Comparison of High Performance Northbridge Architectures in Multiprocessor Servers

Michael Koontz
MS CpE Scholarly Paper
Advisor: Dr. Jens-Peter Kaps
Co-Advisor: Dr. Daniel Tabak

Table of Contents

1. Introduction

With the continuing growth of today's multi-media, Internet based culture, businesses are becoming more dependent on high performance, high throughput data servers. There are two competing front-runners to this market space; servers based on the Intel Xeon processor and servers based on the AMD Opteron processor. In modern computer architecture, data communications between the central processing unit (CPU) and other devices (memory, storage, network interface, etc.) are handled by two chips, know as the northbridge and the southbridge. The northbridge is responsible for communications with devices that are high speed, including memory, peripheral component interconnect (PCI) express, and accelerated graphics port (AGP) graphics devices. Additionally, the northbridge controls communications between the CPU and the southbridge. The southbridge is responsible for communications with low speed devices, including legacy PCI, keyboard, mouse, universal serial bus (USB), and many others. This paper will introduce the northbridge architecture of the AMD Opteron family of CPUs and the Intel Xeon family of CPUs, and will analyze the strengths and weaknesses or each. The Intel Xeon family northbridge (known as Blackford) is based on the standard northbridge/southbridge organization, while the AMD Opteron family is based on a very different, point-to-point architecture.

The paper will start by providing background information on the technologies that AMD and Intel have built their processors around. Next, the paper will describe the architecture of the AMD Opteron Northbridge, and the Intel Blackford Northbridge. Following that, performance data will be presented in an effort to compare how each architecture handles pure data processing, as well as how each architecture performs from a power consumption standpoint. Finally, some over-all conclusions and comments on additional, non-quantifiable aspects of the performance of both architectures will be presented.

2. The x86-64 Instruction Set Architecture (ISA)

The Opteron processors include one to four independent processor cores. Each of these cores is based on AMD's x86-64 instruction set architecture. The

AMD64 ISA is a full 64-bit superset of the standard x86 ISA. It natively supports Intel's standard 32-bit x86 ISA, allowing existing programs to run correctly without being ported or recompiled. AMD64 was created to provide an alternative ISA to the Intel Itanium architecture. The Itanium architecture is drastically different from standard x86, and is not compatible in any way. The main defining characteristic of AMD64 is that it supports 64-bit general purpose registers, 64-bit integer arithmetic and logical operations, and 64-bit virtual addresses.[1] Additional significant features include full support for 64-bit integers, 16 general-purpose registers (as opposed to 8 in standard x86), 16 general-purpose 128-bit XMM registers (as opposed to 8) used for streaming Single Instruction, Multiple Data (SIMD) instructions, a larger virtual and physical address space, instruction pointer relative data access, and a no-execution bit. Due to large performance enhancements possible by using 64-bit registers and native 64-bit operations, it is clear that the AMD64 ISA is a cutting edge instruction set architecture.

Due to lack-luster sales and support of processors based on the Itanium ISA, Intel has cloned the AMD64 ISA, creating its own version known as Intel 64 for use in its multi-core 64-bit processors. The Xeon processors used with the Blackford Northbridge are based on this ISA. Other than a few minor differences, the Intel 64 ISA is the same as the AMD64 ISA.

3. Memory Coherency

One of the biggest challenges faced by designers who are implementing systems that have caches and multiple processors is correct management of memory coherency. Memory coherency is a problem created by a system needing to maintain multiple copies of the same piece of information. When a system has only one CPU, and no cache (main memory only), memory coherency is not a problem. In this situation, the CPU will directly modify information in the main memory, and there is no other copy of this information. Managing memory becomes more difficult when there are multiple processors sharing the same main

memory. The memory model of a shared memory multiprocessor formally specifies how the memory system will appear to the programmer. A memory consistency model restricts the values that a read can return. A read should return the value of the "last" write to the same memory location. In single processor systems, this is precisely defined by the program order. That is not the case in a multiprocessor. As shown in Figure 1, the write and read of the variable "Data" are not related by program order because they reside on two different processors.[2]

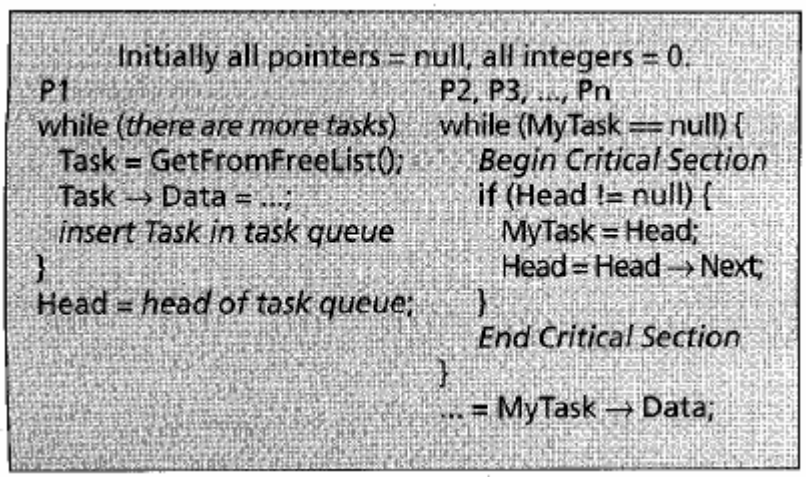

Figure 1: Illustrating the need for a memory consistency model.

Extending the single processor model to multiprocessors results in a model know as the sequential consistency model. This model requires that all memory operations appear to execute in the order described by that processor's program. While sequential consistency provides a simple, intuitive programming model, it disallows many single processor hardware and compiler optimizations. To address this problem, many relaxed consistency models have been proposed.

There are three requirements for a multiprocessor to be sequentially consistent:

- The result of any execution must be the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program;

- Maintaining program order among operations from a single processor;
- Maintaining a single sequential order among all operations.[2]

A sequentially consistent memory model can be looked at as consisting of a global shared memory connected to all processors through a switch. At any given time, only one processor can have a path through the switch to the memory. This provides global serialization among all memory operations.

Caching of shared data can lead to multiple scenarios that violate sequential consistency of data. Systems that use caching must take precautions to maintain the illusion of program order. Most important, even if a read hits in its processor's cache, reading the cached value without waiting for the completion of previous operations can violate sequential consistency. To address the problems with systems that have multiple processors (each with a cache), a cache coherency protocol must be implemented. This protocol needs to propagate a new value to all copies of the modified location. New values are propagated by either invalidating (eliminating) or updating each cached copy of the data. There are two general definitions of cache coherency. The first is very similar to the sequential consistency model. Others impose relaxed program orderings. One common definition requires two conditions for coherence:

- A write must eventually be made visible to all processors;
- Writes to the same location must appear to be seen in the same order by all processors.[2]

These conditions are not enough to satisfy sequential consistency. This is due to the fact that sequential consistency requires that writes to all locations (not just the same location) be seen in the same order by all processors, and also explicitly requires that a single processor's operations appear to execute in program order. A memory consistency model is the policy that places a boundary on when the value can be propagated to a given processor. There are other problems that need to be addressed. For instance, how can write completion be detected and how is write atomicity maintained? Typically, a write to a line replicated in other caches requires an acknowledgment of invalidate or update messages. These acknowledgments must be collected either at the memory or at the issuing

processor. Either way, the writing processor must be notified when all acknowledgements are received. Only then can the processor consider the write to be complete. To address maintaining write atomicity, it is important to prohibit a read from returning a newly written value until all cached copies have acknowledged the updates for the writes.

Maintaining a strict memory consistency model usually has negative impacts on performance. To address this problem, a relaxed memory model can be used. There are two characteristics to categorize relaxed memory consistency models. The first is the way they relax the program order requirement. Models differ on how they relax the order from a write to a following read, between two writes, and from a read to a following read or write. These relaxations apply only to operation pairs with different addresses and are similar to the optimizations for sequential consistency. The second characteristic is how they relax the write atomicity requirement. Some models allow a read to return the value of another processor's write before the write is made visible to all other processors. It is also possible to relax both program order and write atomicity. In this situation, a processor is able to read the value of its own previous write before the write is made visible to other processors and before the write is serialized. This relaxation is commonly used to forward the value of a write in a write buffer to a following read from the same processor. Relaxed models also provide methods for overriding their default relaxations.

One relaxation used is to allow a read to be reordered with respect to previous writes from the same processor. There are three models in this group: the IBM 370, total store ordering (TSO), and processor consistency (PC). These models all differ in when they allow a read to return the value of a write. The IBM 370 model provides serialization instructions that enforce program order between a write and a following read. In contrast, the TSO and PC models do not provide any explicit safety net functions. However, programmers can still use read-modify-write operations to provide the illusion that program order is maintained from a write to a read. Relaxing program order can substantially improve performance by hiding latency of write operations.

4. The MOESI and MESI Cache Coherency Models

The MOESI cache coherency model is a full cache coherency protocol that encompasses all of the possible states commonly used in other protocols.[3] This is the cache coherency model used in the AMD Opteron Northbridge. Each cache line will be in one of five states:

- Invalid — a cache line in the invalid state does not hold a valid copy of the data. Valid copies can be in main memory or another processor's cache.
- Exclusive — a cache line in the exclusive state holds the most recent, correct copy of the data. The copy in main memory is also the most recent, correct copy. No other processor holds a copy of the data.
- Shared — a cache line in the shared state holds the most recent, correct copy of the data. Other processors may hold copies of the data in the shared state. If no other processor holds it in the owned state, then the copy in main memory is also the most recent.
- Modified — a cache line in the modified state holds the most recent, correct copy of the data. The copy in main memory is incorrect and no other processor holds a copy.
- Owned — a cache line in the owned state holds the most recent, correct copy of the data. Other processors can hold a copy of the correct data, however, the copy in main memory can be incorrect. Only one processor can hold the data in the owned state; all other processors must hold the data in the shared state.[4]

The MOESI state transitions are shown in Figure 2.

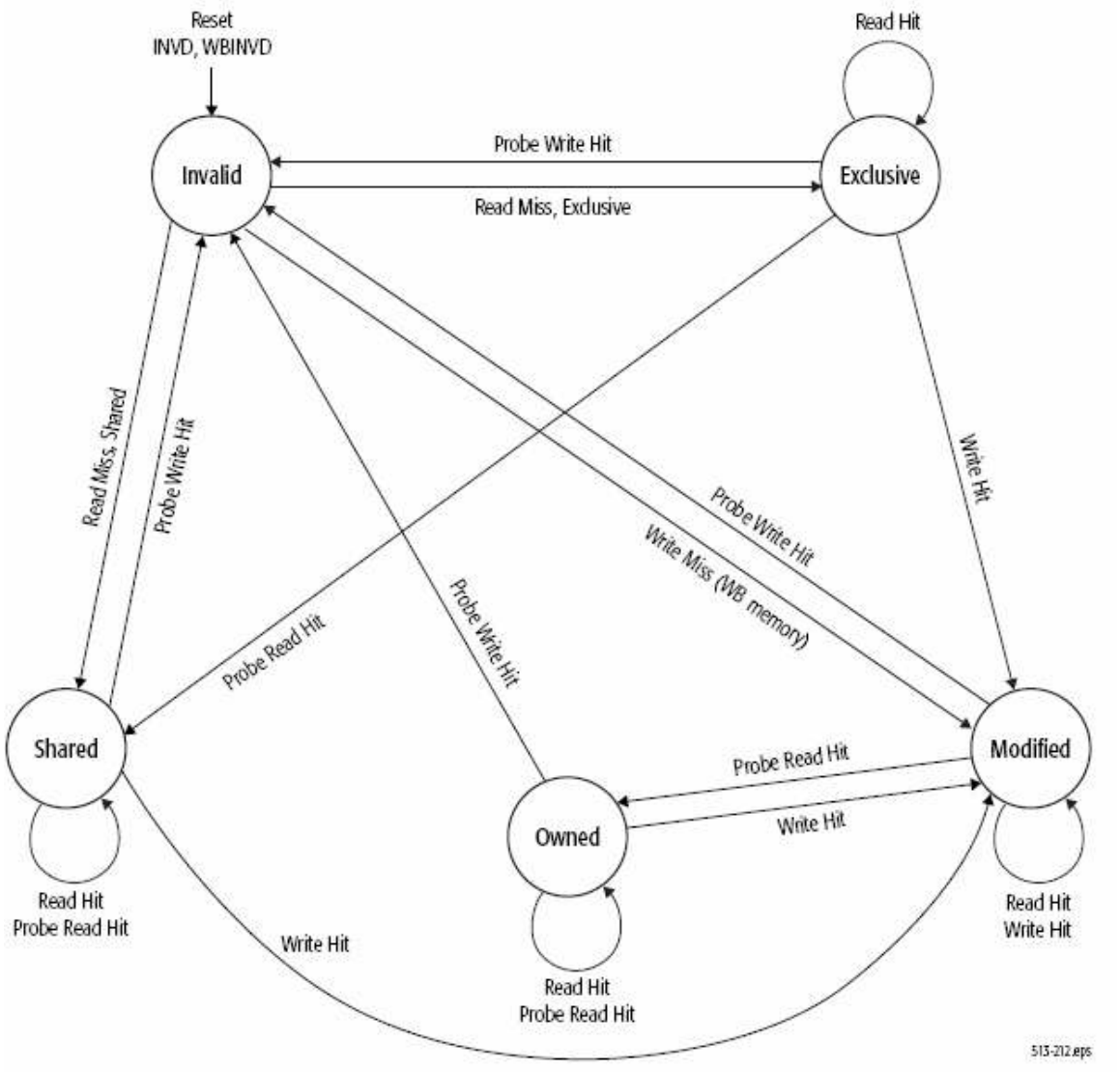

Figure 2: MOESI State Transitions

To maintain memory coherency, processors need to obtain the most recent copy of data before caching it internally. The copy can be in main memory or in the internal caches of other processors. When a processor has a cache read-miss or write-miss, it probes the other processors to determine whether the most recent copy of data is held in any other caches. If one of the other processors holds the most recent copy, it sends it to the requesting processor. Otherwise, the most recent copy is provided by the main memory.

There are two types of processor probes. Read probes are used when the processor is requesting the data for read purposes. Write probes are used when

the processor intends to modify the data after accessing it. State transitions involving probes are initiated by processors that are trying to find out if other processors hold a copy of the requested data. Read hits do not cause a MOESI state change. Write hits usually cause a state change into the modified state. If the cache line is already in the modified state, a write hit does not change its state. The MOESI protocol does not specify operation of external-bus signals, transactions, or how those transactions influence a cache line's MOESI state. The details of how this is handled is dependent on the implementation and is left up to the processor designer.

As an example of the AMD implementation of the MOESI protocol, refer to Figure 3. Processor 3 wants to access data that is located in the memory attached to processor 0. Initially, processor 3 looks up the system address map and using the physical address, determines that the data is attached to processor 0. The processor then sends a read request (RD) to processor 2. Processor 2 forwards the read request to processor 0. Processor 0 reacts by fetching the requested data from its internal memory controller (it might be stored in cache or main memory). Processor 0 also broadcasts a probe (PR) to processors 1 and 2. Processor 1 forwards this probe to processor 3. Each processor combines probe responses (RP) from each of its individual cores into a single probe response. Each processor then sends its probe response back to processor 3. If the line is modified or owned, then a read response is returned instead of a probe response. Once the source processor has received all probe and read responses, it provides the fill data to the requesting core. A source done (SD) is sent to the home processor to signal that all the transaction's side effects, such as invalidating all cached copies for a store, have completed and that the data is now globally visible. The memory controller is then free to process another request to the same address. The latency of memory operations is the longer of two paths: the time it takes to access dynamic random access memory (DRAM) and the time it takes to probe all caches in the system.[5]

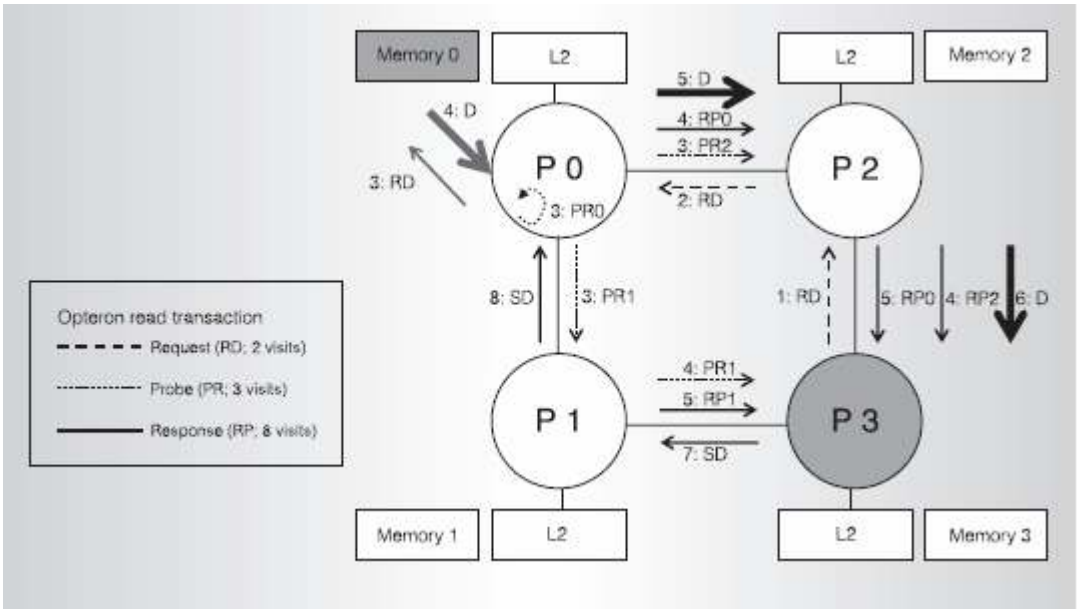

Figure 3: Traffic on the Opteron Processor

The Intel Blackford Northbridge uses a memory coherency protocol that is similar to MOESI, called MESI. The MESI protocol uses a data ownership model, meaning that only one cache can have data that is in the "dirty" state.[6] An additional implication of how the MESI protocol works is that writes are not broadcasted to all caches, which causes data invalidation. The bus activity in a MESI system is broken up into a Master/Slave relationship. The master starts an inquiry cycle telling all slaves that it intends to cache some data. The slaves respond with one of two signals: a Hit signal signifying that the slave has a copy of the data the master is about to cache, or a HitM signal signifying that the slave has a modified copy of the data a master is about to cache.

Each cache line has one of four possible states:

- Modified− the cache line is held exclusively in this cache and the content is modified from the copy that is in shared memory.
- Exclusive− the cache line is held exclusively in this cache and the content is the same as the copy that is in shared memory.
- Shared− the cache line is held in this cache, and is possibly held in other caches. The content of the cache line is the same as the copy that is in shared memory.
- Invalid− the cache line contains no valid copy of the data.[6]

The main operating idea behind MESI is that "modified" and "exclusive" cache lines are owned by the cache they are held in. Those cache lines are allowed to be modified without reporting the new value to other caches. Any attempt to access a cache line that is not "owned" results in a broadcast to all other caches to determine if the cache line is "owned" elsewhere. If it is "owned", the owner will notify the rest of the caches as to the current value of the cache line, and will give up ownership to the cache that initiated the request. The MESI state of each cache line is stored in a two-bit number. Table 1 below shows all of the possible cache line states:

Cache Line State	**Modified**	**Exclusive**	**Shared**	**Invalid**
Cache line is valid?	Yes	Yes	Yes	No
Status of copy in memory?	Out-of-date	Valid	Valid	N/A
Other cached copies exist?	No	No	Maybe	Maybe
Reaction to a write to this line?	Do not write to bus	Do not write to bus	Write to bus and update cache	Write to bus

Table 1: The MESI States

The MESI protocol has a state transition diagram as shown in Figure 4.

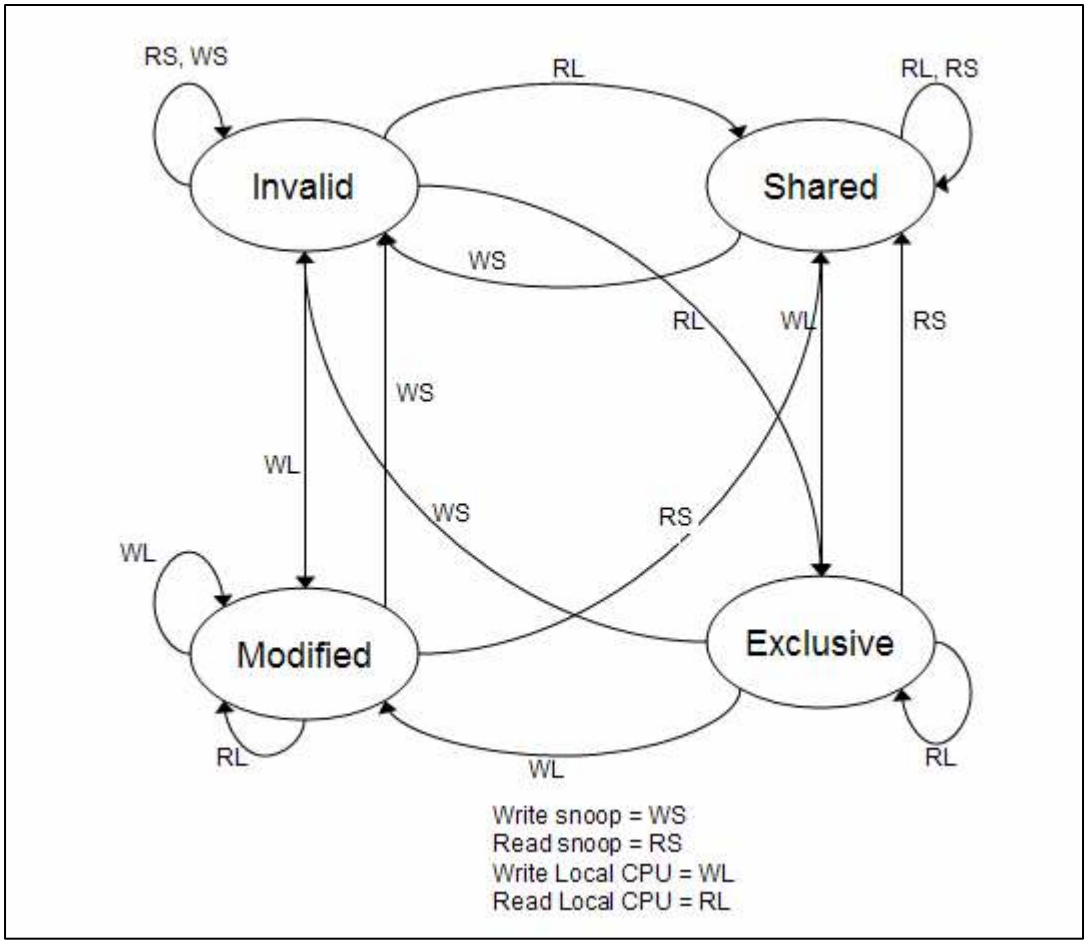

Figure 4: MESI State Transitions

The Blackford Northbridge makes use of MESI by implementing a central coherency engine located in the system's memory controller. The coherency manager sends snoops from one bus to another bus, and handles passing resulting state transitions to the requestor. The coherency manager contains a local staging buffer which holds cache lines as they move from source to destination through the northbridge.

The northbridge offers an optional snoop filter for workstations. This snoop filter is intended to reduce the number of snoops that are initiated by remote front-side buses or inbound I/O requests that target coherent system

memory. The organization of the snoop filter is shown in Figure 5. The snoop filter stores tags and coherency state information for all caches in the processor, using a directory scheme. The filter is organized as two front-side bus interleaves. Each interleave contains 8,192 sets, each organized as a 16-way set-associative array. This allows tracking of up to 2^{18} cache lines.

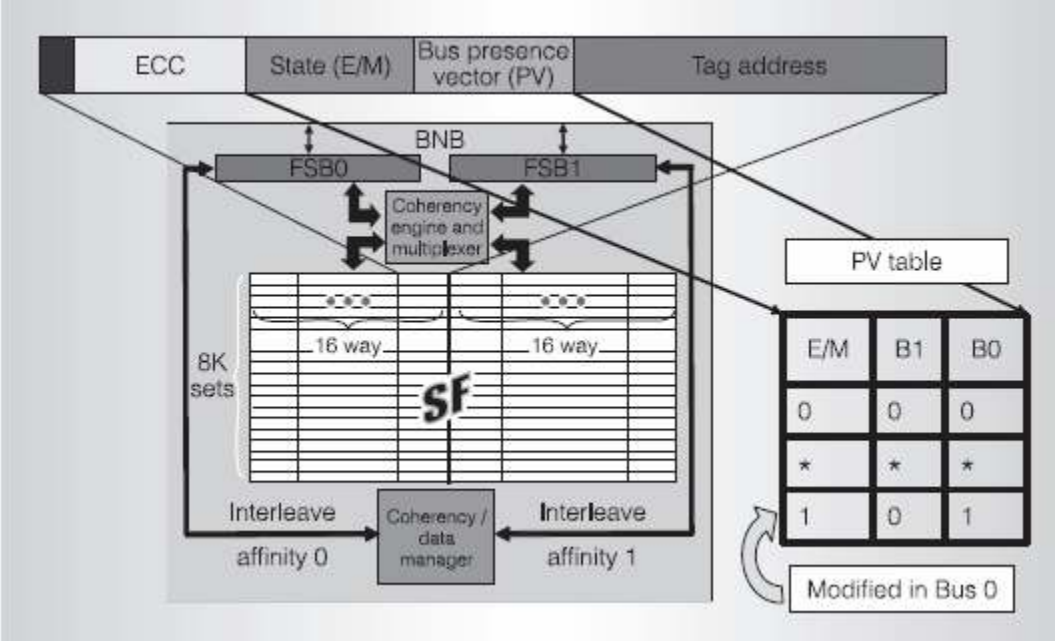

Figure 5: The Blackford Snoop Filter

When the coherency engine intercepts a request from the front-side bus, a snoop filter lookup is performed to determine a hit or miss. The state of an existing entry is then updated or a new entry is allocated. Depending on the search results of the snoop filter, the coherency engine will make a decision as to whether or not a cross-buss snoop will be launched.

5. Scalable Coherent Interface (SCI) and HyperTransport

With processing power increasing constantly, it has become very important to eliminate bottle-necks in the system. One of the most important areas where a bottle-neck can occur is in the processor's interface to system memory and other I/O devices. AMD based its decision to use the HyperTransport protocol on past interfaces, in particular the Scalable Coherent Interface.

The Scalable Coherent Interface started as an offshoot from the IEEE Standard Futurebus+ project.[8] The SCI group was searching for a new approach that would be bus-like in nature, yet would be able to avoid bottle-necks and be able to easily scale for multi-processor applications. The resulting interface included protocols that were simple, allowing processors to run fast. Also, since the protocols were simple, the interface could be built with a relatively low gate count, which kept expense down as well as successfully allowing the interface circuitry to have low latency, thus eliminating data transferring bottle-necks. Additionally, since the bus interface was rethought to be scalable and distance-independent, areas that make bus interface logic more expensive then necessary cleared up.

The SCI interface includes a hardware protocol to handle cache coherency. This protocol was based on a distributed-directory type cache coherency protocol. The protocol is multiple-reader, single-writer with write invalidation. There are 29 stable cache states and many pending states. Despite it's simplicity in hardware implementation, the SCI protocol was determined to be too complex to be implemented in the Opteron Northbridge due to the very large amount of cache states. However, SCI made it clear that there is a lot of value in using a similar data transfer topology. Figure 6 shows a top-level diagram of the SCI Cache Coherency Layer.

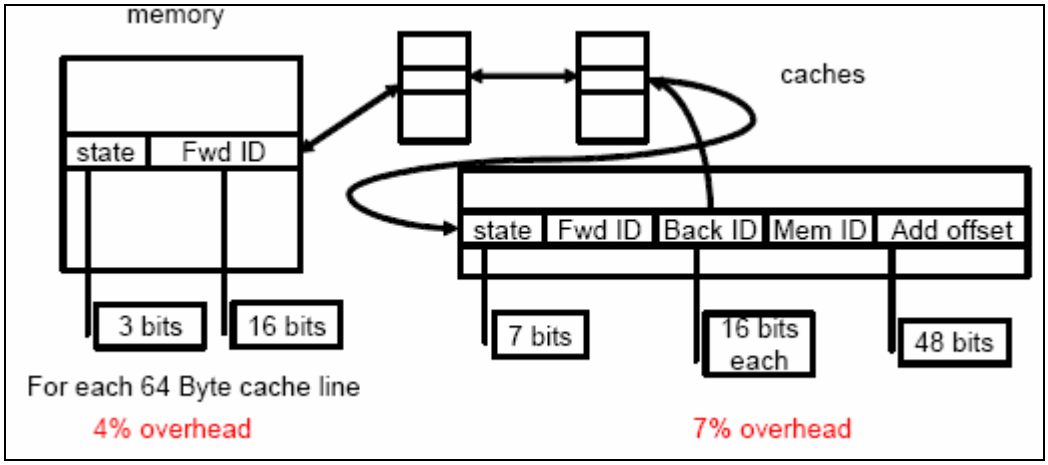

Figure 6: SCI Cache Coherency Layer

AMD chose to integrate HyperTransport based interfaces into the Opteron processor. HyperTransport was introduced in 2001. It is a bidirectional serial/parallel high-bandwidth, low-latency point-to-point link.[9] HyperTransport is available in three versions which run from 200 MHz to 2.6 GHz. For comparison, a PCI bus runs at either 33 or 66 MHz. HyperTransport also runs at a double data rate (DDR), which means that it sends data on both edges of the clock. This allows a maximum data rate of 5200 MegaTransfers/second when running at 2.6 GHz. The operating frequency of the HyperTransport link is auto-negotiated. HyperTransport also supports variable bit-width data paths; any width between 2-bits to 32-bits is possible. When operating at full speed with a full-width data bus, HyperTransport has a transfer rate of 41.6 GB/s, which makes it much faster then most other bus standards. The system is also very flexible, since links of various data widths can be mixed together into a single application. Wider data paths could be used for high-speed memory, while narrow data paths could be used for slow I/O devices.

HyperTransport can be used to generate system management messages, signaling interrupts, as well as to issue probes to adjacent processors. Additionally, it is compliant with the Advanced Configuration and Power Interface specification, which allows changes in processor sleep states to signal changes in device states. For example, the protocol can be used to power off a hard drive when the CPU goes to sleep.

6. Fully-Buffered DIMMS

A key technology that is central to the operation of the Intel Blackford Northbridge is an alternate memory architecture called Fully-Buffered dual in-line memory modules (FB-DIMMs).

Due to the continuing need for increased capacity on high-end servers, the need for larger amounts of memory has continued to present problems for system designers. While improvements in technology have allowed capacity needs to be met with increasingly dense DRAM chips, and bandwidth needs to be met by

scaling front-side bus data rates, the traditional bus organization has reached a point where it no longer scales well.

The electrical constraints of high-speed parallel buses complicate bus scaling in terms of loads, speed, and widths.[7] As a result, each generation of DRAM technology has allowed for fewer DIMMs per channel. The serpentine routing required for path-length matching becomes more challenging as bus widths increase. Currently, motherboard area used in routing just one channel is significant. This makes increasing capacity by adding channels very difficult. FB-DIMMs have been introduced to address these scalability issues in today's memory subsystems.

FB-DIMMs replace standard memory architectures that are typically used in servers. The traditional shared parallel interface between the memory controller and DRAM chips is replaced with a point-to-point serial interface between the memory controller and an intermediate buffer, called the Advanced Memory Buffer (AMB). The interface on each DIMM between the AMB and the DRAM chips is the same as what is currently used in DDR2 and DDR3 systems. The serial interface is divided into two uni-directional buses. One is for read traffic and status messages, called the northbound channel, and another is for write traffic and commands, called the southbound channel. Figure 7 shows this connectivity. FB-DIMMs use a packet-based protocol that bundles commands and data into frames that are transmitted on the channel and then converted to the DDR protocol by the AMB.[7] Frames can contain data and/or commands, including DRAM commands such as row activate, column read, and refresh. There are also special commands for a channel that include a write to configuration register and synchronization commands. The AMB acts like a pass-through switch, directly forwarding the requests it receives from the memory controller to successive DIMMs and forwarding frames from southerly DIMMs to northerly DIMMs or the memory controller. All frames are processed to determine whether the data and commands are for the local DIMM. Since frame scheduling is performed only by the memory controller, the AMB can only convert serial data to DDR based commands.

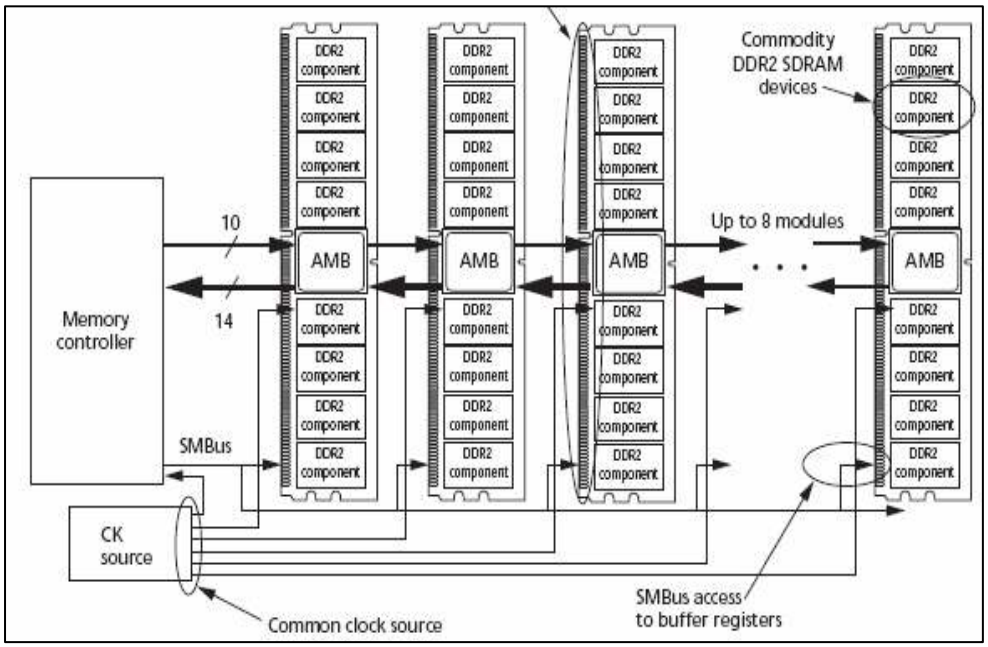

Figure 7: Fully-Buffered DIMMs

The FB-DIMM architecture was designed to allow further scaling of system memory, but serializing the memory interface between the memory controller and the actual DIMMs adds some performance considerations. By changing the data paths from a parallel shared bus to a serial point-to-point bus, there are some performance issues that become apparent. In particular, latency is increased in a system using FB-DIMMs if that system has low utilization. This is due to the added overhead of changing data to and from FB-DIMM's serial, framed protocol. When a system is at high utilization, the average latency of memory transactions is actually lower than an equivalent DDR based system. This is because the FB-DIMM protocol allows a large number of in-flight transactions concurrently. Additionally, since transactions to different DIMMs and transmit data to and from the memory controller can be scheduled simultaneously, FB-DIMM systems are able to offset the increased cost of serialization overhead. Finally, moving the parallel bus off the motherboard and onto the DIMM helps hide the overhead of bus turn-around time.[7]

7. The AMD Opteron Northbridge

The main goal of AMD's Opteron Northbridge architecture is to increase performance while operating within a fixed power budget. The AMD Opteron processor is built around multiple (currently one, two, or four) AMD64 based cores. A data router and memory controller is integrated on each chip with the cores, as well as a HyperTransport interconnect interface including three independent HyperTransport ports. The challenges that had to be overcome for this design effort included designing a system interconnect, memory hierarchy, and I/O that can easily scale with the number of cores and number of CPU sockets in a system.

AMD's Opteron Northbridge was introduced in 2005 with the release of the first dual-core Opteron processor. Included in this design is the AMD Direct Connect architecture, which is AMD's marketing term for its Opteron processor northbridge. The Direct Connect architecture was created to replace the more traditional external northbridge chipsets typically found external to a microprocessor. In the Opteron, the northbridge includes all of the logic that is on the CPU die, but is external to the processing cores. Figure 8 shows a comparison between a standard, traditional northbridge and the Opteron Direct Connect northbridge. Figure 8a shows a traditional northbridge. All data traffic into and out of the processing cores runs through the memory controller hub. This creates a bottle-neck for all traffic in the CPU and is also a single point of failure; if the memory controller hub stops working, so does the entire processor. The need for external memory buffers (XMBs) increases latency of memory reads and writes, as well as requiring additional power. Finally, a solution like this does not scale very well, as the logic inside the memory controller hub would become larger and therefore have a higher latency as more processors are added to the system.

Figure 8b shows the AMD Direct Connect northbridge. As shown, each processor has an integrated memory controller. This allows each processor to be able to directly interface to a bank of memory, eliminating the bottle-neck seen in

traditional northbridges. Additionally, each core has three independent HyperTransport based I/O interfaces. This allows each processor in a system to be interconnected in a point-to-point topology. This allows the system to scale easily by adding additional processors with HyperTransport interfaces. It also provides multiple paths from one processor to another, which eliminates the single point of failure seen in the previous northbridge. PCI bridges are installed as I/O devices using one of the HyperTransport interfaces on one of the processors. An additional reason for the high data throughput in this topology is that all of the integrated northbridge logic runs at the same frequency as the processing cores; in traditional northbridge designs, the northbridge circuitry runs at a lower multiple of the core frequency.

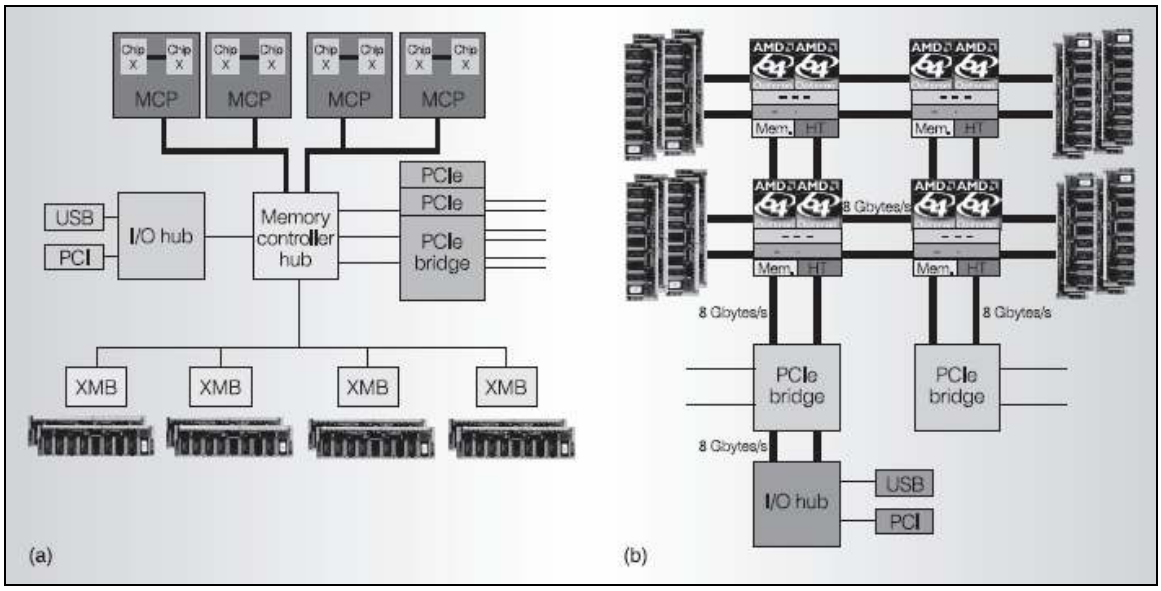

Figure 8: A comparison of (a) traditional northbridge and (b) the Opteron Direct Connect northbridge

A simplified view of the Direct Connect northbridge is shown in Figure 9. The northbridge consists of the System Request Interface (SRI) and Host Bridge, a crossbar, an on-chip memory controller, a DRAM controller, as well as three HyperTransport transceivers. The SRI contains the system address map, which is used to map memory ranges to specific processors. This map is essential to the system's ability to figure out which processor "owns" the memory that contains

the data it wishes to read or write to. If the memory to be accessed is determined to be in the on-chip memory (cache), the SRI sends the request to the on-chip memory controller. If the memory requested is off the chip, the SRI sends the request to one of the processor's HyperTransport ports. The DRAM controller will be used to interface to memory attached to the host processor, or the data will be brought local from one of the other processors via the other HyperTransport ports. The crossbar has five ports; one for the SRI, one for the memory controller, and one for each of the three HyperTransport ports. This allows very flexible system operation by allowing data and command requests to be easily routed between all possible pieces of circuitry in the northbridge.

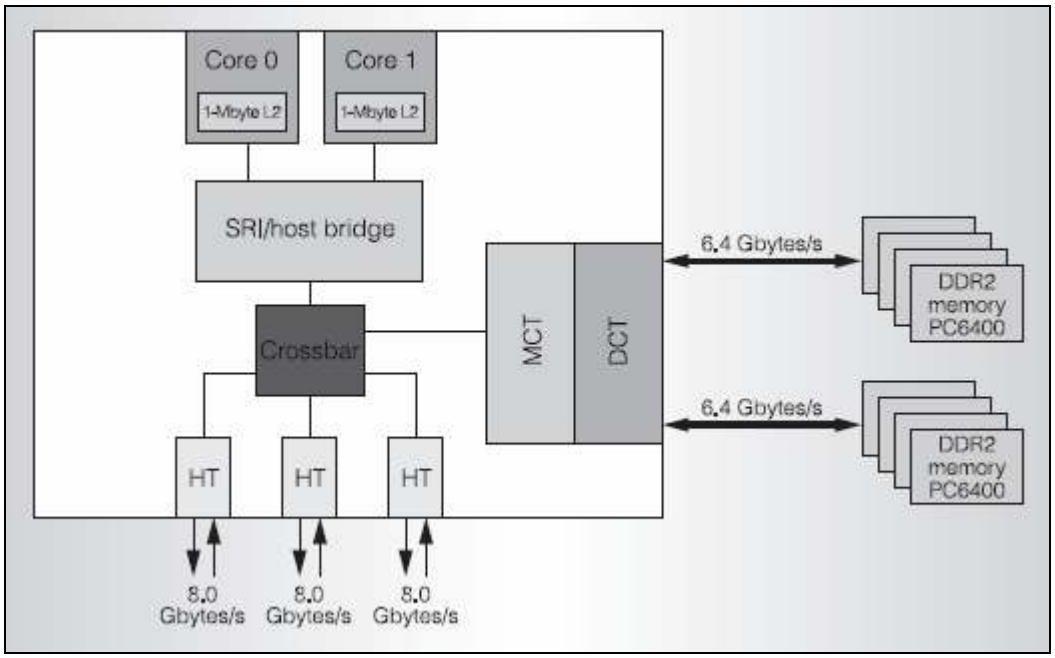

Figure 9: A Simplified View of the Opteron Northbridge

The northbridge separates processing of command and data packets between two different logical units. There are separate command and data crossbars in the northbridge. The command crossbar is dedicated to routing command packets and the data crossbar is dedicated to routing the data payload associated with commands, 4 to 64 bytes of data per command. Figure 10 shows the command flow for the northbridge. The command crossbar is responsible for routing

coherent HyperTransport commands. It can deliver one HyperTransport packet header per clock cycle. There is a pool of command sized buffers at each input port. These buffers are divided between four virtual channels: Request, Posted request, Probe, and Response. These command buffers are statically allocated at each of the crossbar inputs. The memory access buffers (MABs) store outstanding requests to memory. The address MAP stores the addresses of memory address windows in the system. The MAP allows each processor to be able to determine which other processor in the system contains the data requested (by physical memory address range). The graphics aperture resolution table (GART) maps memory requests from the graphics controllers in the system. The victim buffer is used to store cache lines that have been evicted from the processor while they wait for idle cycles from the memory controller so that they can be written back to main memory. This buffer is different from the standard write buffer because evicted cache lines in the victim buffer are not scheduled to be written to memory until the memory system becomes idle. Data in the write buffer therefore takes precedence over data in the victim buffer, and is written to main memory first.

The Request virtual channel is used to service memory reads, nonposted memory writes, and cache block commands. The Posted request virtual channel is used to service posted write commands. The difference between posted and nonposted writes is that posted writes do not require a response from a target, whereas nonposted writes do require a "target done" response from the receiver.[8] The Probe virtual channel is used to broadcast probes to all other processors in the system. These probes are used to "snoop" on other processors to determine whether or not there is more then one copy of the data, and to manage updating the data to ensure coherency. Finally, the Response virtual channel is used to service all responses resulting from requests and probes to other processors.

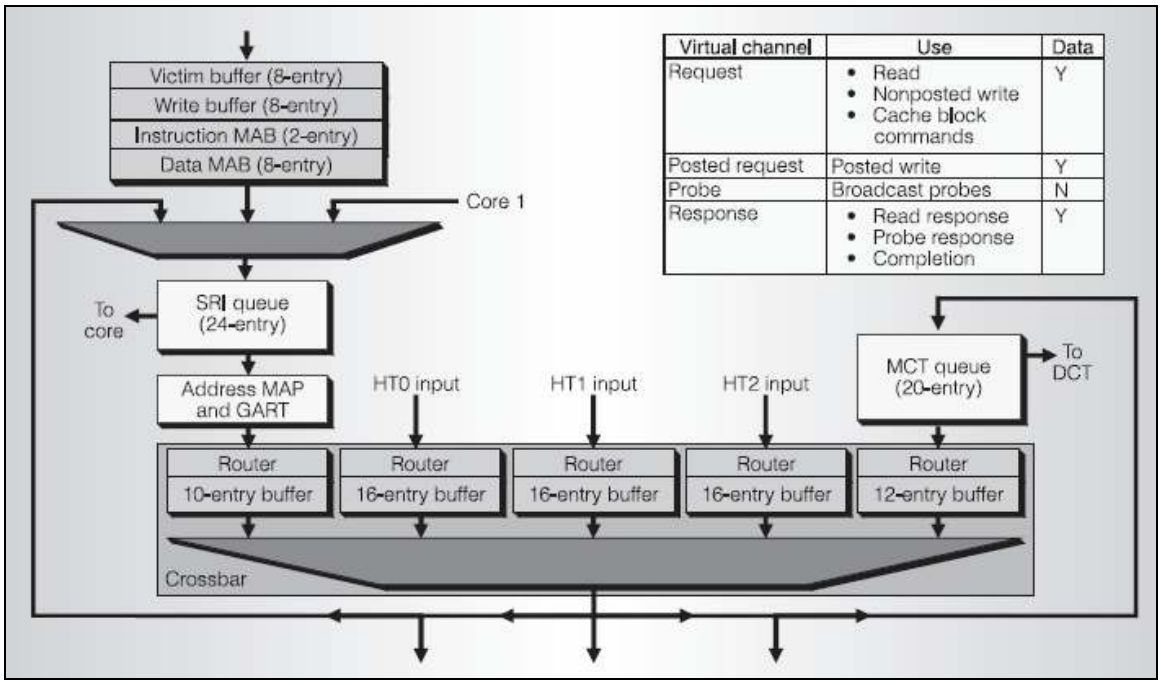

Figure 10: The Opteron Northbridge Command Flow

The Opteron Northbridge data crossbar is shown in Figure 11. The system cache line size is 64 bytes. To optimize the transfer of cache-line-size data packets, all buffers are sized in multiples of 64 bytes. Data packets move around inside the processor through data paths, each transfer taking 8 clock cycles. Transfers to different output ports are time multiplexed clock by clock to support high concurrency. Similar to other routing in the northbridge, HyperTransport routing is table driven. This supports arbitrary system topologies and the crossbar has separate routing tables for routing requests, probes, and responses. Responses are always point-to-point, and Probes are broadcast to all processors in the system.

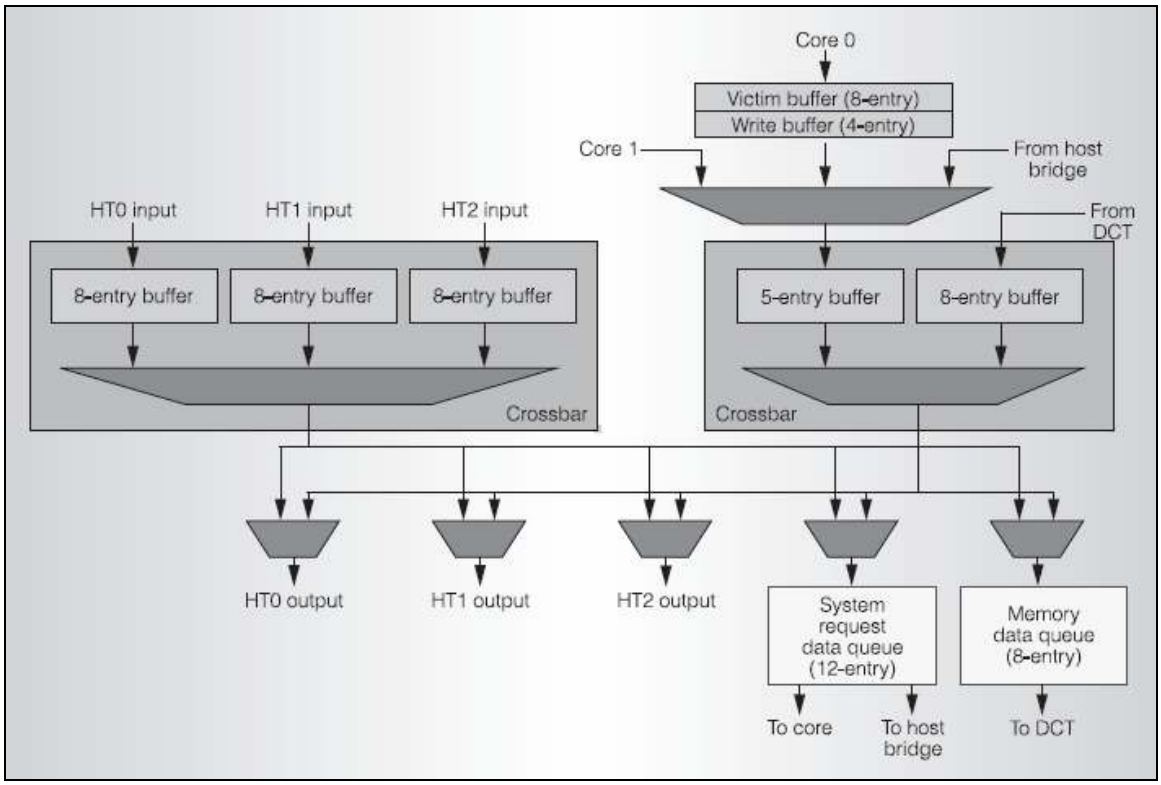

Figure 11: The Opteron Northbridge Data Flow

The Opteron was designed to have a higher performance memory interface than systems with an external memory controller. An important aspect of maintaining high performance when processors are added is the topology or the way the processors are connected together. Figure 12 shows the performance benchmarks of a number of different processor connection topologies. The most important factor in performance is the average network diameter. Network diameter is the number of "hops" that the average request has to make to get from one processor to another. Scaling is positive from one to eight processors. Unfortunately, processor performance decreases as the average network diameter increases. As seen in Figure 12, the average diameter for a two processor system is 0.5 hops. Moving to a four processor system connected in a square results in an average diameter of 1 hop. This would be reduced if the four processors could be fully connected, but since there are only three HyperTransport ports on each processor, this is not possible. There are two options for connecting eight

processors together, and both lead to different results. Connecting the processors in a ladder arraignment results in a network diameter of 1.8 hops, while network diameter is reduced to 1.5 hops by using the twisted ladder topology shown.

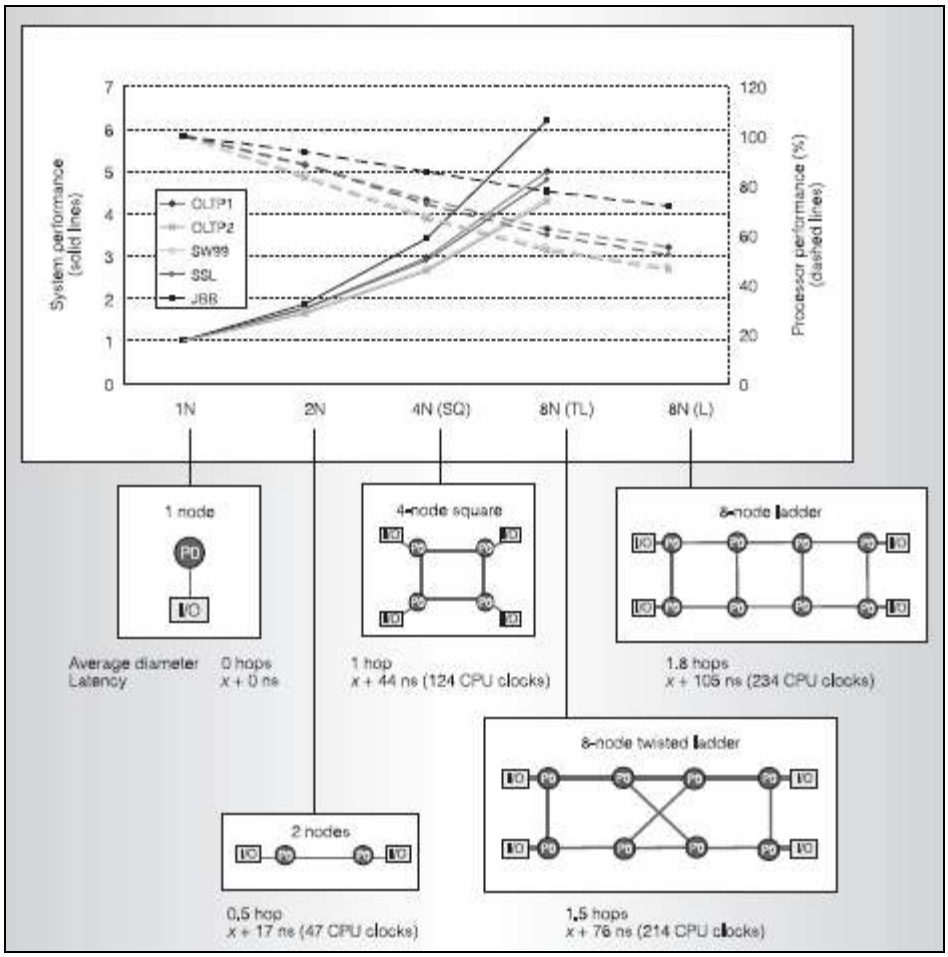

Figure 12: Performance of Various Connection Topologies

To further analyze performance of connection topologies, it is helpful to consider a metric known as Xfire memory bandwidth. Xfire memory bandwidth is defined as link-limited, all-to-all communication bandwidth (data only).[5] Figure 13 shows two 4-processor square topologies. Figure 13a shows the standard 4-processor square topology with an average diameter of 1 hop. This topology has an Xfire bandwidth of 14.9 Gbytes/s. If an additional HyperTransport port was added to each processor, then the 4-processor fully connected topology shown in Figure 13b would be possible. This topology would

have an average diameter of 0.75 hops and an Xfire bandwidth of 29.9 Gbytes/s. If the latest HyperTransport protocol (version 3.0) was used instead of version 2.0, the fully connected topology would achieve a bandwidth of 65.8 Gbytes/s.

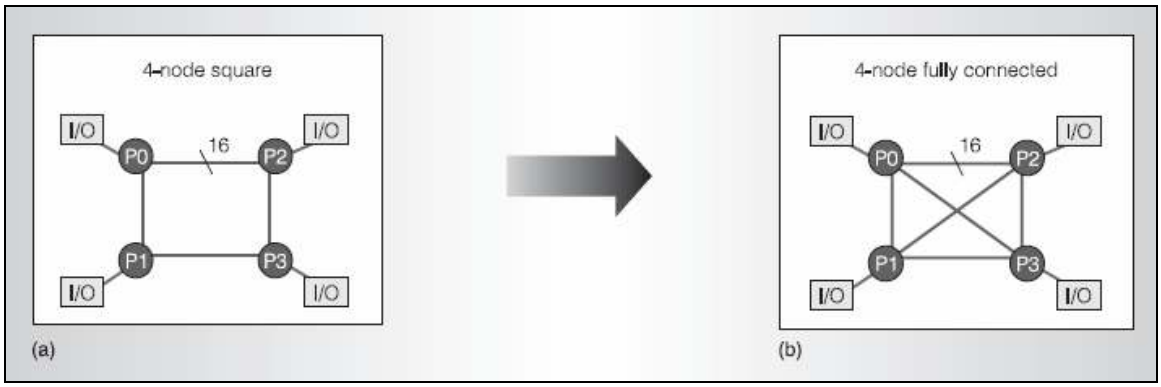

Figure 13: Four-processor Network Topologies

As shown in Figure 14, the benefits of fully connected topologies are more dramatic for eight-processor systems such as the ones shown in Figure 14. The Xfire bandwidth increases by a factor of 6 when a fully connected topology is used.

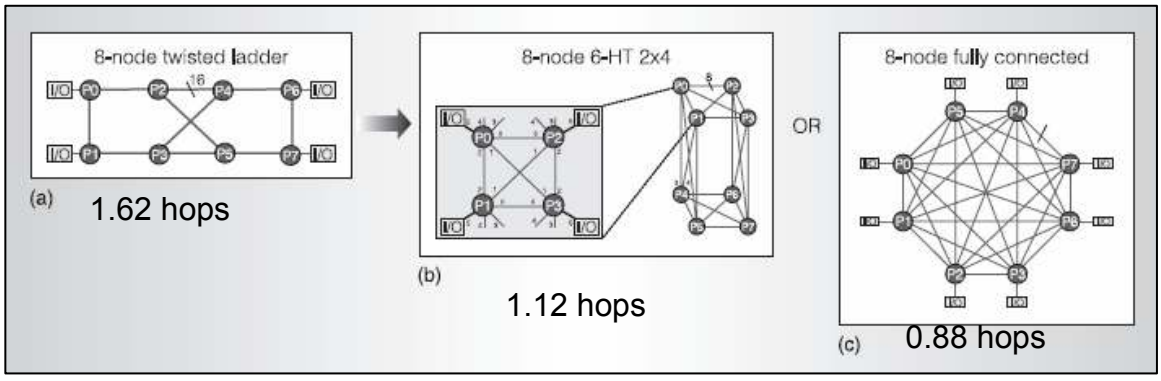

Figure 14: Eight-Processor Topologies

8. The Intel Blackford Northbridge Architecture

The Intel Blackford Northbridge Architecture is based on a traditional northbridge architecture. An overview of the architecture is shown in Figure 15. The system is made up of up to two Intel Xeon multi-core processors, a memory controller hub (MCH), an I/O hub, and a PCI hub. Some of the main features of this architecture include:

- Two independent front-side bus (FSB) interfaces
- A coherency engine
- Optional 16-Mbyte snoop filter (used to eliminate the number of cross-buss snoops)
- Four channel FB-DIMM memory controller
- Two I/O unit clusters, allowing connection to six x4 width Generation 1 PCI Express ports

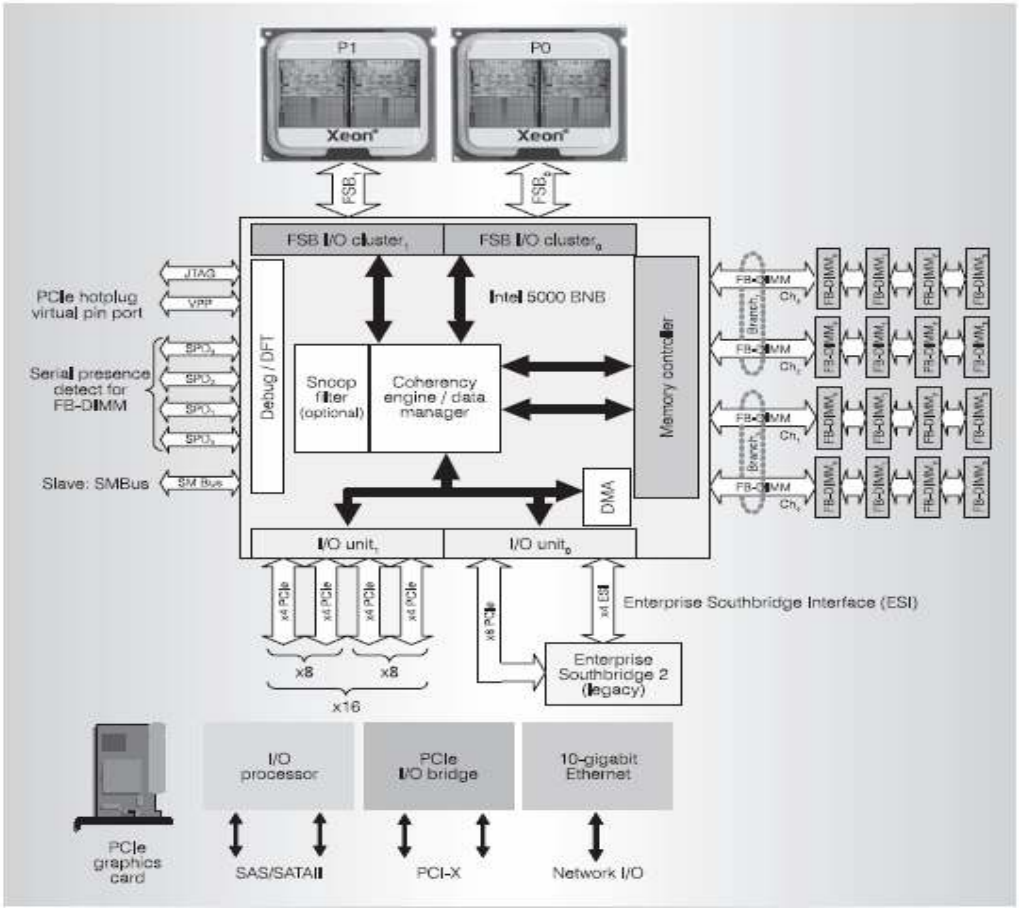

Figure 15: The Blackford Northbridge

The Blackford Northbridge is designed to make use of Intel's dual or quad core Xeon processors. These processors contain 4-Mbytes of L2 cache and 8-Mbytes of L2 cache, respectively and have a core clock frequency that can scale to 3 GHz. Multiple power saving features have been implemented on these processors including an enhanced halt state, clock gating, and intelligent use of on-die caches to prevent thrashing. The protocol used in the front-side bus allows the CPU and chipset to arbitrate the bus, drive transactions, initiate snoops, and provide responses to snoops and/or data.

The MCH chip is used to handle all traffic from the CPUs to the rest of the system. This controller has four FB-DIMM channels, and each channel can accommodate up to four DIMMs for a total of up to sixteen DIMMs in the entire system. As previously mentioned, the MCH contains the coherency engine and optional snoop filter, and all memory coherency is managed by the MCH. Two paired FB-DIMM channels form a lockstep branch, and there are two lockstep branches in the system, with channels 0 and 1 forming one branch, and channels 2 and 3 forming the other. A 64-byte cache line is split across a branch, and each channel supplies 32 bytes to and from the corresponding DIMM of the lockstep pair. Both branches operate independently of each other, and can be interleaved. Each channel has a peak data rate of 5.33 Gbytes/s (for DDR2 667) or 4.26 Gbytes/s (for DDR2 533).

The MCH provides the interface to memory from the CPUs and I/O. It has a built-in decoder to translate system addresses to a DDR2 device map, a protocol engine for DDR2 timing, an arbitration unit to service internal request queues, and an reliability, availability, and serviceability (RAS) unit for handling memory and channel errors.[10] The MCH also includes extensive reordering capabilities which help maximize channel throughput and efficiency. These capabilities include:

- Simultaneous read/write data transfer to different FB-DIMMS on a channel
- No turnarounds between back-to-back data transfers to different FB-DIMMs on a channel

- Minimal turnarounds and bubbles between back-to-back data transfers to the same FB-DIMM
- A posted column address strobe of the DDR to improve channel efficiency by removing scheduler conflicts[10]

The Blackford Northbridge provides many features that allow I/O connectivity between client devices or bridges to be scalable. This is accomplished through six x4 PCI Express (PCIe) ports. These ports are broken into two I/O unit clusters. Each port is deeply pipelined for maximizing read/write throughput. These ports can be combined to form larger widths such as x8 or x16. The northbridge provides peer-to-peer memory-mapped I/O, I/O read/write support for applications such as remote keyboard or video sharing, and I/O processor support for redundant-array-of-inexpensive-disks (RAID) applications. By combining read completetions when data already exists in the buffer, extra store-and-forward penalties of larger packet sizes are avoided. The result of this is to decrease first data latency, as well as improving bus efficiency on the PCIe links due to larger packets.

Blackford's Enterprise Southbridge Interface operates through a proprietary x4 PCIe link for connection and operation of legacy devices. It handles devices including PCI, PCI-X, USB, Low Pin Count (LPC), and Trusted Platform Module (TPM). Additionally, other peripheral components can be used in the system by attaching them to the PCIe backplane expansion ports. These components can include slot accounting system (SAS), small computer system interface (SCSI), serial advanced technology attachment (ATA) I and II controllers, or 10-Gbit Ethernet controllers.

The Blackford Northbridge includes I/O Acceleration Technology (I/OAT) to reduce processor utilization and maximize I/O throughput for networking applications. There are four main technologies that make up I/OAT: parallel processing of transmission control protocol (TCP) and memory functions, affinitized data flows, asynchronous low-cost copy, and improved TCP/IP processing. Parallel processing lowers the system overhead and increases efficiency of the TCP stack processing by using the abilities of the CPU to

execute multiple instructions per clock. By logically grouping data flows, the system partitions the network stack processing across multiple physical or logical CPUs. This allows CPU cycles to be allocated to the application for faster execution. Intel Quick Data Technology is used to allow payload data copies from the network interface card (NIC) buffer in system memory to the application buffer with fewer CPU cycles, reducing the overhead typically required to move data from the NIC to applications.[11] Finally, the system uses separate packet data and control paths to optimize processing of the packet header from the packet payload. This helps to reduce CPU cycles that are typically used to process the protocol.

Additionally, the Blackford Northbridge supports a four-channel direct memory access (DMA) engine. This engine has the ability to perform byte-aligned memory-to-memory and memory-to-MIMO transfers using a linked-list descriptor access mechanism. When used properly, the engine can move large amounts of data from dedicated memory to the application buffer with very low latency. This is also done in a way that PCIe and FSB bandwidth are not consumed. The measured DMA bandwidth is greater than 4.5 Gbytes/s for four channels and greater than 3.3 Gbytes/s for a single channel.[11]

An increasing performance concern for multi-gigabit Ethernet I/O cards and storage applications is efficient, high-bandwidth delivery of I/O data to the host. Direct cache access (DCA) is a protocol used in the Blackford Northbridge to improve I/O performance. This is accomplished by placing data from an end device directly into a CPU cache by directly interfacing with the FSB. Test results have shown throughput improvement for the Layer 3 forwarding IPv4 networking benchmark of 17 percent, as well as over 37 percent improvement for streaming exclusive-or (XOR) operations common in storage applications such as RAID.[11]

The Blackford Northbridge uses several additional performance optimizations to improve system throughput.

- **Speculative Memory Read**

A read request is sent to memory as soon as the processor request is decoded by the memory controller hub. A snoop filter lookup is launched at the same time, but the read request is sent without waiting for the results of the snoop filter lookup. A read cancel can be issued to the memory controller if a snoop results in "dirty" data or a retry.

- **Early Defer Reply**

This allows a memory read defer-reply transaction on the FSB to overlap the bus arbitration phase before the arrival of data from memory. When read data gets to the FSB cluster, it is immediately sent to the FSB with no additional latency.

- **Snoops**

When there is no snoop filter present in the system (as is the case with server chipsets), the Blackford Northbridge eliminates cross-buss snoops for instruction fetch or data read transactions that are in the modified or shared state.

- **Arbitration**

The Blackford Northbridge uses a three-tier, weighted, round-robin arbitration algorithm to balance performance between data replies and snoops on the FSB. The weights in the algorithm can be adjusted to suit the expected traffic mix on the system.

- **Memory Interleaving**

The memory controller can be programmed to scatter sequential addresses with fine-grained interleaving across memory branches, ranks, and DRAM banks. This is used to minimize the number of bank conflicts and reduces the application's power consumption.

- **Partial Writes**

This optimization allows applications such as graphics adapters and video rendition programs that rely on partial writes to system memory to function in a more optimal manner. By coalescing the partial writes and reducing the possibility of conflict serialization in multi-bus systems, increased performance and memory channel utilization are achieved.

This optimization is implemented through logic in the coherency engine and memory controller.[10]

9. Performance and Power Consumption

There is a lot of data available to compare the performance of both of these competing architectures. Both Intel and AMD offer benchmark test results show that each side performs better than the other. There are some trends in the data that help point toward the fact that both systems have strong and weak points. There are many benchmarks available that claim each processor performs better than the other, but the reality is that each serves a different purpose.

The Standard Performance Evaluation Corporation (SPEC) is a non-profit organization that develops and maintains a standardized set of benchmarks that can be used to evaluate computers. These benchmarks are frequently used by many companies to compare computer systems, and as a result, to compare CPU performance. There are many systems that have had SPEC results submitted using various configurations of the Opteron CPU and the Xeon CPU. It is very difficult to find SPEC results that allow a true "apples to apples" comparison. Furthermore, it is sometimes questionable as to whether or not the SPEC benchmark tests accurately simulate real world situations. Given all of these factors, it is still useful to look at SPEC results to help determine which CPU has better performance. For most SPEC benchmarks that target pure processing power, the Xeon based systems out-performed equivalent Opteron based systems.

There is a SPEC benchmark aimed at evaluating performance and power of volume sever class computers. This benchmark does a questionable job of simulating real world results. One of the key short-comings of this benchmark is that it uses a small database to test queries on. Additionally, all data tables are in memory, so disk I/O is not considered. Finally, the scale of the simulated user-base is very small; a maximum of four simultaneous users and only one client computer. To address these short-comings of the SPECpower benchmark, a different efficiency test has been developed. This test uses a large database,

includes disk I/O, and has a more realistic amount of simultaneous users (500) and a more realistic amount of client computers (32). Table 2 below shows the other differences between the SPECpower benchmark and this newer efficiency test (called the Neal Nelson Power Efficiency Test).

Characteristic	Nelson Power Efficiency Test	SPECpower Benchmark
Application Software	Apache2, "C" programs	Proprietary Java Application
Application Memory Footprint	Large, Complex	Small, Simple
Test Database Size	Larger (Approx. 140 GB)	Smaller
Location of Data Tables	Disk	Memory
Disk Input/Output During Test	Yes	No
Database Management	MySQL, Oracle, Sybase	Undocumented
Access to Data Tables	Structured Query Language	Undocumented
Record Locking	Yes (50% of all transactions)	No
No. Trans. Per screen/batch	1	1,000
Max. Simultaneous Users	500	1, 2, or 4
No. of Client Computers	32	1
Network Traffic During Test	Complex	Simple
Operating System	Suse Linux Ent. Server	Varies
Operating System Tunables	Always identical	Varies

Table 2: Comparison of Power Efficiency Benchmarks

The most interesting test data available compares a two CPU Intel Xeon Quad-core system with a two CPU AMD Opteron Quad-core system. The systems were compared with 4 GB of memory, 8 GB of memory, and 16 GB of memory. The Intel system outperformed the AMD system in all throughput tests by an average of 2.48% in the 4 GB configuration, an average of 2.79% in the 8 GB configuration, and 2.59% in the 16 GB configuration. These results match up with the SPEC results for throughput, where the Intel based systems almost always outperformed equivalent AMD based systems. It is important to note that

at the heaviest utilizations, in all configurations the Xeon based system outperformed the Opteron based system by less than 1%. It is also important to note that while these results are on very similar systems, the Xeon based system had a higher clock frequency (2.33GHz vs. 2.0GHz) than the Opteron based system.[12]

While the Xeon based systems perform best in throughput, the Opteron based systems have a clear edge in power consumption. As more memory was added to each system, the advantage in these results was increased to the Opteron based system. In the 16 GB configuration, the Opteron based system ran at 21.3% (71 fewer watts) less power at full loading, and at 41.2% (121.6 fewer watts) less power when idle.[12] The power savings when the systems are idle is particularly important as studies indicate that many servers are powered on and idle nearly 80% of the time.[12] The Xeon processors are generally efficient, but the additional need of a separate memory controller adds to power consumption (up to 47W) and lowers efficiency. Additionally, since the Xeon based systems use FB-DIMMs, there is an additional amount of power consumed for each FB-DIMM added to the system (about 4W per DIMM). This is due to the addition of the Advanced Memory Buffer on each FB-DIMM that is not present on standard DDR2 DIMMs. This explains why the power usage of both systems is closer when less memory is used. Finally, since the Xeon processors are being produced with a 45nm process instead of the 65nm process still being used for the Opteron, there is more leakage current, which is an additional explanation of the addition power consumption of the Xeon processor.

10. Additional Considerations

There are some additional important things to note about both of these northbridge architectures. While digging into all of the available information about both of these architectures, it is interesting to question why most Intel configurations boast higher performance specs than equivalent Opteron configurations. Both systems have similar memory bandwidth, and while

Opteron has a clear edge in power consumption, it could be expected that throughput performance would be closer to that of the Blackford based systems. One likely reason for the performance disparity is cache size. This is one of the trade-offs made by the Opteron's designers. While Blackford moves all of the logic relating to memory management to a separate chip, the Opteron has much more complexity on its die. This is the likely reason that Opteron processors have only 2MB of L2 cache to share (512KB each) with four cores, while the Xeon based processors share 8MB of cache dynamically between four cores.

When it comes to power saving and improved efficiency from previous generations of processor, there is a great difference in how Intel and AMD are approaching their solutions. Most of the gains in efficiency from the Xeon 7100 to the Xeon 7300 series come from a process improvement that allows for smaller transistors. This is of interest because the current competing generation of Opteron Quad-Core CPUs is already far more power efficient, yet has not made the move to the same smaller transistors. Opteron accomplishes this by segmenting each core into sections that can be powered on and off as needed (such as the floating-point unit). Will there be more gains in efficiency and will AMD be able to close the throughput performance gap by switching to smaller transistors in the future?

Next, it is important to consider system scalability when evaluating both of these architectures. As systems continue to grow, performance will continue to improve as the ability to add more CPUs to a system continues to grow. The Blackford Northbridge (and all other current Intel Xeon Northbridges) currently allow for only a maximum of four CPUs in one system. Any additional CPUs past four would require the central memory controller hub to grow in complexity to accommodate additional Front-side bus links to the extra CPUs. It is worth wondering how possible it will be to continue adding support form additional CPUs in this configuration, and it certainly seems that having a central memory controller would make scaling more difficult. The current Opteron Northbridge also accommodates up to four processors, and work is being done to allow that to scale to eight and beyond. By avoiding the need for a central memory controller,

it seems that the Opteron Northbridge should be able to continue scaling more easily than the Blackford Northbridge.

The last point to consider is the future feasibility of each of these architectures. The Blackford Northbridge is very dependent on the performance gains it gets from the FB-DIMM architecture. There is some question in the industry as to whether or not memory manufacturers are going to continue supporting FB-DIMMs. The industry is questioning whether or not the trade-off of additional power consumed (each FB-DIMM uses around 4W more than an equivalent DDR2 DIMM) is worth the gain in performance achieved.

11. Conclusion

When comparing two very different architectures, it is important to be aware of the strengths and weaknesses of both, and how those apply to the intended use of a system based on either architecture. The current generation of Intel Xeon processors based on the Blackford Northbridge boast better over-all through-put performance (although this does drop off with increasing system utilization), but has a substantial weakness efficiency and power usage. This is due to Blackford's reliance on a separate memory controller, as well as the additional power requirements of the FB-DIMMs used as the system memory. The current Opteron based systems traded off cache size in favor of including an integrated memory controller on the CPU die. This has resulted in lower throughput performance than the competing Blackford architecture, but has resulted in very large gains in power performance, and in particular a much higher performance-per-watt.

The strengths of the current Blackford system seem to lend itself to being used in situations where computational performance is the only factor to be considered, such as in graphical rendering, or for other scientific purposes where "super-computers" would typically be used. The power efficiency performance of the Opteron seems to lend itself to being used in very high utilization servers, and in main-stream IT configurations where economical factors (cost to maintain the system for example) would be of great importance. It will be very interesting

to see if future versions of each of these architectures can successfully blend the raw computing power of the Blackford based systems with the power efficiency of the Opteron based systems.

The battle between AMD and Intel in the high-end server market-place will continue getting more interesting. Intel is planning to introduce a point-to-point processor interconnect known as QuickPath. It is being created to compete directly with the HyperTransport interconnect system AMD is using in Opteron. Beginning with the release of the Nehalem processor, Intel will have integrated memory controllers onto their CPUs, much like Opteron, and will have eliminated the separate memory controller hub currently in use.[13] Additionally, AMD is working to move the next Opteron processors to a 45nm process. Doing so will allow AMD to add much more L2 cache to their processors, but will increase their leakage current, and potentially increase power consumption. It will be very interesting to see if the Opteron can catch up to the Xeon in throughput, and if the Xeon can improve its power consumption to better compete with Opteron.

References

[1] *x86-64,* http://en.wikipedia.org/wiki/X86-64.
[2] S.V. Adve and K. Gharachorioo, "Shared Memory Consistency Models: A Tutorial," *Computer*, vol. 29, no. 12, Dec. 1996, pp. 66-76.
[3] *MOESI Protocol,* http://en.wikipedia.org/wiki/MOESI_protocol.
[4] *AMD64 Architecture Programmer's Manual Volume 2: System Programming*, Advanced Micro Devices, Revision 3.14, Publication No. 24593, September 2007.
[5] P. Conway and B. Hughes, "The AMD Opteron Notrhbridge Architecture," *IEEE Micro*, March-April 2007, pp. 10-21.
[6] Y. Sheffer, *Advanced Cache Topics,* http://www.ee.technion.ac.il/courses/044800/lectures/MESI.pdf.
[7] B. Ganesh, A. Jaleel, D. Wang, and B. Jacob, "Fully-Buffered DIMM Memory Architectures: Understanding Mechanisms, Overheads and Scaling."
[8] *What is the Scalable Coherent Interface?,* http://www.scizzl.com/WhatIsSCI.html.
[9] *HyperTransport*, http://en.wikipedia.org/wiki/Hyper_transport.
[10] S. Radhakrishnan, S. Chinthamani, and K. Cheng, "The Blackford Northbridge Chipset for the Intel 5000," *IEEE Micro,* March-April 2007, pp. 22-33.
[11] *Accelerating High-Speed Networking with Intel I/O Acceleration Technology*, http://www.intel.com/technology/ioacceleration/306517.pdf.
[12] N. Nelson & Associates, "Throughput and Power Efficiency for AMD and Intel Quad Core Processors," http://www.worlds-fastest.com/wfz991.html.
[13] *Intel QuickPath Interconnect*, http://en.wikipedia.org/wiki/Quickpath.