

UNITED STATES PATENT AND TRADEMARK OFFICE

---

**BEFORE THE PATENT TRIAL AND APPEAL BOARD**

---

Arm Ltd, MediaTek Inc.,

Petitioners

v.

Daedalus Prime LLC,

Patent Owner

Patent No. 8,984,228  
Case No. IPR2025-00207

---

**PETITION FOR *INTER PARTES* REVIEW**

## **TABLE OF CONTENTS**

|      |                                                                                                                                                                                                          |    |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| I.   | Introduction.....                                                                                                                                                                                        | 1  |
| II.  | The ‘228 Patent’s Overview .....                                                                                                                                                                         | 2  |
| III. | Level of Ordinary Skill.....                                                                                                                                                                             | 4  |
| IV.  | Claim Construction.....                                                                                                                                                                                  | 5  |
| V.   | Statement of Precise Relief Requested.....                                                                                                                                                               | 5  |
| A.   | Ground 1: <i>Agarwal</i> Renders Obvious Claims 1-2, 4-7, and 11-14.....                                                                                                                                 | 6  |
| 1.   | Overview of <i>Agarwal</i> .....                                                                                                                                                                         | 6  |
| 2.   | Independent Claim 1 .....                                                                                                                                                                                | 10 |
| a.   | 1[pre]: “An apparatus comprising:” .....                                                                                                                                                                 | 10 |
| b.   | [1a]: “a multicore processor including a plurality of cores” .....                                                                                                                                       | 11 |
| c.   | [1b]: “a shared cache memory” .....                                                                                                                                                                      | 12 |
| d.   | [1c]: “an integrated input/output (IIO) module to interface between the multicore processor and at least one IO device coupled to the multicore processor” .....                                         | 13 |
| e.   | [1d]: “a caching agent to perform cache coherency operations for the plurality of cores and the IIO module” .....                                                                                        | 15 |
| f.   | [1e]: “the caching agent a single caching agent for the multicore processor and including a plurality of distributed portions each associated with a corresponding one of the plurality of cores.” ..... | 22 |
| 3.   | Claim 2: “The apparatus of claim 1, wherein the caching agent includes a table of requests to store pending requests for the plurality of cores and the IIO module.” .....                               | 24 |

|    |                                                                                                                                                                                                                                                                                                                                         |    |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4. | Claim 4: “The apparatus of claim 1, wherein the shared cache memory comprises a last level cache (LLC) including a plurality of distributed portions each associated with a corresponding one of the plurality of cores.” .....                                                                                                         | 25 |
| 5. | Claim 5: “The apparatus of claim 4, wherein the plurality of distributed portions of the caching agent are coupled via a ring interconnect.” .....                                                                                                                                                                                      | 27 |
| 6. | Claim 6: “The apparatus of claim 1, wherein the caching agent is to receive an allocation transaction from the IIO module, the allocation transaction received in the IIO module from an IO device coupled to the multicore processor.” .....                                                                                           | 28 |
| 7. | Claim 7: “The apparatus of claim 6, wherein the caching agent is to directly store data of the allocation transaction into the shared cache memory and to not initiate a memory transaction to a system memory coupled to the multicore processor.” .....                                                                               | 30 |
| 8. | Independent Claim 11 .....                                                                                                                                                                                                                                                                                                              | 32 |
| a. | 11[pre]: “A method comprising:” .....                                                                                                                                                                                                                                                                                                   | 32 |
| b. | [11a]: “receiving an allocation transaction in an integrated input/output (IIO) module of a multicore processor from an IO device coupled to the multicore processor, the allocation transaction comprising a request to directly store data of the allocation transaction into a shared cache memory of the multicore processor” ..... | 32 |
| c. | [11b]: “forwarding the allocation transaction to a caching agent of the multicore processor, the caching agent shared by a plurality of cores of the multicore processor and the IIO module” .....                                                                                                                                      | 32 |

|     |                                                                                                                                                                                                                                                                                          |    |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| d.  | [11c]: “directly storing the data of the allocation transaction into the shared cache memory of the multicore processor.” .....                                                                                                                                                          | 33 |
| 9.  | Claim 12: “The method of claim 11, further comprising accessing the data in the shared cache memory via at least one of the plurality of cores of the multicore processor a number of times, without sending a transaction to a system memory coupled to the multicore processor.” ..... | 34 |
| 10. | Claim 13: “The method of claim 12, further comprising sending data of the cache line to the system memory from the shared cache memory if the cache line is evicted from the shared cache memory, when the cache line includes modified data.” .....                                     | 36 |
| 11. | Claim 14: “The method of claim 12, further comprising directly storing the data of the allocation transaction into the shared cache memory without communicating with the system memory.” .....                                                                                          | 37 |
| B.  | Ground 2A: <i>Sinharoy</i> Renders Obvious Claims 1, 4, 6, 7 and 11-14. ....                                                                                                                                                                                                             | 37 |
| 1.  | Overview of <i>Sinharoy</i> .....                                                                                                                                                                                                                                                        | 37 |
| 2.  | Independent Claim 1 .....                                                                                                                                                                                                                                                                | 40 |
| a.  | 1[pre]: “An apparatus comprising:” .....                                                                                                                                                                                                                                                 | 40 |
| b.  | [1a]: “a multicore processor including a plurality of cores” .....                                                                                                                                                                                                                       | 40 |
| c.  | [1b]: “a shared cache memory” .....                                                                                                                                                                                                                                                      | 40 |
| d.  | [1c]: “an integrated input/output (IIO) module ...” .....                                                                                                                                                                                                                                | 40 |
| e.  | [1d]: “a caching agent...” .....                                                                                                                                                                                                                                                         | 44 |
| f.  | [1e]: “the caching agent ....” .....                                                                                                                                                                                                                                                     | 46 |

|    |                                                                                                                                                                            |    |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3. | Claim 4: “...wherein the shared cache memory comprises a last level cache (LLC) ....” .....                                                                                | 57 |
| 4. | Claim 6: “...wherein the caching agent is to receive an allocation transaction from the IIO module, ....” .....                                                            | 57 |
| 5. | Claim 7: “... wherein the caching agent is to directly store data ....” .....                                                                                              | 58 |
| 6. | Independent Claim 11 .....                                                                                                                                                 | 59 |
| a. | 11[pre]: “A method comprising:” .....                                                                                                                                      | 59 |
| b. | [11a]: “receiving an allocation transaction ...” .....                                                                                                                     | 59 |
| c. | [11b]: “forwarding the allocation transaction ...” .....                                                                                                                   | 59 |
| d. | [11c]: “directly storing ....” .....                                                                                                                                       | 60 |
| 7. | Claim 12: “... accessing the data in the shared cache memory ... without sending a transaction to a system memory ....” .....                                              | 60 |
| 8. | Claim 13: “... sending data of the cache line to the system memory ....” .....                                                                                             | 61 |
| 9. | Claim 14: “... directly storing ....” .....                                                                                                                                | 62 |
| C. | Ground 2B: Claims 2, 3 and 16-18 Are Obvious over the <i>Sinharoy-Blumrich</i> Combination .....                                                                           | 63 |
| 1. | Overview of <i>Blumrich</i> .....                                                                                                                                          | 63 |
| 2. | Rationale to Combine .....                                                                                                                                                 | 64 |
| 3. | Claim 2: “The apparatus of claim 1, wherein the caching agent includes a table of requests to store pending requests for the plurality of cores and the IIO module.” ..... | 66 |

|    |                                                                                                                                                                                                                                                                                                                    |    |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4. | Claim 3: “The apparatus of claim 2, wherein the table of requests includes at least one dedicated entry to store a posted transaction.” .....                                                                                                                                                                      | 68 |
| 5. | Independent Claim 16 .....                                                                                                                                                                                                                                                                                         | 69 |
| a. | 16[pre]: “A system comprising:” .....                                                                                                                                                                                                                                                                              | 69 |
| b. | 16[a]: “a first multicore processor including a first plurality of cores” .....                                                                                                                                                                                                                                    | 70 |
| c. | 16[b]: “a first shared cache memory” .....                                                                                                                                                                                                                                                                         | 70 |
| d. | 16[c]: “a first integrated input/output (IIO) module to interface between the first multicore processor and a first plurality of IO devices coupled to the first multicore processor” .....                                                                                                                        | 70 |
| e. | 16[d]: “a first caching agent to perform cache coherency operations for the first plurality of cores and the first IIO module,” .....                                                                                                                                                                              | 70 |
| f. | 16[e]: “wherein the first caching agent includes a table of requests having a plurality of entries each to store a pending request for one of the first plurality of cores or the first IIO module, including at least one dedicated entry to store a pending request corresponding to a posted transaction” ..... | 71 |
| g. | 16[f]: “a second multicore processor including a second plurality of cores” .....                                                                                                                                                                                                                                  | 71 |
| h. | 16[g]: “a second shared cache memory” .....                                                                                                                                                                                                                                                                        | 71 |
| i. | 16[h]: “a second IIO module to interface between the second multicore processor and a second plurality of IO devices coupled to the second multicore processor” .....                                                                                                                                              | 71 |

|    |                                                                                                                                                                                                                                                                                                                                                                                                               |    |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| j. | 16[i]: “a second caching agent to perform cache coherency operations for the second plurality of cores and the second IIO module” .....                                                                                                                                                                                                                                                                       | 71 |
| k. | 16[j]: “a peripheral controller coupled to at least one of the first and second multicore processors.” .....                                                                                                                                                                                                                                                                                                  | 72 |
| 6. | Claim 17: “The system of claim 16, wherein the first caching agent is to receive an allocation transaction from the first IIO module, the allocation transaction comprising a request to directly store data of the allocation transaction into the first shared cache memory, and directly store the data of the allocation transaction into the first shared cache memory.” .....                           | 74 |
| 7. | Claim 18: “The system of claim 17, wherein the data in the shared cache memory is to be accessed via at least one of the first plurality of cores a number of times, without sending a transaction to a system memory coupled to the first multicore processor.” .....                                                                                                                                        | 74 |
| D. | Ground 2C: Claim 5 Is Obvious over the <i>Sinharoy-Sistla</i> Combination .....                                                                                                                                                                                                                                                                                                                               | 74 |
| 1. | Overview of <i>Sistla</i> .....                                                                                                                                                                                                                                                                                                                                                                               | 74 |
| 2. | Rationale to Combine .....                                                                                                                                                                                                                                                                                                                                                                                    | 75 |
| 3. | Claim 5: “... coupled via a ring interconnect.” .....                                                                                                                                                                                                                                                                                                                                                         | 77 |
| E. | Ground 2D: Claim 19 Is Obvious over the <i>Sinharoy-Blumrich-Sistla</i> Combination .....                                                                                                                                                                                                                                                                                                                     | 77 |
| 1. | Claim 19: “The system of claim 16, wherein the first caching agent is a single caching agent for the first multicore processor and includes a plurality of distributed portions each associated with a corresponding one of the first plurality of cores, the first shared cache memory comprises a last level cache (LLC) including a plurality of distributed portions each associated with a corresponding |    |

|       |                                                                                                                                                                     |    |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
|       | one of the first plurality of cores, and wherein the plurality<br>of distributed portions of the first caching agent are coupled<br>via a ring interconnect.” ..... | 77 |
| VI.   | The Petition Is Proper Under 35 U.S.C. §325(d) .....                                                                                                                | 78 |
| VII.  | The Petition Is Proper Under <i>Fintiv</i> .....                                                                                                                    | 78 |
| VIII. | Mandatory Notices.....                                                                                                                                              | 81 |
|       | A. Real Party-in-Interest .....                                                                                                                                     | 81 |
|       | B. Related Matters.....                                                                                                                                             | 81 |
|       | C. Lead and Back-Up Counsel.....                                                                                                                                    | 81 |
|       | D. Service Information.....                                                                                                                                         | 82 |
| IX.   | Grounds for Standing.....                                                                                                                                           | 82 |
| X.    | Conclusion .....                                                                                                                                                    | 83 |

## **TABLE OF AUTHORITIES**

### **Cases**

|                                                                                                                                               |        |
|-----------------------------------------------------------------------------------------------------------------------------------------------|--------|
| <i>Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH</i> , IPR2019-01469, Paper 6 (P.T.A.B. Feb. 13, 2020) (precedential) ..... | 79     |
| <i>Apple Inc. v. Fintiv, Inc.</i> , IPR2020-00019, Paper 11 (PTAB March 20, 2020) (precedential).....                                         | 79, 80 |
| <i>Coolit Sys., Inc. v. Asetek Danmark A/S</i> , IPR2021-01195, Paper 10 (P.T.A.B. Dec. 28, 2021) .....                                       | 80     |
| <i>Ericsson Inc. v. Intellectual Ventures I LLC</i> , IPR2014- 00527, Paper 41 (P.T.A.B. May 18, 2015) .....                                  | 37     |

### **LIST OF EXHIBITS**

| <b><u>Exhibit</u></b> | <b><u>Description</u></b>                                                                                                                                                                         |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ex-1001               | U.S. Patent No. 8,984,228 to Liu et al. (“the ’228 patent”).                                                                                                                                      |
| Ex-1002               | Patent Center Prosecution History of U.S. Patent No. 8,984,228.                                                                                                                                   |
| Ex-1003               | Declaration of Dr. Marwan Hassoun.                                                                                                                                                                |
| Ex-1004               | Curriculum Vitae of Dr. Marwan Hassoun.                                                                                                                                                           |
| Ex-1005               | U.S. Patent No. 7,805,575 to Agarwal et al. (“ <i>Agarwal</i> ”).                                                                                                                                 |
| Ex-1006               | B. Sinharoy et al., “IBM POWER7 Multicore Server Processor,” IBM Journal of Research and Development, vol. 5, no. 3, pp. 1:1-1:29 (2011) (“ <i>Sinharoy</i> ”).                                   |
| Ex-1007               | U.S. Patent Appl. Pub. No. 2011/0219188 to Blumrich et al. (“ <i>Blumrich</i> ”).                                                                                                                 |
| Ex-1008               | U.S. Patent Appl. Publ. No. 2008/0126750 to Sistla (“ <i>Sistla</i> ”).                                                                                                                           |
| Ex-1009               | <i>Multicore Processors and Systems</i> (S.W. Keckler et al. eds., 2009) (“ <i>Keckler</i> ”).                                                                                                    |
| Ex-1010               | U.S. Patent Appl. Pub. No. 2006/0041715 to Chrysos et al. (“ <i>Chrysos</i> ”).                                                                                                                   |
| Ex-1011               | G. Blake et al., “A Survey of Multicore Processors,” IEEE Signal Processing Magazine, vol. 26, no. 6, pp. 26-37 (Nov. 2009) (“ <i>Blake</i> ”).                                                   |
| Ex-1012               | V. Getov et al., “Codesign for Systems and Applications: Charting the Path to Exascale Computing,” Computer, vol. 44, no. 11, pp. 19-21 (Nov. 2011).                                              |
| Ex-1013               | T. Zhou, “Deriving Voltage Tolerance Specification for Processor Circuit Design,” 2011 IEEE 20th Conference on Electrical Performance of Electronic Packaging and Systems, pp. 17-20 (Oct. 2011). |

| <b><u>Exhibit</u></b> | <b><u>Description</u></b>                                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ex-1014               | Library of Congress Record: IBM Journal of Research and Development.                                                                                      |
| Ex-1015               | MARC Record: IBM Journal of Research and Development.                                                                                                     |
| Ex-1016               | IBM Journal of Research and Development, vol. 5, no. 3 (2011).                                                                                            |
| Ex-1017               | M.R. Marty, <i>Cache Coherence Techniques for Multicore Processors</i> , Ph.D. Dissertation, University of Wisconsin – Madison (2008) (“ <i>Marty</i> ”). |
| Ex-1018               | Docket Control Order                                                                                                                                      |
| Ex-1019               | Infringement Contentions Cover                                                                                                                            |
| Ex-1020               | IPR2023-01344 Paper 10 05/28/2024 Patent Owner’s Response                                                                                                 |
| Ex-1021               | IPR2023-01344 Paper 20 10/22/2024 Patent Owner’s Sur-reply                                                                                                |
| Ex-1022               | Median Time To Trial                                                                                                                                      |
| Ex-1023               | Declaration of Kelley Hayes Greenfield                                                                                                                    |

## **I. Introduction**

Arm Ltd. and MediaTek Inc. (collectively, Petitioners) petition for *Inter Partes Review* (“IPR”) of claims 1-7, 11-14, and 16-19 (“the Challenged Claims”) of U.S. Patent 8,984,228 (“the ’228 patent”) (Ex-1001). The ’228 patent relates to multiprocessor systems having a plurality of cores, a shared cache memory, and an integrated input/output (“IIO”) module to interface between the multicore processor and at least one I/O device. (Ex-1001, Title, Abstract.) The patent claims an allegedly novel system and method of providing a common caching agent with distributed portions to perform cache coherency for the microprocessor cores and IIO module. (*Id.*)

As references like *Agarwal* (Ex-1005), *Sinharoy* (Ex-1006), *Blumrich* (Ex-1007), and *Sistla* (Ex-1008) demonstrate, there was nothing novel or non-obvious about any of the claimed elements or their combination. The PTO has not ever evaluated *Agarwal* against claims of the ’228 patent or its family. *Sinharoy* is a primary reference in instituted IPR2023-01344 asserted against U.S. Pat. No. 9,575,895 which is a continuation of the ’228 patent.<sup>1</sup>

---

<sup>1</sup> The ‘895 patent’s claim 1 includes all elements of the ‘228 patent’s claim 1 largely verbatim. (Ex-1003 ¶¶46-48.) Thus, if IPR2023-01344 invalidates claim 1

This Petition, supported by Dr. Hassoun's declaration, explains why the claims are invalid. Accordingly, Petitioner requests that the Board institute review and cancel the Challenged Claims.

## **II. The '228 Patent's Overview**

The '228 patent sought to solve a problem resulting from adding "IO components" to integrated circuit chips.

But problems arise once an IO component is integrated on the same chip with a multiprocessor. Traditional IO integration treats the IO component as a separate caching agent, meaning that dedicated logic is associated with the IO component to handle cache coherency operations. When an IO agent is performing read/write operations to main memory, it has to snoop the CPU side cache to maintain cache coherency. In MP systems, this becomes a major scaling problem. .... (Ex-1001, 1:19-28.)

The '228 patent allegedly resolves this problem by disclosing that "an IO agent can be integrated into a multiprocessor socket." (*Id.*, 1:53-55.) A "combined caching agent both for the CPU as well as for an integrated IO agent" provides cache coherency. (*Id.*, 2:19-20). The combined "CPU caching agent ... support[s] both CPU traffic and IO traffic as well, thereby resolving the scaling

---

of the '895 patent based on *Sinharoy*, the Board may institute on that basis alone.

issue.” (*Id.*, 1:55-58, 3:4-12.) The combined “caching agent can be distributed such that each of different portions of the caching agent can be associated with a corresponding core and LLC bank or slice.” (*Id.*, 2:67-3:3).

Claim 1 recites this alleged solution in element [1e]:

the caching agent a **single caching agent** for the multicore processor and including **a plurality of distributed portions** each associated with a corresponding one of the plurality of cores

(Ex-1001, 7:43-46.) Figure 6 depicts a “single caching agent” with distributed portions:

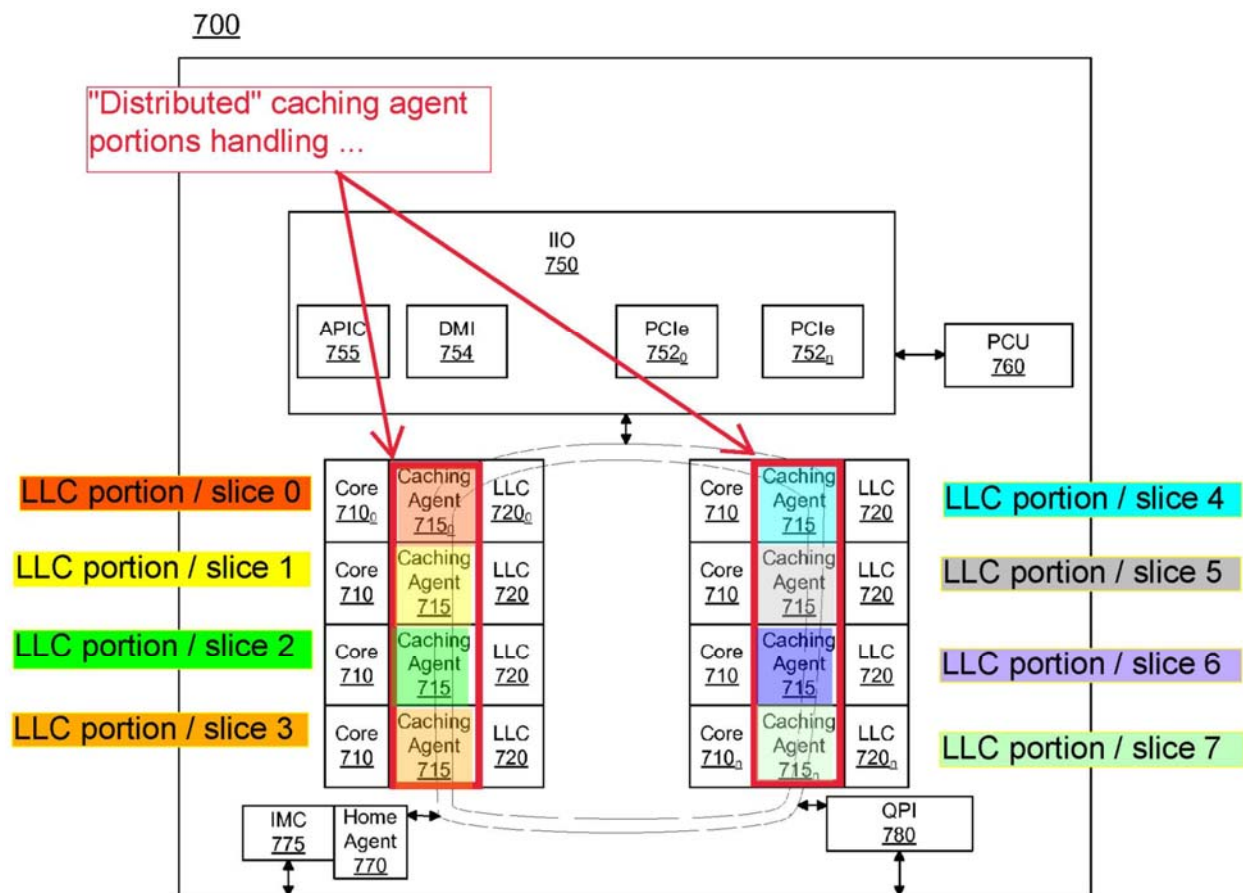


FIG. 6

(Ex-1001, Fig. 6.<sup>2</sup>)

As shown in the embodiment of FIG. 6, processor 700 includes a distributed configuration having partitions or slices each including a core 710 and a partition of a caching agent 715 and a LLC 720. Note that while distributed caching agents are shown, understand that these distributed portions form a single caching agent, and which is configured to handle cache coherency operations both for the cores as well as an IIO module 750.

(Ex-1001, 6:22-29.) Thus, the '228 patent teaches at least one embodiment of a “distributed” single caching agent by having separate distributed portions associated with a slice / portion of LLC that collectively handle the cache coherency operations for the entire multi core and IIO module system.

As further explained below, this approach was already known in the art.

### **III. Level of Ordinary Skill**

A person of ordinary skill in the art (“POSITA”) would have (1) at least a bachelor’s degree in electrical engineering, computer engineering, physics, or a similar field, and (2) approximately two years of industry or academic experience in a field related to microelectronic architecture design, including experience with cache management in multicore systems-on-a-chip (SoCs). (Ex-1003 ¶¶50-51.)

---

<sup>2</sup> Unless otherwise noted, all annotations and emphases herein are added.

More related education would compensate for fewer years of work experience (and vice versa). (Ex-1003 ¶¶50-51.)

#### **IV. Claim Construction**

Given the similarities between the prior art and the '228 patent, no terms require construction to resolve this Petition. Related litigations may involve controversies not presented here that require construction.

#### **V. Statement of Precise Relief Requested**

Petitioners respectfully request review under 35 U.S.C. §311 of claims 1-7, 11-14, and 16-19 and cancellation of those claims as obvious under pre-AIA 35 U.S.C. §103 based on the following unpatentability grounds.

| <b>Grounds</b> |                                                                                                    |
|----------------|----------------------------------------------------------------------------------------------------|
| <b>1</b>       | <i>Agarwal</i> renders obvious claims 1-2, 4-7, 11-14                                              |
| <b>2A</b>      | <i>Sinharoy</i> renders obvious claims 1, 4, 6, 7, 11-14                                           |
| <b>2B</b>      | The combination of <i>Sinharoy</i> and <i>Blumrich</i> renders obvious claims 2, 3, 16-18          |
| <b>2C</b>      | The combination of <i>Sinharoy</i> and <i>Sistla</i> renders obvious claim 5                       |
| <b>2D</b>      | The combination of <i>Sinharoy</i> , <i>Sistla</i> , and <i>Blumrich</i> renders obvious claim 19. |

The '228 patent was filed on December 13, 2011. (Ex-1001, Cover.) All references herein are prior art under pre-AIA 35 U.S.C. §§102(a), (b), and/or (e):

| <b>Exhibit No.</b> | <b>Description</b>                                                                                                                                                                                                                                      |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Ex-1005</b>     | U.S. Patent No. 7,805,575 (“ <i>Agarwal</i> ”), filed May 25, 2007 and issued September 28, 2010; prior art under §§102(a), (b), and (e).                                                                                                               |
| <b>Ex-1006</b>     | B. Sinharoy et al., “IBM POWER7 Multicore Server Processor,” IBM Journal of Research and Development, vol. 5, no. 3, pp. 1:1-1:29 (2011) (“ <i>Sinharoy</i> ”); prior art under §102(a), as detailed in Section V.B.1; Ex-1023 (Greenhill Declaration). |
| <b>Ex-1007</b>     | U.S. Patent Appl. Pub. No. 2011/0219188 (“ <i>Blumrich</i> ”), filed January 18, 2011 and published September 8, 2011; prior art under §§102(a) and (e).                                                                                                |
| <b>Ex-1008</b>     | U.S. Patent Appl. Pub. No. 2008/0126750 (“ <i>Sistla</i> ”), filed November 29, 2006 and published May 29, 2008; prior art under §§102(a), (b), and (e).                                                                                                |

**A. Ground 1: *Agarwal* Renders Obvious Claims 1-2, 4-7, and 11-14**

**1. Overview of *Agarwal***

*Agarwal* teaches the same solution for I/O cache coherency as taught in the ’228 patent: a distributed caching agent in which individual portions of the

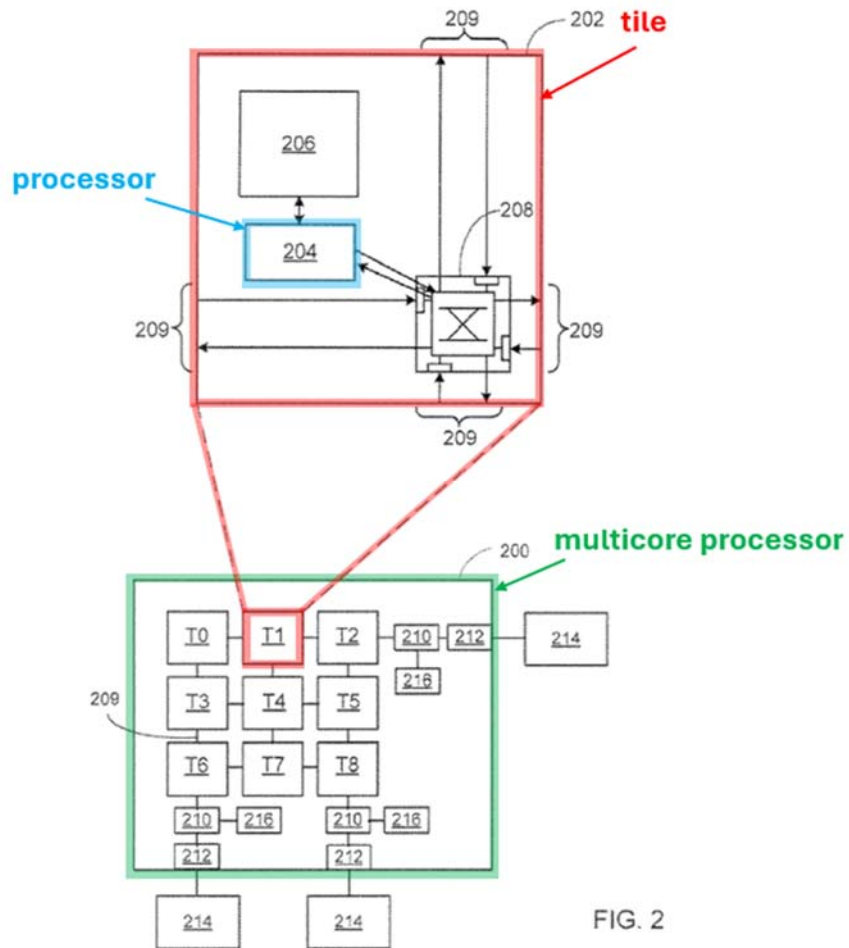
caching agent collectively handle the different portions of a shared cache to form a collective single caching agent that handles cache coherency for both CPU and IO component accesses.

*Agarwal* is an analogous art to the '228 patent. (Ex-1003 ¶¶64-65.) Like the '228 patent, *Agarwal* discloses “[a] multicore processor compris[ing] a plurality of cache memories; [and] a plurality of processor cores, each associated with one of the cache memories... .” (*Agarwal*, Abstract.) *Agarwal* teaches that “[c]ache coherence is a useful mechanism in multiple processor systems” because “[w]hen multiple processors cache a given shared memory location, a cache coherence problem may arise because a copy of the same memory location exists in multiple caches.” (*Agarwal*, 1:34-45.). “Directory based protocols are one way of maintaining cache coherence” using a “directory state and directory controllers.” (*Agarwal*, 1:46-2:42.)

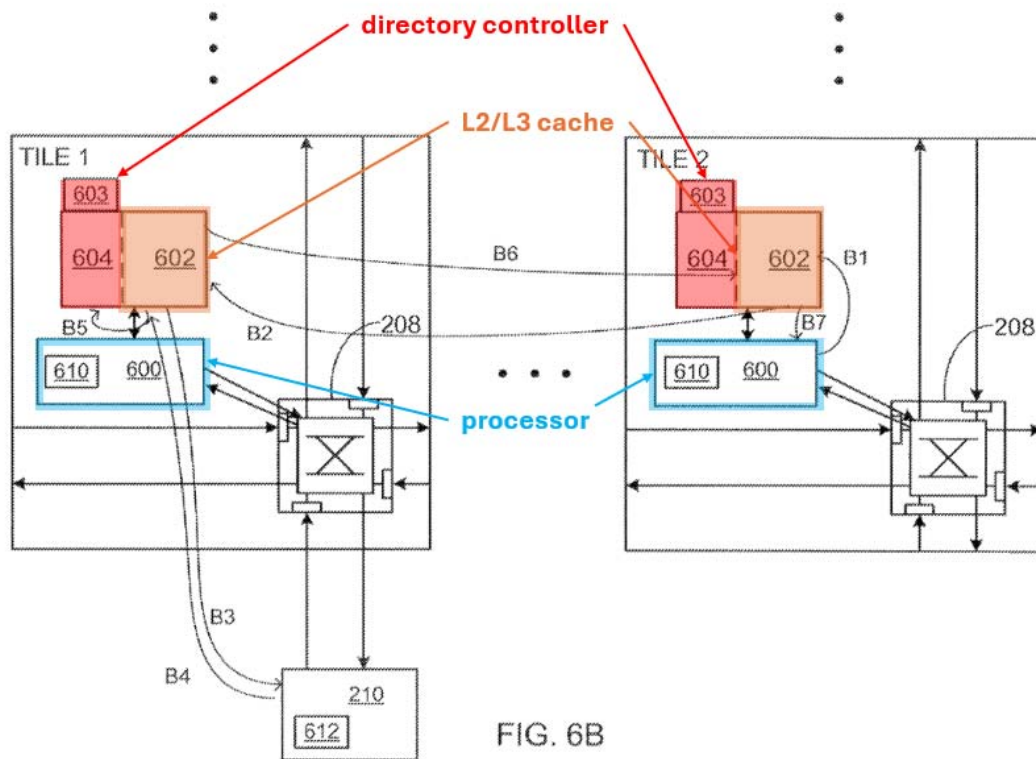
*Agarwal*’s “proxy cache coherence” protocol (“PCC”) teaches the directory controller and directory storage can be distributed across “tiles” (or “nodes”) in a tiled multicore processor. (*Agarwal*, 11:22-26; Ex-1003 ¶¶68-69.) The directory controller(s) and directory storages, collectively, maintain cache coherence – just like the distributed “portions” of the caching agent in the '228 patent’s Fig. 6. *Supra*, Section II; (Ex-1003 ¶¶64, 72-73.)

*Agarwal*’s exemplary multicore processor includes nine instances (T0-T8) of

a tile 202:



(Agarwal, Fig. 2, 7:34-36, 11:53-59; Ex-1003 ¶¶74-77.) Each tile includes an L1 cache 610 and an L2/L3 cache 602 that includes private L2 entries and L3 entries that can be shared by all cores / tiles:



(Agarwal, Fig. 6B, 20:1-16; Ex-1003 ¶¶78-81.) Each tile includes a directory storage to store directory entries and a directory controller coupled to the directory storage. (Agarwal, 2:47-54, 8:38-42, 1:46-50, 11:12-26, 12:28-32.) These entities manage the cache operations for the partition / slice / portion of shared cache assigned to that tile—just like the individual portions shown above in the '228 patent's Figure 6.

Under Agarwal's PCC protocol, a given cache memory location is assigned to the portion of the shared cache on a "home tile." (Agarwal, 16:26-45, 23:51-53.) The home tile performs cache coherency operations as a "proxy" for memory locations assigned to the shared cache portion / partition on that home tile, hence

the term Proxy Cache Coherence. (*Agarwal*, 16:26-45.) The “home” tile assignment may use different approaches. (*Agarwal*, 17:50-51.)

Using an extension of *Agarwal*’s PCC protocol called “ProxyIO coherence,” the home tile manages “I/O coherence” (i.e., cache coherence for an integrated I/O module) in the same manner as managing CPU accesses to the cache.<sup>3</sup> (*Agarwal*, 27:39-28:9; *see also Agarwal*, 16:26-45.) (Ex-1003 ¶¶94-97.)

## 2. Independent Claim 1

### a. 1[pre]: “An apparatus comprising:”

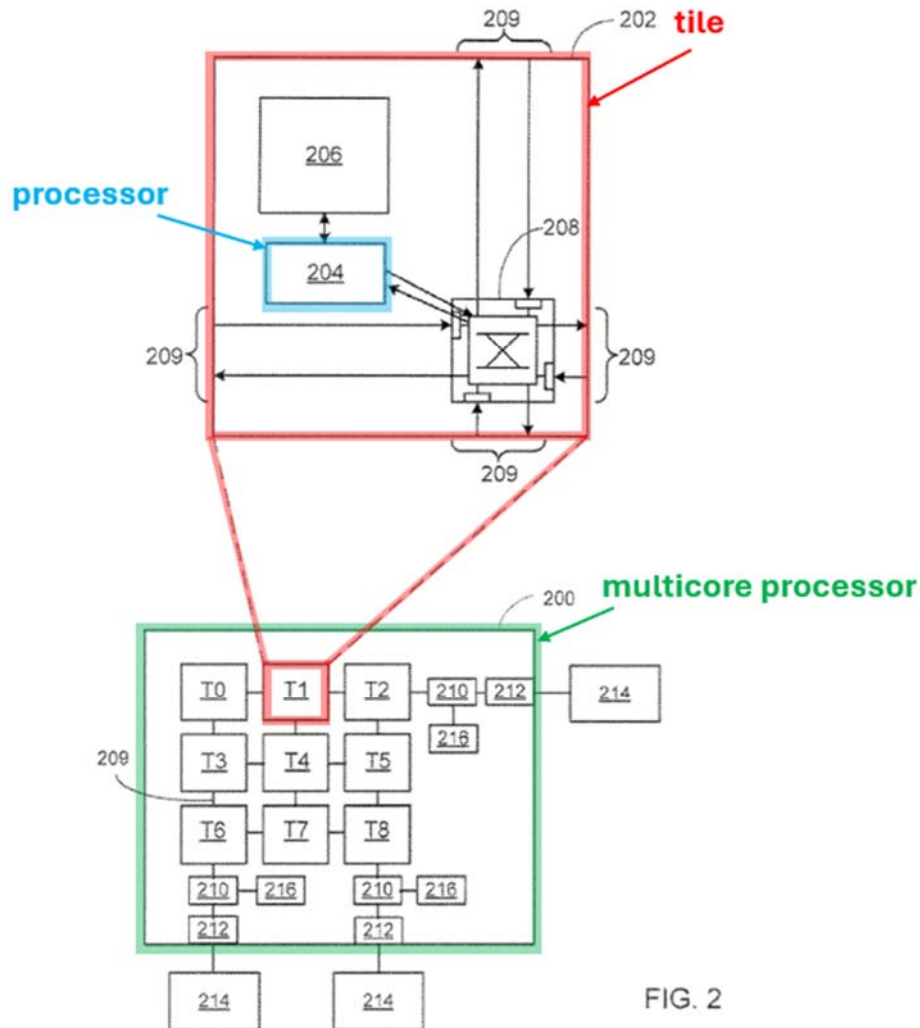
*Agarwal* discloses “a multicore processor,” which is an apparatus as detailed below. (*Agarwal*, Abstract, 1:32-33, Fig. 2; Ex-1003 ¶¶119-121.)

---

<sup>3</sup> A POSITA would have understood *Agarwal*’s Figures 2, 6A-B, and 9A-C describe the same invention. (Ex-1003 ¶¶97.) Figure 2 illustrates an “exemplary multicore processor 200 having a tiled processor architecture.” (*Agarwal*, 11:53-54.) Figures 6A-6B use the exemplary architecture to illustrate the “proxy cache coherence (PCC) scheme.” (*Agarwal*, 10:25-26.) Figures 9A-9C illustrate the exemplary architecture “with an I/O module” to explain how “an extension of the proxy cache coherence approach” operates to ensure “I/O coherence.” (*Agarwal*, 10:31-32, 27:40-46.)

**b. [1a]: “a multicore processor including a plurality of cores”**

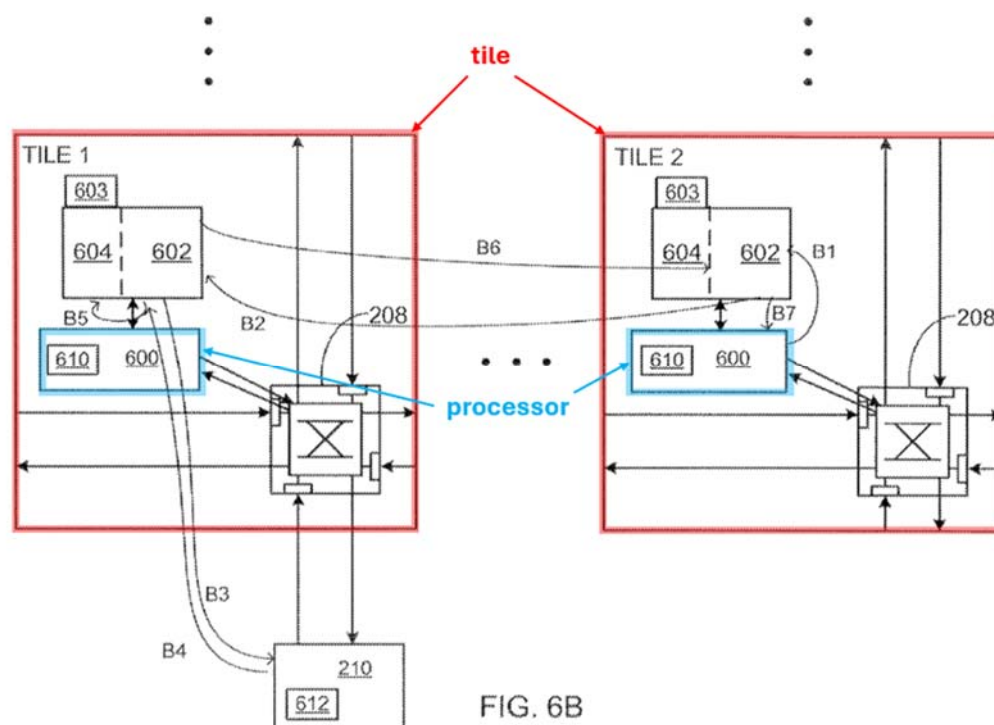
*Agarwal* discloses element [1a]. (Ex-1003 ¶¶122-130.) *Agarwal*’s Fig. 2 discloses a multicore processor 200 with nine instances (T0-T8) of a tile 202:



(*Agarwal*, Fig. 2, 7:34-36, 11:53-59; Ex-1003 ¶¶123-126.) Each tile includes a processor corresponding to one of the claimed “plurality of cores.” (*Agarwal*, 1:32-33, 7:34-37, 11:53-58, Fig. 2; Ex-1003 ¶127.)

*Agarwal*’s Figure 6B shows a multicore processor with multiple tiles, each

including a processor (i.e., “plurality of cores”):



(*Agarwal*, Fig. 6B, 20:1-4; Ex-1003 ¶¶128-130.) Thus, *Agarwal* discloses a multicore processor with multiple cores. (Ex-1003 ¶131.)

**c. [1b]: “a shared cache memory”**

*Agarwal* discloses element [1b]. (Ex-1003 ¶¶131-142.) *Agarwal* discloses “a shared distributed cache.” (*Agarwal*, 23:47-48.) *Agarwal*’s L2/L3 cache can be shared by multiple processor cores. (*Agarwal*, 10:1-10, 11:35-41, 18:39-47, 20:12-16, 23:45-63.) For example, Figure 6B (annotated below) shows each tile includes an L2/L3 cache 602 whose L3 entries can be shared by all tiles’ cores (i.e., “a shared cache memory”):

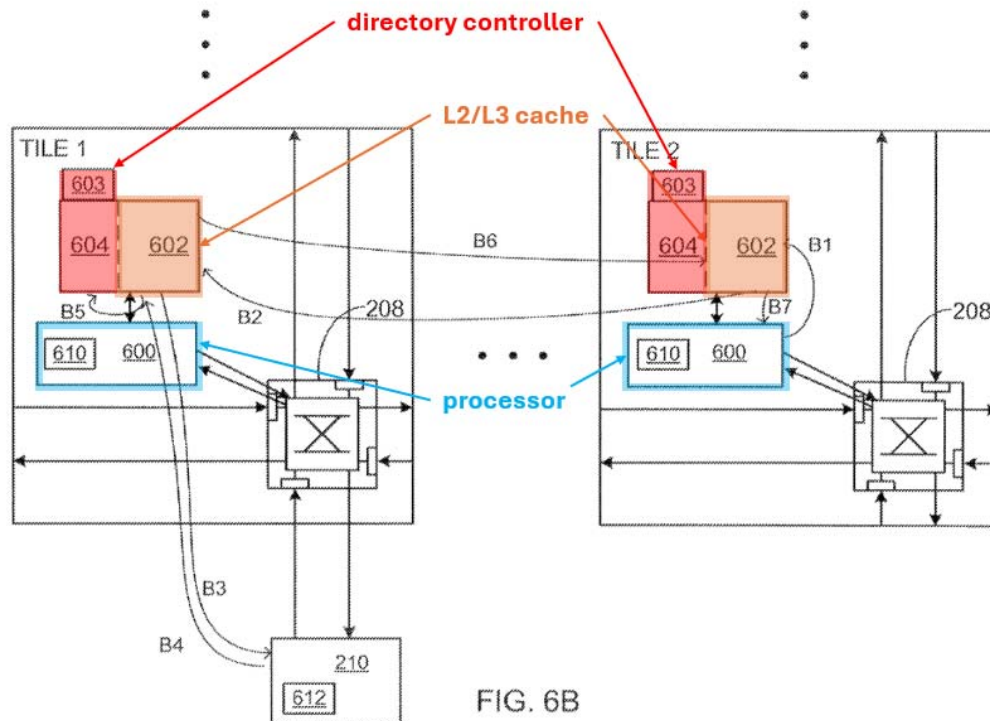


FIG. 6B

(*Agarwal*, Fig. 6B, 20:1-16, 23:45-51; Ex-1003 ¶¶138-141.) *Agarwal* thus discloses a shared cache memory, as claimed. (Ex-1003 ¶142.)

**d. [1c]: “an integrated input/output (IIO) module to interface between the multicore processor and at least one IO device coupled to the multicore processor”**

*Agarwal* discloses<sup>4</sup> element [1c]. (Ex-1003 ¶¶143-162.) *Agarwal*’s multicore processor ([1a]) includes an “I/O module,” which is the claimed IIO module. (*Agarwal*, 10:31-32; *see also id.* at Figs. 9A-9C; Ex-1003 ¶144.)

---

<sup>4</sup> As used herein, “discloses” encompasses what a reference would teach, suggest, or render obvious to a POSITA.

*Agarwal's* IIO module can access memory. (*Agarwal*, 27:40-42.) *Agarwal* explains that the multicore processor must therefore be able to maintain cache coherence in the presence of the I/O module. (*Agarwal*, 27:42-46.) *Agarwal's* Figures 9A-9C illustrate cache coherence operations of the multicore processor ([1a]) with an IIO module. (*Agarwal*, 10:31-32; Figs. 9A-9C; Ex-1003 ¶¶145-149.) As Figure 9B shows, multicore processor 900 includes an “I/O module 912” integrated onto the same multicore processor as the plurality of processor cores.

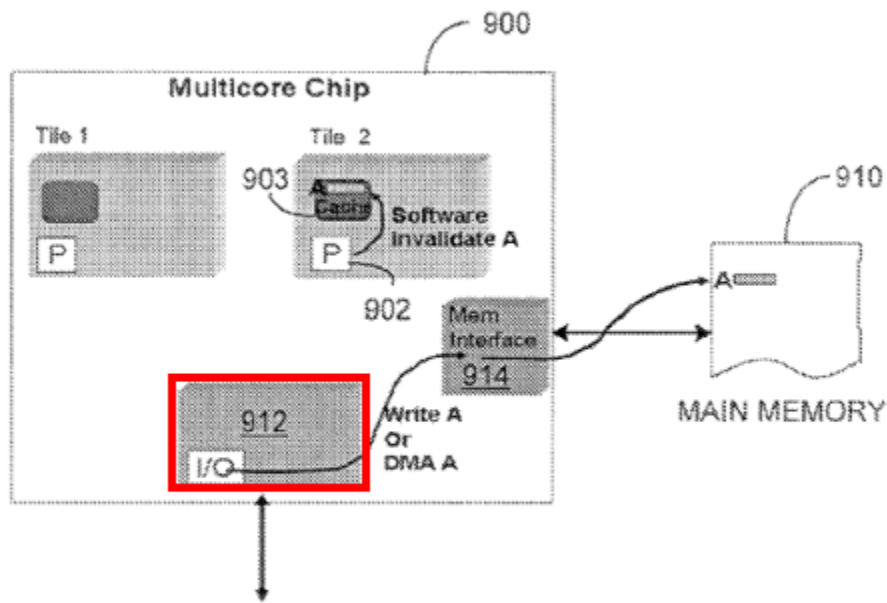


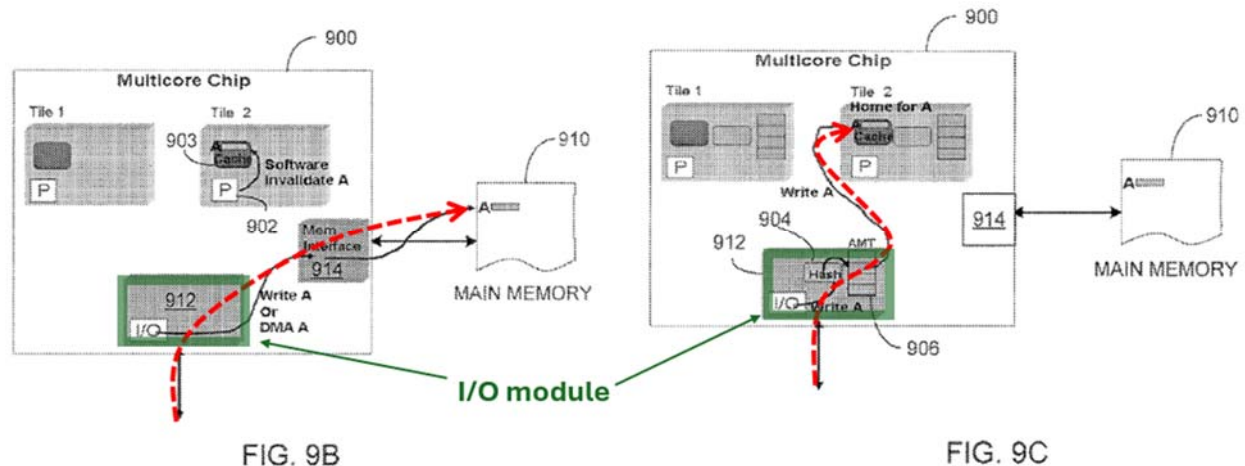
FIG. 9B

(*Agarwal*, Fig. 9B, 28:11-22, 29:5-7; Ex-1003 ¶¶150-152.) Figure 9B above also shows L2/L3 cache 903 in each tile (equivalent to cache 602 in *Agarwal's* Fig. 6B). (*Agarwal*, 28:44-47.)

I/O module 912 interfaces between the multicore processor and an IO device

coupled to the multicore processor. (Ex-1003 ¶¶156-157.) For example, *Agarwal*'s I/O module 912 “includes an I/O interface 913 such as a network port (e.g., a XAUI interface or a PCI-e interface).” (*Agarwal*, 28:11-22.)

*Agarwal*'s I/O module 912 supports transactions that send data from an off-chip IO device (1) directly to “main memory” (Fig. 9B and 28:11-22, 28:33-43) or (2) directly into the L2/L3 cache (Fig. 9C and 28:23-25, 29:5-7). The annotated figures below show both transactions and illustrate how I/O module 912 interfaces between multicore processor 900 and an IO device (not shown) coupled to the multicore processor:



(*Agarwal*, Fig. 9B (left), Fig. 9C (right); Ex-1003 ¶¶158-161.)

**e. [1d]: “a caching agent to perform cache coherency operations for the plurality of cores and the I/O module”**

*Agarwal* discloses element [1d]. (Ex-1003 ¶¶163-193.) To manage cache coherency, *Agarwal* teaches several cache coherence management techniques,

including Shim Cache Coherence (SCC) and Proxy Cache Coherence (PCC) protocols. (*Agarwal*, 16:26-20:34, 27:39-30:4.) Both are directory-based protocols used to maintain cache coherence (*Agarwal*, 1:46-50, 2:21-24). Both protocols employ: (1) a directory controller and (2) directory storage that collectively operate as the claimed “caching agent” to perform cache coherency operations for *Agarwal*’s plurality of cores and IIO module. (*Agarwal*, 2:47-54, 8:38-42, 1:46-50, 11:12-26, 12:28-32, 27:39-28:9) (Ex-1003 ¶¶166-174.)

Under *Agarwal*’s PCC protocol, the directory controller and directory storage are distributed across the multicore processor tiles.<sup>5</sup> (*Agarwal*, 11:22-26; Ex-1003 ¶175.) A particular cache operation is managed by one of the tiles in the multiprocessor, referenced as the “home tile” for that cache memory operation (*Agarwal*, 16:26-45, 23:51-53.) The home tile, including its directory controller and directory storage, performs cache coherency operations as a “proxy” for cache memory locations assigned to that home tile, giving rise to the term Proxy Cache

---

<sup>5</sup> The directory storage 604 “can be included within the same memory that provides the cache 602, or can be included in a separate memory module on the tile.” (*Agarwal*, 16:39-42.) The “directory controller 603 is coupled to the directory storage 604.” (*Agarwal*, 16:44-45.)

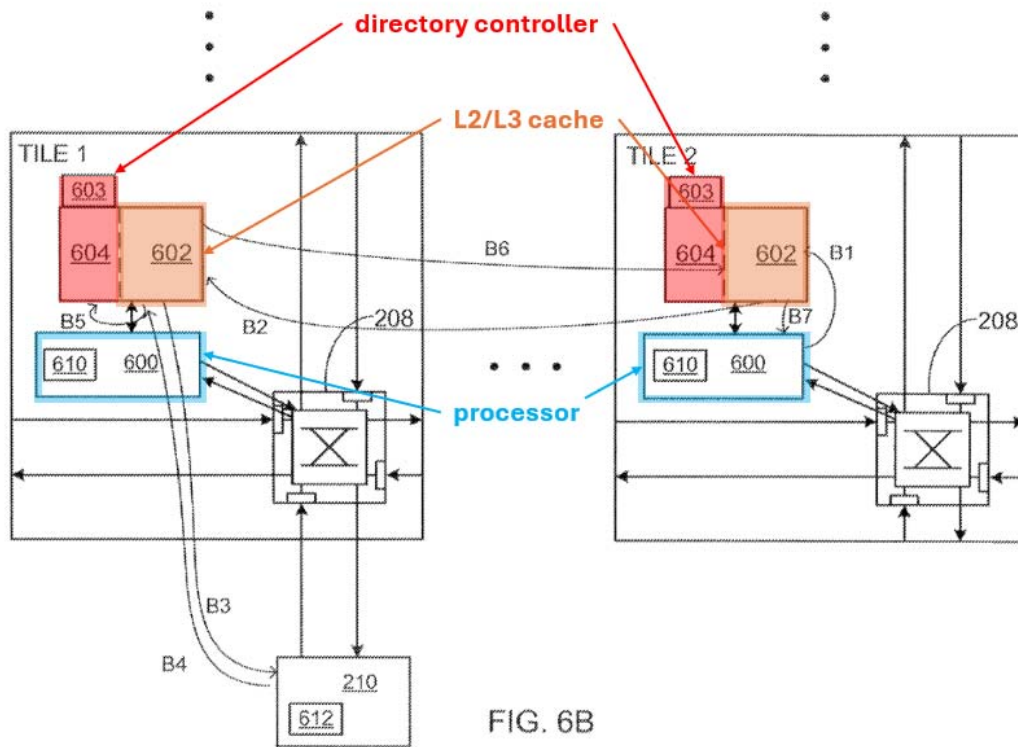
Coherence (PCC). (*Agarwal*, 16:26-45; Ex-1003 ¶¶176-177.) The assignment of a “home” tile to manage cache operations can be “based on a function of the address such as a hash function.” (*Agarwal*, 17:50-51.)

In *Agarwal*, just like in the ’228 patent, the caching agent portions at each CPU (tile) manage the cache coherency for a portion / slice of the shared cache. Both *Agarwal* and the ’228 patent show a single caching agent with distributed portions associated with each processor core (tile) that manages cache coherency for accesses by CPUs and I/O components. (*Agarwal*, Figs. 2, 6A-6B, 9C, 27:39-28:8, 29:3-30:12; Ex-1001, Fig. 6, 2:66-3:3). ( Ex-1003 ¶¶178-181.)

Comparing *Agarwal*’s different embodiments highlights the distributed nature of the cache control portions in the PCC protocol. *Agarwal* discloses that the SCC and PCC schemes work similarly. One difference is that in SCC the directory controller (i.e., coherency controller) and directory storage are centralized (e.g., DCC 216 with directory adjacent to an on-chip memory controller shim, *Agarwal*, 11:15-21 and FIG. 2). In contrast, with PCC, the directory controller and directory storage are distributed across the tiles—e.g., directory controller 603 and directory storage 604 within each core in the multicore system. (*Agarwal*, 16:27-34, 11:11-26, Ex-1003 ¶¶182-183).

*Agarwal* depicts a PCC tile architecture configured in Figures 6A-6B. Each tile includes a processor 600, L2/L3 cache 602, directory controller 603, and

directory storage 604:



(Agarwal, Fig. 6B; Ex-1003 ¶¶184-186.)

Each directory controller 603 and directory storage 604 manages cache coherence for the cache memory locations assigned to that home tile regardless of whether the cache access is from a tile (CPU) or an I/O module. (Agarwal, 29:25-29; 27:39-28:9, Ex-1003 ¶¶185-186.)

Thus, when a given cache line is accessed (for both I/O module access and CPU access to cache), the directory controller and directory storage assigned to the “home tile” for the shared cache partition that includes that cache line manage the coherence. All directory controllers 603 and directory storages 604, collectively

for all tiles, thus manage cache coherence for the entire shared cache (and memory space). Thus, *Agarwal*'s distributed components teach, or render obvious, a “single” caching agent with distributed portions managing cache coherence for the plurality of processor cores and the IIO module in the same manner as the '228 patent teaches for distributed caching agent partitions managing memory partitions / slices. (Ex-1001, Fig. 2, 2:66-3:3, Fig. 6, 6:20-35; Ex-1003 ¶186.)

***Coherence operations for the “plurality of cores”:*** Figure 9A describes *Agarwal*'s cache coherence operations for the plurality of cores:

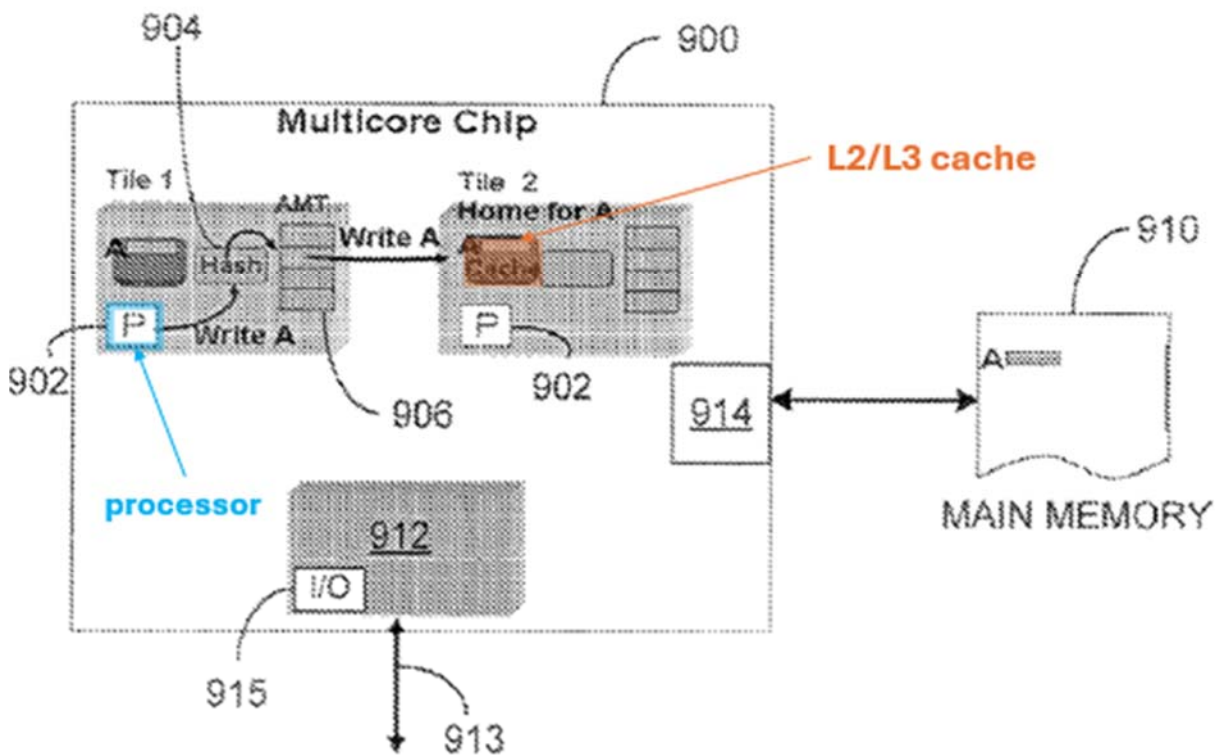


FIG. 9A

(*Agarwal*, Fig. 9A; Ex-1003 ¶¶187-188.) When Tile 1's processor 902 writes data

to memory location having address A, Tile 1 sends the write request to the home tile of address A (Tile 2) based on a hash function and an address mapping table (AMT) or TLB. (*Agarwal*, 27:46-28:8, 29:17-30:11, 23:6-53, 27:32-38, 17:42-67, 19:3-29.) The home tile (Tile 2) performs cache coherence operations as a proxy and writes the data into a shared cache line located in the L2/L3 cache in Tile 2. (*Agarwal*, 27:46-28:8; *see also Agarwal*, 20:1-16, Fig. 6B.)

***Coherence operations for the IIO:*** *Agarwal*'s distributed caching agent portions also manage access to the cache by IO components in the same manner as taught/claimed in the '228 patent. In *Agarwal*, in an extension of the PCC protocol called "ProxyIO coherence," the home tile manages cache coherence in the presence of an I/O module ("I/O coherence"). (*Agarwal*, 27:39-28:9); (*Agarwal*, 16:26-45, 29:3-30:11.) "Coherence processing at the home tile for memory access requests from the I/O module 912 ("I/O coherence") includes a set of actions that are similar to those for PCC." (*Agarwal*, 29:29-31.) (Ex-1003 ¶¶190-193.)

Figure 9C describes *Agarwal*'s ProxyIO cache coherence operations for the integrated I/O module:

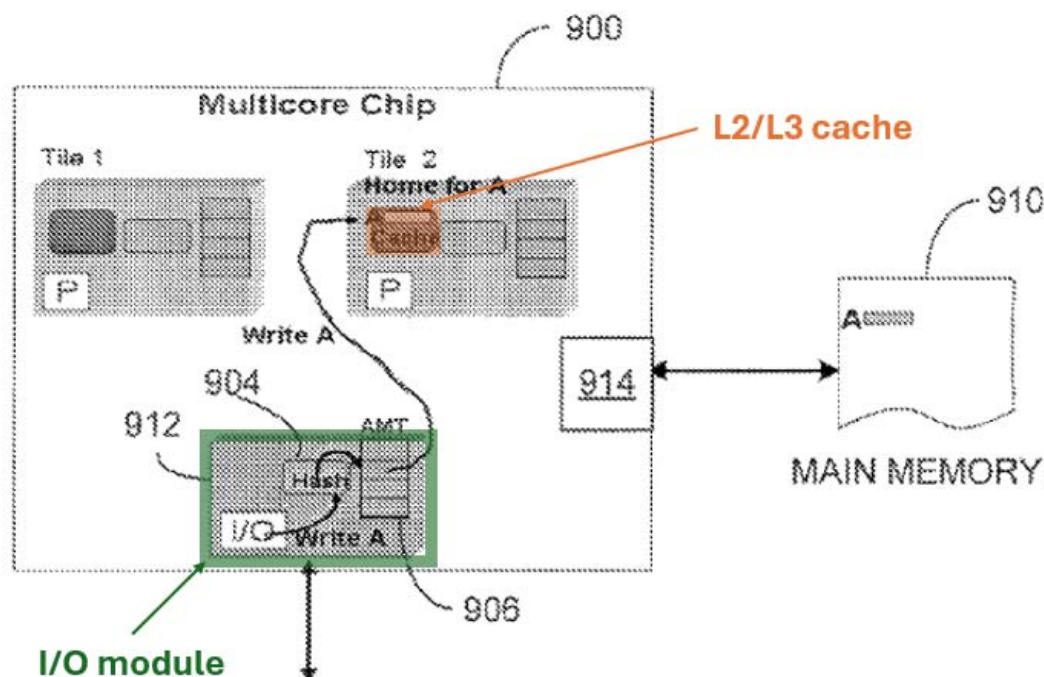


FIG. 9C

(Agarwal, Fig. 9C, 29:5-30:8; 27:39-28:9; Ex-1003 ¶¶190-191.) Agarwal's I/O module 912 includes circuitry implementing a hash function and address mapping table (AMT) used to deliver a transaction to the "home" tile of the address A (Tile 2). (Agarwal, 29:17-20.)

FIG. 9C illustrates an example in which a ProxyIO coherence protocol is used for situations in which an I/O module 912 writes or reads data to or from a tile's cache.... The home tile accepts the write transactions from the I/O module 912 just as it would write transactions from a tile. The home tile performs the normal coherence processing, with some differences as noted below ....

(*Agarwal*, 29:5-28 (emphasis added));

*Agarwal*'s "ProxyIO" is the same proxy I/O coherence concept the '228 patent describes, explaining that "by proxying I/O-related transactions through a CPU caching agent as opposed to having separate dedicated IO agents," "a better scaling profiling and performance advantages can be realized." (Ex-1001, 6:5-7, 6:35-37.)

Thus, in the same manner as the '228 patent teaches, *Agarwal* teaches distributed caching agent portions each managing the partition / portion / slice of shared cache assigned to that portion (e.g., the "home tile" for the shared distributed cache/ portion in *Agarwal*). (Ex-1003 ¶193.)

**f. [1e]: "the caching agent a single caching agent for the multicore processor and including a plurality of distributed portions each associated with a corresponding one of the plurality of cores."**

*Agarwal* discloses element [1e].<sup>6</sup> (Ex-1003 ¶¶195-203.) As discussed for [1d], *Agarwal* discloses a PCC protocol with distributed L3 caches and cache management in each tile's directory controller and directory storage. (*Supra* [1d];

---

<sup>6</sup> The language "the caching agent a single caching agent" lacks a verb. If this does not render claim 1 indefinite, Petitioner interprets this language to mean "the caching agent [is/forms/constitutes] a single caching agent."

*Agarwal*, 16:26-20:45, 27:39-30:4, Figs 6A-6B.) Under this protocol, a given cache location is assigned to a home tile, such that the shared distributed cache is partitioned based on the number of tiles, and each partitioned cache region is assigned to one home tile. This matches the '228 patent discussed above in which the distributed portions of each caching agent managed specific portions (banks / slices) of the shared LLC cache partitioned between those shared distributed caching agent portions. (*Supra* [1d]; Ex-1003 ¶196.) Thus, *Agarwal* teaches or renders obvious a plurality of distributed portions of a “single caching agent” for the whole multicore processor as detailed above. (Ex-1003 ¶¶197-200.)

Claim 1 requires that the caching agent portions are “each associated with a corresponding one of the plurality of cores.” The '228 patent teaches that “the caching agent can be associated with a corresponding core *and* LLC bank or slice.” (Ex-1001, 2:67-3:3.) The claimed association is with the core.

*Agarwal* teaches this association explicitly, as [1d] detailed above regarding Figs. 6A-6B. For example, *Agarwal* teaches “[a]t least some of the directory controllers are associated with a processor core....” (*Agarwal*, 3:42-44, claim 9). Similarly, *Agarwal* teaches “[e]ach of at least some of the processor cores includes a directory controller...” and “at least some of the storage for directory state.” *Agarwal*, 4:7-8, 6:35-36, claims 14, 63. Each directory controller 603 / directory storage 604 pair (“distributed portion [of the caching agent]”) is “associated with”

a processor 600 within each tile (i.e., “corresponding one of the plurality of cores”). (*Supra* [1d]; Ex-1003 ¶202.)

**3. Claim 2: “The apparatus of claim 1, wherein the caching agent includes a table of requests to store pending requests for the plurality of cores and the IIO module.”**

*Agarwal* discloses claim 2. (Ex-1003 ¶¶204-213.) The ’228 patent specification defines a table of requests (TOR) as “a buffer or other temporary storage for holding incoming requests.” (Ex-1001, 2:46-48.)

*Agarwal* explains each “tile can ... keep a small buffer that contains a list of outstanding transactions with an acknowledgement counter associated with each [outstanding transaction].” (*Agarwal*, 27:21-25; *see also id.*, 16:1-5, 20:53-54.) *Agarwal* identifies this “small buffer” as being part of “cache state that is used to store directory state” (*Agarwal*, 27:11-25.) Moreover, in *Agarwal*, “each directory controller ... is configured to maintain directory state....” (*Agarwal*, 3:39-41, 4:9-12, claims 8, 15.) *Agarwal* teaches a “directory storage 604 for directory state.” *Agarwal*, 16:39-42. Thus, a POSITA would have understood this buffer is part of the directory controller / directory storage (part of the “caching agent”), which maintains coherence for the incoming requests or, at a minimum, that it would have been obvious to include this buffer as part of the directory controller / directory storage for each tile to allow for fast, efficient access in coherency operations. (Ex-1003 ¶¶205-208.)

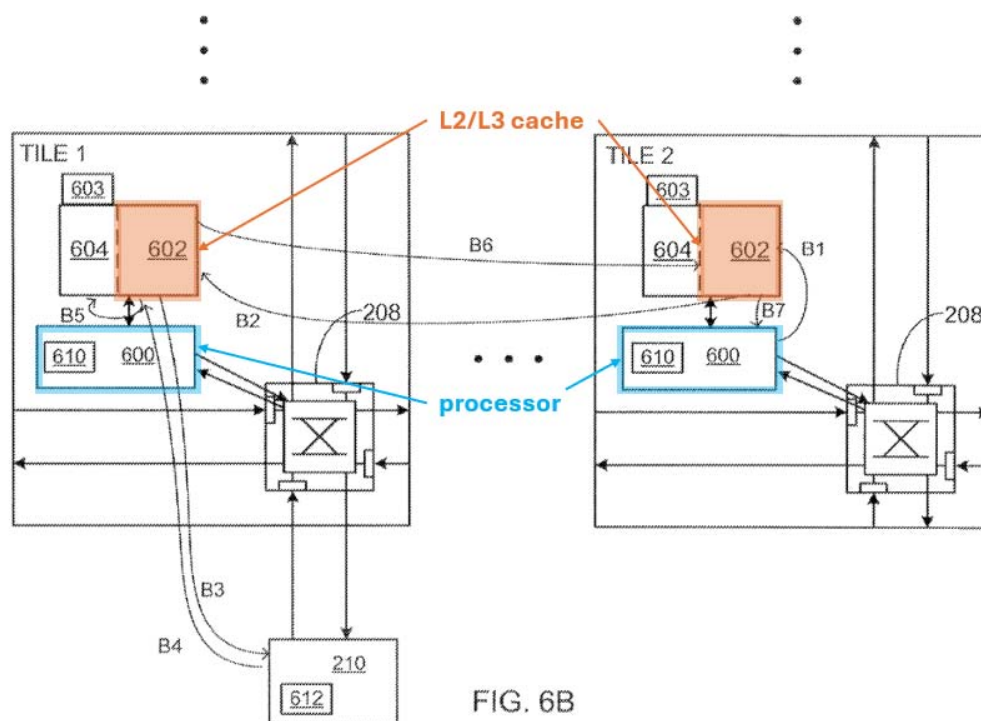
A POSITA further would have understood that a buffer would beneficially be used to operate *Agarwal*'s directory controller, so it can track outstanding requests. Moreover, it would have been obvious that such a buffer is beneficially part of the directory controller / directory access for timely and efficient access in managing coherence for cache access requests. (Ex-1003 ¶¶209-210; *see also Blumrich* ¶[0118]-[0128] (disclosing the use of a request queue for directory based coherence); *infra* V.C.3 V.C.2, V.C.3.) Using this buffer, the directory controller stores the list of outstanding transactions (i.e., “pending requests”) and associated acknowledgment counters involving the plurality of cores and I/O module discussed above in [1a], [1c], and [1d]. (*Agarwal*, 27:11-25, 5:32-37; Ex-1003 ¶¶210-212.)

**4. Claim 4: “The apparatus of claim 1, wherein the shared cache memory comprises a last level cache (LLC) including a plurality of distributed portions each associated with a corresponding one of the plurality of cores.”**

*Agarwal* discloses claim 4. (Ex-1003 ¶¶214-222.) A POSITA would have understood that the claimed “last level cache (LLC),” refers to the highest-level cache shared by all cores. E.g., Ex-1001, 2:14-17 (“one or more levels of cache memories can be present within the cores. ... a shared cache memory 130, ... may be a last level cache (LLC)”), 6:30-36; (Ex-1003 ¶215.)

As discussed for [1b], *Agarwal* discloses an L2/L3 cache 602 for each tile in

the multicore processor with the L3 cache lines being shared among all the tiles (i.e., “a last level cache (LLC”). (*Supra* [1b]; *Agarwal*, 20:1-16, Fig. 6B.) Each tile’s L2/L3 cache 602 is contained within that tile. E.g., *Agarwal*, 20:12-16 (“cache 602 can serve as both a local L2 cache for the local tile and as a shared L3 cache for other tiles”); 23:45-50 (“cache 206 ... can be configured to cache data in two ways: (a) as part of a shared distributed cache (equivalent to a shared global/distributed L3 cache) ...”); Fig. 2. *Agarwal* repeatedly references the “core associated with [one of the] the cache memory/memories.” *Agarwal*, Abstract, 3:63-64, 4:28, 4:42-43, 7:23-24, 9:32-33. Thus, the shared L3 cache is “associated with” a corresponding processor 600 for that tile:



(*Agarwal*, Fig. 6B; Ex-1003 ¶¶216-220; *see also Agarwal*, 16:35-45, Fig. 6A.) The shared L3 cache lines of the L2/L3 cache 602 “serve ... as a shared L3 cache for other tiles.” (*Agarwal*, 20:12-16; *see also Agarwal*, 23:45-51 (a “shared distributed cache.”).)

Collectively, the shared L3 cache lines constitute “a plurality of distributed portions each associated with a corresponding one of the plurality of cores” because the shared L3 cache lines are distributed across all cores (e.g., Tiles 1 and 2 above), each L3 partition being associated with a single core. (Ex-1003 ¶222.)

**5. Claim 5: “The apparatus of claim 4, wherein the plurality of distributed portions of the caching agent are coupled via a ring interconnect.”**

*Agarwal* discloses claim 5. (Ex-1003 ¶¶223-226.) *Agarwal*’s mesh network may connect (via data paths) all the tiles—including the processor cores, L2/L3 caches, and directory controllers—through switches 208. (*Agarwal*, 11:53-64, Fig. 2.)

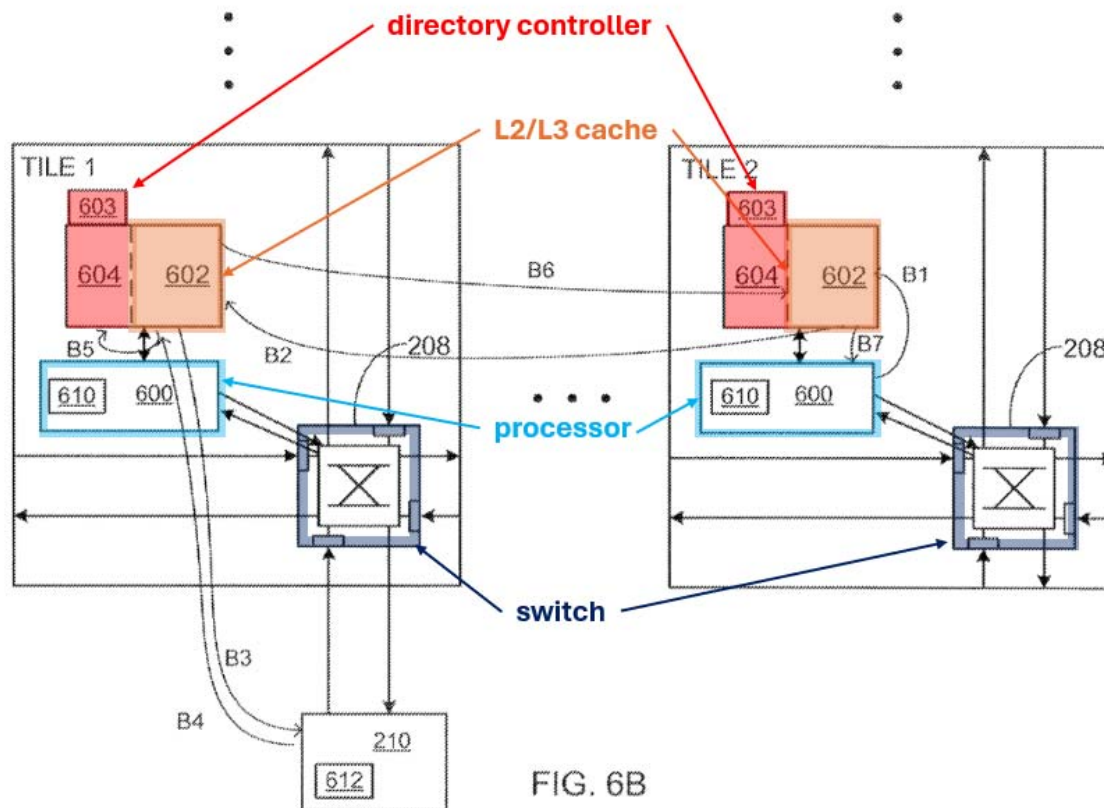


FIG. 6B

(*Agarwal*, Fig. 6B; Ex-1003 ¶226.) As an alternative to the mesh network, *Agarwal*'s tiles (including directory controllers) "are interconnected over a network that includes ... a ring...." (*Agarwal*, 3:45-47, 7:16-18, 11:66-12:2, claim 10.) *Agarwal* explains that cache coherency can similarly be maintained using such a ring interconnect topology for the multicore processor. (*Agarwal*, 20:63-67.)

**6. Claim 6: "The apparatus of claim 1, wherein the caching agent is to receive an allocation transaction from the IIO module, the allocation transaction received in the IIO module from an IO device coupled to the multicore processor."**

*Agarwal* discloses claim 6. (Ex-1003 ¶¶227-235.) In the '228 patent, an allocation transaction can be generated by an I/O device to "lodge data directly into

the [last level cache (LLC)].” (Ex-1001, 3:22-25); “[T]his allocation transaction can be a request to write data directly into a cache memory of the processor, and it can be received from an IO device.” (Ex-1001, 3:47-51)

*Agarwal* discloses this feature. (Ex-1003 ¶¶229-234.) “An I/O module is also able to ... store data directly into the cache of a given tile....” (*Agarwal*, 28:23-32.) *Agarwal* describes two alternatives of an I/O module writing into memory (the “memory coherence case”) or writing into cache (the “cache coherence case”). (*Agarwal*, 28:11-32). Figure 9C shows the “cache coherence case.”

FIG. 9C illustrates an example ... in which an I/O module 912 writes or reads data to or from a tile’s cache.... The home tile accepts the write transactions from the I/O module 912 just as it would write transactions from a tile. The home tile performs the normal coherence processing, with some differences as noted below ....  
(*Agarwal*, 29:5-28 (emphasis added); *see also supra* [1c], [1d], [1e] (describing *Agarwal* caching agent managing cache reads / writes from tiles and I/O devices); *Agarwal*, 27:39-28:8, 28:23-32 (“An I/O module is also able to ... store data directly into the cache of a given tile...”), Fig. 9C.) *Agarwal*’s multicore processor allocates a shared L3 cache line of the L2/L3 cache to the write data from the I/O device. (*Agarwal*, 29:5-30:11.)

Relevant to claim 6, *Agarwal* discloses that an I/O device sends a request

(i.e., “allocation transaction”) to the shared cache of a home tile through the I/O module (i.e., “the caching agent is to receive an allocation transaction from the IIO module” and “the allocation transaction received in the IIO module from an IO device”). (*Agarwal*, 29:5-30:11.) The directory controller, in turn, allocates a shared cache line to the targeted cache. (*Id.*) *Agarwal* thus discloses claim 6. (Ex-1003 ¶234.)

**7. Claim 7: “The apparatus of claim 6, wherein the caching agent is to directly store data of the allocation transaction into the shared cache memory and to not initiate a memory transaction to a system memory coupled to the multicore processor.”**

*Agarwal* discloses claim 7. (Ex-1003 ¶¶236-247.) In *Agarwal*’s “cache coherence case” described in [6], the I/O write data is written directly into the shared L3 cache of the appropriate home tile (e.g., Tile 2 in Fig. 9C). (*Agarwal*, 28:23-32, 29:5-30:11.) Figure 9C below shows this process:

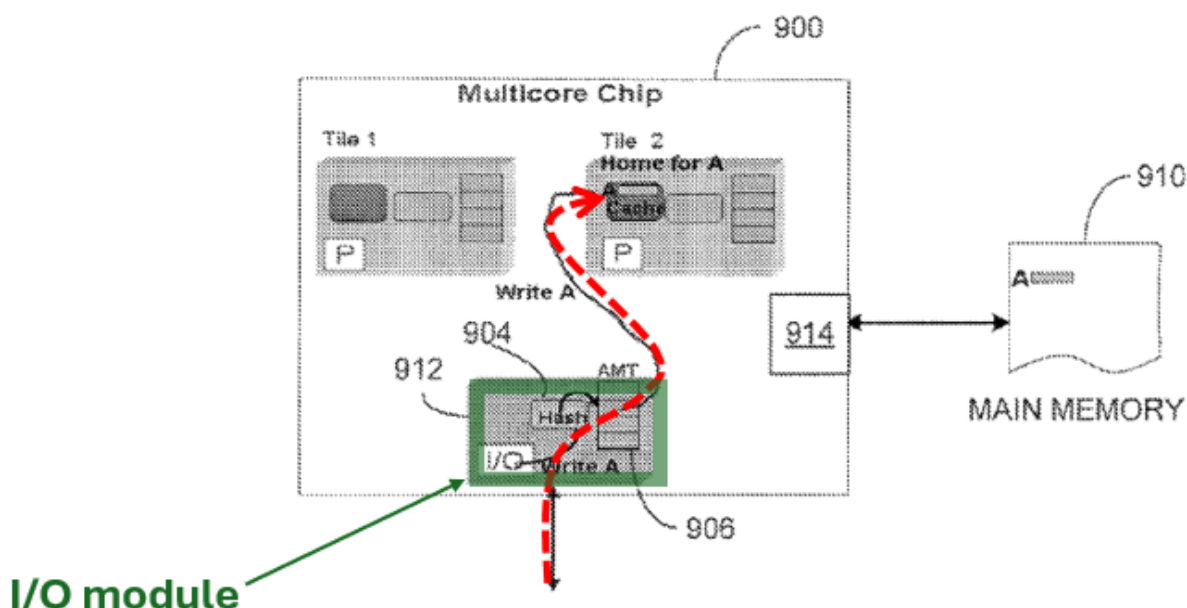


FIG. 9C

(*Agarwal*, Fig. 9C; Ex-1003 ¶¶242-245.) Thus, in the “cache coherence case,” *Agarwal*’s distributed directory controller and directory storage (*i.e.*, “caching agent”) directly store data of the allocation transaction in Tile 2’s shared cache memory. (Ex-1003 ¶246.)

Furthermore, as detailed for claim 6, *Agarwal* details two distinct scenarios: the “memory coherence case” (Fig. 9B) and the “cache coherence case” (Fig. 9C). In *Agarwal*’s description of the “cache coherence case” in Fig. 9C, the directory controller and directory storage do not initiate a memory transaction to a main memory (*i.e.*, “system memory”), unlike the system memory transaction shown in Fig. 9B. *Agarwal*, 29:5-30:11. Describing two such distinct scenarios teaches, or

at a minimum, renders obvious that the cache coherence case does not write into the system memory as is done in the memory coherence case because no transaction to system memory is shown. (Ex-1003 ¶247.)

## 8. Independent Claim 11

### a. 11[pre]: “A method comprising:”

*Agarwal* discloses the claimed method as detailed below. (*Agarwal*, 4:55-64, 7:22-31; Ex-1003 ¶248.)

### b. [11a]: “receiving an allocation transaction in an integrated input/output (IIO) module of a multicore processor from an IO device coupled to the multicore processor, the allocation transaction comprising a request to directly store data of the allocation transaction into a shared cache memory of the multicore processor”

*Agarwal* discloses element [11a]. (Ex-1003 ¶¶249-255.) *Agarwal*’s “cache coherence case” discloses the claimed “receiv[ing] an allocation transaction ...” from an IO device and teaches a request to write directly into the shared cache. (*Supra*, [6][7]; *Agarwal*, 27:39-28:8, 28:23-32, 29:5-30:11, Fig. 9C.)

### c. [11b]: “forwarding the allocation transaction to a caching agent of the multicore processor, the caching agent shared by a plurality of cores of the multicore processor and the IIO module”

*Agarwal* discloses element [11b]. (Ex-1003 ¶¶256-261.) As discussed for claims 6 and 7, *Agarwal* discloses “wherein the caching agent [of the multicore processor] is to receive an allocation transaction from the IIO module.” (*Supra* [6],

[7].) The discussion of claim 6 above explains that (1) *Agarwal*'s I/O module receives a request ("allocation transaction") from an IO device coupled to the multicore processor, and (2) *Agarwal*'s directory controller and directory storage for the home tile (part of the distributed "caching agent") receives that request from the I/O module. (*Supra* claim 6; Ex-1003 ¶¶260, 227-235.) In other words, the I/O module "forwards" the IO device's request ("allocation transaction") to the directory controller and directory storage of the appropriate home tile (i.e., to the "caching agent" portion in the home tile). (*Supra* claims 6, 7) *Agarwal* thus discloses forwarding the allocation transaction to a caching agent of the multicore processor. (Ex-1003 ¶260.)

Claim [11b] also recites: "the caching agent is shared by a plurality of cores of the multicore processor and the I/O module." As discussed for [1d] and [1e], *Agarwal* discloses a single caching agent with distributed portions that maintain cache coherency for the plurality of cores and I/O module thus disclosing this shared caching agent. (*Supra* [1d], [1e]; Ex-1003 ¶261.).

**d. [11c]: "directly storing the data of the allocation transaction into the shared cache memory of the multicore processor."**

As discussed for claims 6 and 7, *Agarwal*'s "cache coherency case" directly stores data into the L3 shared cache. (*Supra*, [6], [7] (Ex-1003 ¶261.).

**9. Claim 12: “The method of claim 11, further comprising accessing the data in the shared cache memory via at least one of the plurality of cores of the multicore processor a number of times, without sending a transaction to a system memory coupled to the multicore processor.”**

*Agarwal* discloses claim 12. (Ex-1003 ¶¶263-274.) Claim 12 essentially describes a cache’s purpose: maintain data locally for access(es) so that the (slower) system memory does not have to be accessed if the data is in the cache. (Ex-1003 ¶¶264-265.) The ’228 patent agrees with this understanding: “With the data now present in the LLC, it can be accessed a number of times for read and write operations ....” (Ex-1001, 3:65-4:4).

Unsurprisingly, *Agarwal* discloses this “conventional” concept:

In response to a memory request, the home tile supplies the data contained in the address *a* (the relevant cache line denoted as *<a>*) to the requesting tile 2 if tile 1 has that cache line in its cache ....

... However, when the home tile has the data in its cache, the request to main memory can be eliminated. ... Furthermore, notice that once the home tile has the data, it can further supply the data to other tiles (e.g., tile 3) without having to go to main memory.

(*Agarwal* 18:1-23; *See also id.*, Fig. 6A, 19:60-67.)

As discussed for claims 7 and 11, in *Agarwal*’s “cache coherence case,” I/O write data is written directly into the shared cache lines of the L3 cache without initiating a main (“system”) memory update transaction. (*Supra* claims 7, 11)

Thus, tiles can access the data a number of times without system memory transactions through either: (1) a cache coherency write followed by a cache read or (2) multiple cache reads after data has been written to the cache.

*Agarwal's* "Simple Read Flow" explains that data is returned on a cache hit without any memory transaction:

#### 4.1 Simple Read Flow

On a read, a tile first checks its private level 1 and level 2 caches. The index function used is the PLIF. If hit, data is returned. If miss, based on PA<11:6>, the tile sends a read request over the TDN to the home tile. .... The home tile checks its cache for the requested block, and assuming hit, returns the data to the requesting tile.

(*Agarwal*, 25:28-45). In contrast, *Agarwal* "handles cache misses ...by sending them to a home memory controller." (*Agarwal*, 8:14-17). "Exemplary steps for maintaining cache coherency" include "[o]n a cache miss (e.g., for a shared data read) on a tile...[t]he directory controller will then get the cache line data ... from main memory if there is no L3 cache on-chip." (*Agarwal*, 14:60-15:28).

*Agarwal's* teaching that data, once present in the cache, can be accessed without accessing memory teaches, or renders obvious, this claim. (Ex-1003 ¶272-274.)

**10. Claim 13: “The method of claim 12, further comprising sending data of the cache line to the system memory from the shared cache memory if the cache line is evicted from the shared cache memory, when the cache line includes modified data.”**

*Agarwal* discloses claim 13. A POSITA would understand this claim to be, at a minimum, obvious because it describes typical cache operation in which evicted cache lines with modified data are written to memory to preserve the modifications. (Ex-1003 ¶¶275-285.)

*Agarwal*’s L3 cache uses “a write-back policy for writing copies of cache lines stored as shared L3 cache lines.” (*Agarwal*, 24:61-64). For such a write-back policy, a cache line is marked as “dirty” when the data in the line is modified (e.g., by a write request). (*Agarwal*, 15:34-43, 20:39-42.) If a “dirty” cache line is evicted, that data is written back to *Agarwal*’s main memory (i.e., “system memory”), and any copies of that cache line are invalidated in the other tiles. (*Agarwal*, 26:23-30; 27:1-5; 21:55-56 (explaining “the block is either written back to memory (if dirty) or dropped (if clean)”). (Ex-1003 ¶¶277-282.)

Writing dirty evicted cache lines back to memory was (and remains) a well-understood and fundamental part of using a write-back policy. (Ex-1003 ¶¶283-284.) *Agarwal* thus discloses this claim. (Ex-1003 ¶285.)

**11. Claim 14: “The method of claim 12, further comprising directly storing the data of the allocation transaction into the shared cache memory without communicating with the system memory.”**

*Agarwal* discloses claim 14. (Ex-1003 ¶¶286-288.) As discussed above for claims 7, 11 and 12, *Agarwal* discloses a “cache coherence case,” wherein I/O write data is written directly into the shared L3 cache without initiating a main memory (i.e., “system memory”) update transaction. (*Supra* claims 7, 11, 12; *Agarwal*, 23:26-41, 28:23-32, 29:5-30:11, Fig. 9C.) Thus, *Agarwal* discloses claim 14. (Ex-1003 ¶¶288.)

**B. Ground 2A: *Sinharoy* Renders Obvious Claims 1, 4, 6, 7 and 11-14.**

**1. Overview of *Sinharoy***

*Sinharoy* is an IBM paper describing IBM’s “POWER7” processor chip. (*Sinharoy*, 1.) In IPR2023-01344, PO did not challenge that *Sinharoy* constitutes prior art to the ’228 patent family.

*Sinharoy* is prior art to the ’228 patent under at least pre-AIA 35 U.S.C. §102(a). Ex-1023 (Greenhill Declaration). First, *Sinharoy* has been available on the IEEE Xplore database since April 29, 2011. (Ex-1023, ¶¶29-35[.] “IEEE is a well-known, reputable compiler and publisher of scientific and technical publications.” *Ericsson Inc. v. Intellectual Ventures I LLC*, IPR2014- 00527, Paper 41, at 11 (P.T.A.B. May 18, 2015). This alone is sufficient evidence of *Sinharoy*’s

public availability.

Second, *Sinharoy* itself shows that it was published in the “May/June 2011” edition of the IBM Journal of Research and Development (ISSN: 0018-8646), a well-known bimonthly serial publication that ran from 1957 to 2020. (Ex-1023 ¶¶19-25; Ex-1016, 1; Ex-1014, 1 (LoC Record at 1); Ex-1015, 1 (MARC Record at 1 (field 008, “001120d19572020nyubr pso 0 a0eng c,” indicating a serial publication (“d”) beginning in “1957” and ending in “2020” (emphasis added))).)

Before the ’228 patent’s priority date, at least two documents cite the May/June 2011 edition of the IBM Journal in which *Sinharoy* appears, confirming its public availability and accessibility. (Ex-1023, ¶¶26-28; Ex-1013 (cited reference no. [3]); Ex-1012 (cited reference no. [6]).) *Sinharoy* was thus publicly available before the ’228 patent’s priority date.

*Sinharoy* is an analogous art to the ’228 patent. (Ex-1003 ¶¶103-105.) Like the ’228 patent, *Sinharoy* discloses a processor chip (such as the POWER7 chip below). (*Sinharoy*, 1.)

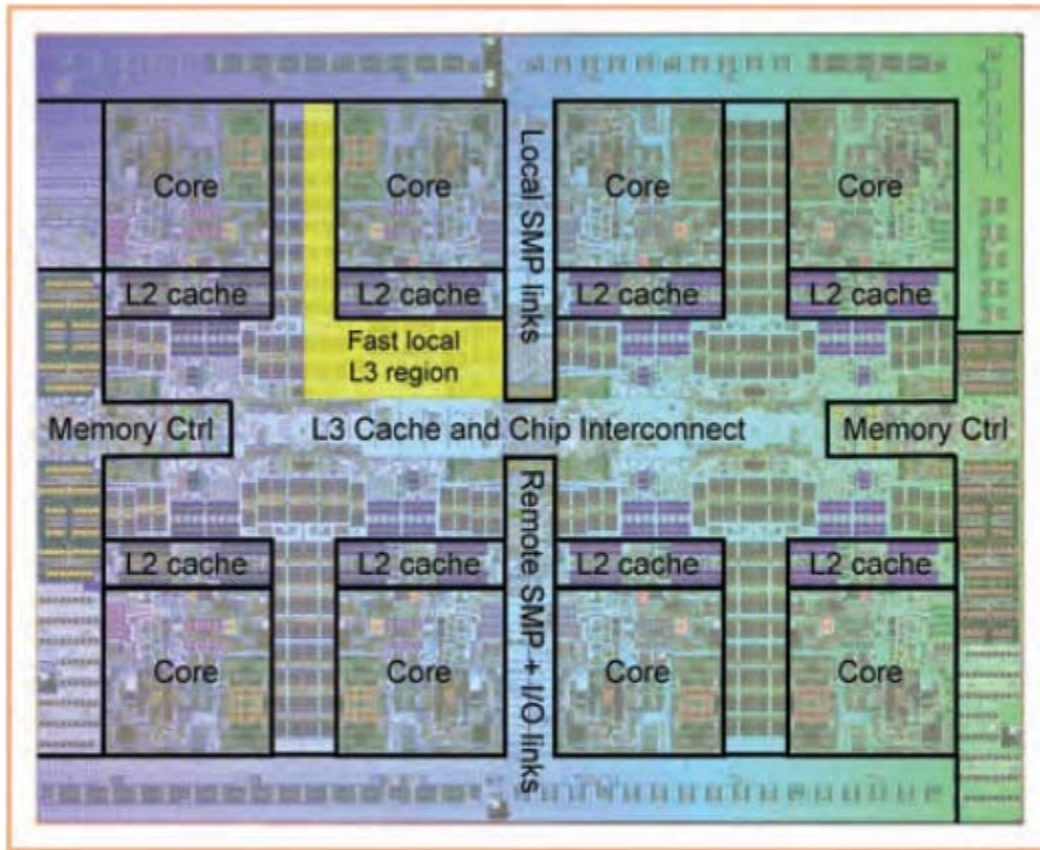


Figure 1

Die photo of the IBM POWER7 chip.

(*Sinharoy*, Fig. 1.)

*Sinharoy* describes “an eight-core processor” with “three levels of on-chip cache.” (*Sinharoy*, 1.) *Sinharoy*’s on-chip L3 cache “provides several advantages such as improved L3 latency” and “improved L3 bandwidth per core ....”

(*Sinharoy*, 2.) Additionally, *Sinharoy* teaches “[a]n I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory.” (*Sinharoy*, 17.)

**2. Independent Claim 1**

**a. 1[pre]: “An apparatus comprising:”**

*Sinharoy* discloses a “multicore server processor” (i.e., an apparatus).

(*Sinharoy*, 1, 23-25, Figs. 11-13; Ex-1003 ¶¶292-293)

**b. [1a]: “a multicore processor including a plurality of cores”**

*Sinharoy*’s “multicore server processor” (i.e., “multicore processor”) includes “eight high-performance cores” (i.e., “a plurality of cores”). (*Sinharoy*, 1-2, Fig. 1; Ex-1003 ¶¶294-298.)

**c. [1b]: “a shared cache memory”**

*Sinharoy*’s multicore server processor includes a “32-MB-shared-L3 cache” (i.e., “shared cache memory”). (*Sinharoy*, 2; Ex-1003 ¶¶299-303; *see also Sinharoy*, 17-21, 22, Tbl. 1.)

**d. [1c]: “an integrated input/output (IIO) module ...”**

*Sinharoy* discloses element [1c]. (Ex-1003 ¶¶304-321.) The ’228 patent’s “IIO module may act as an interface to one or more off-chip peripheral or IO devices ... include[ing] storage devices, network interfaces, graphics cards.” (Ex-1001, 2:22-26.) *Sinharoy* teaches two types of such IIO modules.

First, *Sinharoy* discloses “two integrated I/O controllers,” each being an “integrated input/output (IIO) module,” as claimed. (*Sinharoy*, 22; Ex-1003 ¶¶305-307.) These integrated I/O controllers connect (i.e., “interface between”) one or

more input/output devices (e.g., PCI device) and the multicore processor via an I/O hub. (*Sinharoy*, 17, 22, Fig. 14 (illustrating a PCI adapter connected to I/O hub); Ex-1003 ¶307.) *Sinharoy*’s “I/O controllers also support cache injection, which enables DMA write traffic to be written directly into a cache, instead of to memory.” (*Sinharoy*, 22.) In IPR2023-01344, PO did not challenge that *Sinharoy*’s I/O controllers disclosed the claimed IIO module. Ex-1020.

Second, *Sinharoy*’s “SMP link” disclosed also as a “cluster interconnect” or “multichip SMP interconnect” corresponds to the ‘228 patent’s “network interface” which is a form of an IIO module (Ex-1001, 2:22-26). (*Sinharoy*, 22-23), (Ex-1003 ¶¶308-320.) This interconnect uses SMP links and GX I/O links to network POWER7 chips. *Sinharoy*’s Fig. 14 depicts these I/O modules interfacing to I/O devices: an I/O controller via a “GX I/O bus” to an I/O hub that connects to a “PCI adapter” and the SMP links that interface to a network interface (e.g. “fabric interface”).

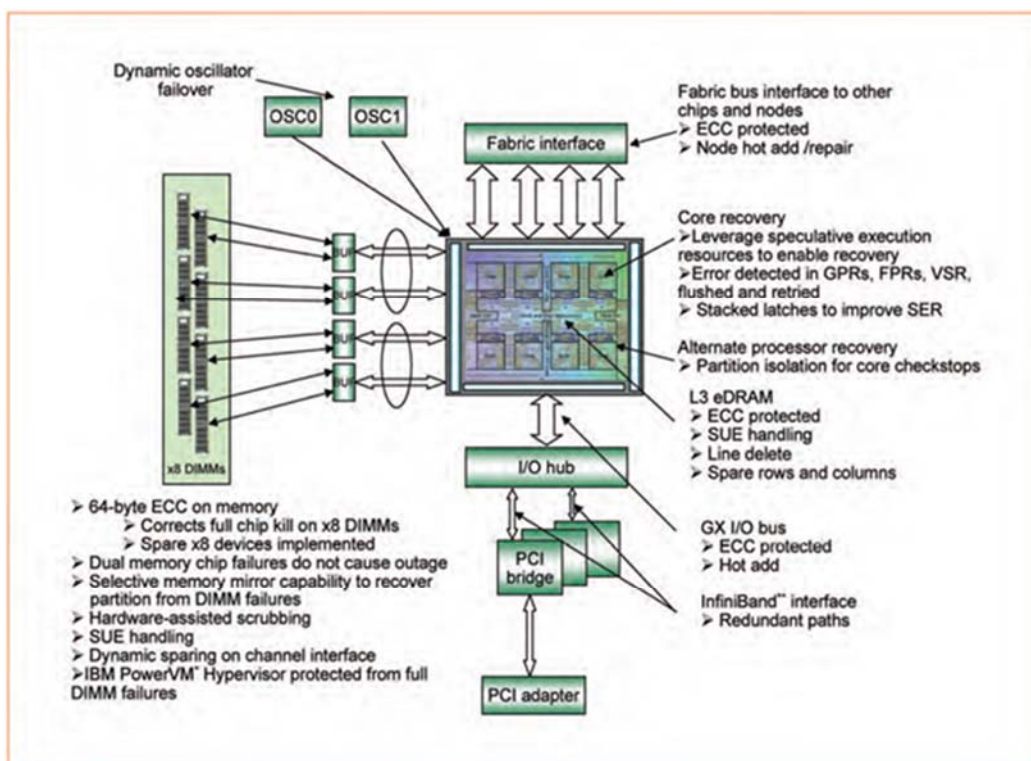


Figure 14

IBM POWER7 reliability and availability features.

(Sinharoy, Fig. 14.) (Ex-1003 ¶¶308-310.) Fig. 13 shows the cluster network interconnects:

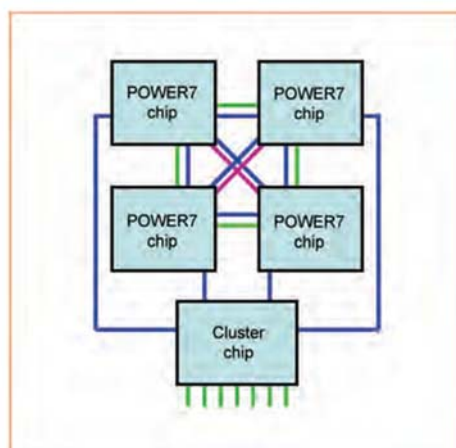


Figure 13

Cluster system first-level coherent nodal topology.

(*Sinharoy*, Fig. 13) (Ex-1003 ¶¶312-316.). *Sinharoy*’s network interface (“cluster interconnect” logic) “couples the POWER7 coherence and data interconnect seamlessly into a cluster interconnect.” (*Sinharoy*, 25.) “The cluster interconnect chip has an on- and off-chip coherence and data interconnect that interoperates with the POWER7 chip and provides an on-board memory coherence directory (MCD) .... The MCD improves I/O subsystem throughput by qualifying and vectoring DMA reads and writes to a single target node ....” (*Sinharoy*, 25.) (Ex-1003 ¶317.)

“Each cluster interconnect chip incorporates up to three 16PCI Express\*\* Gen2 I/O subsystem interfaces” as shown in Figure 14 and Table 4 (*Sinharoy*, 25.)

**Table 4** IBM POWER7 standard system and cluster system interfaces.

| <i>Logical interface</i>                                   | <i>Standard system</i>                     | <i>Cluster system</i>                                   |
|------------------------------------------------------------|--------------------------------------------|---------------------------------------------------------|
| POWER7 to POWER7 nodal interconnect                        | Use three of four POWER7 on-node SMP links | Use three of four POWER7 on-node SMP links              |
| POWER7 to POWER7 nodal interconnect 2× bandwidth expansion | n/a                                        | Use two POWER7 off-node SMP links and both GX I/O links |
| POWER7 to cluster chip nodal interconnect                  | n/a                                        | Use fourth POWER7 on-node SMP link                      |
| System interconnect                                        | Use one or two POWER7 off-node SMP links   | Use seventh cluster chip off-node SMP/cluster links     |
| I/O subsystem attachment                                   | Use POWER7 GX I/O links                    | Use cluster chip PCI interfaces                         |
| Memory channels                                            | Use eight POWER7 channels                  | Use eight POWER7 channels                               |

(*Sinharoy*, Table 4.) *Sinharoy*’s network interface facilitates “coherent I/O sharing among the eight flat 32-way SMPs.” *Sinharoy*, Col 1, p 1:25. (Ex-1003 ¶¶318-

320.).

Thus, Sinharoy discloses this element. (Ex-1003 ¶321.)

**e. [1d]: “a caching agent...”**

*Sinharoy* discloses element [1d]. (Ex-1003 ¶¶322-352.) *Sinharoy*’s distributed “coherence fabric” comprises eight L3 cache controllers (i.e., “caching agent” as claimed). (*Sinharoy*, 19; *see also Sinharoy*, 17-19.) (Ex-1003 ¶323.)

*Sinharoy*’s multicore processor includes three cache layers. The L1 and L2 caches are private to each core, and the 32-MB L3 cache is shared by all cores. (*Sinharoy*, 19, Table 1.) This L3 cache comprises eight distributed 4-MB local L3 regions – with each region incorporated into a core to provide faster access for that corresponding core. (*Sinharoy*, 19, Fig. 1; Ex-1003 ¶¶324-334.)

**“Coherence operations for the plurality of cores.”** Each of *Sinharoy*’s eight L2 and L3 cache pairs (one each per core) includes a controller to help maintain cache coherency across all eight cores. (*Sinharoy*, 18-19.) For example, each L2 and L3 controller supports the “13-cache-state protocol” defined in Table 2. (*Sinharoy*, 17, Tbl. 2.) These cache states support “coherent operations.” (*Sinharoy*, 17.)

To maintain cache coherency, each L2 and L3 controller includes a pool of read/claim (RC) machines and write machines that perform cache coherency operations, such as fetches, read-write operations, and core prefetch operations.

(*Sinharoy*, 18-19; Ex-1003 ¶¶324-340.) *Sinharoy* explains, for example, that “[s]torage accesses that miss both the L2 cache and the 4-MB local L3 region are broadcast to the coherence fabric” where they are snooped by “other L2 caches, possibly other L3 caches, and by the seven remote 4-MB L3 regions that comprise the remaining 28 MB of the on-chip L3 cache.” (*Sinharoy*, 19.) This enables checking whether another core contains modified data for the same address in the cache (e.g., a core’s L2 cache or the local L3 cache region). (*Sinharoy*, 19.)

**“Coherence operations ... for the IIO module.”** *Sinharoy*’s cache architecture also supports “coherent operations” like “cache injection” that allows an I/O device to perform coherency operations (e.g., data reads or writes) targeted directly at the cache. (*Sinharoy*, 17, 22; Ex-1003 ¶¶342-346.) For example, when an I/O device sends a write command to the CPU caches via the integrated I/O controller, “[i]f a given core’s L2 cache or local L3 region owns a copy of the targeted cache line ... the data will be installed into the local L3 region.” (*Sinharoy*, 17.)

In addition, *Sinharoy* discloses the SMP links used for the “multichip SMP interconnect” discussed in [1c] as an IIO module (e.g., “network interface”). “As shown in Figure 1, the POWER7 chip contains ... two I/O controllers, and a multichip SMP interconnect capable of scaling up to 32 POWER7 chips into a single large SMP system.” (*Sinharoy*, 22.) “Each chip has four 10-B/s on-node

*SMP links* associated with the vertical spine that emanates from the center of the chip toward the top edge. In a standard commercial system, *three of these four SMP links are connected to each of the three other chips that comprise a four-chip node*. In this manner, the *four chips are fully connected*.” (Sinharoy, 23-24.)

Sinharoy’s multicore processor maintains cache coherency for the cores and the IIO modules (integrated I/O controllers and SMP interconnect) through an on-chip coherence interconnect that includes “coherence request, snoop, response, and notification interfaces and several data interfaces.” (Sinharoy, 22; Ex-1003 ¶¶347-351.)

**f. [1e]: “the caching agent ....”**

Sinharoy discloses [1e]. (Ex-1003 ¶¶353-407.) Sinharoy teaches the same architecture that the ‘228 patent characterizes as a “single caching agent... including ... distributed portions.”

In the ‘228 patent, the single caching agent: (1) handles cache requests and coherency for CPUs and I/O devices by implementing “various coherency logic” and (2) corresponds to an embodiment in the ‘228 patent’s, Figs. 2 and 6. (Ex-1001, 2:58-3:3, 6:22-29, Figs. 2, 6). The ‘228 patent’s Fig. 6 shows the distributed caching agents.

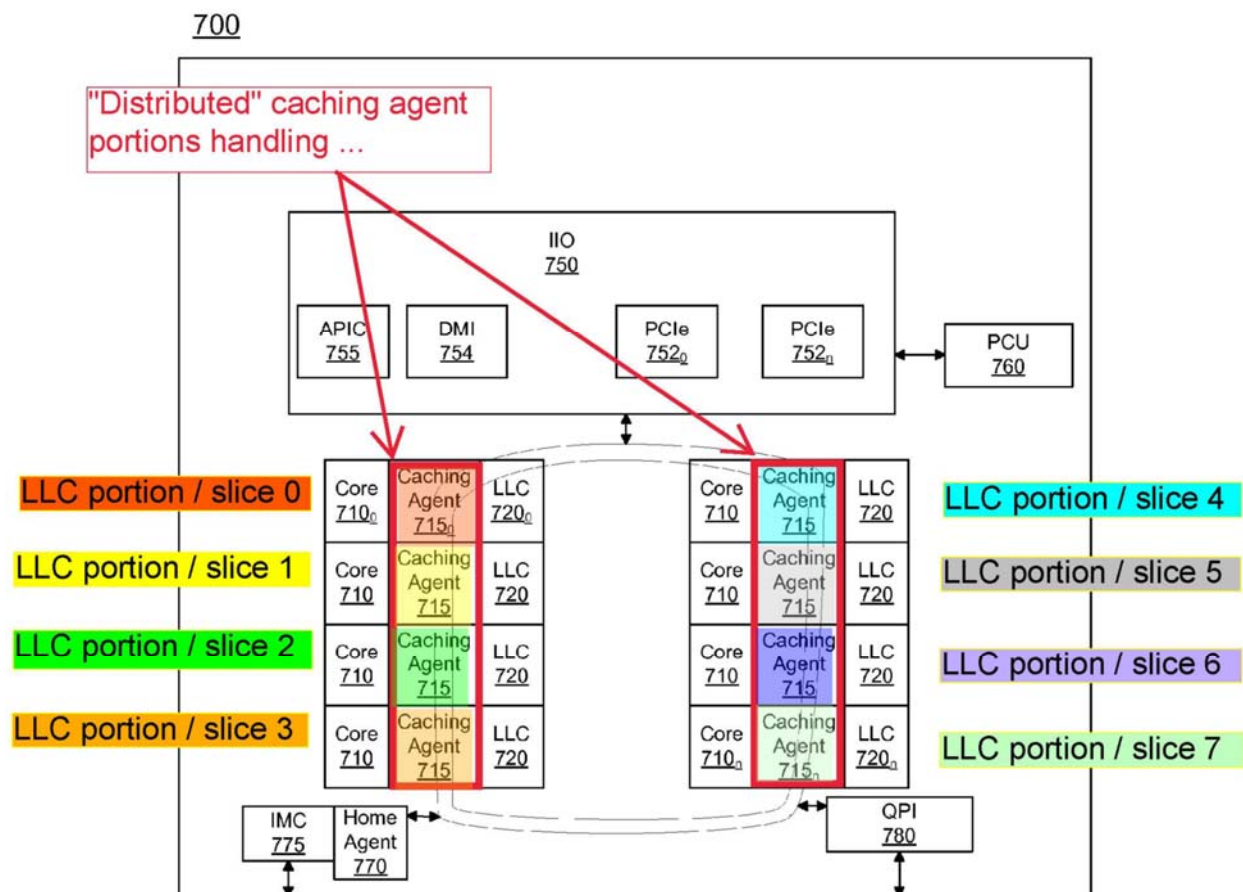
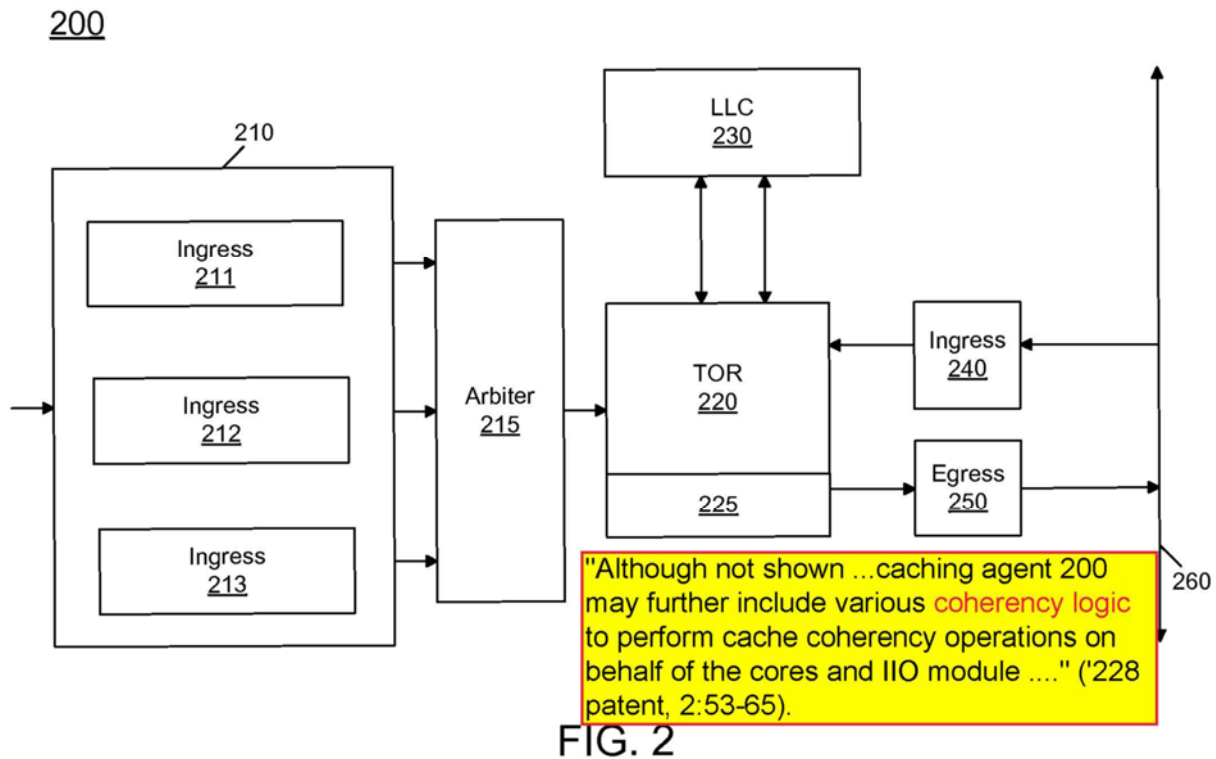


FIG. 6

(Ex-1001, Fig. 6; 6:20-39.) (Ex-1003 ¶¶354-359.) The '228 patent's Fig. 2 depicts a "block diagram" of one portion of the distributed caching agent in Fig. 6. (Ex-1001, 2:36-3:3.)



(Ex-1001, Fig. 2.) As annotated above, the Fig. 2 caching agent includes “coherency logic” to manage cache coherency. (Ex-1001, 2:53-65.) Thus, each “distributed portion” caching agent in Fig. 6 above includes the Fig. 2 components above, including “coherency logic” needed for cache coherency recited in the claims. (Ex-1003 ¶¶360-363.)

*Sinharoy*’s architecture is the same as the ’228 patent’s architecture above. *Sinharoy* teaches that each core includes: (1) a core, (2) a portion of the shared L3 cache; and (3) an L3 cache “controller.” (*Sinharoy*, 17-19.) Each controller corresponds to one “caching agent 715” portion in the ’228 patent’s Fig. 6 above shown placed between each of the cores 710 and shared LLC cache regions 720.

(Ex-1003 ¶¶364-367.)

Similarly, the architecture of the '228 patent's caching agent portions in Fig. 2 matches *Sinharoy*'s controller architecture. In the '228 patent, each core's caching agent includes various "ports" for ingress / egress to communicate with other cores and I/O ports, as well as "coherency logic" to manage coherency. (Ex-1001, 2:34-65.) *Sinharoy*'s "controller" also has various "ports" for ingress / egress to communicate with other cores and I/O ports and coherency logic *Sinharoy*, 19. (Ex-1003 ¶¶368-372.)

*Sinharoy*'s controller (and the '228 patent's Fig. 2) have "coherency logic" that manages coherency among the cores' L3 regions to form a "coherency fabric." *Sinharoy*, 19-21. *Sinharoy* discloses a more integrated set of distributed caching agent portions than '228 patent discloses. For example, *Sinharoy*'s "lateral cast-out" operation moves a cache line from one L3 portion to another L3 portion – but still preserves the cache line within the accessible L3 cache. *Sinharoy*, 19-20. By using the other distributed portions of the L3 caches as "victim caches" (for lateral cast-out between L3 distributed portions), *Sinharoy*'s system manages the overall cache space as a single cache managed by a single caching agent controlling the cache lines across all of the distributed portions. (*Sinharoy*, 20-21, Ex-1003 ¶¶373-376.).

*Sinharoy*'s cache controllers (caching agent portions) act collectively.

As illustrated in Figure 10, the capacity, data flow logic, and control flow logic of a 4-MB local L3 region are shared between the L2 cache to which the local L3 region is coupled, and the other seven 4-MB L3 regions on the chip.

(*Sinharoy*, p. 19, Fig. 10). This “sharing” of “data flow logic” and “control flow logic” corresponds to the “coherency logic” that the ‘228 patent describes for its Fig. 2 as distributed across different physical cores in the ‘228 patent’s Fig. 6. (Ex-1003 ¶¶377-378.)

*Sinharoy* also discloses that the multicore processor utilizes a “nonblocking-broadcast-based coherence-transport mechanism” providing distributed cache coherency management based on the “13-cache-state protocol” shown in Table 2. (*Sinharoy*, 17, 23, Tbl. 2.) This “coherence-transport” mechanism uses the L3 dispatch ports and the SMP interconnect to manage the cache. (*Sinharoy*, 19.) *Sinharoy* discloses that when a core’s memory access misses both its L2 cache and its 4-MB local L3 cache region, snoop messages are broadcasted throughout this distributed coherence fabric to the other cache regions, and the snooped cache regions send snoop responses and, if needed, forward the shared data to the requester. (*Sinharoy*, 19.) This is a further form of the “coherency logic” that the ‘228 patent references for the “single caching agent.” (Ex-1001, 2:58-64; Ex-1003 ¶379.)

*Sinharoy*’s limited snooping between L3 cache portions facilitates

distributed coherency operations and reduces the overall snooping—just like the '228 patent. For example, the L3 snooping allows for the cache to appear as a single entity to all CPUs and the I/O controllers for which a request that is routed to the distributed cache agent logic which also responds to that request and ensures coherency based on that request. (*Sihharoy*, 23) (“The on-chip coherence interconnect routes ... coherence requests through the horizontal trunk, inward toward the even/odd arbitration logic located at the center of the chip. .... Once granted, the requests are broadcast within the chip on the even/odd snoop buses outward toward the left and right edges of the chip. .... Requests that have arrived from other chips are managed by the even/odd arbitration logic and broadcast to the snoop buses....Coherence responses from the snoopers are routed inward along the horizontal trunk toward the even/odd coherence decision logic located at the center of the chip...” ). (Ex-1003 ¶380.)

Thus, *Sinharoy* operates just like the '228 patent in reducing snooping, for example, by the I/O component. (Ex-1001, 1:19-30, Ex-1003 ¶¶381-383.) As shown below, the '228 patent's Fig. 6 shows that the cache is accessed as a single entity by the IIO unit which needs to make only single access to the cache – not an access to each individual core's caching unit. The distributed caching agent's “coherency logic” manages the cache for that access internally. However, to the IIO module, the cache appears as a single entity and the IIO module does not need

a separate I/O agent for each core – which is the problem identified in the '228 patent's Background. (Ex-1001, 12-30, Ex-1003 ¶384.)

'228 Patent,  
Fig. 6

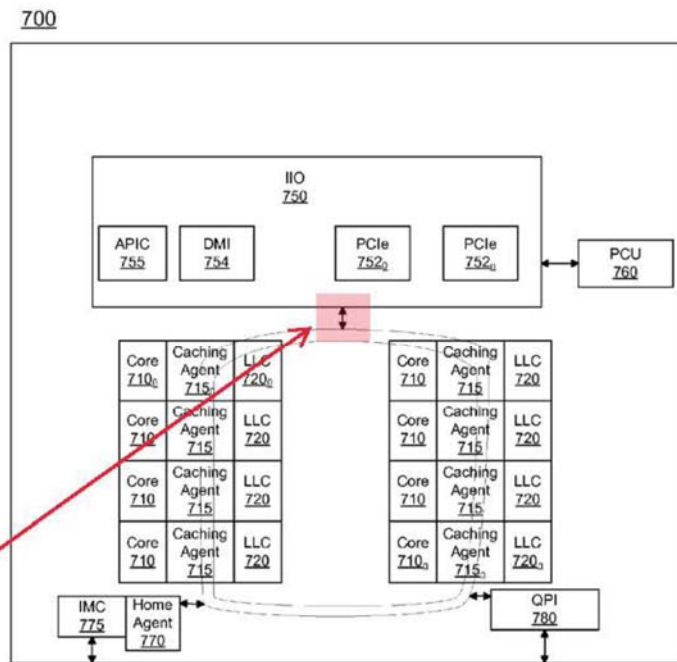
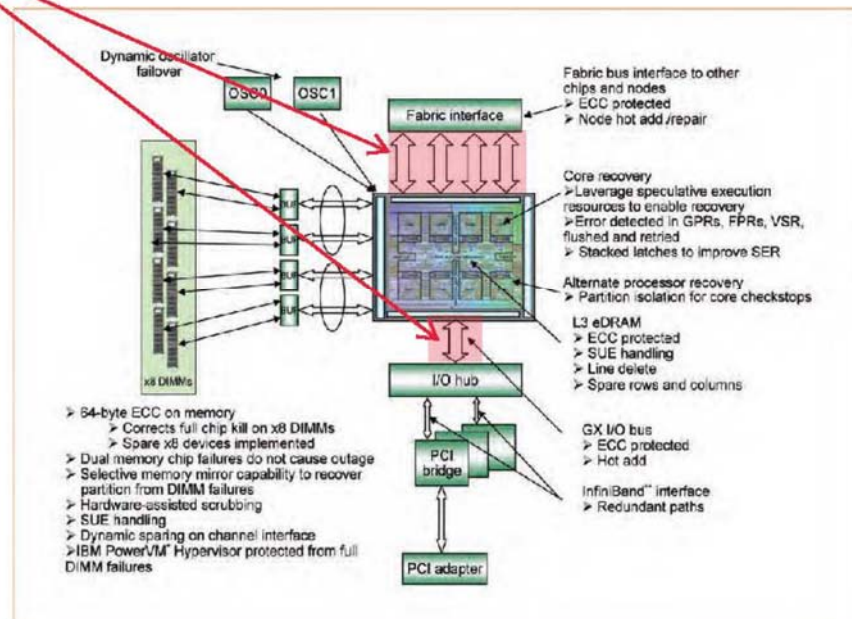


FIG. 6

I/O interface  
to cache is a  
single access

Sinharoy,  
Fig. 14



(Ex-1001, Fig. 6; Sinharoy, Fig. 14). As shown above, Sinharoy operates the same

way that the '228 patent operates. *Sinharoy*'s I/O units (the two I/O controllers at bottom and the SMP interconnect at top) treat the cache as a single integrated unit – needing to only submit a single cache access (read/write) to the cache coherency logic. *Sinharoy*'s distributed L3 caching agent logic manages that access. For example, *Sinharoy*'s L3 cache may need to perform lateral cast-out operations or other coherency logic operations, but those operations are abstracted from the I/O unit. As with the '228 patent, *Sinharoy*'s I/O devices do not need separate I/O agents to access cache associated with each core. In this manner, *Sinharoy* operates in the same manner as the '228 patent identifies for a single caching agent. (Ex-1001, 1:12-30; Ex-1003 ¶384).

In IPR2023-01344, PO made two arguments against *Sinharoy* regarding this same claim term (“single caching agent ... including ... distributed portions”). *E.g.*, IPR2023-01344, paper 10 (05/28/2024) at 15-22 (Ex-1020), paper 20 at 4-11 (10/22/2024) (Ex-1021). First, PO argued that *Sinharoy*'s reference to “snooping” among L3 caches prevents *Sinharoy* from disclosing the '228 patents “single caching agent.” *E.g.*, Ex-1021, pp. 5-9.

The presence of “snooping,” however, cannot exclude *Sinharoy*'s architecture from disclosing the “single caching agent” because the '228 patent expressly acknowledges that snooping still occurs with the distributed cache agents / controllers. The '228 patent states that its “distributed” architecture “*reduces* the

amount of snoop traffic....” Ex-1001, 3:1-8; 6:17-18. “Reduces” is not “eliminates.” Clearly, the ’228 patent contemplates “snooping” even with its distributed architecture. Such snooping is shown throughout the ’228 patent. (*E.g.*, Ex-1001, Fig. 4 (“receiving snoop request ... grant ownership to snoop initiator”), 4:15-2 (“snoop request”), 4:38, 4:51-54); (Ex-1003 ¶¶385-391). Claim 8 (dependent on claim 1) recites a “snoop request.” Thus, PO’s arguments that “snooping” excludes *Sinharoy* from meeting this claim element contradict the ’228 patent.

As detailed above, *Sinharoy*’s lateral L3 cache reduces snooping (by I/O devices) that is identified as the problem that the ’228 patent seeks to solve. Moreover, some amount of L3 snooping (or other coordination) is necessary among the distributed L3 cache lines for coherency. (Ex-1003 ¶392.) The ’228 patent abstracts this coordination as “various cache coherency logic to perform cache coherency operations on behalf of the cores and IIO module.” (Ex-1001, 2:58-65). However, the ’228 patent does not exclude L3 cache cross-snooping as part of the coherency logic. (Ex-1003 ¶¶392-394.)

PO’s second argument in IPR2023-01344 is “individual cache controllers are *not* part of the same physical unit.” (Ex-1021, p. 5.) This is irrelevant. As Fig. 6 shows above, this is true in the ’228 patent as well. The eight separate “caching agents” in the ’228 patent’s Fig. 6 (elements 715<sub>0</sub>-715<sub>7</sub>) “are *not* part of the same

physical unit” (in PO’s words) but are still taught in the ’228 patent as being a single caching agent. (Ex-1001, 6:25-28.) (Ex-1003 ¶¶395-396.)

PO’s argument devolves into an assertion that because the ’228 patent recited the word “single” to characterize the collection of individual caching agents in Fig. 6, the ’228 patent is immune from challenge. However, *Sinharoy*’s architecture implements the ’228 patent’s “various coherency logic to perform cache coherency operations on behalf of the cores and IIO module” to allow the “cores... and an IIO module [to] maintain coherency without the need for additional caching agents.” (Ex-1001, 2:58-64). Thus, *Sinharoy* discloses, or renders obvious, characterizing its architecture as a single caching agent with distributed portions as claimed. (Ex-1003 ¶398.)

Regarding the claim limitation of being “associated with a corresponding one of the plurality of cores,” as discussed for [1d], *Sinharoy*’s cores each have a 4-MB L3 cache region managed by a “single centralized controller” *for that L3 cache slice* which collectively form a single distributed “coherence fabric” (i.e., “a single caching agent”) spanning all L3 slices. (*Supra* [1d]; *Sinharoy*, 17-19, Fig. 1.)

*Sinharoy*’s eight processor cores each comprises, and supplies, an associated 4-MB L3 cache region for collective usage across all the cores (like the ’228 patent Fig. 6 above). *E.g.*, *Sinharoy*, 17 (“32-MN L3 cache comprised of the 4-MN local L3 regions from the eight cores”); 16 (“a 32-KB L1 D-cache, a 256-KB L2 cache,

and a 4-MB local L3 region in the POWER7 processor”),<sup>23</sup> (“Each core’s associated L2 cache and local L3 region...”); <sup>22</sup> (“POWER7 chip contains eight cores, each with an associated L2 cache and local L3 region...”).

*Sinharoy*’s Fig. 1 shows each core contains this L3 cache:

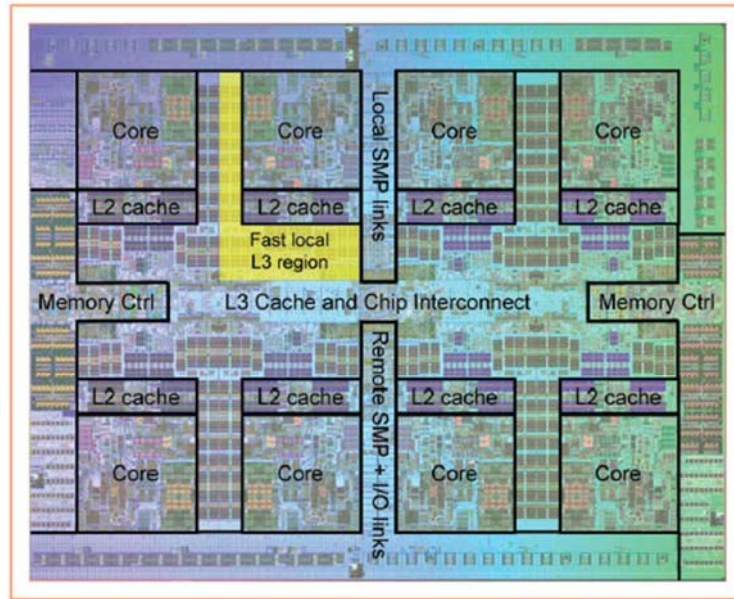


Figure 1

Die photo of the IBM POWER7 chip.

(*Sinharoy*, Fig. 1.). To be clear, it is not a proximity between *Sinharoy*’s core and L3 region that establishes the “associated with” element or mere ability to access the shared cache but rather *Sinharoy*’s teachings that each core includes / incorporates the L3 cache local region.

Accordingly, each local 4-MB L3 cache region is “associated with” a corresponding core. (Ex-1003 ¶¶399-405.)

At a minimum, given *Sinharoy*’s overlap in architecture and functionality

with that described in the '228 patent as being a “single caching agent,” *Sinharoy* renders obvious the claimed single caching agent. The shared L3 cache management is distributed across the different cores with “data flow logic and control logic” being “shared” between the L3 controller associated with each cache. This distributed sharing of the logic renders obvious the “single caching agent.” (Ex-1003 ¶406.).

*Sinharoy* thus discloses, including rendering obvious, this element. (Ex-1003 ¶407.)

**3. Claim 4: “...wherein the shared cache memory comprises a last level cache (LLC) ....”**

*Sinharoy* discloses claim 4. (Ex-1003 ¶¶408-416.) The claimed “last level cache (LLC)” refers to the highest level cache shared by all cores. (Ex-1003 ¶409.) *Sinharoy*’s share L3 cache is “a last level cache (LLC),” as claimed as detailed in [1d], [1e]. (*Sinharoy*, 17, 19, Fig. 1, Tbl. 1.) (Ex-1003 ¶¶410-415.)

**4. Claim 6: “...wherein the caching agent is to receive an allocation transaction from the IIO module, ....”**

*Sinharoy* discloses claim 6. (Ex-1003 ¶¶417-429.) In the '228 patent, an allocation transaction may “lodge data directly into the [‘last level cache’ (LLC)].” (Ex-1001, 3:22-25; *see also* Ex-1001, 3:47-51). *Sinharoy* discloses this feature. (Ex-1003 ¶¶418-420.)

*Sinharoy*’s coherence fabric (i.e., “the caching agent”) receives an allocation

transaction from an I/O device via an integrated I/O controller or the SMP interconnect. In particular, *Sinharoy*'s "cache injection" operation allows an I/O device to send a "write operation" (i.e., an "allocation transaction") writing data directly into the local L3 cache region using the cache controller "write machines." (*Sinharoy*, 17, 19, 22.) "An I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory." (*Sinharoy*, 17.) (Ex-1003 ¶¶421-426)

As discussed for [1c], integrated I/O controllers connect (i.e., "interface between") one or more input/output devices (e.g., PCI device) and the multicore processor via an I/O hub. (*Supra* [1c].) Similarly, the SMP links (see [1c], [1d]) "facilitate coherent I/O sharing among the eight flat 32-way SMP" nodes to provide cache coherence. *Sinharoy*, 25. Accordingly, *Sinharoy* meets this claim. (Ex-1003 ¶¶427-428.)

**5. Claim 7: "... wherein the caching agent is to directly store data ...."**

*Sinharoy* discloses claim 7. (Ex-1003 ¶¶430-433.) *Sinharoy*'s L2 and L3 caches support a "13-cache-state protocol" enabling "new coherent operations" such as "cache injection." (*Sinharoy*, 17.) During a cache injection, "[a]n I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory." (*Id.*; see also *id.*, 22.)

*Sinharoy*'s disclosure of "instead of to memory" confirms the cache injection does not initiate a memory transaction. If the cache injection initiated a memory transaction, *Sinharoy* would have taught that the injection would "target the operation to the cache *in addition to* memory." *Sinharoy* used "instead of" and a POSITA would understand such language to disclose, including rendering obvious, that there is not a memory transaction associated with the cache injection. (Ex-1003 ¶432.)

**6. Independent Claim 11**

**a. 11[pre]: "A method comprising:"**

*Sinharoy* discloses the claimed method as detailed below. (Ex-1003 ¶434.)

**b. [11a]: "receiving an allocation transaction ..."**

*Sinharoy* discloses [11a]. (Ex-1003 ¶¶435-440.) *Sinharoy* discloses "receiv[ing] an allocation transaction ...coupled to the multicore processor" as detailed above. (*Supra* claim 6.) (Ex-1003 ¶439.)

Furthermore, as discussed for claims 6 and 7, *Sinharoy*'s cache injection write operation (i.e., an "allocation transaction") writes data directly into the local L3 cache region as a result of a request from an IO device. (*Supra* claims 6-7; *Sinharoy*, 17, 22; Ex-1003 ¶440.)

**c. [11b]: "forwarding the allocation transaction ..."**

*Sinharoy* discloses [11b]. (Ex-1003 ¶¶441-447.) As discussed for claim 6,

*Sinharoy* discloses “wherein the caching agent [of the multicore processor] is to receive an allocation transaction from the IIO module.” (*Supra* claim 6.)

*Sinharoy*’s coherence fabric (i.e., “caching agent”) receives a cache injection write operation (i.e., “allocation transaction”), which has been forwarded by an integrated I/O controller (i.e., “IIO module”) or SMP interconnect to the coherence fabric so it can be received. (*Supra* claims 6, 7; *Sinharoy*, 17, 22; Ex-1003 ¶¶444-445.) *Sinharoy* thus discloses the claimed forwarding. (Ex-1003 ¶446.)

As discussed for [1d] and [1e], *Sinharoy* discloses a “shared” single caching agent with distributed portions that maintain cache coherency for the plurality of cores and IIO module. (*Supra* [1d], [1e]; Ex-1003 ¶444.)

**d. [11c]: “directly storing ....”**

*Sinharoy* discloses [11b]. (Ex-1003 ¶¶448-449.) As discussed for claim 7, *Sinharoy*’s “cache injection” directly stores data of the allocation transaction into the shared cache memory, thus meeting this element. (*Supra* claim 7.)

**7. Claim 12: “... accessing the data in the shared cache memory ... without sending a transaction to a system memory ....”**

*Sinharoy* discloses claim 12. (Ex-1003 ¶¶450-454.) As discussed for claims 7 and 11, *Sinharoy*’s “cache injection” operation allows an I/O device to write data directly into L3 cache instead of to memory. (*Supra* claims 7, 11; *Sinharoy*, 17.) A POSITA would understand that *Sinharoy* supports a well-known write-back

procedure for caches and therefore discloses access (including both reading and writing) data as many times as needed until a write-back needed (i.e. line has a “[dirty] state (i.e., “modified data,” as claimed in claim 13)) for writing copies of cache lines stored as shared L3 cache lines causes the cache line to be written to main memory. (Ex-1003 ¶451.)

Once data is in *Sinharoy*’s shared cache, that data can be accessed without accessing the system memory – that is a cache’s purpose. (Ex-1003 ¶452.) The ‘228 patent does not describe anything new related to this claim and merely states “[w]ith the data now present in the LLC, it can be accessed a number of times for read and write operations ....” (Ex-1001, 3:65-67.)

In *Sinharoy*, a core needing to access a particular data location checks first in its L2 cache, then in its local L3 cache, and then in the “remaining 28 MB of the on-chip L3 cache” – all without accessing system memory. *Sinharoy*, 19, Fig. 10. Thus, *Sinharoy* meets this claim. (Ex-1003 ¶453.)

A core needing access the I/O data resident in *Sinharoy*’s shared L3 cache accesses that data without sending a transaction to system memory for as many accesses as the data remains in the cache. (Ex-1003 ¶454.)

**8. Claim 13: “... sending data of the cache line to the system memory ....”**

*Sinharoy* discloses claim 13. (Ex-1003 ¶¶455-457.) In *Sinharoy*, “[w]hen a

line is evicted from an L3 region, if the evicted line is second class and has a ‘dirty state, it is written back to memory.” *Sinharoy*, at 21. *Sinharoy*’s Table 2 shows “modified” data can be “cast out” to memory. *Sinharoy*’s Figure 10 shows the cast-out (evicted / modified) lines being written to system memory:

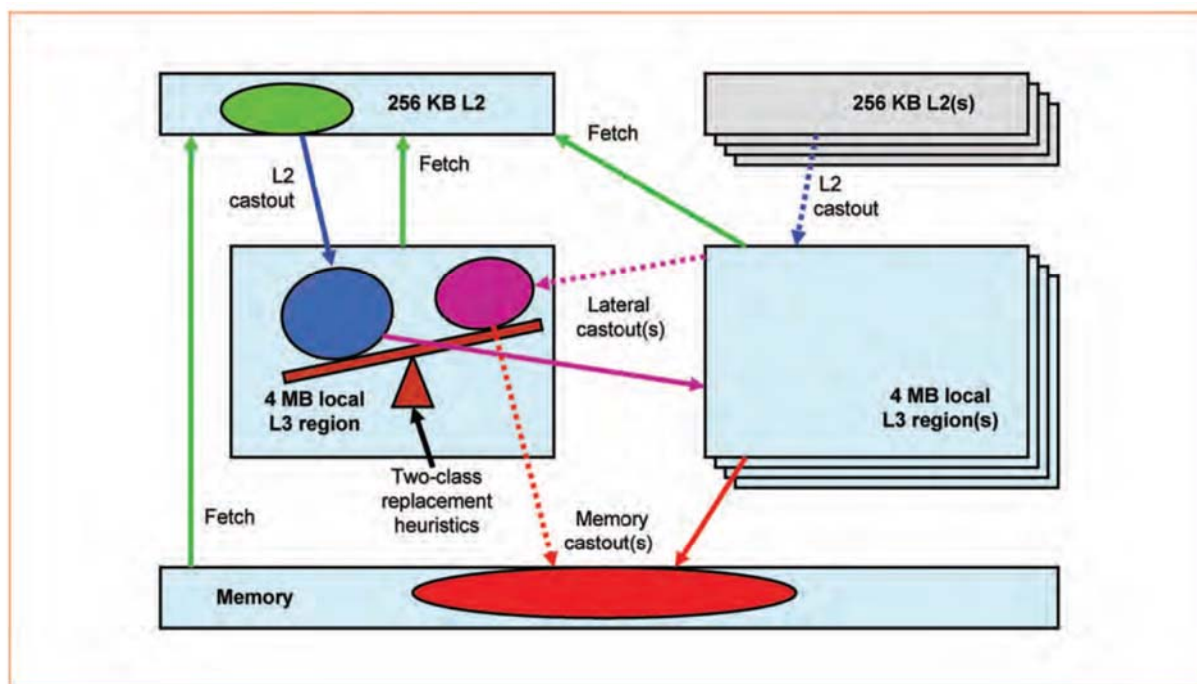


**Figure 10**

L3 region behavior.

*Sinharoy*, Fig. 10. *Sinharoy*’s “cast out scenarios are described in detail” on pages 19-21. *Sinharoy*, 19-21. Those scenarios include writing the evicted cast-out line to memory if it is dirty (modified). *Sinharoy*, 19-21. (Ex-1003 ¶457.)

### **9. Claim 14: “... directly storing ....”**

*Sinharoy* discloses claim 13 for the same reasons as cited above for claims 11 and 12. (*Supra* [11], [12], Ex-1003 ¶¶458-459.)

**C. Ground 2B: Claims 2, 3 and 16-18 Are Obvious over the *Sinharoy-Blumrich* Combination**

If *Sinharoy* by itself does not render claims 2 and 3 obvious as detailed below, *Sinharoy* combined with *Blumrich* renders these claims obvious.<sup>7</sup>

**1. Overview of *Blumrich***

*Blumrich* is an analogous art to the '228 patent. (Ex-1003 ¶¶107-108.) Like the '228 patent, *Blumrich* discloses a “multiprocessor system,” (*Blumrich*, Abstract), and discloses methods to “maintain cache coherence” (*Blumrich*, ¶[0124]).

*Blumrich*’s “L2 caches serve as the point of coherence for all processors.” (*Blumrich*, ¶[0119].) As part of “[c]oherence tracking,” *Blumrich* uses a “request queue 302,” which “buffers incoming read and write requests.” (*Blumrich*, ¶[0128].) In one embodiment, this queue “is 16 entries deep,” although *Blumrich* notes “other request buffers might have more or less entries.” (*Id.*) Using *Blumrich*’s queue, “[t]he addresses of incoming requests are matched against all pending requests to determine ordering restrictions.” (*Id.*)

---

<sup>7</sup> Claims 2 and 3 are substantively identical to claims 3 and 4 of the '228 patent’s child patent, 9,575,895. In IPR2023-01344, PO did not dispute that *Sinharoy* taught these claims.

## 2. Rationale to Combine

A POSITA would have been motivated to implement *Sinharoy*'s "coherence fabric" using *Blumrich*'s request queues. (Ex-1003 ¶¶461-467.) Like *Sinharoy*, *Blumrich*'s disclosure originated at IBM. (*Blumrich*, Cover.)

Each *Sinharoy* controller provides a "pool of operational resources." (*Sinharoy*, 19.) All incoming requests to an L3 controller are arbitrated and processed using this pool. (Ex-1003 ¶462.) *Sinharoy*'s pool includes "eight read machines," "24 lightweight RC [read/claim] machines," and "ten write machines." (*Sinharoy*, 19.) Read machines manage read hits; RC machines manage store misses and prefetches; and write machines manage data writes. (*Sinharoy*, 19; Ex-1003 ¶462.) Write machines also perform core prefetch operations, L2 cast-out operations, and I/O cache injection operations. (*Sinharoy*, 17-20.)

Because *Sinharoy*'s pool of resources has limited entries (e.g., capable of handling eight outstanding reads, 24 outstanding store misses, and ten outstanding data writes), a POSITA would have understood the controller would beneficially use a storage element (e.g., queue or buffer) to track outstanding requests and avoid overflowing the limited pool of resources. (Ex-1003 ¶463.) A POSITA would have understood that either a queue or a buffer would have been an obvious solution for keeping track of outstanding requests. (Ex-1003 ¶464.) For example, *Sinharoy* suggests using queues to keep track of requests within the core. (See

*Sinharoy*, 9-10 (explaining the core includes buffers/queues to track outstanding requests, such as a 32-entry store-reorder-queue (“SRQ”) and a 32-entry load-reorder-queue (“LRQ”)); Ex-1003 ¶464.)

A POSITA would have considered similar, known options for *Sinharoy*’s coherence fabric (“caching agent”) to track outstanding requests, including *Blumrich*’s disclosure of a tracking queue. (Ex-1003 ¶465.) First, a POSITA would have understood that a queue like the one *Blumrich* discloses would be beneficially used by *Sinharoy* for tracking requests. (Ex-1003 ¶465.) For example, a queue enables ordering requests, allowing incoming requests’ addresses to be matched against outstanding requests. (*Blumrich*, 0128) When a write request is followed by a read request for the same address, the two requests must be executed in order to insure the read accesses the correct data. (Ex-1003 ¶466.) A POSITA thus would have understood caching agents (e.g., *Sinharoy*’s coherence fabric) require a queue to store pending requests to perform address-checking for an incoming request. (*Blumrich*, 0128-0130) (Ex-1003 ¶466.)

Additionally, a POSITA would have reasonably expected to succeed in implementing *Sinharoy*’s coherence fabric by including a request queue, as *Blumrich* teaches. (Ex-1003 ¶467.) The combination would have involved nothing more than combining prior art elements arranged according to known methods to yield predictable results. (Ex-1003 ¶467.)

**3. Claim 2: “The apparatus of claim 1, wherein the caching agent includes a table of requests to store pending requests for the plurality of cores and the IIO module.”**

Sinharoy combined with Blumrich discloses Claim 2. (Ex-1003 ¶¶468-475.)

*Sinharoy*’s caching agent (cache controllers) include RC and write machines as detailed above. Because RC machines “aggressively reorder storage operations” and manage “core fetches, read-modify-write operations, on behalf of core stores, and core prefetch operations,” it would be obvious that they include temporary storage to store incoming (pending) requests, such as requests to read or to write for the cores. (*Sinharoy*, 1:18; Ex-1003 ¶469.) The temporary storage in the RC machines render obvious “tables of requests,” and the incoming requests to read or to write are “pending requests.”

Furthermore, write machines in the same “caching agent” perform write operations including “cache-injections.” (*Sinharoy*, 19.) Integrated I/O controllers (i.e., the IIO modules) also “support cache injection.” (*Sinharoy*, 22.) The write operations include incoming (“pending”) requests to write for the IIO module. (Ex-1003 ¶470.) When performing cache injection operations, it would be obvious that the write machines include temporary storages (i.e., “table of request”) for incoming requests to write. (*Id.*) Thus, the caching agent with RC and write machines include “table[s] of requests to store pending requests for the plurality of cores and the IIO module.” (*Id.*)

*Sinharoy* also discloses this feature in its pool of operational resources and temporary storages discussed above. (Ex-1003 ¶471.) Each L3 cache controller provides a “pool of operational resources.” (*Sinharoy*, 19.) Because the pool of resources has limited entries (e.g., capable of handling eight outstanding reads, 24 outstanding store misses, and ten outstanding data writes), the controller would beneficially use a storage element (e.g., queue or buffer) to track outstanding requests and avoid overflowing the limited pool of resources. (Ex-1003 ¶471.)

Furthermore, *Sinharoy* discloses using queues for cache interactions as shown in Fig. 3 with reference to the “Load reorder queue,” “Store reorder queue,” and “Store data queue,” used to interface L1 and L2 caches. (*Sinharoy*, 18-19, Fig. 3; Ex-1003 ¶472)

Therefore, *Sinharoy* renders obvious “the caching agent” (e.g. the RC machines) “includes a table of requests to store pending requests for the plurality of cores and the IIO module” (e.g., aggressively reorders storage operations via, for example, a “Store reorder queue” and a “Store data queue”). (Ex-1003 ¶473).

If *Sinharoy* alone does not render claim 2 obvious, the *Sinharoy-Blumrich* combination renders claim 2 obvious. (Ex-1003 ¶474.) Although *Sinharoy* does not specifically state which type of storage element the controllers use for “pool” or temporary storage above, a POSITA would have understood that either a queue or a buffer would have been an obvious solution for keeping track of outstanding

requests. (Ex-1003 ¶474) For example, *Blumrich* teaches to use a “request queue 302” to “buffer[] incoming read and write requests.” (*Blumrich*, ¶[0128]; *see also id.* at Fig. 3.) *Blumrich*’s request queue allows “[t]he addresses of incoming requests [to be] matched against all pending requests to determine ordering restrictions.” (*Id.*)

Thus, a POSITA at least would have found a queue like the one *Blumrich* discloses to be an obvious implementation of *Sinharoy* and would have reasonably expected to succeed in implementing such a solution. (Ex-1003 ¶475; *Supra* V.C.2.)

**4. Claim 3: “The apparatus of claim 2, wherein the table of requests includes at least one dedicated entry to store a posted transaction.”**

*Sinharoy-Blumrich* discloses Claim 3. (Ex-1003 ¶¶476-479.) As stated above, the RC and write machines within *Sinharoy*’s caching agent include temporary storages (i.e., tables of request) to store incoming read / write requests. When write machines perform cache injection operations, they write into L3 cache. (*Sinharoy*, 17, 19.) Such a write transaction is a “posted transaction” because (as identified in the ‘228 patent) *Sinharoy* does not disclose any “completion or confirmation message regarding the” injection to the I/O device that is the injection source. (Ex-1001, 5:3-7) (Ex-1003 ¶477.)

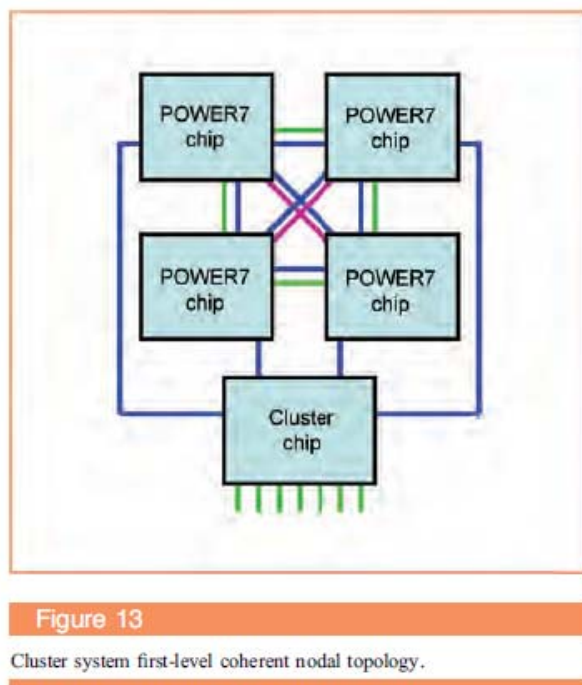
Because write machines within *Sinharoy*’s caching agent have temporary

storages, it would be obvious to include dedicated queues or resources (i.e., dedicated entry) to store such a write transaction (i.e., posted transaction) when performing the cache injection. (Ex-1003 ¶478.) Thus, if *Sinharoy* alone does not render claim 3 obvious, the *Sinharoy-Blumrich* combination renders claim 3 obvious. (Ex-1003 ¶479.)

## 5. Independent Claim 16

### a. 16[pre]: “A system comprising:”

*Sinharoy* discloses the claimed system as detailed below. (Ex-1003 ¶¶482-484.) *Sinharoy*’s “up to 32 POWER7 chips may be connected to form a single 256-way SMP system.” (*Sinharoy*, 23; *see also id.*, 23-25, Figs. 11-13.) Figure 13 shows a four multicore processor system:



(*Sinharoy*, Fig. 11.)

**b. 16[a]: “a first multicore processor including a first plurality of cores”**

**c. 16[b]: “a first shared cache memory”**

*Sinharoy* discloses 16[a] and 16[b] for the reasons cited for [1a], [1b].

(*Supra* [1a], [1b], Ex-1003 ¶¶483-484.)

**d. 16[c]: “a first integrated input/output (IIO) module to interface between the first multicore processor and a first plurality of IO devices coupled to the first multicore processor”**

*Sinharoy* discloses 16[c]. (Ex-1003 ¶¶485-486.) *Sinharoy*’s server system includes up to 32 multicore server processors. (*Supra* [16pre].) Each server processor includes an integrated I/O controller and SMP links (i.e., “an integrated input/output (IIO) module”) to interface between the multicore processor and a plurality of IO devices coupled to the multicore processor. (*Supra* [1c].) (Ex-1003 ¶486.)

**e. 16[d]: “a first caching agent to perform cache coherency operations for the first plurality of cores and the first IIO module,”**

*Sinharoy* discloses 16[d]. (Ex-1003 ¶¶487-488.) As discussed for [1d] and [1e], each of *Sinharoy*’s server processors includes a coherence fabric (i.e., “caching agent”) to perform cache coherency operations for the plurality of cores and the IIO module thus disclosing this element. (*Supra* [1d], [1e]; Ex-1003 ¶488.)

**f. 16[e]: “wherein the first caching agent includes a table of requests having a plurality of entries each to store a pending request for one of the first plurality of cores or the first IIO module, including at least one dedicated entry to store a pending request corresponding to a posted transaction”**

*Sinharoy* discloses 16[e]. (Ex-1003 ¶¶489-491.) As discussed for [16pre], *Sinharoy*’s server system includes up to 32 multicore server processors. (*Supra* [16pre].) As discussed for claim 2, for each of these multicore processors, the coherence fabric (i.e., “caching agent”) includes a table of requests to store pending requests for the plurality of cores and the IIO module. (*Supra* claim 2.) As discussed for claim 3, the table of requests includes at least one dedicated entry to store write requests (corresponding to a posted transaction) for a cache injection. (*Supra* claim 3.) (Ex-1003 ¶491.)

**g. 16[f]: “a second multicore processor including a second plurality of cores”**

**h. 16[g]: “a second shared cache memory”**

**i. 16[h]: “a second IIO module to interface between the second multicore processor and a second plurality of IO devices coupled to the second multicore processor”**

**j. 16[i]: “a second caching agent to perform cache coherency operations for the second plurality of cores and the second IIO module”**

*Sinharoy* discloses 16[f], 16[g], 16[h], and 16[i]. (Ex-1003 ¶¶492-499.) As discussed for [16pre], *Sinharoy*’s server system includes up to 32 multicore server

processors. (*Supra* [16pre].) This would include at least a first and a second multicore processor. (Ex-1003 ¶493.) As discussed for [1a], each server processor includes a plurality of cores. (*Supra* [1a].) (Ex-1003 ¶493.)

As discussed for [1b], each server processor, including a second multicore server processor, includes a “32-MB-shared-L3 cache” (i.e., “shared cache memory”) within processor. (*Supra* [1b].) (Ex-1003 ¶495.)

As discussed for [1c], each server processor, including a second multicore server processor, includes an integrated I/O controller and/or SMP links (i.e., “an integrated input/output (IIO) module”) to interface between the second multicore processor and a plurality of IO devices coupled to the second multicore processor. (*Supra* [1c].) (Ex-1003 ¶497.)

As discussed for [1d], each server processor, including a second multicore server processor, includes a coherence fabric (i.e., “caching agent”) to perform cache coherency operations for the plurality of cores and the IIO module. (*Supra* [1d].) (Ex-1003 ¶499.)

**k. 16[j]: “a peripheral controller coupled to at least one of the first and second multicore processors.”**

*Sinharoy* discloses 16[j]. (Ex-1003 ¶¶501-506.) As discussed for [16pre], *Sinharoy* discloses a server system with up to 32 multicore server processors. (*Supra* [16pre].) In one example, as part of that server system, *Sinharoy* further

discloses a “cluster interconnect chip,” which incorporates up to three PCI express interfaces and provides a connection between the least two multicore server processors and any PCI devices that connect to the PCI express interfaces:

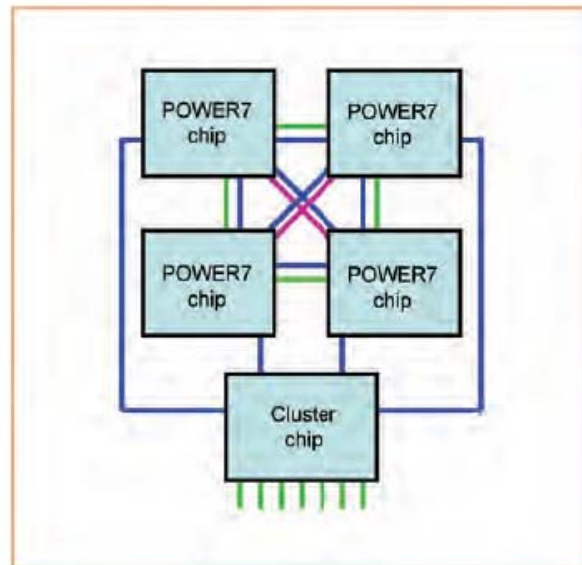


**Figure 13**

Cluster system first-level coherent nodal topology.

(*Sinharoy*, 25, Fig. 13; Ex-1003 ¶¶502-504.) This cluster interconnect chip is a “peripheral controller,” as claimed, because it provides a connection between at least one multicore processor and peripheral components. (Ex-1003 ¶¶505.) As shown above, the cluster interconnect chip is “coupled to at least one of the first and second multicore processors.” (Ex-1003 ¶¶506.)

**6. Claim 17: “The system of claim 16, wherein the first caching agent is to receive an allocation transaction from the first IIO module, the allocation transaction comprising a request to directly store data of the allocation transaction into the first shared cache memory, and directly store the data of the allocation transaction into the first shared cache memory.”**

*Sinharoy* discloses claim 17 for the reasons discussed in claims 6 and 11 above. (Ex-1003 ¶¶507-508.) *Sinharoy* discloses “wherein the [first] caching agent [of the first multicore processor] is to receive an allocation transaction from the IIO module.” (*Supra* claim 6.) *Sinharoy* discloses that “the allocation transaction compris[es] a request to directly store data of the allocation transaction into a [first] shared cache memory.” (*Supra* [11a].) *Sinharoy* discloses “directly storing the data of the allocation transaction into the [first] shared cache memory of the [first] multicore processor.” (*Supra* [11c].)

**7. Claim 18: “The system of claim 17, wherein the data in the shared cache memory is to be accessed via at least one of the first plurality of cores a number of times, without sending a transaction to a system memory coupled to the first multicore processor.”**

*Sinharoy* discloses claim 18 for the reasons discussed for claim 12 above. (*Supra* claim 12, Ex-1003 ¶¶509.)

**D. Ground 2C: Claim 5 Is Obvious over the *Sinharoy-Sistla* Combination**

**1. Overview of *Sistla***

*Sistla* is an analogous art to the '228 patent. (Ex-1003 ¶¶109-110.) *Sistla*

discloses a “processor” comprising “a memory,” “a plurality of processor cores in communication with the cache memory,” and “a scalability agent unit that operates as an interface between an on-die interconnect and both multiple processor cores and memory.” (*Sistla*, Abstract.)

## 2. Rationale to Combine

One of ordinary skill would have been motivated to modify *Sinharoy*’s interconnect topology in view of *Sistla*’s disclosure of a ring interconnect topology. (Ex-1003 ¶¶511-517.)

*Sinharoy*’s L3 local regions connect through an “interconnect that physically occupies a horizontal equatorial trunk that runs across the center of the POWER7 chip.” (*Sinharoy*, Fig. 1, 22-23.) *Sinharoy*’s interconnect uses “arbitration logic located at the center of the chip.” (*Sinharoy*, 23.) Each L3 cache controller receives snoops through two sets (odd, even) of coherent interconnect routes from the cores or other instances of the distributed L3 cache controllers. (*Sinharoy*, 19, 22-23, Fig. 1; Ex-1003 ¶512.)

A POSITA would have found it obvious to use a ring interconnect in *Sinharoy*’s architecture. (Ex-1003 ¶513.) *Sistla* teaches a multicore processor with a distributed cache like *Sinharoy*. (*Sistla* ¶[0001], Figs. 3-4.) *Sistla*’s multicore processor comprises “a plurality of core-cache clusters 310<sub>1</sub>-310<sub>N</sub> in communication with each other over an on-die interconnect 320.” (*Sistla*,

¶¶[0030]-[0031].) Each core-cache cluster includes a scalability agent (a caching agent's distributed portion) that maintains memory coherency. (*Sistla* ¶[0018].) *Sistla* teaches that “on-die interconnect 320 is configured as a ring interconnect.” (*Sistla* ¶[0030].) *Sistla* thus demonstrates that the relevant features in *Sinharoy*'s architecture are compatible with a ring interconnect, and a POSITA would have had a reasonable expectation of success in modifying *Sinharoy*'s interconnect topology to include a ring interconnect. (Ex-1003 ¶513.)

Using a ring interconnect would be obvious because a ring interconnect would have provided a high-performance, scalable interconnect topology that would have been expected improve *Sinharoy*'s bus-style interconnect architecture with a centralized arbitration logic. (Ex-1003 ¶¶514-515) A ring interconnect provides a simple, fast, and efficient interconnection between cores and caches in a multicore processor with higher bandwidth. (*Marty*, 24, 45; *Chrysos* ¶[0018] *Blake*, Tbl. 2).

A POSITA also would have understood that “[r]ings offer simple, distributed arbitration and access methods” (*Marty*, 48) which had advantages over a bus interconnect, like *Sinharoy*'s. (Ex-1003 ¶515; *Marty*, 46.) A ring provides other advantages over *Sinharoy*'s interconnect including scalability and reduced complexity. (Ex-1003 ¶¶515-516) (*Blake*, Tbl. 2.)

*Sistla*'s teachings would have been a simple substitution of one element for

another, which would have been compatible with the design of *Sinharoy*'s architecture. (Ex-1003 ¶517.) And because *Sistla*'s architecture is analogous to *Sinharoy*'s architecture, a POSITA would have reasonably expected of success. (*Compare Sinharoy*, Fig. 1, with *Sistla*, Figs. 3-4.) The combination reflects known prior art elements arranged according to known methods to yield predictable results. (Ex-1003 ¶517.)

**3. Claim 5: "... coupled via a ring interconnect."**

The *Sinharoy-Sistla* combination renders claim 5 obvious. (Ex-1003 ¶¶518-519.) The *Sinharoy-Sistla* result would have been that *Sinharoy*'s 4-MB L3 regions including their controllers ("plurality of distributed portions of the caching agent") would be coupled via a ring interconnect. (Ex-1003 ¶519.)

**E. Ground 2D: Claim 19 Is Obvious over the *Sinharoy-Blumrich-Sistla* Combination**

**1. Claim 19: "The system of claim 16, wherein the first caching agent is a single caching agent for the first multicore processor and includes a plurality of distributed portions each associated with a corresponding one of the first plurality of cores, the first shared cache memory comprises a last level cache (LLC) including a plurality of distributed portions each associated with a corresponding one of the first plurality of cores, and wherein the plurality of distributed portions of the first caching agent are coupled via a ring interconnect."**

The *Sinharoy-Blumrich-Sistla* combination renders claim 19 obvious. (Ex-1003 ¶¶520-521.). Claim 19 is a combination of claim 16, element [1e] and claims

4 and 5.

*Sinharoy* discloses [1e] and claim 4 as per Ground 2A. *Sinharoy-Blumrich* discloses claim 16 as per Ground 2B. *Sinharoy-Sistla* renders obvious claim 5 as per ground 2C. This combination merely adds the well-known ring interconnect from *Sistla* to the *Sinharoy-Blumrich* combination. The motivations to combine *Sistla* with *Sinharoy* are detailed in Ground 2C. *Blumrich*'s disclosure is separate and does not impact either these motivations to combine or reasonable expectation of success. (Ex-1003 ¶511.)

#### **VI. The Petition Is Proper Under 35 U.S.C. §325(d)**

The analysis prescribed by *Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6 at 8-11 (P.T.A.B. Feb. 13, 2020) (precedential) favors institution.

The references in this Petition were not previously presented to the Office during prosecution of the '228 patent. Ex-1002; '228 patent. The references are not cumulative of the prosecution prior art. *See Advanced Bionics* at 8.

#### **VII. The Petition Is Proper Under *Fintiv***

Under *Fintiv*, the Board takes a holistic view of whether efficiency, fairness, and the merits support the exercise of authority to deny institution in view of an earlier trial date in the parallel proceeding. *Apple Inc. v. Fintiv, Inc.*, IPR2020-

00019, Paper 11 at 6 (PTAB March 20, 2020) (precedential). *Fintiv* presents six factors to guide this analysis.

As to Factor 1, regarding a stay of the related proceeding, no stay has yet been requested, so this factor is neutral.

As to Factor 2, regarding the time to trial in the related proceeding, trial is scheduled for January 2026, consistent with the district's median time-to-trial of 21.6 months. Ex-1022. In this proceeding, a final written decision would be anticipated by May 2026. In light of the Board's holistic analysis balancing all *Fintiv* factors, the trial date of the related litigation is not determinative but weighed in concert with other factors. *See Netnut Ltd., v. Bright Data Ltd.*, IPR2021-01492, Paper 12, at 9-16 (PTAB March 21, 2022) (instituting review with trial date six months before final written decision deadline; *Coolit*, Paper 10, at 11, 14 (same by five months). The USPTO's Guidance indicates that "when other relevant factors weigh against exercising discretion to deny institution or are neutral, the proximity to trial should not alone outweigh all of those other factors." *See* June 21, 2022, Interim Procedure for Discretionary Denials in AIA Post-Grant Proceedings with Parallel District Court Litigation ("Guidance").

Factor 3 concerns the investment in the related proceeding and favors institution because Petitioner acted diligently in filing this Petition. Patent Owner served infringement contentions asserting eight patents approximately three

months ago, on September 4, 2024. Petitioners filed this Petition as quickly as possible thereafter. *Fintiv*, IPR2020-00019, Paper 11, at 11; *Coolit Sys., Inc. v. Asetek Danmark A/S*, IPR2021-01195, Paper 10 at 11, 14 (P.T.A.B. Dec. 28, 2021). Moreover, almost no parallel investment has occurred. Fact discovery opened October 9, 2025, and the claim construction hearing will not take place until July 31, 2025. Therefore, the district court’s investment by institution – expected in late April 2025—will be nominal. Ex-1022.

Factor 4 concerns the overlap between issues and, in this case, favors institution or, at worst, is neutral. Although the prior art asserted here is also asserted in district court, this Petition additionally challenges claims 3 and 16-19, which are not involved in the litigation. Accordingly, this proceeding addresses additional issues that will not be resolved in district court, favoring institution.

Factor 5 considers whether the parties here and in the related litigation are the same and favors institution. While Petitioner MediaTek Inc. is a party to the litigation, Petitioner Arm is not a party to the litigation and has separate interests that would not necessarily be vindicated by the district court proceedings.

Factor 6 consider other circumstances, such as the merits of the Petition. Any consideration of *Fintiv* Factor 6 heavily favors institution because Petitioner presents a compelling unpatentability challenge. Guidance at 4.

The merits of this Petition are strong because the prior art herein teaches the very features the Applicant argued to be lacking during prosecution. Moreover, the prior art herein matches the architecture and functionality of the allegedly novel “single caching agent” with “distributed portions.”

## **VIII. Mandatory Notices**

### **A. Real Party-in-Interest**

Petitioners are the real parties-in-interest. No other parties have control over the present Petition, and no other parties funded the present Petition.

### **B. Related Matters**

Patent Owner has asserted the '228 patent against MediaTek Inc. in *Daedalus Prime LLC v. MediaTek Inc.*, No. 2:24-cv-00235-JRG (E.D. Tex.) (filed Apr. 8, 2024) (“Related Litigation”). Ex-1019.

### **C. Lead and Back-Up Counsel**

|                                                                                                                                                                            |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Lead Counsel:</b>                                                                                                                                                       |
| Kevin Anderson, USPTO Reg. No. 43,471<br>DUANE MORRIS LLP, 901 New York Ave. NW Suite 700 East,<br>Washington, D.C. 20001<br>P: (202) 776-5213; KPAnderson@duanemorris.com |
| <b>Back-Up Counsel:</b>                                                                                                                                                    |
| Patrick D. McPherson, USPTO Reg. No. 46,255<br>DUANE MORRIS LLP, 901 New York Ave. NW Suite 700 East,                                                                      |

|                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Washington, D.C. 20001<br>P: (202) 776-5214; PDMcPherson@duanemorris.com                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Back-Up Counsel</b><br><br>Cory C. Bell, USPTO Reg. No. 75,096<br>Finnegan, Henderson, Farabow, Garrett & Dunner, LLP<br>Two Seaport Lane, 6th Floor, Boston, MA 02210-2001<br>P: (617) 646-1641; cory.bell@finnegan.com                                                                                                                                                                                                                      |
| <b>Back-Up Counsel</b><br><br>J. Preston Long, Ph.D., USPTO Reg. No. 65,125<br>Finnegan, Henderson, Farabow, Garrett & Dunner, LLP<br>901 New York Ave., NW, Washington D.C. 20001<br>P: (202) 408-4347; jp.long@finnegan.com<br><br>Alexander M. Boyer, USPTO Reg. 66,599<br>Finnegan, Henderson, Farabow, Garrett & Dunner, LLP<br>1875 Explorer Street, Suite 800<br>Reston, VA 20190-6023<br>P: (571) 203-2700; alexander.boyer@finnegan.com |

**D. Service Information**

Petitioners consent to being served electronically at the e-mail addresses provided above.

**IX. Grounds for Standing**

Petitioners certify that the '228 patent is available for IPR and that

Petitioners are not barred or estopped from requesting IPR challenging the patent claims on the grounds herein.

**X. Conclusion**

Petitioners request IPR and cancellation of the Challenged Claims. The Office may charge any required fees to Deposit Account No. 04-1679.

Date: December 13, 2024

Respectfully submitted,

By: Kevin P. Anderson

Lead Counsel  
Kevin Anderson, USPTO Reg. No. 43,471

### **CERTIFICATION OF WORD COUNT**

Pursuant to 37 C.F.R. § 42.24, Petitioners certify that the word count of Petitioners' Petition for *Inter Partes* Review (exclusive of any table of contents, table of authorities, mandatory notices under § 42.8, certificates of service and word count, and list of exhibits or claim listing) as measured by Microsoft Word is 13,843 and also includes 124 words hand-counted in the annotations for a total of 13,967 words.

Dated: December 13, 2024

By: Kevin P. Anderson  
Kevin P. Anderson, Lead Counsel  
Reg. No. 43,471

**CERTIFICATE OF SERVICE**

Pursuant to 37 C.F.R. §§ 42.6(e) and 42.105(a), the undersigned certifies that on December 13, 2024 a copy of the foregoing Petition for *Inter Partes* Review, the associated powers of attorney, and Exhibits were served Pursuant to 37 CFR § 42.105 and agreement on counsel via electronic download to the following attorneys:

Tarek Fahmi  
tarek.fahmi@ascendallaw.com

John Brinkmann  
John@bluepeak.law

Garland Stephens  
garland@bluepeak.law

PO litigation distribution list  
Dprime-mediatek-dist@bluepeak.law

Date: December 13, 2024

By: Kevin P. Anderson  
Kevin P. Anderson