The SGI Origin: A ccNUMA Highly Scalable Server

James Laudon and Daniel Lenoski
Silicon Graphics, Inc.
2011 North Shoreline Boulevard
Mountain View, California 94043
laudon@sgi.com lenoski@sgi.com

Abstract

The SGI Origin 2000 is a cache-coherent non-uniform memory access (ccNUMA) multiprocessor designed and manufactured by Silicon Graphics, Inc. The Origin system was designed from the ground up as a multiprocessor capable of scaling to both small and large processor counts without any bandwidth, latency, or cost cliffs.The Origin system consists of up to 512 nodes interconnected by a scalable Craylink network. Each node consists of one or two R10000 processors, up to 4 GB of coherent memory, and a connection to a portion of the XIO IO subsystem. This paper discusses the motivation for building the Origin 2000 and then describes its architecture and implementation. In addition, performance results are presented for the NAS Parallel Benchmarks V2.2 and the SPLASH2 applications. Finally, the Origin system is compared to other contemporary commercial ccNUMA systems.

1 Background

Silicon Graphics has offered multiple generations of symmetric multiprocessor (SMP) systems based on the MIPS microprocessors. From the 8 processor R3000-based Power Series to the 36 processor R4000-based Challenge and R10000-based Power Challenge systems, the cache-coherent, globally addressable memory architecture of these SMP systems has provided a convenient programming environment for large parallel applications while at the same time providing for efficient execution of both parallel and throughput based workloads.

The follow-on system to the Power Challenge needed to meet three important goals. First, it needed to scale beyond the 36 processor limit of the Power Challenge and provide an infrastructure that supports higher performance per processor. Given the factor of four processor count increase between the Power Series and Power Challenge lines, it was desired to have the next system support at least another factor of four in maximum processor count. Second, the new system had to retain the cache-coherent globally addressable memory model of the Power Challenge. This model is critical for achieving high performance on loop-level parallelized code and for supporting the existing Power Challenge users. Finally, the entry level and incremental cost of the system was desired to be lower than that of a high-performance SMP, with the cost ideally approaching that of a cluster of workstations.

Simply building a larger and faster snoopy bus-based SMP system could not meet all three of these goals. The second goal might be achievable, but it would surely compromise performance for larger processor counts and costs for smaller configurations.

Therefore a very different architecture was chosen for use in the next generation Origin system. The Origin employs distributed shared memory (DSM), with cache coherence maintained via a directory-based protocol. A DSM system has the potential for meeting all three goals: scalability, ease of programming, and cost. The directory-based coherence removes the broadcast bottleneck that prevents scalability of the snoopy bus-based coherence. The globally addressable memory model is retained, although memory access times are no longer uniform. However, as will be shown in this paper, Origin was designed to minimize the latency difference between remote and local memory and to include hardware and software support to insure that most memory references are local. Finally, a low initial and incremental cost can be provided if the natural modularity of a DSM system is exploited at a relatively fine granularity by the product design.

In the following section of this paper, the scalable shared-memory multiprocessing (S^2MP) architecture of the Origin is presented. Section 3 details the implementation of the Origin 2000. Performance of the Origin 2000 is presented in Section 4. Section 5 compares the Origin system with other contemporary ccNUMA systems. Finally, Section 6 concludes the paper.

ISCA '97 Denver, CO, USA

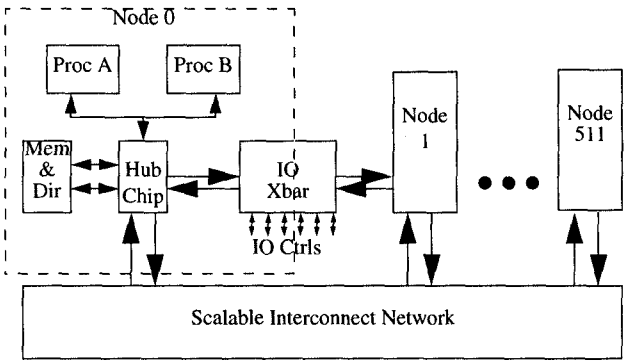

Figure 1 Origin block diagram

2 The Origin S^2MP Architecture

A block diagram of the Origin architecture is shown in Figure 1. The basic building block of the Origin system is the dual-processor node. In addition to the processors, a node contains up to 4 GB of main memory and its corresponding directory memory, and has a connection to a portion of the IO subsystem.

The architecture supports up to 512 nodes, for a maximum configuration of 1024 processors and 1 TB of main memory. The nodes can be connected together via any scalable interconnection network. The cache coherence protocol employed by the Origin system does not require in-order delivery of point-to-point messages to allow the maximum flexibility in implementing the interconnect network.

The DSM architecture provides global addressability of all memory, and in addition, the IO subsystem is also globally addressable. Physical IO operations (PIOs) can be directed from any processor

to any IO device. IO devices can DMA to and from all memory in the system, not just their local memory.

While the two processors share the same bus connected to the Hub, they do not function as a snoopy cluster. Instead they operate as two separate processors multiplexed over the single physical bus (done to save Hub pins). This is different from many other ccNUMA systems, where the node is a SMP cluster. Origin does not employ a SMP cluster in order to reduce both the local and remote memory latency, and to increase remote memory bandwidth. Local memory latency is reduced because the bus can be run at a much higher frequency when it needs to support only one or two processor than when it must support large numbers of processors. Remote memory latency is also reduced by a higher frequency bus, and in addition because a request made in a snoopy bus cluster must generally wait for the result of the snoop before being forwarded to the remote node[7]. Remote bandwidth can be lower in a system with a SMP cluster node if memory data is sent across the remote data bus before being sent to the network, as is commonly done in DSM systems with SMP-based nodes[7][8]. For remote requests, the data will traverse the data bus at both the remote node and at the local node of the requestor, leading to the remote bandwidth being one-half the local bandwidth. One of the major goals for the Origin system was to keep both absolute memory latency and the ratio of remote to local latency as low as possible and to provide remote memory bandwidth equal to local memory bandwidth in order to provide an easy migration path for existing SMP software. As we will show in the paper, the Origin system does accomplish both goals, whereas in Section 6 we see that all the snoopy-bus clustered ccNUMA systems do not achieve all of these goals.

In addition to keeping the ratio of remote memory to local memory latency low, Origin also includes architectural features to address the NUMA aspects of the machine. First, a combination of hardware and software features are provided for effective page migration and replication. Page migration and replication is important as it reduces effective memory latency by satisfying a greater percentage of accesses locally. To support page migration Origin provides per-page hardware memory reference counters, contains a block copy engine that is able to copy data at near peak memory speeds, and has mechanisms for reducing the cost of TLB updates.

Other performance features of the architecture include a high-performance local and global interconnect design, coherence protocol features to minimize latency and bandwidth per access, and a rich set of synchronization primitives. The intra-node interconnect consists of single Hub chip that implements a full four-way crossbar between processors, local memory, and the I/O and network interfaces. The global interconnect is based on a six-ported router chip configured in a multi-level fat-hypercube topology.

The coherence protocol supports a clean-exclusive state to minimize latency on read-modify-write operations. Further, it allows cache dropping of clean-exclusive or shared data without notifying the directory in order to minimize the impact on memory/directory bandwidth caused by directory coherence. The architecture also supports request forwarding to reduce the latency of interprocessor communication.

For effective synchronization in large systems, the Origin system provides fetch-and-op primitives on memory in addition to the standard MIPS load-linked/store-conditional (LL/SC) instructions. These operations greatly reduce the serialization for highly contended locks and barrier operations.

Origin includes many features to enhance reliability and availability. All external cache SRAM and main memory and directory DRAM are protected by a SECDED ECC code. Furthermore, all high-speed router and I/O links are protected by a full CRC code and a hardware link-level protocol that detects and automatically retries faulty packets. Origin's modular design provides the overall

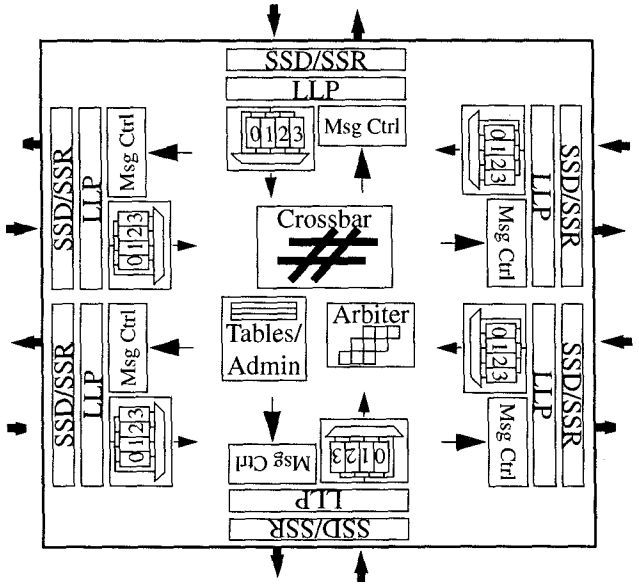

Figure 2 SPIDER ASIC block diagram

basis for a highly available hardware architecture. The flexible routing network supports multiple paths between nodes, partial population of the interconnect, and the hot plugging of cabled-links that permits the bypass, service, and reintegration of faulty hardware.

To address software availability in large systems, Origin provides access protection rights on both memory and IO devices. These access protection rights prevent unauthorized nodes from being able to modify memory or IO and allows an operating system to be structured into cells or partitions with containment of most failures to within a given partition[10][12].

3 The Origin Implementation

While existence proofs for the DSM architecture have been available in the academic community for some time[1][6], the key to commercial success of this architecture will be an aggressive implementation that provides for a truly scalable system with low memory latency and no unexpected bandwidth bottlenecks. In this section we explore how the Origin 2000 implementation meets this goal. We start by exploring the global interconnect of the system. We then present an overview of the cache coherence protocol, followed with a discussion of the node design. The IO subsystem is explored next, and then the various subsystems are tied together with the presentation of the product design. Finally, this section ends with a discussion of interesting performance features of the Origin system.

3.1 Network Topology

The interconnect employed in the Origin 2000 system is based on the SGI SPIDER router chip[4]. A block diagram of this chip is shown in Figure 2. The main features of the SPIDER chip are:

- six pairs of unidirectional links per router
- low latency (41 ns pin-to-pin) wormhole routing
- DAMQ buffer structures[4] with global arbitration to maximize utilization under load.
- four virtual channels per physical channel
- congestion control allowing messages to adaptively switch between two virtual channels

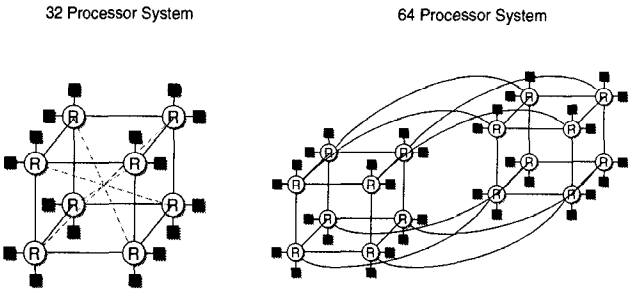

Figure 3 32P and 64P Bristled Hypercubes

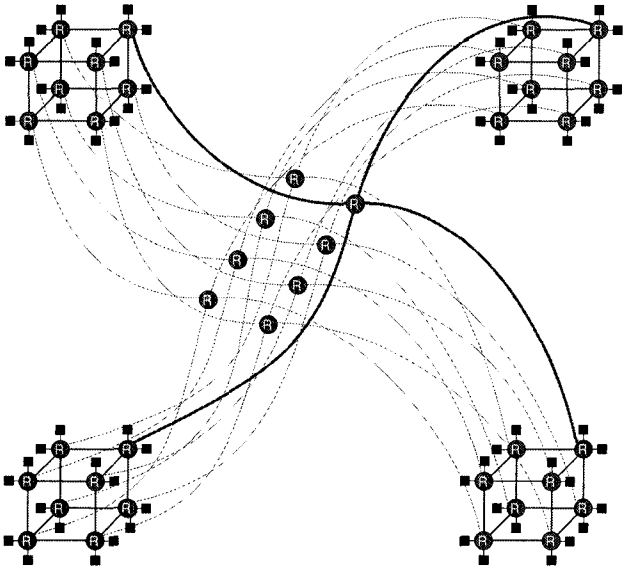

Figure 4 128P Heirarchical Fat Bristled Hypercube

- support for 256 levels of message priority with increased priority via packet aging
- CRC checking on each packet with retransmission on error via a go-back-n sliding window protocol
- software programmable routing tables

The Origin 2000 employs SPIDER routers to create a bristled fat hypercube interconnect topology. The network topology is bristled in that two nodes are connected to a single router instead of one. The fat hypercube comes into play for systems beyond 32 nodes (64 processors). For up to 32 nodes, the routers connect in a bristled hypercube as shown in Figure 3. The SPIDER routers are labeled using R, the nodes are the block boxes connecting to the routers. In the 32 processor configuration, the otherwise unused SPIDER ports are shown as dotted lines being used for Express Links which connect the corners of the cube, thereby reducing latency and increasing bisection bandwidth.

Beyond 64 processors, a hierarchical fat hypercube is employed. Figure 4 shows the topology of a 128 processor Origin system. The vertices of four 32-processor hypercubes are connected to eight *meta*-routers. To scale up to 1024 processors, each of the single meta-routers in the 128 processor system is replaced with a 5-D hypercubes.

3.2 Cache Coherence Protocol

The cache coherence protocol employed in Origin is similar to the Stanford DASH protocol[6], but has several significant performance improvements. Like the DASH protocol, the Origin cache coherence protocol is non-blocking. Memory can satisfy any incoming request immediately; it never buffers requests while waiting for another message to arrive. The Origin protocol also employs the request forwarding of the DASH protocol for three party transactions. Request forwarding reduces the latency of requests which target a cache line that is owned by another processor.

The Origin coherence protocol has several enhancements over the DASH protocol. First, the Clean-exclusive (CEX) processor cache state (also known as the exclusive state in MESI) is fully supported by the Origin protocol. This state allows for efficient execution of read-modify-write accesses since there is only a single fetch of the cache line from memory. The protocol also permits the processor to replace a CEX cache line without notifying the directory. The Origin protocol is able to detect a rerequest by a processor that had replaced a CEX cache line and immediately satisfy that request from memory. Support of CEX state in this manner is very important for single process performance as much of the gains from the CEX state would be lost if directory bandwidth was needed each time a processor replaced a CEX line. By adding protocol complexity to allow for the "silent" CEX replacement, all of the advantages of the CEX state are realized.

The second enhancement of the Origin protocol over DASH is full support of upgrade requests which move a line from a shared to exclusive state without the bandwidth and latency overhead of transferring the memory data.

For handling incoming I/O DMA data, Origin employs a write-invalidate transaction that uses only a single memory write as opposed to the processor's normal write-allocate plus writeback. This transaction is fully cache coherent (i.e., any cache invalidations/interventions required by the directory are sent), and increases I/O DMA bandwidth by as much as a factor of two.

Origin's protocol is fully insensitive to network ordering. Messages are allowed to bypass each other in the network and the protocol detects and resolves all of these out-of-order message deliveries. This allows Origin to employ adaptive routing in its network to deal with network congestion.

The Origin protocol uses a more sophisticated network deadlock avoidance scheme than DASH. As in DASH, two separate networks are provided for requests and replies (implemented in Origin via different virtual channels). The Origin protocol does have requests which generate additional requests (these additional requests are referred to as *interventions* or *invalidations*). This request-to-request dependency could lead to deadlock in the request network. In DASH, this deadlock was broken by detecting a potential deadlock situation and sending negative-acknowledgments (NAKs) to all requests which needed to generate additional requests to be serviced until the potential deadlock situation was resolved. In Origin, rather than sending NAKs in such a situation, a *backoff* intervention or invalidate is sent to the requestor on the reply network. The backoff message contains either the target of the intervention or the list of sharers to invalidate, and is used to signal the requestor that the memory was unable to generate the intervention or invalidation directly and therefore the requestor must generate that message instead. The requestor can always sink the backoff reply, which causes the requestor to then queue up the intervention or invalidate for injection into the request network as soon as the request network allows. The backoff intervention or invalidate changes the request-intervention-reply chain to two request-reply chains (one chain being the request-backoff message, one being the intervention-reply chain), with the two networks preventing deadlock on these two request-reply chains. The ability to generate backoff interventions and invalidations allows for better forward progress in the face of very heavily loaded systems since the deadlock detection in both DASH and Origin is conservatively

done based on local information, and a processor that receives a backoff is guaranteed that it will eventually receive the data, while a processor that receives a NAK must retry its request.

Since the Origin system is able to maintain coherence over 1024 processors, it obviously employs a more scalable directory scheme than in DASH. For tracking sharers, Origin supports a bit-vector directory format with either 16 or 64 bits. Each bit represents a node, so with a single bit to node correspondence the directory can track up to a maximum of 128 processors. For systems with greater than 64 nodes, Origin dynamically selects between a full bit vector and coarse bit vector[12] depending on where the sharers are located. This dynamic selection is based on the machine being divided into up to eight 64 node *octants*. If all the processors sharing the cache line are from the same octant, the full bit vector is used (in conjunction with a 3-bit octant identifier). If the processors sharing the cache line are from different octants, a coarse bit vector where each bit represents eight nodes is employed.

Finally, the coherence protocol includes an important feature for effective page migration known as *directory poisoning*. The use of directory poisoning will be discussed in more detail in Section 3.6.

A slightly simplified flow of the cache coherence protocol is now presented for both read, read-exclusive, and writeback requests. We start with the basic flow for a read request.

1. Processor issues read request.
2. Read request goes across network to home memory (requests to local memory only traverse Hub).
3. Home memory does memory read and directory lookup.
4. If directory state is Unowned or Exclusive with requestor as owner, transitions to Exclusive and returns an exclusive reply to the requestor. *Go to 5a.*
   If directory state is Shared, the requesting node is marked in the bit vector and a shared reply is returned to the requestor. *Go to 5a.*
   If directory state is Exclusive with another owner, transitions to Busy-shared with requestor as owner and send out an intervention shared request to the previous owner and a speculative reply to the requestor. *Go to 5b.*
   If directory state is Busy, a negative acknowledgment is sent to the requestor, who must retry the request. QED

5a. Processor receives exclusive or shared reply and fills cache in CEX or shared (SHD) state respectively. QED

5b. Intervention shared received by owner. If owner has a dirty copy it sends an shared response to the requestor and a sharing writeback to the directory. If owner has a clean-exclusive or invalid copy it sends an shared ack (no data) to the requestor and a sharing transfer (no data) to the directory.

6a. Directory receives shared writeback or shared transfer, updates memory (only if shared writeback) and transitions to the shared state.

6b. Processor receives both speculative reply and shared response or ack. Cache filled in SHD state with data from response (if shared response) or data from speculative reply (if shared ack). QED

The following list details the basic flow for a read-exclusive request.

1. Processor issues read-exclusive request.
2. Read-exclusive request goes across network to home memory (only traverses Hub if local).
3. Home memory does memory read and directory lookup.
4. If directory state is Unowned or Exclusive with requestor as owner, transitions to Exclusive and returns an exclusive reply to the requestor. *Go to 5a.*
   If directory state is Shared, transitions to Exclusive and a exclusive reply with invalidates pending is returned to the requestor. Invalidations are sent to the sharers. *Go to 5b.*
   If directory state is Exclusive with another owner, transitions to Busy-Exclusive with requestor as owner and sends out an intervention exclusive request to the previous owner and a speculative reply to the requestor. *Go to 5c.*
   If directory state is Busy, a negative acknowledgment is sent to the requestor, who must retry the request. QED

5a. Processor receives exclusive reply and fills cache in dirty exclusive (DEX) state. QED

5b. Invalidates received by sharers. Caches invalidated and invalidate acknowledgments sent to requestor. *Go to 6a.*

5c. Intervention shared received by owner. If owner has a dirty copy it sends an exclusive response to the requestor and a dirty transfer (no data) to the directory. If owner has a clean-exclusive or invalid copy it sends an exclusive ack to the requestor and a dirty transfer to the directory. *Go to 6b.*

6a. Processor receives exclusive reply with invalidates pending and all invalidate acks. (Exclusive reply with invalidates pending has count of invalidate acks to expect.) Processor fills cache in DEX state. QED

6b. Directory receives dirty transfer and transitions to the exclusive state with new owner.

6c. Processor receives both speculative reply and exclusive response or ack. Cache filled in DEX state with data from response (if exclusive response) or data from speculative reply (if exclusive ack). QED

The flow for an upgrade (write hit to SHD state) is similar to the read-exclusive, except it only succeeds for the case where the directory is in the shared state (and the equivalent reply to the exclusive reply with invalidates pending does not need to send the memory data). In all other cases a negative acknowledgment is sent to the requestor in response to the upgrade request.

Finally, the flow for a writeback request is presented. Note that if a writeback encounters the directory in one of the busy states, this means that the writeback was issued before an intervention targeting the cache line being written back made it to the writeback issuer. This race is resolved in the Origin protocol by "bouncing" the writeback data off the memory as a response to the processor that caused the intervention, and sending a special type of writeback acknowledgment that informs the writeback issuer to wait for (and then ignore) the intervention in addition to the writeback acknowledgment.

1. Processor issues writeback request.
2. Writeback request goes across network to home memory (only traverses Hub if local).
3. Home memory does memory write and directory lookup.
4. If directory state is Exclusive with requestor as owner, transitions to Unowned and returns a writeback exclusive acknowledge to the requestor. *Go to 5a.*
   If directory state is Busy-shared, transitions to Shared, a shared response is returned to the owner marked in the directory. A writeback busy acknowledgment is also sent to the requestor. *Go to 5b.*
   If directory state is Busy-exclusive, transitions to Exclusive, an exclusive response is returned to the owner marked in the directory. A writeback busy acknowledgment is also sent to the requestor. *Go to 5b.*

5a. Processor receives writeback exclusive acknowledgment. QED

5b. Processor receives both a writeback busy acknowledgment and an intervention. QED

3.3 Node Design

The design of an Origin node fits on a single 16" x 11" printed circuit board. A drawing of the Origin node board is shown in Figure

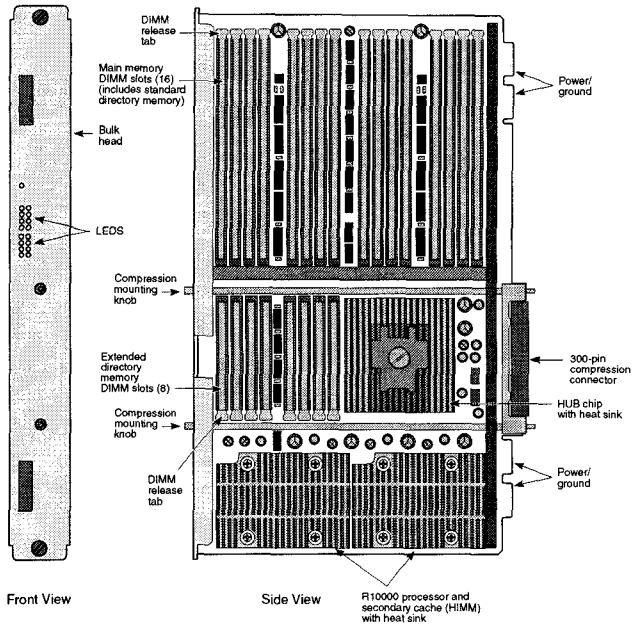

Figure 5 An Origin node board

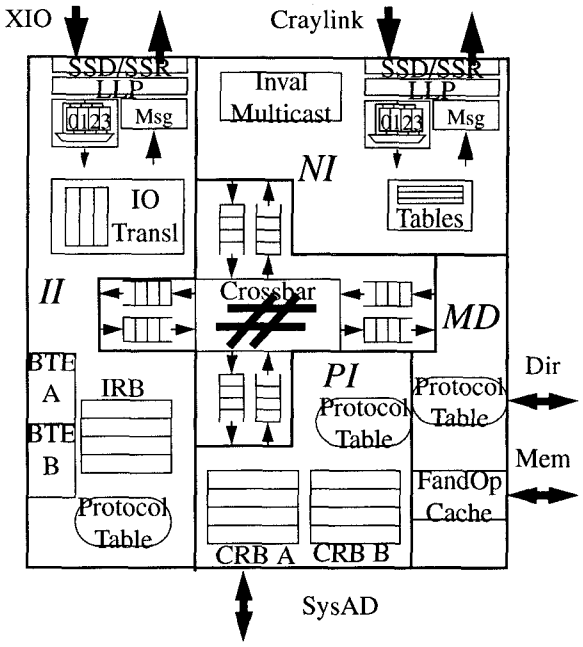

Figure 6 Hub ASIC block diagram

5. At the bottom of the board are two R10000 processors with their secondary caches. The R10000 is a four-way out-of-order superscalar processor[14]. Current Origin systems run the processor at 195 MHz and contain 4 MB secondary caches. Each processor and its secondary cache is mounted on a horizontal in-line memory module (HIMM) daughter card. The HIMM is parallel to the main node card and connects via low-inductance fuzz-button processor and HIMM interposers. The system interface buses of the R10000s are connected to the Hub chip. The Hub chip also has connections to the memory and directory on the node board, and has two ports that exit the node board via the 300-pin CPOP (compression pad-on-pad) connector. These two ports are the Craylink connection to router network and the XIO connection to the IO subsystem.

As was mentioned in Section 3.2, a 16 bit-vector directory format and a 64 bit-vector format are supported by the Origin system. The directory that implements the 16-bit vector format is located on the same DIMMs as main memory. For systems larger than 32 processors, additional expansion directory is needed. These expansion directory slots, shown to the left of the Hub chip in Figure 5, operate by expanding the width of the standard directory included on the main memory boards. The Hub chip operates on standard 16-bit directory entries by converting them to expanded entries upon their entry into the Hub chip. All directory operations within the Hub chip are done on the expanded directory entries, and the results are then converted back to standard entries before being written back to the directory memory. Expanded directory entries obviously bypass the conversion stages.

Figure 6 shows a block diagram of the Hub chip. The hub chip is divided into five major sections: the crossbar (XB), the IO interface (II), the network interface (NI), the processor interface (PI), and the memory and directory interface (MD). All the interfaces communicate with each other via FIFOs that connect to the crossbar.

The IO interface contains the translation logic for interfacing to the XIO IO subsystem. The XIO subsystem is based on the same low-level signalling protocol as the Craylink network (and uses the same interface block to the XIO pins as in the SPIDER router of Figure 2), but utilizes a different higher level message protocol. The IO section also contains the logic for two block transfer engines (BTEs) which are able to do memory to memory copies at near the peak of a node's memory bandwidth. It also implements the IO request tracking portion of the cache coherence protocol via the IO request buffers (IRB) and the IO protocol table. The IRB tracks both full and partial cache line DMA requests by IO devices as well as full cache line requests by the BTEs.

The network interface takes messages from the II, PI, and MD and sends them out on the Craylink network. It also receives incoming messages for the MD, PI, II, and local Hub registers from the Craylink network. Routing tables for outgoing messages are provided in the NI as the software programmable routing of the SPIDER chip is pipelined by one network hop[4]. The NI also is responsible for taking a compact intra-Hub version of the invalidation message resulting from a coherence operation (a bit-vector representation) and generating the multiple unicast invalidate messages required by that message.

The processor interface contains the logic for implementing the request tracking for both processors. Read and write requests are tracked via a coherent request buffer (CRB), with one CRB per processor. The PI also includes the protocol table for its portion of the cache coherence protocol. The PI also has logic for controlling the flow of requests to and from the R10000 processors and contains the logic for generating interrupts to the processors.

Finally, the memory/directory section contains logic for sequencing the external memory and directory synchronous DRAMs (SDRAMs). Memory on a node is banked 4-32 way depending on how many memory DIMMs are populated. Requests to different banks and requests to the same page within a bank as the previous request can be serviced at minimum latency and full bandwidth. Directory operations are performed in parallel with the memory data access. A complete directory entry (and page reference counter, as will be discussed in Section 3.6) read-modify-write can be performed in the same amount of time it takes to fetch the 128B cache line from memory. The MD performs the directory portion of the cache coherence protocol via its protocol table and generates the appropriate requests and/or replies for all incoming messages.

The MD also contains a small fetch-and-op cache which sits in front of the memory. This fetch-and-op cache allows fetch-and-op variables that hit in the cache to be updated at the minimum net-

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.