

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

Mercedes-Benz USA, LLC,  
Petitioner,

v.

Daedalus Prime LLC,  
Patent Owner.

---

Case (to be assigned)  
Patent 9,575,895

---

**PETITION FOR *INTER PARTES* REVIEW OF  
U.S. PATENT NO. 9,575,895  
UNDER 35 U.S.C. §§311-319 AND 37 C.F.R. §42.100 *et seq.***

---

Filed on behalf of Petitioner:

Helen Trac (Reg. No. 62,250)  
HOGAN LOVELLS US LLP  
Four Embarcadero, #3500  
San Francisco, CA 94111  
Telephone: 415-374-2300  
Facsimile: 415-374-2399

Celine Jimenez Crowson (Reg. No. 40,357)  
Joseph Raffetto (Reg. No. 66,218)  
Scott Hughes (Reg. No. 68,385)  
Ryan Stephenson (Reg. No. 76,608)  
Nicholas Rotz (Reg. No. 75,959)  
HOGAN LOVELLS US LLP  
555 13th Street N.W.  
Washington, D.C. 20004  
Telephone: 202.637.5600  
Facsimile: 202.637.5710

## TABLE OF CONTENTS

|              |                                                                                                       |          |
|--------------|-------------------------------------------------------------------------------------------------------|----------|
| <b>I.</b>    | <b>INTRODUCTION .....</b>                                                                             | <b>1</b> |
| <b>II.</b>   | <b>MANDATORY NOTICES .....</b>                                                                        | <b>2</b> |
|              | A. Real Parties-in-Interest.....                                                                      | 2        |
|              | B. Related Matters.....                                                                               | 2        |
|              | C. Counsel and Service Information.....                                                               | 3        |
| <b>III.</b>  | <b>NOTICE OF FEES PAID .....</b>                                                                      | <b>3</b> |
| <b>IV.</b>   | <b>CERTIFICATION OF GROUNDS FOR STANDING .....</b>                                                    | <b>4</b> |
| <b>V.</b>    | <b>PRECISE RELIEF REQUESTED .....</b>                                                                 | <b>4</b> |
| <b>VI.</b>   | <b>THE '895.....</b>                                                                                  | <b>4</b> |
|              | A. '895 Overview.....                                                                                 | 4        |
|              | B. '895 File History .....                                                                            | 7        |
| <b>VII.</b>  | <b>CLAIM CONSTRUCTION AND POSITA DEFINITION.....</b>                                                  | <b>8</b> |
|              | A. Claim Construction.....                                                                            | 8        |
|              | B. Definition of a Person of Ordinary Skill in the Art.....                                           | 8        |
| <b>VIII.</b> | <b>ANALYSIS OF GROUNDS FOR UNPATENTABILITY.....</b>                                                   | <b>8</b> |
|              | A. Grounds Based on Sinharoy .....                                                                    | 8        |
|              | 1. <u>Ground 1a</u> – Sinharoy Anticipates or Renders Obvious Claims<br>1, 3-7, 11-13, and 15-17..... | 11       |
|              | a. Independent Claim 1 .....                                                                          | 11       |
|              | b. Independent Claim 11 .....                                                                         | 20       |
|              | c. Independent Claim 15 .....                                                                         | 22       |
|              | d. Dependent Claim 3 .....                                                                            | 25       |
|              | e. Dependent Claim 4.....                                                                             | 26       |
|              | f. Dependent Claim 5.....                                                                             | 27       |
|              | g. Dependent Claim 6.....                                                                             | 28       |
|              | h. Dependent Claim 7.....                                                                             | 29       |
|              | i. Dependent Claim 12.....                                                                            | 30       |
|              | j. Dependent Claim 13.....                                                                            | 31       |
|              | k. Dependent Claim 16.....                                                                            | 32       |
|              | l. Dependent Claim 17.....                                                                            | 32       |
|              | 2. <u>Ground 1b</u> – Sinharoy in View of Xu Renders Obvious Claims<br>2, 10.....                     | 33       |
|              | a. Dependent Claim 2 .....                                                                            | 33       |
|              | b. Dependent Claim 10.....                                                                            | 34       |
|              | 3. <u>Ground 1c</u> – Sinharoy in View of Tang Renders Obvious<br>Claims 8-9 and 14 .....             | 36       |

|            |                                                                                              |           |
|------------|----------------------------------------------------------------------------------------------|-----------|
| a.         | Dependent Claims 8, 14 .....                                                                 | 36        |
| b.         | Dependent Claim 9 .....                                                                      | 41        |
| B.         | Grounds Based on Tang .....                                                                  | 42        |
| 1.         | <u>Ground 2a</u> – Tang Anticipates or Renders Obvious Claims 1, 5-9, 11-14, 15, 17 .....    | 43        |
| a.         | Independent Claim 1 .....                                                                    | 43        |
| b.         | Independent Claim 11 .....                                                                   | 55        |
| c.         | Independent Claim 15 .....                                                                   | 58        |
| d.         | Dependent Claim 5 .....                                                                      | 61        |
| e.         | Dependent Claim 6 .....                                                                      | 62        |
| f.         | Dependent Claim 7 .....                                                                      | 63        |
| g.         | Dependent Claim 8 .....                                                                      | 64        |
| h.         | Dependent Claim 9 .....                                                                      | 65        |
| i.         | Dependent Claim 12 .....                                                                     | 66        |
| j.         | Dependent Claim 13 .....                                                                     | 68        |
| k.         | Dependent Claim 14 .....                                                                     | 70        |
| l.         | Dependent Claim 17 .....                                                                     | 70        |
| 2.         | <u>Ground 2b</u> – Tang in View of Harikumar Renders Obvious Claims 2, 3-4, 10, and 16 ..... | 70        |
| a.         | Dependent Claims 2, 10 .....                                                                 | 70        |
| b.         | Dependent Claims 3, 4, 16 .....                                                              | 73        |
| <b>IX.</b> | <b>TRIAL SHOULD BE INSTITUTED.....</b>                                                       | <b>77</b> |
| A.         | <i>Fintiv</i> Factors.....                                                                   | 77        |
| B.         | 35 U.S.C. § 325(d).....                                                                      | 78        |
| <b>X.</b>  | <b>CONCLUSION .....</b>                                                                      | <b>78</b> |

## **TABLE OF AUTHORITIES**

### **Page(s)**

#### **Cases**

|                                                                                                  |        |
|--------------------------------------------------------------------------------------------------|--------|
| <i>Apple, Inc. v. Fintiv, Inc.</i> ,<br>IPR2020-00019 (PTAB Mar 20, 2020) .....                  | 77, 78 |
| <i>Boss Corp. v. Koss Corp.</i> ,<br>IPR2021-00680 (PTAB Oct. 13, 2021) .....                    | 78     |
| <i>Ericsson Inc. v. Intellectual Ventures I LLC</i> ,<br>IPR2014-00527 (PTAB May 18, 2015) ..... | 9, 10  |
| <i>Sotera Wireless, Inc. v. Masimo Corp.</i> ,<br>IPR2020-01019 (PTAB Dec. 1, 2020) .....        | 78     |

#### **Statutes**

|                          |               |
|--------------------------|---------------|
| 35 U.S.C. § 102(a) ..... | 9, 33, 37, 71 |
| 35 U.S.C. §§ 102(b)..... | 37            |

## I. INTRODUCTION

Petitioner Mercedes-Benz USA, LLC (“**Mercedes**”) challenges Claims 1-17 of U.S. Patent No. 9,575,895 (“**’895**”) (EX1001). The ’895 discloses and claims nothing more than a well-known multicore processor with cores, a shared cache memory, a memory controller, an integrated input/output (IIO) module, and a caching agent to perform cache coherency operations. The processor is adapted to received data from an IO device coupled to the processor and directly store such data into the shared cache memory. The caching agent is a single agent, which includes distributed portions, each associated with a corresponding core.

None of this was new at the time of the alleged invention. Myriad prior art references disclose the basic elements of a multicore processor that (1) has a shared cache memory, a memory controller, an IIO module, and a caching agent, (2) can directly store data from an I/O device coupled to the processor into the shared cache memory, and (3) has a caching agent with distributed portions each associated with a corresponding core could be integrated together something new; such was done long before the ’895.

The two primary references discussed herein, together with the Declaration of computer processor expert Robert Horst,<sup>1</sup> underscore this. The first reference,

---

<sup>1</sup> The Declaration of Robert Horst (the “**Horst Decl.**”) is Exhibit 1008.

**Sinharoy**<sup>2</sup> (EX1003), is an IBM paper on its POWER7 multicore server processor and multichip systems comprising the processors (“**Sinharoy**”). Sinharoy discloses an eight-core processor with memory controllers, I/O controllers, L2 cache and L3 cache, and a single cache controller for the L2 and L3 caches. The L2 cache and L3 cache have distributed portions each corresponding to a given core. The second reference, **Tang**<sup>3</sup> (EX1004), discloses a processor design including multiple cores, memory controllers, an I/O bus, a last level cache (LLC) with distributed portions, a cache coherence controller, and an LLC-controller to manage cache coherence operations. Tang further discloses storing I/O data in the LLC.

Neither Sinharoy nor Tang was considered during prosecution of the ’895.

## **II. MANDATORY NOTICES**

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

The real parties-in-interest are Petitioner Mercedes-Benz USA, LLC; Mercedes-Benz AG; Mercedes-Benz Group AG; and Mercedes-Benz Intellectual Property GmbH & Co. KG.

### **B. Related Matters**

---

<sup>2</sup> “**Sinharoy**” is titled “*IBM POWER7 multicore server processor.*”

<sup>3</sup> “**Tang**” is titled “*DMA Cache: Using On-Chip Storage to Architecturally Separate I/O Data from CPU Data for Improving I/O Performance.*”

The '895 is at issue in *Daedalus Prime LLC v. Arrow Elecs., Inc.*, No. 1-22-cv-01107 (D. Del. August 23, 2022) (terminated), *Daedalus Prime LLC v. Mazda Motor Corp.*, No. 1-22-cv-01108 (D. Del. August 23, 2022) (terminated), *Daedalus Prime LLC v. Mazda Motor Corp.*, No. 1-22-cv-01109 (D. Del. August 23, 2022) (pending), and *In re Certain Semiconductors and Devices and Products Containing the Same, Including Printed Circuit Boards, Automotive Parts, and Automobiles*, Inv. No. 337-TA-1332 (ITC August 23, 2022) (terminated).

### **C. Counsel and Service Information**

Lead counsel is Celine Crowson (Reg. 40,357). Backup counsel are Joe Raffetto (Reg. 66,218), Scott Hughes (Reg. 68,385), Helen Y. Trac (Reg. No. 62,250), Ryan Stephenson (Reg. No. 76,608), and Nicholas Rotz (Reg. No. 75,959). Service information is as follows:

|                               |                                                                                                                                                                                                           |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Post and Hand Delivery</b> | Hogan Lovells US LLP<br>555 13th Street, N.W.<br>Washington, D.C. 20004                                                                                                                                   |
| <b>Email</b>                  | celine.crowson@hoganlovells.com<br>joseph.raffetto@hoganlovells.com<br>scott.hughes@hoganlovells.com<br>helen.trac@hoganlovells.com<br>ryan.stephenson@hoganlovells.com<br>nicholas.rotz@hoganlovells.com |
| <b>Telephone / Facsimile</b>  | 202.637.5600 / 202.637.5910                                                                                                                                                                               |

### **III. NOTICE OF FEES PAID**

Fees are submitted herewith. If additional fees are due during the proceeding, the undersigned authorizes the Office to charge them to Deposit Account No. 50-1349.

#### IV. CERTIFICATION OF GROUNDS FOR STANDING

Petitioner certifies that the '895 is available for *inter partes* review, and that Petitioner is not barred or estopped from requesting this review.

#### V. PRECISE RELIEF REQUESTED

The relief requested is cancellation of the challenged claims, as follows:

| Ground | References                   | Claims <sup>4</sup>                              | Basis     |
|--------|------------------------------|--------------------------------------------------|-----------|
| 1a     | Sinharoy                     | <b>1</b> , 3-7, <b>11-13</b> ,<br><b>15-17</b>   | 102 / 103 |
| 1b     | Sinharoy<br>in view of Xu    | 2, 10                                            | 103       |
| 1c     | Sinharoy<br>in view of Tang  | 8-9, 14                                          | 103       |
| 2a     | Tang                         | <b>1</b> , 5-9, <b>11-14</b> ,<br><b>15</b> , 17 | 102 / 103 |
| 2b     | Tang<br>in view of Harikumar | 2, 3-4, 10, 16                                   | 103       |

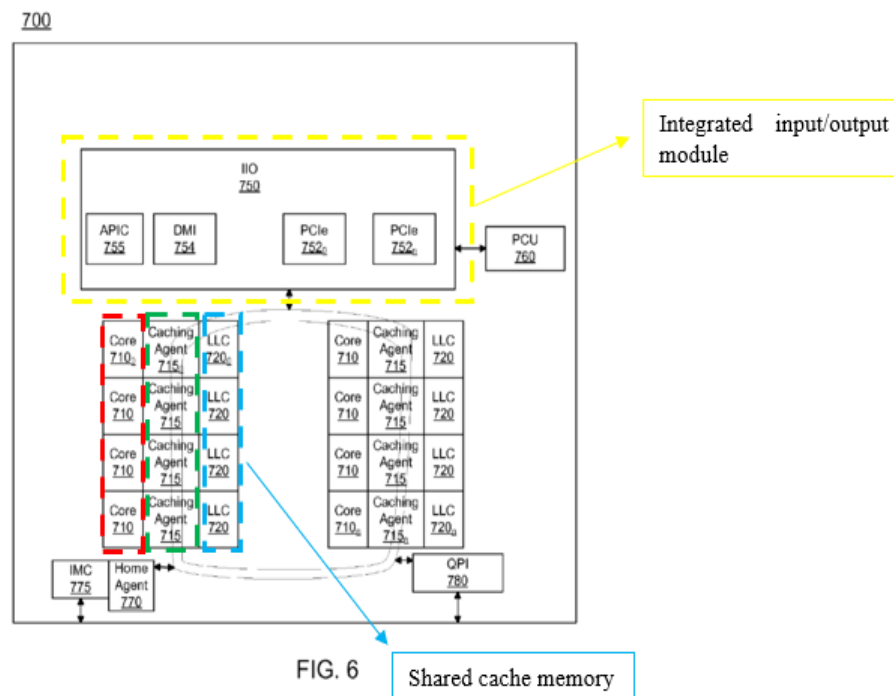
#### VI. THE '895

##### A. '895 Overview

---

<sup>4</sup> Independent claims in bold.

Independent Claim 1 of the '895 recites a “processor” that includes multiple “cores,” a “**shared cache memory**,” an “**integrated input/output module**” or “IIO” (e.g., I/O bus or I/O controller) coupled with an IO device, a “**caching agent**” (e.g., cache controller). ('895, Claim 1.) Specifically, the “processor” is adapted to “receive an allocation transaction from the IO device and directly store data of the allocation transaction into the **shared cache memory**.” (*Id.*) Further, the “**caching agent**” is a “single caching agent” with “**distributed portions**” each associated with a corresponding core. (*Id.*) In essence, the processor is adapted to store data from an I/O device directly into a shared cache memory without first storing the data in a system memory. (*Id.* at 3:34-37.)



(*Id.*, Fig. 6 (annotated).)

Referring to annotated Figure 6 above, the '895 describes how I/O data from an I/O device coupled to the IIO can be directly stored into the shared cache memory, i.e., the last level cache (LLC). (*Id.*, 3:20-3:32.) It provides a mechanism of cache injection without “first storing the data in a system memory.” (*See id.*, 3:34-37.) For example, an “allocation transaction” can be a “request to write data directly into a cache memory,” which can be received by the caching agent from an I/O device. (*Id.*, 3:55-59.) Moreover, 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. (*See id.*, 6:33-39.) The patent notes that although “distributed caching agents are shown, understand that these distributed portions form a single caching agent.” (*See id.*, 6:35-37.) The language in the '895 implies a differentiation between “logical” sharing/distribution and “physical” sharing/distribution. (*See Horst Decl.*, ¶25.) For example, the single logical caching agent 715 in FIG. 6 has physically “distributed portions.”

The multiple cores may be coupled to each other and to a “shared cache memory formed of a plurality of units of a LLC 740<sub>0</sub>-740<sub>n</sub><sup>5</sup> via caching agent 710<sub>n</sub>-715<sub>n</sub>.” ('895, 6:42-45.) The '895 describes that the shared cache memory (LLC) can be partitioned as well. (*Id.*, 6:34-35.) Each distributed portion of the LLC is

---

<sup>5</sup> The LLC 740<sub>0</sub>-740<sub>n</sub> should have been 720<sub>0</sub>-720<sub>n</sub> in FIG. 6.

associated with a corresponding core. (*Id.*, 6:42-46.) The cores, LLC, IIO module, and caching agent may be coupled together via a ring interconnect. (*Id.*, 6:45-49.)

## **B. '895 File History**

The '895 was filed January 30, 2015 and issued February 21, 2017. The '895 is a continuation of U.S. Application No. 13/324,053, filed December 13, 2011. During examination, a Non-Final Office Action was issued, rejecting claims on double patenting, §101, and §103 grounds. ('895 Patent File History, Non-Final Office Action mailed June 7, 2016 (EX1002, 79, 86-87)). However, no prior art rejections were made against original dependent claim 3, which recited “wherein the caching agent is a single caching agent for the processor and including a plurality of distributed portions each associated with a corresponding one of the plurality of cores.” (*Id.*, 87).<sup>6</sup> In response, the Applicant incorporated these limitations from original claim 3 into the independent claims and filed a terminal disclaimer. (*Id.*, 117, 120). An allowance was then issued. (*Id.*, 128).

---

<sup>6</sup> During prosecution of the parent application, U.S. Application No. 13/324,053, in response to an Office Action the Applicant argued that a “single caching agent” including a plurality of “distributed portions” was not present in the cited prior art, and amended the independent claims to include this feature. (EX1015, 42 ('053 File History). A Notice of Allowance as then mailed. (EX1015, 15.)

## **VII. CLAIM CONSTRUCTION AND POSITA DEFINITION**

### **A. Claim Construction**

Claims in an *inter partes* review “shall be construed using the same claim construction standard that would be used to construe the claim in a civil action under 35 U.S.C. 282(b), including construing the claim in accordance with the ordinary and customary meaning of such claim as understood by one of ordinary skill in the art and the prosecution history pertaining to the patent.” 37 C.F.R. §42.100(b). Unless indicated otherwise, all claim terms herein are given such ordinary and customary meaning.

### **B. Definition of a Person of Ordinary Skill in the Art**

A person of ordinary skill in the art (“**POSITA**”) in the field of the ’895 would have had a good working knowledge of computer science or electrical engineering and multicore processor systems. (Horst Decl., ¶30.) She would have gained this through an undergraduate Bachelor’s degree in computer science or electrical engineering, or a comparable field, as well as some experience in the field (*e.g.*, in multicore processor systems). (*Id.*) The more education one has, the less experience needed to attain an ordinary level of skill. (*Id.*) Similarly, more experience in the field may serve as a substitute for formal education. (*Id.*)

## **VIII. ANALYSIS OF GROUNDS FOR UNPATENTABILITY**

### **A. Grounds Based on Sinharoy**

Sinharoy describes IBM's POWER7 multicore server processor chip. (Horst Decl., ¶45; Sinharoy 1:1.)

Sinharoy is prior art under at least pre-AIA §102(a). Sinharoy was published in *IBM Journal of Research and Development*, Volume 55, Issue 3 (May-June 2011) with a 2011 copyright notice. (Sinharoy 1:1.) Sinharoy was published and is available on IEEE Xplore (<https://ieeexplore.ieee.org/document/5757896>), which states that a copy has been available on that site since April 29, 2011. (EX1010.)

Moreover, Gordon MacPherson, Director, Board Governance & Policy Development at the Institute of Electrical and Electronics Engineers (IEEE), confirmed that Sinharoy was available on IEEE Xplore at least as of the last day of April 2011. (EX1009.) Sinharoy bears a publication date of April 2011, which is inherently reliable and trustworthy given the publication source, the IEEE.

*Ericsson Inc. v. Intellectual Ventures I LLC*, IPR2014-00527, Paper 41, at 11 (PTAB May 18, 2015) (stating "IEEE is a well-known, reputable compiler and publisher of scientific and technical publications, and we take Official Notice that members in the scientific and technical communities who both publish and engage in research rely on information published on the copyright line of IEEE publications."). The Board has further found that "[a]llowing IPR petitioners to rely on the IEEE publication date in an IPR proceeding ... serves the interests of justice." *Id.* at 12.

Sinharoy discloses a server processor, the “POWER7,” that is the seventh-generation processor in IBM’s POWER processor line. (Sinharoy, 1:1-2.) The POWER7 has eight cores, two memory controllers, an L2 cache, and an L3 cache. (*Id.*, 1:2; 1:18-19). In particular, the POWER7 processor has a “256-KB POWER7 L2 cache” and a “shared 32-MB Power L3 cache,” “composed of eight 4MB L3 regions.” (*Id.*, 1:18-19).

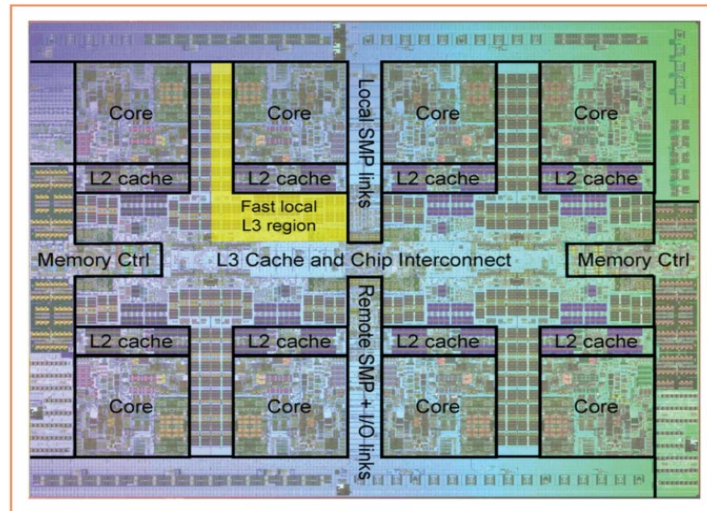


Figure 1

Die photo of the IBM POWER7 chip.

(*Id.*, Figure 1.)

The processor supports “two integrated I/O controllers,” each supporting a “4-byte off-chip read interface and 4-byte off-chip write interface, thereby connecting the POWER7 chip to up to two I/O hub chips.” (*Id.*, 1:22). These I/O hub chips are connected to PCI bridge and PCI adapter. (*Id.*, Figure 14; Horst Decl., ¶45.)

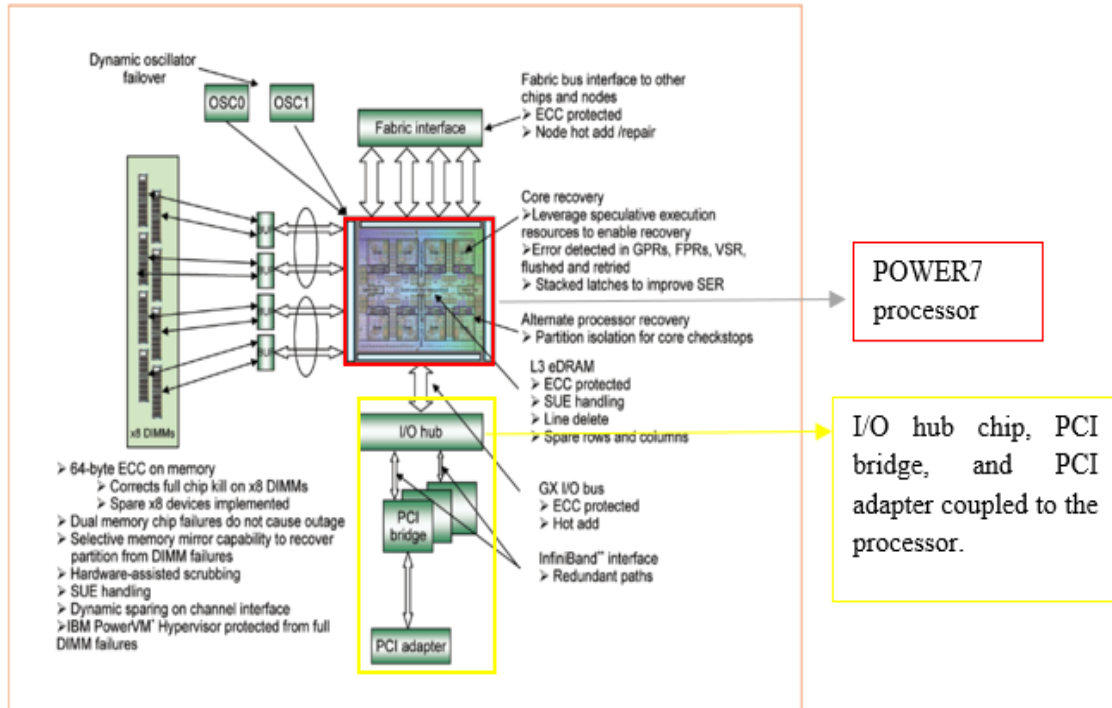


Figure 14

IBM POWER7 reliability and availability features.

(Sinharoy, Figure 14 (annotated).)

The L2 and L3 caches support “cache injection,” i.e., an “I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory.” (*Id.*, 1:17). The L2 and L3 cache each has centralized cache controllers to manage the read/write requests and cache coherence operations. (*Id.*, 1:18-19).

1. **Ground 1a – Sinharoy Anticipates or Renders Obvious Claims 1, 3-7, 11-13, and 15-17**

a. **Independent Claim 1**

Sinharoy anticipates or renders obvious Claim 1. (Horst Decl., ¶¶47-66.)

**Claim 1p: A processor comprising:**

Sinharoy discloses Claim 1p. (*Id.*, ¶47.) Sinharoy discloses a POWER7 “multicore server processor.” (Sinharoy, 1:1, Figure 1.)

**Claim 1a: a plurality of cores,**

Sinharoy discloses Claim 1a. (Horst Decl., ¶48.) Sinharoy discloses the POWER7 processor includes eight cores:

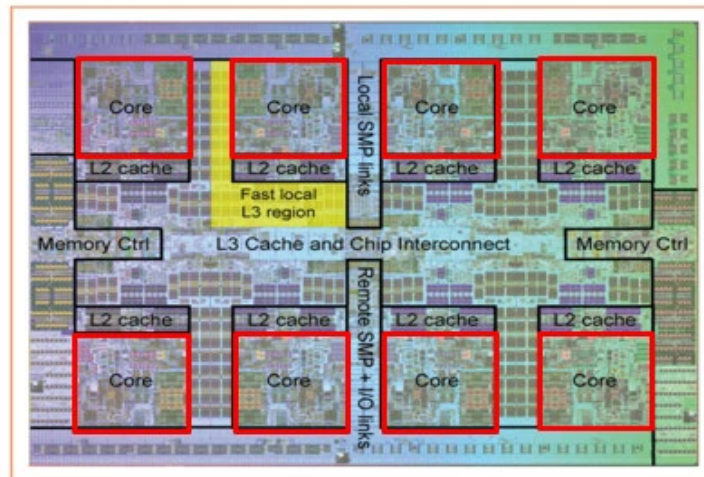


Figure 1

Die photo of the IBM POWER7 chip.

(Sinharoy, Figure 1 (annotated), 1:1, 1:2 (“Figure 1 shows the POWER7 chip, which has eight processor cores . . .”).)

**Claim 1b: a shared cache memory,**

Sinharoy discloses Claim 1b. (Horst Decl., ¶49.) Sinharoy discloses the POWER7 has a “shared 32-MB L3 cache” comprising “eight 4-MB L3 regions.” (Sinharoy, 1:18, 1:2 (“the large 32-MB-shared-L3 cache to be on-chip”).) Each local L3 region is “tightly coupled to the L2 cache associated with a given core.”

(*Id.*, 1:18.) Moreover, the “shared 32-MB L3 cache comprised of the 4-MB local L3 regions from the eight cores” are “shared by the eight cores.” (*Id.*, 1:17; Horst Decl., ¶49.) Sinharoy explains that 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, 1:18.)

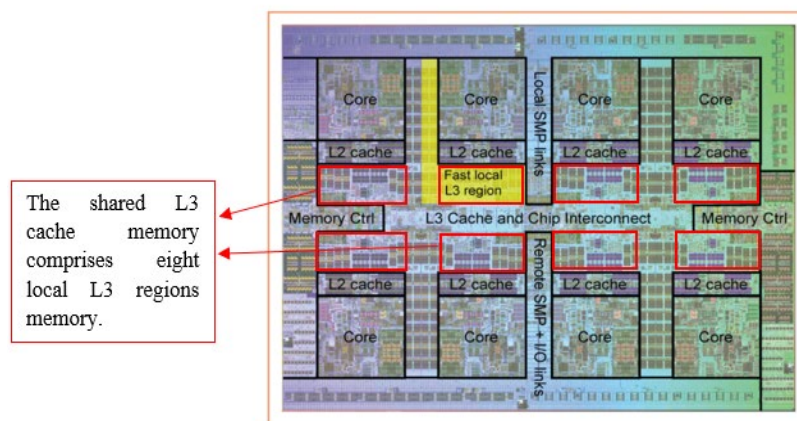


Figure 1

Die photo of the IBM POWER7 chip.

(*Id.*, Figure 1 (annotated).)

**Claim 1c: a memory controller to interface with a memory coupled to the processor,**

Sinharoy discloses Claim 1c. (Horst Decl., ¶¶50-52.) Sinharoy discloses the POWER7 processor has “two memory controllers—one on each side of the chip.” (Sinharoy, 1:2.) Each memory controller interfaces with memory coupled to the processor, as each supports “four channels of double-data-rate-three (DDR3)

memory” that “provide 100 GB/s of sustained memory bandwidth.” (*Id.*) The DDR3 memory is part of a “memory subsystem.” (*Id.*, 1:21-22.) In this memory subsystem, a given memory channel “can be connected through an off-chip interface to a memory buffer chip.” (*Id.*, 1:21.) The memory subsystem provides “a dual port interface between the buffer chip and the memory chips” and may use “DDR3 memory chips.” (*Id.*, 1:21.) By providing the memory bandwidth to the POWER7 processor chip and connecting to the processor chip through the off-chip interface and the memory buffer chip, the DDR memory chips are “coupled to” the POWER7 processor. (*Id.*, 1:22; Horst Decl., ¶51.)

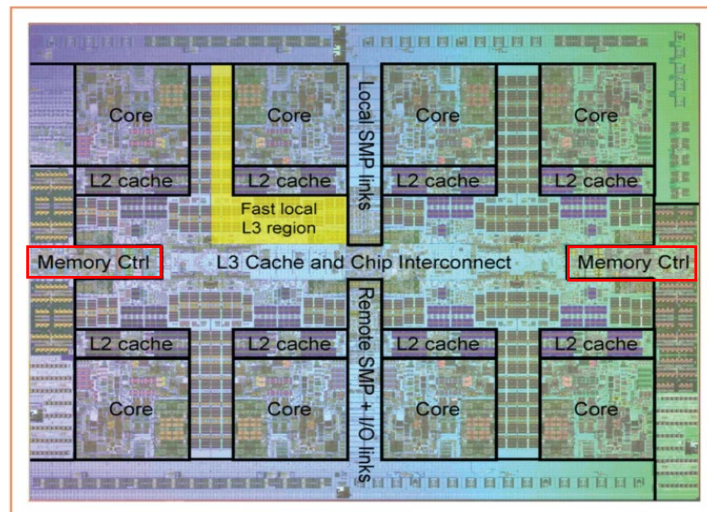


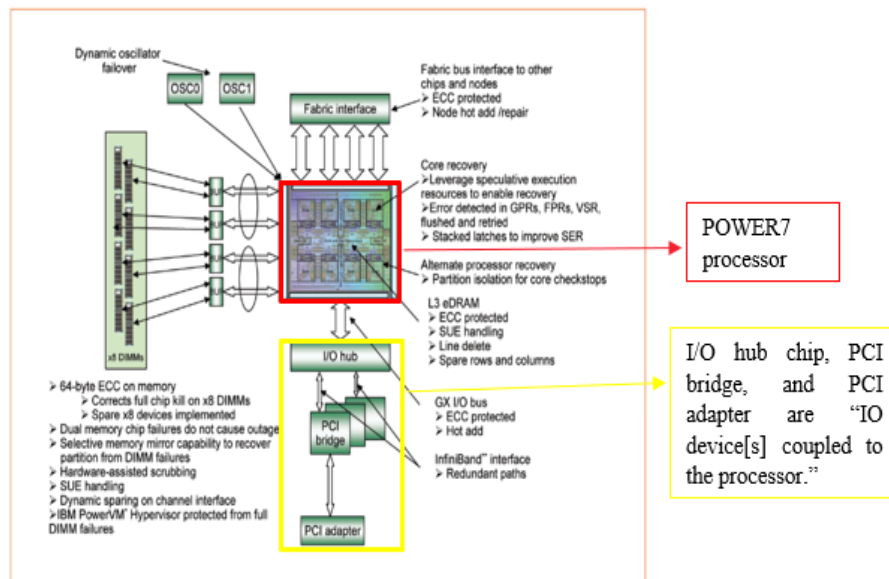
Figure 1

Die photo of the IBM POWER7 chip.

(Sinharoy, Figure 1 (annotated).)

**Claim 1d:** *an integrated input/output (IIO) module to interface between the processor and an IO device coupled to the processor and*

Sinharoy discloses Claim 1d. (Horst Decl., ¶¶53-54). Sinharoy discloses the POWER7 includes integrated input/output modules namely “two integrated I/O controllers,” that interface between the POWER7 processor and “IO devices,” including I/O hub chips connected to PCI bridges and PCI adapters, that are coupled to the processor. (Sinharoy 1:22, Figure 14; *see* Horst Decl., ¶53.) For example, each integrated I/O controller “supports a proprietary 4-byte off-chip read interface and a 4-byte off-chip write interface, thereby connecting the POWER7 chip to up to two I/O hub chips.” (*Id.*) That is, the two integrated I/O controllers interact between the POWER7 processor and the I/O hub chips, PCI bridge, and PCI adapter through the off-chip read and write interfaces:



(*Id.*, Figure 14 (annotated), 1:21-22, 1:17-1:18; Horst Decl., ¶54.)

**Claim 1e: a caching agent to perform cache coherency operations for the plurality of cores and the IIO module,**

Sinharoy discloses Claim 1e. (Horst Decl., ¶¶55-58). Sinharoy discloses a “caching agent,” as it discloses centralized controllers for the L2 and L3 caches, including 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, 1:18-19.) In particular, Sinharoy discloses the L2 cache includes a “single centralized controller, with two address-hashed cache data arrays” and each L3 cache region includes a “single centralized controller.” (*Id.*, 1:18-19.) Both L2 and L3 cache controllers have read/claim (RC) machines incorporated in them, which manage “core fetches, read-modify-write operations on behalf core stores, and core prefetch operations,” which are cache coherency operations. (*Id.*, 1:18-19; Horst Decl., ¶56.) Furthermore, the pool of “ten write machines” also handles cache coherency operations. (Sinharoy, 1:19, 1:22; Horst Decl., ¶57.) Thus, the cache controllers for the L2 and L3 caches including the pool of RC and write machines are a “caching agent to perform cache coherency operations for the plurality of cores.” (Horst Decl., ¶¶56-58.)

The same “caching agent” comprising the L2 and L3 cache controllers with the pool of RC and write machines also performs cache coherency operations for the integrated I/O controllers (“IIO modules”). (Horst Decl., ¶57). Sinharoy discloses that the POWER7’s integrated I/O controllers “support cache injection,

which enables DMA write traffic to be written directly into a cache, instead of to memory.” (Sinharoy, 1:22.) This “cache injection” operation is a “cache coherency operation for the IIO module.” (Horst Decl., ¶57; Sinharoy, 1:17, Table 2 (noting “cache injection” from an IO device is a “coherent operation.”) Furthermore, the write machines within the “caching agent” also handle these “cache-injection” operations. (Sinharoy, 1:19.)

**Claim 1f:** *wherein the processor is to receive an allocation transaction from the IO device and directly store data of the allocation transaction into the shared cache memory,*

Sinharoy discloses Claim 1f. (Horst Decl., ¶¶59-64). Sinharoy discloses the processor receives requests to write data into the shared cache memory (*i.e.*, “allocation transactions”) from IO devices (such as the I/O hub chips). (Sinharoy, 1:17, 1:22; Horst Decl., ¶¶61-63.) Data from these write requests is directly stored into the shared cache memory. (*Id.*; Horst Decl., ¶61.) Sinharoy discloses the shared L2 and L3 caches support cache injection operations, where an “I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory.” (Sinharoy, 1:17, 1:22; Horst Decl., ¶60.) The pool of the write machines in the L3 cache also handles “cache-injections.” (Sinharoy, 1:19). Thus, when they perform cache injection operations to write data from the I/O hub chip into the L3 cache ( “shared cache memory”),

the processor “directly store[s] data of the allocation transaction into the shared cache memory.” (Horst Decl., ¶64.)

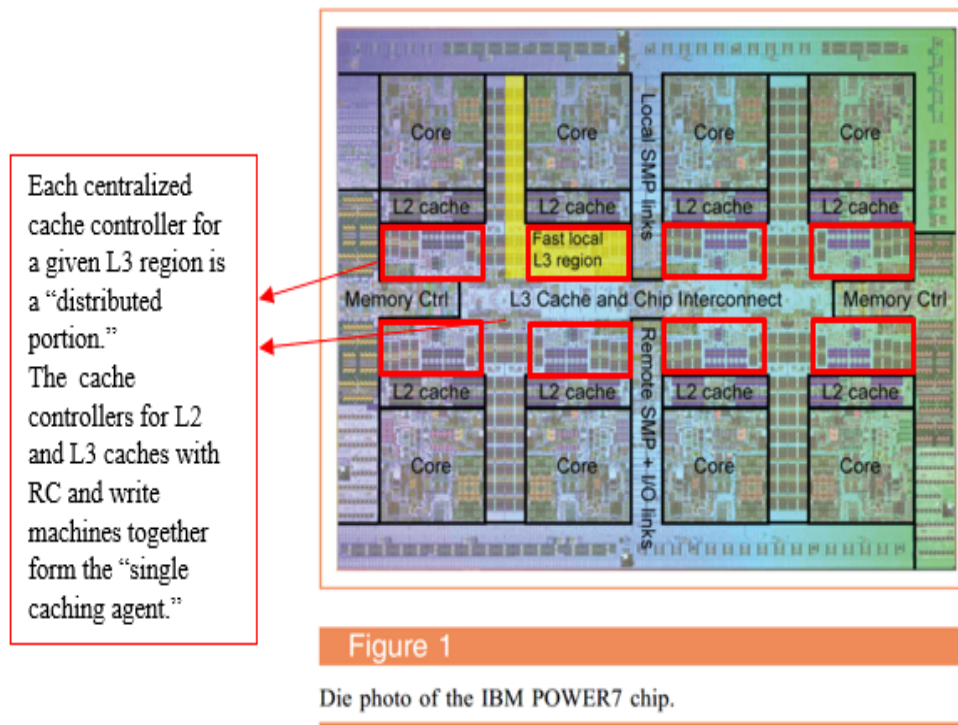
**Claim 1g:** *wherein the caching agent is a single caching agent for the processor and includes a plurality of distributed portions each associated with a corresponding one of the plurality of cores.*

Sinharoy discloses Claim 1g. (Horst Decl., ¶¶65-66.) Sinharoy discloses that the caching agent, comprising the centralized cache controllers for the L2 and L3 caches including RC and write machines, is formed as a single caching agent for the processor. *See supra* Section VIII.A.1.a(Claim 1e). The cache controllers for the L2 and L3 caches, including RC and write machines, form a logical single caching controller, with the single cache controller including a plurality of distributed portions each associated with one of the cores. (*Id.*, Horst Decl., ¶65.)<sup>7</sup>

---

<sup>7</sup> The '895 Patent discloses that 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.” ('895 Patent, Figure 6, 6:33-35.) In particular, “while distributed caching agents are shown, understand that these distributed portions **form** a single cache agent.” (*Id.*, 6:35-37 (emphasis added).) Thus, the “single caching agent” is a single logical unit for the whole processor with distributed physical portions. (Horst Decl., ¶65.) A “single caching agent” is a necessary single

In particular, each centralized cache controller for a given L3 region is a “distributed portion” of the caching agent. (Sinharoy, 1:19; Horst Decl., ¶66.) Each “centralized L3 region controller provides a single core/L2 dispatch port . . .” and thus each L3 region cache controller corresponds to a given core. (Sinharoy, 1:19; Horst Decl., ¶66.) Moreover, because the local L3 region is tightly “coupled to the L2 cache associated with a given core,” each controller for its corresponding L3 region is associated with a “corresponding one of the plurality of cores.” (Sinharoy 1:19.; Horst Decl., ¶66.)



logical component for the processor formed by physically distributed portions. (Horst Decl., ¶65.)

(Sinharoy, Figure 1 (annotated).)

**b. Independent Claim 11**

Sinharoy anticipates or renders obvious Claim 11. (Horst Decl., ¶¶67-73.)

**Claim 11p:** *A non-transitory machine-readable medium having stored thereon instructions, which if performed by a machine cause the machine to perform a method comprising:*

Sinharoy discloses Claim 11p. (Horst Decl., ¶¶67.) Sinharoy discloses a multicore processor that can perform certain instruction sets. (Sinharoy, 1:1.) The processor is a “non-transitory machine-readable medium having stored thereon instructions.” (Horst Decl., ¶¶67).

**Claim 11a:** *receiving an allocation transaction in an integrated input/output (I/O) module of a multicore processor from an IO device coupled to the multicore processor;*

Sinharoy discloses Claim 11a. (Horst Decl., ¶¶68-69.) Sinharoy discloses the POWER7 is an eight-core processor. (Sinharoy, 1:1-1:2, Figure 1.) Sinharoy further discloses the POWER7 includes integrated input/output modules, namely “two integrated I/O controllers,” that interface between the processor and “IO devices,” including I/O hub chips connected to PCI bridges and PCI adapters. (*Supra* Section VIII.A.1.a(Claim 1d)) These I/O integrated controllers “support cache injection, which enables DMA write traffic to be written directly into a cache, instead of to memory.” (Sinharoy, 1:22.) The processor receives requests to write data into the shared cache memory (*i.e.*, receives “allocation transactions”)

from IO devices (such as the I/O hub chips ). (Sinharoy, 1:17-19, 1:22; *see also supra* Section VIII.A.1.a(Claim 1f).)

**Claim 11b:** *forwarding the allocation transaction to a caching agent of the multicore processor to cause the data to be stored into a shared cache memory of the multicore processor, the caching agent shared by a plurality of cores of the multicore processor and the IIO module,*

Sinharoy discloses Claim 11b. (Horst Decl., ¶¶70-71.) Sinharoy discloses that the allocation transactions are forwarded to a “caching agent” of the processor, as the requests to write are sent to cache controllers for L2 and L3 caches including RC and write machines. (*See supra* VIII.A.1.a(Claim 1e); Horst Decl., ¶70.) As stated in Claim 1f, writing data into a shared cache memory (L3 cache) from an I/O hub chip (i.e., the IO device) includes a request to write (i.e., allocation transaction) during the cache injection operation. (*See supra* VIII.A.1.a(Claim 1f); Horst Decl., ¶71). Thus, when the processor performs the cache injection operation to directly store data into the shared L3 cache, the operation involves forwarding the allocation transaction to the caching agent “to cause the data to be stored into a shared cache memory.” (Horst Decl., ¶71). Furthermore, as stated in Claim 1e, the caching agent is shared by the cores and the IIO module, because it performs cache coherency operations for both of them. (*See supra* VIII.A.1.a(Claim 1e); Horst Decl., ¶70.)

**Claim 11c:** *wherein the caching agent is a single caching agent for the multicore processor and includes a plurality of distributed portions each associated with a corresponding one of the plurality of cores; and*

Sinharoy discloses Claim 11c. (*See supra* VIII.A.1.a(Claim 1g); Horst Decl., ¶72.)

**Claim 11d:** *directly storing the data of the allocation transaction into the shared cache memory of the multicore processor without sending the data to a system memory coupled to the multicore processor.*

Sinharoy discloses Claim 11d. (Horst Decl., ¶73.) By providing the memory bandwidth to the POWER7 processor chip and connecting to the processor chip through the off-chip interface and the memory buffer chip, the DDR memory chips in Sinharoy are “system memory coupled to the processor.” (*See supra* Section VIII.A.1.a(Claim 1c).) Further, as explained above (*see supra* Section VIII.A.1.a(Claim 1f), the shared L2 and L3 caches support cache injection operations, where an “I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory.” (Sinharoy, 1:17, 1:19, 1:22; Horst Decl., ¶73.) Thus, when the processor performs cache injection operations to write data from the I/O hub chip into the L3 cache (i.e., the “shared cache memory”), it “directly store[s] data of the allocation transaction into the shared cache memory” of the multicore processor without “sending the data to a system memory.” (Horst Decl., ¶73.)

**c. Independent Claim 15**

Sinharoy anticipates or renders obvious Claim 15. (Horst Decl., ¶¶74-80.)

**Claim 15p:** *A system comprising:*

Sinharoy discloses Claim 15p. (*Id.*, ¶74.) Sinharoy discloses a system with multiple POWER7 processors. (Sinharoy, 1:1-1:2, 1:23.)

**Claim 15a:** *a first multicore processor including a first plurality of cores, a first shared cache memory, 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, and a first caching agent to perform cache coherency operations for the first plurality of cores and the first IIO module,*

Sinharoy discloses Claim 15a. (Horst Decl., ¶75.) (*See supra* VIII.A.1.a(Claims 1a-1e).)

**Claim 15b:** *wherein the first caching agent is to receive an allocation transaction from the first IIO module and directly store data of the allocation transaction into the first shared cache memory,*

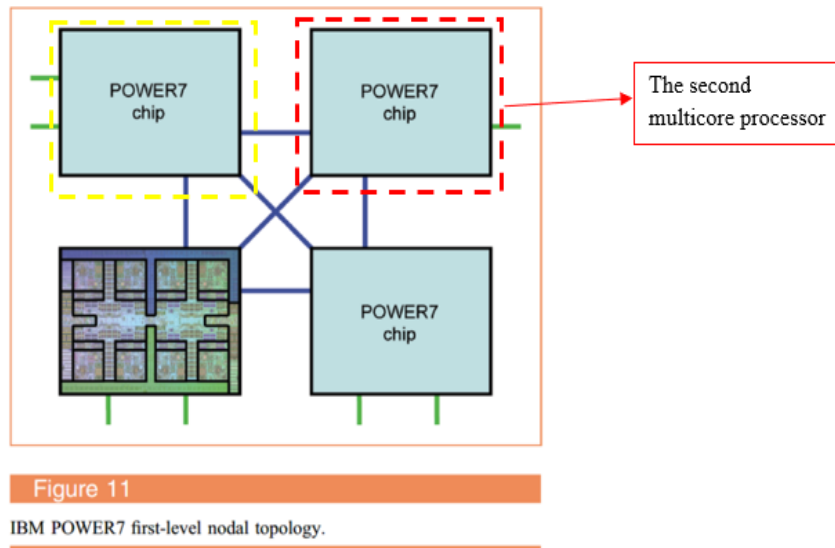
Sinharoy discloses Claim 15b. (Horst Decl., ¶76.) (*See supra* VIII.A.1.a (Claim 1f).)

**Claim 15c:** *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,*

Sinharoy discloses. (Horst Decl., ¶77.) (*See supra* VIII.A(Claim 1g).)

**Claim 15d:** *a second multicore processor including a second plurality of cores, a second shared cache memory, a second IIO module to interface between the second multicore processor and a second plurality of IO devices coupled to the second multicore processor, and a second caching agent to perform cache coherency operations for the second plurality of cores and the second IIO module; and*

Sinharoy discloses Claim 15d. (Horst Decl., ¶78.) Sinharoy discloses a system which combines up to four POWER7 chips. (Sinharoy, 1:23.) Each POWER7 processor is a multicore processor, and thus Sinharoy discloses both a first and second multicore processor:



(Sinharoy, Figure 11 (annotated), 1:1-1:2, 1:23; Horst Decl., ¶78.)

This second multicore POWER7 processor, just like the first POWER7 processor, includes the same components. (*See supra* VIII.A.1.a(Claims 1a-1e).)

**Claim 15e:** *a peripheral controller coupled to at least one of the first and second multicore processors.*

Sinharoy discloses Claim 15e. (Horst Decl., ¶79.) The first and second POWER7 processors in the system are coupled to a peripheral controller as they are connected to a cluster interconnect chip that incorporates up to three  $16 \times$  PCI (peripheral component interconnect) Express\*\* Gen2 I/O subsystem interfaces:

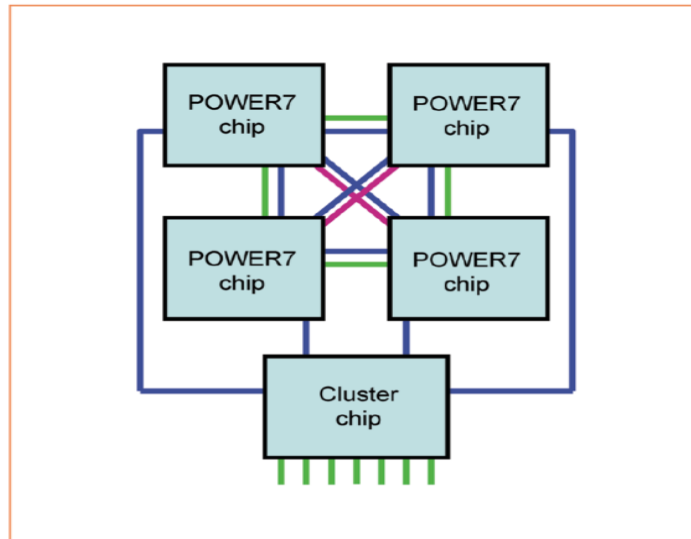


**Figure 13**  
Cluster system first-level coherent nodal topology.

(Sinharoy, Figure 13, 1:25.)

#### d. Dependent Claim 3

**Claim 3:** *The processor 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 discloses Claim 3. (Horst Decl., ¶¶81-83.) Sinharoy’s caching agent is comprised of the centralized cache controllers for L2 and L3 caches including RC and write machines. *See supra* Section VIII.A.1.a(Claim 1e). Because RC machines “aggressively reorder storage operations” and manage “core fetches, read-modify-write operations, on behalf of core stores, and core prefetch operations,” they include temporary storage to store incoming (pending) requests, such as requests to read or to write for the cores. (Sinharoy, 1:18; Horst Decl.,

¶82.) The temporary storage<sup>8</sup> in the RC machines are “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, 1:19; *see supra* Section VIII.A.1.a(Claim 1e).) Integrated I/O controllers (i.e., the IIO modules) also “support cache injection.” (Sinharoy 1:22.) The write operations include incoming (“pending”) requests to write for the IIO module. (Horst Decl., ¶83.) When performing cache injection operation, 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.*)

**e. Dependent Claim 4**

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

Sinharoy discloses Claim 4. (Horst Decl., ¶¶84-86.) As stated above, the RC and write machines within the caching agent of Sinharoy include temporary storages (i.e., tables of request) to store incoming requests (e.g., requests to write or read). (*See supra* Section VIII.A.1.d(Claim 3).) When write machines perform

---

<sup>8</sup> The '895 discloses that a “table of requests (TOR)” may be a buffer or other temporary storage for holding incoming requests. ('895, 2:51-53.)

cache injection operations, they involve writing a cache line into a local L3 region. (Sinharoy, 1:17, 1:19.) Such a write transaction is a “posted transaction.”<sup>9</sup> (Horst Decl., ¶¶84-86.) Because write machines within the caching agent of Sinharoy have temporary storages, they include dedicated queues or resources (i.e., dedicated entry) to store such a write transaction (i.e., posted transaction) when performing the cache injection. (Horst Decl., ¶¶85-86.)

**f. Dependent Claim 5**

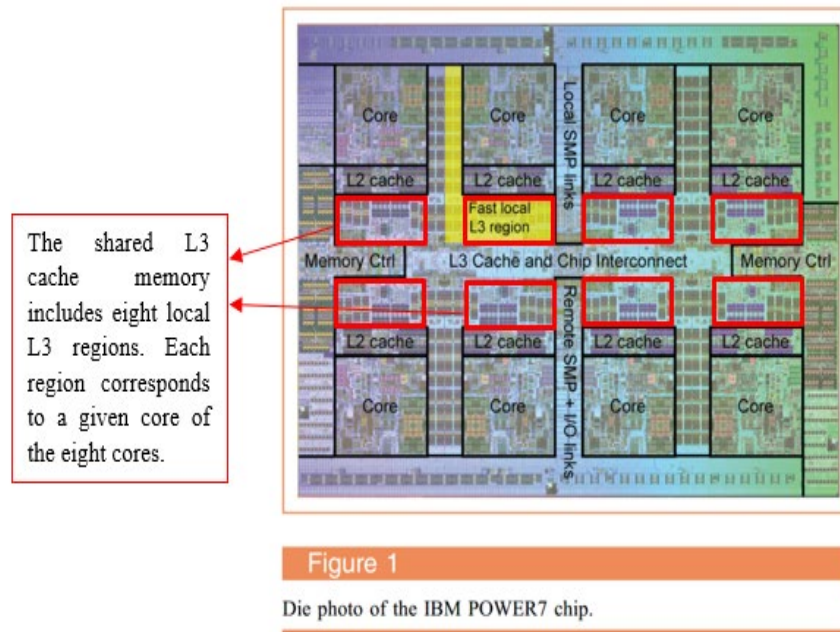
***Claim 5: The processor of claim 1, wherein the shared cache memory comprises a last level cache (LLC) including a plurality of distributed portions each associated with a corresponding one of the plurality of cores.***

Sinharoy discloses Claim 5. (Horst Decl., ¶¶87-89.) The shared L3 cache memory is the last level cache. (Sinharoy, Abstract, 1:19, FIG. 1; Horst Decl., ¶87; *see also supra* Section VIII.A.1.a(Claim 1b).) The shared L3 cache in Sinharoy includes a plurality of distributed portions (i.e., “eight 4-MB L3 regions”). (Sinharoy, Abstract, 1:19.) Each L3 region is tightly coupled to the L2 cache “associated with a given core.” (*Id.*) As such, each local L3 region is a “distributed

---

<sup>9</sup> The '895 Patent discloses the TOR may include at least one “dedicated resource for post transactions,” and that one example of a posted transaction is a “write transaction.” ('895 Patent, 2:53-57, 5:12-17.)

portion” of the shared cache memory, which corresponds to one of the eight cores of the POWER7 processor. (Horst Decl., ¶89; *see also* Sinharoy, Figure 1.)



(Sinharoy, Figure 1 (annotated).)

### g. Dependent Claim 6

***Claim 6:*** *The processor of claim 1, wherein the processor is to directly store the data of the allocation transaction into the shared cache memory without initiation of a memory transaction to the memory.*

Sinharoy discloses Claim 6. (Horst Decl., ¶90.) Sinharoy discloses the L2 and L3 caches support “cache injection” operations, and an “I/O device performing [cache injection] operation may target the operation to the cache, **instead of to memory.**” (Sinharoy, 1:17 (emphasis added); 1:22.) Thus, a cache injection operation directly stores data into the shared L3 cache (shared cache) and does not initiate a “memory transaction to the memory.” (Horst Decl., ¶90.)

#### **h. Dependent Claim 7**

**Claim 7:** *The processor of claim 6, wherein the processor is to write the data of the allocation transaction to the memory responsive to eviction of a cache line including the data.*

Sinharoy discloses Claim 7. (Horst Decl., ¶¶91-92.) A cache line is the smallest portion of data that can be mapped into a cache. (Horst Decl., ¶91.) The POWER7 processor can move a given cache line among the caches. (Sinharoy 1:18; Horst Decl., ¶91.) When prefetched data is staged to the L3 cache, it is managed by a pool of “ten write machines” each having an associated cast-out machine to manage the “evictions of lines” displaced by writes. (Sinharoy, 1:19.) The “evictions of lines” are evictions of cache lines including corresponding data. (Horst Decl., ¶91.)

The POWER7 processor is to write the data of the allocation transaction to the memory responsive to eviction of a cache line. (See Sinharoy, 1:20-21, Figure 10.) When a “line is evicted from an L3 region (due to an L2 cast-out, a lateral cast-out from another L3 region, a prefetch operation, or a cache injection operation), if the evicted line is second class and has a ‘dirty’ state, it is written back to memory.” (*Id.*, 1:21.) A “dirty” line is a cache line that has been modified and needs to be written back to the memory. (Horst Decl., ¶92.)

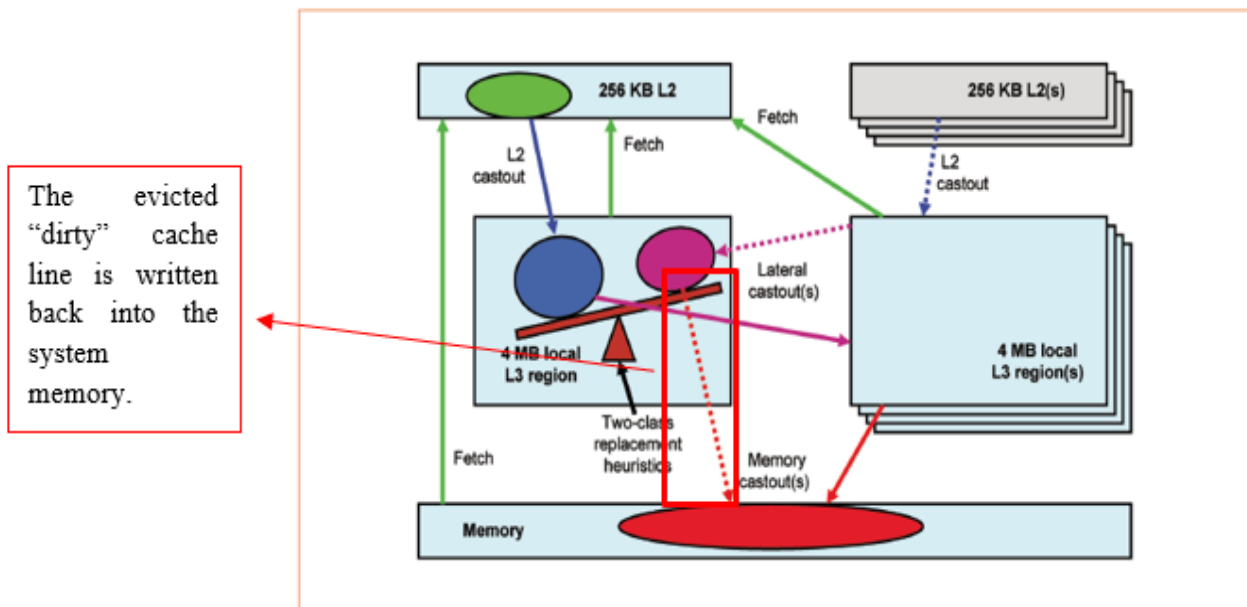


Figure 10

L3 region behavior.

(Sinharoy, Figure 10 (annotated).)

### i. Dependent Claim 12

**Claim 12:** *The non-transitory machine-readable medium of claim 11, wherein the method further comprises 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 the system memory.*

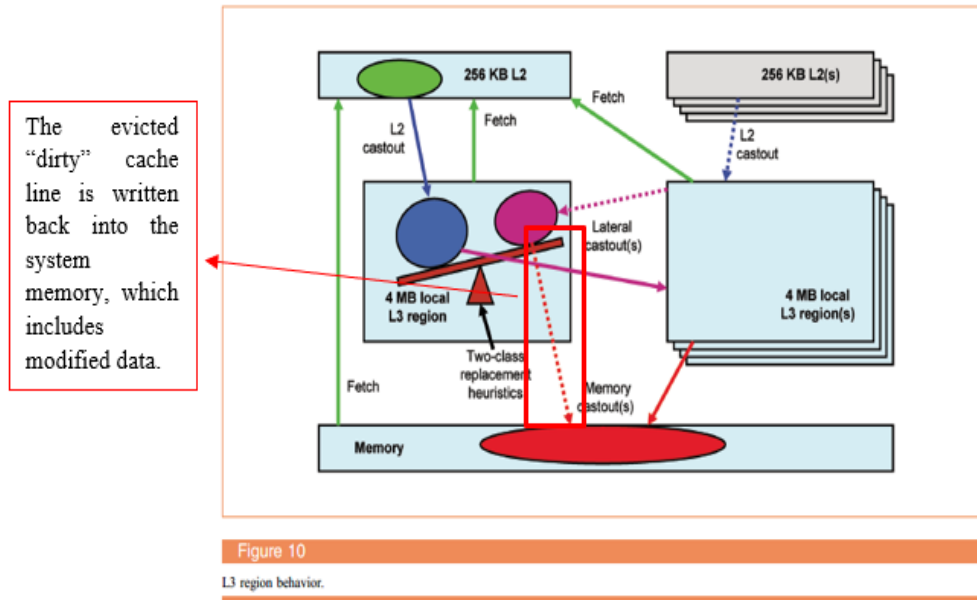
Sinharoy discloses Claim 12. (Horst Decl., ¶93.) Sinharoy discloses accessing data in the shared cache memory via at least one of the cores, as the “adaptive L3 cache management policy routes instructions and data to the 4-MB local L3 regions of the core or cores that most actively utilize them, enabling them to experience reduced latency, even cloning copies when multiple cores are actively sharing them but also destroying copies as they fall into disuse in order to

preserve capacity.” (Sinharoy, 1:17.) When a core actively utilizes a region of L2 with reduced latency, at least one core is accessing the data a number of times without sending the transaction to system memory. (Horst Decl., ¶93.) The purpose of a local cache for a core is to access data without sending a transaction to memory. (Horst Decl., ¶93.) When the POWER7 processor performs the cache injection operation and stores data into the local L3 cache, the data can be accessed a number of times by at least one of the cores connected to the cache without initiating a memory transaction.’ (Horst Decl., ¶93.)

**j. Dependent Claim 13**

***Claim 13: The non-transitory machine-readable medium of claim 11, wherein the method further comprises sending the data to the system memory from the shared cache memory if a cache line including the data is evicted from the shared cache memory, when the cache line includes modified data.***

Sinharoy discloses Claim 13. (Horst Decl., ¶94.) As stated in Claim 7 *supra*, when the POWER7 processor performs cache injection operations and a cache line is evicted from the L3 region due to its dirty state, the data is evicted from the L3 cache and includes “modified data” that needs to be written back to the memory. (See *supra* Section VIII.A.1.h(Claim 7).)



(Sinharoy, Figure 10 (annotated).)

#### k. Dependent Claim 16

**Claim 16:** *The system of claim 15, 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 I/O module, including at least one dedicated entry to store a pending request corresponding to a posted transaction.*

Sinharoy anticipates or renders obvious Claim 16. (Horst Decl., ¶95.) (See *supra* VIII.A.1.d-e(Claims 3-4).)

#### l. Dependent Claim 17

**Claim 17:** *The system of claim 15, 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 anticipates or renders obvious Claim 17. (Horst Decl., ¶96.) (See *supra* Section VIII.A.1.i(Claim 12).)

2. **Ground 1b – Sinharoy in View of Xu Renders Obvious Claims 2, 10**

a. **Dependent Claim 2**

**Claim 2:** *The processor of claim 1, wherein the plurality of distributed portions of the caching agent are coupled via a ring interconnect.*

Sinharoy in view of Xu<sup>10</sup> renders obvious Claim 2. (Horst Decl., ¶¶97-101.)

The on-chip interconnect of POWER7 processor connects the eight cores with their associated L2 caches and local L3 regions, two memory controllers, and two I/O controllers. (See Sinharoy 1:22-23; Horst Decl., ¶99.) However, Sinharoy does not explicitly disclose that on-chip interconnect is a ring interconnect. Xu, in the same field of art as Sinharoy and in fact describing the same POWER7 chip as Sinharoy, discloses using on-chip ring interconnect. (Xu at TABLE 1.) For example, Xu discloses that IBM POWER7, Intel Sandy Bridge, Nehalem-EX, and Westmere-

---

<sup>10</sup> Xu is prior art under at least pre-AIA 35 U.S.C. § 102(a). Xu, “*Explorations of Optimal Core and Cache Placements for Chip Multiprocessor*” has a 2011 IEEE copyright notice. (EX1005.) Xu was published in *2011 NORCHIP* on or about November 14 to 15, 2011. See MacPherson Declaration Regarding Xu (EX1013.) Moreover, Gordon MacPherson confirmed that copies of Xu were made available no later than the “last date of the conference [November 15, 2011].” (EX1013.) Xu is also available for public download from the IEEE Xplore. (See <https://ieeexplore.ieee.org/document/6126728> (EX1014).)

EX processors rely on a ring interconnect. (*Id.*, 1) In Table 1, Xu expressly discloses that the on-chip interconnect connecting the portions of the caches for the IBM POWER7 processor chip is a ring interconnect. (*Id.*, 2.) A POSITA would have thus understood that the same POWER7 processor chip described in Sinharoy uses this ring interconnect as the on-chip interconnect. (Horst Decl., ¶98.)

TABLE I: Comparison of different Chip Multiprocessors.

| CMP                   | Cores    | Last Level Cache | Number of Transistors | Process | Die Size                                   | On-chip Interconnect  |
|-----------------------|----------|------------------|-----------------------|---------|--------------------------------------------|-----------------------|
| AMD MagnyCours [3]    | 12 (2×6) | 12MB (2×6MB)     | 1,808M (2×904M)       | 45nm    | 692mm <sup>2</sup> (2×346mm <sup>2</sup> ) | Crossbar              |
| IBM Power 7 [6]       | 8        | 32MB             | 1,200M                | 45nm    | 567mm <sup>2</sup>                         | Ring                  |
| Intel Nehalem [1]     | 4        | 8MB              | 731M                  | 45nm    | 263mm <sup>2</sup>                         | Crossbar              |
| Intel Westmere [1]    | 6        | 12MB             | 1,170M                | 32nm    | 239mm <sup>2</sup>                         | Crossbar              |
| Intel SandyBridge [4] | 4        | 8MB              | 995M                  | 32nm    | 216mm <sup>2</sup>                         | Ring                  |
| Intel Nehalem-EX [1]  | 8        | 24MB             | 2,300M                | 45nm    | 684mm <sup>2</sup>                         | Ring                  |
| Intel Westmere-EX [4] | 10       | 30MB             | 2,600M                | 32nm    | 513mm <sup>2</sup>                         | Ring                  |
| Sun SPARC T3 [5]      | 16       | 6MB              | 1,000M                | 40nm    | 377mm <sup>2</sup>                         | Hierarchical Crossbar |

(Xu, Table 1 (annotated).)

A POSITA would have understood, given the descriptions of the on-chip interconnect in Sinharoy, that Sinharoy includes a ring interconnect, and this is confirmed by Xu's express disclosure that the on-chip interconnect in the POWER7 is a ring interconnect, that a ring interconnect couples the centralized cache controllers for each local L3 regions (i.e., the plurality of distributed portions of the caching agent). (Horst Decl., ¶100.)

#### b. Dependent Claim 10

**Claim 10:** *The processor of claim 1, further comprising a ring interconnect to couple the plurality of cores and the shared cache memory via a caching agent.*

Sinharoy in view of Xu renders obvious Claim 10. (Horst Decl., ¶101.)

Sinharoy discloses an on-chip interconnect that is a “high-bandwidth intelligent on-chip coherence and data interconnect that physically occupies a horizontal equatorial trunk that runs across the center of the POWER7 chip.” (Sinharoy, 1:22-23.) The on-chip coherence interconnect “routes two sets (even and odd cache line) of coherence requests through the horizontal trunk, inward toward the even/odd arbitration logic located at the center of the chip.” (*Id.*, 1:23, Figure 1.) Xu further discloses on-chip interconnect in the POWER7 chip is a ring interconnect. (Xu, TABLE I, *see supra* Section VIII.A.2.a(claim2).) Because the on-chip ring interconnect runs across the center of the POWER7 chip, a POSITA would have readily understood that the ring on-chip interconnect connects the eight cores and the shared L3 cache, as well as the cache controllers for the L2 and L3 caches including RC and write machines as the caching agent:

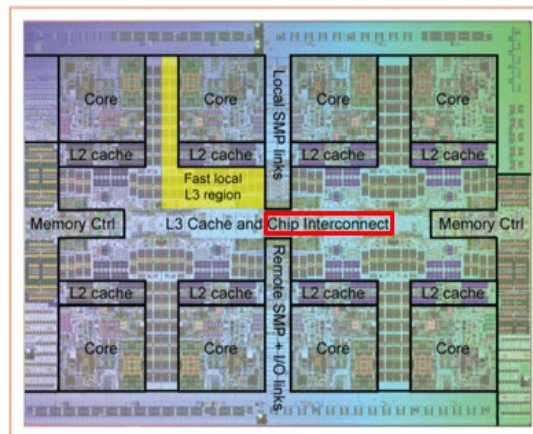


Figure 1

Die photo of the IBM POWER7 chip.

(Sinharoy, Figure 1 (annotated).)

3. **Ground 1c – Sinharoy in View of Tang Renders Obvious Claims 8-9 and 14**

a. **Dependent Claims 8, 14**

**Claim 8:** *The processor of claim 1, wherein the caching agent is to send a drop ownership indication to the IIO module for a cache line owned by the IIO module for the IO device responsive to receipt of a snoop request in the caching agent for the cache line from another agent.*

**Claim 14:** *The non-transitory machine-readable medium of claim 11, wherein the method further comprises sending a drop ownership indication to the IIO module for a cache line owned by the IIO module responsive to receipt of a snoop request in the caching agent for the cache line from another agent.*

Sinharoy in view of Tang<sup>11</sup> renders obvious Claims 8 and 14. (Horst Decl., ¶¶102-111.) Sinharoy discloses that the caching agent, comprising the centralized

---

<sup>11</sup> Tang, “DMA cache: Using on-chip storage to architecturally separate I/O data from CPU data for improving I/O performance”, has a 2009 IEEE copyright notice. (EX1004). As confirmed by Gordon MacPherson, it was published in *HPCA – 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture* on or about January 9 to 14, 2010 by the IEEE. (Gordon Declaration Regarding Tang (EX1011).) Moreover, Gordon MacPherson confirmed that copies of Tang were made available no later than the “last date of the conference [January 14, 2010].” (EX1011.) Tang has been available for public download from the IEEE Xplore since April 1, 2010.

cache controllers for the L2 and L3 caches including RC and write machines, receives snooping requests for cache lines from other agents. (Sinharoy, 1:22-23, 1:19-1:20.) In particular, Sinharoy discloses that the caching agent handles “storage accesses,” such as requests to write data from an I/O controller (and thereby owned by the I/O controller) into a cache line during supported cache injection operations. (*Id.*, 1:17-19, 1:22.) Sinharoy further discloses that storage accesses that “miss both the L2 cache and the 4-MB local L3 region are broadcast to the coherence fabric and snooped by the memory controller, 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.” (*Id.*, 1:19, 1:22-23.) That is, where a cache line owned by an I/O controller during a cache injection operation “miss” the L2 and L3 caches associated with a particular core, another core (i.e., another agent) and its associated remote L3 region can send a “snoop request” to the caching agent for that cache line.

Furthermore, Sinharoy discloses 13-cache-state protocol used for cache “coherent operations,” in the POWER7 processor. (*Id.*, 1:17, Table 2.) These include, for example, “TE” (formerly exclusive, now shared), “T” (formerly

---

(<https://ieeexplore.ieee.org/document/5416638> (EX1012).) Tang is prior art under at least pre-AIA 35 U.S.C. §§ 102(a), (b).

modified bias toward sharing, now shared), “I” (Invalid), “IG” (Invalid, cached scope-state), and “IN” (Invalid, scope predictor). (*Id.*) However, Sinharoy does not explicitly disclose the caching agent sends a “drop ownership indication” to the IIO module in receipt of the snoop request. Tang, in the same field of art as Sinharoy, discloses a multicore-processor system with shared cache with distributed portions. (Tang, §3.2.2.)<sup>12</sup> Moreover, Tang discloses cache-state transitions during cache coherence operations. (*Id.*, §3.1.3.) Tang discloses when cache B tries to read the cache line (i.e., data) from cache A, cache B<sup>13</sup> issues a

---

<sup>12</sup> For clarity and ease of reference, citations to Tang are to the Section numbers included therein, rather than the page numbers added by Petitioner to the exhibit.

<sup>13</sup> Cache B in Tang is a DMA cache. Tang discloses “we have refined the MOESI protocol [3] and the ESI protocol [16] for WB and WT policies respectively, i.e., the IO-MOESI (Figure 8) and the IO-ESI (Figure 9) protocols. Essentially, the refined cache coherence protocols for the DMA Cache are the same as the original MOESI/ESI protocols.” (Tang, § 3.1.3.) Tang further discloses the “only difference between the IO-MOESI/IO-ESI and the original protocols is exchanging the local sources and the probe sources of the state transitions (see Table 3) . . . Therefore, the DMA Cache can be viewed as a remote cache.” (*Id.*) The “DMA cache” cache-state transitions disclosed in Tang would be understood by a POSITA to apply equally to

“Probe-Write hit” to cache A, and when cache A receives the “Probe-Write hit”, it changes the data block with “Owned State” to “Invalid State”.” (See Tang, §3.1.3; Figure 8.) A POSITA would have understood that the cache controllers (i.e., caching agent) for cache A and B manage the “Probe-Write hit” (i.e., a snoop request) and cache-state transition. (Horst Decl., ¶106.) A POSITA would have understood that, in other words, the caching agent, in receipt of a snoop request, changes the cache state from “owned” to “invalid,” which involves “send[ing] a drop ownership indication.” (Horst Decl., ¶106.)

---

cache state transitions in other remote cache systems, such as those in Sinharoy. (Horst Decl., ¶107.)

|                          | Local Cache | Remote Cache | DMA Cache       |
|--------------------------|-------------|--------------|-----------------|
|                          | MOESI/ESI   | MOESI/ESI    | IO-MOESI/IO-ESI |
| <b>CPU Read Request</b>  | CPU-Read    | Probe-Read   | Probe-Read      |
| <b>CPU Write Request</b> | CPU-Write   | Probe-Write  | Probe-Write     |
| <b>DMA Read Request</b>  | Probe-Read  | Probe-Read   | DMA-Read        |
| <b>DMA Write Request</b> | Probe-Write | Probe-Write  | DMA-Write       |



Because Tang and Sinharoy are in the same field of art, a POSITA would have been motivated to look to the cache-state transitions during cache coherence operations in Tang to understand the cache states listed in Sinharoy. (Horst Decl., ¶106.) A POSITA would have understood that Tang provides details on similar cache states discussed in Sinharoy, which would be common across multi-core processors implementing the cache coherence protocols described in Sinharoy and Tang. (*Id.*) Moreover, a POSITA would recognized that Tang and Sinharoy are just

using slightly different terminology for the same well-known cache-state transitions, and a POSITA would have readily understood how the details on cache state transitions described in Tang apply to the corresponding cache state changes in Sinharoy. (*Id.*)

A POSITA would have understood that “TE” (formerly exclusive, now shared) and “T” (formerly modified bias toward sharing, now shared) in Sinharoy have substantially the same function as the “Owned” state in Tang. (Sinharoy, 1:17; Horst Decl., ¶105.) A POSITA would have understood that three invalid states (I “Invalid”, IG “Invalid, cached scope-state” and IN “Invalid, scope predictor”) have substantially the same function the “Invalid” state in Tang. (Sinharoy, 1:17; Horst Decl., ¶105.) As discussed above, where a cache line owned by an I/O controller during a cache injection operation “miss” the L2 and L3 caches associated with a particular core, another core (i.e., another agent) and its associated remote L3 region can send a “snoop request” to the caching agent for that cache line in Sinharoy. Thus, a POSITA would have understood that when the caching agent in Sinharoy receives the snoop request, in view of the cache-state transition in Tang, it would change the cache state for that cache line from T or TE to one of the “Invalid” states, which involves “send[ing] a drop ownership indication” to the IIO module. (Horst Decl., ¶¶109-110.)

**b. Dependent Claim 9**

**Claim 9:** *The processor of claim 8, wherein the IO device prefetched the cache line, and another agent is one of the plurality of cores.*

Sinharoy in view of Tang renders obvious Claim 9. (Horst Decl., ¶111.)

Where a cache line owned by an I/O controller during a cache injection operation “miss” the L2 and L3 caches associated with a particular core, another core (“another agent”) and its associated remote L3 region can send a “snoop request” to the caching agent for that cache line. (*See supra* Section VIII.A.3.a(Claim 8).) Furthermore, Sinharoy discloses “[p]refetches and some L2 store misses also access the L3 region via the core/L2 dispatch port.” (Sinharoy, 1:19.) A POSITA would have understood that such prefetches include an IO device connected to the I/O controller already prefetching that cache line, because the cache line will be invalidated due to the cache state change. (Horst Decl., ¶111.)

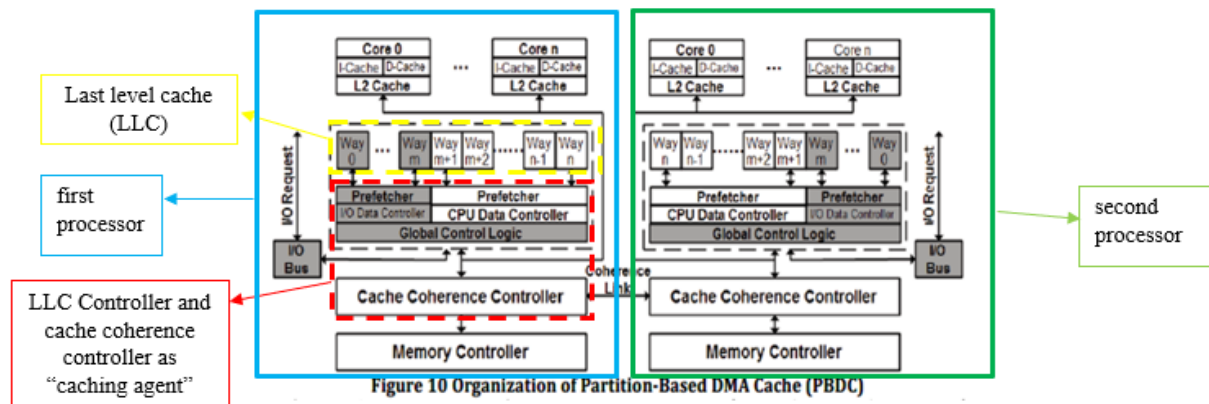
## **B. Grounds Based on Tang**

Tang describes a method and system to store I/O data into a shared last level cache (LLC). (Horst Decl., ¶114.)

Tang discloses systems and methods to directly inject or store I/O data into a cache, because “moving I/O data in/out memory” exhausts portion of memory bandwidth as well as accounts for a considerable part of I/O operation latency. (*See* Tang, §1.)

In one design, “Partition-Based DMA Cache (PBDC),” Tang discloses a multiprocessor system using the processor’s last level cache (LLC) as the

“dedicated DMA cache.” (Tang, §3.2.1, Abstract.) The PBDC design includes two multicore processors are connected via a coherence link:



(*Id.*, Fig. 10 (annotated); Horst Decl., ¶115.)

Each processor includes multiple cores, indicated as “Core 0 ... Core n,” an I/O bus, and a memory controller. (Tang, Fig. 10; Horst Decl., ¶115.) Each processor also includes a “caching agent” comprising a cache coherence controller and an LLC-Ctrler, which includes “Global Control Logic, IO Data Controller, CPU Data Controller, two Prefetchers, and Configuration Module.” (Tang, §3.2.2, Figs. 10-11.) The LLC Controller components manage “write policy, cache coherence protocol, replacement policy” for the LLC and the processor. (Tang, §§3.2.2; Horst Decl., ¶115.) In particular, Tang discloses that I/O data from an I/O device can be written directly into the DMA cache. (*See* Tang, §2; §3.2.2.)

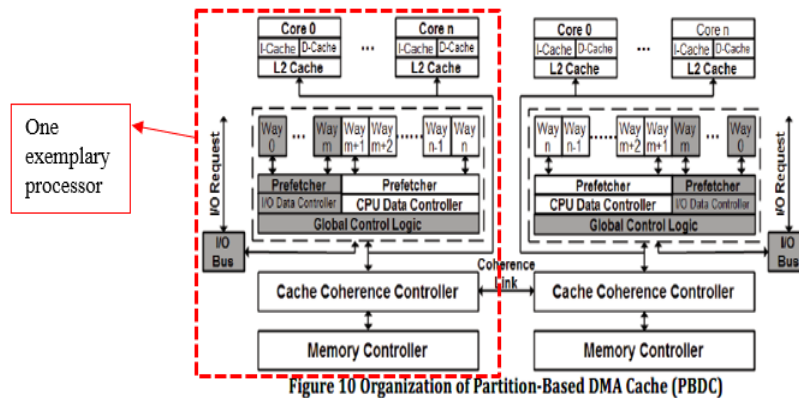
**1. Ground 2a – Tang Anticipates or Renders Obvious Claims 1, 5-9, 11-14, 15, 17**

**a. Independent Claim 1**

Tang anticipates or renders obvious Claim 1. (Horst Decl., ¶¶116-129.)

**Claim 1p: A processor comprising:**

Tang discloses Claim 1p. (*Id.*, ¶116.) Tang discloses a PBDC design that is a “multiprocessor” system with two processors:



(Tang, Figure 10 (annotated), §3.2.1).)

**Claim 1a: a plurality of cores,**

Tang discloses Claim 1a. (Horst Decl., ¶117.) Tang discloses that each processor in the PBDC multiprocessor system includes a plurality of cores—namely cores Core0 through Core n:

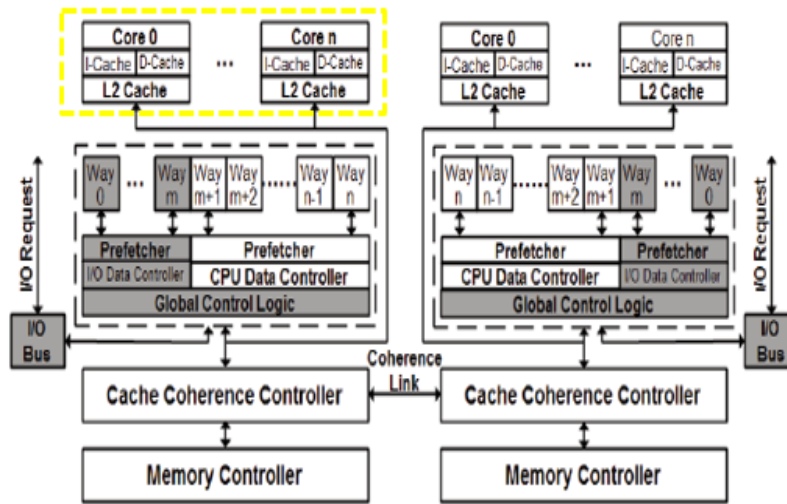


Figure 10 Organization of Partition-Based DMA Cache (PBDC)

(Tang, Figure 10 (annotated), §3.2.1.)

**Claim 1b: a shared cache memory,**

Tang discloses Claim 1b. (Horst Decl., ¶118.) Tang discloses a shared cache memory in the PBDC system, stating for example that the PBDC uses the “processor’s LLC” as a “dedicated DMA cache.” (Tang, §3.2.1, Fig. 10; Horst Decl., ¶118.). The LLC is a shared cache, and can be partitioned into several ways to store I/O data and CPU data, including I/O data ways (Way<sub>0</sub>~Way<sub>m</sub>) and the CPU data ways (Way<sub>m+1</sub>~Way<sub>n</sub>):

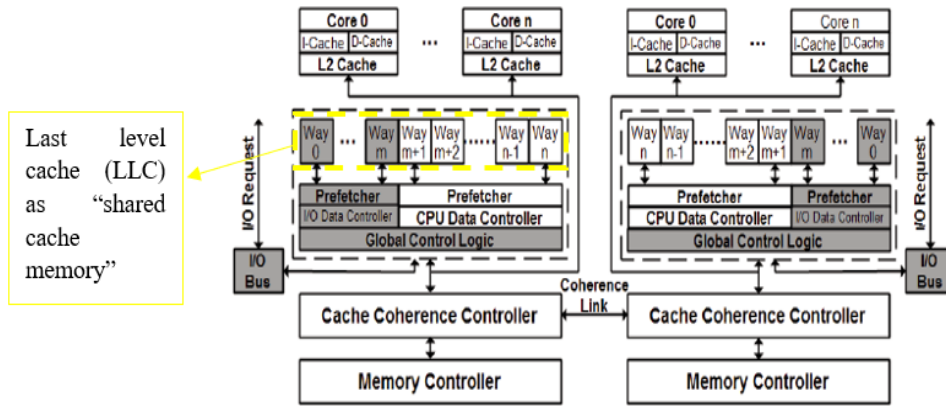


Figure 10 Organization of Partition-Based DMA Cache (PBDC)

(Tang, Figure 10 (annotated), Tang, §3.2.1, §6, Fig. 10.)

The partitioned LLC (Way<sub>0</sub>~Way<sub>n</sub>) use the same “cache coherence protocol” as the same design of Decoupled DMA cache (DDC). (Tang, §3.2.1, §3.1.2.) As shown above, the partitioned LLC ((Way<sub>0</sub>~Way<sub>n</sub>) is shared among Core 0 to Core n, and thus is a “shared cache memory.”(Horst Decl., ¶118.)

**Claim 1c:** *a memory controller to interface with a memory coupled to the processor,*

Tang discloses Claim 1c. (Horst Decl., ¶¶119-121.) Tang discloses the PBDC system includes a memory controller:

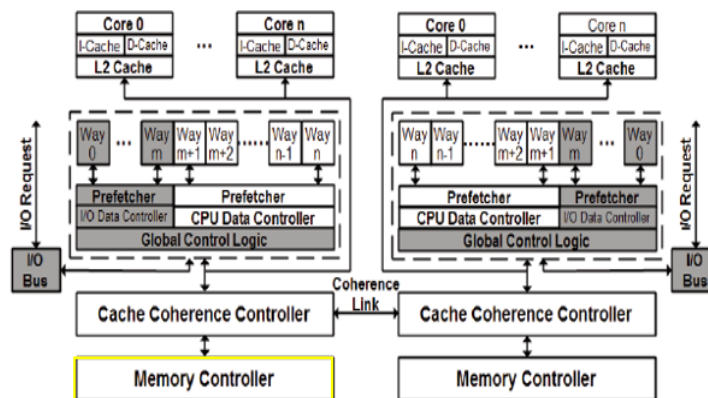
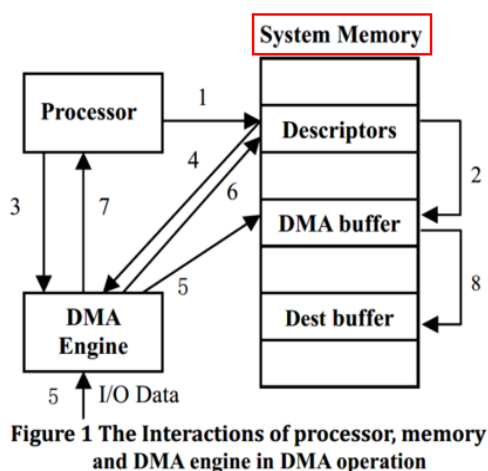


Figure 10 Organization of Partition-Based DMA Cache (PBDC)

(Tang, Figure 10 (annotated), Tang, §3.2.1 ).

Tang discloses that the PBDC system performs DMA operations, and that the components for the DMA operation of the PBDC includes a system memory coupled to the processor. (*Id.*, Abstract, §1, §2.1, §3.2, Figs. Figs. 1, 10; Horst Decl., ¶121.) In particular, Tang illustrates the system memory as being coupled to the processor in Figure 1:



(Tang, FIG. 1 (annotated).)

Furthermore, Tang discloses the memory controller interfaces with the DMA cache (i.e., the shared last level cache) and the system memory. (Tang, §2.1.) For example, Tang discloses DMA data transfer flow in the “DMA-Produce-CPU-Consume (DPCC)” direction. (*Id.*) When the CPU consumes the I/O data, any compulsory misses that take place and trigger “memory read requests” to the memory controller. (*Id.*) The “memory read requests” and “memory write

requests” are interactions with the system memory, illustrating how the memory controller interfaces with the system memory coupled to the processor:

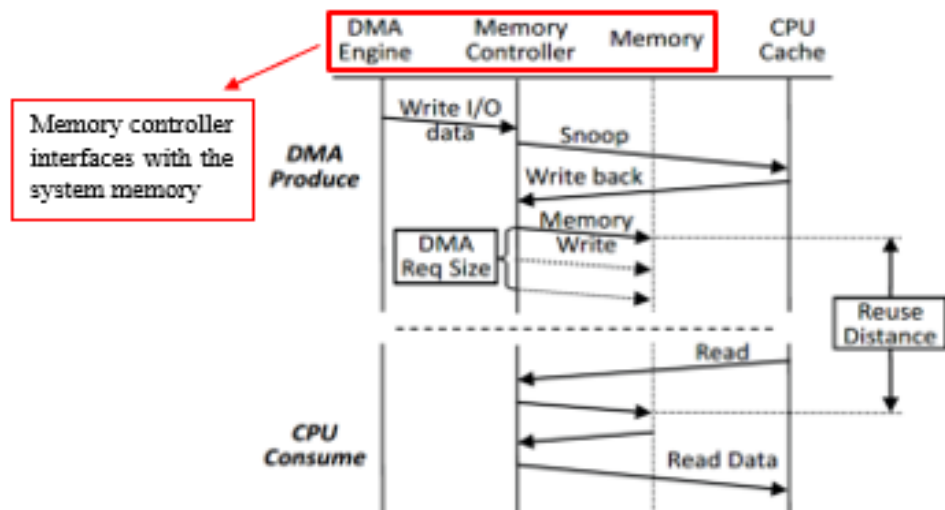


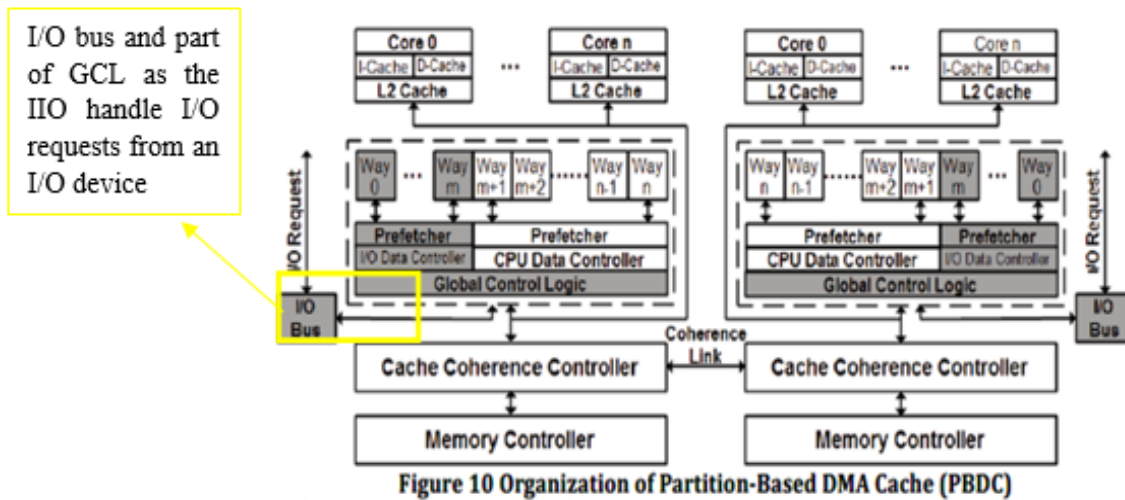
Figure 2 DMA data transfer flow in the “DMA-Producer-CPU-Consumer (DPCC)” direction

(Tang, Fig. 2 (annotated) (illustrating the Memory Controller interfacing with the DMA and the system “Memory”), §2.1.)

**Claim 1d:** *an integrated input/output (IIO) module to interface between the processor and an IO device coupled to the processor and*

Tang discloses Claim 1d. (Horst Decl., ¶122.) Tang discloses that “[n]ew I/O bus technologies, such as PCI-Express 2.0, AMD’s HyperTransport 3.0 and Intel’s QPI, can provide bus bandwidths of 10~20GB/s that are close to the bandwidth of a DDR2/3 DRAM memory system.” (Tang, §1.) Further, Tang discloses that part of Global Control Logic (GCL) “receives the read/write requests from the I/O bus and forward them to IOD-Ctrler.” (Tang, §3.2.2.) As such, the I/O bus and part of the GCL are an “integrated input/output (IIO) module,” as they are

integrated in the chip and interface between I/O devices connected to the processor chip and the processor itself. (Horst Decl., ¶122.) Tang discloses that such I/O devices are included in its system, as Tang discloses, for example, a DMA cache technique to use “dedicated on-chip storage to store I/O data.” (*Id.*, §1.) Tang defines I/O data as “a piece of data produced by I/O devices directly.” (*Id.*, §2.) Thus, Tang discloses that its system includes I/O devices. (Horst Decl., ¶122.)



(Tang, Fig. 10 (annotated) (illustrating the I/O bus handling I/O requests and interfacing with the multicore processor through the Global Control Logic), §3.2.1, §3.2.2.)

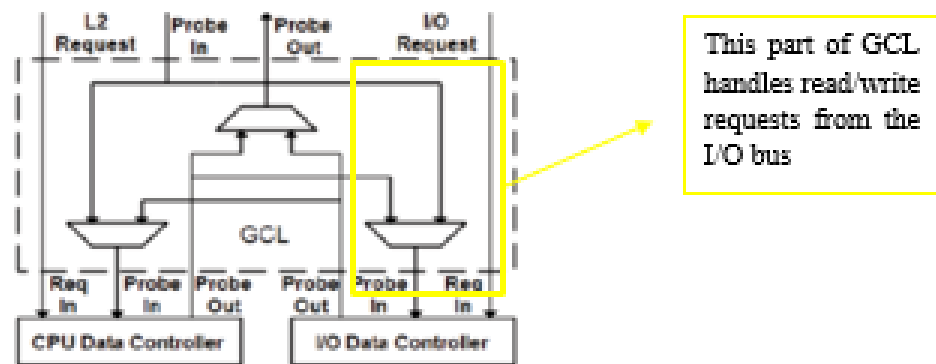


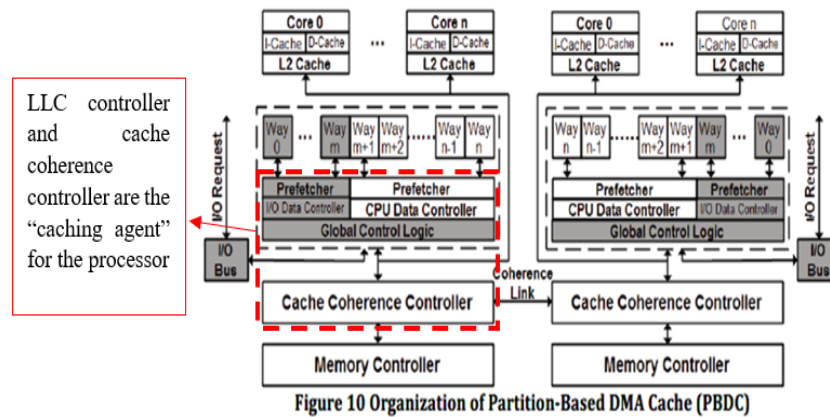
Figure 11 Global Control Logic (GCL) Delivering Requests

(Tang, §3.2.1 Figure 11 (annotated).)

**Claim 1e:** *a caching agent to perform cache coherency operations for the plurality of cores and the I/O module,*

Tang discloses Claim 1e. (Horst Decl., ¶¶123-124). Tang discloses that its system includes performing cache coherency operations. In particular, Tang discloses that in order to “address the I/O data coherence challenge, we have refined both the MOESI [3] and ESI [16] cache coherence protocols for the DMA cache.” (Tang, §1.) Tang further discloses that “the refined cache coherence protocols for the DMA Cache are the same as the original MOESI/ESI protocols.” (*Id.*, §3.1.3; §3.2.1 (using the same “write policy, cache coherence protocol, replacement policy” for both designs.) Tang, in Figure 10, discloses a Partition-Based DMA Cache (PBDC) system includes a cache coherence controller and a last level cache controller (“LLC-Ctrler”), which are a “caching agent” and “perform cache coherency operations.” (Horst Decl., ¶123, Tang, §3.2.2, Fig. 10.) in particular, the LLC-Ctrler includes IO Data Controller and CPU Data Controller,

which perform cache coherency operations for the plurality of cores Core 0 to Core n as they are, for example, responsible for “maintaining cache coherence” and “tagging data for the I/O data ways (Way<sub>0</sub>~Way<sub>m</sub>) and the CPU data ways (Way<sub>m+1</sub>~Way<sub>n</sub>).” (*Id.*, §3.2.1, §3.2.2.)



(*Id.*, Fig. 10 (annotated), §3.2.1.)

In addition, Tang discloses the LLC-Ctrler includes prefetchers, which are responsible “for fetching I/O data into DMA Cache.” (*Id.*, §3.1.1 (adopting the same cache coherence policy for the PBDC design in §3.2); Horst Decl., ¶124.) As shown in Figures 10 and 11 of Tang, the I/O bus interfaces with the LLC-Ctrler I/O, and requests are communicated from the I/O bus to the component of the LLC-Ctrler including the Global Control Logic and I/O Data Controller. (Tang, §3.2.1, §3.2.2, Figs. 10-11.) Part of this process may involve prefetches, and when the prefetchers fetch the I/O data from the I/O bus (part of the “I/O module”), the

cache coherence controller and LLC-Ctrler perform cache coherency operations for the IIO module, as well as the cores. (Horst Decl., ¶124; *see also* Tang, §3.1.3.)

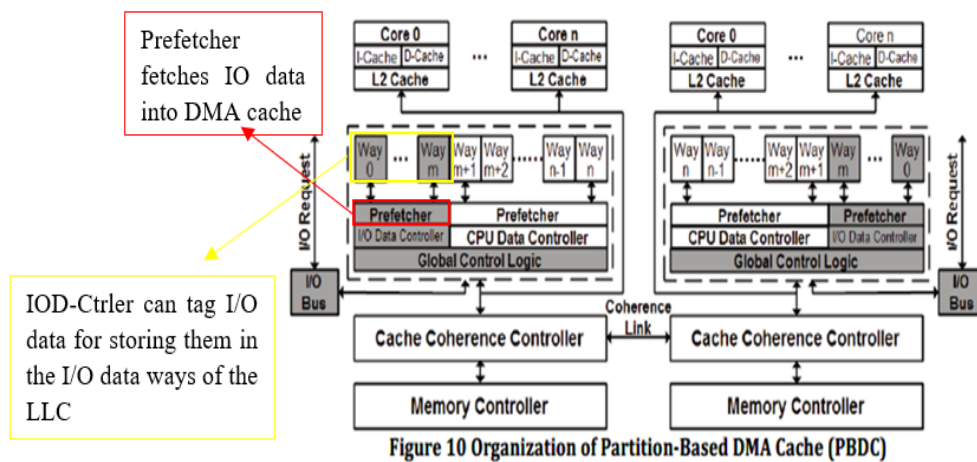
**Claim 1f:** *wherein the processor is to receive an allocation transaction from the IO device and directly store data of the allocation transaction into the shared cache memory,*

Tang discloses Claim 1f. (Horst Decl., ¶¶125-126). Tang discloses that its multicore processor can implement a “DMA cache technique, which uses dedicated on-chip storage to store I/O data.” (Tang, §1.) In the Decoupled DMA cache, Tang discloses a prefetcher that “is responsible for fetching I/O data into DMA cache [i.e., the LLC cache],” and discloses that this same cache coherence policies of the Decoupled DMA design can be implemented in the PBDC design. (*Id.*, §3.1.1, §3.2.1, Fig. 10 (illustrating “prefetchers” in the PBDC multiprocessor design), Fig. 11.) As such, the prefetchers in the PBDC design also “fetch” I/O data from I/O devices. (Horst Decl., ¶125). Fetching I/O data to be stored into the cache includes requests to write the I/O data into the DMA cache, which are allocation transactions. (Horst Decl., ¶125).

Tang discloses that the processor’s LLC with several ways (Way<sub>0</sub>-Way<sub>n</sub>) is the “DMA cache” in the PBDC design. (Tang, §3.2.1.) As discussed with respect to Claim 1b, the LLC with partitioned Way<sub>0</sub> to Way<sub>n</sub> is “shared cache memory.” *See supra* Claim 1b. When Tang’s processor receives a request to “fetch” I/O data to write it to the cache, such I/O data is stored directly into the LLC. (Tang, §3.2.1,

Tang, §3.2.2, Figs. 10-11.) Thus, when I/O data is stored into the LLC, Tang discloses “directly stor[ing] data of the allocation transaction into the shared cache memory,” where the allocation transaction can be requests to write from the I/O device. (Horst Decl., ¶125).

In addition, Tang discloses the LLC-Ctrler includes an IO Data Controller (IOD-Ctrler). (Tang, §3.2.2.) The IOD-Ctrler are responsible for “write policy, maintaining cache coherence, replacement policy, and tagging data for the I/O data ways (Way<sub>0</sub>-Way<sub>m</sub>).” (*Id.*) As part of this, the I/O data is directly stored in the shared LLC (shared cache memory):



(Tang, §3.2.1 Figure 10 (annotated); Horst Decl., ¶126.)

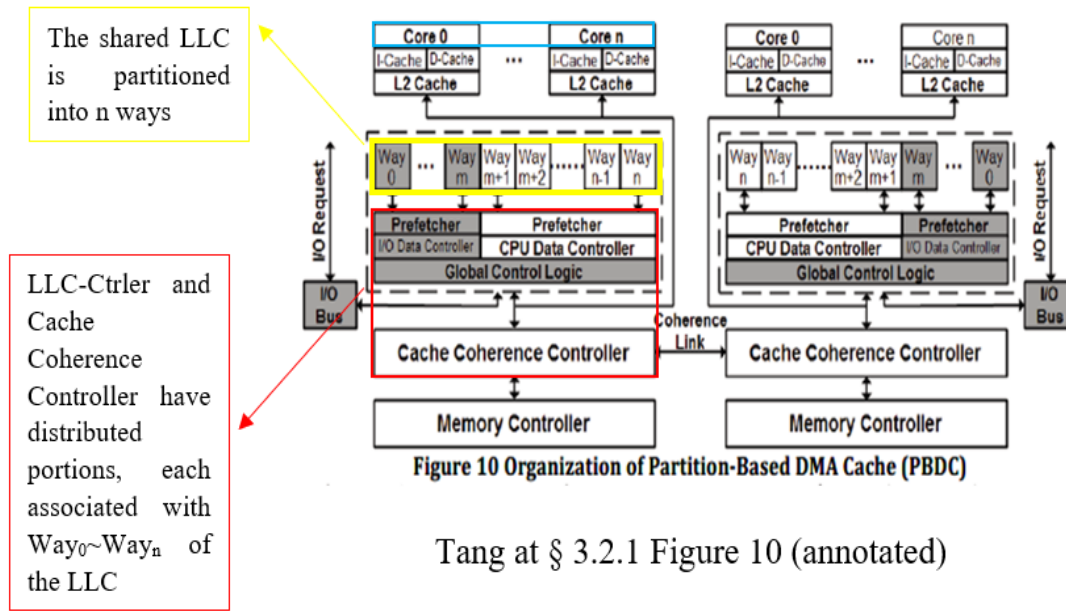
**Claim 1g:** *wherein the caching agent is a single caching agent for the processor and includes a plurality of distributed portions each associated with a corresponding one of the plurality of cores.*

Tang discloses Claim 1g. (Horst Decl., ¶¶127-129.) Tang discloses that the caching agent, comprised of the cache coherence controller and the LLC Controller

(LLC-Ctrler), is a single caching agent for the processor. (*See supra* Section VIII.B.1.a(Claim 1e).) In particular, the cache coherence controller and LLC-Ctrler form a logical single caching agent, and this single caching agent is further formed of a plurality of distributed portions each associated with one of the cores. (*Id.*, Horst Decl., ¶127.) Moreover, in the PBDC design, Tang discloses that the LLC is partitioned into “several ways” as the “dedicated DMA Cache.” (Tang, §3.2.1; *see also* §6 (“The partitioned cache approaches are proposed to reduce cache interferences among the threads in multicore platform.”))

Within the single caching agent, Tang discloses the LLC-Ctrler includes Global Control Logic (GCL), IO Data Controller (IOD-Ctrler), CPU Data Controller (CPUD-Ctrler), two Prefetchers and Configuration Module (CM). (Tang, §3.2.2.) The data ways Way<sub>0</sub> to Way<sub>n</sub> are partitioned LLC, and the caching agent therefore has distributed portions to handle the cache coherence for the n cores and the n-way LLC of the processor. (Tang, Fig.10; Horst Decl., ¶129.)

Because the LLC-Controller and the cache coherence controller (i.e., the caching agent) manage cache coherence protocol for the plurality of cores (core 0 to core n) and the n-way LLC (Way<sub>0</sub> to Way<sub>n</sub>), they have partitioned regions or units (i.e., distributed portions), each associated with a given core. (Horst Decl., ¶129.)



(Tang, Fig. 10 (annotated), §3.2.1.)

## b. Independent Claim 11

Tang anticipates or renders obvious Claim 11. (Horst Decl., ¶¶130-136.)

**Claim 11p:** *A non-transitory machine-readable medium having stored thereon instructions, which if performed by a machine cause the machine to perform a method comprising:*

Tang discloses Claim 11p. (*Id.*, ¶131.) In the PBDC design, Tang discloses a multiprocessor system, which has a non-transitory machine readable medium having stored thereon instructions, which when performed by the multiprocessor system allow it perform functions, including handling write policy, cache coherence protocol, replacement policy of the DMA cache. (Tang, §3.2.1, Fig. 10; Horst Decl., ¶131)

**Claim 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;***

Tang discloses Claim 11a. (Horst Decl., ¶¶131-132.) Tang discloses a multicore processor in a PBDC system that includes a plurality of cores, Core 0 through Core n. (Tang, Figure 10, §3.2.1; *see supra* Section VIII.B.1.a(Claim 1a).) As stated above, part of GCL and the I/O bus in Tang are the IIO Module, and they receive allocation transactions, including requests to read and write I/O data, from an IO device coupled to the multicore processor. (Tang, Figure 10, §3.2.1, §3.2.2; *see supra* Section VIII.B.1.a(Claims 1d, 1f); Horst Decl., ¶¶131-132.)

**Claim 11b: *forwarding the allocation transaction to a caching agent of the multicore processor to cause the data to be stored into a shared cache memory of the multicore processor, the caching agent shared by a plurality of cores of the multicore processor and the IIO module,***

Tang discloses Claim 11b. (Horst Decl., ¶¶133-134.) Tang discloses that the allocation transactions, or read and write requests from the I/O devices, are forwarded to a “caching agent” of the multicore processor, as the requests are sent to a “cache coherence controller” and “LLC-Ctrler” which includes Global Control Logic (GCL), IO Data Controller (IOD-Ctrler), CPU Data Controller (CPUD-Ctrler), two Prefetchers and Configuration Module (CM). (Tang, §3.2.2, Figs. 10-12; *see supra* VIII.B.1.a(Claims 1e).) In particular, Tang discloses that the GCL can “receive the read/write requests from the I/O bus and forward them to the IOD-Ctrler” and IOD-Ctrler can tag data for the I/O data ways. (Tang, §3.2.2.) Tang

further discloses that this caching agent performs cache coherency operations for the plurality of cores and the I/O bus device and portion of the GCL (“IIO module”), and is thus shared by them. (*See supra* VIII.B.1.a(Claim 1e); Horst Decl., ¶133.)

In response to receiving the read/write requests from an I/O device, the caching agent in Tang may fetch I/O data to store it in the DMA cache. (*Id.*; *see also supra* VIII.B.1.a(Claim 1f).), Again, fetching I/O data to be stored into the DMA cache includes allocation transactions, such as requests to store the I/O data into the DMA cache. (*Id.*, Horst Decl., ¶134). And, as explained above, the DMA cache is a “shared cache memory” of the multicore processor. (*See supra* VIII.B.1.a(Claim 1b).)

Thus, Tang discloses “forwarding the allocation transaction to a caching agent of the multicore processor to cause the data to be stored into a shared cache memory,” where the allocation transaction can be requests from the I/O device via the prefetcher. (Horst Decl., ¶134).

**Claim 11c: *wherein the caching agent is a single caching agent for the multicore processor and includes a plurality of distributed portions each associated with a corresponding one of the plurality of cores; and***

Tang discloses Claim 11c. (Horst Decl., ¶135.) (*See supra* Section VIII.B.1.a(Claim 1g).)

**Claim 11d:** *directly storing the data of the allocation transaction into the shared cache memory of the multicore processor without sending the data to a system memory coupled to the multicore processor.*

Tang discloses Claim 11d. (Horst Decl., ¶136.) For the reasons stated in Claim 1f *supra*, Tang discloses directly storing the data of the allocation transaction into the shared cache memory. (*See supra* Section VIII.B.1.a(Claims 1f, 1c.)) Furthermore, Tang discloses a write policy for the DMA cache, including write-back (WB) and write-through (WT) policies. (Tang, §3.1.2.) Tang discloses that when adopting WB policy, the I/O data written to the DMA Cache (“shared cache memory”) is initiated as Modified-state and “deferred to be updated into the memory until it is replaced.” (*Id.*) When adopting WT policy, the I/O data is “written to the DMA Cache as well as the memory.” (*Id.*) Thus, when the DMA Cache adopts WB policy and before the I/O data is replaced, the data of the allocation transaction is stored directly into the shared DMA cache without “sending the data to the system memory.” (Horst Decl., ¶136.)

**c. Independent Claim 15**

Tang anticipates or renders obvious Claim 15. (Horst Decl., ¶137-142.)

**Claim 15p:** *A system comprising:*

Tang discloses Claim 15. (*Id.*, ¶137.) In the PBDC design, Tang discloses a multiprocessor system that can handle write policy, cache coherence protocol, replacement policy of the DMA cache. (Tang, §3.2.1, §3.2.1, Figs. 10-12.)

**Claim 15a:** *a first multicore processor including a first plurality of cores, a first shared cache memory, 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, and a first caching agent to perform cache coherency operations for the first plurality of cores and the first IIO module,*

Tang discloses Claim 15a. (Horst Decl., ¶138.) (See *supra* VIII.B.1.a(Claims 1a-1e.))

**Claim 15b:** *wherein the first caching agent is to receive an allocation transaction from the first IIO module and directly store data of the allocation transaction into the first shared cache memory,*

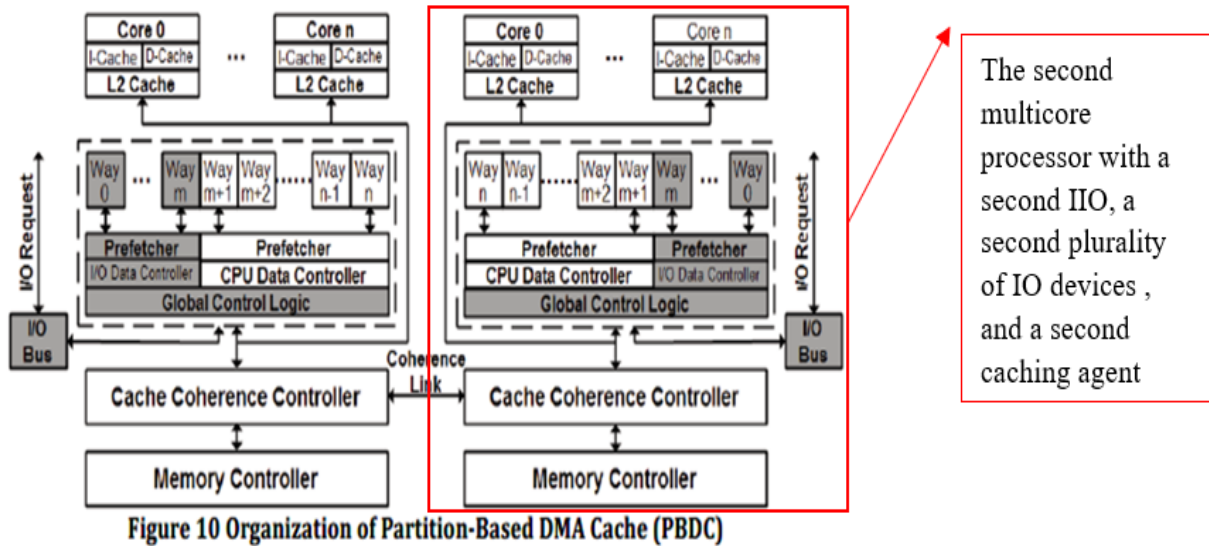
Tang discloses Claim 15b. (Horst Decl., ¶139.) (See *supra* VIII.B.1.a(Claim 1f.))

**Claim 15c:** *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,*

Tang discloses Claim 15c. (Horst Decl., ¶140.) (See *supra* Claim VIII.B.1.a(Claim 1g).)

**Claim 15d:** *a second multicore processor including a second plurality of cores, a second shared cache memory, a second IIO module to interface between the second multicore processor and a second plurality of IO devices coupled to the second multicore processor, and a second caching agent to perform cache coherency operations for the second plurality of cores and the second IIO module; and*

Tang discloses Claim 15d. (Horst Decl., ¶141.) Tang discloses a PDBC configuration in which the system includes a first multicore processor and a second multicore processor, each having cores Core 0 to Core n:



(Tang, Fig. 10 (annotated), §3.2.1.)

As shown, the first multicore processor and second multicore processor are connected through a coherence link. (*Id.*) This second multicore processor, just like the first multicore processor, includes a second plurality of cores, a second shared cache memory, a second IIO module to interface between the second multicore processor and a second plurality of IO devices coupled to the second multicore processor, and a second caching agent to perform cache coherency operations for the second plurality of cores and the second IIO module. (*See supra* VIII.B.1.a(Claims 1a-1e).)

**Claim 15e:** *a peripheral controller coupled to at least one of the first and second multicore processors.*

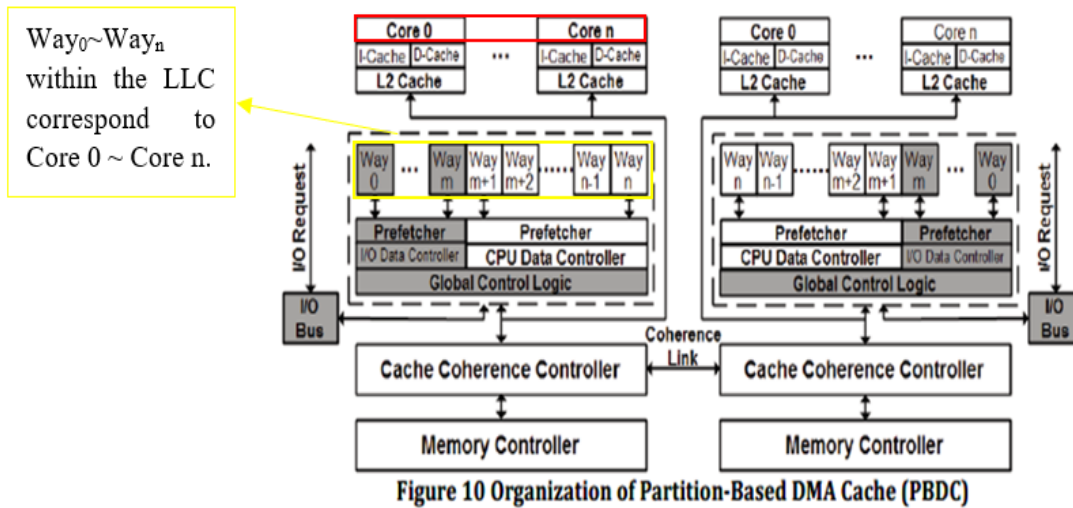
Tang discloses Claim 15e. (Horst Decl., ¶142.) Tang discloses each of the first and second multicore processors include a “peripheral controller” coupled to

the process, as each have I/O bus technologies such as “PCI-Express 2.0, AMD’s HyperTransport 3.0, and Intel’s QPI, that connect to peripheral devices having peripheral controllers. (Tang, §1.) For example, many peripheral devices with peripheral controllers are connected to processors via Tang’s disclosed PCE Express 2.0 bus technology. (Horst Decl., ¶142.)

**d. Dependent Claim 5**

***Claim 5: The processor 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 .***

Tang anticipates Claim 5. (Horst Decl., ¶143.) As stated in Claim 1b, Tang discloses that the shared DMA cache in the PBDC design uses “several ways of the processor’s last level cache (LLC) as the DMA cache..”(Tang, Abstract, §§1,3, 3.2.1. *See supra* VIII.B.1.a(Claim 1b).) Tang further discloses that the shared DMA cache that includes the “ways” of the processor LLC is formed of distributed portions, each associated with a corresponding one of the plurality of cores.” (Horst Decl., ¶143.) For example, the ways Way<sub>0</sub>~Way<sub>m</sub> and Way<sub>m+1</sub>~Way<sub>n</sub> are distributed portions of the LLC, and each is associated with a core of the plurality of cores Core 0 to Core n:



(Tang, Fig. 10 (annotated), §§3.2.1, 3.2.2.)

#### e. Dependent Claim 6

**Claim 6:** *The processor of claim 1, wherein the processor is to directly store the data of the allocation transaction into the shared cache memory without initiation of a memory transaction to the memory.*

Tang anticipates Claim 6. (Horst Decl., ¶¶144-145.) Tang discloses a write policy for the DMA cache (which can be used in both designs therein), including write-back (WB) and write-through (WT) policies. (Tang, §3.1.2.) Tang discloses that when adopting WB policy, the I/O data written to the DMA Cache is initiated as Modified-state and deferred to be updated into the memory until it is replaced. (*Id.*) When adopting WT policy, the I/O data is written to the DMA Cache as well as the memory. (*Id.*) Thus, when the DMA Cache adopts WB policy and before the I/O data is replaced, the data of the allocation transaction is stored directly into

the shared cache without touching the memory. (Horst Decl., ¶144.) For the PBDC design, Tang discloses that it uses the same WB and WT policies. (Tang, §3.2.1.)

Moreover, Tang discloses that for the DMA cache, when using WB policy, it can use the IO-MOESI protocol. (*Id.*, §3.1.3.) When the DMA Cache receives a DMA write request, it first broadcasts a Probe-Read request to other caches. (*Id.*) If the Probe-Read request hits in a CPU's LLC, the LLC changes the hit block's state to either Shared-state or Owned-state, and sends data to the DMA Cache. (*Id.*) After receiving the most recent data, the DMA Cache broadcasts a Probe-Write. (*Id.*) The LLCs, in which the Probe-Write hit, change the hit block with Shared-state or Owned-state to Invalid-state. (*Id.*) Then, the DMA Cache changes the block to Modified-state (using IO-MOESI with WB policy) or Exclusive-state (using IO-ESI with WT policy where the data is updated into memory as well). (*Id.*) The DMA write request process using IO-MOESI with WB policy does not initiate a memory transaction, while using IO-ESI with WT policy involves a memory transaction. (Horst Decl., ¶145.)

**f. Dependent Claim 7**

**Claim 7:** *The processor of claim 6, wherein the processor is to write the data of the allocation transaction to the memory responsive to eviction of a cache line including the data.*

Tang anticipates or renders obvious Claim 7. (Horst Decl., ¶¶146-147.) Tang discloses that the processor can implement allocation transactions, including write

requests, to write data to memory. (Tang, §§3.1.2, 3.2.1, 3.2.2; *see also supra* VIII.B.1.a(Claim 1f).) Claim 7 requires the cached I/O data to be later written to memory when cache line is evicted. A POSITA would understand that when the cache line was written, it was “dirty.” (Horst Decl., ¶146.) Later, when that cache line is evicted from the cache, it needs to be written back to memory unless it is known that the line will never be reused. (*Id.*) Tang teaches a standard MOESI cache protocol that includes writing I/O cache lines back to memory (as well as the IO-MOESI protocol in which the modified line is not written back). (Tang, §§3.1.2, 3.2.1, 3.2.2.) Tang discloses a write policy for the DMA cache, including write-back (WB) and write-through (WT) policies. (Tang, §3.1.2.) Tang discloses that when adopting a standard WT policy, the “I/O data is written to the DMA Cache as well as the memory.” (*Id.*)

A POSITA would understand that the dirty cache line would be written back to the memory in the process of being evicted from the cache. (Horst Decl., ¶147.) Thus, when the DMA cache adopts a standard WB policy, the I/O data of the allocation transaction is evicted from the DMA cache and written back into the system memory.

**g. Dependent Claim 8**

**Claim 8:** *The processor of claim 1, wherein the caching agent is to send a drop ownership indication to the HIO module for a cache line owned by the HIO module for the IO device responsive to receipt of a snoop request in the caching agent for the cache line from another agent.*

Tang anticipates or renders obvious Claim 8. (Horst Decl., ¶148.) For the reasons stated in Claim 8, when L2 cache (cache B) associated with a given core tries to read the cache line (i.e., data) from the shared LLC, the L2 cache for the core issues a “Probe-Write hit” to the LLC, where the “Probe-Write hit” is a “snoop request.” (*See supra* Section VIII.A.3.a(Claim 8).) Tang discloses that the DMA cache, when using WB policy, can use the IO-MOESI protocol. (*See* Tang, §3.1.3.) “After receiving the most recent data, the DMA Cache **broadcasts a Probe-Write. The LLCs, in which the Probe-Write hit, change the hit block with Shared-state or Owned-state to Invalid-state.** Then, the DMA cache changes the block to Modified-state (using IO-MOESI with WB policy)” (Tang, §3.1.3 (emphasis added)) In response to the snoop request, the caching agent (portion associated with the IIO module) changes the cache state of the cache line from “Owned” to “Invalid,” which involves “send[ing] a drop ownership indication to the IIO module.” (Tang, §3.1.3.) A POSITA would have understood that the Probe-Write hit is a “snoop request” that is managed and received by the cache controllers (caching agent) which causes the state to be changed to Invalid (a “drop ownership indication”). (Horst Decl., ¶148.)

#### **h. Dependent Claim 9**

**Claim 9:** *The processor of claim 8, wherein the IO device prefetched the cache line, and another agent is one of the plurality of cores.*

Tang anticipates or renders obvious Claim 9. (Horst Decl., ¶149.) As stated in Claim 8, when L2 cache (cache B) associated with a given core tries to read the cache line (i.e., data) from the shared LLC (the DMA cache), the core associated with the L2 cache is “another agent.” (*See supra* Section VIII.B.1.g(Claim 8).) Furthermore, Tang discloses “a DMA read request on the DMA cache can trigger a prefetching operation,” as “I/O data has very regular reference patterns.” (Tang, §3.1.5.) Where the cache line is owned by the IIO module, an IO device connected to the IIO already prefetched the cache line during the prefetching operation, because the cache line will be invalidated. (Horst Decl., ¶149.)

**i. Dependent Claim 12**

**Claim 12:** *The non-transitory machine-readable medium of claim 11, wherein the method further comprises 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 the system memory.*

Tang anticipates or renders obvious Claim 12. (Horst Decl., ¶150-152.) In the PBDC design, Tang discloses accessing data in the DMA cache (“shared cache memory”) via at least one of the plurality of cores, as it includes a “Last Level Cache Controller (LLC-Ctrler)”, which includes Global Control Logic (GCL), IO Data Controller (IOD-Ctrler), CPU Data Controller (CPUD-Ctrler), two Prefetchers and Configuration Module (CM). (Tang, §3.2.2.) I/O Data Controller (IOD-Ctrler) is responsible for write policy, maintaining cache coherence, replace policy and tagging data for the I/O data ways (Way<sub>0</sub> ~ Way<sub>m</sub>). (*Id.*) Again, storing the I/O data

into the I/O data ways of the shared DMA cache discloses storing data into the “shared cache memory.” (*See supra* Section VIII.B.1.a(Claim 1f).) Furthermore, when GCL receives the read request from the L2 cache, it forwards the request to CPUD-Ctrler. (*Id.*) An L2 cache corresponds to a specific core, thus, the read request from the L2 cache involves “accessing the data in the shared cache memory via at least one of the plurality of cores.” (Horst Decl., ¶151.) If the request hits the I/O data ways, an internal probe-read request is generated to IOD-Ctrler, sending the required I/O data to the L2 cache, which corresponds to a specific core. (Horst Decl., ¶151.) The process does not send a transaction to the system memory. (Horst Decl., ¶152.)

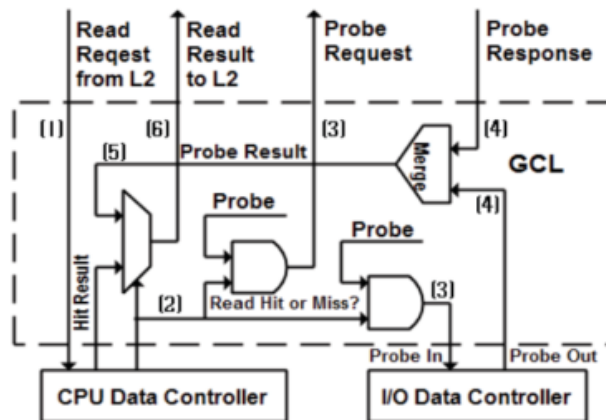


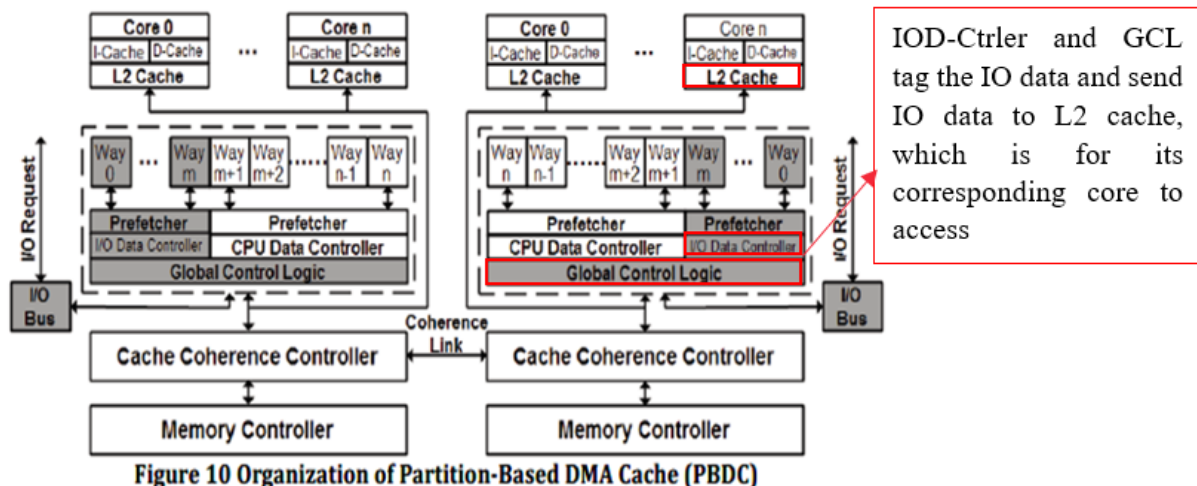
**Figure 12 The Process Flow of L2 Cache Read Request**

(Tang, Fig. 12 (annotated), §3.2.1.)

Furthermore, as discussed in Claim 6, when the DMA Cache adopts WB policy and before the I/O data is replaced, the data of the allocation transaction is

stored directly into the shared cache without sending the data to a system memory.

(See *supra* Section VIII.B.1.e(Claim 6).)



(Tang, Fig. 10 (annotated), §3.2.1.)

#### j. Dependent Claim 13

**Claim 13:** *The non-transitory machine-readable medium of claim 11, wherein the method further comprises sending the data to the system memory from the shared cache memory if a cache line including the data is evicted from the shared cache memory, when the cache line includes modified data.*

Tang anticipates or renders obvious Claim 13. (Horst Decl., ¶153.) Tang discloses a write policy for the DMA cache, including write-back (WB) and write-through (WT) policies. (Tang, §3.1.2.) Tang discloses that when adopting WB policy, the I/O data written to the DMA Cache is initiated as Modified-state and deferred to be updated into the memory until it is replaced. (*Id.*) Thus, when the DMA updates the system memory with such data, the data is evicted from the shared cache memory. (Horst Decl., ¶153.) Further, in Figure 8, Tang discloses

that when the DMA Cache receives a DMA write request, it first broadcasts a Probe-Read request to other caches, such as the CPU’s LLC, and the LLC can send data to the DMA cache. (*Id.*, §3.1.3.) After receiving the most recent data from the LLC, the DMA Cache changes the block to Modified-state (using IO-MOESI with WB policy). Data in the Modified state may be evicted as shown in Figure 8 with the arrow from the Modified state to the Invalid state. (Horst Decl., ¶153.) If that data is in the CPU portion of the shared cache (not the I/O portion), the eviction involves writing the modified data back to memory. (*Id.*)

### Table 3 Sources of State Transitions

|                          | Local Cache | Remote Cache | DMA Cache       |
|--------------------------|-------------|--------------|-----------------|
|                          | MOESI/ESI   | MOESI/ESI    | IO-MOESI/IO-ESI |
| <b>CPU Read Request</b>  | CPU-Read    | Probe-Read   | Probe-Read      |
| <b>CPU Write Request</b> | CPU-Write   | Probe-Write  | Probe-Write     |
| <b>DMA Read Request</b>  | Probe-Read  | Probe-Read   | DMA-Read        |
| <b>DMA Write Request</b> | Probe-Write | Probe-Write  | DMA-Write       |

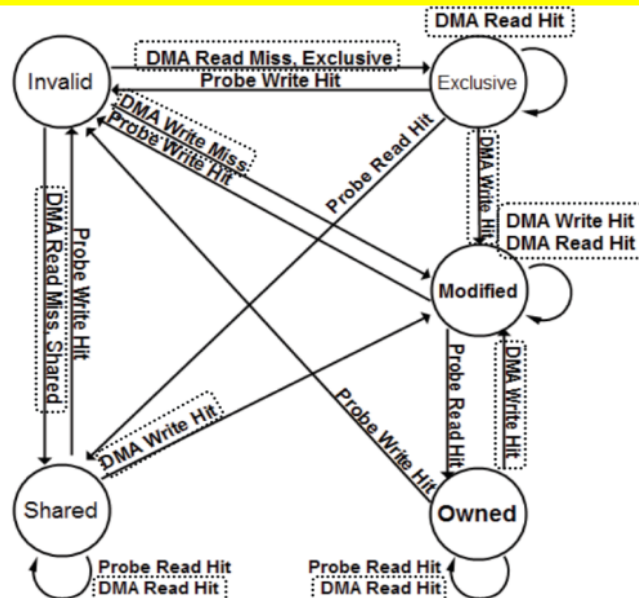


**Figure 8 IO-MOESI State Transitions for DMA Cache with Write-Back Policy<sup>1</sup>**

(Tang, §Table 3 and Figure 8 (annotated), §3.1.3)

**k. Dependent Claim 14**

**Claim 14:** *The non-transitory machine-readable medium of claim 11, wherein the method further comprises sending a drop ownership indication to the HIO module for a cache line owned by the HIO module responsive to receipt of a snoop request in the caching agent for the cache line from another agent.*

Tang discloses Claim 14. (Horst Decl., ¶154.) (See *supra* Section VIII.B.1.e(Claim 8).)

**l. Dependent Claim 17**

**Claim 17:** *The system of claim 15, 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.*

Tang discloses Claim 17. (Horst Decl., ¶155.) (See *supra* Section VIII.B.1.i(Claim 12).)

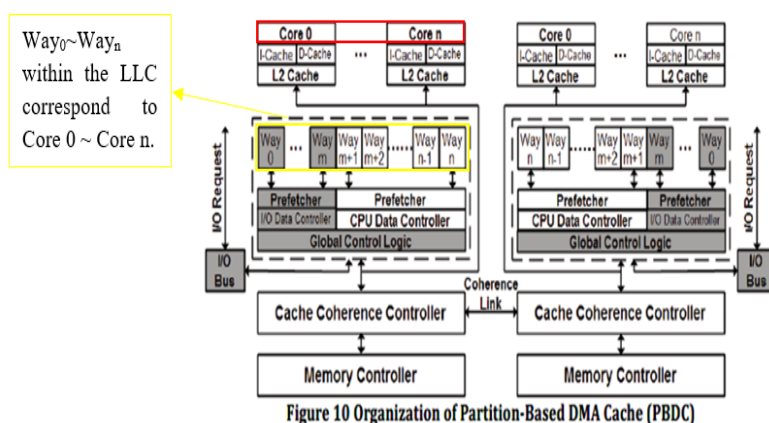
**2. Ground 2b – Tang in View of Harikumar Renders Obvious Claims 2, 3-4, 10, and 16**

**a. Dependent Claims 2, 10**

**Claim 2:** *The processor of claim 1, wherein the plurality of distributed portions of the caching agent are coupled via a ring interconnect.*

**Claim 10:** *The processor of claim 1, further comprising a ring interconnect to couple the plurality of cores and the shared cache memory via a caching agent.*

Tang in view of Harikumar<sup>14</sup> renders obvious Claims 2 and 10. (Horst Decl., ¶¶156-161.) Tang discloses a caching agent formed of a plurality of distributed portions, as Tang’s cache coherence controller and LLC-Ctrler form a logical single caching agent that includes distributed portions to handle the cahce coherence of the pliraity of cores and the distributed, partitioned “ways” of the LLC. (See *supra* Section VIII.B.1.a(Claim 1f).) Moreover, the LLC has distributed portions each associated with a corresponding core:



(Tang, Fig. 10 (annotated), §3.2.1; *see also supra* Section VIII.B.1.d(Claim 5.))

However, Tang does not specifically disclose that a “ring interconnect” coupled

<sup>14</sup> US2009/0164739 to Harikumar (“**Harikumar**”) (Ex. 1006), titled “Method, System and Apparatus for Handling Events for Partitions in a Socket with Sub-Socket Partitioning,” was filed November 7, 2008 and published June 25, 2009. Harikumar is prior art under at least pre-AIA 35 U.S.C. § 102(a).

the plurality of distributed portions of the caching agent, or the cores and shared cache memory via the caching agent.

Harikumar, in the same field of art as Tang, discloses a multicore processor with a distributed LLC. (Harikumar, [0031]). Harikumar explicitly discloses that cores and distributed LLC banks 408 in a multicore processor can be connected to each other within a socket by a first level interconnect 403, and that this first level interconnect 403 is an on-die ring interconnect. (*Id.*) Harikumar further discloses that this same ring interconnect also connects components such as fabric interfaces and a home agent. (*Id.*)

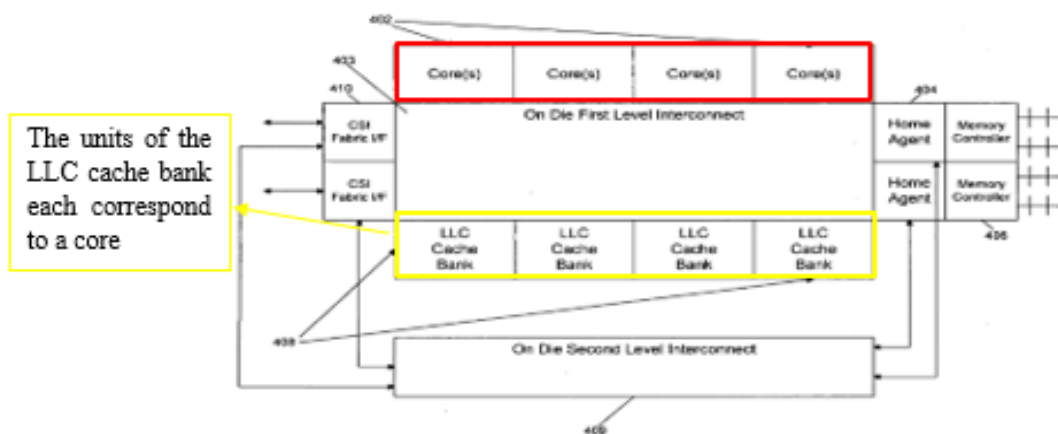


FIG. 4

(Harikumar, FIG. 4(annotated))

It would have been obvious to a POSITA to use the ring interconnect disclosed in Harikumar in the processor of Tang. (Horst Decl., ¶157.) Harikumar and Tang are analogous art; both disclose multicore processors with distributed LLCs, and each discloses coupling a plurality of cores and distributed LLC cache

regions to each other, as well coupling of other components. (*Compare* Tang, Figure 10 (annotated), *with* Harikumar Figure 4 (annotated); *see also* Tang, §3.2.2; Harikumar, [0031]; Horst Decl., ¶157.)

A POSITA would have recognized that using the ring interconnect disclosed in Harikumar in the processor of Tang would be the expected use of such well-known interconnect. (Horst Decl., ¶158.) A POSITA would have been motivated to use the ring interconnect disclosed in Harikumar in the processor of Tang to achieve the same purpose to connect the cores, the shared cache memory, and the caching agent. (*Id.*)

A POSITA would have a reasonable expectation of success, and such would have been a simple substitution, as the ring interconnect of Harikumar, when used in Tang's multicore processor, would continue to perform their same function: connecting different components of the processor and carrying information between the components. (Harikumar, [0013], [0031]; (Horst Decl., ¶¶159-161.)) Applying the teachings of Harikumar to Tang's multicore processor would have merely involved replacing one known technique of connecting the components of the processor (the cores, caching agent, the shared cache memory) to yield a predictable result (to provide interconnect protocol among the different components of the processor). (*Id.*)

**b. Dependent Claims 3, 4, 16**

**Claim 3:** *The processor 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.*

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

**Claim 16:** *The system of claim 15, 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.*

Tang in view of Harikumar renders obvious Claims 3-4 and 16. (Horst Decl., ¶¶162-168.)

Tang discloses methods and systems to directly store I/O data into a shared cache memory. (Tang, §3.2.2.) Further, in the PBDC design, Tang discloses the processor's last level cache (LLC) can be divided into "several ways" to store I/O data. (*Id.*, §3.2.2.) As stated in Claim 1e *supra*, the entire cache coherence controller and LLC-Controller (LLC-Ctrler) disclose the "caching agent." The LLC-Ctrler includes Global Control Logic (GCL), which is responsible for routing the read/write requests and the Probe-Read/Probe-Write requests to the right destinations. (*See supra* Section VIII.B.1.a(Claim 1e).)

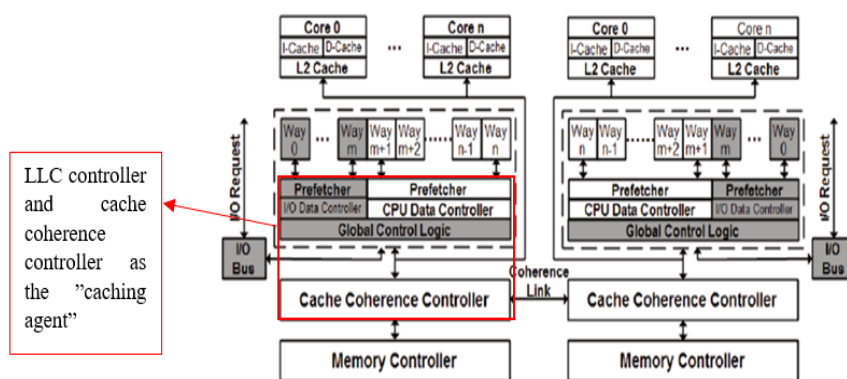


Figure 10 Organization of Partition-Based DMA Cache (PBDC)

(Tang §3.2.1 Figure 10 (annotated))

Harikumar explicitly discloses a caching agent may have a corresponding home agent, which includes a tracker and data buffers<sup>15</sup>. (*Id.* ¶[0014].) Harikumar teaches that a “tracker” is dedicated storage for memory requests from a particular device, which may include multiple entries associated with a given caching agent. (*Id.*) As illustrated in FIG. 3, each caching agent source 310A, ..., or 310D is associated with a tracker that has a predetermined number of tracker entries. (*Id.* ¶[0025].) The number of tracker entries is limited in size to the number of requests.

<sup>15</sup> The '895 describes the caching agent may include a table of requests, which may be a buffer or other temporary storage for holding incoming requests. (*See* '895 Patent 2:51-53.) The table of requests may include at least one dedicated resource for post transactions. (*Id.*, 2:53-55.) One example of a posted transaction is a write transaction. (*Id.*, 5:16-17.)

*Id.* In addition, the caching agents can each further include a credit counter, which maintains a count value representative of the number of unused tracker entries in the trackers. (*Id.*, [0029].) For example, when a new transaction is issued by a caching agent to its corresponding home agent, its credit counter is decremented. *Id.* If a transaction completes, the credit counter is incremented. (*Id.*) It would have been obvious to a POSITA that the tracker or credit counter in Harikumar stores pending requests and thus is the “table of requests.” (Horst Decl., ¶165.) When the transaction is a write transaction, the tracker or credit counter has a dedicated entry for a “posted transaction,” i.e., the write transaction. (*Id.*, ¶167.)

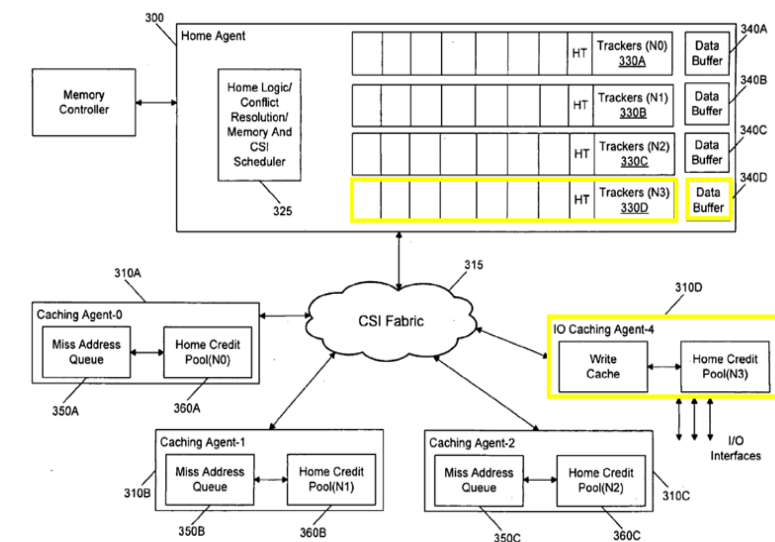


FIG. 3

(Harikumar FIG. 3 (annotated).)

It would have been obvious to a POSITA to use the tracker or credit counter disclosed in Harikumar in the LLC-Controller of Tang. (Horst Decl., ¶164.)

Harikumar and Tang are analogous art; both disclose multicore processors with

distributed LLCs, each distributed portion associated with a corresponding core. (*Compare* Tang, Figure 10 (annotated), *with* Harikumar Figure 4 (annotated) *supra*; *see also* Tang, §3.2.2; Harikumar, [0031]; Horst Decl., ¶164.)

A POSITA would have recognized that using the tracker or credit counter disclosed in Harikumar in the LLC-Controller of Tang (the caching agent) would be the expected use of such well-known temporary storage for holding incoming requests. (Horst Decl., ¶165.) A POSITA would have been motivated to use the tracker or credit counter taught in Harikumar in the LLC-Controller of Tang to achieve the same purpose to provide temporary storage of incoming requests and manage incoming read/write requests. (*Id.*)

A POSITA would have a reasonable expectation of success, and such would have been a simple substitution, as the tracker or credit counter of Harikumar, when used in Tang's LLC-Controller, would continue to perform their same function: storing incoming requests to manage and route read/write requests. (*Id.*) Applying the teachings of Harikumar to Tang's caching agent would have merely involved replacing one known technique of providing temporary data storage for incoming requests to yield a predictable result (for storing pending read/write requests for the caching agent). (*Id.*)

## **IX. TRIAL SHOULD BE INSTITUTED**

### **A. *Fintiv* Factors**

The Board should institute review because Petitioner has been diligent in filing the Petition, and the circumstances surrounding the parallel district court cases and International Trade Commission investigations do not warrant discretionary denial under § 314(a). *Sotera Wireless, Inc. v. Masimo Corp.*, IPR2020-01019, Paper 12 at 11-21 (PTAB Dec. 1, 2020) (precedential) (citing *Apple, Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (PTAB Mar 20, 2020) (precedential)). A *Fintiv* analysis shows that discretionary denial is not applicable here, as Petitioner has been dismissed from any pending litigation in district court and terminated from any pending ITC investigations. *See, e.g., Boss Corp. v. Koss Corp.*, IPR2021-00680, Paper 15 at 14 (PTAB Oct. 13, 2021). Moreover, any parallel district court and ITC proceedings were dismissed early-on, before *Markman* or any substantive decisions.

**B. 35 U.S.C. § 325(d)**

The Board should decline to exercise discretion under § 325(d). None of the combinations of references presented herein were considered by the Office during Examination.

Should Patent Owner raise any discretionary denial challenge to the Petition, Petitioner intends to respond to any counter-arguments Patent Owner may raise

**X. CONCLUSION**

Mercedes respectfully requests that a Trial be instituted. Claims 1-17 of the '895 should be cancelled as anticipated or obvious over the prior art.

Dated: August 23, 2023

Respectfully submitted,

Celine J. Crowson

---

Celine Jimenez Crowson (Reg. No. 40,357)

Joseph J. Raffetto (Reg. No. 66,218)

Scott Hughes (Reg. 68,385)

Ryan Stephenson (Reg. No. 76,608)

Nicholas Rotz (Reg. No. 75,959)

HOGAN LOVELLS US LLP

555 13th Street N.W.

Washington, D.C. 20004

Telephone: 202.637.5600

Facsimile: 202.637.5910

Helen Y. Trac (Reg. No. 62,250)

Hogan Lovells US LLP

Four Embarcadero Ctr, Suite 3500

San Francisco, CA 94111

Telephone: 415.374.2300

Facsimile: 415.374.2399

*Counsels for Petitioner Mercedes-Benz USA, LLC*

**APPENDIX A**  
**Table of Exhibits**

| <b>Exhibit No.</b> | <b>Description</b>                                                                                                                                        |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1001               | U.S. Patent No. 9,575,895 ( <i>Challenged Patent</i> )                                                                                                    |
| 1002               | File History for the '895 Patent                                                                                                                          |
| 1003               | <i>IBM POWER7 multicore server processor</i> to Sinharoy et al. (“ <b>Sinharoy</b> ”)                                                                     |
| 1004               | <i>DMA Cache: Using On-Chip Storage to Architecturally Separate I/O Data from CPU Data for Improving I/O Performance</i> to Tang et al. (“ <b>Tang</b> ”) |
| 1005               | <i>Explorations of Optimal Core and Cache Placements for Chip Multiprocessor</i> to Thomas Canhao Xu et al. (“ <b>Xu</b> ”)                               |
| 1006               | U.S. Patent Application Publication No. 2009/0164739 to Harikumar (“ <b>Harikumar</b> ”)                                                                  |
| 1007               | Declaration of Dr. Robert Horst                                                                                                                           |
| 1008               | Curriculum Vitae of Dr. Robert Horst                                                                                                                      |
| 1009               | Declaration of Gordon MacPherson, IEEE, Re Sinharoy                                                                                                       |
| 1010               | Sinharoy IEEE Xplore Page                                                                                                                                 |
| 1011               | Declaration of Gordon MacPherson, IEEE, Re Tang                                                                                                           |
| 1012               | Tang IEEE Xplore Page                                                                                                                                     |
| 1013               | Declaration of Gordon MacPherson, IEEE, Re Xu                                                                                                             |
| 1014               | Xu IEEE Xplore Page                                                                                                                                       |
| 1015               | File History for the '895 Patent Parent Application U.S. Application No. 13/324,053                                                                       |
| 1016               | <i>Architectural Implications of Cache Coherence Protocols with Network Applications on Chip Microprocessors</i> to Kyueun Yi et                          |

|  |            |
|--|------------|
|  | al. (“Yi”) |
|--|------------|

**APPENDIX B**  
**Certificate of Word Count**

I certify that, in accordance with 37 C.F.R. §42.24(d), the word count for the foregoing Petition totals 13,786. This is less than the permitted total under the 37 C.F.R. 42.24(a)(1)(i).

/s/ Joseph J. Raffetto  
Joseph J. Raffetto (Reg. No. 66,218)  
*One of the attorneys for Petitioner Mercedes-Benz  
USA, LLC*

**APPENDIX C**  
**Certificate of Service**

The undersigned certifies that, in accordance with 37 C.F.R. § 42.6(e) and 37 C.F.R. § 42.105, the foregoing were served on **August 23, 2023**, by filing this document through the Patent Trial and Appeal Board P-TACTS System and by Priority Mail Express® or by means at least as fast and reliable as Priority Mail Express® at the following address of record for the '895:

TROP, PRUNER & HU, P.C.  
PO Box 41790  
HOUSTON, TX 77241

/s/ Joseph J. Raffetto  
Joseph J. Raffetto (Reg. No. 66,218)  
*One of the attorneys for Petitioner*  
*Mercedes-Benz USA, LLC*