Public Version – Confidential Information Redacted and Confidentiality Designation Removed Per Agreement Between the Parties

Renée E. Rothauge, OSB #903712
ReneeRothauge@markowitzherbold.com
MARKOWITZ HERBOLD PC
Suite 3000, Pacwest Center
1211 SW Fifth Avenue
Portland, OR 97204-3730
Telephone: (503) 295-3085
Fax: (503) 323-9105

Michael J. Summersgill (*pro hac vice*)
Jordan L. Hirsch (*pro hac vice*)
Sean K. Thompson (*pro hac vice*)
WILMER HALE LLP
60 State Street
Boston, MA 02109
(617) 526-6000

Grant K. Rowan (*pro hac vice*)
WILMER HALE LLP
1875 Pennsylvania Avenue, NW
Washington, DC 20006
(202) 663-6000

Arthur W. Coviello (*pro hac vice*)
WILMER HALE LLP
950 Page Mill Road
Palo Alto, CA 94304
(650) 663-6000

Attorneys for Defendant Intel Corporation

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF OREGON
PORTLAND DIVISION**

MEMORY INTEGRITY, LLC,

Plaintiff,

v.

INTEL CORPORATION,

Defendant.

Case No.: 3:15-cv-00262-SI

Defendant Intel Corporation's INITIAL INVALIDITY CONTENTIONS

Memory Integrity, LLC
IPR2015-00158, -00159, -00163
EXHIBIT
Memory Integrity - 2039

I. INTRODUCTION

Pursuant to Paragraph 5(b) of the Scheduling Order (Dkt. No. 23) and the Court's February 17, 2015 Order Granting in Part Stipulated Motion to Amend Scheduling Order (Dkt. No. 47), Defendant Intel Corporation ("Intel") hereby provides its Initial Invalidity Contentions ("Invalidity Contentions") with respect to the claims of U.S. Patent Nos. 7,296,121 ("the '121 Patent"); 7,107,409 ("the '409 Patent"); 7,103,636 ("the '636 Patent"); 8,572,206 ("the '206 Patent"); and 8,898,254 ("the '254 Patent") (collectively the "Asserted Patents") identified by Plaintiff Memory Integrity, LLC ("MI" or "Memory Integrity") in its March 26, 2015 Initial Infringement Contentions ("Infringement Contentions").

Memory Integrity has asserted the claims listed below against Intel in its Infringement Contentions:

- '121 Patent: claims 1-6, 8, 11-17, 19-25;
- '409 Patent: claims 1-3, 6-12, 18-20, 22-23, 25-30, 34, 36-38, 42-43, 45, 47-49, 51-52;
- '636 Patent: claims 11-18, 21-31, 33-36;
- '206 Patent: claims 1-2, 7, 14-15, 19, 21-22, 24-32, 34-35, 37-41, 43-44; and
- '254 Patent: claims 1-3, 5-8.

With respect to each asserted claim, and based on its investigation to date, Intel hereby: (a) identifies each item of prior art that either anticipates or renders obvious each asserted claim; (b) specifies whether each such item of prior art (or combination of several of the same) anticipates each asserted claim or renders it obvious; (c) submits a chart identifying where specifically in each item of prior art each limitation of each claim is disclosed, described, or taught in the prior art; (d) identifies the grounds for invalidating asserted claims for failing to claim patentable subject matter under 35 U.S.C. § 101, or for invalidating asserted claims based on indefiniteness under 35 U.S.C. § 112(2) or enablement or written description under 35 U.S.C. § 112(1).

II. RESERVATIONS

Intel reserves the right to amend its Invalidity Contentions should Memory Integrity attempt to supplement its deficient Infringement Contentions. Memory Integrity served its Infringement Contentions on March 16, 2015. By letter dated April 22, 2015, Intel informed MI of multiple deficiencies in its Infringement Contentions. In particular, among other deficiencies, Intel informed MI that its Infringement Contentions were deficient because they included improper or inadequate contentions with respect to:

- previously undisclosed products (Intel's Broadwell microprocessors) and claims (claims 12, 21, 23 and 35 of the '636 patent; claims 6, 8, and 52 of the '409 patent; claims 4-6, 13 of the '121 patent; and claims 21-22, 24-29 of the '206 patent);
- Westmere, Ivy Bridge, and Broadwell microprocessors;
- doctrine of equivalents; and
- certain claim limitations for which MI cited no evidence whatsoever.

MI has done nothing to address these deficiencies. Intel reserves its right to amend its Invalidity Contentions should MI attempt to serve amended Infringement Contentions.

MI has also failed to complete its production of documents in response to Intel's October 8, 2014 First Set of Requests for the Production of Documents (Nos. 1-70). Intel reserves its right to revise, supplement, or amend its Invalidity Contentions based on subsequently produced documents and information.

Intel further reserves its right to revise, supplement, or amend its Invalidity Contentions to reflect any additional information learned during the course of fact and expert discovery. In particular, on January 28, 2015 and January 22, 2015 respectively, Intel subpoenaed documents from Oracle Corporation and International Business Machines Corporation regarding prior art systems—certain Sun Server products and IBM's POWER4 microprocessor—that Intel believes show that certain of the asserted claims are invalid, but Intel has not yet received all requested documents from either company. Intel specifically reserves its right to amend its Invalidity Contentions to add additional details regarding this prior art.

The references discussed in the claim charts attached hereto may disclose the elements of the asserted claims explicitly, implicitly, or inherently, or they may be relied upon to show the state of the art in the relevant time frame.

Intel's claim charts cite particular teachings and disclosures of the prior art as applied to features of the asserted claims. However, persons having ordinary skill in the art generally may view an item of prior art in the context of other publications, literature, products, and understanding. As such, the cited portions are only examples, and Intel reserves its right to rely on uncited portions of the prior art references and on other publications and expert testimony as aids in understanding and interpreting the cited portions, as providing context thereto, and as additional evidence that the prior art discloses a claim limitation. Intel further reserves its right to rely on uncited portions of the prior art references, other publications, and testimony to establish bases for combinations of certain cited references that render the asserted claims obvious.

For purposes of these Invalidity Contentions, Intel identifies prior art references and provides element-by-element claim charts based on MI's infringement allegations as set forth in its Infringement Contentions. To the extent MI adopts different positions, Intel reserves its right to revise, supplement, or amend its Invalidity Contentions.

Nothing stated herein shall be treated as an admission or suggestion that Intel agrees with MI's apparent interpretation of the claims. Moreover, nothing in these Contentions shall be treated as an admission that any of Intel's accused technology meets any limitations of the claims. Finally, references to the preamble of a claim in these Contentions shall not be treated as an admission that the preamble is a limitation of a claim.

III. Invalidity Contentions

A. The '121 Patent

1. *Identification of Prior Art*

The references set forth in the table below, in Section III.F Obviousness of the Claimed Concepts, and in Exhibit 1, anticipate and/or render obvious the asserted claims of the '121 patent.

Exhibit No.	Name
C-1	David Chaiken et al., Directory-Based Cache Coherence in Large-Scale Multiprocessors, COMPUTER, June 1990 ("Chaiken")
C-2	U.S. Pat. No. 6,088,769 to Luick ("Luick '769")
C-3	U.S. Pat. No. 6,598,123 to Anderson ("Anderson")
C-4	U.S. Pat. No. 6,810,467 to Khare ("Khare '467")
C-5	U.S. Pat. App. No. 2002/0053004 A1 to Pong ("Pong ")
C-6	Intel 870 Chipset and related references ("870 Chipset and its related publications")
C-7	U.S. Pat. No. 7,698,509 to Koster ("Koster '509")
C-8	Daniel Lenoski et al., The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor, 17th Annual International Symposium on Computer Architecture (1990) ("Lenoski")

2. *Disclosure of Invalidity Due to Anticipation*

Subject to the reservation of rights above and based on Intel's present understanding of the asserted claims of the Asserted Patents and Memory Integrity's apparent construction of the asserted claims as applied in Memory Integrity's infringement contentions, the prior art references identified in Exhibits C-1 – C-8 anticipate the asserted claims, at least under Memory Integrity's apparent infringement and claim construction theories. The charts identify where each element of each asserted claim can be found in each item of prior art.

3. *Disclosure of Invalidity Due to Obviousness*

To the extent a finder of fact determines that a limitation of a given claim was not disclosed by one of the references identified above, those claims are nevertheless unpatentable as obvious because the Asserted Claims contain nothing that goes beyond ordinary innovation. To the extent not anticipated, no asserted claim goes beyond combining known elements to achieve predictable results or does more than choose between clear alternatives known to those of skill in the art.

To the extent that Memory Integrity contends that Chaiken, Luick '769, Anderson, Khare '467, Pong, 870 Chipset and its related publications, Koster '509, and/or Lenoski does not disclose or suggest a point-to-point architecture, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.1 ("Point-to-Point Architecture"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.1 ("Point-to-Point Architecture"), Exhibits C-1 to C-8, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Chaiken, Luick '769, Anderson, Khare '467, Pong, 870 Chipset and its related publications, Koster '509, and/or Lenoski does not disclose or suggest one or more clusters, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.4 ("Clusters"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.4 ("Clusters"), Exhibits C-1 to C-8, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Chaiken, Luick '769, Anderson, Khare '467, Pong, 870 Chipset and its related publications, Koster '509, and/or Lenoski does not disclose or suggest a cache coherence controller and/or interconnection controller, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.7 ("Cache Coherence Controller" and "Interconnection Controller"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.7 ("Cache Coherence Controller" and "Interconnection Controller"), Exhibits C-1 to C-8, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Chaiken, Luick '769, Anderson, Khare '467, Pong, 870 Chipset and its related publications, Koster '509, and/or Lenoski does not disclose or suggest an integrated circuit comprising a probe filtering unit or interconnection controller, computer-readable medium having data structures stored therein representative of a probe filtering unit or interconnection controller, data structures comprising a simulatable representation of a probe filtering unit or interconnection controller, a simulatable representation comprising a netlist, data structures comprising a code description of a probe filtering unit or interconnection controller, code description corresponding to a hardware description language, and/or a set of semiconductor processing masks representative of at least a portion of a probe filtering unit or interconnection controller, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.14 ("Integrated circuit, computer-readable medium, semiconductor processing masks"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.14 ("Integrated circuit, computer-readable medium, semiconductor processing masks"), Exhibits C-1 to C-8, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Chaiken, Luick '769, Anderson, Khare '467, Pong, 870 Chipset and its related publications, Koster '509, and/or Lenoski does not disclose or suggest a probe filtering unit corresponding to an additional node interconnected with the processing nodes, an additional node comprises a cache coherence controller, and/or a cache coherence controller comprises the probe filtering unit, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.15 ("Probe filtering unit corresponds to an additional node interconnected with the processing nodes, additional node comprises a cache coherence controller, cache coherence controller comprises the probe filtering unit"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.15 ("Probe filtering unit corresponds to an additional node interconnected with the processing nodes, additional node comprises a cache coherence controller, cache coherence controller comprises the probe filtering unit"), Exhibits C-1 to C-8, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Chaiken, Luick '769, Anderson, Khare '467, Pong, 870 Chipset and its related publications, Koster '509, and/or Lenoski does not disclose or suggest probe filtering information comprising a cache coherence directory which includes entries corresponding to memory lines stored in the selected cache memories, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.16 ("Probe filtering information comprises a cache coherence directory which includes entries corresponding to memory lines stored in the selected cache memories"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.16 ("Probe filtering information comprises a cache coherence directory which includes entries corresponding to memory lines stored in the selected cache memories"), Exhibits C-1 to C-8, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Chaiken, Luick '769, Anderson, Khare '467, Pong, 870 Chipset and its related publications, Koster '509, and/or Lenoski does not disclose or suggest that each of the processing nodes is operable to transmit the probes only to the probe filtering unit, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.17 ("Each of the processing nodes is operable to transmit the probes only to the probe filtering unit"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.17 ("Each of the processing nodes is operable to transmit the probes only to the probe filtering unit"), Exhibits C-1 to C-8, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Chaiken, Luick '769, Anderson, Khare '467, Pong, 870 Chipset and its related publications, Koster '509, and/or Lenoski does not disclose or suggest each of the processing nodes programmed to complete a memory transaction after receiving a first number of responses to a first probe, the first number being fewer than the number of processing nodes; a probe filtering unit having temporary storage associated therewith for holding read response data from one of the cache memories, where the first number is one; and/or a probe filtering unit operable to forward read response data to a requesting node before accumulating all probe responses associated with the memory transaction, where the first number is two, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.18 ("Each of the processing nodes is programmed to complete a memory transaction after receiving a first number of responses to a first probe, the first number being fewer than the number of processing nodes; Probe filtering unit having temporary storage associated therewith for holding read response data from one of the cache memories, where the first number is one; Probe filtering unit operable to forward read response data to a requesting node before accumulating all probe responses associated with the memory transaction, where the first number is two"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.18 ("Each of the processing nodes is programmed to complete a memory transaction after receiving a first number of responses to a first probe, the first number being fewer than the number of processing nodes; Probe filtering unit having temporary storage associated therewith for holding read response data from one of the cache memories, where the first number is one; Probe filtering unit operable to forward read response data to a requesting node before accumulating all probe responses associated with the memory transaction, where the first number is two"), Exhibits C-1 to C-8, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Chaiken, Luick '769, Anderson, Khare '467, Pong, 870 Chipset and its related publications, Koster '509, and/or Lenoski does not disclose or suggest a probe filtering unit operable to modify the probes such that the selected processing nodes transmit responses to the probes to the probe filtering unit, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.19 ("Probe filtering unit operable to modify the probes such that the selected processing nodes transmit responses to the probes to the probe filtering unit"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.19 ("Probe filtering unit operable to modify the probes such that the selected processing nodes transmit responses to the probes to the probe filtering unit"), Exhibits C-1 to C-8, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Chaiken, Luick '769, Anderson, Khare '467, Pong, 870 Chipset and its related publications, Koster '509, and/or Lenoski does not disclose or suggest a probe filtering unit operable to accumulate responses to each probe, and respond to requesting nodes in accordance with the accumulated responses, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.20 ("Probe filtering unit operable to accumulate responses to each probe, and respond to requesting nodes in accordance with the accumulated responses"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.20 ("Probe filtering unit operable to accumulate responses to each probe, and respond to requesting nodes in accordance with the accumulated responses"), Exhibits C-1 to C-8, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Chaiken, Luick '769, Anderson, Khare '467, Pong, 870 Chipset and its related publications, Koster '509, and/or Lenoski does not disclose or suggest any other particular claimed feature(s), any of these references can be combined with any of the references cited in Exhibits C-1 to C-8 as teaching the particular claimed feature(s). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Exhibits C-1 to C-8 and/or elsewhere in Intel's Invalidity Contentions.

B. The '409 Patent

1. *Identification of Prior Art*

The references set forth in the table below, in Section III.F Obviousness of the Claimed Concepts, and in Exhibit 1, anticipate and/or render obvious the asserted claims of the '409 patent.

Exhibit No.	Name
A-1	U.S. Pat. No. 6,055,610 to Smith et al. ("Smith '610")
A-2	U.S. Pat. No. 6,799,217 to Wilson et al. ("Wilson '217")
A-3	U.S. Pat. No. 6,631,447 to Morioka et al. ("Morioka '447")
A-4	U.S. Pat. No. 6,081,874 to Carpenter ("Carpenter '874")
A-5	U.S. Pat. No. 6,516,391 to Tsushima et al. ("Tsushima")
A-6	U.S. Pat. No. 6,338,122 to Baumgartner ("Baumgartner")

A-7	IBM POWER4 Processor system and related references ("IBM POWER4")
A-8	U.S. Pat. No. 6,615,322 to Arimilli ("Arimilli")
A-9	SunFire server system and related references ("SunFire")

2. *Disclosure of Invalidity Due to Anticipation*

Subject to the reservation of rights above and based on Intel's present understanding of the asserted claims of the Asserted Patents and Memory Integrity's apparent construction of the asserted claims as applied in Memory Integrity's infringement contentions, the prior art references identified in Exhibits A-1 – A-9 anticipate the asserted claims, at least under Memory Integrity's apparent infringement and claim construction theories. The charts identify where each element of each asserted claim can be found in each item of prior art.

3. *Disclosure of Invalidity Due to Obviousness*

To the extent a finder of fact determines that a limitation of a given claim was not disclosed by one of the references identified above, those claims are nevertheless unpatentable as obvious because the Asserted Claims contain nothing that goes beyond ordinary innovation. To the extent not anticipated, no asserted claim goes beyond combining known elements to achieve predictable results or does more than choose between clear alternatives known to those of skill in the art.

To the extent that Memory Integrity contends that Smith '610; IBM Power 4; Wilson '217; Morioka '447; Carpenter '874; Tsushima; Baumgartner; IBM POWER4; Arimilli; and/or SunFire does not disclose or suggest a "point to point architecture," any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.1 ("Point-to-Point Architecture"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.1 ("Point-to-Point Architecture"), Exhibits A-1 to A-9, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Smith '610; IBM Power 4; Wilson '217; Morioka '447; Carpenter '874; Tsushima; Baumgartner; IBM POWER4; Arimilli; and/or SunFire does not disclose or suggest "speculative" "probing," any of these references can be combined with any of those disclosed or cited in Section III.F.2 ("Speculative" "Probing"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.2 ("Speculative" "Probing"), Exhibits A-1 to A-9, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Smith '610; IBM Power 4; Wilson '217; Morioka '447; Carpenter '874; Tsushima; Baumgartner; IBM POWER4; Arimilli; and/or SunFire does not disclose or suggest "determining if speculative probing of the local node can be performed," any of these references can be combined with any of those disclosed or cited in Section III.F.3 ("Determining if Speculative Probing of the Local Node Can Be Performed"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.3 ("Determining if Speculative Probing of the Local Node Can Be Performed"), Exhibits A-1 to A-9, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Smith '610; IBM Power 4; Wilson '217; Morioka '447; Carpenter '874; Tsushima; Baumgartner; IBM POWER4; Arimilli; and/or SunFire does not disclose or suggest "clusters," any of these references can be combined with any of the disclosed or cited in Section III.F.4 ("Clusters"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.4 ("Clusters"), Exhibits A-1 to A-9, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Smith '610; IBM Power 4; Wilson '217; Morioka '447; Carpenter '874; Tsushima; Baumgartner; IBM POWER4; Arimilli; and/or SunFire does not disclose or suggest a "memory controller," any of these references can be combined with any of those disclosed or cited in Section III.F.5 ("Memory Controller"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.5 ("Memory Controller"), Exhibits A-1 to A-9, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Smith '610; IBM Power 4; Wilson '217; Morioka '447; Carpenter '874; Tsushima; Baumgartner; IBM POWER4; Arimilli; and/or SunFire does not disclose or suggest "locking" a "memory line," any of these references can be combined with any of those disclosed or cited in Section III.F.6 ("Locking" a "Memory Line"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.6 ("Locking" a "Memory Line"), Exhibits A-1 to A-9, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Smith '610; IBM Power 4; Wilson '217; Morioka '447; Carpenter '874; Tsushima; Baumgartner; IBM POWER4; Arimilli; and/or SunFire does not disclose or suggest a "cache coherence controller" and/or an "interconnection controller," any of these references can be combined with any of the disclosed or cited in Section III.F.7 ("Cache Coherence Controller" and "Interconnection Controller"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.7 ("Cache Coherence Controller" and "Interconnection Controller"), Exhibits A-1 to A-9, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Smith '610; IBM Power 4; Wilson '217; Morioka '447; Carpenter '874; Tsushima; Baumgartner; IBM POWER4; Arimilli; and/or SunFire does not disclose or suggest a "cache coherence controller" "constructed to act as an aggregate remote cache," any of these references can be combined with any of the disclosed or cited in Section III.F.8 ("Cache Coherence Controller" and "Constructed to Act As An Aggregate Remote Cache"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.8 ("Cache Coherence Controller" and "Constructed to Act As An Aggregate Remote Cache"), Exhibits A-1 to A-9, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Smith '610; IBM Power 4; Wilson '217; Morioka '447; Carpenter '874; Tsushima; Baumgartner; IBM POWER4; Arimilli; and/or SunFire does not disclose or suggest a "cache coherence controller" "constructed to act as a probing agent pair," any of these references can be combined with any of the disclosed or cited in Section III.F.9 ("Cache Coherence Controller" and "Constructed to Act As A Probing Agent Pair"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.9 ("Cache Coherence Controller" and "Constructed to Act As A Probing Agent Pair"), Exhibits A-1 to A-9, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Smith '610; IBM Power 4; Wilson '217; Morioka '447; Carpenter '874; Tsushima; Baumgartner; IBM POWER4; Arimilli; and/or SunFire does not disclose or suggest a "cache coherence controller" "constructed to act as a remote memory," any of these references can be combined with any of those disclosed or cited in Section III.F.10 ("Cache Coherence Controller" and "Constructed to Act As A Remote Memory"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.10 ("Cache Coherence Controller" and "Constructed to Act As A Remote Memory"), Exhibits A-1 to A-9, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Smith '610; IBM Power 4; Wilson '217; Morioka '447; Carpenter '874; Tsushima; Baumgartner; IBM POWER4; Arimilli; and/or SunFire does not disclose or suggest a "shared memory address space," any of these references can be combined with any of those disclosed or cited in Section III.F.11 ("Shared Memory Address Space"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.11 ("Shared Memory Address Space"), Exhibits A-1 to A-9, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Smith '610; IBM Power 4; Wilson '217; Morioka '447; Carpenter '874; Tsushima; Baumgartner; IBM POWER4; Arimilli; and/or SunFire does not disclose or suggest a "protocol engine," any of these references can be combined with any of those disclosed or cited in Section III.F.12 ("Protocol Engines"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.12 ("Protocol Engines"), Exhibits A-1 to A-9, and/or elsewhere in Intel's Invalidity Contentions

To the extent that Memory Integrity contends that US6081874 (Carpenter); IBM Power 4; US 6615322 (Arimilli); US6516391 (Tsushima); US6055610 (Smith); US6799217 (Wilson); US6631447 (Morioka); US6338122 (Baumgartner); and/or Oracle does not disclose or suggest any other particular claimed feature(s), any of these references can be combined with any of the references cited in Exhibits A-1 to A-9 as teaching the particular claimed feature(s). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Exhibits A-1 to A-9 and/or elsewhere in Intel's Invalidity Contentions.

C. The '636 Patent

1. *Identification of Prior Art*

The references set forth in the table below, in Section III.F Obviousness of the Claimed Concepts, and in Exhibit 1, anticipate and/or render obvious the asserted claims of the '636 patent.

Exhibit No.	Name
B-1	U.S. Pat. No. 7,234,029 to Khare ("Khare '029")
B-2	U.S. Pat. No. 6,631,447 to Morioka et al. ("Morioka '447")
B-3	U.S. Pat. No. 6,799,217 to Wilson ("Wilson '217")
B-4	"Using Hints to Reduce the Read Miss Penalty for Flat COMA Protocols" ("Björkman")
B-5	SunFire server system and related references ("SunFire")
B-6	STARRING: Slotted Ring-Based Multiprocessor System with a Central Directory ("Chung")
B-7	Lenoski et al., "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor," IEEE (1990) ("Lenoski DASH 1990")
B-8	U.S. Pat. No. 6,598,120 to Berg et al. ("Berg")
B-9	U.S. Pat. App. Pub. No. 2002/0087807 to Gharachorloo et al. ("Gharachorloo '807")
B-10	U.S. Pat. No. 6,067,611 to Carpenter et al. ("Carpenter '611")
B-11	U.S. Pat. No. 6,101,420 to VanDoren ("VanDoren '420")
B-12	U.S. Pat. App. Pub. No. 2003/0131201 to Khare ("Khare '201")
B-13	International Publication No. WO 00/38070 ("Keller WO '070")
B-14	U.S. Pat. No. 6,516,391 to Tsushima et al. ("Tsushima")
B-15	U.S. Pat. No. 6,711,662 to Peir ("Peir")

B-16	Gharachorloo et al., "Architecture and Design of AlphaServer GS320," Ninth Intrernational Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX) (November 2000) ("Gharachorloo AlphaServer GS320")
B-17	Lenoski et al. "The Stanford Dash Multiprocessor," IEEE (March 1992) ("Lenoski DASH 1992")
B-18	Hsiao et al., Boosting the Performance of NOW-based Shared Memory Multiprocessors Through Directory Hints, Proceedings of the 20th International Conference on Distributed Computing Systems (April 2000) ("Hsiao - Directory Hints")
B-19	U.S. Pat. No. 6,338,122 to Baumgartner ("Baumgartner")

2. *Disclosure of Invalidity Due to Anticipation*

Subject to the reservation of rights above and based on Intel's present understanding of the asserted claims of the Asserted Patents and Memory Integrity's apparent construction of the asserted claims as applied in Memory Integrity's infringement contentions, the prior art references identified in Exhibits B-1 to B-19 anticipate the asserted claims, at least under Memory Integrity's apparent infringement and claim construction theories. The charts identify where each element of each asserted claim can be found in each item of prior art.

3. *Disclosure of Invalidity Due to Obviousness*

To the extent a finder of fact determines that a limitation of a given claim was not disclosed by one of the references identified above, those claims are nevertheless unpatentable as obvious because the Asserted Claims contain nothing that goes beyond ordinary innovation. To the extent not anticipated, no asserted claim goes beyond combining known elements to achieve predictable results or does more than choose between clear alternatives known to those of skill in the art.

To the extent that Memory Integrity contends that Khare '029; Morioka '447; Wilson '217; Björkman; SunFire; Chung; Lenoski DASH 1990; Berg; Gharachorloo '807; Carpenter '611; VanDoren '420; Khare '201; Keller WO '070; Tsushima; Peir; Gharachorloo AlphaServer GS320; Lenoski DASH 1992; Hsiao – Directory Hints; and/or Baumgartner does not disclose or suggest a "point-to-point architecture," any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.1 ("Point-to-Point Architecture"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.1 ("Point-to-Point Architecture"), Exhibits B-1 to B-19, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Khare '029; Morioka '447; Wilson '217; Björkman; SunFire; Chung; Lenoski DASH 1990; Berg; Gharachorloo '807; Carpenter '611; VanDoren '420; Khare '201; Keller WO '070; Tsushima; Peir; Gharachorloo AlphaServer GS320; Lenoski DASH 1992; Hsiao – Directory Hints; and/or Baumgartner does not disclose or suggest "speculative" "probing," any of these references can be combined with any of those disclosed or cited in Section III.F.2 ("Speculative" "Probing"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.2 ("Speculative" "Probing"), Exhibits B-1 to B-19, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Khare '029; Morioka '447; Wilson '217; Björkman; SunFire; Chung; Lenoski DASH 1990; Berg; Gharachorloo '807; Carpenter '611; VanDoren '420; Khare '201; Keller WO '070; Tsushima; Peir; Gharachorloo AlphaServer GS320; Lenoski DASH 1992; Hsiao – Directory Hints; and/or Baumgartner does not disclose or suggest "clusters," any of these references can be combined with any of those disclosed or cited in Section III.F.4 ("Clusters"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.4 ("Clusters"), Exhibits B-1 to B-19, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Khare '029; Morioka '447; Wilson '217; Björkman; SunFire; Chung; Lenoski DASH 1990; Berg; Gharachorloo '807; Carpenter '611; VanDoren '420; Khare '201; Keller WO '070; Tsushima; Peir; Gharachorloo AlphaServer GS320; Lenoski DASH 1992; Hsiao – Directory Hints; and/or Baumgartner does not disclose or suggest a "memory controller," any of these references can be combined with any of those disclosed or cited in Section III.F.5 ("Memory Controller"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.5 ("Memory Controller"), Exhibits B-1 to B-19, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Khare '029; Morioka '447; Wilson '217; Björkman; SunFire; Chung; Lenoski DASH 1990; Berg; Gharachorloo '807; Carpenter '611; VanDoren '420; Khare '201; Keller WO '070; Tsushima; Peir; Gharachorloo AlphaServer GS320; Lenoski DASH 1992; Hsiao – Directory Hints; and/or Baumgartner does not disclose or suggest "locking" a "memory line," any of these references can be combined with any of those disclosed or cited in Section III.F.6 ("Locking" a "Memory Line"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.6 ("Locking" a "Memory Line"), Exhibits B-1 to B-19, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Khare '029; Morioka '447; Wilson '217; Björkman; SunFire; Chung; Lenoski DASH 1990; Berg; Gharachorloo '807; Carpenter '611; VanDoren '420; Khare '201; Keller WO '070; Tsushima; Peir; Gharachorloo AlphaServer GS320;

Lenoski DASH 1992; Hsiao – Directory Hints; and/or Baumgartner does not disclose or suggest a "cache coherence controller" and/or "interconnection controller," any of these references can be combined with any of those disclosed or cited in Section III.F.7 ("Cache Coherence Controller" and "Interconnection Controller"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.7 ("Cache Coherence Controller" and "Interconnection Controller"), Exhibits B-1 to B-19, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Khare '029; Morioka '447; Wilson '217; Björkman; SunFire; Chung; Lenoski DASH 1990; Berg; Gharachorloo '807; Carpenter '611; VanDoren '420; Khare '201; Keller WO '070; Tsushima; Peir; Gharachorloo AlphaServer GS320; Lenoski DASH 1992; Hsiao – Directory Hints; and/or Baumgartner does not disclose or suggest a "cache coherence controller" "constructed to act as an aggregate remote cache," any of these references can be combined with any of those disclosed or cited in Section III.F.8 ("Cache Coherence Controller" and "Constructed to Act As An Aggregate Remote Cache"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.8 ("Cache Coherence Controller" and "Constructed to Act As An Aggregate Remote Cache"), Exhibits B-1 to B-19, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Khare '029; Morioka '447; Wilson '217; Björkman; SunFire; Chung; Lenoski DASH 1990; Berg; Gharachorloo '807; Carpenter '611; VanDoren '420; Khare '201; Keller WO '070; Tsushima; Peir; Gharachorloo AlphaServer GS320; Lenoski DASH 1992; Hsiao – Directory Hints; and/or Baumgartner does not disclose or suggest a "cache coherence controller" "constructed to act as a probing agent pair," any of these references can be combined with any of those disclosed or cited in Section III.F.9 ("Cache Coherence Controller" and "Constructed to Act As A Probing Agent Pair"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.9 ("Cache Coherence Controller" and "Constructed to Act As A Probing Agent Pair"), Exhibits B-1 to B-19, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Khare '029; Morioka '447; Wilson '217; Björkman; SunFire; Chung; Lenoski DASH 1990; Berg; Gharachorloo '807; Carpenter '611; VanDoren '420; Khare '201; Keller WO '070; Tsushima; Peir; Gharachorloo AlphaServer GS320; Lenoski DASH 1992; Hsiao – Directory Hints; and/or Baumgartner does not disclose or suggest a "cache coherence controller" "constructed to act as a remote memory," any of these references can be combined with any of those disclosed or cited in Section III.F.10 ("Cache Coherence Controller" and "Constructed to Act As A Remote Memory"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.10 ("Cache Coherence Controller" and "Constructed to Act As A Remote Memory"), Exhibits B-1 to B-19, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Khare '029; Morioka '447; Wilson '217; Björkman; SunFire; Chung; Lenoski DASH 1990; Berg; Gharachorloo '807; Carpenter '611; VanDoren '420; Khare '201; Keller WO '070; Tsushima; Peir; Gharachorloo AlphaServer GS320; Lenoski DASH 1992; Hsiao – Directory Hints; and/or Baumgartner does not disclose or suggest a "shared memory address space," any of these references can be combined with any of the disclosed or cited in Section III.F.11 ("Shared Memory Address Space"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.11 ("Shared Memory Address Space"), Exhibits B-1 to B-19, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Khare '029; Morioka '447; Wilson '217; Björkman; SunFire; Chung; Lenoski DASH 1990; Berg; Gharachorloo '807; Carpenter '611; VanDoren '420; Khare '201; Keller WO '070; Tsushima; Peir; Gharachorloo AlphaServer GS320; Lenoski DASH 1992; Hsiao – Directory Hints; and/or Baumgartner does not disclose or suggest a "protocol engine," any of these references can be combined with any of the disclosed or cited in Section III.F.12 ("Protocol Engines"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.12 ("Protocol Engines"), Exhibits B-1 to B-19, and/or elsewhere in Intel's Invalidity Contentions

To the extent that Memory Integrity contends that Khare '029; Morioka '447; Wilson '217; Björkman; SunFire; Chung; Lenoski DASH 1990; Berg; Gharachorloo '807; Carpenter '611; VanDoren '420; Khare '201; Keller WO '070; Tsushima; Peir; Gharachorloo AlphaServer GS320; Lenoski DASH 1992; Hsiao – Directory Hints; and/or Baumgartner does not disclose or suggest any other particular claimed feature(s), any of these references can be combined with any of the references cited in Exhibits A-1 to A-9 as teaching the particular claimed feature(s). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Exhibits B-1 to B-19 and/or elsewhere in Intel's Invalidity Contentions.

D. The '206 Patent

1. *Identification of Prior Art*

The references set forth in the table below, in Section III.F Obviousness of the Claimed Concepts, and in Exhibit 1, anticipate and/or render obvious the asserted claims of the '206 patent.

Exhibit No.	Name
D-1	Ilanthiraiyan Pragaspathy and Babak Falsafi, "Address Partitioning in DSM Clusters with Parallel Coherence Controllers," IEEE (2000) ("Pragaspathy")
D-2	U.S. Pat. No. 6,721,858 to Joseph ("Joseph")
D-3	The Thread-Based Protocol Engines for CC-NUMA Multiprocessor by Hung-Chang Hsiao and Chung-Ta King ("Hsiao")
D-4	Ashwini K. Nanda, Anthony-Trung Nguyen, Maged Michael, Doug Joseph, "High-Throughput Coherence Controllers," Proceedings of High-Performance Computer Architecture, pp. 145 – 155, January 2000 ("Nanda HPCA")
D-5	Maged M. Michael, Ashwini K. Nanda, Beng-Hong Lim, Michael L. Scott, "Coherence Controller Architectures for SMP-Based CC-NUMA Multiprocessors," Proceedings of the 24th International Symposium on Computer Architecture, June 1997, pp. 219-228 ("Michael ISCA")
D-6	Maged M. Michael, Ashwini K. Nanda, Beng-Hong Lim, "Coherence Controller Architectures for Scalable Shared-Memory Multiprocessors," IEEE Transactions on Computers, Vol. 48, No. 2, pp.245-255, February 1999 ("Michael ITC")
D-7	S3.mp Scalable Shared Memory Multiprocessor by Nowatzyk et al. ("Nowatzyk B")
D-8	Exploiting Parallelism in Cache Coherency Protocol Engines by Nowatzyk et al. ("Nowatzyk A")
D-9	U.S. Pat. App. Pub. No. 2002/0007443 to Gharachorloo et al. ("Gharachorloo '443")
D-10	U.S. Pat. No. 6,370,585 to Hagersten et al. ("Hagersten") in view of U.S. Pat. No. 6,304,910 ("Roach") in further view of U.S. Pat. App. Pub. No. 2002/0007443 ("Gharachorloo '443") ("Hagersten in view of Roach in view of Gharachorloo '443")
D-11	Ashwini Nanda, Anthony-Trung Nguyen, Maged Michael, Doug Joseph, "High-throughput Coherence Control and Hardware Messaging in Everest," IBM Journal of Research and Development, Vol. 45, No. 2, March 2001, pp. 229-243. ("Nanda IJRD")
D-12	U.S. Pat. App. Pub. No. 2002/0087807 ("Gharachorloo '807")
D-13	"Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing", 28:2 ACM SIGARCH Computer Architecture News – Special Issue: Proceedings of the 27th annual international symposium on Computer architecture (ISCA '00) pp. 282-293 (May 2000) ("Piranha")
D-14	Anthony Nguyen, "High-Throughput Coherence Controllers," PhD Thesis, University of Illinois at Urbana-Champaign, 2001 ("Nguyen")

2. *Disclosure of Invalidity Due to Anticipation*

Subject to the reservation of rights above and based on Intel's present understanding of the asserted claims of the Asserted Patents and Memory Integrity's apparent construction of the asserted claims as applied in Memory Integrity's infringement contentions, the prior art references identified in Exhibits D-1 to D-14 anticipate the asserted claims, at least under Memory Integrity's apparent infringement and claim construction theories. The charts identify where each element of each asserted claim can be found in each item of prior art.

3. *Disclosure of Invalidity Due to Obviousness*

To the extent a finder of fact determines that a limitation of a given claim was not disclosed by one of the references identified above, those claims are nevertheless unpatentable as obvious because the Asserted Claims contain nothing that goes beyond ordinary innovation. To the extent not anticipated, no asserted claim goes beyond combining known elements to achieve predictable results or does more than choose between clear alternatives known to those of skill in the art.

To the extent that Memory Integrity contends that Pragaspathy, Joseph, Hsiao, Nanda HPCA, Michael ISCA, Michael ITC, Nowatzyk B, Nowatzyk A, Gharachorloo '443, Hagersten in view of Roach in view of Gharachorloo '443, Nanda IJRD, Gharachorloo '807, Piranha, and/or Nguyen does not disclose or suggest a point-to-point architecture, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.1 ("Point-to-Point Architecture"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.1 ("Point-to-Point Architecture"), Exhibits D-1 to D-14, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Pragaspathy, Joseph, Hsiao, Nanda HPCA, Michael ISCA, Michael ITC, Nowatzyk B, Nowatzyk A, Gharachorloo '443, Hagersten in view of Roach in view of Gharachorloo '443, Nanda IJRD, Gharachorloo '807, Piranha, and/or Nguyen does not disclose or suggest one or more clusters, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.4 ("Clusters"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.4 ("Clusters"), Exhibits D-1 to D-14, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Pragaspathy, Joseph, Hsiao, Nanda HPCA, Michael ISCA, Michael ITC, Nowatzyk B, Nowatzyk A, Gharachorloo '443, Hagersten in view of Roach in view of Gharachorloo '443, Nanda IJRD, Gharachorloo '807, Piranha, and/or Nguyen does not disclose or suggest a cache coherence controller, interconnection controller, and/or an interconnection controller operable to facilitate cache coherency, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.7 ("Cache Coherence Controller" and "Interconnection Controller"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.7 ("Cache Coherence Controller" and "Interconnection Controller"), Exhibits D-1 to D-14, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Pragaspathy, Joseph, Hsiao, Nanda HPCA, Michael ISCA, Michael ITC, Nowatzyk B, Nowatzyk A, Gharachorloo '443, Hagersten in view of Roach in view of Gharachorloo '443, Nanda IJRD, Gharachorloo '807, Piranha, and/or Nguyen does not disclose or suggest one or more protocol engines and related features thereto, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.12 ("Protocol Engines"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.12 ("Protocol Engines"), Exhibits D-1 to D-14, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Pragaspathy, Joseph, Hsiao, Nanda HPCA, Michael ISCA, Michael ITC, Nowatzyk B, Nowatzyk A, Gharachorloo '443, Hagersten in view of Roach in view of Gharachorloo '443, Nanda IJRD, Gharachorloo '807, Piranha, and/or Nguyen does not disclose or suggest an integrated circuit comprising a probe filtering unit or interconnection controller, computer-readable medium having data structures stored therein representative of a probe filtering unit or interconnection controller, data structures comprising a simulatable representation of a probe filtering unit or interconnection controller, a simulatable representation comprising a netlist, data structures comprising a code description of a probe filtering unit or interconnection controller, code description corresponding to a hardware description language, and/or a set of semiconductor processing masks representative of at least a portion of a probe filtering unit or interconnection controller, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.14 ("Integrated circuit, computer-readable medium, semiconductor processing masks"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.14 ("Integrated circuit, computer-readable medium, semiconductor processing masks"), Exhibits D-1 to D-14, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Pragaspathy, Joseph, Hsiao, Nanda HPCA, Michael ISCA, Michael ITC, Nowatzyk B, Nowatzyk A, Gharachorloo '443, Hagersten in view of Roach in view of Gharachorloo '443, Nanda IJRD, Gharachorloo '807, Piranha, and/or Nguyen does not disclose or suggest any other particular claimed feature(s), any of these references can be combined with any of the references cited in Exhibits D-1 to D-14 as teaching the particular claimed feature(s). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Exhibits D-1 to D-14 and/or elsewhere in Intel's Invalidity Contentions.

E. The '254 Patent

1. *Identification of Prior Art*

The references set forth in the table below, in Section III.F Obviousness of the Claimed Concepts, and in Exhibit 1, anticipate and/or render obvious the asserted claims of the '254 patent.

Exhibit No.	Name
E-1	Anthony Nguyen, "High-Throughput Coherence Controllers," PhD Thesis, University of Illinois at Urbana-Champaign, 2001 ("Nguyen")
E-2	U.S. Pat. No. 6,370,585 to Hagersten et al. ("Hagersten") in view of U.S. Pat. No. 6,304,910 ("Roach") in further view of U.S. Pat. App. Pub. No. 2002/0007443 ("Gharachorloo '443") ("Hagersten in view of Roach in view of Gharachorloo '443")
E-3	S3.mp Scalable Shared Memory Multiprocessor by Nowatzyk et al. ("Nowatzyk B")
E-4	U.S. Pat. No. 6,721,858 to Joseph ("Joseph")
E-5	Ashwini K. Nanda, Anthony-Trung Nguyen, Maged Michael, Doug Joseph, "High-Throughput Coherence Controllers," Proceedings of High-Performance Computer Architecture, pp. 145 – 155, January 2000 ("Nanda HPCA")
E-6	U.S. Pat. App. Pub. No. 2002/0087807 to Gharachorloo et al. ("Gharachorloo '807")
E-7	Maged M. Michael, Ashwini K. Nanda, Beng-Hong Lim, Michael L. Scott, "Coherence Controller Architectures for SMP-Based CC-NUMA Multiprocessors," Proceedings of the 24th International Symposium on Computer Architecture, June 1997, pp. 219-228 ("Michael ISCA")
E-8	Exploiting Parallelism in Cache Coherency Protocol Engines by Nowatzyk et al. ("Nowatzyk A")
E-9	Anthony Nguyen, "High-Throughput Coherence Controllers," PhD Thesis, University of Illinois at Urbana-Champaign, 2001 ("Nguyen")
E-10	The Thread-Based Protocol Engines for CC-NUMA Multiprocessor by Hung-Chang Hsiao and Chung-Ta King ("Hsiao")
E-11	U.S. Pat. App. Pub. No. 2002/0007443 to Gharachorloo et al. ("Gharachorloo '443")
E-12	Maged M. Michael, Ashwini K. Nanda, Beng-Hong Lim, "Coherence Controller Architectures for Scalable Shared-Memory Multiprocessors," IEEE Transactions on Computers, Vol. 48, No. 2, pp.245-255, February 1999 ("Michael ITC")
E-13	"Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing", 28:2 ACM SIGARCH Computer Architecture News – Special Issue: Proceedings of the 27th annual international symposium on Computer architecture (ISCA '00) pp. 282-293 (May 2000) ("Piranha")
E-14	Ilanthiraiyan Pragaspathy and Babak Falsafi, "Address Partitioning in DSM Clusters with Parallel Coherence Controllers, 2000 ("Pragaspathy")

2. *Disclosure of Invalidity Due to Anticipation*

Subject to the reservation of rights above and based on Intel's present understanding of the asserted claims of the Asserted Patents and Memory Integrity's apparent construction of the asserted claims as applied in Memory Integrity's infringement contentions, the prior art references identified in Exhibits E-1 to E-14 anticipate the asserted claims, at least under Memory Integrity's apparent infringement and claim construction theories. The charts identify where each element of each asserted claim can be found in each item of prior art.

3. *Disclosure of Invalidity Due to Obviousness*

To the extent a finder of fact determines that a limitation of a given claim was not disclosed by one of the references identified above, those claims are nevertheless unpatentable as obvious because the Asserted Claims contain nothing that goes beyond ordinary innovation. To the extent not anticipated, no asserted claim goes beyond combining known elements to achieve predictable results or does more than choose between clear alternatives known to those of skill in the art.

To the extent that Memory Integrity contends that Nguyen, Hagersten in view of Roach in view of Gharachorloo '443, Nowatzyk B, Joseph, Nanda HPCA, Gharachorloo '807, Michael ISCA, Nowatzyk A, Nguyen, Hsiao, Gharachorloo '443, Michael ITC, Piranha, and/or Pragaspathy does not disclose or suggest a point-to-point architecture, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.1 ("Point-to-Point Architecture"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.1 ("Point-to-Point Architecture"), Exhibits E-1 to E-14, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Nguyen, Hagersten in view of Roach in view of Gharachorloo '443, Nowatzyk B, Joseph, Nanda HPCA, Gharachorloo '807, Michael

ISCA, Nowatzyk A, Nguyen, Hsiao, Gharachorloo '443, Michael ITC, Piranha, and/or Pragaspathy does not disclose or suggest one or more clusters, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.4 ("Clusters"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.4 ("Clusters"), Exhibits E-1 to E-14, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Nguyen, Hagersten in view of Roach in view of Gharachorloo '443, Nowatzyk B, Joseph, Nanda HPCA, Gharachorloo '807, Michael ISCA, Nowatzyk A, Nguyen, Hsiao, Gharachorloo '443, Michael ITC, Piranha, and/or Pragaspathy does not disclose or suggest a cache coherence controller and/or interconnection controller, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.7 ("Cache Coherence Controller" and "Interconnection Controller"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.7 ("Cache Coherence Controller" and "Interconnection Controller"), Exhibits E-1 to E-14, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Nguyen, Hagersten in view of Roach in view of Gharachorloo '443, Nowatzyk B, Joseph, Nanda HPCA, Gharachorloo '807, Michael ISCA, Nowatzyk A, Nguyen, Hsiao, Gharachorloo '443, Michael ITC, Piranha, and/or Pragaspathy does not disclose or suggest a shared memory address space, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.11 ("Shared Memory Address Space"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.11 ("Shared Memory Address Space"), Exhibits E-1 to E-14, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Nguyen, Hagersten in view of Roach in view of Gharachorloo '443, Nowatzyk B, Joseph, Nanda HPCA, Gharachorloo '807, Michael ISCA, Nowatzyk A, Nguyen, Hsiao, Gharachorloo '443, Michael ITC, Piranha, and/or Pragaspathy does not disclose or suggest one or more protocol engines and related features thereto, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.12 ("Protocol Engines"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.12 ("Protocol Engines"), Exhibits E-1 to E-14, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Nguyen, Hagersten in view of Roach in view of Gharachorloo '443, Nowatzyk B, Joseph, Nanda HPCA, Gharachorloo '807, Michael ISCA, Nowatzyk A, Nguyen, Hsiao, Gharachorloo '443, Michael ITC, Piranha, and/or Pragaspathy does not disclose or suggest a protocol engine configured to process interrupts, any of these references can be combined with each other and/or any of those disclosed or cited in Section III.F.13 ("Protocol engine configured to process interrupts"). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Section III.F.13 ("Protocol engine configured to process interrupts"), Exhibits E-1 to E-14, and/or elsewhere in Intel's Invalidity Contentions.

To the extent that Memory Integrity contends that Nguyen, Hagersten in view of Roach in view of Gharachorloo '443, Nowatzyk B, Joseph, Nanda HPCA, Gharachorloo '807, Michael ISCA, Nowatzyk A, Nguyen, Hsiao, Gharachorloo '443, Michael ITC, Piranha, and/or

Pragaspathy does not disclose or suggest any other particular claimed feature(s), any of these references can be combined with any of the references cited in Exhibits E-1 to E-14 as teaching the particular claimed feature(s). It would have been obvious for one of ordinary skill in the art at the time of the alleged invention of the Asserted Claims to have made such combination(s) for the reasons set forth in Exhibits E-1 to E-14 and/or elsewhere in Intel's Invalidity Contentions.

F. Obviousness of the Claimed Concepts

The Supreme Court has held that "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." *KSR Int'l Co. v. Teleflex Inc.*, 127 S. Ct. 1727, 1731 (2007). "When a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one." *Id.* As the Supreme Court made clear, "[f]or the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id. 1740.*

The Supreme Court was also clear that "[w]here there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* at 1742. "[A]ny need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed." *Id.* Common sense also teaches that "familiar items may have obvious uses beyond their primary purposes, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle." *Id.*

1. *"Point-to-Point Architecture"*

Some of the Asserted Claims are directed to components to be interconnected in a "point-to-point architecture." For example, claim 1.3 of the '409 patent recites "the first plurality of processors and the first cache coherence controller interconnected in a point-to-point architecture." *See also*, *e.g.*, '409 patent claims 1.5, 6.3, 6.5, 7.3, 23.1, 25.3, 25.5, 34.3, 34.5, 42.2, 43.1, 51.3, and 52.3; '636 patent claims 15.3, 15.5, 21.3, 21.5, 25.1, 26.1, and 34.1; '121 patent claims 1.2, 2.1, 4.2, 5.1, 16.2, and 25.2; '206 patent claims 1.3, 21.3, 31.1, and 32.1; and '254 patent claims 2.1 and 3.1. At least under Memory Integrity's apparent infringement theories, interconnecting components in a "point-to-point architecture" was well-known in the art before the priority dates of the Asserted Patents. *See*, *e.g.*, Exhibits A-1 – A-9, claims 1.3, 1.5, 6.3, 6.5, 7.3, 23.1, 25.3, 25.5, 34.3, 34.5, 42.2, 43.1, 51.3, and 52.3; Exhibits B-1 – B-19, claims 15.3, 15.5, 21.3, 21.5, 25.1, 26.1, and 34.1; Exhibits C-1 – C-8 claims 1.2, 2.1, 4.2, 5.1, 16.2, and 25.2; D-1 – D-14 claims 1.3, 21.3, 31.3, and 32.1; and Exhibits E-1 – E-14 claims 2.1 and 3.1. The following discussion shows that, at least under Memory Integrity's apparent infringement theories, it was well-known and conventional before the priority dates of the Asserted Patents to connect components in a multiprocessor system via a "point-to-point architecture."

As an initial matter, the Asserted Patents acknowledge that a "point-to-point architecture" was well-known. *See*, *e.g.*, the '409 patent at 2:30-35 ("Background of the Invention … Performance limitations have led to the development of a point-to-point architecture for connecting processors in a system with a single memory space. In one example, individual processors can be directly connected to each other through a plurality of point-to-point links to form a cluster of processors."); '636 patent at 1:33-2:59; '121 patent at 1:20-2:38; '206 patent at 1:13-38; and '254 patent at 1:16-41. In addition, during prosecution of all the patents in suit, Applicant cited the HyperTransport I/O Link Specification Revision 1.03, published October 10, 2001 ("HyperTransport Specification"), as prior art to the patent office. *See, e.g.*, '636 patent, Jan. 1, 2003, Information Disclosure Statement; '409 patent, Dec. 12, 2002, Information Disclosure Statement; '121 patent, Apr. 4, 2005, Information Disclosure Statement; '206 patent, Sep. 18, 2012, Information Disclosure Statement; '254 patent, Sep. 9, 2013, Information Disclosure Statement; and '121 patent at 30:20-37, 31:32-34.

HyperTransport technology, developed at AMD, provides a "high-speed, high-performance, point-to-point link for interconnecting integrated circuits on a board," for delivering a "scalable interconnect between CPU, memory, and I/O devices." *See, e.g.*, HypertTransport Specification at 17. The HyperTransport Specification describes a "link [that] is packet-based, nominally point-to-point, and connects exactly two devices." *See, e.g.*, HyperTransport Specification at 19. The prior art HyperTransport technology was again confirmed as a "point-to-point architecture" in U.S. Patent No. 7,719,964 to Morton at 1:4-20 ("Background of the Invention … One such point-to-point architecture is the HyperTransport™ architecture pioneered by AMD of Sunnyvale, Calif.")

Indeed, at least under Memory Integrity's apparent infringement theories, there are many examples of prior art references that disclose a "point-to-point architecture." Two such examples include:

- International Publication No. WO 2000/038069 to Keller, assigned to AMD: *See, e.g.*, 3:35-38 ("Processing nodes 12A-12D implement a packet-based link for inter-processing node communication. In the present embodiment, the link is implemented as sets of unidirectional lines (e.g., lines 24A are used to transmit packets from processing node 12A to processing node 12B and lines 24B are used to transmit packets from processing node 12B to processing node 12A).")

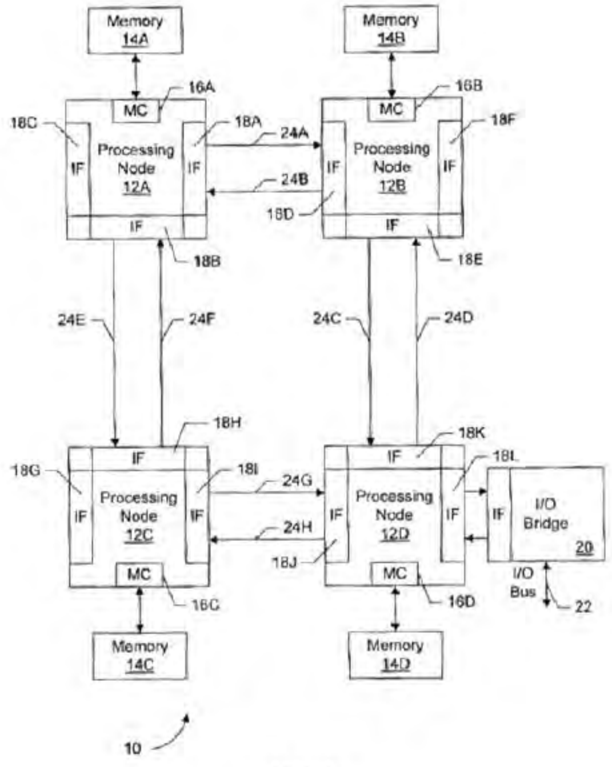
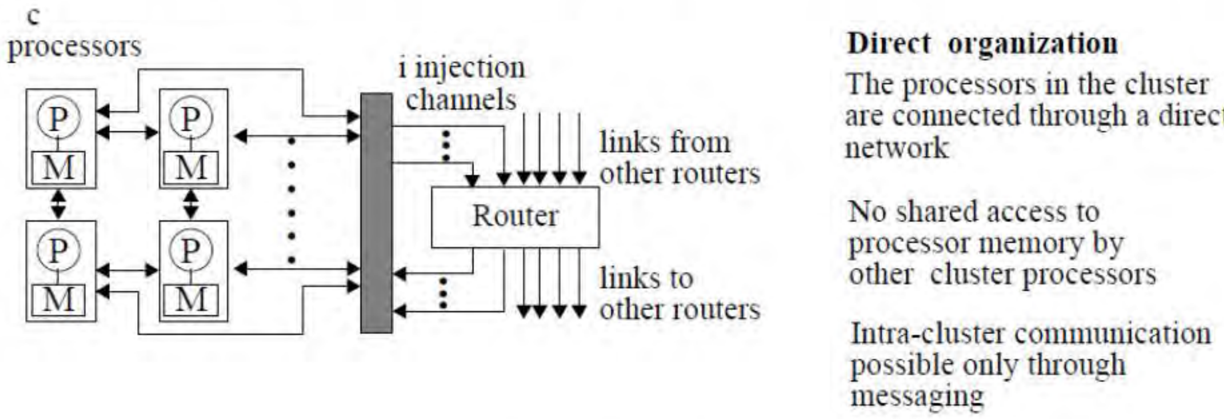


FIG. 1

Keller, Figure 1

- "Designing Processor-cluster Based Systems: Interplay Between Cluster Organizations and Collective Communication Algorithms," Basak (1996): *See, e.g.*, p.5 ("3.2 Direct In this organization the processors inside a cluster are interconnected by a point-point direct network, e.g., a mesh, as shown in Fig. 2").


c processors
P
M
i injection channels
links from other routers
Router
links to other routers
Direct organization
The processors in the cluster are connected through a direct network
No shared access to processor memory by other cluster processors
Intra-cluster communication possible only through messaging

Basak, Figure 2

It would have been obvious to connect processors and a cache coherence/interconnection controller in a "point-to-point architecture" as both were well known before the priority dates of the Asserted Patents. Indeed, the asserted '636 patent states that a cache coherence controller may simply be "a general purpose processor with an interface to the point-to-point links 232." *See, e.g.*,'636 patent, 8:46-48. Thus, it would have been obvious for one of the processing components of Keller or Basak to simply be implemented as a cache coherence/interconnection controller interconnected with the other processing nodes in a "point-to-point architecture" via the point-to-point links to effectuate a coherent cluster of processing components.

Additional prior art references that discuss systems with point-to-point architectures—at least under Memory Integrity's apparent infringement theories—are discussed below.

a) Using a Ring Interconnect to Connect Components in a Multiprocessor System was Known

Under Memory Integrity's apparent infringement theories, a ring appears to correspond to a "point-to-point architecture." Connecting processing components via a ring, however, was well known before the priority dates of the Asserted Patents. Indeed, the Asserted Patents acknowledge that it was known to use a ring (*e.g.,* a Token Ring) to connect components in a multiprocessor system. *See, e.g.*, '409 patent, 2:3-6. Additional examples of prior art references that disclose a ring interconnect and further demonstrate that such a structure was well known include:

- "Parallel Computer Architecture," Culler et al. (1998): *See, e.g.*, p. 769 ("A ring or torus of N nodes can be formed by simply connecting the two ends of an array. With unidirectional links, the diameter is N - 1, the average distance is N/2, the bisection width is 1 link, and there is one route between any pair of nodes.")

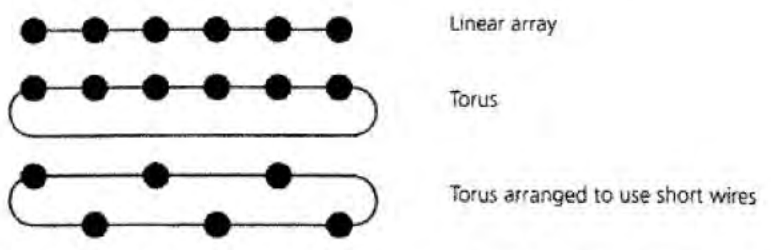


FIGURE 10.6 Linear and ring topologies. The linear array and torus are easily laid out to use uniformly short wires. The distance and cost grow as $O(N)$ whereas the aggregate bandwidth is only $O(1)$.

Culler Figure 10.6

- Culler: *See, e.g.*, pgs. 442-443 ("The potential advantage of rings over buses, other than the use of distributed memory, is that the short, point-to-point nature of the links allows them to be driven at very high clock rates. For example, the IEEE scalable coherent interface (SCI) transport standard (Gustavson 1992; IEEE 1993) is based on 500-MHz 16-bit-wide point-to-point links. The linear point-to-point nature also allows the links to extensively pipelined, that is, new bits can be pumped onto the wire by the before the previous bits have reached the destination." "While it may seem at first that broadcast and snooping waste bandwidth on a point-to-point interconnect such as a ring, in reality it is not necessarily so.")

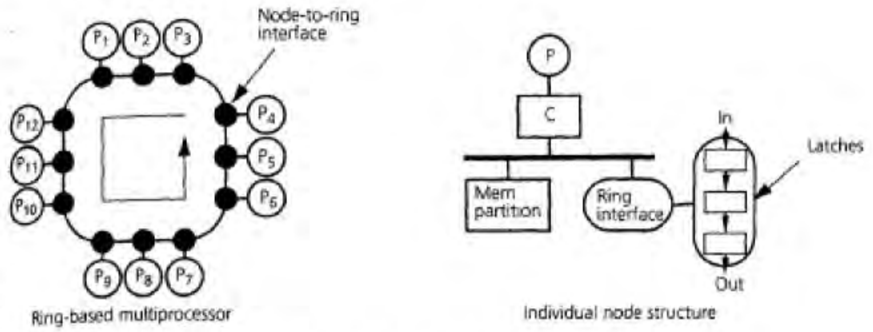


FIGURE 6.25 Organization of a single-ring multiprocessor

Culler, Figure 6.25

- "Scalable Parallel Computing," Hwang (1998): *See, e.g.*, 287 ("A ring is obtained by connecting the two terminal nodes of a linear array with one more link to form a closed loop [Fig. 6.8(b)]. A ring can be unidirectional or bidirectional. It is symmetric with a constant node degree of 2. The diameter equals N/2 for a bidirectional ring, and N- 1 for a unidirectional ring. The IBM Token Ring has this topology, in which messages circulate along the ring until they reach the destination with a matching token.")

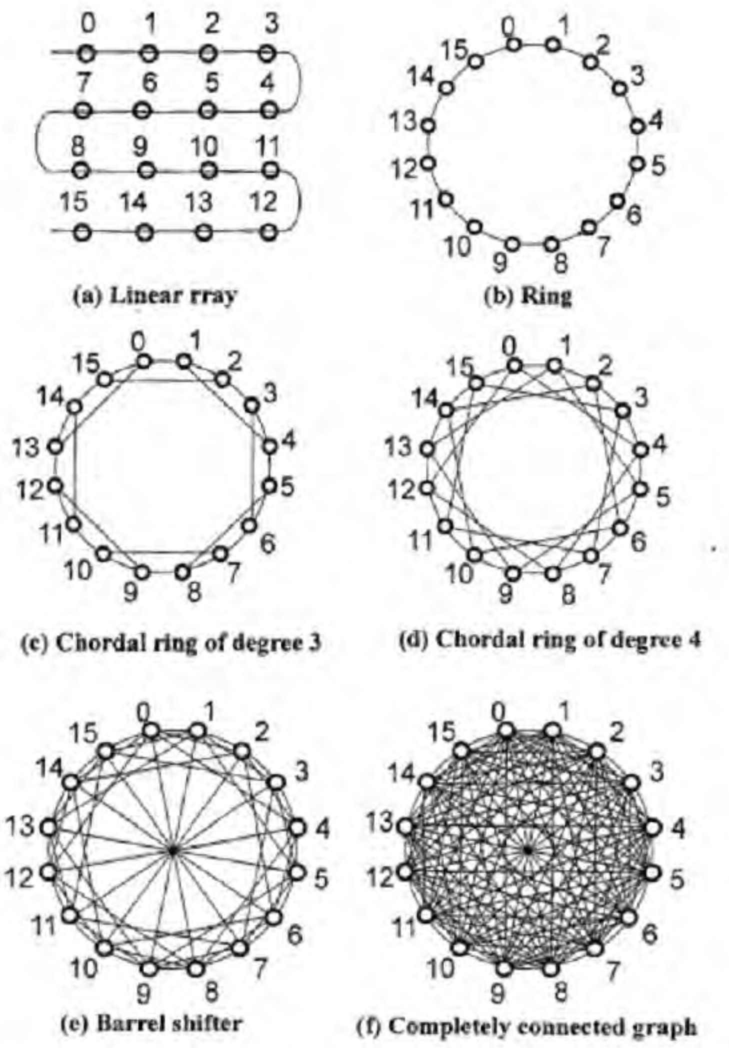

0 1 2 3
7 6 5 4
8 9 10 11
15 14 13 12
(a) Linear rray
(b) Ring
(c) Chordal ring of degree 3
(d) Chordal ring of degree 4
(e) Barrel shifter
(f) Completely connected graph

Figure 6.8 **Six network topologies with increasing node degree and connectivity from a linear array to a ring, two chordal rings, a barrel shifter, and a completely connected graph.**

Hwang, Figure 6.8

- "Interconnection Networks for Multiprocessors and Multicomputers Theory and Practice," Varma (1994): *See, e.g.*, 8 ("For example, in a ring network of N nodes numbered from 0 to N - 1, each processor i is directly connected to processors (i - 1) mod N and (i + 1) mod N.")

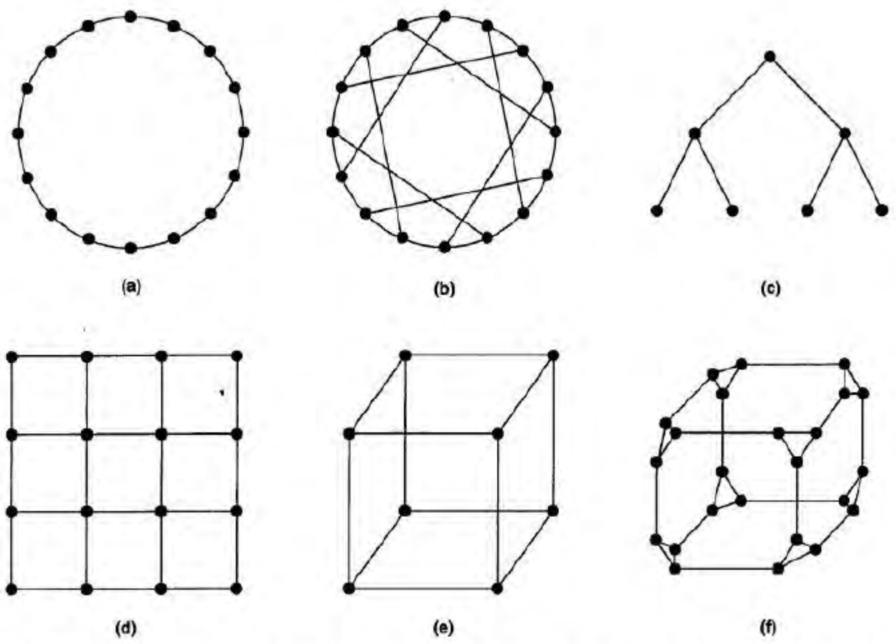
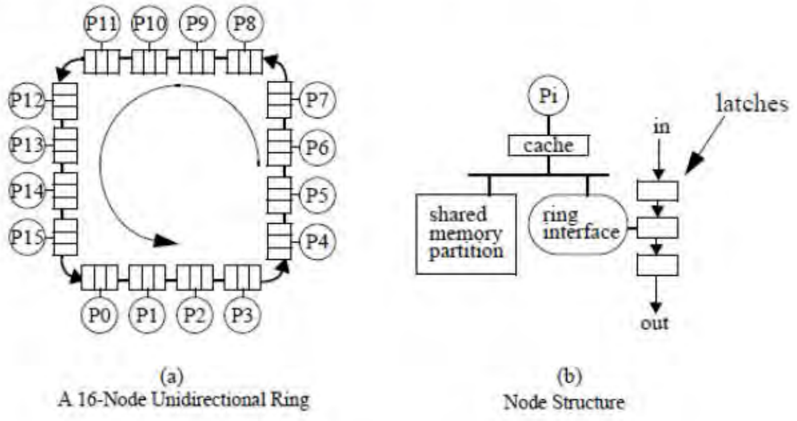


Figure 1: Some examples of static network topologies: (a) A simple ring (b) chordal ring (c) binary tree (d) 2-D mesh (e) binary 3-cube (3-dimensional hypercube) (f) cube-connected cycles.

Varna, Figure 1

- "Design Options for Small Scale Shared Memory Multiprocessors," Barroso (1996): *See, e.g.*, p. 10 ("Point-to-point links are not networks per se, but instead they are the building blocks for a myriad of network topologies. The simplest among those is an unidirectional ring network. Unidirectional rings have the smallest number of links per node (smallest degree), and they do not need intermediate switching elements (as in multistage interconnection networks or crossbars). Consequently, ring networks are likely to be the least expensive point-to-point based interconnects for both multiprocessors and local area networks.")



Barroso, Figure 2.1

- US 5,551,048 to Steely: *See, e.g.*, 2:33-39 ("Referring now to FIG. 1, a multiprocessor system 10 according to the invention is shown to include a ring 24 including a plurality of nodes, shown here labelled as elements 12, 14, 16, 18, 20, and 22 respectively. The plurality of nodes include a plurality of processing systems, 12, 14, 16, and 18 coupled to provide point to point connection between neighboring nodes on the ring 24.")

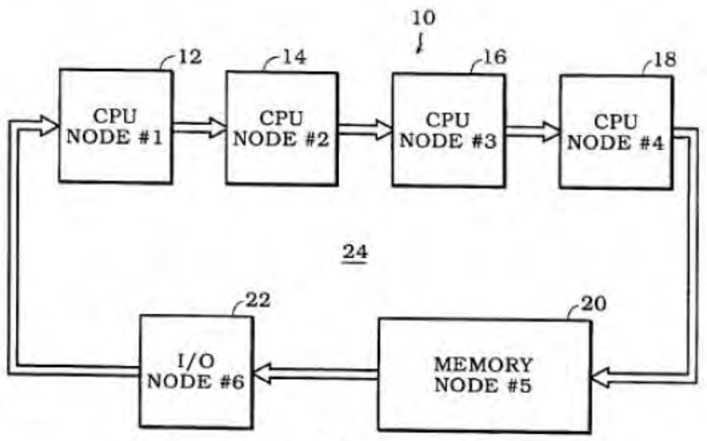


FIG. 1

US 5,551,048, Figure 1

- U.S. Patent No. 5,908,468 to Hartmann: *See, e.g.,* 4:46-50 ("Referring now to FIG. 2, an embodiment is shown of computer chip 100 with a data transfer network utilizing a multiple circular topology for interconnecting a plurality of modules 210A-210H on a single computer chip 100 in an on-chip network.")

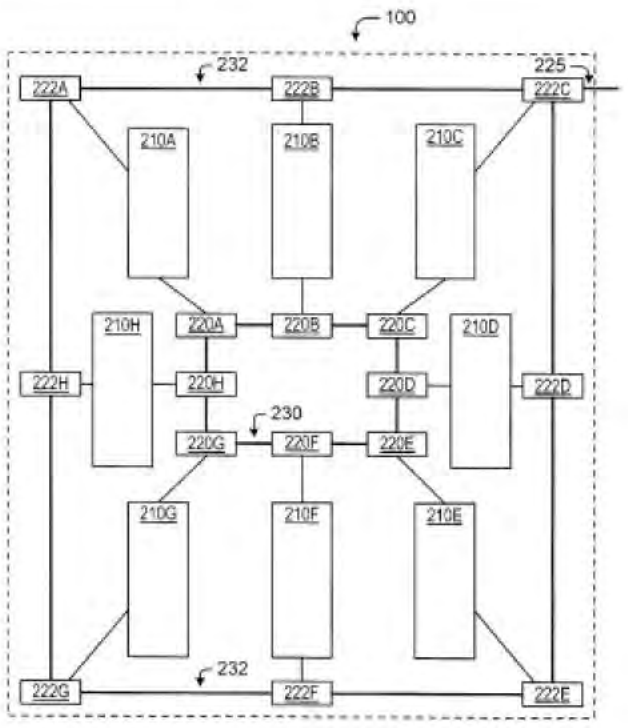


FIG. 2

Hartmann '468, Figure 2

- U.S. Patent No. 5,974,487 to Hartmann: *See, e.g.,* 3:24-28 ("Referring now to FIG. 2A, an embodiment is shown of computer chip 100 with a data transfer network utilizing a mesh of rings topology for interconnecting a plurality of modules 210A-210I on a single computer chip 100 in an on-chip network.")

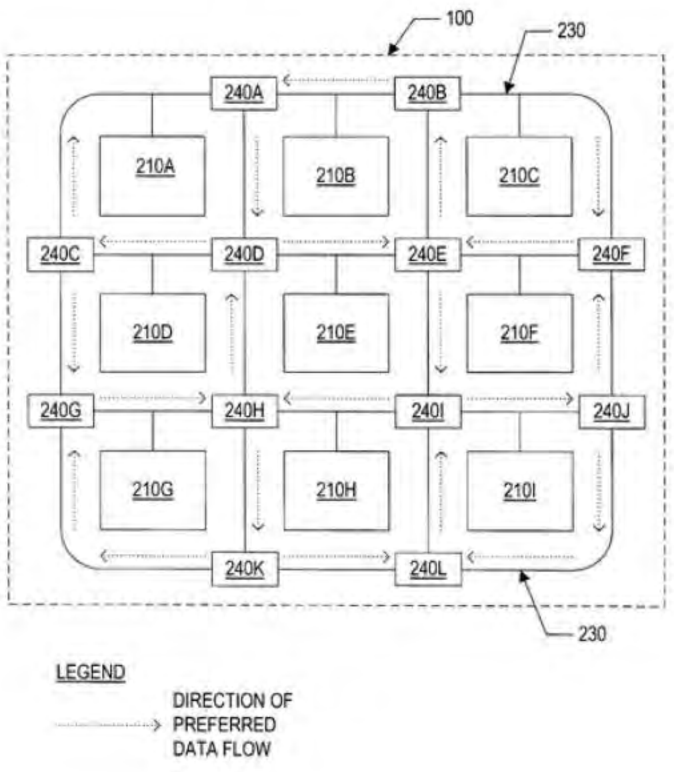


FIG. 2A

Hartmann '487, Figure 2A

- U.S. Patent No. 6,111,859 to Godfrey: *See, e.g.,* 1:57-62 ("Furthermore, the plurality of communication ports are further interconnected in a ring topology forming a circular bus or a semi-circular bus, wherein at least a subset of the plurality of communication ports are operable to transmit and receive data on the circular bus or semi-circular bus.")

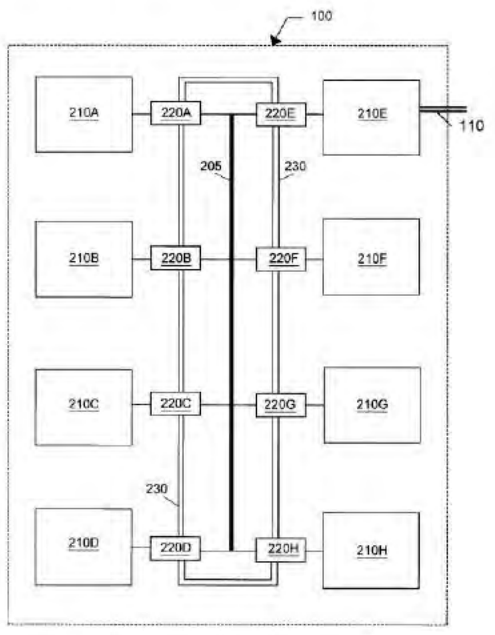


FIG. 2A

US 6,111,859, Figure 2A

- U.S. Patent No. 6,112,283 to Neiger: *See, e.g.,* 2:39-42 ("Referring to FIG. 1, an exemplary computer system 10 includes nodes N0, N1, N2, and N3. The nodes communicate with each other through a point-to-point ring topology rather than a shared bus.")

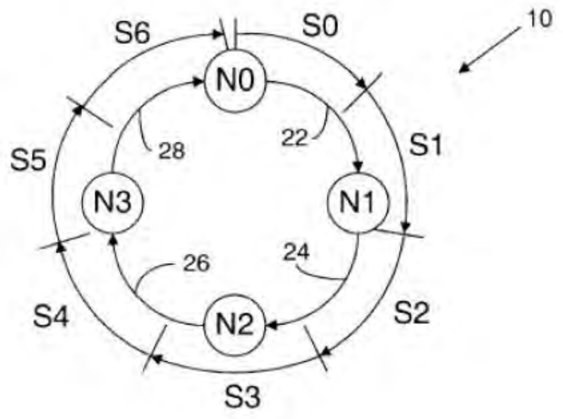


FIG. 1

Neiger, Figure 1

- U.S. Patent No. 6,839,808 to Gruner: *See, e.g.,* 4:1-11 ("In one embodiment, as shown in FIG. 1, clusters 12, 14, 16, and 18 are coupled to global snoop controller 22 via point-to-point connections 13, 15, 17, and 19, respectively. A snoop ring includes coupling segments $21_{1\text{-}4}$, which will be collectively referred to as snoop ring 21. Segment 21_1 couples global snoop controller 22 to cluster 18. Segment 21 2 couples cluster 18 to cluster 12. Segment 21_3 couples cluster 12 to cluster 14. Segment 21 $_4$ couples cluster 14 to cluster 16. The interaction between global snoop controller 22 and clusters 12, 14, 16, and 18 will be described below in greater detail.")

- Gruner: *See, e.g.,* 4:12-18 ("Global snoop controller 22 initiates accesses to main memory 26 through external bus logic (EBL) 24, which couples snoop controller 22 and clusters 12, 14, 16, and 18 to main memory 26. EBL 24 transfers data between main memory 26 and clusters 12, 14, 16, and 18 at the direction of global snoop controller 22. EBL 24 is coupled to receive memory transfer instructions from global snoop controller 22 over point-to-point link 11.")

- Gruner: *See, e.g.,* 4:19-30 ("EBL 24 and processing clusters 12, 14, 16, and 18 exchange data with each other over a logical data ring. In one embodiment of the invention, MPU 10 implements the data ring through a set of point-to-point connections. The data ring is schematically represented in FIG. 1 as coupling segments $20_{1\text{-}5}$ and will be referred to as data ring 20. Segment 20_1 couples cluster 18 to cluster 12. Segment 20_2 couples cluster 12 to cluster 14. Segment 20_3 couples cluster 14 to cluster 16. Segment 20_4 couples cluster 16 to EBL 24, and segment 20_5 couples EBL 24 to cluster 18. Further details regarding the operation of data ring 20 and EBL 24 appear below.")

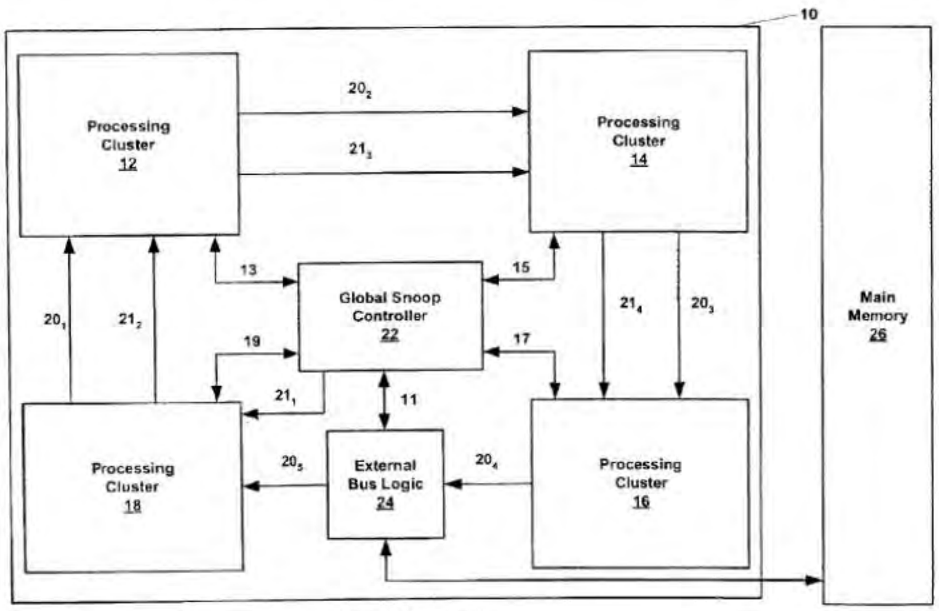


Figure 1

Gruner, Figure 1

b) Using a Crossbar Switch to Connect Components in a Multiprocessor System was Known

Public Version – Confidential Information Redacted and Confidentiality Designation Removed Per Agreement Between the Parties

Under Memory Integrity's apparent infringement theories, a crossbar switch appears to correspond to a "point-to-point architecture." However, connecting processing components via a crossbar switch was well known before the priority dates of the Asserted Patents. Examples of prior art references that disclose a crossbar switch and further demonstrate that such a structure was well known include:

- Culler: *See*, *e.g.*, p. 30 ("Therefore, these systems were typically organized with a crossbar switch connecting the CPU and several I/O channels to several memory banks, as indicated by Figure 1.16a. Adding processors was primarily a matter of expanding the switch; the hardware structure to access a memory location from a port on the processor and I/O side of the switch was unchanged.")

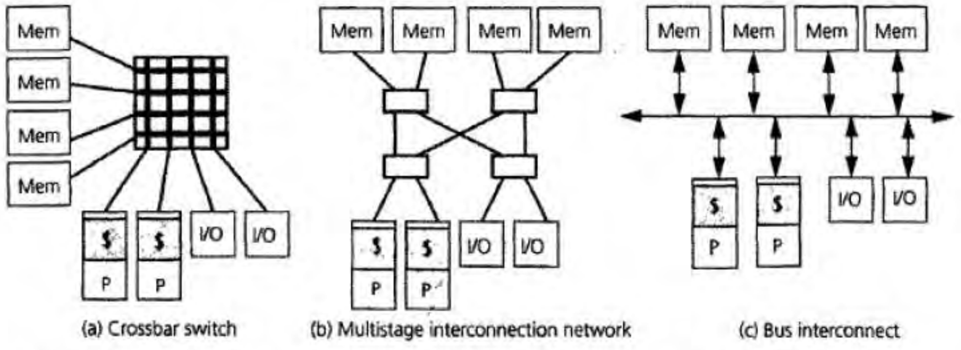


FIGURE 1.16 Typical shared memory multiprocessor interconnection schemes. The interconnection of multiple processors, with their local caches (indicated by $), and I/O controllers to multiple memory modules may be via crossbar, multistage interconnection network, or bus.

Culler, Figure 1.16

- Hwang: *See*, *e.g.*, 298 ("Upgrading bus architecture with crossbar switches or multistage networks will overcome these shortcomings to some extent, as studied in the next two subsections.")

- Hwang: *See*, *e.g.*, 298 ("**Crossbar Switches** Much higher bandwidth can be provided by a crossbar switched network for the same datapath width per port and equal number of connecting ports. A crossbar is a single-stage switched network.")

- Hwang: *See*, *e.g.*, 300 ("**Processor-Memory Crossbar Switch** A bus-connected multiprocessor is limited by its bus bandwidth. Very often, the bus becomes the bottleneck in accessing the shared memory by a large number of processors. A better approach is to replace the interprocessor memory bus by a crossbar switch as illustrated in Fig. 6.19.")

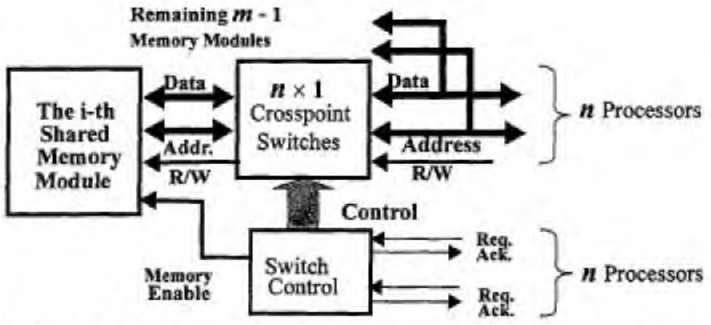
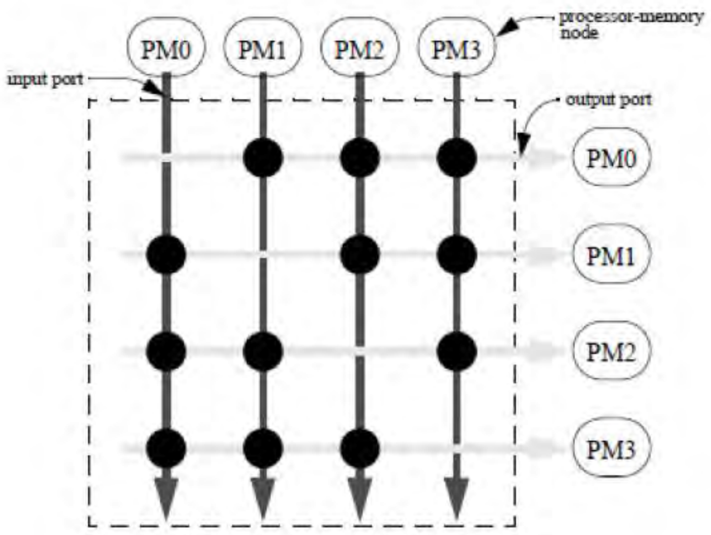


Figure 6.19 Partial circuitry of a $n \times m$ crossbar switch between n processors and m memory modules.

Hwang, Figure 6.19

- Barroso; *See, e.g.*, p. 11 ("Point-to-point links can also be used to connect nodes to switching elements, such as crossbars. Ideal crossbar switches are an example of a conflict-free network in the sense that it is possible for all nodes to communicate through the crossbar simultaneously, as long as every sender chooses a different destination. In other words, there can only be output conflicts.")
- Barroso; *See, e.g.*, p. 105 ("Crossbars have been considered as an interconnection for multiprocessors since the early days of parallel computing.")

Figure 7.1. Diagram of a Symmetric Crossbar for a NUMA system



Barroso, Figure 7.1

- "System Design of a CC-NUMA Multiprocessor Architecture using Formal Specification, Model-Checking, Co-Simulation, and Test Generation," Garavel (2000): *See, e.g.*, p. 7 ("Each module of Polykid is built simply by reusing an existing Bull multiprocessor machine already available (named Pegasus), in a slightly modified version with a reduced number of processors. A Polykid module consists of 4 nodes connected by a data bus and an address bus. The data is implemented as a crossbar, i.e., a 4×4 matrix of switches allowing every processor to communicate directly with every other processor, thus providing a higher bandwidth than with a standard bus. Amongst the 4 nodes, two are dedicated to the processors, one is dedicated to the main memory and one is dedicated to Pci bus management and input/output devices.")

- "Advances in Computers," Yovits (1992): *See, e.g.*, p. 136-138 ("Figure 17 illustrates some major alternatives for connecting multiple processors to shared memory outlined below. *4.2.1 Bus Interconnections* Time-shared buses (Fig. 17a) offer a fairly simply and relatively inexpensive way to give multiple processors access to a shared memory. However, a single, time-shared bus can effectively accommodate only a moderate number of processors (4-20), since only one processor can access the bus at a given time … *4.2.2 Crossbar Interconnections* Crossbar interconnection technology uses a crossbar switch of n^2 crosspoints to connect n processors to n memories (Fig. 17b). ")

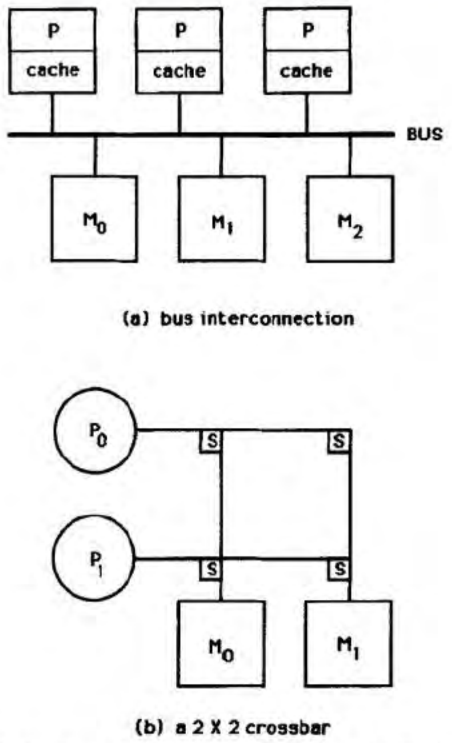
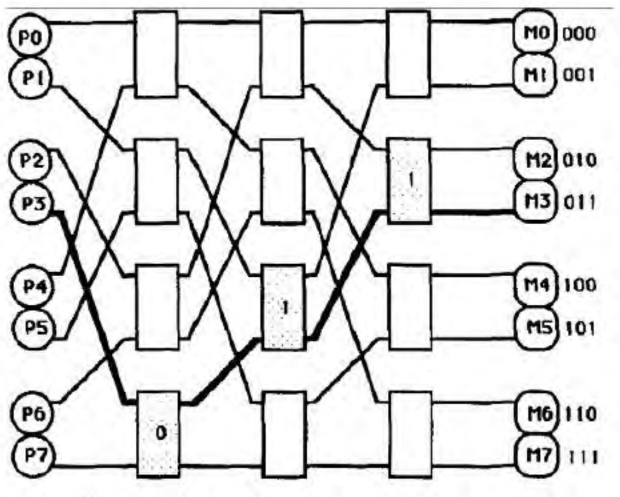


FIG. 17. MIMD shared memory interconnection schemes: (a) bus interconnection; (b) 2 × 2 crossbar; (c) 8 × 8 omega MIN routing a P_3 request to M_3. © 1990 IEEE.

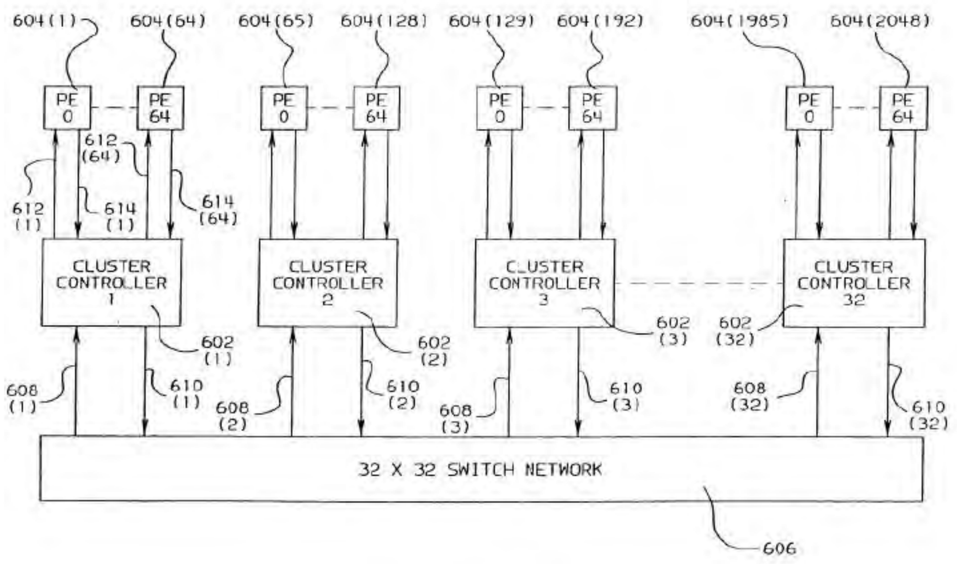


(c) an 8 X 8 omega MIN routing a P3 request to M3

Yovits, Figure 17

- U.S. Patent No. 5,166,674 to Baum: *See, e.g.*, 5:1-27 ("The switch that interconnects the processing elements is hierarchical, comprising a network of clusters. Up to 64 processing elements are combined to form a cluster, and up to 64 clusters are linked by way of a Banyan network. Messages are routed through the switch in the form of packets, each of which comprise a quadword of data and a word of control information." "FIG. 6 is a structural overview of the present multiprocessing system. The system of FIG. 6 is a cluster connected network (cluster network) comprising 32 cluster controllers 602(1)-602(32). Each cluster controller provides a system interface for 64 Processing Elements (PEs) 604(1-64), 604(65-128) . . . 604(1985-2048). Each group of one cluster controller and 64 processing elements is referred to as a 'cluster'." "Each processing element in a given cluster is connected to the cluster controller by way of an input bus (e.g. 612(1)) and an independent (separate) output bus (e.g. 614(1)). Similarly, the 32 cluster controllers are each connected to a 32×32 switch network 606 by way of an input bus 608(1-32) and an independent output bus 610(1-32). The entire system thus includes 2048 processing elements 604(1-2048). Both the cluster controllers and the switch network operate to assemble and transfer data between the processing elements synchronously, under control of a high speed clock (e.g. 5 ns cycle time).")

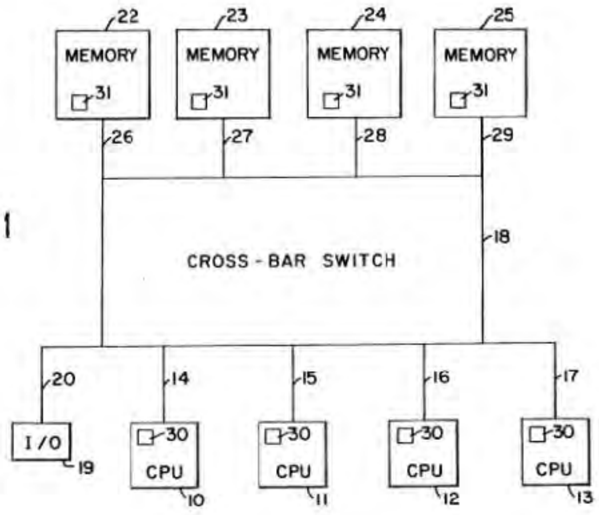
FIG.6



Baum, Figure 6

- U.S. Patent No. 5,297,269 to Donaldson: *See, e.g.*, 4:5-13 ("The present invention provides a multi-node system. The nodes may be, for example, several central processing units, each being connected to one or more memories by a memory/node coupling mechanism. The memory/node coupling mechanism may be a cross bar switch unit coupled point-to-point to one or more main memory modules and nodes and wherein each node (central processing unit or other devices) of the system has a write-back cache.")

FIG. 1



US 5,297,269, Figure 1

- U.S. Patent No. 5,900,015 to Herger: *See, e.g.*, 4:13-24 ("FIG. 1 shows a block diagram of a system according to the present invention. As depicted, the system includes multiple processors (labelled 'P') 13-1 to 13-8 connected through multiple interconnects 1-1 to 1-3 to a memory 9. The present invention is scalable to a large number of processors because it uses a distributed directory coherency scheme which does not require separate entries for each processor associated cache." "According to the present invention, individual coherency directories (labelled 'CD') 11 are associated with the different interconnects in the system. An interconnect may use a switch, a bus, a ring, or any other means for routing traffic between two or more external interconnect ports 3, 4 and 5.")

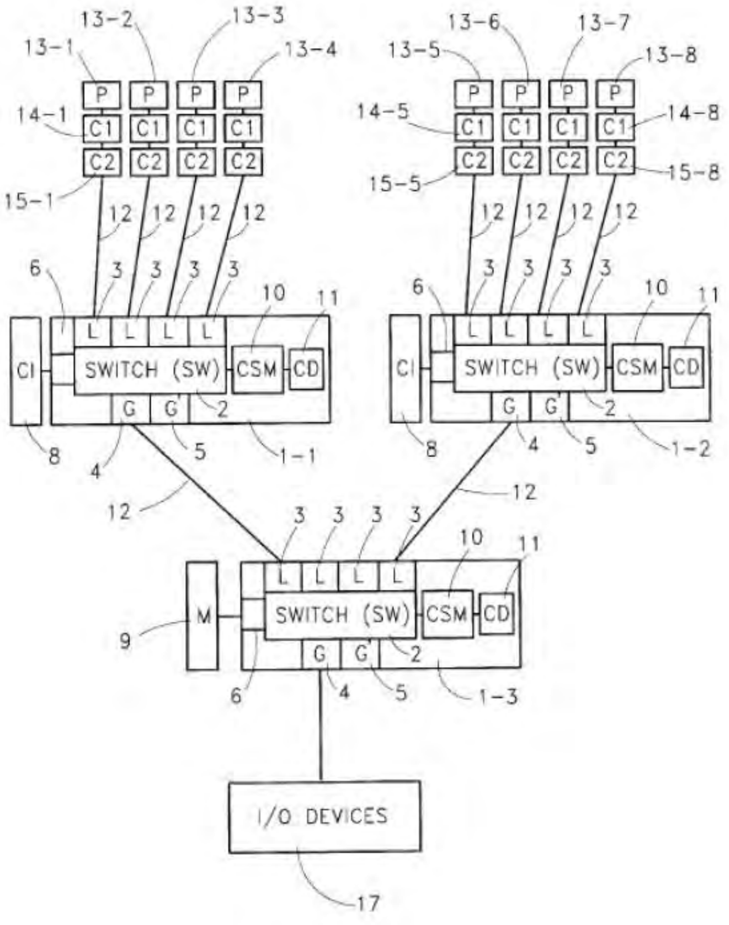
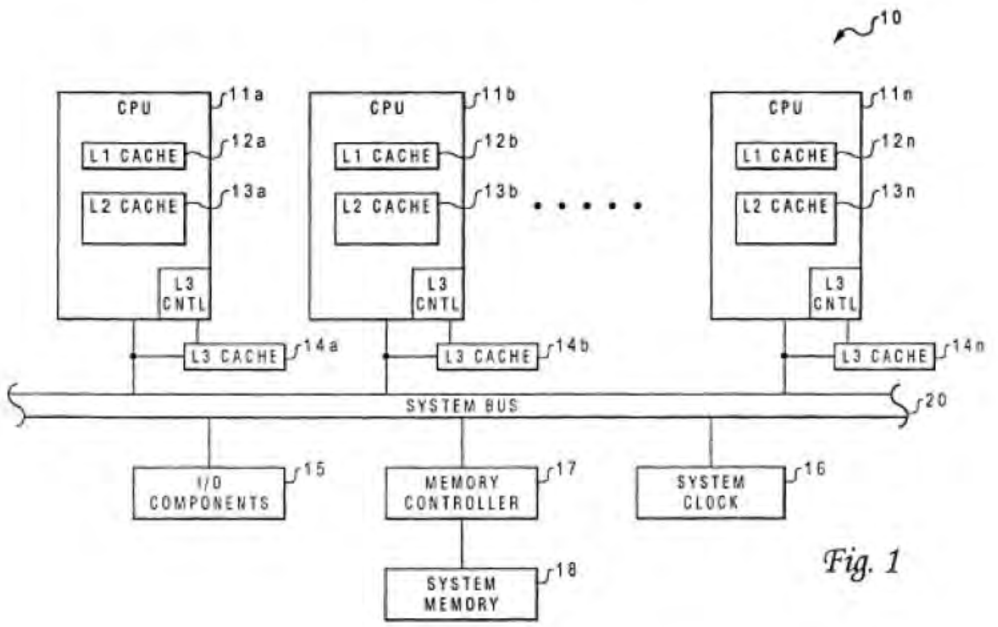


FIG. 1

Herger, Figure 1

- U.S. Patent No. 6,038,642 to Arimilli: *See, e.g.*, 2:55-63 ("All CPUs 11a-11n are coupled to an interconnect such as a system bus 20. For enhanced scalability, the interconnect may also be implemented by utilizing a cross-bar switch. A system clock 16 is coupled to system bus 20 for supplying clock signals to all the components within SMP data-processing system 10, including I/O components 15. Memory controller 17 is coupled to a system memory 18, which contains various instructions and data for the normal operations of SMP data-processing system 10.")



Arimilli '642, Figure 1

- U.S. Patent No. 6,018,782 to Hartmann: *See, e.g.,* 3:66-4:2 ("Referring now to FIG. 2, an embodiment is shown of computer chip 100 with an on-chip system for interconnecting a plurality of modules 210A-210G on a single computer chip 100 in an on-chip network."); *See, e.g.,* 4:18-19 ("An inter-module network switch 240 is comprised on computer chip 100. Which joins the inter-module links 230.")

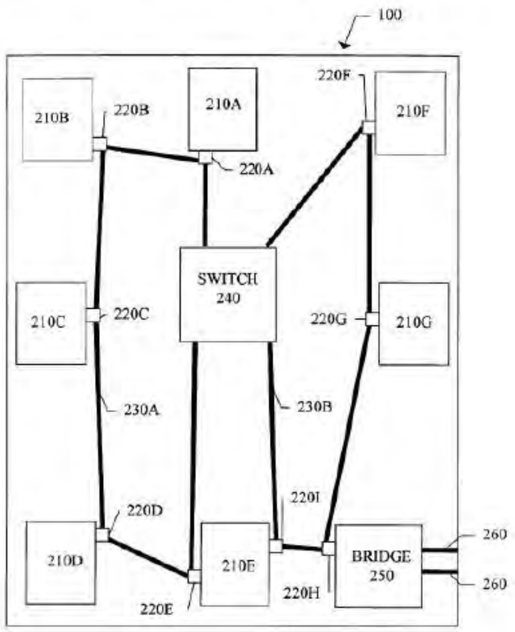
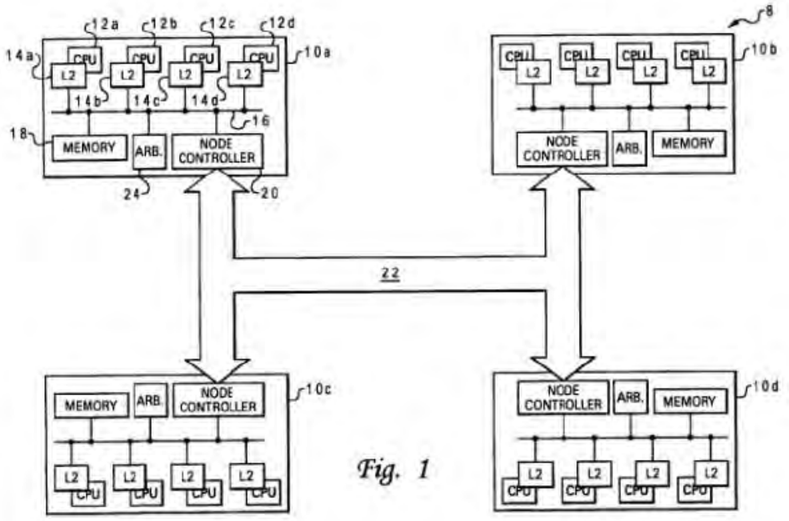


FIG. 2

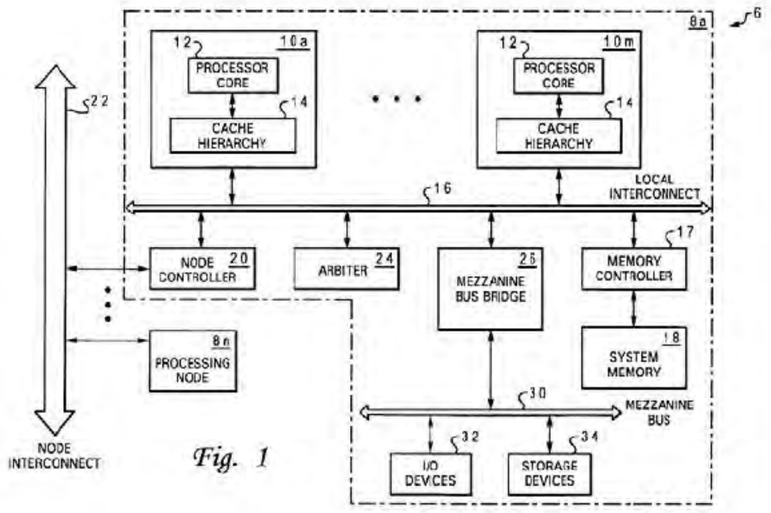
Hartmann '782, Figure 2

- U.S. Patent No. 6,067,603 to Carpenter: *See, e.g.*, 4:49-51 ("Local interconnects 16 and node interconnect 22 can each be implemented with any broadcast or point-to-point interconnect architectures, for example, a bus or cross-bar.")



Carpenter, Figure 1

- U.S. Patent No. 6,338,122 to Baumgartner: *See, e.g.*, 4:41-44 ("Local interconnects 16 and node interconnect 22 can each be implemented with any bus-based broadcast architecture, switch-based broadcast architecture, or switch-based non-broadcast architecture.")



US 6,546,429, Figure 1

- U.S. Patent No. 6,378,029 to Venkitakrishnan: *See, e.g.*, 5:46-49 ("In order to provide the point to point communication links among the SCU, the local processors, and the local memory units, the SCUA and the SCUDs contain one crossbar switch each, as shown in FIG. 2.")

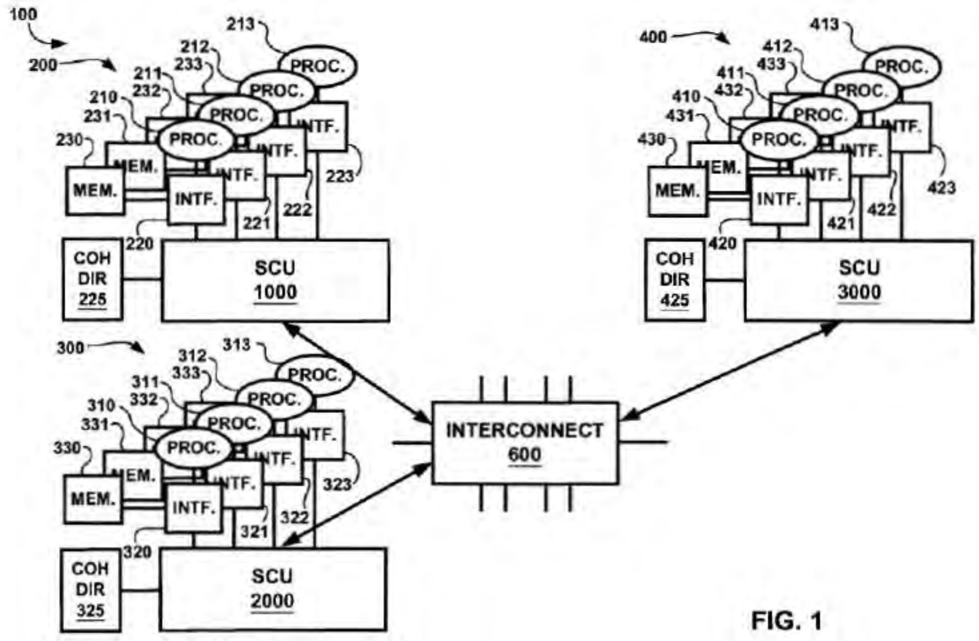


FIG. 1

Venkitakrishnan, Figure 1

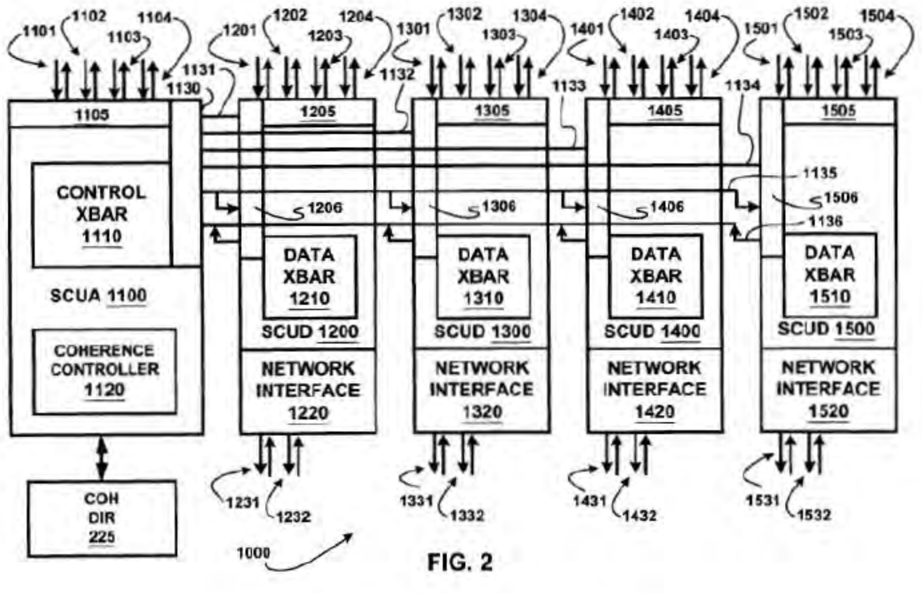
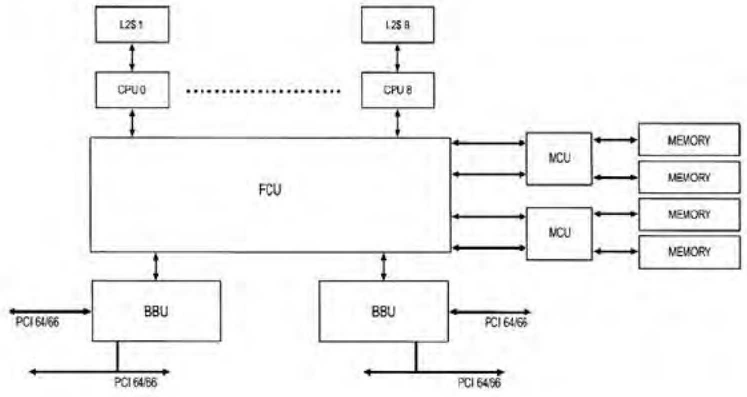


FIG. 2

Venkitakrishnan, Figure 2

- U.S. Patent No. 6,633,945 to Fu: *See, e.g.*, 2:66-3-2 ("The FCU internally implements a switched-fabric data path architecture. Point-to-Point (PP) interconnect 112, 113, and 114 and an associated protocol define dedicated communication channels for all FCU I/O.")



Fu, Figure 1

- U.S. Patent No. 6,760,819 to Dhong: *See, e.g.*, 4:16-22 ("Unlike the current processor configuration of FIG. 1A, in which the L1 caches 105A-105D are interconnected to each other and to the L2 cache 109 via multiple cache coherency busses 111, the illustrated processor configuration provides a single (point-to-point) bus connection running from each L1 cache 205A-205D to the L2 cache 209."; *see, e.g.,* 4:42-44 "L2 cache 209 may also be connected to a system memory via a system-level bus or switch.")

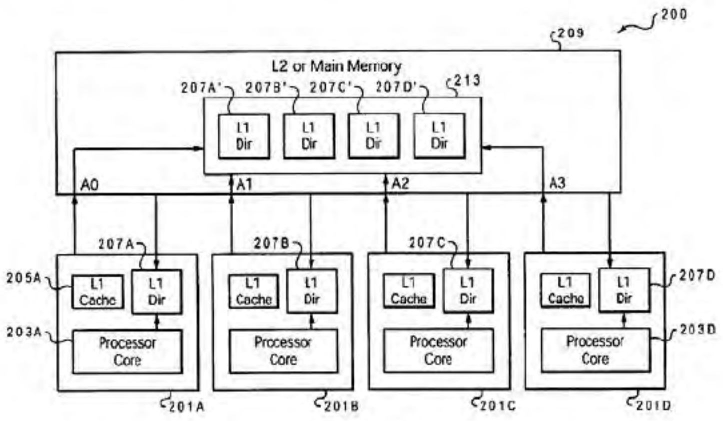


Fig. 2

US 6,760,819, Figure 2

- U.S. Patent No. 6,799,252 to Bauman: *See, e.g.,* 6:31-47 ("FIG. 1 shows one embodiment of multiprocessor computer system 100 of the present invention having one or more node clusters 170, each node cluster 170 having zero to N processors 74, zero to M memories 77, and zero to I input/output (I/O) subsystems 79. Depending on the needs of a user, interconnection network 175 can be set up as a three-dimensional torus, an N-dimensional hypercube, or any other suitable interconnection network between routers 76. In one embodiment, each router 76 includes eight ports 211, wherein each port 211 can be used to either connect to other routers 76, or to one to N node controllers 75 each having zero or more processor elements (PEs) 74. Thus, in some embodiments, a router 76 can be used as just an interconnection node in the network 175 (i.e., a circuit within block 175 rather than within node cluster 170), having no PEs 74 or memory 77 or I/O subsystems 79, and all of its ports are used to connect to other routers 76.")

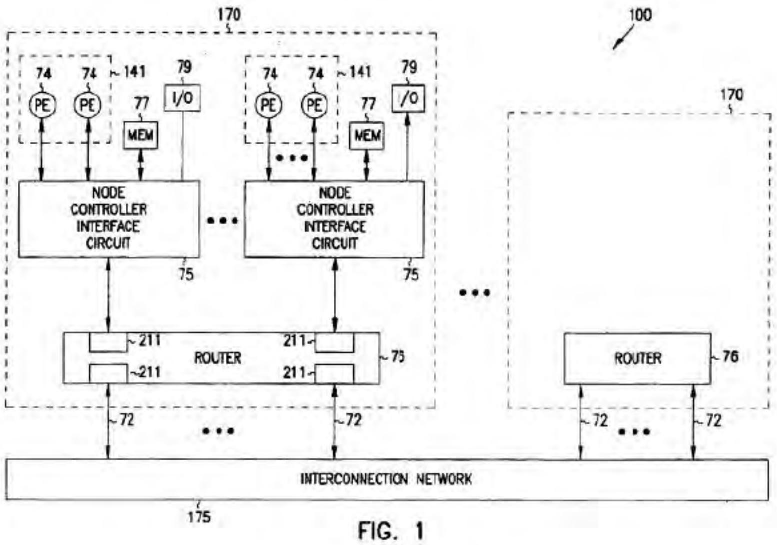
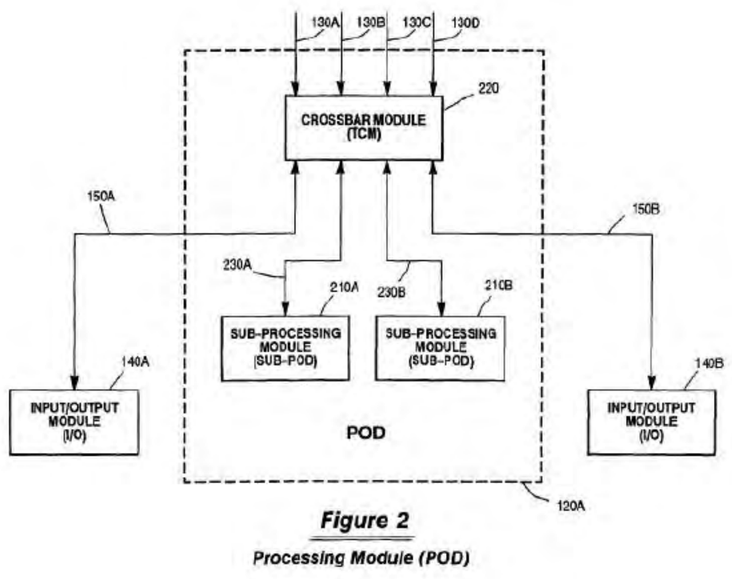


FIG. 1

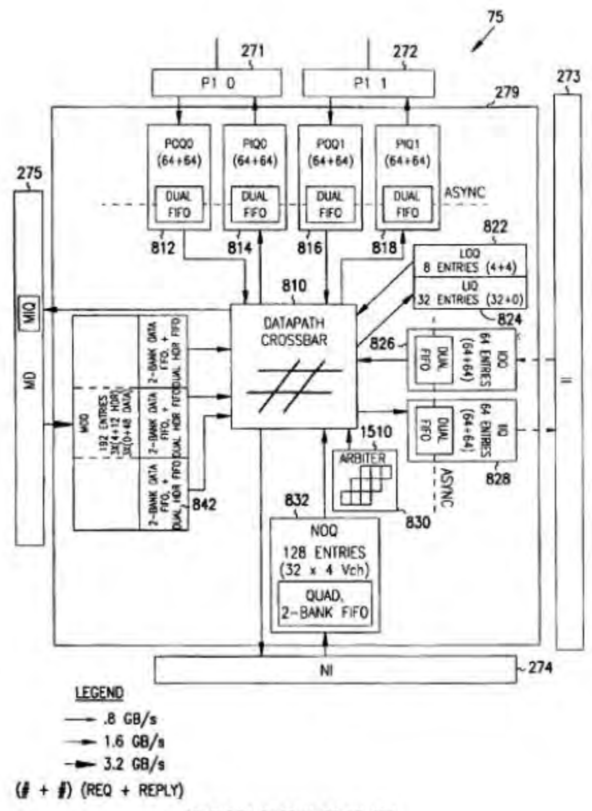
Bauman, Figure 1

- Bauman: *See, e.g.,* 7:28-36 ("FIG. 2 is a block diagram of a processing module (POD) according to one embodiment of the present invention. POD 120A is shown, but each of the PODs 120A through 120D have a similar configuration. POD 120A includes two Sub-Processing Modules (Sub-PODs) 210A and 210B. Each of the Sub-PODs 210A and 210B are interconnected to a Crossbar Module (TCM) 220 through dedicated point-to-point Interfaces 230A and 230B, respectively, that are similar to the MI interconnections 130.")


130A
130B
130C
130D
220
CROSSBAR MODULE (TCM)
150A
150B
230A
210A
230B
210B
SUB-PROCESSING MODULE (SUB-POD)
SUB-PROCESSING MODULE (SUB-POD)
140A
140B
INPUT/OUTPUT MODULE (I/O)
INPUT/OUTPUT MODULE (I/O)
POD
120A
Figure 2
Processing Module (POD)

Bauman, Figure 2

- U.S. Patent No. 6,751,698 to Deneroff: *See, e.g.*, 5:14-34 ("The crossbar unit (XB) 279 of node controller 75 provides connectivity between the two PI 270s, the MD 275, II 273, LB, and NI 274 units in a fair and efficient manner. The XB 279 supports the flow of messages in Bedrock internal format along two virtual channels, multiplexed across the physical channel(s) connecting each unit to the XB 279. The XB 279 is designed for minimal latency under light loads by means of buffer/queue bypass paths and arbitration hints, and maximum throughput under heavy loads by means of per virtual channel arbitration requests and a wavefront arbiter. Message ordering between each pair of units is maintained within each virtual channel. Messages targeting different destination units from a single source virtual channel may be transmitted in any order. Messages along different virtual channels may be interleaved across an interface or along a physical channel at the flit1 level.")



XB BLOCK DIAGRAM OF XB 279

FIG. 8

Deneroff, Figure 8

- U.S. Patent No. 6,751,721 to Webb: *See, e.g.*, 5:14-34 ("Referring now to FIG. 1, in accordance with the preferred embodiment of the invention, computer system 90 comprises one or more processors 100 coupled to a memory 102 and an input/output ('I/O') controller 104. As shown, computer system 90 includes twelve processors 100, each processor coupled to a memory and an I/O controller. Each processor preferably includes four ports for connection to adjacent processors. The interprocessor ports are designated 'North,' 'South,' 'East,' and 'West' in accordance with the well-known Manhattan grid architecture also known as a crossbar interconnection network architecture. As such, each processor 100 can be connected to four other processors. The processors on both ends of the system layout wrap around and connect to processors on the opposite side to implement a 2D torus-type connection. Although twelve processors 100 are shown in the exemplary embodiment of FIG. 1, any desired number of processors (e.g., 256) can be included. For purposes of the following discussion, the processor in the upper, left-hand comer of FIG. 1 will be discussed with the understanding that the other processors 100 are similarly configured in the preferred embodiment.")

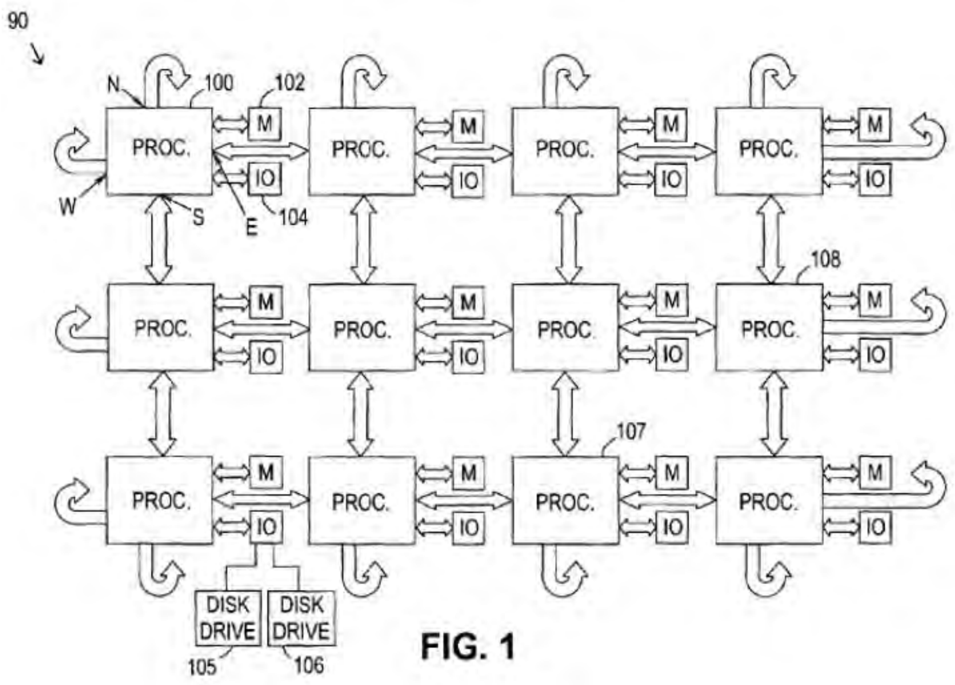


FIG. 1

Webb, Figure 1

- Webb: *See, e.g.*, 13:36-40 ("FIG. 3 represents a multi-processor system configured to use the hybrid invalidate scheme. Whereas the system shown in FIG. 1 comprises 12 processors 100, the system shown in FIG. 3 comprises 12 clumps or clusters 300. Within each cluster, there are four processors 310.")

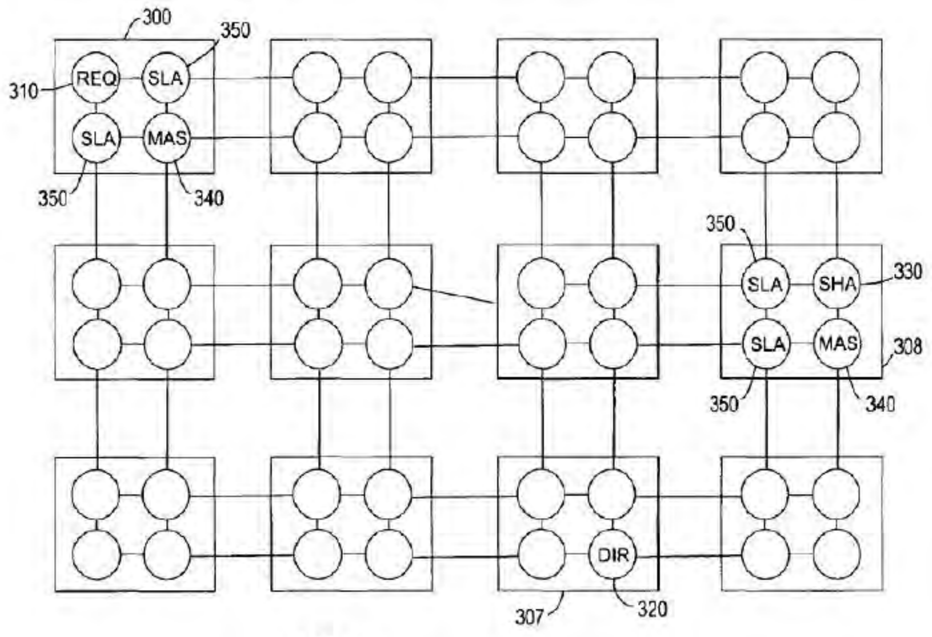


FIG. 3

Public Version – Confidential Information Redacted and Confidentiality Designation Removed Per Agreement Between the Parties

Webb, Figure 3

- U.S. Patent No. 7,017,011 to Lesmanne: *See, e.g.*, 5:6-11 ("By way of a nonlimiting example and in order to simplify the description, each module 50–53 is constituted by n=4 sets of basic multiprocessors 60–63 MP0–MP3, respectively linked to a coherence controller 64 SW (Switch) by two-point high-speed links 70–73 controlled by four control units PU0, PU1, PU2, PU3 80–83 of local ports 90–93.")

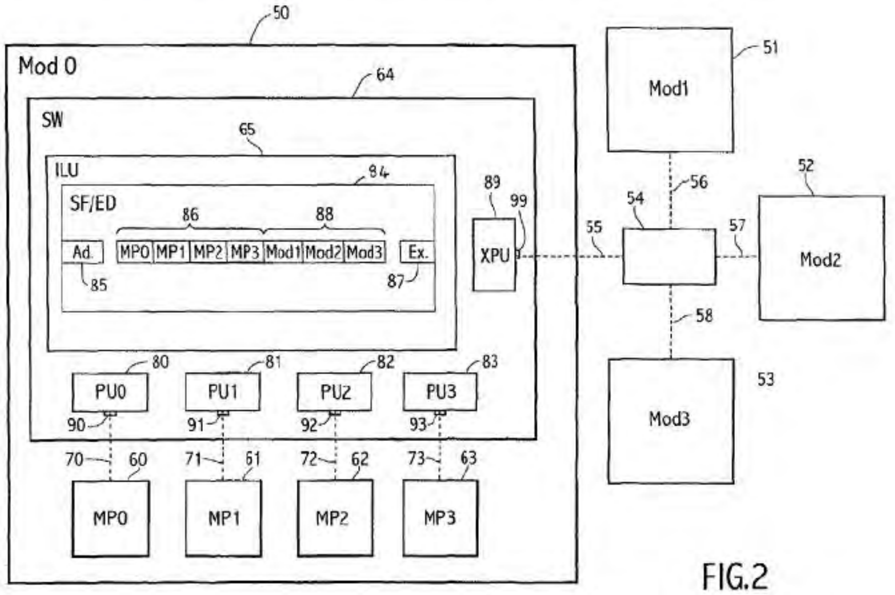
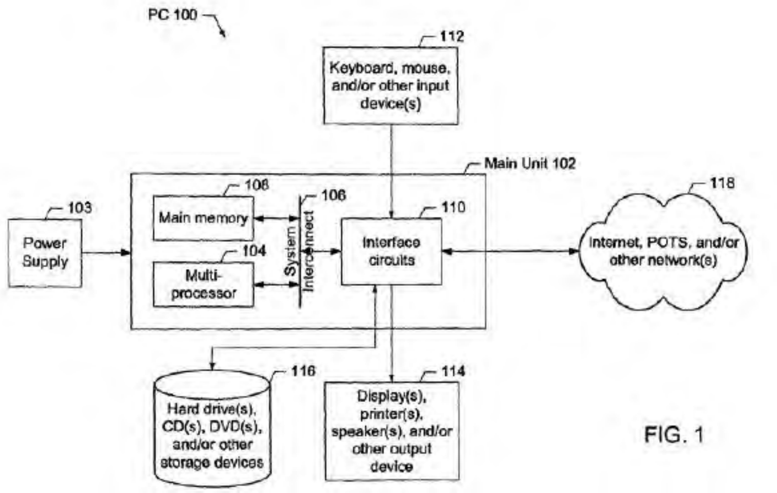


FIG.2

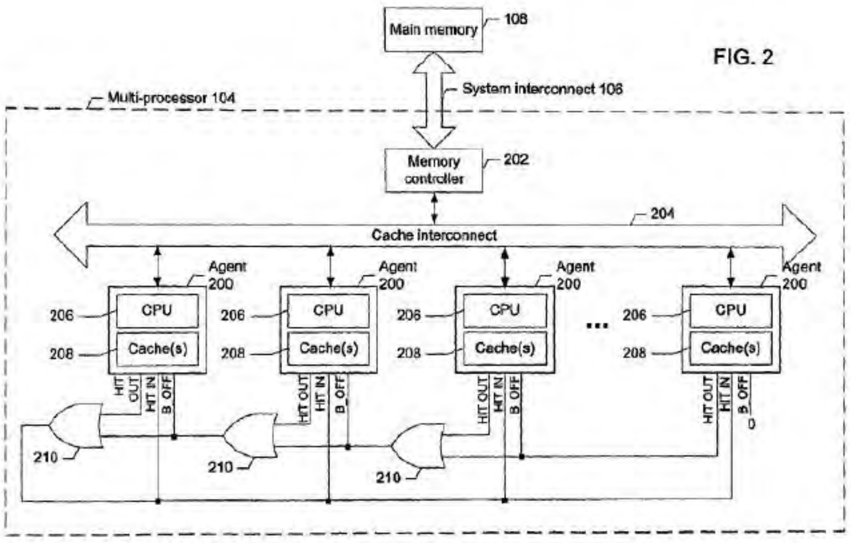
Lesmanne, Figure 2

- U.S. Patent No. 7,100,001 to Edirisooriya: *See, e.g.*, 2:28-35 ("The main processing unit 102 may include a multi-processor unit 104 electrically coupled by a system interconnect 106 to a main memory device 108 and one or more interface circuits 110. For one embodiment, the system interconnect 106 is a address/data bus. Of course, a person of ordinary skill in the art will readily appreciate that interconnects other than busses may be used to connect the multi-processor unit 104 to the main memory device 108. For example, one or more dedicated lines and/or a crossbar may be used to connect the multi-processor unit 104 to the main memory device 108.")


PC 100
112
Keyboard, mouse, and/or other input device(s)
Main Unit 102
108
106
110
118
103
Main memory
System Interconnect
Power Supply
104
Multi-processor
Interface circuits
Internet, POTS, and/or other network(s)
116
114
Hard drive(s), CD(s), DVD(s), and/or other storage devices
Display(s), printer(s), speaker(s), and/or other output device
FIG. 1

Edirisooriya, Figure 1

- Edirisooriya: *See, e.g.*, 2:28-35 ("In the embodiment illustrated in FIG. 2, the multi-processor 104 includes a plurality of processing agents 200 and a memory controller 202 electrically coupled by a cache interconnect 204. The cache interconnect 204 may be any type of interconnect such as a bus, one or more dedicated lines, and/or a crossbar. Each of the components of the multi-processor 104 may be on the same chip or on separate chips.")



Edirisooriya, Figure 2

- U.S. Patent No. 7,120,755 to Jamil: *See, e.g.*, 3:14-35 ("In one embodiment, control logic 130 enables efficient cache coherency by reading one or more modified cache lines from a first dedicated cache and writing them back to a second dedicated cache so that the second dedicated cache will have the latest version of the associated data. Dedicated caches 121–123 are connected to control logic 130 through write-back lines 151–153, respectively. While in one embodiment, write-back lines 151–153 are unidirectional lines to transfer modified data from a dedicated cache to control logic 130, in another embodiment write-back lines 151–153 may be bi-directional. Write-back lines 151–153 maybe substantially independent of each other so that data may be transferred over more than one of the write-back lines 151–153 at the same time. Dedicated caches 121–123 are also connected to control logic 130 through supply lines 154–156, respectively. While in one embodiment, supply lines 154–156 are unidirectional lines to transfer modified data from control logic 130 to a dedicated cache, in another embodiment supply lines 154–156 may be bi-directional. Supply lines 154–156 maybe substantially independent of each other so that data may be transferred over more than one of supply line 154–156 at the same time.")

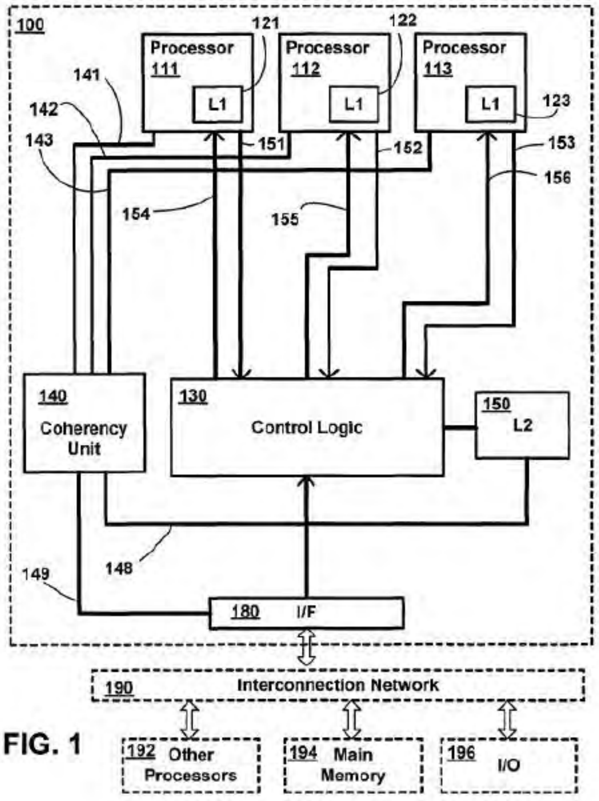


FIG. 1

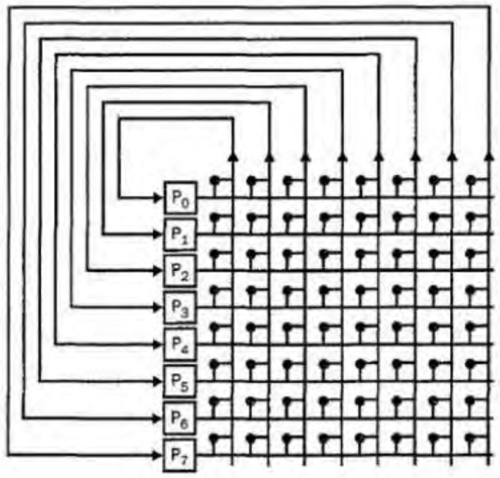
Jamil, Figure 1

c) Using a Fully Connected Crossbar Switch to Connect Components in a Multiprocessor System was Known

In addition to using a crossbar switch, using "fully connected" crossbar switches was well known before the priority dates of the Asserted Patents. Examples of prior art references that disclose a "fully connected" crossbar switch and further demonstrate that such a structure was well known include:

- Culler: *See, e.g.*, pgs. 768-769: ("A fully connected network is essentially a single switch, which connects all inputs to all outputs." "Another fully connected network is a crossbar. It provides O(N) bandwidth, but the cost of the interconnect is proportional to the number of cross-points, or O(N2).")

- "Computer Organization & Design," Patterson et al. (1998): *See, e.g.*, p. 736 ("The straightforward way to connect processor-memory nodes is to have a dedicated communication link between every node. Between the high cost/performance of this fully connected network and the low cost/performance of a bus are a set of networks that constitute a wide range of trade-offs in cost/performance."); p. 737 ("At the other extreme from a ring is a fully connected network, where every processor has a bidirectional link to every other processor."); p. 738 ("A fully connected or crossbar network allows any node to communicate with any other node in one pass through the network.")



a. Crossbar

Patterson, Figure 9.14(a)

- "Computer Organization," Hamacher (2001): *See, e.g.*, p. 625 ("A versatile switching arrangement is shown in Figure 12.5. It is known as the crossbar switch, which was originally developed for use in telephone networks." "Such networks, where there is a direct link between all pairs of nodes, are called *fully connected* networks.")

- U.S. Patent No. 5,197,130 to Chen: *See, e.g.*, 9:42-53 ("Referring now to FIG. 3, the preferred embodiment of the arbitration node means 20 for a single cluster 40 will be described. At a conceptual level, the arbitration node means 20 comprises a plurality of cross bar switch mechanisms that symmetrically interconnect the processors 10 and external interface means 22 to the shared resources 12 in the same cluster 40, and to the shared resources 12 in other clusters 40 through the remote cluster adapter means 42. Typically, a full cross bar switch would allow each requestor to connect to each resource where there are an equivalent number of requestors and resources.")

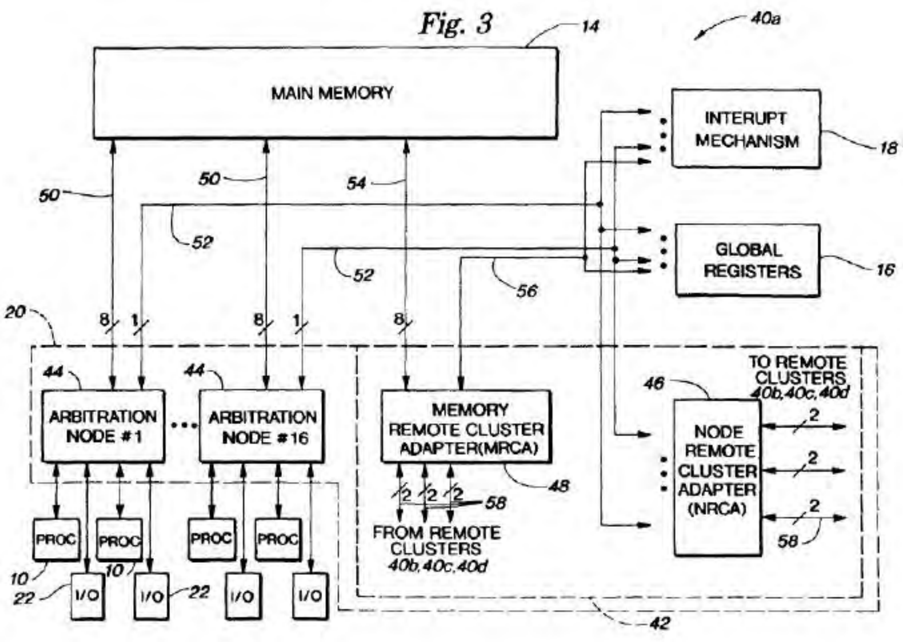


Fig. 3
MAIN MEMORY
INTERUPT MECHANISM
GLOBAL REGISTERS
ARBITRATION NODE #1
ARBITRATION NODE #16
MEMORY REMOTE CLUSTER ADAPTER(MRCA)
NODE REMOTE CLUSTER ADAPTER (NRCA)
TO REMOTE CLUSTERS 40b,40c,40d
FROM REMOTE CLUSTERS 40b,40c,40d
PROC
I/O
14
40a
18
16
50
52
54
56
20
44
46
48
58
10
22
42

Chen, Figure 3

d) **Using Point-to-Point Links to Connect Components in a Multiprocessor System was Known**

The use of point-to-point links to connect components in a multiprocessor system was also well known before the priority dates of the Asserted Patents. Indeed, the Asserted Patents acknowledge that a "point-to-point architecture" having, for example, processors "directly connected to each other through a plurality of point-to-point links to form a cluster of processors" was well known. *See, e.g.,* '409 patent, 2:30-35. Examples of prior art references that disclose the use of point-to-point links and further demonstrate that such a structure was well known include:

- Varma (1994): *See, e.g.*, p. 8 ("In a static network, point-to-point links interconnect the network nodes in some fixed topology; a regular topology such as a mesh or hypercube is common….These networks are also sometimes referred to as direct networks because the interconnecting links are directly connected to the network nodes, as opposed to being switched dynamically.")

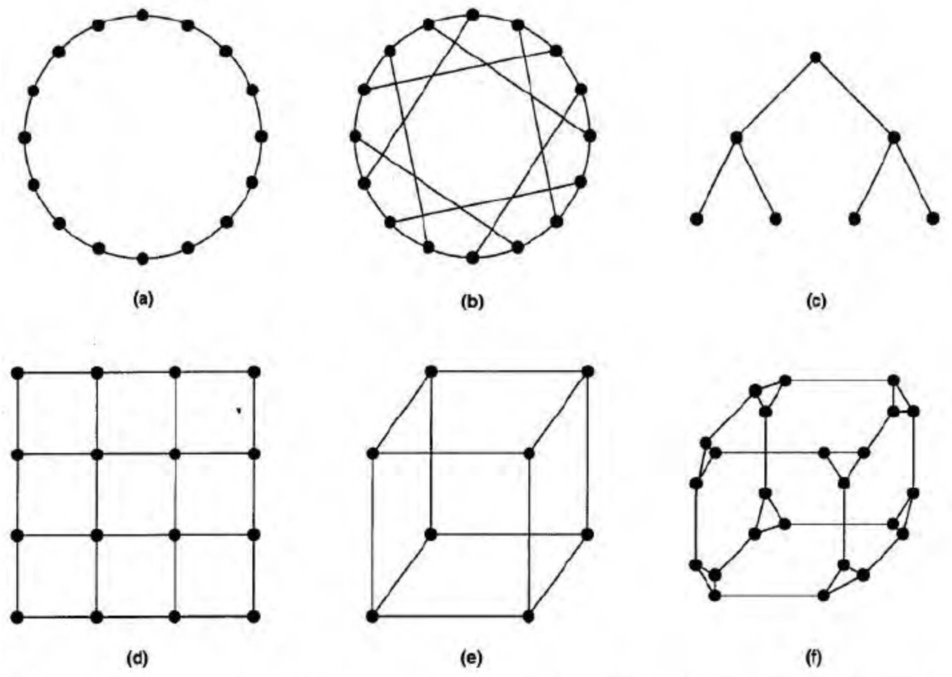


Figure 1: Some examples of static network topologies: (a) A simple ring (b) chordal ring (c) binary tree (d) 2-D mesh (e) binary 3-cube (3-dimensional hypercube) (f) cube-connected cycles.

Varna, p. 9

- "Issues in Designing Scalable Systems with *k*-ary *n*-cube cluster-*c* Organization," Panda (1994): *See, e.g.*, p. 5 ("Direct-network based clusters … In such configurations, processors in the cluster are connected by a point-to-point network.")

- Barroso: *See, e.g.*, p. 9 ("The alternatives to a bus interconnect are topologies that rely on a non-shared physical communication medium. All such topologies therefore use point-to-point links as building blocks. Point-to-point links have several attractive features and have been growing increasingly popular in the last few years. By having only a single driver and a single receiver at each end of the wire, point-to-point links are much simpler electrically than buses. Point-to-point links are easy to terminate properly because there is only one termination point. Better termination and lower characteristic impedances allow fast signal rise time and propagation speed with lower driving currents, which makes it easier to use low voltage signaling while still maintaining reasonable noise immunity.")

- Culler: *See, e.g.*, p. 271 ("The dancehall approach also places the interconnect between the caches and main memory, but the interconnect is now a scalable point-to-point network rather than a bus, and memory is divided into many logical modules that connect to logically different points in the interconnect.")

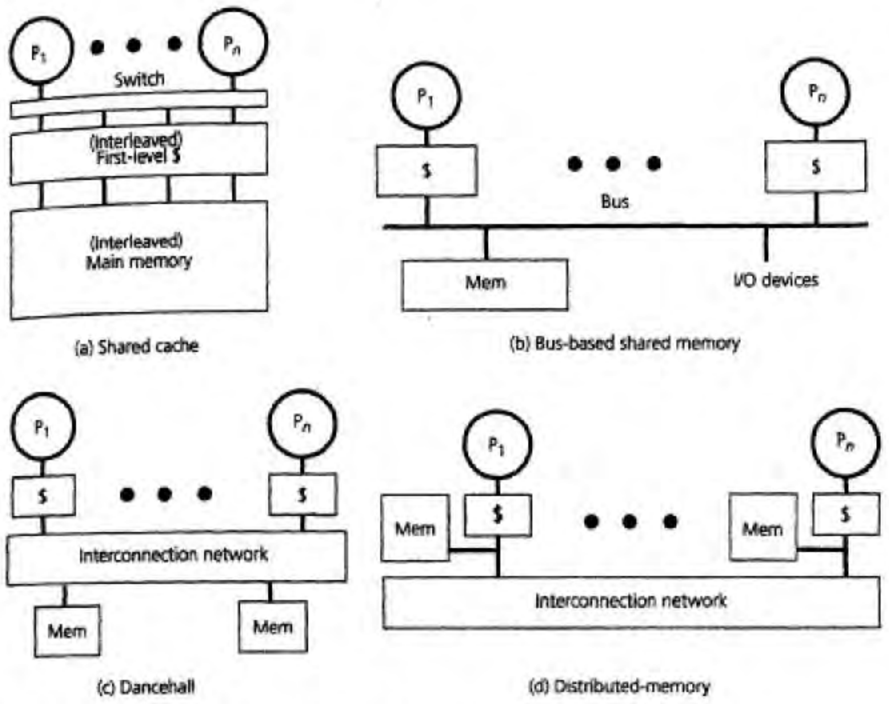


FIGURE 5.2 Common extended memory hierarchies found in multiprocessors

Culler, Figure 5.2

- Culler: *See*, *e.g.*, p. 553 ("We have also seen that in order to scale up machines, memory is distributed, a scalable point-to-point interconnection network is introduced, and a communication assist provides varying degrees of interpretation of network transactions to support programming models.")
- Culler: *See*, *e.g.*, p. 555 ("Since directory schemes rely on point-to-point network transactions, they can be used with any interconnection network. Important questions for directories include the form in which the directory information is stored and how correct, efficient protocols may be designed using these representations.")
- Hwang: *See*, *e.g.*, p. 18 ("For now, we briefly define two overhead metrics (first introduced by Hockney [313]) for one node to communicate a message to another node. Such a communication is called point-to-point. There is another type, called collective communication, where more than one messages are communicated among multiple nodes simultaneously.")
- Hwang: *See*, *e.g.*, p. 82 ("One-to-One: This is also known as point-to-point communication. There is one sender and one receiver, as illustrated in Fig. 2.10(a).")
- Hwang: *See*, *e.g.*, p. 244 ("The interconnection network is not restricted to a bus anymore. A multistage or crossbar switch; or any other point-to-point system-area network (SAN) or local-area network (LAN), will apply.")

- Hwang: *See, e.g.*, p. 278 ("Static networks are built with point-to-point links that will not change during program execution. In other words, static networks have fixed connections among processing nodes. Static networks are also known as direct networks, because only one host computer is connected to each node switch. On the other hand, a dynamic network is implemented with switched channels, which are dynamically configured to meet the communication requirements in user programs.")
- Hwang: *See, e.g.*, p. 286 ("Static networks use point-to-point links that are fixed once connected. This type of network often shares the media and is more suitable for predictable traffic patterns. These topologies can be also applied to build switched networks.")

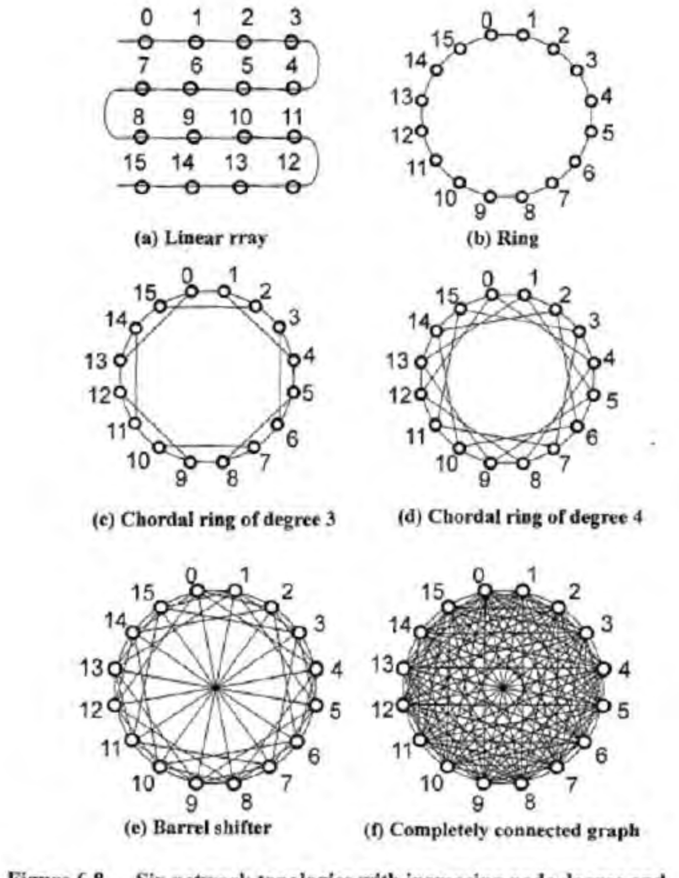
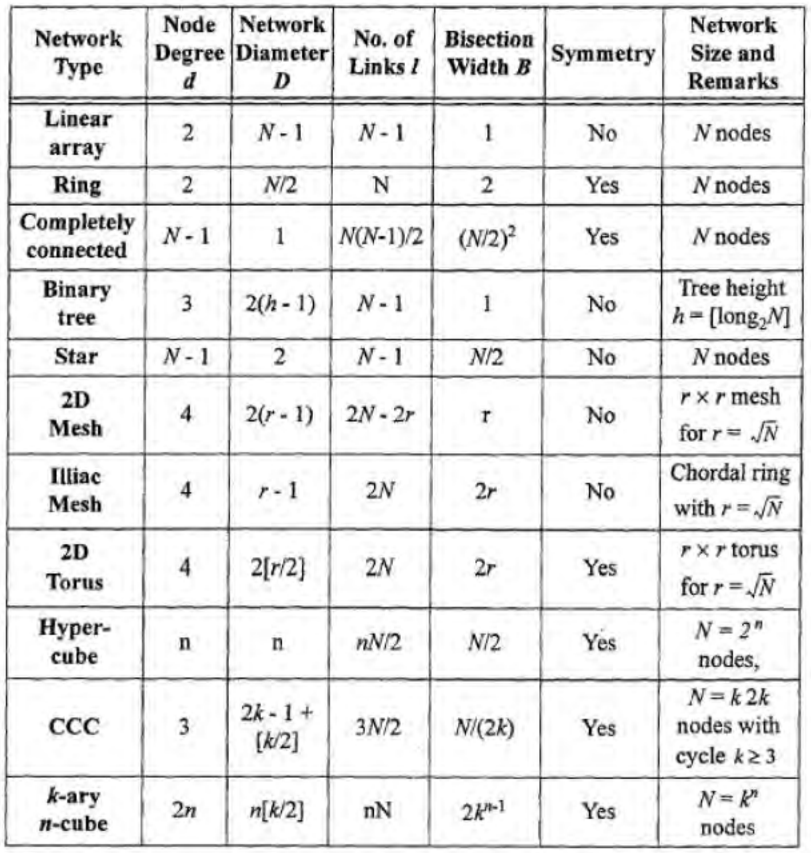

(a) Linear rray
(b) Ring
(c) Chordal ring of degree 3
(d) Chordal ring of degree 4
(e) Barrel shifter
(f) Completely connected graph

Figure 6.8 Six network topologies with increasing node degree and connectivity from a linear array to a ring, two chordal rings, a barrel shifter, and a completely connected graph.

Hwang, Figure 6.8

- Hwang, Table 6.1:

Table 6.1 Topological Properties of Static-Connection Networks



Network Type	Node Degree d	Network Diameter D	No. of Links l	Bisection Width B	Symmetry	Network Size and Remarks
Linear array	2	$N - 1$	$N - 1$	1	No	N nodes
Ring	2	$N/2$	N	2	Yes	N nodes
Completely connected	$N - 1$	1	$N(N-1)/2$	$(N/2)^2$	Yes	N nodes
Binary tree	3	$2(h - 1)$	$N - 1$	1	No	Tree height $h = [\text{long}_2 N]$
Star	$N - 1$	2	$N - 1$	$N/2$	No	N nodes
2D Mesh	4	$2(r - 1)$	$2N - 2r$	r	No	$r \times r$ mesh for $r = \sqrt{N}$
Illiac Mesh	4	$r - 1$	$2N$	$2r$	No	Chordal ring with $r = \sqrt{N}$
2D Torus	4	$2[r/2]$	$2N$	$2r$	Yes	$r \times r$ torus for $r = \sqrt{N}$
Hyper-cube	n	n	$nN/2$	$N/2$	Yes	$N = 2^n$ nodes,
CCC	3	$2k - 1 + [k/2]$	$3N/2$	$N/(2k)$	Yes	$N = k2k$ nodes with cycle $k \geq 3$
k-ary n-cube	$2n$	$n[k/2]$	nN	$2k^{n-1}$	Yes	$N = k^n$ nodes

Hwang, Table 6.1

- "Design and Performance of SMPs With Asynchronous Caches," Pong (1999): *See, e.g.*, p. 1 ("This design overcomes some of the scalability problem of a multi-drop shared-bus by employing high-speed point-to-point links, whose scalability prospects are much better than for shared buses.")

- Pong: *See, e.g.*, p. 5 ("We advocate an asynchronous cache system such as the one shown in Figure 3. In this design, processors and memory controller communicate via unidirectional high-speed links [10, 25]. A set of queues buffer requests and data blocks in case of access conflicts and also serve as adaptor between data paths of different width.")

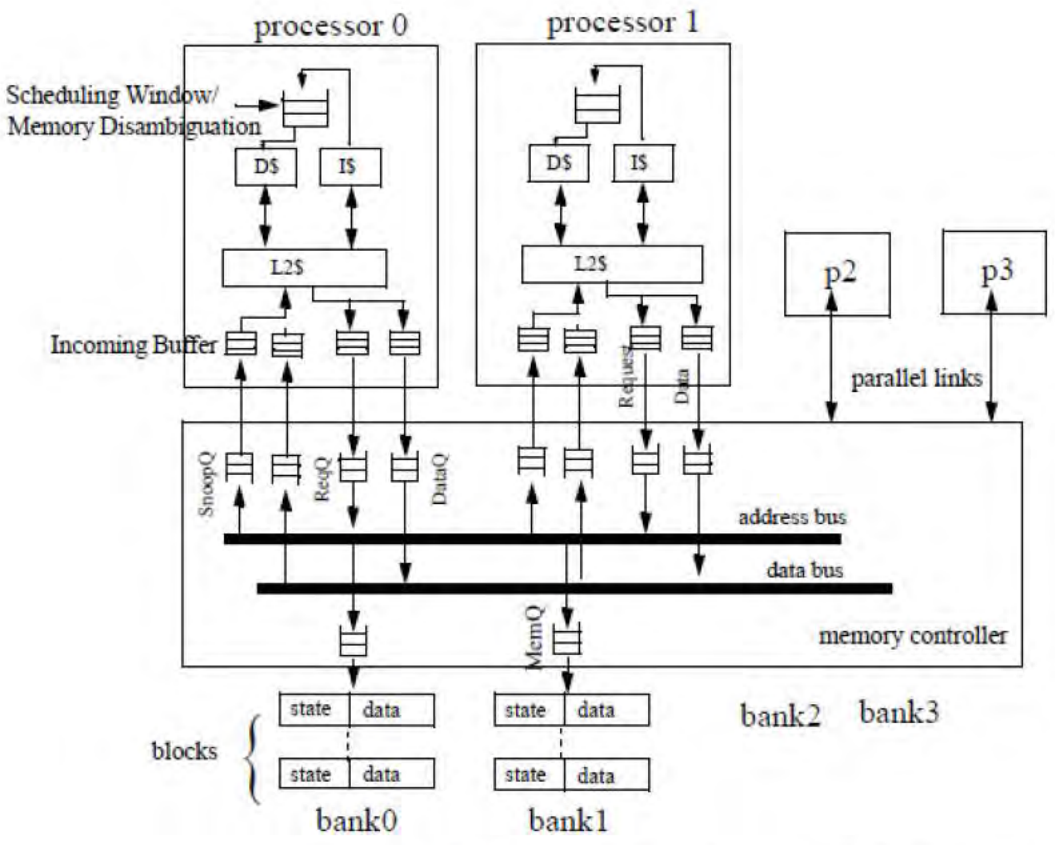


Figure 3 The Proposed Asynchronous Cache System.

Pong, Figure 3

- "Computer Architecture," the Computational Education Project (1995): *See*, *e.g.*, p. 42-43 ("A PMS diagram of a simple distributed memory parallel processor is shown in Figure 9. On the left is the diagram of a single node often called a processing element, or PE. The organization of a PE explains how messages are passed from one PE to another." "The following discussion of the properties of interconnection networks is based on a collection of nodes that communicate via links. In an actual system the nodes can be either processors, memories, or switches. Unless otherwise noted the links will always be point-to-point data paths, i.e., not buses that are shared by several nodes.")

Public Version – Confidential Information Redacted and Confidentiality Designation Removed Per Agreement Between the Parties

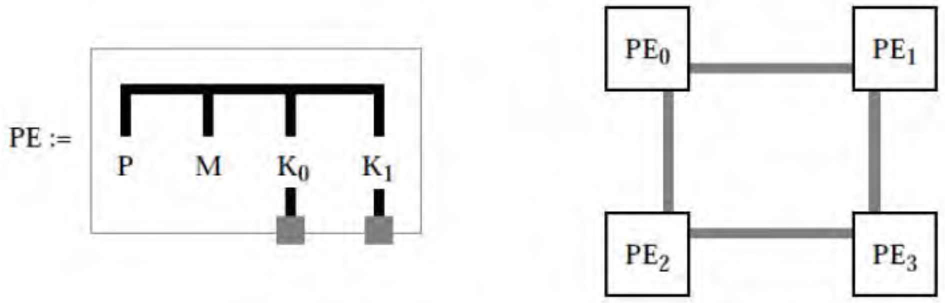


Figure 9: Distributed Memory Parallel Processor

Computational Education Project, Figure 9

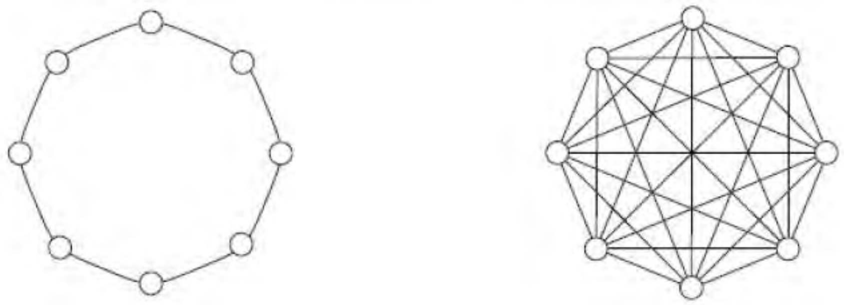


Figure 10: Ring vs. Fully Connected Network

Computational Education Project, Figure 10

- U.S. Patent No. 4,907,232 to Harper: *See, e.g.*, 3:43-55 ("As can be seen in the exemplary system of FIG. 1, a plurality of fault containment regions 10, shown in the example as four separate regions, each includes a plurality of processing elements 11, there being in this example four processing elements in each region. A network element 12 in each region is connected to each of the processing elements in its associated fault containment region and is, in turn, inter-connected with each of the other network elements of the other fault containment regions, as shown by the inter-connecting lines 12A. The use of such network elements and interconnections provides a fully connected, tightly synchronized overall system.")

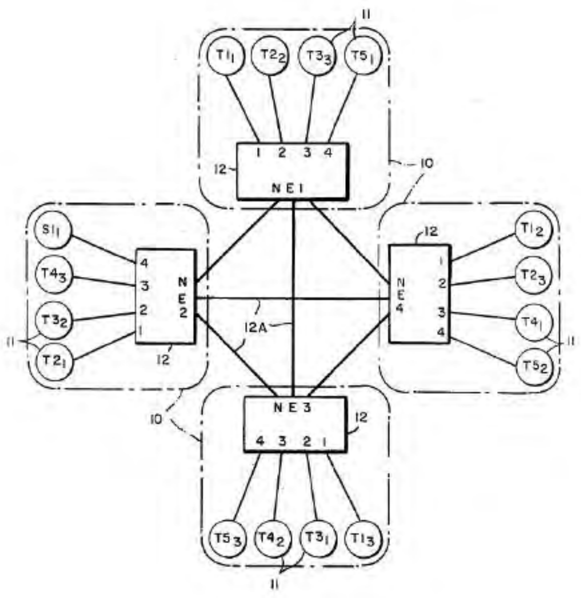
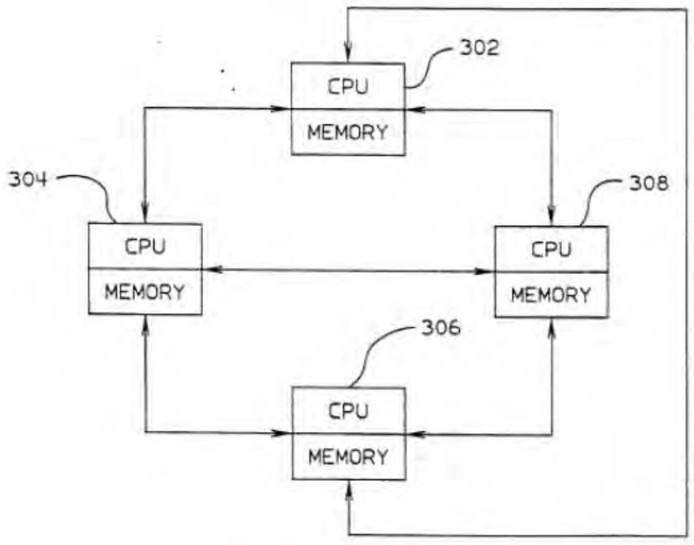


FIG. 1

Harper, Figure 1

- Baum: *See*, *e.g.*, 1:49-61 ("A second type of commonly known multiprocessing system is the multicomputer message passing network (FIG. 3). Message passing networks are configured by interconnecting a number of processing nodes. Each node 302-308 includes a central processing unit and a local memory that is not globally accessible. In order for an application to share data among processors the programmer must explicitly code commands to move data from one node to another. In contrast to shared memory systems, the time that it takes for a processor to access data depends on its distance (in nodes) from the processor that currently has the data in its local memory.")



Baum, Figure 3

- U.S. Patent No. 5,623,644 to Self: *See, e.g.*, 8:65-9:12 ("FIG. 8 shows an example topology used in one embodiment of the present invention for coupling a number of processors in a mesh configuration using embodiments of the present invention. For example, 800 illustrates a mesh configuration utilizing routers in a backplane of a supercomputer system employing the point-to-point interconnect technology described above. Each router such as 850 shown in FIG. 8 may comprise pairs of uni-directional point-to- point communication interfaces such as 851-854 for coupling to other routers in both the X+ and X- and Y+ and Y directions as illustrated for different communication channels. Thus, communication is provided between each of the routers in the mesh. Each router in the mesh backplane is also coupled via a pair of uni-directional point-to-point interconnects 855 to a processing node 860.")

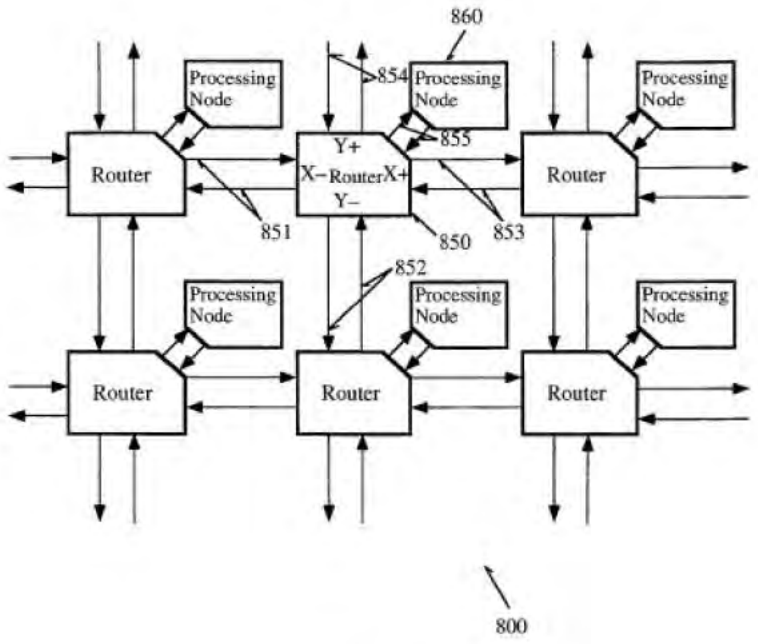
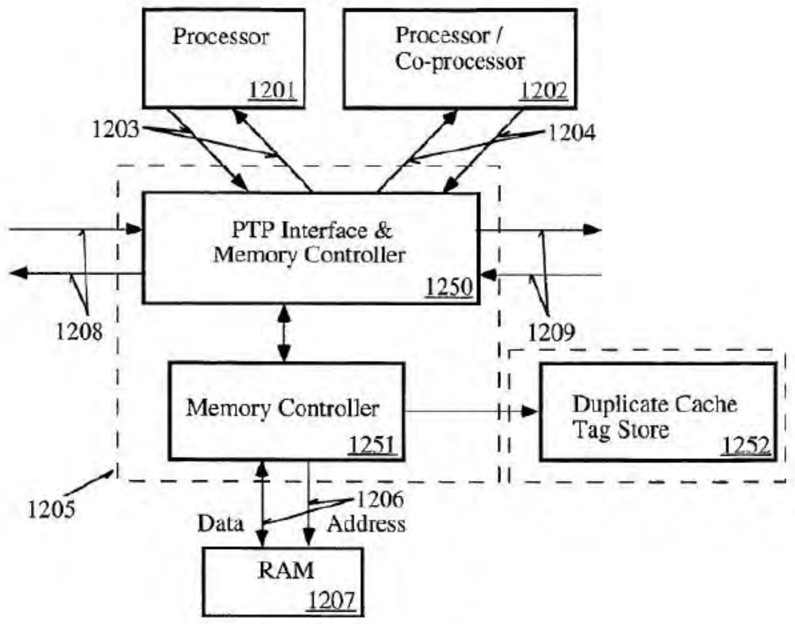


Fig. 8

Self, Figure 8

- Self: *See, e.g.*, 10:35-42 ("Another alterative configuration for a computer system, such as that having a plurality of microprocessors (e.g., 1201 and 1202) is shown in FIG. 12a. In this example, both processor 1201 and processor 1202 (which also may be a co-processor such as a math co-processor or digital signal processor) are coupled to a point-to-point communication interface and memory controller 1205 via pairs of unidirectional point-to-point communication links 1203 and 1204.")

Public Version – Confidential Information Redacted and Confidentiality Designation Removed Per Agreement Between the Parties



Self, Figure 12B

- U.S. Patent No. 5,752,264 to Blake: *See, e.g.*, 5:60-6:6 ("Referring to FIG. 2, there is shown a Multi-Processor computer system incorporating the present invention. The system includes: a system memory module 210; an input/output device (I/O device) 212; a plurality of clusters 214a-214n, and a common bus 216 that links the memory module, I/O device, and clusters together. Each cluster includes a level two cache, level two caches 224a-224n, and a plurality of microprocessors (CPU), CPUs 218aa-218an for cluster 214a, CPUs 218ba-218bm for cluster 224b, and CPUs 218na-218nm for cluster 214n. Each CPU has a level one cache, and is coupled to its respective level two cache through the level one cache via a point to point bus. Each cluster is coupled to the shared bus through its level two cache.")

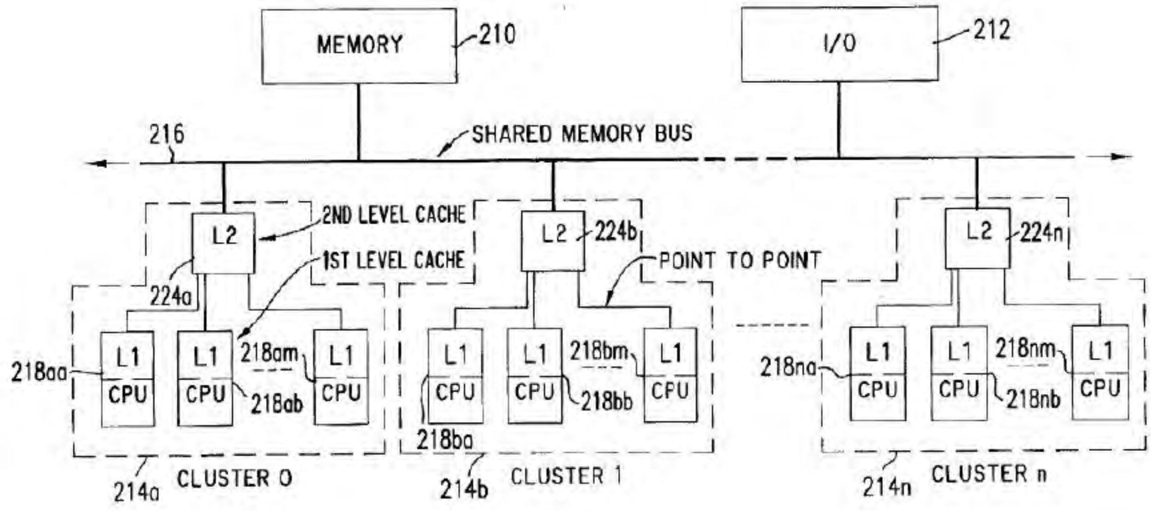


FIG. 2

Blake, Figure 2

- U.S. Patent No. 6,247,161 to Lambrecht: *See, e.g.,* 3:66-4:18 ("Referring now to FIG. 2, an embodiment of computer chip 100 with an on-chip data transfer network is shown, for interconnecting a plurality of devices or modules 210A-210H linked by a plurality of communication nodes 220A-220H on single computer chip 100. The components of the network preferably include a bus 230 with the plurality communication nodes 220A-220H coupled to the bus 230 as well as a plurality of communication nodes 225A-225D coupled to segments of the bus 230. The bus 230 is comprised of the individual buses connecting between and among nodes 220A-220H and/or nodes 225A-225D. Communications nodes 225 are preferably a subset of the plurality of communication nodes 220 operable to transmit and receive data only on the bus 230. A reference to bus 230 may refer to the entire bus system or to a particular segment or component. The term bus as used in this disclosure is meant to extend to a bus which passes data along its entire physical length during a single transmission, as well as to a communications or transfer link which uses point-to-point data transmission.")

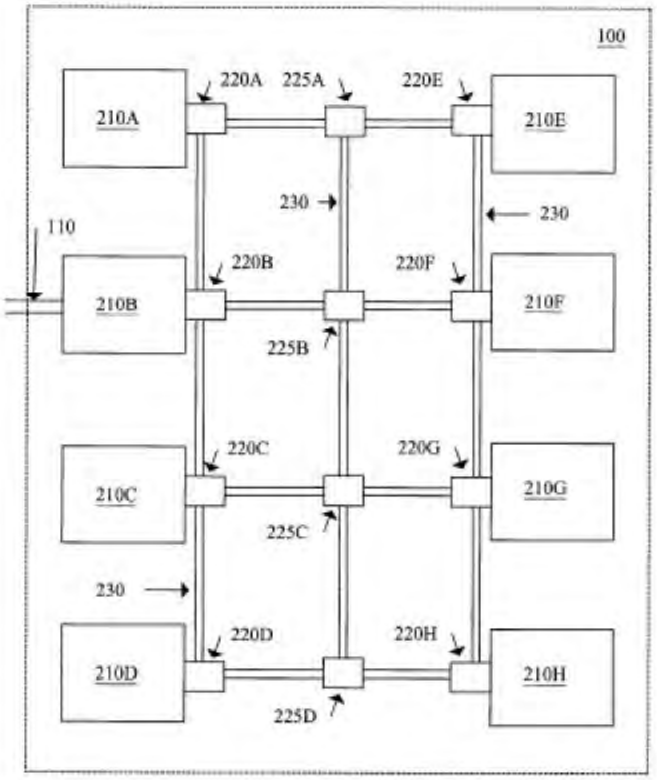
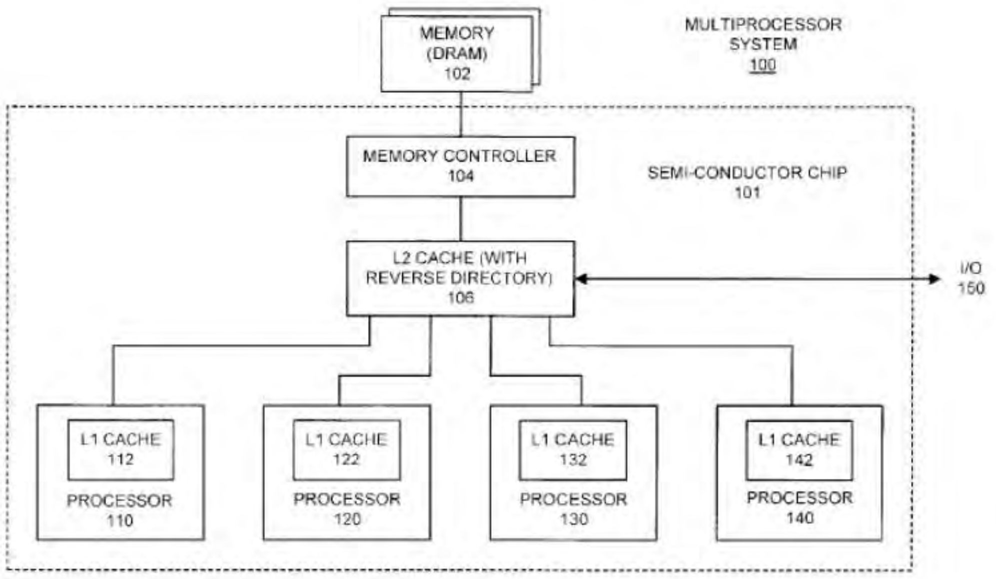


FIG. 2

Lambrecht, Figure 2

- U.S. Patent No. 6,684,297 to Chaudry: *See, e.g.,* 3:66-4:5 ("FIG. 1B illustrates a multiprocessor system 100 with a reverse directory in accordance with an embodiment of the present invention. Note much of multiprocessor system 100 is located within a single semiconductor chip 101. More specifically, semiconductor chip 101 includes a number of processors 110, 120, 130 and 140, which contain level one (L1) caches 112, 122, 132 and 142, respectively.")



Chaudry, Figure 1B

- Chaudry: *See, e.g.,* 4:27-42 ("FIG. 2 illustrates an L2 cache 106 with multiple banks in accordance with an embodiment of the present invention. In this embodiment, L2 cache 106 is implemented with four banks 202-205, which can be accessed in parallel by processors 110, 120, 130 and 140 through switch 220. Note that only two bits of the address are required to determine which of the four banks 202-205 a memory request is directed to. Switch 120 additionally includes an I/O port 150 for communicating with I/O devices. Also note that each of these banks 202-205 includes a reverse directory. Furthermore, each of the banks 202-205 has its own memory controller 212-215, which is coupled to an associated bank of off-chip memory 232-235." "Note that with this architecture, it is possible to concurrently connect each L1 cache to its own bank of L2 cache, which increases the bandwidth to the L2 cache 106.")

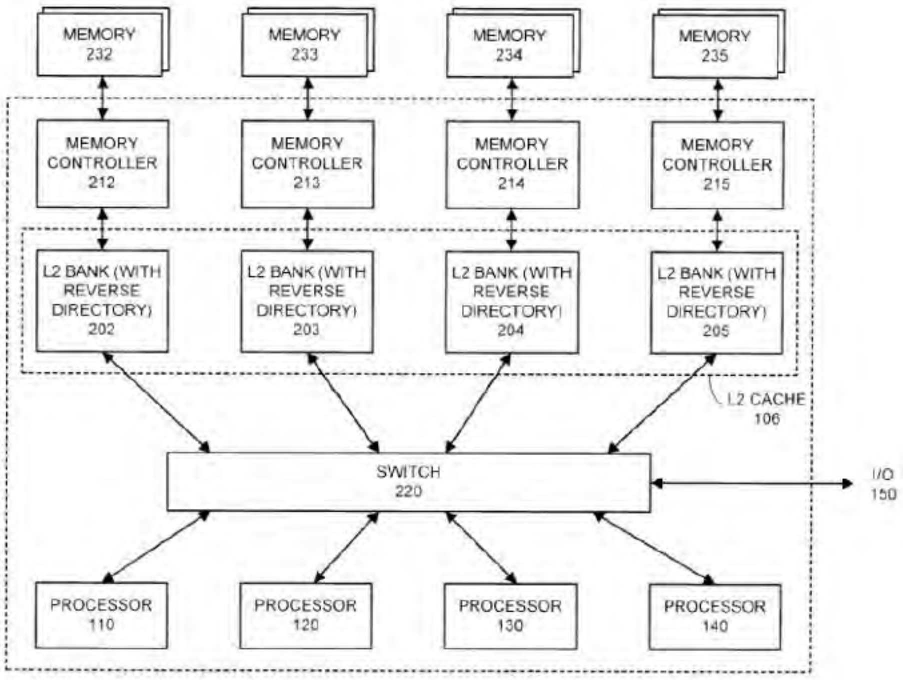
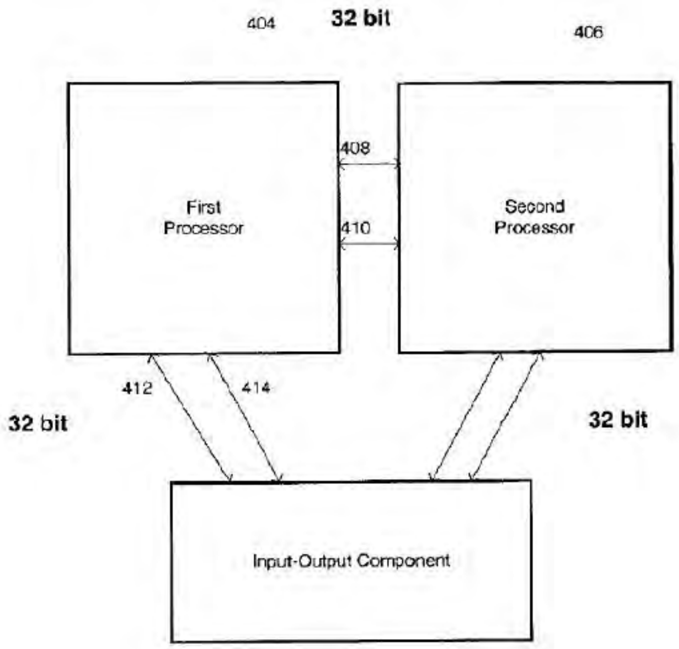


FIG. 2

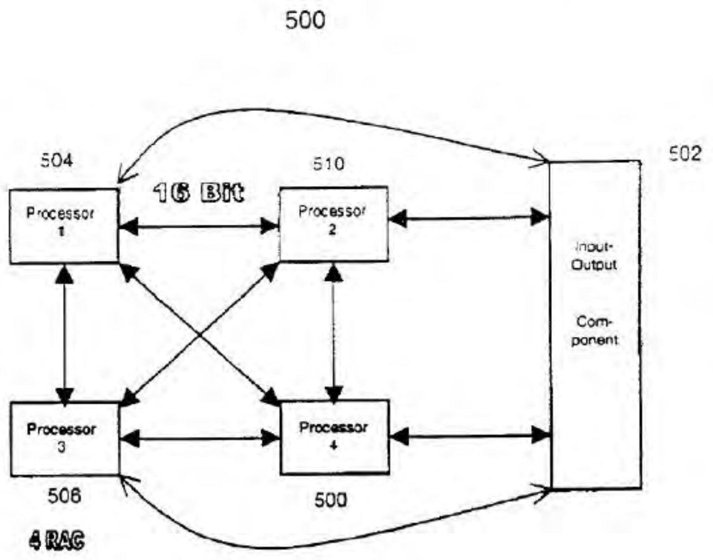
Chaudry, Figure 2

- U.S. Patent No. 6,826,645 to Kosaraju: *See, e.g.,* 4:4-8 ("FIG. 4 illustrates a two processor point-to-point architecture 400 having a 32-bit point-to-point connection between: 1) the input-output component 402 and the first processor 404; as well as 2) the input-output component 402 and the second processor 406.")



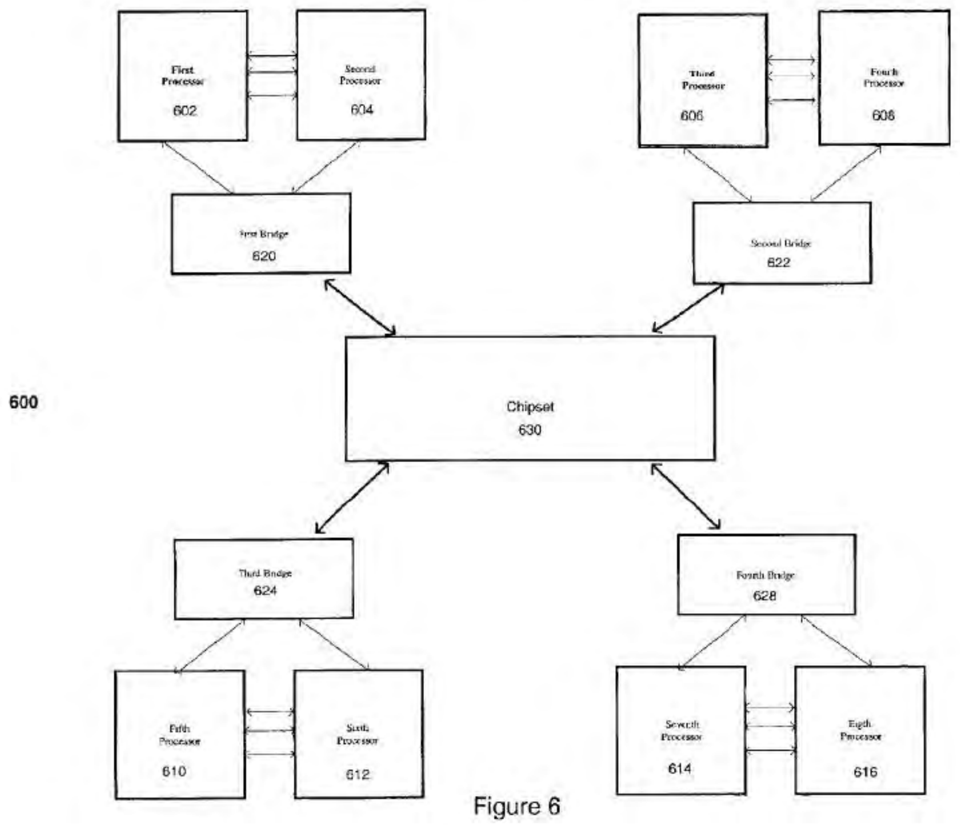
Kosaraju, Figure 4

- Kosaraju: *See, e.g.,* 4:27-42 ("FIG. 5 illustrates an embodiment of a four processor point-to-point architecture 500 having a 16-bit point-to-point connection between an input-output component 502 and each of the four processors 504, 506, 508, 510. The four processors are processor 1 504, processor 2 506, processor 3 508, and processor 4 510.")



Kosaraju, Figure 5

- Kosaraju: *See, e.g.,* 5:11-35 ("FIG. 6 illustrates an embodiment of an eight processor (8P) point-to-point architecture 600 comprised of four pairs of processors, 602, 604, 606, 608, 610,612, 614 and 616, linked to a corresponding bridge, 620, 622, 624, 628, and each bridge, 620, 622, 624, 628, connected to a chipset 630. A first processor 602 and a second processor 604 each have a 16-bit point to point connection with a first bridge 620. The first bridge 620 has a connection, such as a 32-bit connection, with the chipset 630. Similarly, a third processor 606 and a fourth processor 608 each have a 16-bit point to point connection with a second bridge 622. The second bridge 622 has a connection, such as a 32-bit connection, with the chipset 630. In a similar fashion, the fifth processor 610 through eighth processor 612 eventually link with the chipset 630. In an embodiment, the arbiter may allow the signal paths internal to each processor 602, 604, 606, 608, 610,612, 614 and 616, to be changed by a programmable setting. This allows a manufacturer to fabricate a single version of a processor with a generic but flexible architecture within the processor to service multiple processor platforms. Thus, in an embodiment, an arbiter linked to a processor having a single flexible architecture may be employed to service, a 2P architecture, 4P architecture, 8P architecture or other multiple processor architecture, as well as service a server application and a workstation application.")



Kosaraju, Figure 6

- U.S. Patent No. 6,996,681 to Autechaud: *See, e.g.,* 5:9-17 ("Each building block QBi is linked to the other four via four respective point-to-point links, not referenced in order not to overcomplicate the figure. These links are considered, at the protocol level, as virtual busses. Thus, in the example described, there are four virtual busses and hence four levels of interleaving handled by each connection agent NCSi. Each bus thus allows four accesses, or transactions, in each clock cycle.")

Public Version – Confidential Information Redacted and Confidentiality Designation Removed Per Agreement Between the Parties

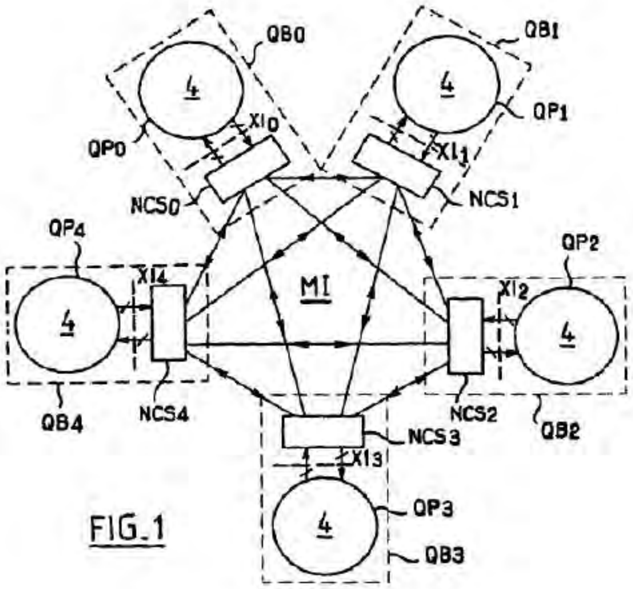


FIG_1

Autechaud, Figure 1

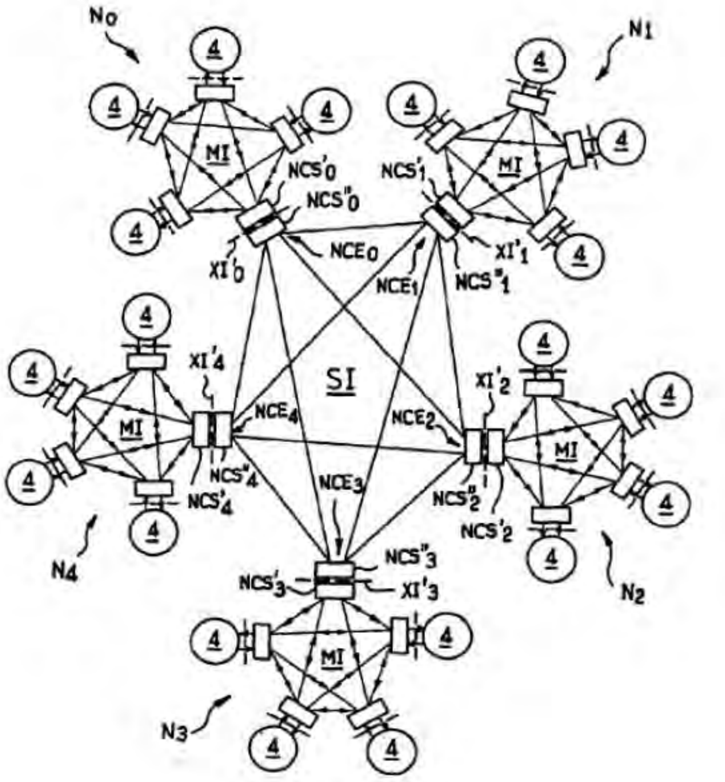


FIG_4

Autechaud, Figure 4

e) Using a Bus to Connect Components in a Multiprocessor System was Known

Furthermore, it was well known before the priority dates of the Asserted Patents to connect processing components via a bus. Indeed, the Asserted Patents acknowledge that "[c]onventional multiple processor computer systems have processors coupled to a system memory through a shared bus." *See, e.g.,* '409 patent, 1:29-31. Examples of prior art references that disclose connecting components via a bus and further demonstrate that such a structure was well known include:

- Hamacher: *See, e.g.*, p. 624 ("The simplest and most economical means for interconnecting a number of modules is to use a single bus.")
- Hennessy: *See, e.g.*, p. 717 ("Several microprocessors can usefully be placed on a common bus for several reasons:
    - Each microprocessor is much smaller than a multichip processor, so more processors can be placed on a bus.
    - Caches can lower bus traffic.
    - Mechanisms were invented to keep caches and memory consistent for multiprocessors, just as caches and memory are kept consistent for I/O, thereby simplifying programming.")

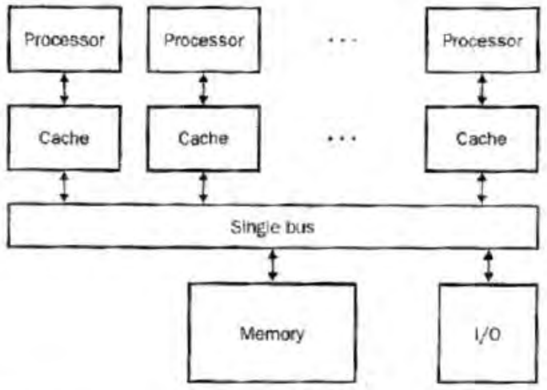


FIGURE 9.2 A single-bus multiprocessor. Typical size is between 2 and 32 processors.

Hennessy, Figure 9.2

- U.S. Patent No. 5,394,555 to Hunter: *See, e.g.*, 2:48-54 ("Each CPU has the property of coherence in its communications with other system components and typically incorporates a primary cache 2, and each node includes a local shared memory 3 which communicates through the primary caches with the CPUs 1 in the node via a node-local communications system such as a node-local bus 4.")

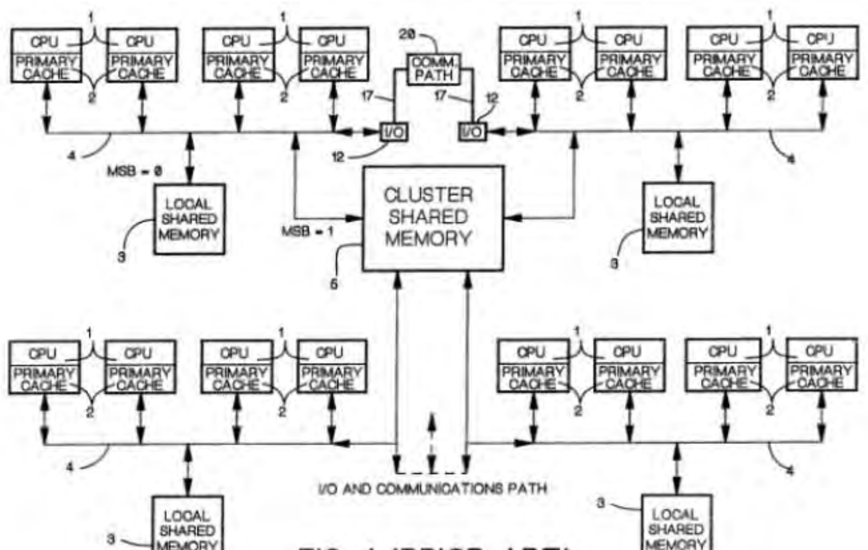


FIG. 1 (PRIOR ART)

US 5,394,555, Figure 1

f) It Was Known That The Various Interconnects Could Be Interchanged in Multiprocessor Systems

In multiprocessor systems before the priority dates of the Asserted Patents, it was well known to interconnect processors using at least the above-noted topologies. Examples of prior art references that disclose that using a particular one of different topologies is merely an engineering design choice and that substituting one topology for another topology was well known include:

- Hamacher: *See, e.g.*, p. 636 ("We have considered several possible network topologies and showed that all existing topologies have certain advantages and disadvantages. Designers of multiprocessor systems strive to achieve superior performance at a reasonable cost. In an effort to exploit the most advantageous characteristics of different topologies, many successful machines feature mixed topologies. Bus and crossbar are excellent choices for connecting a few processors together. So, we often see a cluster of processors, typically from 2 to 8, connected using a bus or a crossbar. Such clusters, usually referred to as nodes, are then interconnected using a suitable topology to form a larger system.")

- Culler: *See, e.g.*, p. 556 ("However, other combinations such as snooping-snooping (Frank, Burkhardt, and Rothnie 1993), directory-directory (Convex Computer Corporation 1993), and even snooping-directory may be used (see Figure 8.4).")

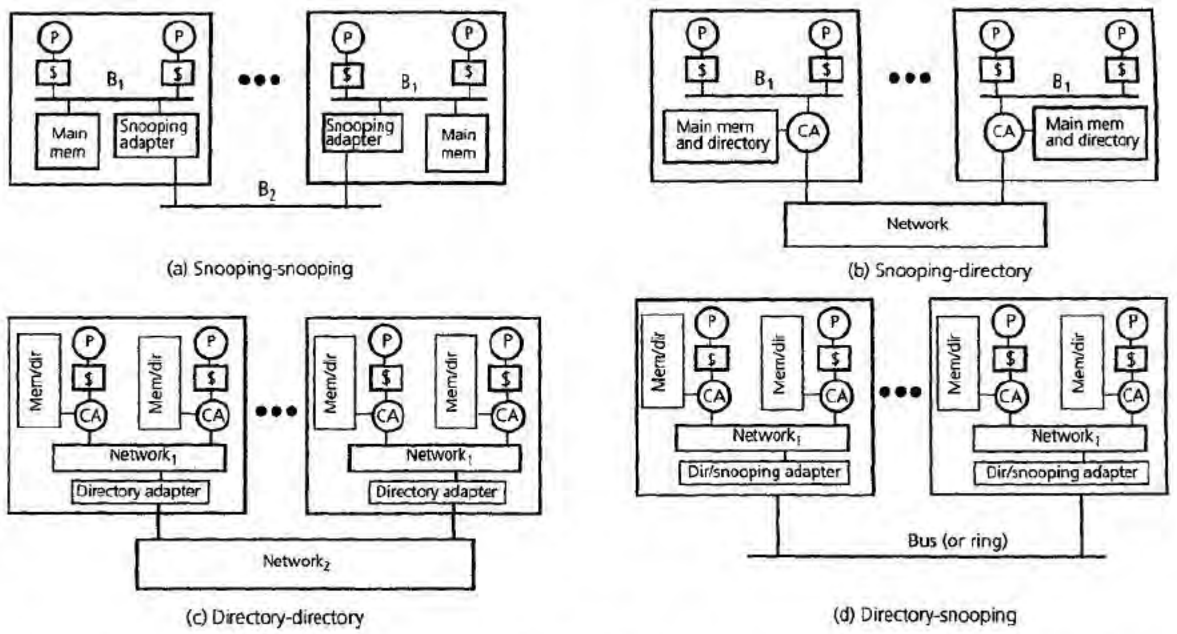


FIGURE 8.4 Some possible organizations for two-level cache-coherent systems. Each node visible at the outer level is itself a multiprocessor. B_1 is a first-level bus, and B_2 is a second-level bus. CA is the communication assist. The label snooping-directory, for example, means that a snooping protocol is used to maintain coherence within a multiprocessor node, and a directory protocol is used to maintain coherence across nodes.

Culler, Figure 8.4

- Culler: *See, e.g.*, p. 665 ("**Block diagram for a hierarchical ring-based multiprocessor**. In the two-level hierarchy shown, each local ring is a node as viewed by the global ring, and an inter-ring interface propagates relevant transactions between the two.")

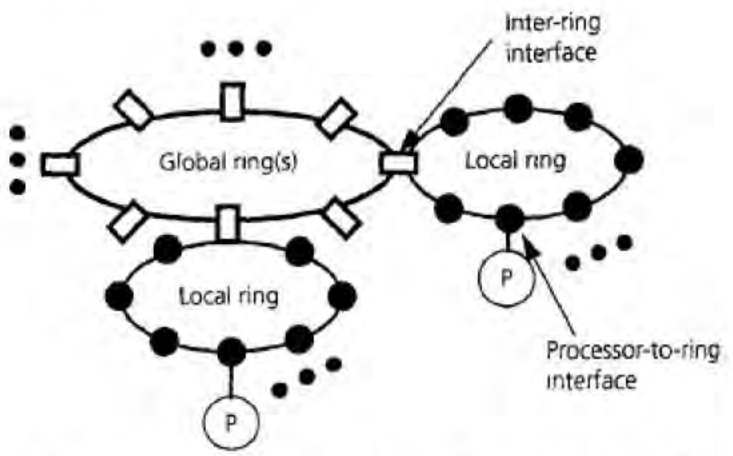


FIGURE 8.39 Block diagram for a hierarchical ring-based multiprocessor. In the two-level hierarchy shown, each local ring is a node as viewed by the global ring, and an inter-ring interface propagates relevant transactions between the two|

Culler, Figure 8.39

- Computational Education Project: *See, e.g.*, p. 56 ("A recent commercial system in this category with computing power and scalability that could potentially make it widely used in computational science, is the KSR-1 from Kendall Square Research. Processing elements are connected in rings, with from 8 to 32 PEs per ring. Larger systems have a second level ring that connects up to 34 first- level rings, for a maximum machine size of 1088 processors. Each ring is unidirectional, i.e., information flows in only one direction, with a bandwidth of 1 GB/sec.")

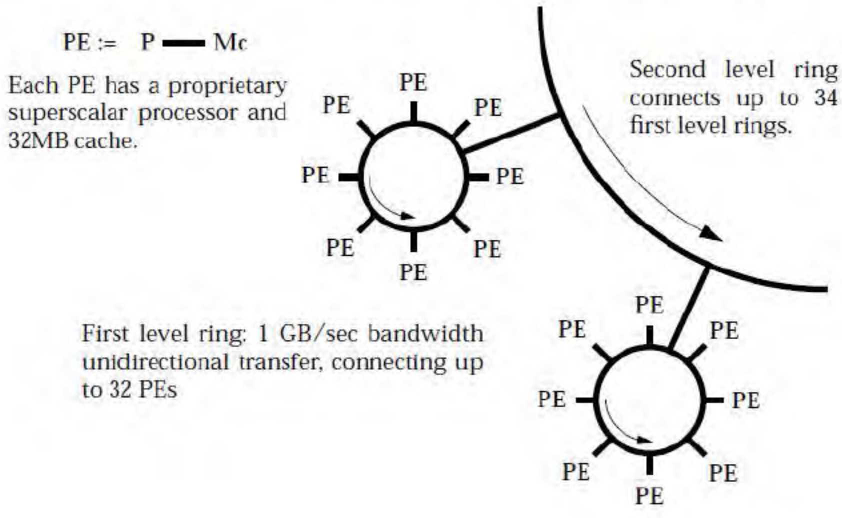

PE := P — Mc
Each PE has a proprietary superscalar processor and 32MB cache.
PE
Second level ring connects up to 34 first level rings.
First level ring: 1 GB/sec bandwidth unidirectional transfer, connecting up to 32 PEs

Figure 18: Kendall Square Research KSR 1

Computational Education Project, Figure 18

- Barroso: *See, e.g.*, p. 12 ("Buses, crossbars and rings are clearly not the only interconnection options for shared-memory multiprocessor systems. However, we argue that in the context of small scale multiprocessors, these are the ones that make the most sense. Large MINs, meshes, or fat trees scale well to large numbers of processors, but are not as effective for small configurations.")

- Barroso: *See, e.g.*, p. 12-13 ("Cluster-based architectures are particularly effective when a significant fraction of the application parallelism can be captured by a single SMP node, or when the application can be mapped so that there is communication locality within a SMP node. Therefore, SMP nodes with a larger number of processors are preferred. Since bus based systems are likely to connect at most four processors in the near future, alternative SMP interconnections such as rings or crossbars could be favored as the building blocks for larger scale systems as well.")

- "Performance Issues in the Design of Hierarchical-ring and Direct Networks for Shared-memory Multiprocessors," Ravindran: *See, e.g.*, p. 11 ("In a two-level hierarchical-ring network, several direct single ring networks, referred to as local rings, are connected by a global ring consisting of switches or inter-ring interfaces [92, 93].")

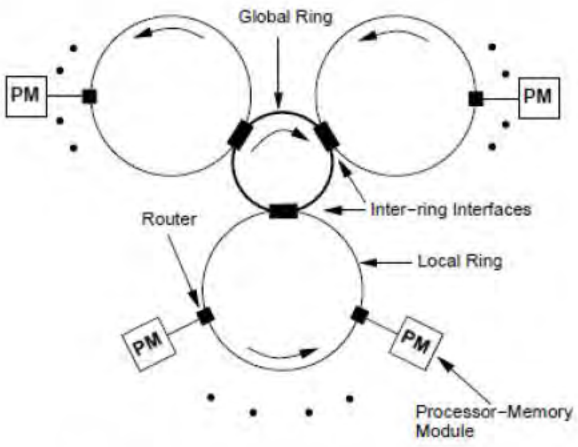


Figure 2.4: A 2-level hierarchical-ring connected network.

Ravindran, Figure 2.4

- Hwang, Table 6.1:

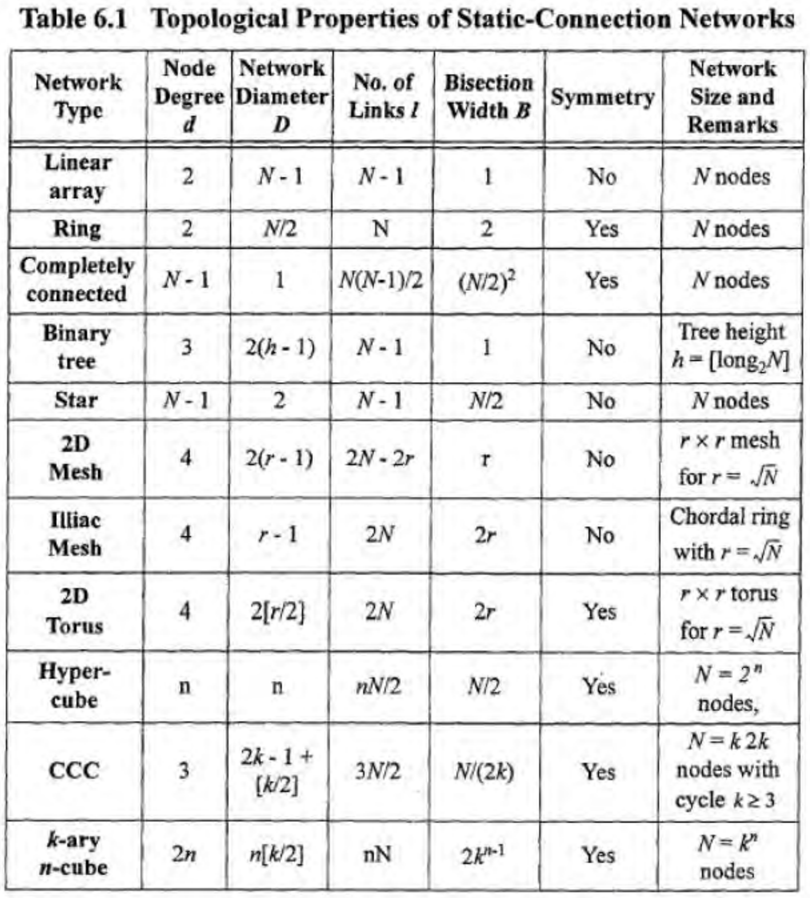


Table 6.1 Topological Properties of Static-Connection Networks

Network Type	Node Degree d	Network Diameter D	No. of Links l	Bisection Width B	Symmetry	Network Size and Remarks
Linear array	2	$N - 1$	$N - 1$	1	No	N nodes
Ring	2	$N/2$	N	2	Yes	N nodes
Completely connected	$N - 1$	1	$N(N-1)/2$	$(N/2)^2$	Yes	N nodes
Binary tree	3	$2(h - 1)$	$N - 1$	1	No	Tree height $h = [\log_2 N]$
Star	$N - 1$	2	$N - 1$	$N/2$	No	N nodes
2D Mesh	4	$2(r - 1)$	$2N - 2r$	r	No	$r \times r$ mesh for $r = \sqrt{N}$
Illiac Mesh	4	$r - 1$	$2N$	$2r$	No	Chordal ring with $r = \sqrt{N}$
2D Torus	4	$2[r/2]$	$2N$	$2r$	Yes	$r \times r$ torus for $r = \sqrt{N}$
Hyper-cube	n	n	$nN/2$	$N/2$	Yes	$N = 2^n$ nodes,
CCC	3	$2k - 1 + [k/2]$	$3N/2$	$N/(2k)$	Yes	$N = k\,2k$ nodes with cycle $k \geq 3$
k-ary n-cube	$2n$	$n[k/2]$	nN	$2k^{n-1}$	Yes	$N = k^n$ nodes

- "Performance Issues in the Design of Hierarchical-ring and Direct Networks for Shared-memory Multiprocessors," Ravindran: *See, e.g.*, p. 11 ("In a two-level hierarchical-ring network, several direct single ring networks, referred to as local rings, are connected by a global ring consisting of switches or inter-ring interfaces [92, 93].")

- "The NUMAchine Multiprocessor," Grindley: *See, e.g.*, p. 1 ("A variety of large-scale multiprocessor architectures has been developed, as indicated in Table 1. The relevant features considered here are: type of clustering, type of interconnect, presence of caches for remote data, and the choice between non-uniform memory (NUMA) or cacheonly memory (COMA) architectures. Clustering processors together is a means of leveraging commodity symmetric multiprocessor (SMP) nodes. There are a number of possibilities for the system-wide interconnect including meshes, multistage switch networks, and rings. Each has advantages and disadvantages in terms of performance, complexity, and cost. Some systems include caches for remote data to mitigate longer memory access latencies as the system size increases. Finally, some systems employ a cache-only architecture (COMA) to automatically replicate and migrate data in hardware, rather than rely on caching with home memory locations as in NUMA systems. The systems listed in Table 1 use NUMA architecture, unless otherwise stated. The variety of architectures in Table 1 suggests that there is no single best approach when engineering such systems.")

Table 1. Some commercial and experimental multiprocessors

Name	Cluster	Interconnect	Features
DASH [14]	bus	mesh	remote access cache
FLASH [7]	non-clustered	mesh	programmable protocol processor, page replication/migration
Origin2000 [13]	paired-processors	cube	page replication/migration
I-ACOMA [20]	bus	mesh	simultaneous multithreading, cache-only memory architecture
Teracomputer [19]	non-clustered	multistage switch	multithreaded execution, no caching or data replication
Starfire [1]	bus	multiple buses	global snooping, crossbar for responses
V-class [8]	crossbar	toroidal ring	remote data caches
KSR1 [12]	non-clustered	hierarchical rings	cache-only memory architecture
NUMA-Q [16]	bus	ring	remote data caches

Grindley, Table 1

- Grindley: *See, e.g.*, p. 2 ("NUMAchine consists of a number of stations connected by a two-level hierarchy composed of Local Rings and a Central Ring, as shown in Figure 1. The ring hierarchy is joined by an Inter-Ring Interface. Data transfers across rings are divided into packets which are sent according to a slotted ring protocol. Routing packets on the NUMAchine ring is simple and fast: each station checks a single bit to determine if a packet has reached its destination.")

- Grindley: *See, e.g.*, p. 2 ("Unidirectional slotted rings were chosen for a number of reasons. First, they can perform as well as meshes for up to 128 processors [17, 18] when some data locality is present. Second, stations can be added one at a time without significant re-wiring or topology changes, making them highly modular and cost-effective. Third, rings exhibit two features useful for implementing cache coherence and memory consistency: inherent sequencing of requests/responses, and natural broadcast capabilities. For example, a single request invalidates multiple copies of a cache line as it traverses the ring hierarchy, and serves as an acknowledgment upon its return to the source of the request.")

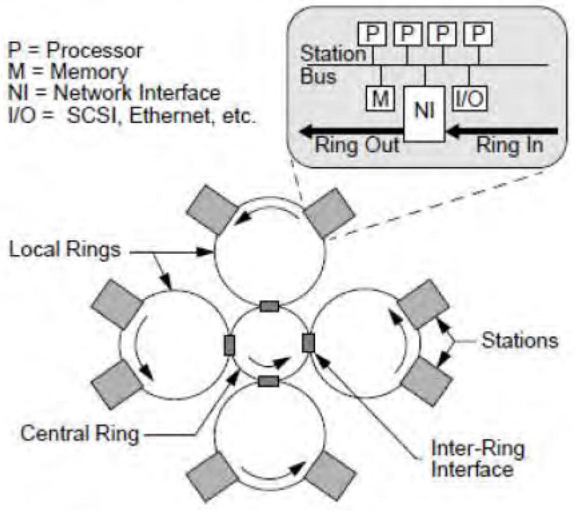


Figure 1. NUMAchine architecture

Grindley, Figure 1

- "Multicast Snooping: A New Coherence Method Using a Multicast Address Network," Bilir: *See, e.g.*, p. 1 ("The Sun Ultra Enterprise 10000 [1], for example, uses four address 'buses' interleaved by address. It implements each 'bus' with a pipelined broadcast tree constructed from point-to-point links (that behave more like ideal transmission lines to facilitate having multiple bits concurrently in flight), and it has a separate unordered data network (a point-to-point crossbar).")

- Bilir: *See, e.g.*, p. 2 ("Figure 1 shows the major components of a system that uses multicast snooping. We assume that addresses traverse a totally-ordered multicast address network, such as the one described in Section 3, that data travels on a separate point-to-point data network[1], as in the Sun E10000, and that memory is physically distributed among processors. We illustrate our ideas using a write-invalidate MOSI protocol.")

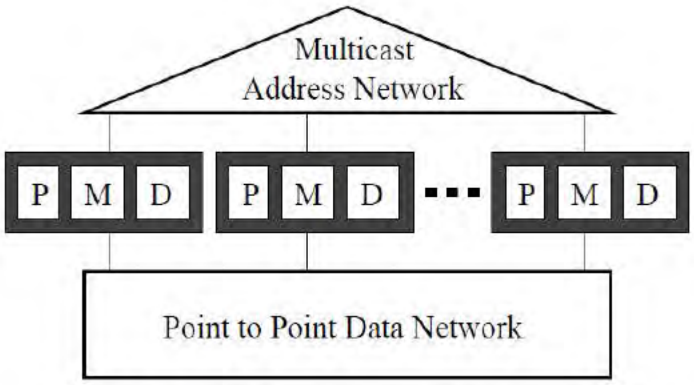


FIGURE 1. Major components of a multicast snooping system. P, M and D refer to processor, memory and associated directory.

Bilir, Figure 1

- U.S. Patent Application No. 2002/046324 to Barroso: *See, e.g.*, para. [0037] ("FIG. 1 shows the block diagram of a single PIRANHA ™ processing chip 10. Each ALPHA™ CPU core (central processing unit or CPU) 110 is directly connected to dedicated instruction (iLI) and data cache (dLI) modules 120 and 121. These first-level caches interface to other modules through the IntraChip Switch (ICS) 122."); *See, e.g.*, para. [0040] ("FIG. 3 shows an example configuration of a PIRANHA ™ system 30 with both processing and I/O chips 10 and 20. The PIRANHA™ design allows for glueless scaling up to 1024 nodes, with an arbitrary ratio of I/O to processing nodes (which can be adjusted for a particular workload."); *See, e.g.*, para. [0046] "Conceptually, the intra-chip switch (ICS), e.g., 122 (FIG. 1), is a crossbar that interconnects most of the modules on a PIRANHA™ chip."); and *See, e.g.*, para. [0084] ("Each PIRANHA™ processing node has four channels that are used to connect it to other nodes in a point-to-point fashion (element 146 in FIG. 1).")

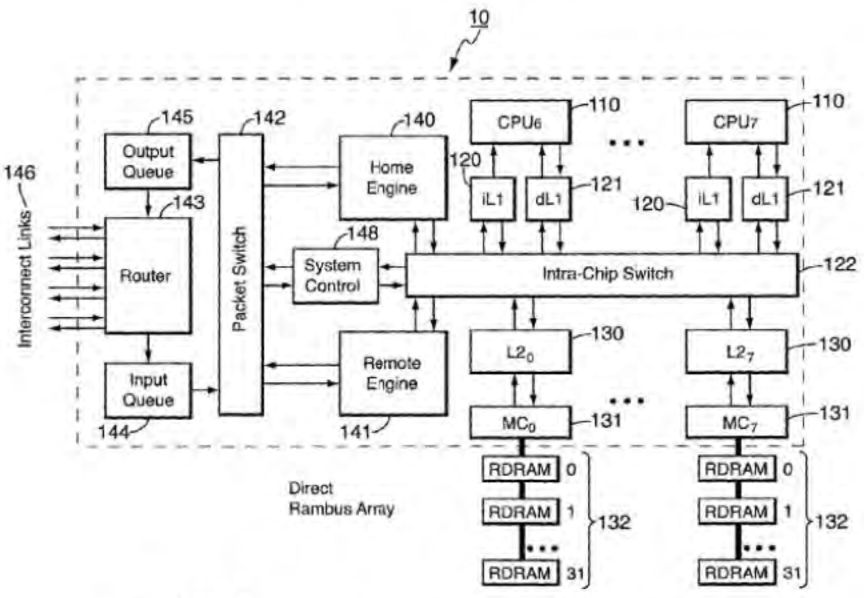


FIG. 1 Block diagram of a single-chip Piranha processing node.

Barroso '324, Figure 1

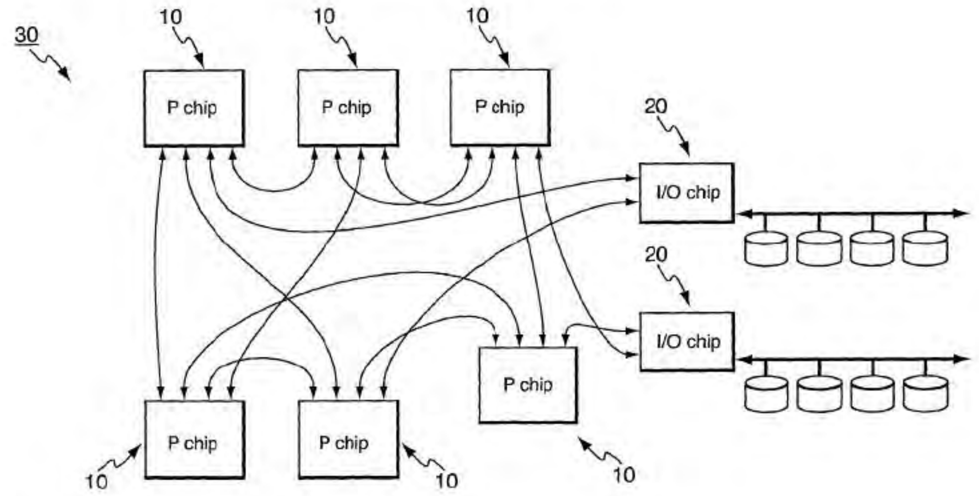


FIG. 3 Example configuration for a Piranha system with six processing (8 CPUs each) and two I/O chips.

Barroso '324, Figure 3

- Keller: *See, e.g.*, 1:31-37 ("One or more of the above problems may be addressed using a distributed memory system. A computer system employing a distributed memory system includes multiple nodes. Two or more of the nodes are connected to memory, and the nodes are interconnected using any suitable interconnect. For example, each node may be connected to each other node using dedicated lines. Alternatively, each node may connect to a fixed number of other nodes, and transactions may be routed from a first node to a second node to which the first node is not directly connected via one or more intermediate nodes. The memory address space is assigned across the memories in each node."); 10:17-27 ("As mentioned earlier, any processing node in Fig. 1 may function as a source node, a target node or a remaining node depending on the particular transaction. The arrangements shown in Figs. 9, 13 and 14 are for illustrative purpose only, and they do not imply similar actual connections among the processing nodes 12A- 12D. That is, a remaining node, e.g. node 76, or the target node 72 may not be directly connected to the source node 70. Hence, additional packet routing may occur. Further, the arrangements of Figs. 9, 13 and 14 are described with reference to the circuit topology in Fig. 1. It is understood that other interconnections between two or more processing nodes may be contemplated and the packet transfer schemes of Figs. 9, 13 and 14 may be easily implemented in those various interconnections. The arrows are used to indicate dependencies and represent packets that must be sent between respective nodes joined by the arrows. Generally, the outgoing arrows may not be taken until all corresponding incoming dependencies have happened. This is illustrated further below with reference to the operations depicted in Figs. 9, 13 and 14.")

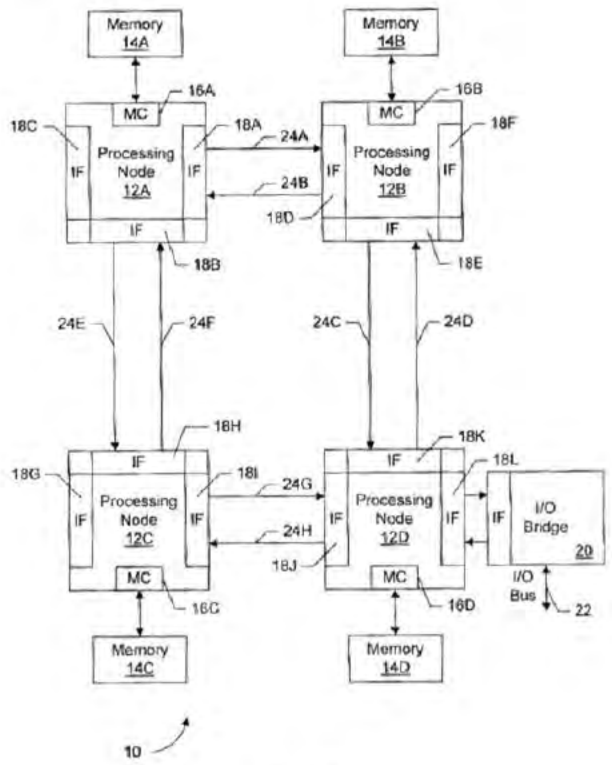


FIG. 1

Keller, Figure 1

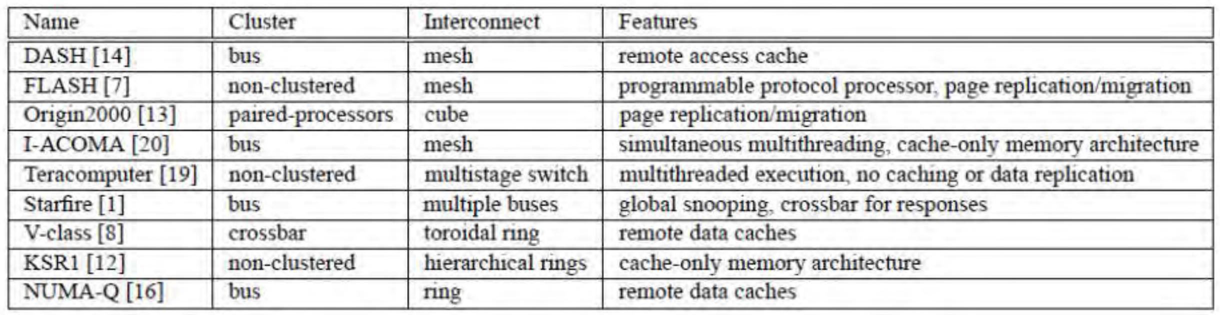
g) It Was Well Known That There Were Many Reasons To Use A "Point-to-Point" Architecture

It was well known before the priority dates of the Asserted Patents that selecting one of a finite number of known topologies (*e.g.*, shared bus, point-to-point architecture, ring, crossbar switch, etc.) was an obvious engineering design choice based on a variety of known design considerations according to the desired multiprocessor system implementation. A person of ordinary skill would understand that these design considerations may include cost, bandwidth, effective throughput, number or processors, and cost as described below:

- Hamacher: *See*, *e.g.*, p. 664 ("The suitability of a particular network is judged in terms of cost, bandwidth, effective throughput, and ease of implementation.")
- Computational Education Project: *See*, *e.g.*, p. 42 ("A major consideration in the design of parallel systems is the set of pathways over which the processors, memories, and switches communicate with each other. These connections define the interconnection network, or topology, of the machine. Attributes of the topology determine how processors will share data and at what cost.")
- Patterson: *See*, *e.g.*, p. 713 ("In addition to the two main communication styles, multiprocessors are constructed in two basic organizations: processors connected by a single bus, and processors connected by a network. The number of processors in the multiprocessor has a lot to do with this choice.")
- Culler: *See*, *e.g.*, pgs. 749-750 ("As with all other aspects of design, network design involved understanding tradeoffs and making compromises so that the solution is near optimal in a global sense rather than optimized for a particular component of interest. The performance impact of the many interacting facets can be quite subtle. Moreover, there is not a clear consensus in the field on the appropriate cost model for networks, since trade-offs can be made between very different technologies, for example, bandwidth of the links may be traded against complexity of the switches.")
- Grindley: *See*, *e.g.*, p. 1 ("A variety of large-scale multiprocessor architectures has been developed, as indicated in Table 1. The relevant features considered here are: type of clustering, type of interconnect, presence of caches for remote data, and the choice between non-uniform memory (NUMA) or cacheonly memory (COMA) architectures. Clustering processors together is a means of leveraging commodity symmetric multiprocessor (SMP) nodes. There are a number of possibilities for the system-wide interconnect including meshes, multistage switch networks, and rings. Each has advantages and disadvantages in terms of performance, complexity, and cost. Some systems include caches for remote data to mitigate longer memory access latencies as the system size increases. Finally, some systems employ a cache-only architecture (COMA) to automatically replicate and migrate data in hardware, rather than rely on caching with home memory locations as in NUMA systems. The systems listed in Table 1 use NUMA architecture, unless otherwise stated. The variety of architectures

in Table 1 suggests that there is no single best approach when engineering such systems.")

Table 1. Some commercial and experimental multiprocessors



Name	Cluster	Interconnect	Features
DASH [14]	bus	mesh	remote access cache
FLASH [7]	non-clustered	mesh	programmable protocol processor, page replication/migration
Origin2000 [13]	paired-processors	cube	page replication/migration
I-ACOMA [20]	bus	mesh	simultaneous multithreading, cache-only memory architecture
Teracomputer [19]	non-clustered	multistage switch	multithreaded execution, no caching or data replication
Starfire [1]	bus	multiple buses	global snooping, crossbar for responses
V-class [8]	crossbar	toroidal ring	remote data caches
KSR1 [12]	non-clustered	hierarchical rings	cache-only memory architecture
NUMA-Q [16]	bus	ring	remote data caches

Grindley, Table 1

Indeed, a person of ordinary skill in the art would have been motivated to implement one of the finite number of topologies for the desired multiprocessor system design in accordance with the well-known limitations and advantages of each topology. For example, one of ordinary skill would have known that a shared bus benefits from its simplicity as described by Hamacher (*see, e.g.*, p. 624 ("The simplest and most economical means for interconnecting a number of modules is to use a single bus.")). However, one of ordinary skill would further understand the limitations of a shared bus topology:

- Patterson: *See, e.g.*, p. 727 ("Single-bus designs are attractive, but limited because the three desirable bus characteristics are incompatible: high bandwidth, low latency, and long length." "If the goal is to connect many more processors together, then the computer designer needs to use more than a single bus.")
- "Directory-Based Cache Coherence in Large-Scale Multiprocessors," Chaiken et al. (1990): *See, e.g.*, p. 49 ("Unfortunately, buses simply don't have the bandwidth to support a large number or processors." "As processors speeds increase, the relative disparity between bus and processor clocks will simply become more evident.")
- Hwang: *See, e.g.*, p. 297-298 ("**Shortcomings in Bus Interconnects** Bus interconnects are for time-shared use by many processors. Even when the bus bandwidth is high, per-processor bandwidth is only a fraction of the total bus bandwidth. Furthermore, bus is prone to failure for lack of redundancy. Bus also has limited scalability. These shortcomings are primarily constrained by packaging technology and the cost involved."

- Barroso: *See, e.g.*, p. 7-8 ("**Limits on Bus Performance** In the past few years it has become evident that bus interconnection technology will not be able to keep up with the improvements in microprocessor technology." "The electrical factors are however more serious as they present physical limitations to increasing bus bandwidth. Wires in a bus interconnect have multiple taps, each tap being able to drive and sense the voltage level in the wire. At very high speeds each tap introduces stray impedances that cause reflection and signal attenuation, resulting in longer settling times. Moreover, the length of the wires on backplane a bus increases somewhat linearly with the number of taps, because of the physical spacing necessary to plug in printed circuit boards. Longer wires also translates into longer settling times as the signal has to travel the length of the bus. Since a transmitter has to wait until the signal has safely settled before driving another data, these effects directly bound the minimum bus clock period. Attempts to improve bus clock frequency typically involve increasing the current levels and/or reducing the voltage swing, so to improve signal rise time. Both approaches have limitations. Increasing currents will worsen switching effects, such as ground bounce and crosstalk interference. Reducing the voltage swing makes the bus less noise immune.")

- Pong: *See, e.g.*, p. 3-4 ("First and foremost, the multi-drop bus architecture is reaching its speed limit. When the clocking speed was low, the electrical length of the bus was short enough that distributed behavior of the bus could be ignored. However, as bus speeds increase, the processor boards connected to the bus behave as stubs resulting in reflections and ringing of bus signals. There exist several schemes for terminations and signaling to reduced reflections, but none solves the fundamental problem of stubs. Because of design constraints such as heat dissipation the space needed between stubs is longer at high speeds." "For future processor designs with deep speculation, multiple cores, and/or multithreading [13, 14, 23], the shared-bus will no doubt become a major bottleneck even in small multiprocessor configurations.")

- "Timestamp Snooping: An Approach for Extending SMPs," Martin: *See, e.g.*, p. 2 ("More than a decade ago, Agarwal et al. [2] predicted that limited bus bandwidth would lead to the demise of SMPs. They argued that directory-based coherence protocols provide better scalability by sending memory transactions over a point-to-point network to a directory (usually at memory) that redirects the request either to memory (trivial) or to other processors.")

- "Interconnection Networks an Engineering Approach," Duato (1997): *See, e.g.*, p. 11 ("Scalability is an important issue in designing multiprocessor systems. Bus-based systems are not scalable because the bus becomes the bottleneck when more processors are added. The *direct network* or *point-to-point network* is a popular network architecture that scaled well to a large number of processors.")

- U.S. Patent No. 5,796,605 to Hagersten: *See, e.g.*, 1:59-2:10 ("Unfortunately, shared bus architectures suffer from several drawbacks which limit their usefulness in multiprocessing computer systems. A bus is capable of a peak bandwidth (e.g. a number of bytes/second which may be transferred across the bus). As additional processors are attached to the bus, the bandwidth required to supply the processors with data and instructions may exceed the peak bus bandwidth. Since some processors are forced to wait for available bus bandwidth, performance of the computer system suffers when the bandwidth requirements of the processors exceeds available bus bandwidth." "Additionally, adding more processors to a shared bus increases the capacitive loading on the bus and may even cause the physical length of the bus to be increased. The increased capacitive loading and extended bus length increases the delay in propagating a signal across the bus. Due to the increased propagation delay, transactions may take longer to perform. Therefore, the peak bandwidth of the bus may decrease as more processors are added.")

Similarly, one of ordinary skill would have understood the advantages and disadvantages of the additional topologies described above:

- Chaiken: *See, e.g.*, p. 49-50 ("Consequently, scalable multiprocessor systems interconnect processors using short point-to-point wires in direct or multistage networks. Communications along impedance-matched transmission line channels can occur at high speeds, providing communication bandwidth that scales with the number of processors. Unlike buses, the bandwidth of these networks increases as more processors are added to the system.")

- Barroso: *See, e.g.*, p. 9 ("The alternatives to a bus interconnect are topologies that rely on a non-shared physical communication medium. All such topologies therefore use point-to-point links as building blocks. Point-to-point links have several attractive features and have been growing increasingly popular in the last few years. By having only a single driver and a single receiver at each end of the wire, point-to-point links are much simpler electrically than buses. Point-to-point links are easy to terminate properly because there is only one termination point. Better termination and lower characteristic impedances allow fast signal rise time and propagation speed with lower driving currents, which makes it easier to use low voltage signaling while still maintaining reasonable noise immunity.")

- Barroso: *See, e.g.*, p. 10 ("Overall, point-to-point connections are more technologically scalable than bus connections, and their delivered bandwidth is expected to benefit continuously from improvements in circuit technology.")

- Hwang: *See, e.g.*, p. 298 ("Much higher bandwidth can be provided by a crossbar switched network for the same datapath width per port and equal number of connecting ports.")

- Hwang: *See, e.g.*, p. 300 ("A bus-connected multiprocessor is limited by its bus bandwidth. Very often, the bus becomes the bottleneck in accessing the shared memory by a large number of processors. A better approach is to replace the interprocessor memory bus by a crossbar switch as illustrated in Fig. 6.19.")

- Hwang: *See, e.g.*, p. 328 ("To understand the motivation for SCI [Scalable Coherence Interface]. We need to look at three problems of the bus (see Fig. 6.35) when building a scalable system.

  Signaling Problem. Bus transmission lines are not perfect because they have taps to connect various processor, memory, and I/O devices. This causes reflections and introduces noise, especially since bus drivers require lots of current.

  Bottleneck Problem. Bus is a shared medium which can be used by only one transmitter at a time. Split-transaction protocols help a little. But bus arbitration and addressing always need to be done for each transaction.

  Size Problem. Due to signaling difficulties, a fast bus must be short. The short bus limits the number of devices that can be connected to it.")

- Hwang: *See, e.g.*, p. 328-329 ("SCI adopts the following techniques to overcome the three problems associated with the bus, as shown in Fig. 6.35:

  Point-to-Point Link. SCI views various processor, memory, and I/O devices as nodes, and uses a point-to-point link from one sender node to a receiver node, with differential signaling. There are no connection taps any more, and the reflection/noise problem is significantly alleviated, allowing a large increase in signaling speed. SCI does not rule out more complex nodes that contain processors, memory, and I/O devices.

  Unidirectional Ring. The links are run continuously and in one direction. This makes the driver current remain constant, further reducing noise. Since every node must have at least one input link and one output link, a unidirectional ring is the simplest topology.

  Parallelism. Unlike the bus where only one transaction can use it at a time, multiple nodes can inject and extract packets to the SCI ring simultaneously.")

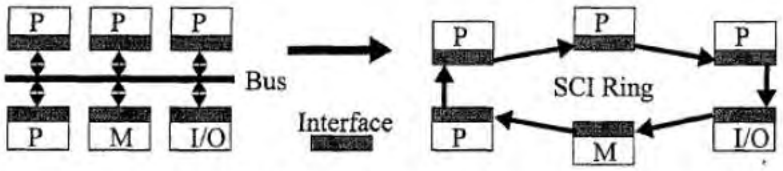


Figure 6.35 Evolution of an SCI ring from a digital bus (Courtesy D. V. James, et al. *IEEE Computer* [348], 1990)

Hwang, Figure 6.35

- Patterson: *See, e.g.*, p. 736-737 ("The straightforward way to connect processor-memory nodes is to have a dedicated communication link between every node.... The first improvement over a bus is a network that connects a sequence of nodes together.... This topology is called a ring." "At the other extreme from a ring is a fully connected network, where every processor has a bidirectional link to every other processor…. Real machines frequently add extra links to these simply topologies to improve performance and reliability.")

- Grindley: *See, e.g.*, p. 2 ("Unidirectional slotted rings were chosen for a number of reasons. First, they can perform as well as meshes for up to 128 processors [17, 18] when some data locality is present. Second, stations can be added one at a time without significant re-wiring or topology changes, making them highly modular and cost-effective. Third, rings exhibit two features useful for implementing cache coherence and memory consistency: inherent sequencing of requests/responses, and natural broadcast capabilities. For example, a single request invalidates multiple copies of a cache line as it traverses the ring hierarchy, and serves as an acknowledgment upon its return to the source of the request.")

- "Virtual Reality: Scientific and Technological Challenges," Committee on Virtual Reality Research and Development, Computer Science and Telecommunications Board, National Research Council (1995): *See, e.g.*, p. 351 ("Point-to-point links circumvent the bottleneck problem of bus-based architectures to some extent: communication between connected point will be fast, but that between distant nodes in a graph will be slow."

- Jamil: *See, e.g.*, 1:16-28 ("Joining several processors in parallel increases processing capacity. Typically, any number from two to eight processors may be joined in parallel. Generally, multiple parallel processors are joined together on a shared bus. FIG. 1 illustrates a four processor (4P) architecture used in conjunction with a shared bus. Four processors, Processor 1, Processor 2, Processor 3, and Processor 4, connect to a shared bus, which in turn connects to the Northbridge chipset. The Northbridge chipset further connects to the Southbridge chipset and external memory. For example, a Pentium™ processor may employ the shared bus architecture illustrated in FIG. 1. However, a point-to point architecture, typically, provides a higher bandwidth than does a shared bus architecture.")

- Jamil: *See, e.g.*, 1:29-45 ("In a shared bus architecture, multiple devices all share the same bus and must follow an order and protocol to use the bus. In contrast, a point-to-point bus architecture provides an uninterrupted connection between two separate devices. Thus, in general, a point-to-point bus creates a higher bandwidth between two separate devices. A higher bandwidth can have the beneficial effect of yielding an increased performance from a single processor or group of processors. For example, if a 48-bit connection exists between two devices, then transactions occur between the two devices three times faster than if only a 16-bit connection exists between the two devices. However, a point-to-point bus architecture may have a disadvantage because the architecture provides an uninterrupted connection between two separate devices. Thus, if at any given time, light transfers of information occur between the two devices, then the excess bandwidth capacity is essentially wasted.")

Accordingly, it would have been obvious to implement a shared bus topology when the desired multiprocessor system design requires simplicity and low cost. However, one of ordinary skill would have been motivated to choose, for example, a point-to-point architecture instead of a shared bus topology for a particular multiprocessor system to avoid the inherent limitations of a shared bus (*e.g.*, low bandwidth, limited scalability, latency, bottleneck, limited processors, etc.) and benefit from the above-noted advantages of a point-to-point architecture (*e.g.*, increased bandwidth, additional processors, reduced latency, and improved reliability, etc.) with a reasonable expectation of success. Indeed, it would have been obvious to implement a point-to-point architecture instead of a shared bus since such an implementation would be a simple substitution of one known element (*e.g.*, a point-to-point architecture) for another (*e.g.*, a shared bus) to obtain predictable results (*e.g.*, multiprocessor system having increased scalability). It would also have been obvious to implement a point-to-point architecture instead of a shared bus because such a modification would simply be the use of a known technique (*e.g.*, a point-to-point architecture) to improve similar devices (*e.g.*, multiprocessor system having a shared bus) in the same way (*e.g.*, increase bandwidth, improve scalability, etc.). Furthermore, to the extent not disclosed, a person of ordinary skill in the art at the time of the alleged invention of the Asserted Claims would have been motivated to modify the prior art references identified in Section III and Exhibits A-1 – A-9; B-1 – B-19; C-1 – C-8; D-1 – D-14; and Exhibits E-1 – E-14 to include a "point-to-point architecture."

2. *"Speculative" Probing*

Some of the claims of the Asserted Patents require operations to be performed "speculatively." For example, claim 1.8 of the '409 patent recites "send a probe to the first plurality of processors in the first cluster before the cache access request is received by a serialization point in the second cluster," claim 6.8 recites "send a probe to the first plurality of processors in the first cluster before a memory line associated with the cache access request is locked," and claim 7.5 recites "speculatively probe a local node." *See also*, *e.g.*, '409 patent claims 8.1, 10.1, 12.1, 25.4, 26.1, 28.1, 30.1, 34.4, 37.1, 42.5, 42.6, 45.1, 51.4, and 52.4. Further, claim 15.8 of the '636 patent recites "send a probe to a third cluster including a third plurality of processors before the cache access request is received by a serialization point in the second cluster," claim 12.1 recites "sending a probe to a remote cluster cache coherence controller before a memory line associated the probe is locked," and claim 22.5 recites "speculatively probe a remote node in the remote cluster." *See also*, *e.g.*, '636 patent claims 11.5, 14.1, 21.8, 23.1, 27.1, and 36.5. At least under Memory Integrity's apparent infringement theories, performing operations "speculatively" was well-known in the art before the priority dates of the Asserted Patents. *See, e.g.*, Exhibits A-1 – A-9, claims 1.8, 6.8, 7.5, 8.1, 10.1, 12.1, 25.4, 26.1, 28.1, 30.1, 34.4, 37.1, 42.5, 42.6, 45.1, 51.4, and 52.4; and Exhibits B-1 – B-19, claims 11.5, 12.1, 14.1, 15.8, 21.8, 22.5, 23.1, 27.1, and 36.5. The following discussion shows that, at least under Memory Integrity's apparent infringement theories, it was well known and conventional before the priority dates of the Asserted Patents to perform operations "speculatively."

a) Local "Speculative" Probing Was Known

At least under Memory Integrity's apparent infringement theories, there are many prior art references that disclose "send[ing] a probe to the first plurality of processors in the first cluster before the cache access request is received by a serialization point in the second cluster," "send[ing] a probe to the first plurality of processors in the first cluster before a memory line associated with the cache access request is locked," and/or "speculatively prob[ing] a local node." Examples of prior art references that disclose and further demonstrate that such was well known include:

- 870 Chipset System Architecture Specification: [redacted]

- "Multi-Ring Performance of the Kendal Square Multiprocessor," Dunigan: *See, e.g.*, p. 4-5 ("We measured the latency from the cache to the subcache at about 1 μs, in close agreement with the number of cycles stated by KSR in Table 2.1. (A cycle is 0.05 μs.) If the datum is not found in the local cache, the processor stalls and a request is issued on the local ring. We measured the latency between two processors on the same ring to be 6.9 μs, which gives a data rate of 18.6 MB/second with the 128-byte data packet. The rate is measured to the subcache." "If the data item is not on the local ring, the request packet is routed to an appropriate ACE:0 via the interconnecting ring (ACE:l). Thus a request packet that must travel to another ring, traverses three rings. The measured latency is 24.7 µs with a corresponding data rate of 5.2 MB/second." "If a KSR processor does not find data in its cache, then the resulting latency or access time will depend on whether the data is found on the local ring or a remote ring. If *p* processors are being used in the parallel application, and the needed data item is equally likely to reside on any of the *p* - 1 other processors, then we can calculate the expected access time for a cache-miss on a multi-ring KSR (Figure 3.1). For a single ring ($p < 33$), the access time is just 6.9 μs. If $p > 32$, then the expected remote access time grows asymptotically toward 24.7 μs. (If another level of the KSR hierarchy were available (ACE:2), another two rings would be traversed, and we conjecture that the curve would ramp up again, asymptotically approaching 35 μs.) Although a function of the application, this increasing access time could cause the performance of an application to degrade as processors axe added. Remote access times for other scalable shared memory multiprocessors (DASH [14]and DDM [10]) also grow with the number of processors. Non-scalable shared memory multiprocessors (Cray Y-MP, Sequent, Encore) have flat remote access times.")

- "The Stanford Dash Multiprocessor," Lenoski (1992): *See, e.g.*, p. 68 ("The Dash memory system can be logically broken into four levels of hierarchy, as illustrated in Figure 3. The first level is the processor's cache. This cache is designed to match the processor speed and support snooping from the bus. A request that cannot be serviced by the processor's cache is sent to the second level in the hierarchy. the local cluster. This level includes the other processors' caches within the requesting processor's cluster. If the data is locally cached, the request can be serviced within the cluster. Otherwise, the request is sent to the home cluster level. The home level consists of the cluster that contains the directory and physical memory for a given memory address. For many accesses (for example, most private data references), the local and home cluster are the same, and the hierarchy collapses to three levels. In general, however, a request will travel through the interconnection network to the home cluster. The home cluster can usually satisfy the request immediately, but if the directory entry is in a dirty state, or in shared state when the requesting processor requests exclusive access, the fourth level must also be accessed. The remote cluster level for a memory block consists of the clusters marked by the directory as holding a copy of the block.")

- "Power Efficient Cache Coherence," Saldanha: *See, e.g.*, p. 2 ("Various forms of speculation are routinely employed to reduce the latency of cache misses and overlap data fetch and transmission latency with checking for cache coherence.")
- Saldanha: *See, e.g.*, p. 3 ("Only if the node determines that it cannot satisfy a request for data locally, will it attempt to satisfy the request from a remote node or memory.")
- "Comparative Modeling and Evaluation of CC-NUMA and COMA on Hierarchical Ring Architectures," Zhang (1995): *See, e.g.*, p. 5 ("Reading the shared-data – the processor will get the data in its local memory if it is available there, otherwise it will get it from one of the memory modules in the local ring, or in a remote ring through searching.")
- U.S. Patent 5,297,265 of Frank: *See, e.g.*, 3:64-4:16 ("Data access requests generated by a processor are handled by the local memory element whenever possible. More particularly, a controller coupled with each memory monitors the cell's internal bus and responds to local processor requests by comparing the request with descriptors listed in the corresponding directory. If found, matching data is transmitted back along the internal bus to the requesting processor." "Data requests that cannot be resolved locally are passed from the processing cell to the memory management system. The management element selectively routes those unresolved data requests to the other processing cells. This routing is accomplished by comparing requested descriptors with directory entries of the domain routing units. Control elements associated with each of those other cells, in turn, interrogate their own associated directories to find the requested data. Data satisfying a pending request is routed along the domain segment hierarchy from the remote cell to the requesting cell.")
- U.S. Patent No. 5,795,605 to Hagersten: *See, e.g.*, Abstract ("A technique for system memory space address mapping in a multiprocessor computer system is provided. The disclosed mapping architecture may be applied to a multiprocessor computer system having multiple processing nodes (SMP nodes), where each processing node may include multiple processors. The system memory address space is split into different regions such that each of the n SMP nodes is assigned 1/n of the total address space. Cache coherency state information is stored for the memory in each SMP node. Memory regions may further be assigned to operate in one of three modes: normal, migratory, or replicate. When operating in normal mode, transaction to an address space assigned to a particular node are tried only locally in that node first. Transactions may be sent globally to other nodes if an improper cache coherency state is returned or if the address corresponds to a memory region assigned to another node. In migratory mode transactions are always sent globally. And in replicate mode duplicate copies of the replicate memory region are assigned to each SMP node so that transactions are always tried locally first, and only sent globally if an improper cache coherency state is returned.")
- U.S. Patent No. 6,081,874 to Carpenter: *See, e.g.*, 12:20-28 ("In such embodiments, waiting to speculatively source request transactions until after the AStatIn votes are received advantageously minimizes the number of speculatively sourced transactions

that must subsequently be killed at the target processing node 10. In addition, forwarding an indication of the AStatIn vote is unnecessary, and the operation shown at block 96 (as well as status channel 19) can be omitted, as indicated by dashed-line illustration.")

- U.S. Patent No. 6,516,391 to Tsushima: *See, e.g.*, 1:46-65 ("(1). A cache miss occurred in a processor, and a memory access request used to access this relevant address is produced. (2). To send a memory access request to a memory, a processor joins a bus arbitration in order to acquire a use right of a system bus. (3). If the system bus use right is acquired, then the memory access request produced at (1) is sent to the system bus. At this time, other processors coupled to the system bus check as to whether or not data of an address contained in the memory access request is cached, and then returns a checked result to such a processor which issues the memory access request. (4). When as a result of (3), the memory access operation must be carried out, the memory access request is sent out to the network. As to sending of the memory access request to the network, there are some cases that the arbitration for obtaining the use right must be carried out similar to (2), depending upon the structure.")

- Tsushima: *See, e.g.*, 10:45-49 ("When a memory read command is issued in the source node, a local snoop is carried out within the source node and it can be seen that the memory access is required. Thereafter, the memory access command is transferred to both the snoop node and the target node.")

- U.S. Patent No. 6,799,217 to Wilson: *See, e.g.*, 2:46-57 ("The present invention also includes memory accesses with and without pipelining. More particularly it includes local and remote coherence protocols that permit legal transactions for dual and multi-node systems. In a pipelined environment, the present invention increases overall system speed for data access because there is a latency reduction. For example, the present invention allows for a speculative snoop and a speculative memory access to occur even as a local memory access for data is occurring. Further, when a directory determines that data resides remotely, it does not need to wait for a follow-up to begin access of this data. This increases overall system efficiency and reduces latency.")

b) Remote "Speculative" Probing Was Known

At least under Memory Integrity's apparent infringement theories, there are many prior art references that disclose "send[ing] a probe to a third cluster including a third plurality of processors before the cache access request is received by a serialization point in the second cluster," "sending a probe to a remote cluster cache coherence controller before a memory line associated the probe is locked," and/or "speculatively prob[ing] a remote node in the remote cluster." Examples of prior art references that disclose and further demonstrate that such was well known include:

- "Design and Performance of the Software-controlled COMA," Moga: *See, e.g.*, p. 110 ("In COMA, the mastership for a line 'migrates' to one of the nodes which have the line in their working set. Due to the round-robin data placement, often times the master and the home node for a line are not one and the same. The consequence is that most remote accesses complete in three hops as the home node must be used as an intermediary to forward the request to the current master. This increases the average remote latency and, in SC-COMA, also creates direct overhead at the home node. Had the local node been able to guess the identity of the master, the request could be sent directly, thus bypassing the home node. If the guess is correct, the access completes in two hops; if it is wrong, the protocol must fall back on the original scheme, thus incurring some overhead and further increase in latency.")

- Saldanha: *See, e.g.*, p. 2 ("Various forms of speculation are routinely employed to reduce the latency of cache misses and overlap data fetch and transmission latency with checking for cache coherence.")

- Zhang: *See, e.g.*, p. 5 ("Writing the shared-data – the processor will either write the data locally if it is available or will do a remote-write in the destination memory module. The associated invalidation operations are defined as follows." "The invalidation of shared data is conducted during the process of a write request traveling to the home node and returning from the home node. Each time a write request passes a global directory which has copies of the requested data, it will produce a invalidation packet to invalidate the copies in the local ring.")

- Carpenter: *See, e.g.*, 1:8-15 ("The present invention relates in general to a method and system for data processing and, in particular, to a non-uniform memory access (NUMA) data processing system and method of communication in a NUMA data processing system. Still more particularly, the present invention relates to a NUMA data processing system and method of communication in which requests are speculatively issued on a node interconnect to reduce communication latency.")

- Carpenter: *See, e.g.*, 13:40-47 ("In accordance with the present invention, communication transactions are speculatively issued to a remote target processing node on the node interconnect prior to the completion of the communication transaction on the local interconnect of a source processing node. In this manner, the latency of communication transactions can be dramatically reduced.")

- U.S. Patent No. 6,711,662 to Peir: *See, e.g.*, Abstract ("A shared-memory system includes processing modules communicating with each other through a network. Each of the processing modules includes a processor, a cache, and a memory unit that is locally accessible by the processor and remotely accessible via the network by all other processors. A home directory records states and locations of data blocks in the memory unit. A prediction facility that contains reference history information of the data blocks predicts a next requester of a number of the data blocks that have been referenced recently. The next requester is informed by the prediction facility of the current owner of the data block. As a result, the next requester can issue a request to the current owner directly without an additional hop through the home directory.")

- Wilson: *See, e.g.*, 2:46-57 ("The present invention also includes memory accesses with and without pipelining. More particularly it includes local and remote coherence protocols that permit legal transactions for dual and multi-node systems. In a pipelined environment, the present invention increases overall system speed for data access because there is a latency reduction. For example, the present invention allows for a speculative snoop and a speculative memory access to occur even as a local memory access for data is occurring. Further, when a directory determines that data resides remotely, it does not need to wait for a follow-up to begin access of this data. This increases overall system efficiency and reduces latency.")

- Wilson: *See, e.g..*, at 7:62-8:6 ("In a multi-node system, the determination 514 of remote coherence includes a directory look-up at the local node to determine if the data is located at a remote node. In a two-node system, the determination 514 of remote coherence includes either a speculative snoop operation or a speculative memory access operation at the remote node. In a two-node system, there is a speculative snoop operation that includes a snoop of remote caches of the remote processors at the remote nodes. This operation is referred to as speculative because the remote node is queried for data while the local node is also queried for data. This may be referred to as snooping.")

- Wilson: *See, e.g..*, at 8:27-36 ("The determination 534 of remote coherence functions in one of two manners. For multi-node systems it includes a directory look-up to determine if the data is located at a remote node. This may be referred to as a speculative directory look-up. In a two-node system there is either a speculative snoop operation or a speculative memory access operation of the remote node. The speculative snoop operation includes a snoop of remote caches of the remote processors at the remote nodes. The speculative memory access operation accesses the local memory system.")

- U.S. Patent No. 6,883,070 to Martin: *See, e.g.*, 5:52-65 ("Referring now to FIG. 2, when a snooping mechanism is used for the transmission of cache coherence messages, for example, from a processor unit 12 a, the cache coherence message is duplicated and broadcast over the ordered request network 28 to each of the remaining processor units 12 b through 12 f and the memory controller 11 of the shared memory system 16 as indicated by the arrows 23 of FIG. 2. When the cache coherence message is a request for a block 19, that cache memory 22 owning the block 19 (or the shared memory system 16 if it is the owner) responds by relinquishing the block 19 to the cache memory 22 of the requesting processor unit 12 a. Snooping is rapid, but requires a large number of messages as is apparent from FIG. 2.")

- Martin: *See, e.g.*, 9:16-27 ("Referring also to FIG. 8, in this embodiment, the cache controller 26 is augmented by a predictor 98, which endeavors to predict those processors units 12 a through 12 f likely to have copies of the block 19 being sought. The predictor 98 may make its predictions in a number of ways including, for example, storing information about recent mispredictions to the same block 19, recent mispredictions to any block 19, behavior of spatially adjacent blocks 19, recent mispredictions of the same static load or store instructions (indexed to the program counter), input form the software (the programmer, compiler, library or runtime system or some combination of these).")

- U.S. Patent No. 7,234,029 to Khare: *See, e.g.*, Abstract ("A method for reducing memory latency in a multi-node architecture. In one embodiment, a speculative read request is issued to a home node before results of a cache coherence protocol are determined. The home node initiates a read to memory to complete the speculative read request. Results of a cache coherence protocol may be determined by a coherence agent to resolve cache coherency after the speculative read request is issued.")

As illustrated by the prior art references above, it was well known before the priority dates of the Asserted Patents to perform local and/or remote operations "speculatively," at least under Memory Integrity's apparent infringement theories.[1]

c) Obvious to Perform Operations "Speculatively"

Furthermore, a person of ordinary skill in the art would have understood the problem of latency in a multiprocessor system:

- "Hiding Memory Latency using Dynamic Scheduling in Shared-Memory Multiprocessors," Gharachorloo: *See, e.g.*, p. 1 ("The large latency of memory accesses is a major impediment to achieving high performance in large scale shared-memory multiprocessors.")
- Culler: *See, e.g.*, p. 37 ("The effectiveness of the shared memory approach depends on the latency incurred on memory accesses as well as the bandwidth of data transfer that can be supported. Just as a memory storage hierarchy allows data that is bound to an address to be migrated toward the processor, expressing communication in terms of the storage address space allows shared data to be migrated toward the processor that accesses it.")
- U.S. Patent No. 6,775,749 to Mudgett: *See, e.g.*, 2:44-52 ("In some systems, the time required to maintain cache coherency (e.g., the time required to send probes and receive responses) may be significant. The total time taken to perform a cache fill may depend on the latency of both the cache coherency mechanism and that of the memory

[1] Additional latency reducing techniques that are speculative in nature include, for example: prefetching, multithreading, and out-of-order execution. *See, e.g.*, Culler, Chapter 11; "Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing," Barroso; U.S. Patent No. 6,457,101; The "POWER4 Processor Introduction and Tuning Guide," Behling; Intel 870 Chipset; and Intel Profusion Chipset.

system. As a result, the time spent maintaining cache coherency may significantly affect performance. Accordingly, one drawback of sharing memory between devices that have caches is that cache fill performance may decrease.")

- U.S. Patent No. 6,615,322 to Arimilli: *See, e.g.*, 2:23-31 ("The present invention recognizes that, while the conventional NUMA architecture illustrated in FIG. 1 can provide improved scalability and expandability over conventional SMP architectures, the conventional NUMA architecture is subject to a number of drawbacks. First, communication between nodes is subject to much higher latency (e.g., five to ten times higher latency) than communication over local interconnects 11, meaning that any reduction in inter-node communication will tend to improve performance.")

- Arimilli: *See, e.g.*, 2:41-44 ("A second drawback of conventional NUMA computer systems related to inter-node communication latency is the delay in servicing requests caused by unnecessary inter-node coherency communication.")

- Carpenter: *See, e.g.*, 2:16-26 ("A principal performance concern with CC-NUMA computer systems is the latency associated with communication transactions transmitted via the interconnect coupling the nodes. Because all data accesses can potentially trigger a coherency or data request transaction on the nodal interconnect, the latency associated with the transmission of requests to remote nodes and transmission of the responses from the remote nodes can dramatically influence overall system performance. As should thus be apparent, it would be desirable to provide a CC-NUMA computer system having low inter-node communication latency.")

As a result, one of ordinary skill in the art would have been motivated to perform local and/or remote operations "speculatively" to address this latency and improve performance:

- "Scalable Shared Memory Multiprocessing," Lenoski (1995): *See, e.g.*, p. viii ("Designing a large-scale, shared-memory machine requires attention to minimizing memory latency and hiding that latency whenever possible.")

- Moga: *See, e.g.*, p. 1 ("Computer architects bridge the gap between computation and communication speeds with advanced hierarchical memory designs and latency-tolerant techniques which overlap computation and communication. They also balance the entire computing system to avoid bottlenecks and to achieve scalability.")

- Culler: *See, e.g.*, p. 155 ("Techniques to hide communication latency come in different, often complementary flavors, and we shall examine them in Chapter 11. One approach is simply to make messages larger, thus incurring the latency of the first word but hiding that of subsequent words through pipelined transfer of the large message. Another approach, which we can call precommunication, is to initiate the communication well before the data is actually needed, so that by the time the data is needed it is likely to have already arrived. A third technique is to initiate the communication where it naturally belongs in the program but to hide its cost by finding something else for the processor to do--some computation or other communication that occurs later in the same process--while the communication is in progress.")

- Culler: *See*, *e.g.*, p. 568 ("In addition to forwarding, other protocol optimizations to reduce latency include overlapping transactions and activities by performing them speculatively.")

- Culler: *See*, *e.g.*, p. 668 ("Another advantage is that upon a miss, if a nearby node in the hierarchy has a cached copy of the block, then the block can be obtained from that nearby node (cache-to-cache sharing) rather than having to go to the home, which may be much further away in the network topology. This can reduce transit latency as well as contention at the home.")

- Culler: *See*, *e.g.*, p. 836-838 ("There are four key approaches to exploiting this overlap of hardware resources and thus tolerating latency … Precommunication: Generating the communication before the point where the operation naturally appears in the program so that it is partially or entirely completed before data is actually needed can be done either in software, by inserting a precommunication operation earlier in the code, or in hardware, by detecting the opportunity and issuing the communication operation early."

- "Using Prediction to Accelerate Coherence Protocols," Mukherjee: *See*, *e.g.*, p. 1 ("Directory protocols maintain a directory entry per memory block that records which processor(s) currently cache the block. On a miss, a processor sends a coherence message over an interconnect to a directory, which often forwards message(s) to processor(s) currently caching the block. These processors may forward data or acknowledgments to the requesting processor and/or directory." "Unfortunately, this cache miss and directory activity can disturb a programmer's performance model of shared memory by making some memory accesses tens to hundreds of times slower than others. This problem has led to many proposals, including weaker memory models [2], multithreading [36], non-blocking caches [18], and application specific coherence protocols [27]." "Another class of proposals predict future sharing patterns [7, 13] and take actions to overlap coherence message activity with current work.")

- "CapNet – Using a Gigabit Network As A High Speed Backplane," Tam: *See*, *e.g.*, p. 1 ("The key to realize this system is to reduce the effect of latency as much as possible.")

- Frank: *See*, *e.g.*, 2: 27-31 ("It is therefore an object of this invention to provide an improved multiprocessing system with improved data coherency, as well as reduced latency and bus contention. A further object is to provide a multiprocessing system with unlimited scalability.")

Accordingly, it would have been obvious to perform local and/or remote operations "speculatively" to address the issue of latency and benefit from the above-noted advantages with a reasonable expectation of success. Indeed, it would have been obvious to perform local and/or remote operations "speculatively" instead of performing operations in a "non-speculative" manner since such an implementation would be a simple substitution of one known element (*e.g.*, performing operations "speculatively") for another (*e.g.*, performing operations "non-speculatively") to obtain predictable results (*e.g.*, multiprocessor system having reduced latency). Further, it would have been obvious to perform local and/or remote operations "speculatively" because such a modification would simply be the use of a known technique (*e.g.*, performing operations "speculatively") to improve similar devices (*e.g.*, multiprocessor system) in the same way (*e.g.*, reduce latency). Furthermore, to the extent not disclosed, a person of ordinary skill in the art at the time of the alleged invention of the Asserted Claims would have been motivated to modify the prior art references identified in Section III and Exhibits A-1 – A-9; B-1 – B-19; C-1 – C-8; D-1 – D-14; and E-1 – E-14 to perform local and/or remote operations "speculatively."

3. *"Determining if Speculative Probing of the Local Node Can Be Performed"*

Some of the claims of the Asserted Patents require "determine if speculative probing of the local node can be performed." For example, claim 12.1 of the '409 patent recites "determine if speculative probing of the local node can be performed." *See also*, *e.g.*, '409 patent claims 25.4, 30.1, and 34.4. At least under Memory Integrity's apparent infringement theories, "determine if speculative probing of the local node can be performed" was well-known in the art before the priority dates of the Asserted Patents. *See*, *e.g.*, Exhibits A-1 – A-9, claims 12.1, 25.4, 30.1, and 34.4. The following discussion shows that, at least under Memory Integrity's apparent infringement theories, it was well known and conventional before the priority dates of the Asserted Patents to "determin[e] if speculative probing of the local node can be performed."

At least under Memory Integrity's apparent infringement theories, there are many examples of prior art references that disclose "determining if speculative probing of the local node can be performed." Examples of prior art references that disclose and further demonstrate that such was well known include:

- 870 Chipset System Architecture Specification: [redacted]
- "The Stanford Dash Multiprocessor," Lenoski (1992): *See, e.g.*, p. 68 ("The Dash memory system can be logically broken into four levels of hierarchy, as illustrated in Figure 3. The first level is the processor's cache. This cache is designed to match the processor speed and support snooping from the bus. A request that cannot be serviced by the processor's cache is sent to the second level in the hierarchy. the local cluster. This level includes the other processors' caches within the requesting processor's cluster. If the data is locally cached, the request can be serviced within the cluster. Otherwise, the request is sent to the home cluster level. The home level consists of the cluster that contains the directory and physical memory for a given memory address. For many accesses (for example, most private data references), the local and home cluster are the same, and the hierarchy collapses to three levels. In general, however, a request will travel through the interconnection network to the home cluster. The home cluster can usually satisfy the request immediately, but if the directory entry is in a dirty state, or in shared state when the requesting processor requests exclusive access, the fourth level must also be accessed. The remote cluster level for a memory block consists of the clusters marked by the directory as holding a copy of the block.")
- "The GLOW Cache Coherence Protocol Extensions for Widely Shared Data," Kaxiras (1996): *See, e.g.*, p. 2 ("For example, in Figure 1, as far as the memory directory is concerned, it points to a number of sharing nodes, whereas in reality it points to the first level of agents that hold the rest of the sharing tree. Similarly as far as the nodes that truly share (the leaves of the sharing tree) are concerned, they have been serviced directly by memory where in fact they were serviced by the agents impersonating the memory." "The SCI lists are created under the agent when requests from nodes on a ring are intercepted and satisfied either directly by the agent or by another close-by node (usually on the same ring). These lists are called child lists and the agent is their parent. Without GLOW, requests would go all the way to the remote memory and would join a global list. As we have explained the agent has a dual personality: toward its children it behaves as if it were an SCI memory directory; toward its parent it behaves as if it were an ordinary SCI cache.")
- "Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing," Barroso (2000): *See, e.g.*, p. 285 ("A memory request from an L1 is sent to the appropriate L2

bank based on the address interleaving. Depending on the state at the L2, the L2 can possibly (a) service the request directly, (b) forward the request to a local (owner) L1, (c) forward the request to one of the protocol engines, or (d) obtain the data from memory through the memory controller (only if the home is local). The L2 is also responsible for all on-chip invalidations, whether triggered by local or remote requests. The ordering characteristics of the intra-chip switch allow us to eliminate the need for acknowledgments for on-chip invalidations. Invalidating and forwarding requests to remote nodes are handled through the protocol engines.")

- U.S. Patent No. 5,297,265 to Frank: *See, e.g.*, 3:64-4:16 ("Data access requests generated by a processor are handled by the local memory element whenever possible. More particularly, a controller coupled with each memory monitors the cell's internal bus and responds to local processor requests by comparing the request with descriptors listed in the corresponding directory. If found, matching data is transmitted back along the internal bus to the requesting processor." "Data requests that cannot be resolved locally are passed from the processing cell to the memory management system. The management element selectively routes those unresolved data requests to the other processing cells. This routing is accomplished by comparing requested descriptors with directory entries of the domain routing units. Control elements associated with each of those other cells, in turn, interrogate their own associated directories to find the requested data. Data satisfying a pending request is routed along the domain segment hierarchy from the remote cell to the requesting cell.")

- U.S. Patent No. 5,752,264 to Blake: *See, e.g.*, 12:23-39 ("The fourteenth operation sequence is the Fetch Exclusive Operation Sequence 4 (FEX-4). A store instruction by processor 0 finds the state of the storage unit to be `invalid` in the L1 cache. This results in a `Fetch Exclusive` request issued to the local L2. The L2 does a directory search for the processor request and determines the storage unit also exists in a state which is 'read-only' to the local L2. This results in a 'Storage Invalidation' request issued on the shared bus to the memory and to all other clusters. Also, a 'Invalidate' command is sent to all other local processors that might have a 'read-only' copy of the storage unit in their L1 caches. Each remote L2 then does a directory search with the bus command request. A status of 'invalid' state is found on all the remote L2'. With no data being expected from the bus, the requested storage unit is accessed from the local L2 cache and sent to processor 0. The L2 updates its directory with a new status of EEUO.")

- U.S. Patent No. 5,864,671 to Hagersten: *See, e.g.*, 18:49-65 ("The advantages of the hybrid protocol may be more fully understood with reference to FIG. 11 and FIG. 12. FIG. 11 is a flowchart illustrating, in one embodiment of the present invention, the steps involved at the home node in servicing a received memory access request. In step 1102, a memory access request pertaining to a home memory block is received from the network infrastructure. In step 1104, the method determines whether a directory entry corresponding to the requested memory block exists in the partial directory cache, e.g., partial directory cache 950 of FIG. 10. If a directory entry corresponding to the requested memory block already exists in the partial directory cache of the home node, the method advantageously employs the directory protocol to service the received memory access request (step 1106). The transition from a directory-less protocol to a directory protocol occurs when there is a partial directory cache hit in partial directory cache 950.")

- U.S. Patent Application No. 2002/0046324 to Barroso: *See, e.g.*, [0058] ("A memory request from an L1 cache is sent to the appropriate L2 bank based on the address interleaving. Depending on the state at the L2, the L2 can possibly: (a) service the request directly, (b) forward the request to a local (owner) L1, (c) forward the request to one of the protocol engines, or (d) obtain the data from memory through the memory controller (only if the home is local). The L2 is also responsible for all instances of on-chip invalidation, whether triggered by local or remote requests.")

- U.S. Patent Application No. 2002/0087804 to Khare: *See, e.g.*, [0037] ("According to one embodiment, computer system 100 includes a conflict detection mechanism for instances where coherent agents in computer system 100 generate transactions addressed to the same cache line. The mechanism orders the transactions in such a way that the coherency is not violated. In one embodiment, the detection and resolution of conflicts among concurrent requests from multiple nodes is done at SNC 210 and SP switch 230. As described above, concurrent accesses from multiple nodes to the same cache line creates a problem if the requests are conflicting in nature. Two requests are considered conflicting with each other if simultaneous processing of these requests will cause the system to get into an incoherent state, or result in loss of most up-to-date data.")

- U.S. Patent No. 6,457,087 to Fu: *See, e.g.*, 30:49-59 ("Referring to FIG. 34, the CIU-IFU 310 latches in the address and command from the snoop path (step 506) and decodes the address and command (step 508). The CIU-IFU 310 determines whether the address is in the lock buffer (step 510). If the requested address is contained in the lock buffer (step 510-Y), the interface unit activates the backoff signal (step 512). The presence of the requested address in the lock buffer indicates that another transaction is being processed that affects the cache line with the requested address. As such, the current transaction will be delayed or 'backoff' until the previous transaction completes.")

- U.S. Patent No. 6,684,297 to Chaudry: *See, e.g.*, 6:11-41 ("FIG. 6 is a flow chart illustrating the process of using reverse directory entries to perform invalidations in accordance with an embodiment of the present invention. The system starts by receiving a request that causes an update of L2 cache 106 (step 602). This request can include: a store hit on the target entry by another processor, a load miss, or a store miss." "Next, the system reconstructs the entry for the request in reverse directory 302. This is accomplished by performing a lookup in L2 cache 106 to determine the L2 way number 429 in which the target entry is located (step 604), and retrieving the L2 set number 404 from address 400 as is illustrated in FIG. 4 (step 606). These values are combined to construct the reverse directory entry 130." "Next, the system uses this entry to search reverse directory 302 in order to determine which L1 caches contain the entry (step 608). Note that the system only has to search the reverse directory that is associated a bank of L2 cache 206 that is specified by L2 bank number 406. Furthermore, the set number within the reverse directory can be determined from the address, which means that the search only has to consider entries in the four possible 'ways' for each set. Also note that if the request is a store hit by another processor, the system does not have to search the bank for the processor that caused the store hit." "For each L1 cache that contains the entry, the system sends an invalidation message to the L1 cache. This invalidation message includes the L1 way number, so that an associative lookup in the L1 cache can be avoided. The system also updates the corresponding reverse directory entry to indicate that it has been invalidated (step 610).")

- U.S. Patent No. 6,973,543 to Hughes: *See, e.g.*, Abstract ("A partial directory cache records addresses of blocks which are known to be cached in a non-exclusive state in any caches currently caching the blocks. If a read command to a block recorded in the partial directory cache is received, one or more probes corresponding to the command may be inhibited. Since probes are selectively inhibited if an affected block is recorded in the partial directory cache, the size of the partial directory cache may be flexible. If a particular block is not represented in the partial directory cache, probes are performed when the particular block is accessed (even if the particular block could have been represented in the partial directory cache). Thus, coherency is maintained even if every non-exclusively cached block is not represented in the partial directory cache.")

As illustrated by the prior art references above, it was well known before the priority dates of the Asserted Patents to "determin[e] if speculative probing of the local node can be performed" in multiprocessor systems, at least under Memory Integrity's apparent infringement theories. Indeed, a person of ordinary skill would have been motivated to "determin[e] if speculative probing of the local node can be performed" in a multiprocessor system as described below:

- Blake: *See, e.g.*, 3:40-57 ("Still another advantage of the cluster architecture is realized by defining a plurality of ownership states that convey information about how individual data units are stored in a cache. In a preferred embodiment of the invention, each cache, both level one and level two, includes a directory which maintains a record of the state for each data unit stored in that cache. The possible directory states for the level one caches are different from the possible directory states for the level two caches. However the possible ownership states for both levels are derived from two fundamental ownership states, 'exclusive', and 'read only'. When a cache has exclusive ownership of a data unit it may freely modify that unit without causing a cache coherency problem because it is the only cache currently storing a copy of that data unit. On the other hand, when a cache holds data in a read only state, one or more other caches are also holding the data, and the cache cannot modify the data without causing a cache coherency problem.")

- Chaudry: *See, e.g.*, 1:35-39 ("Note that coherence problems can arise if a copy of the same data item exists in more than one L1 cache. In this case, modifications to a first version of a data item in L1 cache 161 may cause the first version to be different than a second version of the data item in L1 cache 162.")

- Chaudry: *See, e.g.*, 1:40-47 ("In order to prevent coherency problems, computer systems often provide a coherency protocol that operates across bus 170. A coherency protocol typically ensures that if one copy of a data item is modified in L1 cache 161, other copies of the same data item in L1 caches 162-164, in L2 cache 180 and in memory 183 are updated or invalidated to reflect the modification.")

- Chaudry: *See, e.g.*, 1:48-59 ("In order to remedy this problem, some designers have begun to explore the possibility of maintaining directory information within L2 cache 180. This directory information specifies which L1 caches contain copies of specific data items. This allows the system to send invalidation information to only the L1 caches that contain the data item instead of sending a broadcast message to all L1 caches.")

- Hughes: *See, e.g.*, 1:61-2:3 ("A partial directory cache records addresses of blocks which are known to be cached in a non-exclusive state in any caches currently caching the blocks. If a read command to a block recorded in the partial directory cache is received, one or more probes corresponding to the command may be inhibited. System bandwidth which would be consumed by the probes may be conserved. Furthermore, since probes are inhibited, the latency of the command may be reduced since the command may be completed without waiting for any probe responses.")

- Khare: *See, e.g.*, para. [0003] ("One of the fundamental flaws of these existing memory sharing architectures is that a responding node, containing modified data for a cache line where the home storage location for the memory in question resides on a different node, is expected only to provide a passive response to a read request. No mechanism is built into the architectures to provide intelligent handling of the potential conflict between back-to-back read and write requests to the same line of memory. Therefore, a distributed mechanism for resolving cache coherence conflicts in a multiple processing node architecture is desired.")

Accordingly, it would have been obvious to "determin[e] if speculative probing of the local node can be performed" in a multiprocessor system having multiple clusters of processors while maintaining coherency with a reasonable expectation of success. It would also have been obvious to "determin[e] if speculative probing of the local node can be performed" because such a modification would simply be the use of a known technique (*e.g.*, "determining if speculative probing of the local node can be performed") to improve similar devices (*e.g.*, multiprocessor systems) in the same way (*e.g.*, improve performance while maintaining coherency). Furthermore, to the extent not disclosed, a person of ordinary skill in the art at the time of the alleged invention of the Asserted Claims would have been motivated to modify the prior art references identified in Section III and Exhibits A-1 – A-9; B-1 – B-19; C-1 – C-8; D-1 – D-14; and Exhibits E-1 – E-14 to "determin[e] if speculative probing of the local node can be performed."

4. *"Clusters"*

Some of the Asserted Claims are directed to clusters of processing components. For example, claim 1.2 of the '409 patent recites "a first cluster." *See also*, *e.g.*, '409 patent claims 1.4, 1.8, 2.1, 6.2, 6.4, 6.8, 7.2, 7.4, 7.5, 8.1, 9.1, 10.1, 11.1, 12.1, 25.3-25.6, 26.1, 27.1, 28.1, 29.1, 30.1, 34.3-34.6, 36.1, 37.1, 38.1, 42.2, 42.3, 42.5, 42.6, 45.1, 51.2, 51.4, 52.2, and 52.4; '636 patent claims 11.2-11.5, 12.1, 13.1, 14.1, 15.2, 15.4, 15.8, 16.1, 18.1, 21.2, 21.4, 21.8, 22.2, 22.4, 22.5, 23.1, 24.1, 25.1, 26.1, 27.1, 28.1, 33.1, 34.1, 35.1, and 36.2-36.5; '121 patent claims 1.2-1.5, 2.1, 3.1, 4.1, 4.2, 5.1, 8.1, 11.1, 13.1, 14.1, 15.1, 16.2-16.5, 25.2-25.4, and 25.6-25.8; '206 patent claims 1.2-1.4, 1.6, 2.1, 19.1-19.3, 21.2, 21.3, 30.1, 30.2, 31.1, 32.1, 34.1, 35.1, 37.1, 38.1, 39.2, and 39.3; and '254 patent claims 1.1-1.11, 2.1, 3.1, 5.1, 6.1, 6.2, 7.1, and 8.1. At least under Memory Integrity's apparent infringement theories, clusters of processing components, sometimes referred to as nodes, were well-known in the art before the priority dates of the Asserted Patents. *See*, *e.g.*, Exhibits A-1 – A-9, claims 1.2, 1.4, 1.8, 2.1, 6.2, 6.4, 6.8, 7.2, 7.4, 7.5, 8.1, 9.1, 10.1, 11.1, 12.1, 25.3-25.6, 26.1, 27.1, 28.1, 29.1, 30.1, 34.3-34.6, 36.1, 37.1, 38.1, 42.2, 42.3, 42.5, 42.6, 45.1, 51.2, 51.4, 52.2, and 52.4; Exhibits B-1 – B-19, claims 11.2-11.5, 12.1, 13.1, 14.1, 15.2, 15.4, 15.8, 16.1, 18.1, 21.2, 21.4, 21.8, 22.2, 22.4, 22.5, 23.1, 24.1, 25.1, 26.1, 27.1, 28.1, 33.1, 34.1, 35.1, and 36.2-36.5; C-1 – C-8 claims 1.2-1.5, 2.1, 3.1, 4.1, 4.2, 5.1, 8.1, 11.1, 13.1, 14.1, 15.1, 16.2-16.5, 25.2-25.4, and 25.6-25.8; D-1 – D-14 claims 1.2-1.4, 1.6, 2.1, 19.1-19.3, 21.2, 21.3, 30.1, 30.2, 31.1, 32.1, 34.1, 35.1, 37.1, 38.1, 39.2, and 39.3; and Exhibits E-1 – E-14 claims 1.1-1.11, 2.1, 3.1, 5.1, 6.1, 6.2, 7.1, and 8.1. The following discussion shows that, at least under

Memory Integrity's apparent infringement theories, it was well known and conventional before the priority dates of the Asserted Patents to arrange processing components in clusters in a multiprocessor system.

As an initial matter, the Asserted Patents acknowledge that a cluster of processing components was well known. *See, e.g.*, '409 patent at 2:32-44 ("Background of the Invention … In one example, individual processors can be directly connected to each other through a plurality of point-to-point links to form a cluster of processors. Separate clusters of processors can also be connected. The point-to-point links significantly increase the bandwidth for coprocessing and multiprocessing functions. However, using a point-to-point architecture to connect multiple processors in a multiple cluster system sharing a single memory space presents its own problems."); '636 patent at 1:33-2:59; '121 patent at 1:20-2:38; '206 patent at 1:13-38; and '254 patent at 1:16-41.

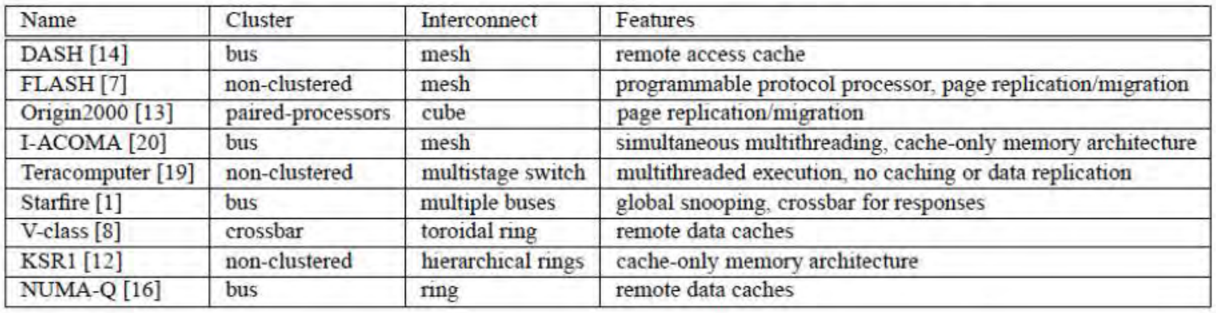
Indeed, under Memory Integrity's apparent infringement theories, there are many examples of prior art references that disclose a cluster of processing components and further demonstrate that such a structure was well known including:

- "Computer Organization," Hamacher (2001): *See, e.g.*, p. 636 ("We have considered several possible network topologies and showed that all existing topologies have certain advantages and disadvantages. Designers of multiprocessor systems strive to achieve superior performance at a reasonable cost. In an effort to exploit the most advantageous characteristics of different topologies, many successful machines feature mixed topologies. Bus and crossbar are excellent choices for connecting a few processors together. So, we often see a cluster of processors, typically from 2 to 8, connected using a bus or a crossbar. Such clusters, usually referred to as nodes, are then interconnected using a suitable topology to form a larger system.")
- "Design Options for Small Scale Shared Memory Multiprocessors," Barroso (1996): *See, e.g.*, p. 12-13 ("Cluster-based architectures are particularly effective when a significant fraction of the application parallelism can be captured by a single SMP node, or when the application can be mapped so that there is communication locality within a SMP node. Therefore, SMP nodes with a larger number of processors are preferred. Since bus based systems are likely to connect at most four processors in the

near future, alternative SMP interconnections such as rings or crossbars could be favored as the building blocks for larger scale systems as well.")

- "The NUMAchine Multiprocessor," Grindley (2000): *See, e.g.*, p. 1 ("A variety of large-scale multiprocessor architectures has been developed, as indicated in Table 1. The relevant features considered here are: type of clustering, type of interconnect, presence of caches for remote data, and the choice between non-uniform memory (NUMA) or cacheonly memory (COMA) architectures. Clustering processors together is a means of leveraging commodity symmetric multiprocessor (SMP) nodes. There are a number of possibilities for the system-wide interconnect including meshes, multistage switch networks, and rings. Each has advantages and disadvantages in terms of performance, complexity, and cost. Some systems include caches for remote data to mitigate longer memory access latencies as the system size increases. Finally, some systems employ a cache-only architecture (COMA) to automatically replicate and migrate data in hardware, rather than rely on caching with home memory locations as in NUMA systems. The systems listed in Table 1 use NUMA architecture, unless otherwise stated. The variety of architectures in Table 1 suggests that there is no single best approach when engineering such systems.")

Table 1. Some commercial and experimental multiprocessors



Name	Cluster	Interconnect	Features
DASH [14]	bus	mesh	remote access cache
FLASH [7]	non-clustered	mesh	programmable protocol processor, page replication/migration
Origin2000 [13]	paired-processors	cube	page replication/migration
I-ACOMA [20]	bus	mesh	simultaneous multithreading, cache-only memory architecture
Teracomputer [19]	non-clustered	multistage switch	multithreaded execution, no caching or data replication
Starfire [1]	bus	multiple buses	global snooping, crossbar for responses
V-class [8]	crossbar	toroidal ring	remote data caches
KSR1 [12]	non-clustered	hierarchical rings	cache-only memory architecture
NUMA-Q [16]	bus	ring	remote data caches

Grindley, Table 1

- "Designing Processor-cluster Based Systems: Interplay Between Cluster Organizations and Collective Communication Algorithms," Basak (1996): *See, e.g.*, p. 2-3 ("Computing nodes having more than one processors on a single multi-chip module or processor-board are becoming increasingly available … The number of processors in a cluster is usually small, ranging from 2-4. However, with advancements in VLSI, larger clusters are expected to become common. A cluster is connected to its inter-cluster network router through a cluster interface.")

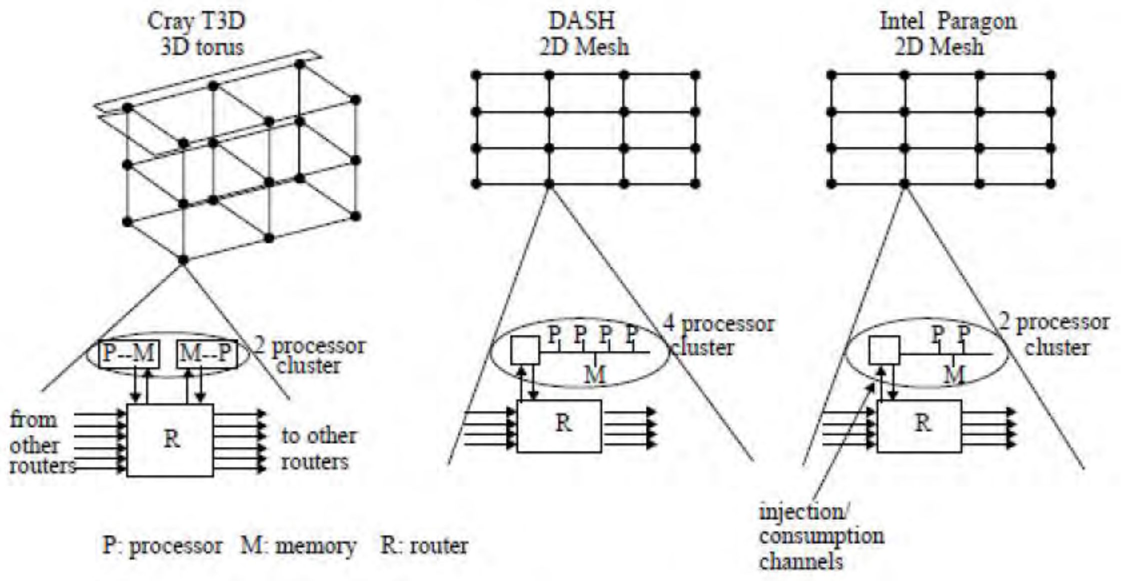
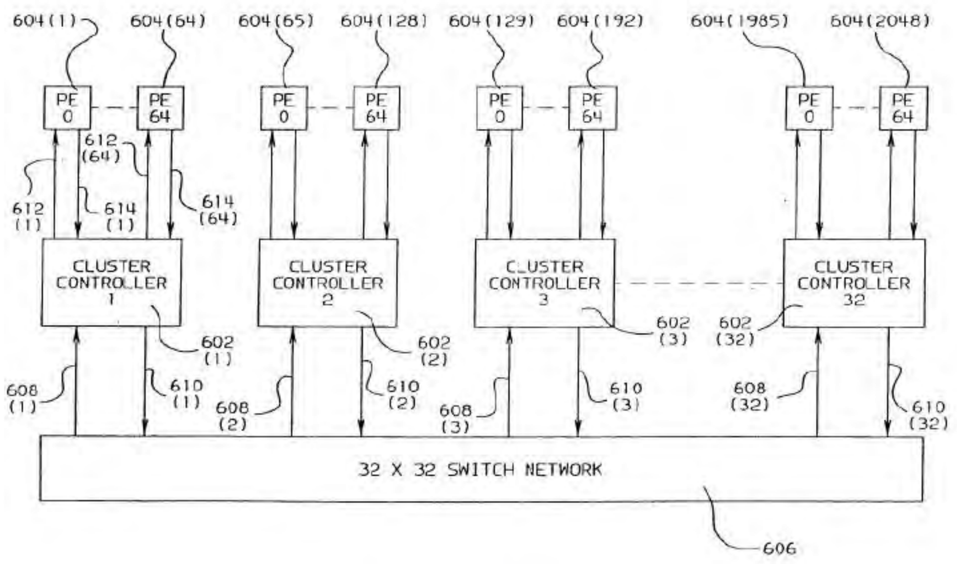


Figure 1: Recent examples of processor-cluster based systems.

Basak, Figure 1

- U.S. Patent No. 5,166,674 to Baum: *See, e.g.*, 5:1-27 ("The switch that interconnects the processing elements is hierarchical, comprising a network of clusters. Up to 64 processing elements are combined to form a cluster, and up to 64 clusters are linked by way of a Banyan network. Messages are routed through the switch in the form of packets, each of which comprise a quadword of data and a word of control information." "FIG. 6 is a structural overview of the present multiprocessing system. The system of FIG. 6 is a cluster connected network (cluster network) comprising 32 cluster controllers 602(1)-602(32). Each cluster controller provides a system interface for 64 Processing Elements (PEs) 604(1-64), 604(65-128) . . . 604(1985-2048). Each group of one cluster controller and 64 processing elements is referred to as a 'cluster'." "Each processing element in a given cluster is connected to the cluster controller by way of an input bus (e.g. 612(1)) and an independent (separate) output bus (e.g. 614(1)). Similarly, the 32 cluster controllers are each connected to a 32×32 switch network 606 by way of an input bus 608(1-32) and an independent output bus 610(1-32). The entire system thus includes 2048 processing elements 604(1-2048). Both the cluster controllers and the switch network operate to assemble and transfer data between the processing elements synchronously, under control of a high speed clock (e.g. 5 ns cycle time).")

FIG. 6



Baum, Figure 6

- U.S. Patent No. 5,752,264 to Blake: *See, e.g.*, 5:60-6:6 ("Referring to FIG. 2, there is shown a Multi-Processor computer system incorporating the present invention. The system includes: a system memory module 210; an input/output device (I/O device) 212; a plurality of clusters 214a-214n, and a common bus 216 that links the memory module, I/O device, and clusters together. Each cluster includes a level two cache, level two caches 224a-224n, and a plurality of microprocessors (CPU), CPUs 218aa-218an for cluster 214a, CPUs 218ba-218bm for cluster 224b, and CPUs 218na-218nm for cluster 214n. Each CPU has a level one cache, and is coupled to its respective level two cache through the level one cache via a point to point bus. Each cluster is coupled to the shared bus through its level two cache.")

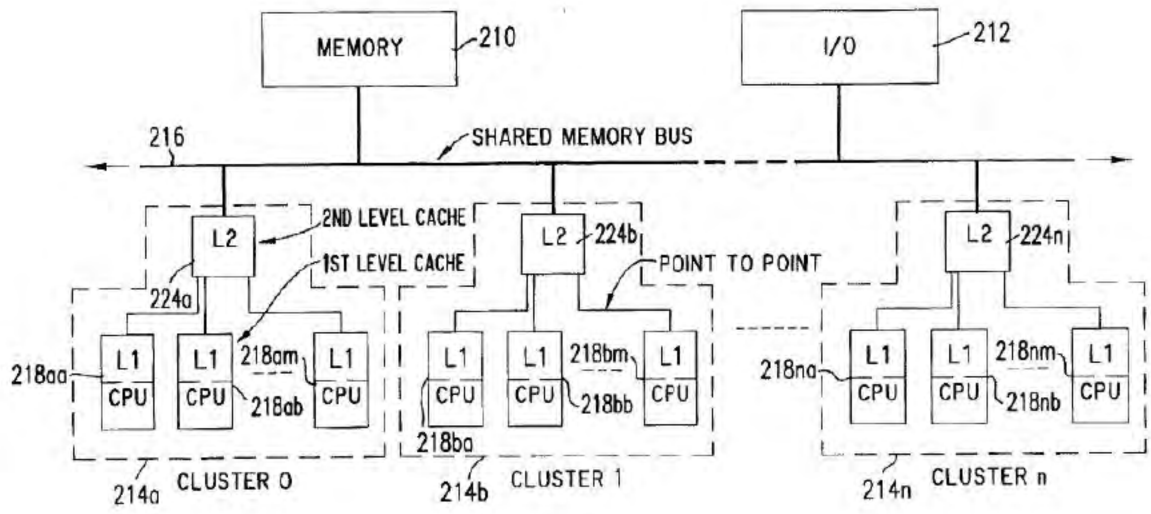


FIG. 2

Blake, Figure 2

- U.S. Patent No. 6,799,252 to Bauman: *See, e.g.,* 6:31-47 ("FIG. 1 shows one embodiment of multiprocessor computer system 100 of the present invention having one or more node clusters 170, each node cluster 170 having zero to N processors 74, zero to M memories 77, and zero to I input/output (I/O) subsystems 79. Depending on the needs of a user, interconnection network 175 can be set up as a three-dimensional torus, an N-dimensional hypercube, or any other suitable interconnection network between routers 76. In one embodiment, each router 76 includes eight ports 211, wherein each port 211 can be used to either connect to other routers 76, or to one to N node controllers 75 each having zero or more processor elements (PEs) 74. Thus, in some embodiments, a router 76 can be used as just an interconnection node in the network 175 (i.e., a circuit within block 175 rather than within node cluster 170), having no PEs 74 or memory 77 or I/O subsystems 79, and all of its ports are used to connect to other routers 76.")

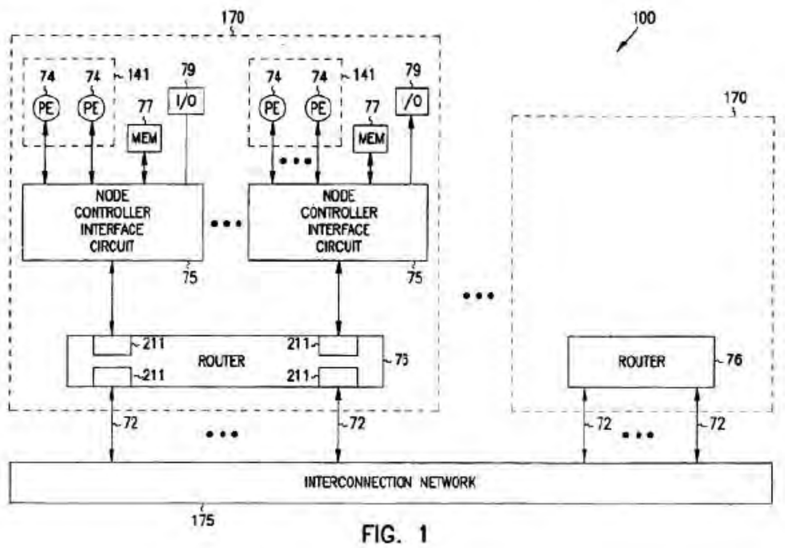


FIG. 1

Bauman, Figure 1

- U.S. Patent No. 6,751,721 to Webb: *See, e.g.*, 13:36-40 ("FIG. 3 represents a multi-processor system configured to use the hybrid invalidate scheme. Whereas the system shown in FIG. 1 comprises 12 processors 100, the system shown in FIG. 3 comprises 12 clumps or clusters 300. Within each cluster, there are four processors 310.")

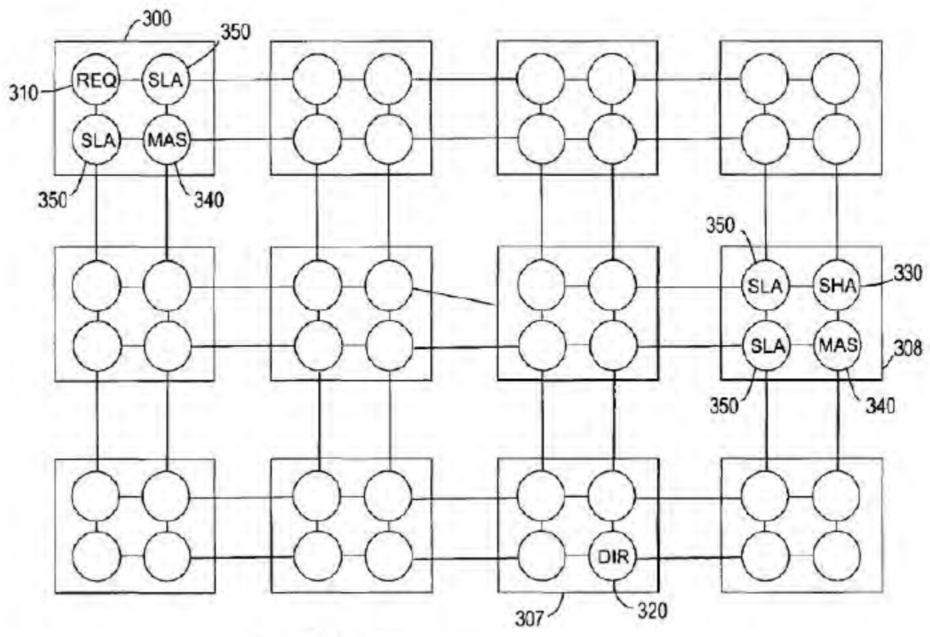


FIG. 3

Webb, Figure 3

Accordingly, it was well known before the priority dates of the Asserted Patents to arrange processing components in a cluster, at least under Memory Integrity's apparent infringement theories. A person of ordinary skill would further understand that arranging processing components in a cluster can provide numerous advantages as described below:

- "Scalable Parallel Computing," Hwang (1998): *See, e.g.*, 32-33 ("The cluster concept brings many benefits as well as challenges. Among them the most important ones are usability, availability, scalability, available utilization, and performance/cost ratio.")
- Hwang: *See, e.g.*, 33-34 ("**Usability** Since individual nodes of a cluster are a traditional platform, users can develop and run their applications in a familiar, mature environment. The platform provides all the powerful workstation programming environment tools and allows the thousands of existing (sequential) applications to run without change. Thus a cluster can be viewed as a huge workstation, providing much increased throughput and reduced response time for multiple sequential user jobs.")
- Hwang: *See, e.g.*, 34 ("**Availability** Instead of using custom components, clusters utilize inexpensive commodity components to provide higher availability, with multitudes of redundancy: *Processors and Memories ... Disk Arrays ... Operating System ...* .")
- Hwang: *See, e.g.*, 34 ("**Scalable Performance** A cluster's computing power can increase with added nodes. Again, clusters' scalability is a multitude scalability. This can be best seen by comparing clusters to SMPs. SMPs are processor-scalable systems, while clusters scale in many components, including processor, memory, disks, and even I/O devices. Being loosely coupled, clusters can scale to hundreds of nodes, while it is extremely difficult to build an SMP of more than tens of processors.")
- Hwang: *See, e.g.*, 35 ("Performance/Cost Ratio Clusters can achieve the above benefits cost-effectively.")

Thus, it would have been obvious to arrange processing components in a cluster in a multiprocessor system because doing so would simply be the use of a known technique to improve similar devices (*e.g.*, multiprocessors) in the same way (*e.g.*, increased usability, availability, scalability, available utilization, and performance/cost). One of ordinary skill would have been motivated to arrange processing components in a cluster to achieve the above-described benefits with a reasonable expectation of success. It would have also been obvious to arrange processing components in a cluster since such an implementation would be a simple substitution of one known element (*e.g.*, cluster of processing components) for another (*e.g.*, single processor) to obtain predictable results (*e.g.*, multiprocessor having increased usability, availability, scalability, available utilization, and performance/cost ratio). Furthermore, to the extent not disclosed, a person of ordinary skill in the art at the time of the alleged invention of the Asserted Claims would have been motivated to modify the prior art references identified in Section III and Exhibits A-1 – A-9; B-1 – B-19; C-1 – C-8; D-1 – D-14; and E-1 – E-14 to arrange processing components in a cluster.

5. *"Memory Controller"*

Some of the Asserted Claims are directed to a "memory access serialization point" that is a "memory controller." For example, claim 2.1 of the '409 patent recites "the memory access serialization point is a memory controller in the second cluster." *See also*, *e.g.*, '409 patent claims 11.1, 29.1, and 38.1; and '636 patent claims 16.1 and 28.1. At least under Memory Integrity's apparent infringement theories, implementing a "memory controller" as a "memory access serialization point" in a multiprocessor system was well-known in the art before the priority dates of the Asserted Patents. *See*, *e.g.*, Exhibits A-1 – A-9, claims 2.1, 11.1, 29.1, and 38.1; and Exhibits B-1 – B-19, claims 16.1 and 28.1. The following discussion shows that, at least under Memory Integrity's apparent infringement theories, it was well known and conventional before the priority dates of the Asserted Patents to implement a "memory controller" as a "memory access serialization point" in a multiprocessor system.

As an initial matter, during prosecution of all the patents in suit, Applicant cited U.S. Patent No. 6,167,492 to Keller, as prior art to the Patent Office. *See*, *e.g.*, '409 patent, Dec. 26, 2002, Information Disclosure Statement. Keller discloses a "multiprocessor computer system [that] includes a plurality of circuit nodes and a plurality of memories" where each "circuit node includes at least one microprocessor coupled to a memory controller which in turn is coupled to one of the plurality of memories." *See, e.g.*, Keller at Abstract. Keller further discloses that the memory controllers "include request queues for queuing memory access transactions." *See, e.g.*, Keller at 5:12-14.

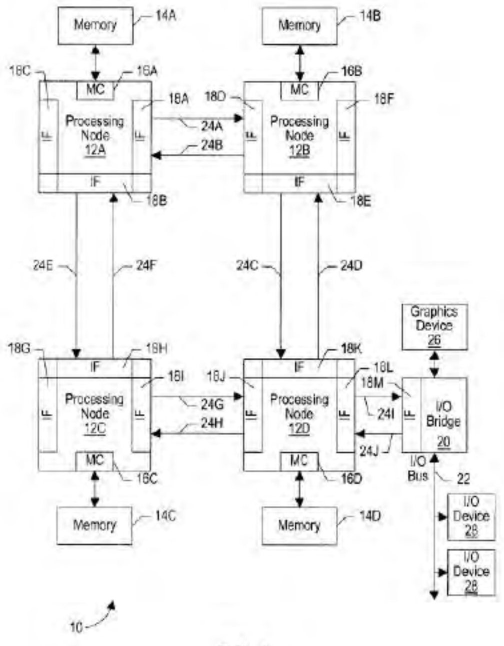


FIG. 1

Keller, Figure 1

At least under Memory Integrity's apparent infringement theories, there are many examples of prior art references that disclose implementing a "memory controller" as a "memory access serialization point." Examples of prior art references that disclose and further demonstrate that such was well known include:

- "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor." Lenoski (1992): *See, e.g.*, p. 150 ("A coherence protocol designed to address the above issues must be partitioned among the distributed components of the multiprocessor. These components include the processors and their caches, the directory and main memory controllers, and the interconnection network.")

- EP 0 681 240 to Hassoun: *See, e.g.*, para. [0006] ("In shared memory multiprocessor systems, the main processors generally are coupled directly to the shared memory interconnect. Each main memory generally is coupled to the shared memory interconnect through a separate main memory controller. If the system has more than one main memory, each memory contains a mutually exclusive set of data addresses. If a processor is to read data from a main memory or write data to a main memory, it must communicate with the main memory controller for that memory.")

- U.S. Patent No. 4,141,067 to McLagan: *See, e.g.*, Abstract ("A multiprocessor system is described in which a plurality of central processor units share the same main memory over a common asynchronous bus. Each central processor directs all memory requests to its own high speed cache memory. If the request is to read data from memory, the cache memory control determines if the addressed data is present in the cache memory. If so, the data is transferred to the processor without accessing main memory over the bus. If the data is not present in the cache memory, the cache memory control gains access to the bus by a priority circuit and reads out the data from memory, storing the data in the cache memory at the same time that it transfers the data to the processor. If the memory request by the processor is to write data in memory, the cache memory control gains access to the bus and initiates a data store in the main memory. At the same time, the cache memory control determines if the existing data being overwritten at the addressed location in main memory is present in the cache memory. If so, it updates the data in the cache memory at the same time it writes the data in main memory.")

- U.S. Patent No. 5,717,897 to McCrory: *See, e.g.*, Abstract ("The system maintains cache coherency by the exchange of commands between the main memory controller and the hosts cache controllers each of which define the state of the blocks of data stored in the host cache memories.")

- U.S. Patent No. 6,044,438 to Olnowich: *See, e.g.*, 8:38-57 ("The memory controller at each node contains intelligence to decide whether an accessed address is located in local memory or remote memory. This is accomplished by comparing memory sector definition bits of the memory address word to the Node ID register. If the compare is equal, the address is located in local memory. In this case, the memory controller accesses and returns the data locally without involving the network adapter. If the compare is not equal, the address is located in remote memory and the memory controller signals the processor that a remote read is required for thread z.")

- U.S. Patent No. 6,115,804 to Carpenter: *See, e.g.*, 3:40-43 ("Processing nodes 8a-8n may each include M (M≧0) processors 10, a local interconnect 16, and a system memory 18 that is accessed via a memory controller 17.")

- U.S. Patent No. 6,295,586 to Novak: *See, e.g.*, Abstract ("A memory controller for a computer memory which decodes memory requests into individual primitive memory operations which are then queued into separate operation queues."); *See, e.g.*, 5:23-29 ("The Northbridge chip includes a memory controller (MCT) 200 which controls and directs the flow of data between the memory requesters 210 and the memory 70 over the memory bus 100. The MCT 200 includes a memory request arbiter (MRA) 220 and an SDRAM memory controller (SMC) 230.")

- U.S. Patent No. 6,449,699 to Franke: *See, e.g.*, Abstract ("All memory access is intercepted and processed by the memory controller.")

- U.S. Patent No. 6,516,393 to Fee: *See, e.g.*, 6:19-36 ("By way of a general example four processors (a P0, P1, P2 and P3) attempting to access a common address, present their requests to the shared memory controller simultaneously. The four processors enter the pipeline in their named order: P0, P1, P2 and then P3. When P0 enters the pipe it will set its lock since it will not encounter any address contention. P1 will see P0, as was the case in the prior-art, set a resource-need for P0. P2 will enter the pipeline and instead of setting its resource-need for P0, it will instead, set it for P1. In the same manner, P3 will set its need for P2. When processor P0 completes, only P1 will make a request to priority: P2 and P3 will have a resource need for P1 and P2 respectively. Likewise, when P1 completes P2 will make a priority request and P3 will wait. If P0 comes back with a second request for the same address, before P3 has completed, P0 will set its resource need for P3. However, if P0 is requesting a different address, it can complete unimpeded by the existing ordered list.")

- U.S. Patent No. 6,546,429 to Baumgartner: *See, e.g.*, 3:20-23 ("Processing nodes 8a-8n may each include M (M≧0) processors 10, a local interconnect 16, and a system memory 18 that is accessed via a memory controller 17.")

- U.S. Patent No. 6,877,077 to McGee: *See, e.g.*, 1:34-44 ("FIG. 1 illustrates a well-known general computer system 100 having a central processing unit (CPU) 102 including CPU execution units 104, an internal (e.g., level 1 (L1)) cache memory 106, an external cache controller 108, and a primary memory controller 110. Typically, internal cache 106 is divided into an instruction cache, in which the most recently requested instructions are stored, and a data cache, in which the most recently requested data is stored. External cache controller 108 is coupled to and controls an external (e.g., level 2 (L2)) cache memory 109, and memory controller 110 is coupled to and controls primary memory 112. Although not shown for simplicity, memory controller 110 may include a write queue to store pending write requests for primary memory 112 and a read queue to store pending read requests for primary memory 112.")

- U.S. Patent No. 6,973,543 to Hughes: *See, e.g.*, 1:34-44 ("Many coherency protocols include the use of probes to communicate between various caches within the computer system. Generally speaking, a 'probe' is a message passed from the coherency point in the computer system to one or more caches in the computer system to determine if the caches have a copy of a block and optionally to indicate the state into which the cache should place the block. The coherency point may transmit the probes in response to a command from a component (e.g. a processor) to read or write the block. Each probe receiver responds to the probe, and once the probe responses are received the command may proceed to completion. The coherency point is the component responsible for maintaining coherency, e.g. a memory controller for the memory system.")

As illustrated by the prior art references above, it was well known before the priority dates of the Asserted Patents to implement a "memory controller" as a "memory access serialization point" in a multiprocessor system, at least under Memory Integrity's apparent infringement theories. Further, a person of ordinary skill would understand that implementing a memory controller as a "memory access serialization point" in a multiprocessor system resolves conflicts between transactions from multiple processors as described below:

- EP 0 681 240 to Hassoun: *See, e.g.*, para. [0006] ("In shared memory multiprocessor systems, the main processors generally are coupled directly to the shared memory interconnect. Each main memory generally is coupled to the shared memory interconnect through a separate main memory controller. If the system has more than one main memory, each memory contains a mutually exclusive set of data addresses. If a processor is to read data from a main memory or write data to a main memory, it must communicate with the main memory controller for that memory.")
- U.S. Patent No. 5,717,897 to McCrory: *See, e.g.*, Abstract ("The system maintains cache coherency by the exchange of commands between the main memory controller and the hosts cache controllers each of which define the state of the blocks of data stored in the host cache memories.")
- U.S. Patent No. 4,399,504 to Obermarck: *See, e.g.*, 1:19-22 ("In large data base systems where many work units or subtasks have a need to share access to the same records, there is a need to manage concurrent access to maintain integrity of the data.")
- U.S. Patent No. 5,404,482 to Stamm: *See, e.g.*, 1:65-2:6 ("Whenever processors communicate via a shared memory, it is desirable to require the processors to follow a protocol insuring that a memory address is not written to simultaneously by more than one processor, or else the result of one processor will be nullified by the result of another processor. Such synchronization of memory access is commonly achieved by requiring a processor to obtain an exclusive privilege to write to an addressed portion of the shared memory, before executing a write operation.")

Accordingly, it would have been obvious to implement a memory controller as a "memory access serialization point" in a multiprocessor system because doing so would simply be the use of a known technique (*e.g.*, a memory controller as a "memory access serialization point") to improve similar devices (*e.g.*, multiprocessors) in the same way (*e.g.*, resolve conflicts and provide proper access to memory). One of ordinary skill would have also been motivated to implement a memory controller as a "memory access serialization point" to resolve conflicts and provide proper access to memory with a reasonable expectation of success. It would have been obvious to implement a memory controller as a "memory access serialization point" since such an implementation would be a simple substitution of one known element (*e.g.*, a memory controller as a "memory access serialization point") for another (*e.g.*, bus arbitration[2]) to obtain predictable results (*e.g.*, multiprocessor having conflict resolution and providing proper access to memory). Furthermore, to the extent not disclosed, a person of ordinary skill in the art at the time of the alleged invention of the Asserted Claims would have been motivated to modify the prior art references identified in Section III and Exhibits A-1 – A-9; B-1 – B-19; C-1 – C-8; D-1 – D-14; and E-1 – E-14 to include a memory controller as a "memory access serialization point."

6. *"Locking" a "Memory Line"*

Some of the Asserted Claims are directed to a "memory line" that is "locked." For example, claim 6.8 of the '409 patent recites "a memory line associated with the cache access request is locked." *See also*, *e.g.*, '409 patent claims 8.1, 26.1, and 52.4; and '636 patent claims 12.1, 21.8, and 23.1. At least under Memory Integrity's apparent infringement theories, locking a memory line in a multiprocessor system was well-known in the art before the priority dates of the Asserted Patents. *See*, *e.g.*, Exhibits A-1 – A-9, claims 6.8, 8.1, 26.1, and 52.4; and Exhibits B-1 – B-19, claims 12.1, 21.8, and 23.1. The following discussion further shows that, at least under Memory Integrity's apparent infringement theories, it was well known and conventional before the priority dates of the Asserted Patents to lock a memory line in a multiprocessor system.

At least under Memory Integrity's apparent infringement theories, there are many examples of prior art references that disclose locking a memory line. Examples of prior art references that disclose and further demonstrate that such was well known include:

- "The NUMAchine Multiprocessor," Grindley (2000): *See*, *e.g.*, p. 3 ("A line in any one of the four states can also be locked. Locking of the line occurs at the beginning of a coherence action that requires multiple stages, and ensures that no other access to the line is possible until the transaction completes."); p. 4 ("NUMAchine provides a retry

[2] "Parallel Computer Architecture," Culler et al. (1998): *See*, *e.g.*, p. 385.

mechanism for requests that are negatively-acknowledged when they encounter a locked state in a directory. This approach avoids the need to buffer an arbitrary number of such requests at the locked site. Instead, the originator of the request employs a modified binary exponential back-off retry algorithm.")

- "The Sun Fireplane System Interconnect," Charlesworth (2001): *See, e.g.*, p. 7 ("Lock line and check coherency (cycle 9–11). The home SSM agent locks the line, so that no other transaction can be made to this cache line. It checks its coherency directory cache. In this case, we assume a hit that indicates that the requested location is not owned. If the coherency directory cache had missed, then the home SSM agent would have had to wait an extra 16 system cycles (106 ns) for the Mtags to arrive from memory.")

- "Using Hints to Reduce the Read Miss Penalty for Flat COMA Protocols," Björkman (1995): *See, e.g.*, 2.2 ("Each directory entry can be in two stable states, EXCLUSIVE and SHARED, that indicate that there is exactly one or more than one memory copy in the system, respectively. Moreover, the directory state can be also in a transient state, WAIT_INVALIDATE, indicating that an ownership transaction is in progress." "Ownership transactions are handled according to Figure 4. When Home receives an ownership request (GWr), it forwards it to the Master (WForward) and the state of the directory entry becomes WAIT_INVALIDATE. From now on, all incoming read miss as well as ownership requests will be rejected and have to be retried.")

- EP 0 489 583 to Tipon: *See, e.g.*, 4:1-23 ("For lock memory operations (where a block of memory is locked for use by a single processor), the semaphores (control flags) needed for the operations are stored in the home directories rather than in the processor caches. This provides fast access, but does not create a high level of traffic, since the semaphores are a low percentage of all operations occurring throughout the computer system 100." "Read operations initiated by processors P from different nodes cause other directories D to have copies of the semaphore. To initiate a lock operation, a requesting processor P issues a read-with-intent-to-modify (RIM) message that passes through its cache to the home directory of the cache line, which RIM causes the home directory to broadcast an invalid message to all directories D." "Once a lock operation has begun, the home directory does not allow access to the semaphore until the write operation associated with the lock operation is complete. The home directory issues retry messages to any request to access the semaphore occurring between the read and completion of the write of the lock operation.")

- U.S. Patent No. 4,399,504 to Obermarck: *See, e.g.*, Abstract ("Data resources are shared by applications executing on a plurality of central electronic complexes. Each complex of a pair includes a resource lock manager (IRLM) which maintains the hold and wait locks for applications executing on the complex and selected wait locks for the other complex. Selective communication of lock request information is controlled by hash tables maintained in synchronization in each IRLM, which denote the interest of each complex in each hash class of data resources.")

- U.S. Patent No. 4,775,955 to Liu: *See, e.g.*, Title ("Cache coherence mechanism based on locking"); *See, e.g.*, 4:19-33 ("The basic idea of the locking-based cache coherence control is as follows. Software accesses to shared data are often controlled through certain authorization mechanisms. Locking is a typical technique for such software synchronization control. A user needs to be granted a lock in order to access an object. After using the object the user releases the lock. There are different types of locking. Here the concern is primarily with locking such that, when a user modifies an object, the locking of the object should guarantee exclusivity (that is, this user is the only one who can access this object before the object is released by the user). As a result, a user is granted an EXCLUSIVE lock for an object only when all other users have released their locks on this object.")

- U.S. Patent No. 4,965,719 to Shoens: *See, e.g.*, Title ("Method for lock management, page coherency, and asynchronous writing of changed pages to shared external store in a distributed computing system"); *See, e.g.*, Abstract ("A method for increasing throughput of N-way central electronic complexes concurrently executing processes to selectively lockable data resources while maintaining coherency among replicates of the information state of any accessed resource.")

- U.S. Patent No. 5,404,482 to Stamm: *See, e.g.*, Abstract ("A processor and method for preventing access to a locked memory block in a multiprocessor computer system. The processor has a cache memory and records a memory lock in a content-addressable memory separate from the cache memory. Preferably, outstanding cache fills are recorded in the same content addressable memory as memory locks, and a memory lock or an outstanding cache fill delays the execution of a cache coherency request upon the same memory block. When a cache coherency request is received from another processor, the address of the cache coherency request is compared to addresses stored in the content addressable memory, and when there is a match, a bit in the matching entry is set to indicate a delayed request that is executed after the lock is unlocked or the cache is refilled. In a specific embodiment, a memory lock or an outstanding cache fill also stalls a processor read or write to the same memory block.")

- U.S. Patent No. 5,594,886 to Smith: *See, e.g.*, 13:7-13 ("In order to increase the versatility of the system 10 it is preferred that the cache control logic 32 can be programmed to 'lock' a memory location so that a cache line of data stored in the locked memory location cannot be written back to main memory RAM1 or RAM2 until the cache control logic 32 changes the status of the memory location from locked to unlocked.")

- U.S. Patent No. 6,044,438 to Olnowich: *See, e.g.*, 11:26-36 ("Likewise, global locking mechanisms for use when two nodes are competing to read-modify-write the same shared memory location are well known in the art. Global locking approaches are described in U.S. Pat. No. 4,399,504, 'Methods and Means for Sharing Data Resources in a Multiprocessing, Multiprogramming Environment' by Watts et al, and U.S. Pat. No. 4,965,719, 'Method for Lock Management, Page Coherency, and Asynchronous Writing of Changed Pages to External Store in a Distributed Computing System' by Shoens et al.")

- U.S. Patent No. 6,101,420 to VanDoren: *See, e.g.*, 48:59-62 ("Other processing systems allow one reference to any given cache line to be in transit at any instance in time. Subsequent references to a cache line in transit are blocked until the reference in transit is completed.")

- U.S. Patent No. 6,182,195 to Laudon: *See, e.g.*, 5:25-45 ("FIG. 2 shows a portion of the main memory space 202, separated into pages 204. Each main memory portion 114 stores N number of pages that are divided into memory blocks. According to the preferred embodiment of the present invention, a memory block is the size of a cache line for each block, which stores the caching state of the block, and pointers to the processors who are caching this block. A memory directory 206 has a corresponding multifield entry 208 for each block. A first bit field 210 points to the node (i.e., location) caching the corresponding block and the state (i.e., poison or not poison) of that block in a second bit field 212. If the poison bit is set (e.g., has a value of logical '1'), the corresponding virtual-to-physical memory translation is stale, and when it is not set, the translation is presumed valid. Further block state information to indicate whether the memory block is locked for atomic operation, so that the corresponding block can not be accessed. The lock on updating a page table entry in memory, for example, is a logical construct which is carried out either by a software algorithm or a hardware lock operation on a normal memory location.")

- U.S. Patent No. 6,356,983 to Parks: *See, e.g.*, Abstract ("A cache coherency directory for a shared memory multiprocessor computer system. A data structure is associated with each cacheable memory location, the data structure comprising locations for storing state values indicating an exclusive state, a shared state, an uncached state, a busy state, a busy uncached state, a locked state, and a pending state. The busy state and pending state cooperate to reserve a cache line for future use by a processor while the cache line is currently being used by one or more other processors.")

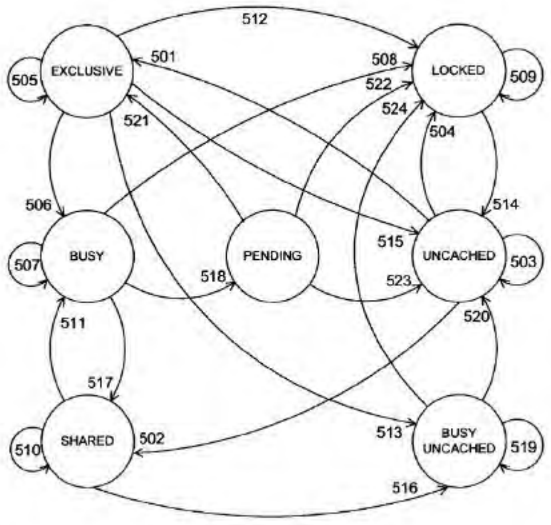


FIG. 5

Parks, Figure 5

- U.S. Patent No. 6,625,698 to Vartti: *See, e.g.*, Abstract ("A system and method for controlling storage locks based on cache line ownership. Ownership of target data segments is acquired at a memory targeted by a first requesting device. A storage lock is enabled that prohibits requesting devices, other than the first requesting device, from acting on the target data segments during the time the targeted memory possesses ownership of the target data segments. A storage lock release signal is issued from the first requesting device to the targeted memory when exclusivity of the target data segments is no longer required at the first requesting device. In response, the storage lock at the targeted memory is released, thereby allowing other requesting devices to act on the target data segments."); *See, e.g.*, 11:2-8 ("In another embodiment, the storage controller includes a content addressable memory (CAM) to store the addresses corresponding to each of the targeted cache lines currently subject to the storage lock. The addresses in this CAM are the compared to addresses of subsequently requested cache lines to determine whether or not the newly requested cache line(s) is subject to the storage lock.")

- U.S. Patent No. 6,640,287 to Gharachorloo: *See, e.g.*, 1:55-2:7 ("Many cache coherence protocol operations require an update of a corresponding directory entry, because they affect the cached copies of a memory line of information that is the subject of the operation. Furthermore, since a particular cache coherence protocol operation may affect many nodes, the operation must be executed in distributed fashion while still updating the directory entry correctly." "One way of achieving this effect is to 'lock' a directory entry while an operation is in progress. The directory entry is unlocked after an acknowledgment message has been received from each node affected by the operation. For example, when a number of cached copies of a memory line of information are to be invalidated, the home node locks the directory entry, sends an invalidation message to each node caching a copy of the memory line of information, and updates and unlocks the directory entry only when acknowledgment messages have been received from each of these nodes. While the directory entry is locked, the home node blocks any other operations on the corresponding memory line of information.")

- U.S. Patent No. 6,711,662 to Peir: *See, e.g.*, claim 18 ("[T]he home directory to a data block is a serialization point for that data block.")

- U.S. Patent No. 6,986,005 to Vo: *See, e.g.*, 3:24-28 ("The memory controller 21 has lock registers 60 a and 63 a. The lock register 60 a has a lock flag 61 of length 1 bit and a field 62 for storing a request 70 (explained in detail in the next paragraph)."); *See, e.g.*, 5:7-14 ("After granting exclusive access to the memory location 26 to the processor 20 a, the memory controller 21 denies all other requests for access to the memory location 26. However, requests for access to other memory locations by other processors are allowed. Thus, processor 20 a can be given exclusive access to a memory location while processors on bus 29 in node 0 and in nodes 1 and 2 can access other memory locations.")

As illustrated by the prior art references above, it was well known before the priority dates of the Asserted Patents to lock a memory line, at least under Memory Integrity's apparent infringement theories. Further, a person of ordinary skill would understand that locking a memory line would provide the benefit of resolving conflicts between transactions in a multiprocessor system as described below:

- Obermarck: *See, e.g.*, 1:19-22 ("In large data base systems where many work units or subtasks have a need to share access to the same records, there is a need to manage concurrent access to maintain integrity of the data.")
- Stamm: *See, e.g.*, 1:65-2:6 ("Whenever processors communicate via a shared memory, it is desirable to require the processors to follow a protocol insuring that a memory address is not written to simultaneously by more than one processor, or else the result of one processor will be nullified by the result of another processor. Such synchronization of memory access is commonly achieved by requiring a processor to obtain an exclusive privilege to write to an addressed portion of the shared memory, before executing a write operation.")
- Smith: *See, e.g.*, 13:7-13 ("In order to increase the versatility of the system 10 it is preferred that the cache control logic 32 can be programmed to 'lock' a memory location so that a cache line of data stored in the locked memory location cannot be written back to main memory RAM1 or RAM2 until the cache control logic 32 changes the status of the memory location from locked to unlocked.")
- "Computer Organization," Hamacher (2001): *See, e.g.*, p. 645 ("12.6.1 ACCESSING SHARED VARIABLES Assume that we have identified two tasks that can run in parallel on a multiprocessor. The tasks are largely independent, but from time to time they access and modify some common, shared variable in the global memory. For example, let a shared variable SUM represent the balance in an account. Moreover, assume that several tasks running on different processors need to update this account. Each task manipulates SUM in the following way: The task reads the current value from SUM, performs an operation that depends on this value, and writes the result back into SUM. It is easy to see how errors can occur if such read-modify-write accesses to SUM are performed by tasks T1 and T2 running in parallel on processors P1 and P2. Suppose that both T1 and T2 read the current value from SUM, say 17, and then proceed to modify it locally. T1 adds 5 for a result of 22, and T2 subtracts 7 for a result of 10. They then proceed to write their individual results back into SUM, with T2 writing first followed by T1. The variable SUM now has the value 22, which is wrong. SUM should contain the value 15 (= 17 ÷ 5 - 7), which is the intended result after applying the modifications strictly one after the other, in either order." "To guarantee correct manipulation of the shared variable SUM, each task must have exclusive access to it during the complete read-modify-write sequence. This can be provided by using a global lock variable, LOCK, and a machine instruction called Test-and-Set. The

variable LOCK has two possible values, 0 or 1. It serves as a guard to ensure that only one task at a time is allowed access to SUM during the time needed to execute the instructions that update the value of this shared variable.")

- Hamacher: *See, e.g.*, p. 645 ("12.6.3 NEED FOR LOCKING AND CACHE COHERENCE We should note that the requirement for lock guard controls on access to shared variables is independent of the need for cache coherence controls -- both types of controls are needed. Consider a situation in which cache coherence is maintained by using the writethrough policy accompanied by cache updating of writes to shared variables. Suppose that the contents of SUM in the example in Section 12.6.1 have been read into the caches of the two processors that execute tasks T1 and T2. If the read operations are part of an update sequence and are not made mutually exclusive by the use of a lock guard control, then the original error can still occur. If task T1 writes its new value last, as before, then SUM will contain the value 22, which is wrong. Cache coherence is maintained throughout this sequence of events. However, incorrect results are obtained because lock guard controls are not used.")

Accordingly, it would have been obvious to lock a memory line because doing so would simply be the use of a known technique to improve similar devices (*e.g.*, multiprocessors) in the same way (*e.g.*, resolve conflicts). One of ordinary skill would have indeed been motivated to lock a memory line to resolve conflicts with a reasonable expectation of success. Furthermore, to the extent not disclosed, a person of ordinary skill in the art at the time of the alleged invention of the Asserted Claims would have been motivated to modify the prior art references identified in Section III and Exhibits A-1 – A-9; B-1 – B-19; C-1 – C-8; D-1 – D-14; and E-1 – E-14 to lock a memory line.

7. *"Cache Coherence Controller" and "Interconnection Controller"*

Some of the Asserted Claims are directed to cache coherence or interconnection controllers. For example, claim 1.2 of the '409 patent recites "a first cache coherence controller" and claim 1.3 of the '206 patent recites "an interconnection controller." *See also, e.g.*, '409 patent claims 1.3-1.7, 6.2-6.8, 7.1, 7.2, 8.1, 9.1, 10.1, 11.1, 12.1, 18.1, 19.1, 20.1, 22.1, 23.1, 25.1, 25.4, 25.5, 34.4, 34.5, 51.1-51.4, and 52.1-52.4; '636 patent claims 11.1-11.5, 12.1, 15.2-15.7, 18.1, 21.2-21.8, 22.1-22.5, 23.1, 24.1, 25.1, 26.1, 27.1, 28.1, 29.1, 30.1, 31.1, 33.1, 34.1, 35.1, and

36.1-36.5; '121 patent claims 3.1, 3.2, 5.1, and 6.1; '206 patent claims 1.4, 1.5, 1.6, 2.1, 15.1, 19.2, 21.1-21.6, 22.1, 24.1, 25.1, 27.1, 29.1, 30.2, 30.3, 31.1, 35.1, 38.2, 39.3, 39.4, 39.6, and 39.8; and '254 patent claims 1.3, 1.4, 2.1, 7.1, and 8.1. At least under Memory Integrity's apparent infringement theory, cache coherence/interconnection controllers were well-known in the art before the priority dates of the Asserted Patents. *See, e.g.*, Exhibits A-1 – A-9, claims 1.2-1.7, 6.2-6.8, 7.1, 7.2, 8.1, 9.1, 10.1, 11.1, 12.1, 18.1, 19.1, 20.1, 22.1, 23.1, 25.1, 25.4, 25.5, 34.4, 34.5, 51.1-51.4, and 52.1-52.4; Exhibits B-1 – B-19, claims 11.1-11.5, 12.1, 15.2-15.7, 18.1, 21.2-21.8, 22.1-22.5, 23.1, 24.1, 25.1, 26.1, 27.1, 28.1, 29.1, 30.1, 31.1, 33.1, 34.1, 35.1, and 36.1-36.5; C-1 – C-8 claim 3.1, 5.1, and 6.1; D-1 – D-14 claims 1.3, 1.4, 1.5, 1.6, 2.1, 15.1, 19.2, 21.1-21.6, 22.1, 24.1, 51.1, 27.1, 29.1, 30.2, 30.3, 31.1, 35.1, 38.2, 39.3, 39.4, and 39.6; and Exhibits E-1 – E-14 claims 1.3, 1.4, 2.1, 7.1, and 8.1. The following discussion further shows that, at least under Memory Integrity's apparent infringement theory, it was well known and conventional to implement cache coherence/interconnection controllers in multiprocessor systems.

At least under Memory Integrity's apparent infringement theories, there are many examples of prior art references that disclose implementing cache coherence/interconnection controllers in multiprocessor systems. Examples of prior art references that disclose and further demonstrate that such was well known include:

- "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor." Lenoski (1992): *See, e.g.*, p. 150 ("A DASH system consists of a number of modified 4D/240 systems that have been supplemented with a directory controller board. This directory controller board is responsible for maintaining the cache coherence across the nodes and serving as the interface to the interconnection network.")

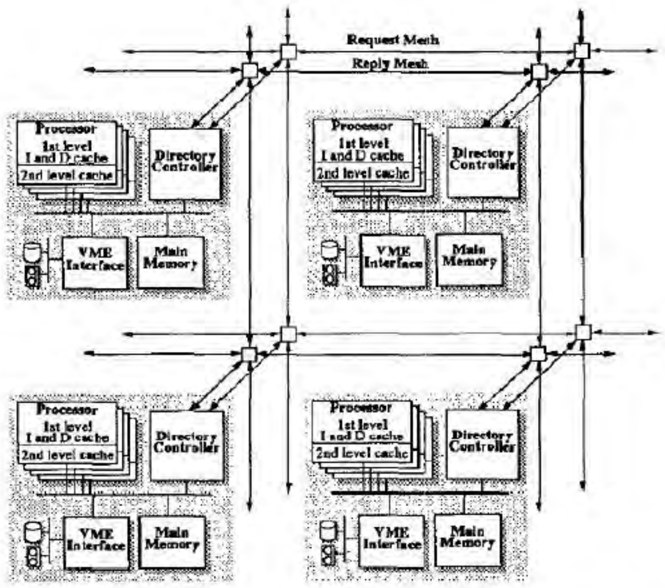
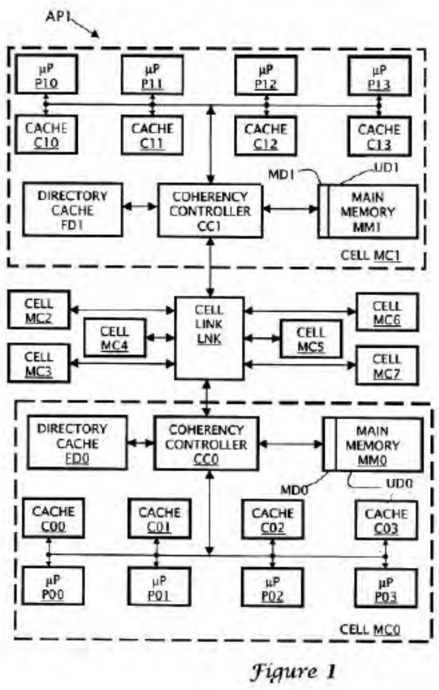


Figure 2: Block diagram of sample 2 x 2 DASH system.

Lenoski, Figure 2

- U.S. Patent No. 6,055,610 to Smith: *See, e.g.*, 6:44-55 ("In accordance with the present invention, a distributed-memory multi-processor system AP1 with directory-based cache coherency comprises eight memory cells MC0-MC7 communicatively coupled via a cell communications link LNK, as shown in FIG. 1. Memory cell MC0 includes four processors P00-P03, four user-data caches C00-C03, main memory MM0, a fast coherency directory FD0, and a coherency controller CC0. Likewise, memory cell MC1 includes four processors P10-P13, four caches C10-C13, main memory MM1, a fast directory FD1, and a coherency controller CC1. Memory cells MC2-MC7 are essentially the same as memory cells MC0 and MC1.")



Smith, Figure 1

- U.S. Patent No, 6,085,295 to Ekanadham: *See, e.g.*, 3:37-45 ("The preferred embodiment of our system that is based on a network of switch-based SMP nodes with an adapter attached to each node. FIG. 1 illustrates a high-level diagram of such a multiprocessing system. Each node has a plurality of processors P1, P2, . . . , PN interconnected to each other by a switch (SW). The switch also interconnects the memory modules M1, M2, . . . , MN and adapters A. The nodes in turn, are connected to each other through a network as shown.")

- Ekanadham: *See, e.g.*, 3:49-56 ("The adapter connects to the switch and plays the role of either a memory or a processor. The behavior of the adapter is different for different memory lines. When a line is homed at the local memory of the node, the adapter behaves as a proxy processor for that line. When a line is homed at the memory of a remote node, the adapter behaves as a proxy memory for that line. These roles are illustrated in FIGS. 3A-3C and are elaborated further below.")

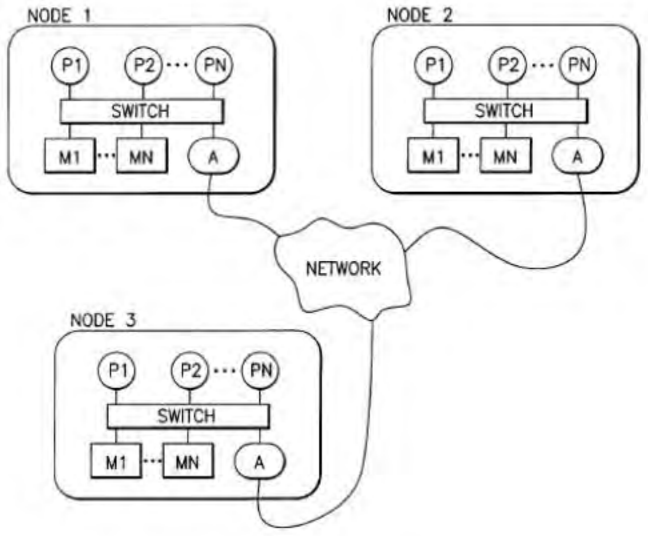


FIG.1
PRIOR ART

Ekanadham, Figure 1

- U.S. Patent No, 6,141,692 to Loewenstein: *See*, *e.g.*, 9:12-27 ("Referring now to FIG. 4, the architecture of the parallel computer system 40 incorporating the invention is characterized by multiple subsystems (also known as nodes) 410, 420, 430 and 440. The various nodes 410, 420, 430, and 440 are interconnected via a global interconnect 450. Although a system having only four nodes is depicted, the invention is applicable to systems having any number of interconnected nodes. Each node is assigned a unique network node address. Each node includes at least one processor, a corresponding number of memory management units (MMUs) and caches, a main memory assigned a portion of a global memory address space, a global interface (GI) and a local-node interconnect (LI). For example, node 410 includes processors 411a, 411b . . . 411i, MMUs 412a, 412b, . . . 412i, cache memories 413a, 413b, . . . 413i, main memory 414, global interface 415, and local-node interconnect 419.")
- Loewenstein: *See*, *e.g.*, 9:39-47 ("Referring now to the block diagram of FIG. 5, each global interface (i.e., items 415, 425, 435, and 445 of FIG. 4) includes a home agent (HA) 502, a slave agent (SA) 504 , and a request agent (RA) 506. The HA 502 is responsible for maintaining its associated directory 503 (directory 503 corresponds to either item 416, 426, 436 or 446 of FIG. 4) by updating the status of data from each main memory address that is copied to a cache line in its own node (the home node) or in any other node.")

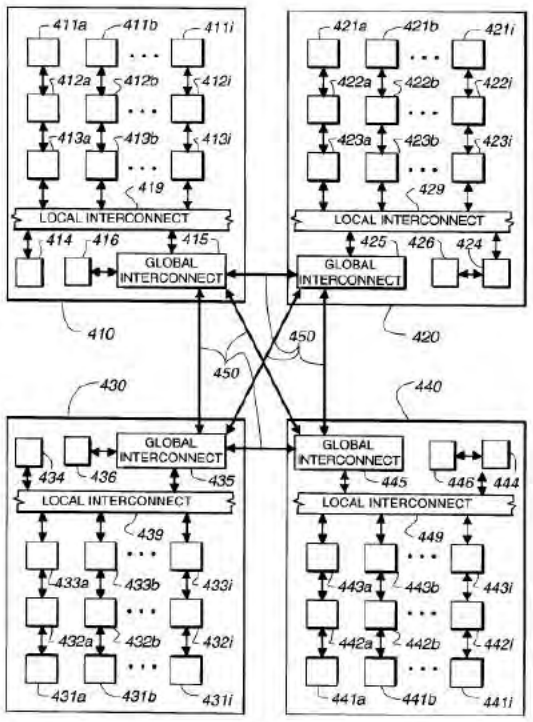
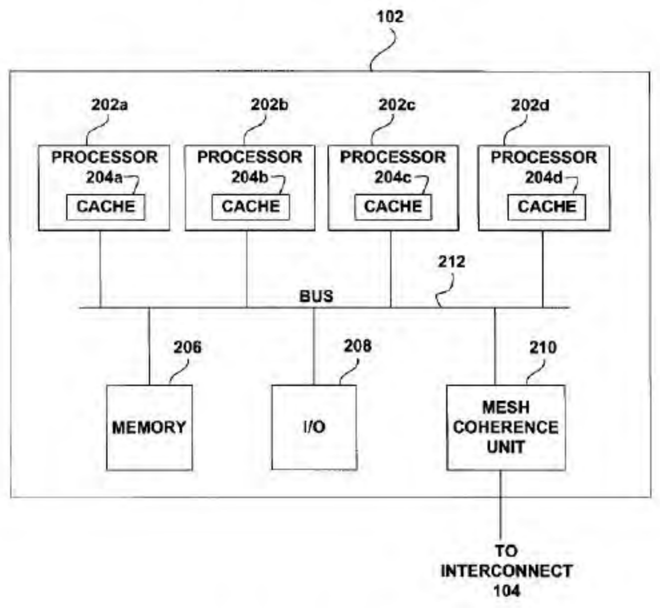


Fig. 4

Loewenstein, Figure 4

- U.S. Patent No. 6,209,064 to Weber: *See, e.g.*, 3:58-4:2 ("FIG. 2 is a functional block diagram of a processor node 102. Processor node 102 is exemplary of FIG. 1 processor nodes 102 a-t and includes processors 202 a-d each having a respective cache 204 a-d, a memory subsystem 206, an input/output subsystem 208, and a mesh coherence unit (MCU) 210. Each of the functional units 202 a-d, 206, 208, and 210 are connected to bus 212 for transmitting control, address, and data signals between the units. The mesh coherence unit 210 is connected to interconnection 104. The mesh coherence unit 210 coordinates inter-processor node cache coherence, inter-processor node message passing, and inter-processor node memory protection.")



Weber, Figure 2

- U.S. Patent No. 6,209,065 to Van Doren: *See, e.g.*, 11:34-46 ("FIG. 6 is a schematic block diagram of an augmented SMP node 600 comprising a plurality of processors (P) 102—108 interconnected with a shared memory 150, an IOP 130 and a global port interface 610 via a local switch 625. The processor, shared memory and IOP entities are similar to the those entities of FIG. 1. The local switch 625 is augmented (With respect to switch 200) to include an additional port coupling the interface 610 by way of a full-duplex, clock forwarded global port (GP) data link 612. In addition to the DTAG 160, an additional shared data structure, or directory (DIR) 650, is coupled to Arb bus 170 to administer the distributed shared memory environment of the large system 400.")

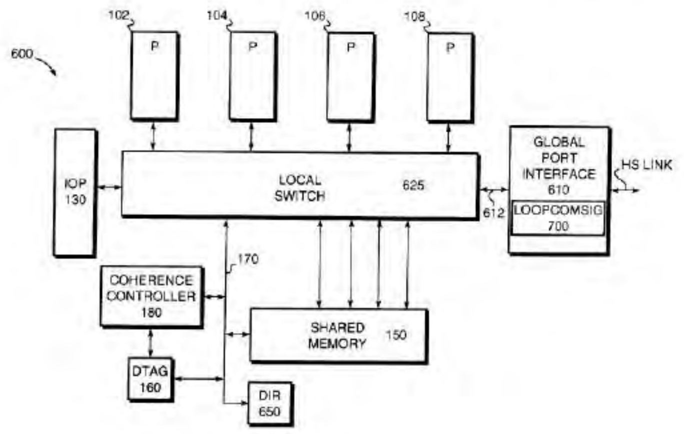
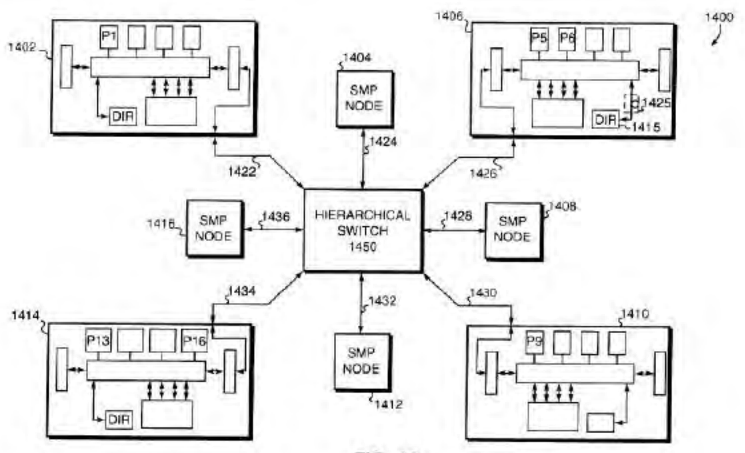


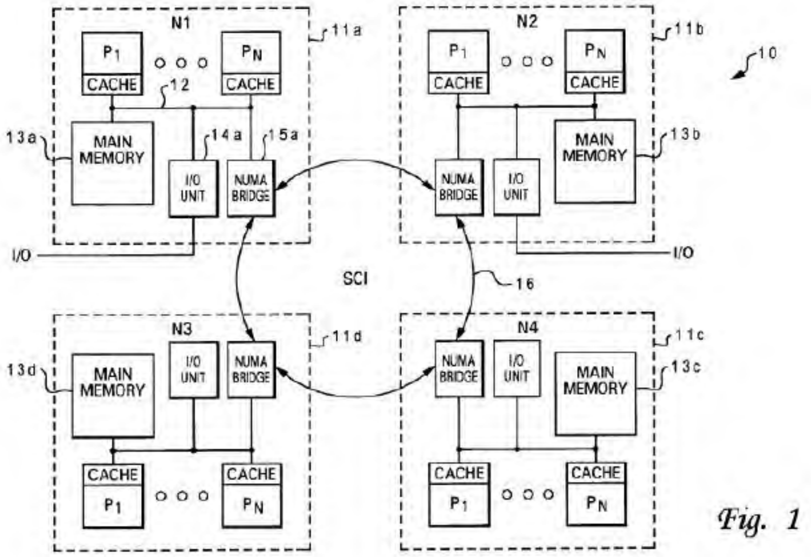
FIG. 6

Van Doren, Figure 6



Van Doren, Figure 14

- U.S. Patent No. 6,334,177 to Baumgartner: *See, e.g.*, 3:13-19 ("All nodes 11 a-11 d are interconnected by a Scalable Coherent Interconnect (SCI) 16. SCI 16 is a high-bandwidth interconnection network capable of providing cache coherence throughout NUMA multiprocessor system 10. Each of nodes 11 a-11 d has a NUMA bridge, such as a NUMA bridge 15 a in node 11 a, to provide connections to SCI 16 in order to maintain inter-nodal connection among nodes 11 a-11 d.")



Baumgartner, Figure 1

- U.S. Patent No. 6,560,681 to Wilson: *See, e.g.*, 5:27-37 ("FIG. 4 shows an overview of a ccNUMA system 400 comprising a number of connected nodes 402, . . . , 408, and 410. Each node, as shown for node 410, includes some number of processors 412, 414, . . . , and 420 having respective caches 422, 424, . . . , and 430 connected to a single memory subsystem 440. Memory subsystem 440 includes an external directory 460, main memory 490, and a coherence control chip 480 which contains a coherence controller 450 and a temporary state buffer 470. Coherence controller 450 reads and writes state information contained in an external directory 460 and temporary state buffer 470.")

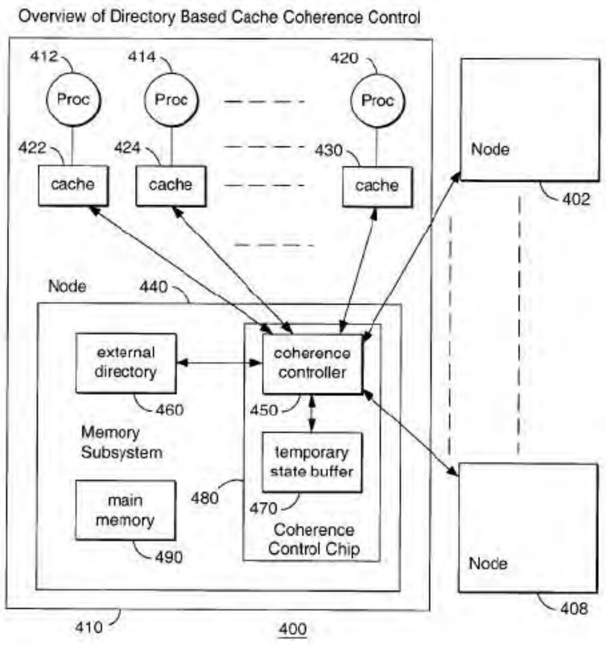
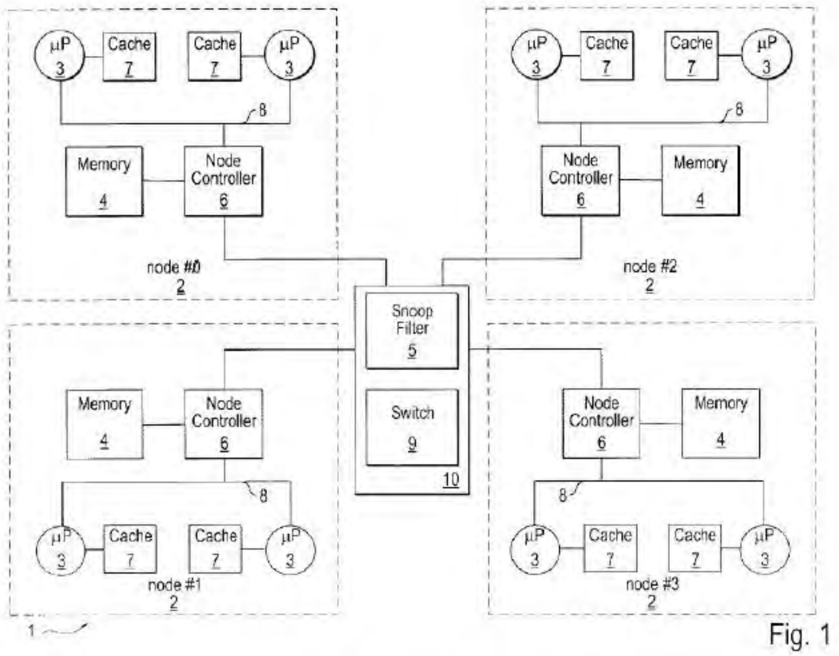


FIG. 4

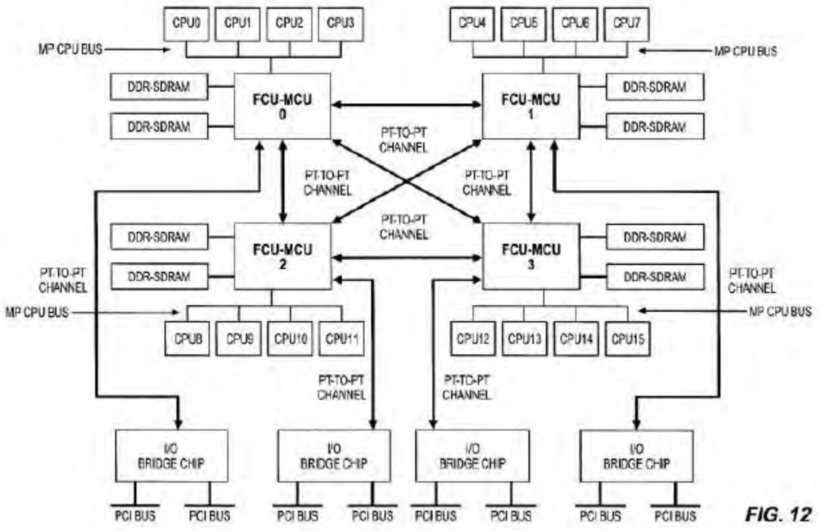
Wilson, Figure 4

- U.S. Patent No. 6,598,123 to Anderson: *See, e.g.*, 1:18-21 ("Each node contains one or more processors, a node controller, and memory, including one or more levels of cache memory associated with each processor.")



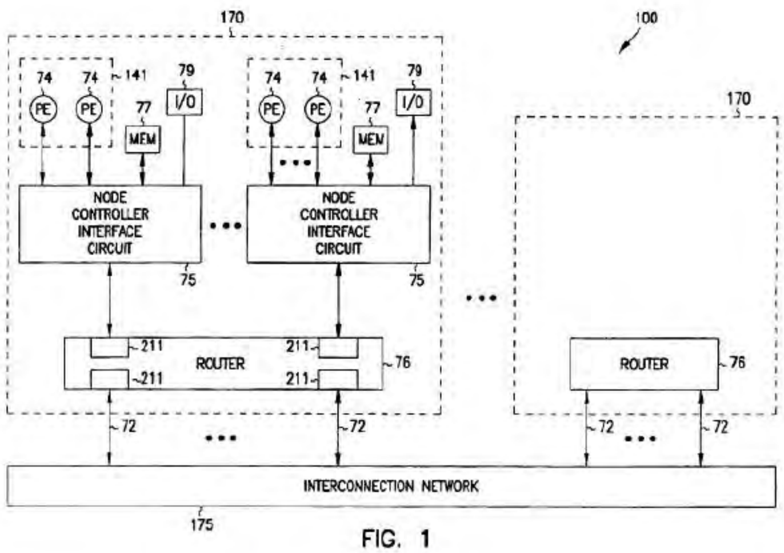
Anderson, Figure 1

- U.S. Patent No. 6,633,945 to Fu: *See, e.g.*, 2:65-3:20 ("The FCU (Flow Control Unit) 220 chip is the central core of the 8P system. The FCU internally implements a switched-fabric data path architecture. Point-to-Point (PP) interconnect 112, 113, and 114 and an associated protocol define dedicated communication channels for all FCU I/O. The terms Channels and PP-Channel are references to the FCU's PP I/O. The FCU provides Point-to-Point Channel interfaces to up to ten Bus Bridge Units (BBUs) 240 and/or CPU Channel Units (CCUs, also known as Chanel Interface Units or CIUs) and one to four Memory Control Units (MCUs) 230. Two of the ten Channels are fixed to connect to BBUs. The other eight Channels can connect to either BBUs or CCUs. In an illustrative embodiment the number of CCUs is eight. In one embodiment the CCUs are packaged as a pair referred herein as a Dual CPU Interface Unit (DCIU) 210. In the 8P system shown, the Dual CPU Interface Unit (DCIU) 210 interfaces two CPUs with the FCU. Throughout this description, a reference to a "CCU" is understood to describe the logical operation of each half of a DCIU 210 and a references to "CCUs" is understood to apply to equally to an implementation that uses either single CCUs or DCIUs 210. CCUs act as a protocol converter between the CPU bus protocol and the PP-Channel protocol.")



Fu, Figure 12

- U.S. Patent No. 6,751,698 to Deneroff: *See, e.g.*, 6:31-47 ("FIG. 1 shows one embodiment of multiprocessor computer system 100 of the present invention having one or more node clusters 170, each node cluster 170 having zero to N processors 74, zero to M memories 77, and zero to I input/output (I/O) subsystems 79. Depending on the needs of a user, interconnection network 175 can be set up as a three-dimensional torus, an N-dimensional hypercube, or any other suitable interconnection network between routers 76. In one embodiment, each router 76 includes eight ports 211, wherein each port 211 can be used to either connect to other routers 76, or to one to N node controllers 75 each having zero or more processor elements (PEs) 74. Thus, in some embodiments, a router 76 can be used as just an interconnection node in the network 175 (i.e., a circuit within block 175 rather than within node cluster 170), having no PEs 74 or memory 77 or I/O subsystems 79, and all of its ports are used to connect to other routers 76.")



Deneroff, Figure 1

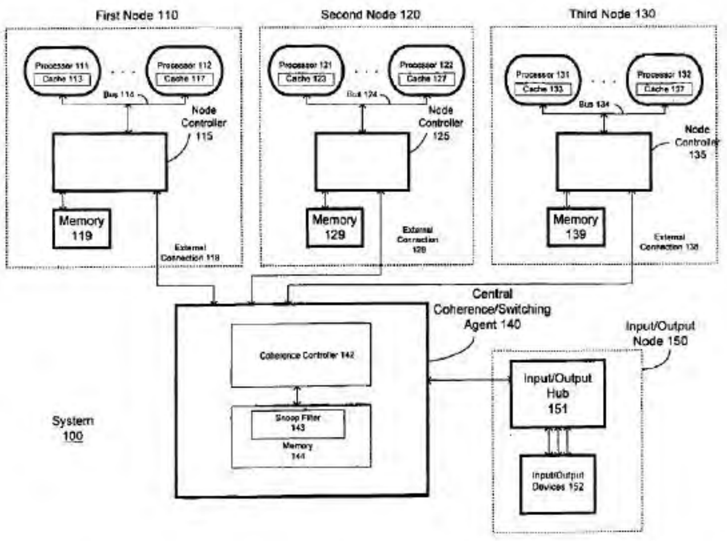
- U.S. Patent No. 7,234,029 to Khare: *See, e.g.*, 3:45-59 ("The details shown in FIG. 1 will now be discussed. As shown in FIG. 1, system 100 includes, for example, first node 110, second node 120, third node 130, and input/output node 150. Each of these nodes is coupled to coherence agent 140. The term "coupled" encompasses a direct connection, an indirect connection, an indirect communication, etc. First node 110 is coupled to coherence agent 140 through external connection 118, second node 120 is coupled to coherence agent 140 through external connection 128, and third-node 130 is coupled to coherence agent 140 through external connection 138. External connections 118, 128, and 138 may be one or more lines capable of communicating information to and from the node. In embodiments of the invention, the nodes may be coupled to each other through direct connections (not shown).")

- Khare: *See, e.g.*, 3:60-4:5 ("First node 110 includes processor 111, processor 112, and node controller 115, which are coupled to each other by bus 114. Processor 111 and processor 112 may be any micro-processors that are capable of processing instructions, such as for example a processor in the INTEL PENTIUM family of processors. Bus 114 may be a shared bus. First node 110 also contains a memory 119 which is coupled to node controller 115. Memory 119 may be a Random Access Memory (RAM). Processor 111 may contain a cache 113, and processor 112 may contain a cache 117. Cache 113 and cache 117 may be Level 2 (L2) cache memories that are comprised of SRAM. Of course, first node 110 may include processors additional to the ones shown (e.g., 111, 112).")

- Khare: *See, e.g.*, 4:54-57 ("In an embodiment, node controller 115, coherence agent 140, and input/output hub 151 may be a chipset that provides the core functionality of a motherboard, such as a modified version of a chipset in the INTEL 815 family of chipsets.")

- Khare: *See*, *e.g.*, 3:23-27 ("In embodiments of the present invention, the coherence agent 140 could be implemented in a central switch for all nodes (as shown in FIG. 1) or, alternatively, the coherence agent could be implemented in a distributed manner integrated in the node controllers of the node(s) (not shown).")

FIG. 1



As illustrated by the prior art references above, it was well known before the priority dates of the Asserted Patents to implement cache coherence/interconnection controllers in a multiprocessor system, at least under Memory Integrity's apparent infringement theories. Further, at least under Memory Integrity's apparent infringement theories, it was well known before the priority dates of the Asserted Patents to implement either a centralized cache coherence/interconnection controller or distributed cache coherence/interconnection controllers in a multiprocessor system. *See, e.g.*, Khare 3:23-27 ("In embodiments of the present invention, the coherence agent 140 could be implemented in a central switch for all nodes (as shown in FIG. 1) or, alternatively, the coherence agent could be implemented in a distributed manner integrated in the node controllers of the node(s) (not shown).") Implementing a central cache coherence/interconnection controller or distributed cache coherence/interconnection controllers would simply be an obvious engineering design choice made by selecting one of a finite number of known options (*i.e.*, central or distributed) according to the desired multiprocessor system implementation. A person of ordinary skill would have indeed been motivated to implement distributed cache coherence/interconnection controllers in a multiprocessor system as described below:

- Ekanadham: *See*, *e.g.*, 1:23-35 ("Technology considerations limit the size of an SMP node to a small number of processors. A method for building a shared-memory multiprocessor with a larger number of processors is to connect a number of SMP nodes with a network, and provide an adapter to extend the SMP's memory across the SMP nodes (see FIG. 1). Existing adapter designs plug into the memory bus of bus-based SMP nodes and collectively provide shared memory across the system, so that any processor in any node can access any location in any memory module in the system. Resources within a node are termed local and resources on other nodes are termed remote.")
- Loewenstein: *See*, *e.g.*, 5:1-8 ("Since global interface 115 is also responsible for maintaining global cache coherency, global interface 115 includes a hardware and/or software implemented cache-coherency mechanism for maintaining coherency between the respective caches and main memories of nodes 110, 120, . . . 180. Cache coherency is essential in order for the system 100 to properly execute shared-memory programs correctly.")

Accordingly, it would have been obvious to implement distributed cache coherence/interconnection controllers to build a shared-memory multiprocessor system with a larger number of processors while maintaining coherency with a reasonable expectation of success. It would have also been obvious to implement distributed cache coherence/interconnection controllers because such a modification would simply be the use of a known technique (*e.g.*, distributed cache coherence/interconnection controllers) to improve similar devices (*e.g.*, multiprocessor systems) in the same way (*e.g.*, improve performance while maintaining coherency). Furthermore, to the extent not disclosed, a person of ordinary skill in the art at the time of the alleged invention of the Asserted Claims would have been motivated to modify the prior art references identified in Section III and Exhibits A-1 – A-9; B-1 – B-19; C-1 –

C-8; D-1 – D-14; and Exhibits E-1 – E-14 to implement distributed cache coherence/interconnection controllers.

a) Interconnection controller operable to facilitate cache coherency

Some of the Asserted Claims are directed to an interconnection controller operable to facilitate cache coherency across the computer system. *See, e.g.*, '206 claim 15.1. At least under Memory Integrity's apparent infringement theories, an interconnection controller operable to facilitate cache coherency across the computer system was well-known in the art at the time of the alleged invention of the Asserted Claims. *See, e.g.*, Exhibits D-1 – D-14, claim 15.1. At least under Memory Integrity's apparent infringement theories, there are many additional exemplary prior art references that disclose an interconnection controller operable to facilitate cache coherency across the computer system. Some examples include:

- *See, e.g.*, Tendler et al., "POWER4 system microarchitecture," at
  - Page 6 ("The two processors share a unified second-level cache, also on the same chip, through a *core* interface unit (CIU), as shown in Figure 1. The CIU is a crossbar switch between the L2, implemented as three separate, autonomous cache controllers, and the two processors. Each L2 cache controller can operate concurrently and feed 32 bytes of data per cycle. The CIU connects each of the three L2 controllers to either the data cache or the instruction cache in either of the two processors. Additionally, the CIU accepts stores from the processors across 8-byte-wide buses and sequences them to the L2 controllers."),
  - Figure 1,
  - Page 7 ("Four POWER4 chips can be packaged on a single module to form an eight-way SMP. Four such modules can be interconnected to form a 32-way SMP. To accomplish this, each chip contains five primary interfaces. To communicate with other POWER4 chips on the same module, there are logically four 16-byte buses. Physically, these four logical buses are implemented with six buses, three on and three off, as shown in Figure 1. To communicate with POWER4 chips on other modules, there are two 8-byte buses, one on and one off. Each chip has its own interface to the off-chip L3 across two 16-bytewide buses, one on and one off, operating at one-third processor frequency."),
  - Page 15 ("The unified second-level cache is shared across the two processors on the POWER4 chip. Figure 5 shows a logical view of the L2 cache. The L2 is

implemented as three identical slices, each with its own controller. Cache lines are hashed across the three controllers."),

- Figure 5,
- Page 16 ("The majority of control for L2 cache management is handled by four coherency processors in each controller. A separate coherency processor is assigned to handle each request to the L2. Requests can come from either of the two processors (for either an L1 data-cache reload or an instruction fetch) or from one of the store queues."),
- Page 16 ("Included in each L2 controller are four snoop processors responsible for managing coherency operations snooped from the fabric."),
- Page 16 ("The L2 cache implements an enhanced version of the MESI coherency protocol . . .").
- Pages 15-18 (section on L2 cache)

- *See*, *e.g.*, Behling et al., "The POWER4 Processor Introduction and Tuning Guide," at
  - Page 15 ("Stores can be sent to the L2 cache at a maximum rate of one store per cycle. Store data is directed to the proper L2 controller (through a hashing function) by way of the storage slice queue (SSQ) and the L2 store queue (STQ)."),
  - Page 17 ("Each POWER4 chip has an L2 cache that is supervised by three L2 controllers, each of which manages 480 KB, for a total L2 size of 1440 KB. Cache lines are hashed across the three controllers. Cache line replacement is implemented as a binary-tree pseudo-LRU algorithm. The L2 cache is a unified cache: it caches instructions, data, and page table entries. The L2 cache is also shared by the processors on the chip."),
  - Page 18 ("Memory coherency in the system is enforced primarily at the L2 cache level by L2 cache controllers. Each L2 has associated command queues, known as coherency processors. Snoop processors within each controller observe all transactions in the system and respond accordingly, providing responses or delivering cache lines if the situation merits."),
  - Page 30 ("The size of the L2 cache is 1440 KB per POWER4 chip, and this is shared between the two processors in the chip. As with the L1 data cache, the cache line size is 128 bytes. The replacement policy is pseudo-LRU (least recently used) so frequently accessed cache lines should be readily maintained in the cache. The L2 cache is a combined data and instruction cache. Instruction caching aspects of the L2 cache are not considered here. The L2 cache is divided into three equal parts, each under control of a separate L2 cache controller. The particular portion a line is stored is in is determined from the real memory address using a hashing algorithm. Sixteen consecutive double-precision Fortran array elements (138 bytes) are held in the same cache line, and therefore under control of the same cache controller. The 17th element will be in a different cache line and the hashing algorithm guarantees it will be stored under control of a different cache controller.").

- *See, e.g.*, IBM POWER4 processor (as disclosed at least in the above references)
- <u>*See, e.g.*, Intel 870 Chipset</u> [redacted]

[redacted] [redacted]

[redacted] [redacted]

[redacted] [redacted]

[redacted] [redacted]

[redacted] [redacted]

Public Version – Confidential Information Redacted and Confidentiality Designation Removed Per Agreement Between the Parties

[redacted]

[redacted] [redacted]

[redacted] [redacted]

[redacted] [redacted]

[redacted] [redacted]

[redacted] [redacted]

[redacted] [redacted]

[redacted] [redacted]

[redacted] [redacted]

[redacted] [redacted]

[redacted]

[redacted] [redacted]

- *See, e.g.*, U.S. Patent No. 7,093,079 to Quach at
  - [2:10-22] ("Referring now to FIG. 1, an example computing device 100 may comprise one or more processor cache nodes 102, one or more input/output (I/O) cache nodes 104, and one or more coherent switches 106 that interconnect the processor nodes 102 and the I/O cache nodes 104. Each processor cache node 102 may comprise one or more processors 108, a node controller (SNC) 110 and memory 112. The processors 108 may execute code or instructions of the memory 112 and may process data of the memory 112 in response to executing such instructions. Further, the processors 108 may have associated caches 114 in which lines of the memory 112 may be stored and accessed more quickly by the associated processors 108."),
  - Figure 1
  - [4:20-32] ("In one embodiment, the snoop filter 204 may be divided into four snoop filter (SF) interleaves 208. While in one embodiment the snoop filter 204 maintains coherency data for all lines of the caches 114, each SF interleave 208 may maintain coherency data for a unique subset of the cache lines. For example, two bits of a line's memory address may be used to identify which of the SF interleaves 208 maintains the coherency data for the particular line. Further, by assigning unique subsets to each of the SF interleaves 208, the SF interleaves 208 may operate in parallel and may increase the overall performance of the coherent switch 106 in comparison to a non-interleaved snoop filter 204 that may process a single request at time."),
  - Figure 2
  - [4:44-60] ("The protocol logic 202 may service transactions such as, for example, requests and responses received from the cache nodes 102, 104 and may issue transactions to the cache nodes 102, 104. In particular, the protocol logic 202 may