4P hardware measurements for SPECJBB2005. Most requests lead to no probe or to a directed probe; therefore, the HT Assist directory eliminates a large amount of probe and response traffic.

If we attach a weight of 0 to filtered, 0.125 to directed (one probe in place of eight for broadcast), and 1 to broadcast, the coherence protocol overhead with probe filter

is (72.2 percent × 0) + ([24.9 percent + 1.6 percent] × 0.125) + (1.3 percent × 1), or 4.6 percent of the broadcast coherence. Thus, the protocol is more than 95 percent effective in reducing probe traffic.

On the other hand, the directory protocol requires that clean victims be sent to the home node whenever E lines age out of the last-level cache (one clean victim in place of eight probes), which introduces 66 percent × 0.125 (or 8.25 percent) additional traffic overhead corresponding to table entry {Load, PF Miss, I}. So, the net effectiveness—or reduction in traffic associated with maintaining cache coherence in this theoretical design—is greater than 87 percent.

This example shows that the traditional cache-hit ratio is not an appropriate measure of the probe filter's effectiveness (in this example, the directory-hit ratio is only 14.6 percent).

Note that downgrades occur in the background and can be overlapped with new memory requests. Additionally, the reduced L3 victim traffic offsets the modest bandwidth overhead of downgrades.

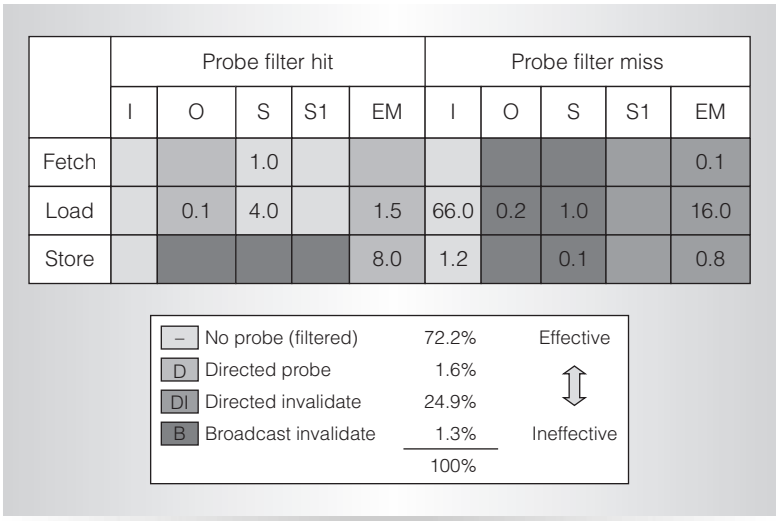


Figure 7. Probe filter transaction scenario frequencies for SPECJBB2005. Each table entry is a percentage of all requests.

NUMA software optimizations

Like previous generations of AMD Opteron processors, "Magny Cours" is a distributed shared memory machine that benefits from operating system/application tuning to optimize memory allocation and process scheduling. Modern operating systems, such as Windows and Linux, are aware of the underlying machine node topology via the Advanced Configuration and Power Interface (ACPI) static resource affinity table/system locality information table (SRAT/SLIT) supplied in BIOS.[9] These tables associate memory with nodes and provide a matrix to assign a latency cost for accessing memory for all {source, destination} node pairs. The Linux nonuniform memory architecture (NUMA) library supports a command line utility, *numactl*, which defines the default memory allocation policy (local, interleaved, or user specified) for all threads spawned by a process and specifies on which cores they should run. The shared library *libnuma* provides applications with an API to control the process policy for allocating new or existing memory and scheduling threads on sets of nodes.

Having the probe filter on "Magny Cours" reduces the observed latency of memory accesses when probes and probe responses are the longest path. By significantly reducing local memory latency, the probe filter amplifies the benefit of NUMA software optimizations.

Memory latency and bandwidth

Figure 8a shows preproduction hardware improvements in maximum main memory bandwidth when running STREAM triad, and Figure 8b shows improvements in local and one-hop memory latency in 2P and 4P (four-node and eight-node, respectively) system configurations. Latency measurements are for DRAM page hits. Each measurement is relative to HT Assist disabled for that platform (2P or 4P), and the 2P and 4P relative memory latencies cannot be compared because they are not normalized to the same baseline.

As these tests show, enabling HT Assist significantly increases memory bandwidth and reduces memory latency. Notably, the memory bandwidth and latency improvements are larger in 4P systems than in 2P systems, illustrating the greater benefit from HT Assist in these larger systems. As might be expected, the improvement in local DRAM latency is larger than one-hop (and two-hop in 4P) latency. We therefore expect NUMA-optimized workloads with high processor-memory affinity to benefit

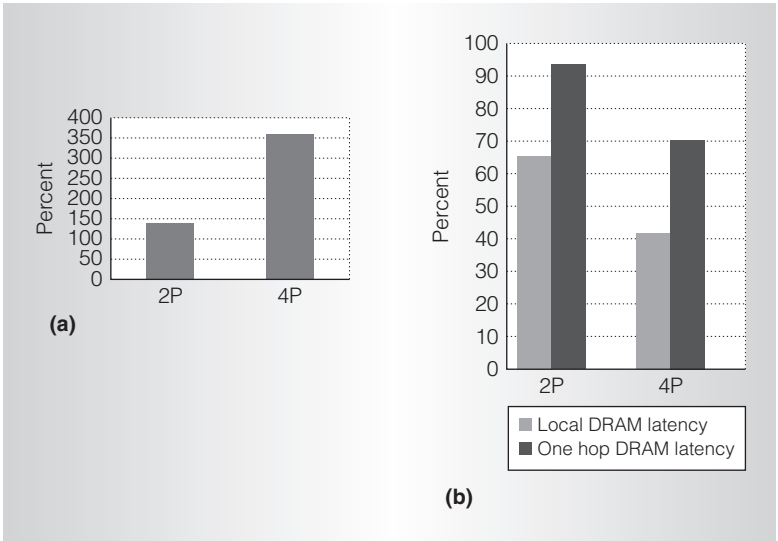


Figure 8. Memory bandwidth improvement (a) and relative memory latency (b) with and without HT Assist. Latency is normalized to HT Assist disabled as 100 percent (lower is better).

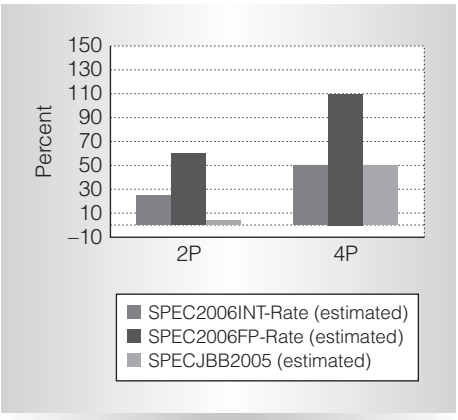


Figure 9. Early hardware measurements of application improvement with HT Assist.

more from HT Assist, and that benefit will be greater in 4P systems.

Application benchmarks

Figure 9 shows early hardware measurements of the benefit of HT Assist for application-level workloads. HT Assist benefits 4P systems more than 2P systems because of the larger relative reduction in memory latency and bandwidth. Note that performance scaling from 2P to 4P (not shown) is greater than 95 percent in these cases, illustrating that HT Assist could enable superior system scalability.

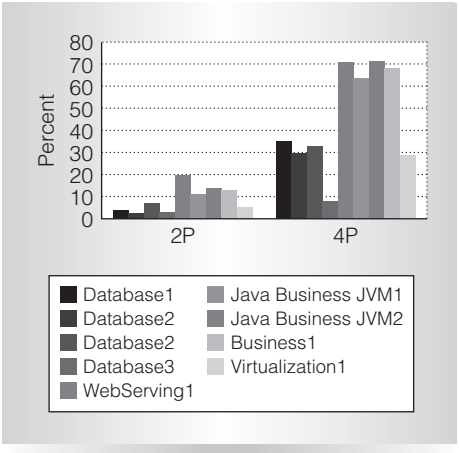


Figure 10. Simulation-based studies of performance improvement with HT Assist across a diverse set of commercial workloads.

Figure 10 shows simulation-based studies of additional workloads with configurations similar to the hardware measurements.

Although real-world results might vary, our simulations demonstrate that HT Assist improves performance more dramatically in 4P configurations than in 2P, and that it benefits the WebServing, Java Business, and Business workloads more significantly than the Database and Virtualization workloads. Detailed evaluation of the underlying performance data (not shown here) indicates that Java workloads have good processor-memory affinity and significant need for memory bandwidth, making them a good fit for HT Assist. The WebServing and Business workloads benefit primarily from the additional achieved memory and interconnection bandwidth, and less from the significant improvement in local memory latency because they have less processor-memory affinity. The Database and Virtualization workloads benefit from reduced memory latency in 2P, but have less processor-memory affinity and less overall memory bandwidth need; therefore, the gains are smaller compared to other workloads. These workloads are more sensitive to reduced L3 capacity. In 4P, the larger relative improvement in average memory latency and the greater number of CPUs contending for effective memory bandwidth lead to larger relative gains compared to 2P.

The "Magny Cours" combination of superscalar cores, high core count, and virtualization support make it an appropriate choice for running a heterogeneous mix of workloads in the data center. The G34 socket infrastructure provides sufficient interconnect and memory bandwidth headroom to accept upgrades of future generations of plug-compatible processors, which are already in development. MICRO

Acknowledgments

An exceptional team of AMD technical staff designed, verified, and tested the "Magny Cours" processor.

References

1. K. Olukotun, L. Hammond, and J. Laudon, *Chip Multiprocessor Architecture: Techniques to Improve Throughput and Latency,* Morgan & Claypool, 2007.
2. J. Brutlag, "Speed Matters for Google Web Search," Google, http://code.google.com/speed/files/delayexp.pdf.
3. K. Asanovic et al., *The Landscape of Parallel Computing Research: A View from Berkeley,* tech. report UCB/EECS-2006-183, Electrical Eng. and Computer Science Dept., Univ. of California, Berkeley, 2006.
4. C. Keltcher et al., "The AMD Opteron Processor for Shared Memory Multiprocessor Systems," *IEEE Micro,* vol. 23, no. 2, 2003, pp. 66-76.
5. P. Conway and W. Hughes, "The AMD Opteron Northbridge Architecture," *IEEE Micro,* vol. 27, no. 2, 2007, pp. 10-21.
6. C. Moore, and P. Conway, "General Purpose Multiprocessors," *Multicore Processors and Systems,* S. Keckler, K. Olukotun, and P. Hofstee, eds., Springer, 2009.
7. D. Culler, J. Pal Singh, and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach,* Morgan Kaufmann, 1999.
8. J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach,* 4th ed., Morgan Kaufmann, 2007.
9. A. Kleen, "A NUMA API for Linux," SUSE Labs white paper, Aug. 2004.

Pat Conway is a principal member of the technical staff at AMD. His technical interests include server design and accelerated computing. Conway has a masters in electrical engineering from University College Cork, Ireland, and an MBA from Golden Gate University. He is a member of the IEEE.

Nathan Kalyanasundharam is a principal member of AMD's technical staff. His technical interests include the design of the load/store unit, cache controller, and northbridge. Kalyanasundharam has an MS in electrical engineering from Texas A&M University.

Gregg Donley is a senior member of AMD's technical staff. His technical interests include third-level cache and system architecture. Donley has an MS in computer engineering from the University of Michigan.

Kevin Lepak is a principal member of AMD's technical staff. His technical interests include various aspects of next-generation server platforms and processor development. Lepak has a PhD in electrical and computer engineering from the University of Wisconsin.

Bill Hughes is a senior fellow at AMD, where he leads the Northbridge and HyperTransport Microarchitecture team. He has a PhD electrical and electronic engineering from Leeds University, England.

Direct questions or comments about this article to Pat Conway, Advanced Micro Devices, MS 362, 1 AMD Place, Sunnyvale, CA 95070; Pat.Conway@amd.com.