

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Appendix 228-02

The following list illustrates certain grounds for invalidity:

- B. Sinharoy et al., “IBM POWER7 Multicore Server Processor,” IBM Journal of Research and Development, vol. 55, no. 3, pp. 1-29 (2011) (“*Sinharoy*”), which was published and has been publicly available since at least April 29, 2011, is prior art to the '228 Patent under at least pre-AIA 35 U.S.C. § 102(a).
- U.S. Patent No. 7,805,575 to Agarwal et al. (“*Agarwal*”), which was filed on May 25, 2007, and issued on September 28, 2010, qualifies as prior art to the '228 Patent under at least pre-AIA §§ 102(a), (b), and (e).
- U.S. Patent Application Publication No. 2006/0041715 to Chrysos et al. (“*Chrysos*”), which was filed on May 28, 2004, and published on February 23, 2006, qualifies as prior art to the '228 Patent under at least pre-AIA §§ 102(a), (b), and (e).
- U.S. Patent Application Publication No. 2008/0086602 to Guthrie et al. (“*Guthrie*”), which was filed on October 9, 2006, and published on April 10, 2008, qualifies as prior art to the '228 Patent under at least pre-AIA §§ 102(a), (b), and (e).
- U.S. Patent Application Publication No. 2008/0126750 to Sistla (“*Sistla*”), which was filed on November 29, 2006, and published on May 29, 2008, qualifies as prior art to the '228 Patent under at least pre-AIA §§ 102 (a), (b), and (e).
- U.S. Patent Application Publication No. 2011/0153942 to Jain et al. (“*Jain*”), which was filed on December 21, 2009, and published on June 23, 2011, qualifies as prior art to the '228 Patent under at least pre-AIA §§ 102(a), (b), and (e).
- U.S. Patent Application Publication No. 2011/0219188 to Blumrich et al. (“*Blumrich*”), which was filed on January 18, 2011, and published on September 8, 2011, qualifies as prior art to the '228 Patent under at least pre-AIA §§ 102(a) and (e).
- T. Berg, “Maintaining I/O Data Coherence in Embedded Multicore Systems,” IEEE Micro, vol. 29, no. 3, pp. 10-19 (May/June 2009) (“*Berg*”), which was published and publicly available by at least June 19, 2009, and qualifies as prior art to the '228 Patent under at least pre-AIA §§ 102(a), (b), and (e).
- E. Herrero, “Power-Efficient Spilling Techniques for Chip Multiprocessors,” Proceedings of the 16th International Euro-Par Conference, pp. 256-67 (Aug. 31-Sept. 3, 2010) (“*Herrero*”), which was published and publicly available at least by September 2, 2010, qualifies as prior art to the '228 Patent under at least pre-AIA §§ 102(a), (b), and (e).

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

As illustrated below, to the extent that *Sinharoy* is found not to anticipate, expressly or inherently, one or more of the asserted claims of the '228 Patent, each of claims 1-2, 4-7, and 11-14 is invalid as obvious in view of *Sinharoy* alone or in combination with other prior art references, including, but not limited to, the prior art discussed below, the prior art identified in Defendant's Invalidity Contentions, and the prior art described in additional claim charts attached as Appendices 228-01 and 228-03 through 228-08. Exemplary invalidity grounds based on the *Sinharoy* reference include the following, although these examples are by no means limiting:

- Anticipation based on *Sinharoy* (claims 1-2, 4, 6, 7, 11-14);
- Obviousness based on *Sinharoy* in combination with *Agarwal*, *Berg*, *Jain*, and/or *Guthrie* (claims 1-2, 4, 6, 7, 11-14);
- Obviousness based on *Sinharoy* in combination with *Blumrich* (claim 2); and
- Obviousness based on *Sinharoy* in combination with *Agarwal*, *Jain*, *Sistla*, *Chrysos*, and/or *Herrero* (claim 5).

The citations included in the table below are exemplary only and are not intended to be limiting or exhaustive. Defendant reserves the right to cite additional disclosures and evidence in support of their positions as this case develops further. Additionally, citation to a particular figure in a reference should be understood to encompass the caption and description of the figure and any text relating to or discussing the figure. Similarly, citation to any text referring to a figure should be understood to include the figure as well. Additionally, Defendant may be basing the arguments below on one or more claim construction positions that Daedalus has not expressly disclosed but seems to apply in its Patent Local Rule 3-1 disclosures or Defendant's present understanding of Daedalus's view and application of claim scope. By applying the prior art in this way, Defendant does not concede its agreement with Daedalus's positions, nor do these Invalidity Contentions constitute any admission regarding claim scope. Nor is Defendant asserting any claim construction positions through these charts. Moreover, claim preambles generally are not limiting, and nothing in these contentions should be deemed an admission that the preamble in the asserted claims is limiting. As this case further develops, Defendant may amend or supplement these exemplary disclosures and any related contentions as appropriate, and in a manner consistent with all applicable rules governing this case. In particular, Defendant reserves the right to rely on (1) uncited portions of the identified prior art; (2) other prior art not identified herein; (3) references that show the state of the art (irrespective of whether such references themselves qualify as prior art to the asserted patents); (4) factual testimony from the inventors or authors of the prior art references; and/or (5) expert testimony, to provide context to or aid in understanding the prior art and the state of the art at the time of the alleged inventions. Defendant also reserves the right to amend or supplement these exemplary disclosures based on changes in Daedalus's positions as the case progresses, including changes to Daedalus's infringement theories, claim interpretations, or apparent claim constructions.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Descriptions of Exemplary Obviousness Combinations Based on *Sinharoy*

Sinharoy and each of the additional references cited below are directed to device architectures with similar structures and functions. They are analogous art because they come from the same field of endeavor as the '228 patent and are reasonably pertinent to the problems the '228 patent addresses. Because they are all directed to the same types of devices and have similar structures and functions, a person of ordinary skill in the art ("POSITA") would have appreciated that *Sinharoy* and the additional references cited provide disclosures that relate to and complement one another. Moreover, a POSITA would have found it obvious to combine their disclosures in the ways discussed below.

Exemplary Reasons for Implementing *Agarwal's*, *Berg's*, *Jain's*, and/or *Guthrie's* Disclosure of an Integrated Input/Output Module into the *Sinharoy's* Processor

To the extent *Sinharoy* does not disclose an integrated input-output module, a POSITA would have found it obvious to use an integrated input-output module in a multicore processor architecture with all the relevant features in the *Sinharoy* architecture. *Agarwal*, *Berg*, *Jain*, and *Guthrie* for example, teach a multicore processor with a distributed cache much like *Sinharoy*. See *Agarwal* at 16:35-45, 19:30-20:25, Fig. 6A-6B; *Guthrie* at [0026], [0031], [0037]; *Berg* at 1, 3; *Jain* at [0048], [0040]. *Agarwal's* multicore processor comprises a plurality of core-cache clusters in communication with each core, and each core-cache cluster includes a scalability agent (a distributed portion of a caching agent) that maintains memory coherency. See *Agarwal* at 1:32-39, 7:34-37, 11:53-58, 16:26-45, Fig. 2. Further, *Agarwal* explains that the architecture may contain an integrated input-output module to interface with an external device coupled to the multicore processor. See *id.* at 28:11-22, 29:5-7, Figs. 9A-9C. *Agarwal* thus demonstrates that the relevant features in the *Sinharoy* architecture are compatible with an integrated input-output module. See *id.* The same is true of *Berg*, *Jain* and *Guthrie*. See, e.g., *Guthrie* at [0026], [0031], [0037]; *Berg* at 1, 3; *Jain* at [0048], [0040]. A POSITA would have understood that incorporating an integrated input-output module would have had numerous performance benefits, including reducing latency, increasing scalability, and improving cache coherency. For instance, *Berg* explains that latency for read requests can be reduced because the coherence manager can return data from the L1 cache instead of from memory. *Berg* at 5.

Multiple rationales support the proposition that a POSITA would have found it obvious to employ an integrated input/output module in *Sinharoy's* basic architecture:

- A combination of prior art elements (*Sinharoy's* multi-core distributed caching architecture and *Agarwal's*, *Berg's*, *Jain's*, and/or *Guthrie's* integrated input/output module) according to known methods (forming on-die interconnects to connect multiple cores and caches in a multicore processor) to yield predictable results (a multicore processor with a physically scalable, logically simplistic, and high-bandwidth on-die interconnect);

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

- A simple substitution of one known element (*Sinharoy*'s integrated input-output controllers) for another (*Agarwal*'s, *Berg*'s, *Jain*'s, and/or *Guthrie*'s integrated input-output modules) to obtain predictable results (a multicore processor with a physically scalable, logically simplistic, and high-bandwidth on-die interconnect);
- A use of a known technique (*Agarwal*'s, *Berg*'s, *Jain*'s, and/or *Guthrie*'s use of an integrated input-output module) to improve similar devices (*Sinharoy*'s multi-core distributed caching architecture) in the same way (by decreasing the complexity of the hardware and software (routing and arbitration logic) used to operate a cache coherency protocol in a multicore processor).
- Applying a known technique (*Agarwal*'s, *Berg*'s, *Jain*'s, and/or *Guthrie*'s teaching of an integrated input-output module in an architecture like *Sinharoy*) to a known device (*Sinharoy*'s multi-core distributed caching architecture) ready for improvement to yield predictable results (a multicore processor with a physically scalable, logically simplistic, and high-bandwidth on-die interconnect); and
- A teaching, suggestion, or motivation in the prior art (e.g., the teachings in *Agarwal*'s, *Berg*'s, *Jain*'s, and/or *Guthrie*'s noted above) that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings (as described above) to arrive at the claimed invention.

Exemplary Reasons for Implementing *Blumrich*'s Table of Requests into *Sinharoy*'s Caching Agent

To the extent *Sinharoy* does not disclose a table of requests in the caching agent, a POSITA would have found it obvious to use a table of requests in the caching agent of *Blumrich*'s multicore processor architecture with all the relevant features in the *Sinharoy* architecture. *Blumrich* is analogous art to the '228 Patent. Like the '228 Patent, *Blumrich* discloses a "multiprocessor system," and discloses methods to "maintain cache coherence" See *Blumrich* at Abstract; [0124]. *Blumrich*'s "L2 caches serve as the point of coherence for all processors." *Id.* at [0119]. As part of "[c]oherence tracking," *Blumrich* uses a "request queue 302," which "buffers incoming read and write requests." *Id.* at [0128]. In one embodiment, this queue "is 16 entries deep," although *Blumrich* notes that "other request buffers might have more or less entries." *Id.* *Blumrich* teaches that, using this queue, "[t]he addresses of incoming requests are matched against all pending requests to determine ordering restrictions." *Id.*

A POSITA would have been motivated to implement *Sinharoy*'s "coherence fabric" using *Blumrich*'s request queues. Like *Sinharoy*, *Blumrich*'s disclosure also originated at IBM. See *Blumrich*, Cover Page. Each *Sinharoy* controller provides a "pool of operational resources." See *Sinharoy* at 19. Accordingly, all incoming requests to an L3 controller must be arbitrated and processed using this pool of resources. *Sinharoy*'s pool of resources includes "a pool of eight read machines," "a pool of 24 lightweight RC [read/claim] machines,"

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

and “a pool of ten write machines.” *Sinharoy* at 19. Read machines manage read hits; RC machines manage store misses and prefetches; and write machines manage data writes. *See id.* Write machines also perform core prefetch operations, L2 cast-out operations, and I/O cache injection operations. *See Sinharoy* at 17-20.

Because the pool of resources has limited entries (e.g., capable of handling eight outstanding reads, 24 outstanding store misses, and ten outstanding data writes), a POSITA would have understood the controller would beneficially use a storage element (e.g., queue or buffer) to track outstanding requests and avoid overflowing the limited pool of resources. A POSITA would have understood that either a queue or a buffer would have been an obvious solution for keeping track of outstanding requests. For example, *Sinharoy* suggests using queues to keep track of requests within the core. *See Sinharoy*, 9-10 (explaining the core includes buffers/queues to track outstanding requests, such as a 32-entry store-reorder-queue (“SRQ”) and a 32-entry load-reorder-queue (“LRQ”).

A POSITA would have considered similar, known options for *Sinharoy*’s coherence fabric (“caching agent”) to track outstanding requests, including *Blumrich*’s disclosure of a tracking queue for several reasons, including:

- A POSITA would have understood that a queue like the one *Blumrich* discloses would be beneficially used for tracking requests. For example, a queue enables ordering requests, allowing the address of all incoming requests to be matched against the outstanding requests. When a write request is followed by a read request for the same address, the two requests must be executed in order; otherwise, the read will not access the correct data. A POSITA thus would have understood that a caching agent (e.g., *Sinharoy*’s coherence fabric) requires a queue to store all pending requests to perform address-checking for an incoming request.
- Additionally, a POSITA would have reasonably expected to succeed in implementing *Sinharoy*’s coherence fabric by including a request queue, as *Blumrich* teaches. The combination would have involved nothing more than combining prior art elements arranged according to known methods to yield predictable results.
- A teaching, suggestion, or motivation in the prior art (e.g., the teachings in *Blumrich*) that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings (as described above) to arrive at the claimed invention.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Exemplary Reasons for Implementing *Agarwal's*, *Jain's*, *Sistla's*, *Chrysos's*, or *Herrero's* Disclosure of a Ring Connect into the *Sinharoy's* Processor

To the extent *Sinharoy* does not disclose a ring interconnect, a POSITA would have found it obvious to use a ring interconnect in a multicore processor architecture with all the relevant features in the *Sinharoy* architecture. *Agarwal*, *Jain*, *Sistla*, *Chrysos*, and *Herrero* for example, teach a multicore processor with a distributed cache much like *Sinharoy*. See *Agarwal* at 1:32-33, 3:7-24, 7:34-37, 11:53-58, 20:1-4; *Jain* at [0048], [0056], Figs. 3, 4; *Sistla* at [0001], Figs. 3-4; *Herrero* at Abstract; *Chrysos* at [0028], Fig. 3. *Sistla's* multicore processor comprises “a plurality of core-cache clusters 310-310 in communication with each other over an on-die interconnect 320,” and each of the core-cache cluster includes a scalability agent (a distributed portion of a caching agent) that maintains memory coherency. See *Sistla* at [0018], [0030]-[0031]. Further, *Sistla* explains the on-die interconnect in such an architecture may be a ring interconnect. See *id.* at [0030]. *Sistla* thus demonstrates that the relevant features in the architecture are compatible with a ring interconnect. See *id.* The same is true of *Agarwal*, *Jain*, *Herrero*, and *Chrysos*. See *Agarwal* at 3:45-47, 7:16-18, 11:53-12:2, 19:7-29, 20:63-67, Figs. 2, 6B; *Jain* at [0067]; *Herrero* at Abstract, pp. 2-3, 5-7; see *Chrysos* at [0025], [0037], [0045], Fig. 3.

A POSITA would have found it obvious to use a ring interconnect to link the cores, caches, and caching agent in an architecture like the *Sinharoy* architecture because using a ring interconnect like the ones *Agarwal*, *Jain*, *Sistla*, *Chrysos*, and *Herrero* disclose would have provided a high-performance scalable interconnect option over *Sinharoy's* apparent bus-style interconnect architecture with a centralized arbitration logic. For example, a ring interconnect provides a simple, fast, and efficient interconnection between cores and caches in a multicore processor. See *Marty*¹ at 24, 45; see also *Chrysos* at [0018]. Specifically, a ring interconnect provides higher bandwidth compared to a bus and supports a larger number of processor cores. See *Blake*² at Table 2. Additionally, a bus interconnect, like *Sinharoy's* interconnect design, requires a centralized arbitration logic to maintain cache coherency. See *Marty* at 46. To support cache coherency, however, such bus interconnect design needs complex control logic and deep buffering that negatively impacts performance. See *id.* 46-47. In contrast, a ring can use simpler, distributed arbitration logic. See *Marty* at 48. A ring interconnect can also increase reliability by resending data that becomes corrupted, which is important in multicore processors. See *id.*

A ring also provides other advantages over a crossbar-type interconnect. For example, unlike a ring, a crossbar suffers from physical scalability limitations because the number of wires required to form the crossbar grows exponentially with the number of nodes in the system. See *Marty* at 47. When scaled beyond a two-core system, a ring has a lower node degree than a crossbar, which decreases implementation complexity and reduces the length of wires required. See *Keckler*³ at 38. And, again, crossbars require sophisticated arbitration logic that a ring does not require. See *Blake* at Table 2. A crossbar also requires more physical die area compared to a rings.

¹ M.R. Marty, *Cache Coherence Techniques for Multicore Processors*, Ph.D. Dissertation, University of Wisconsin – Madison (2008).

² G. Blake et al., “A Survey of Multicore Processors,” *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 26-37 (Nov. 2009).

³ *Multicore Processors and Systems* (S.W. Keckler et al. eds., 2009).

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

See id. And a ring interconnect has few alternate paths between nodes, allowing for data saturation at lower network throughputs. *See Keckler* at 38. Additionally, crossbars face scalability limitations due to their need for centralized arbitration logic, whereas rings do not suffer from the same problem. *See Marty* at 46.

As *Agarwal's*, *Jain's*, *Sistla's*, *Herrero's* and *Chrysos's* teachings demonstrate, a POSITA would have understood that replacing *Sinharoy's* interconnect topology with a ring topology like the one *Agarwal*, *Jain*, *Sistla*, *Chrysos*, and *Herrero* disclose would have been a simple substitution of one element for another, such that doing so would have been compatible with the basic structure and design of the *Sinharoy* architecture. *See Sistla* at [0030]. And because *Agarwal*, *Jain*, *Sistla*, *Chrysos*, and *Herrero* employ architectures analogous to *Sinharoy's* architecture (i.e., a multicore processor having a distributed shared cache and caching agent configuration), a POSITA would have reasonably expected to succeed in making such a substitution. *Compare Sinharoy* at Fig. 1, with *Sistla* at Figs. 3-4 and *Chrysos* at Fig. 3. A POSITA thus would have found it obvious to implement *Sinharoy's* basic architecture using a ring interconnect topology as an alternative to *Sinharoy's* bus-style interconnect. *See, e.g., Marty* at 45-48; *Chrysos* at [0018]; *Blake* at Table 2; *Kecker* at 38.

Multiple rationales support the proposition that a POSITA would have found it obvious to employ a ring interconnection in *Sinharoy's* basic architecture:

- A combination of prior art elements (*Sinharoy's* multi-core distributed caching architecture and *Agarwal's*, *Jain's*, *Sistla's*, *Herrero's* and *Chrysos's* ring interconnect) according to known methods (forming on-die interconnects to connect multiple cores and caches in a multicore processor) to yield predictable results (a multicore processor with a physically scalable, logically simplistic, and high-bandwidth on-die interconnect);
- A simple substitution of one known element (*Sinharoy's* bus-style interconnect with a centralized arbitration logic) for another (*Agarwal's*, *Jain's*, *Sistla's*, *Herrero's*, and *Chrysos's* ring interconnect) to obtain predictable results (a multicore processor with a physically scalable, logically simplistic, and high-bandwidth on-die interconnect);
- A use of a known technique (*Agarwal's*, *Jain's*, *Sistla's*, *Herrero's*, and *Chrysos's* use of a ring interconnect) to improve similar devices (*Sinharoy's* multi-core distributed caching architecture) in the same way (by decreasing the complexity of the hardware (wiring and routers) and software (routing and arbitration logic) used to operate a cache coherency protocol in a multicore processor);
- Applying a known technique (*Agarwal's*, *Jain's*, *Sistla's*, *Herrero's*, and *Chrysos's* teaching to use a ring interconnect in an architecture like *Sinharoy*) to a known device (*Sinharoy's* multi-core distributed caching architecture) ready for

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

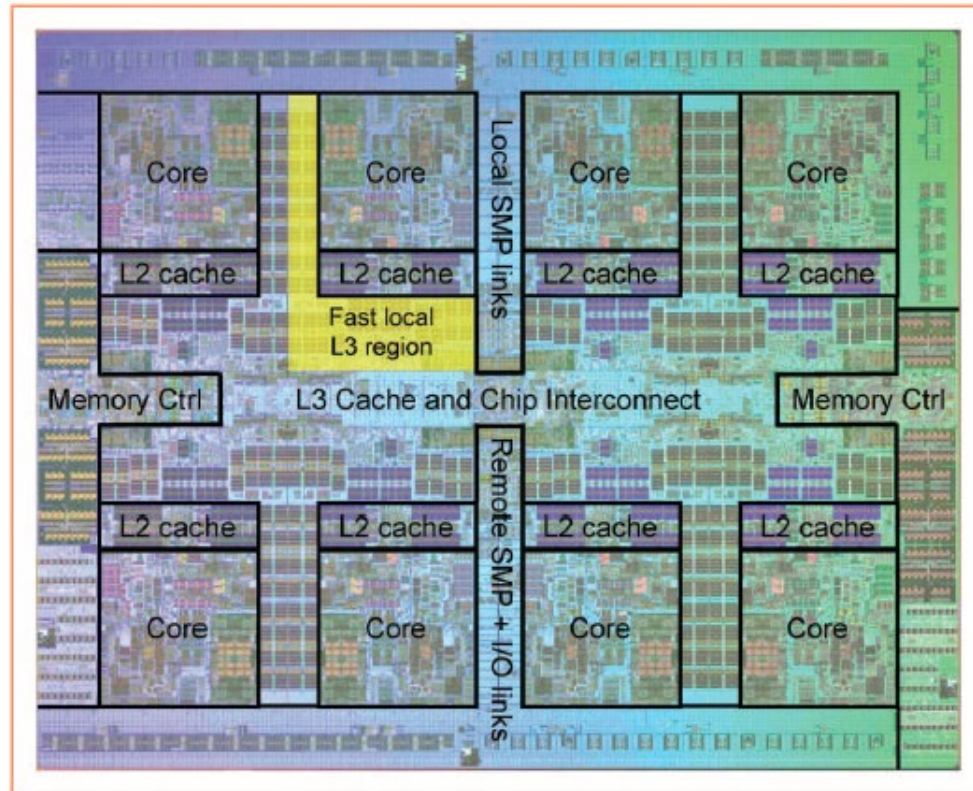
improvement to yield predictable results (a multicore processor with a physically scalable, logically simplistic, and high-bandwidth on-die interconnect); and

- A teaching, suggestion, or motivation in the prior art (e.g., the teachings in *Agarwal*, *Jain*, *Sistla*, *Herrero*, *Chrysos*, *Marty*, *Blake*, and *Keckler* noted above) that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings (as described above) to arrive at the claimed invention.

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[1p] An apparatus comprising:	<i>Sinharoy</i> discloses or suggests “an apparatus” <i>Sinharoy</i> discloses a multicore server processor. <i>See Sinharoy</i> at 1. <i>Sinharoy</i> also discloses a server system with up to 32 multicore server processors. <i>See id.</i> at 23-25, Figs. 11-13. <i>Sinharoy</i> discloses the following at 1: On the chip is an eight-core processor, with each core capable of four-way simultaneous multithreaded operation. Fabricated in IBM’s 45-nm silicon-on-insulator (SOI) technology with 11 levels of metal, the chip contains more than one billion transistors. The processor core and caches are significantly enhanced to boost the performance of both single-threaded response-time-oriented, as well as multithreaded, throughput-oriented applications. The memory subsystem contains three levels of on-chip cache, with SOI embedded dynamic random access memory (DRAM) devices used as the last level of cache. A new memory interface using buffered double-data-rate-three DRAM and improvements in reliability, availability, and serviceability are discussed.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

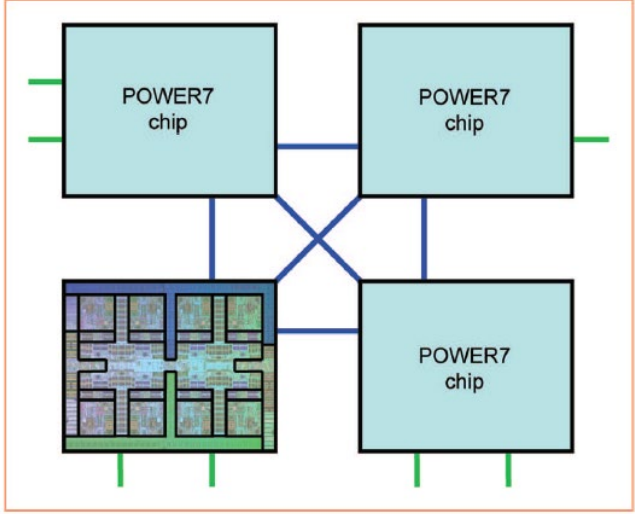
Sinharoy discloses the following at Fig. 1:



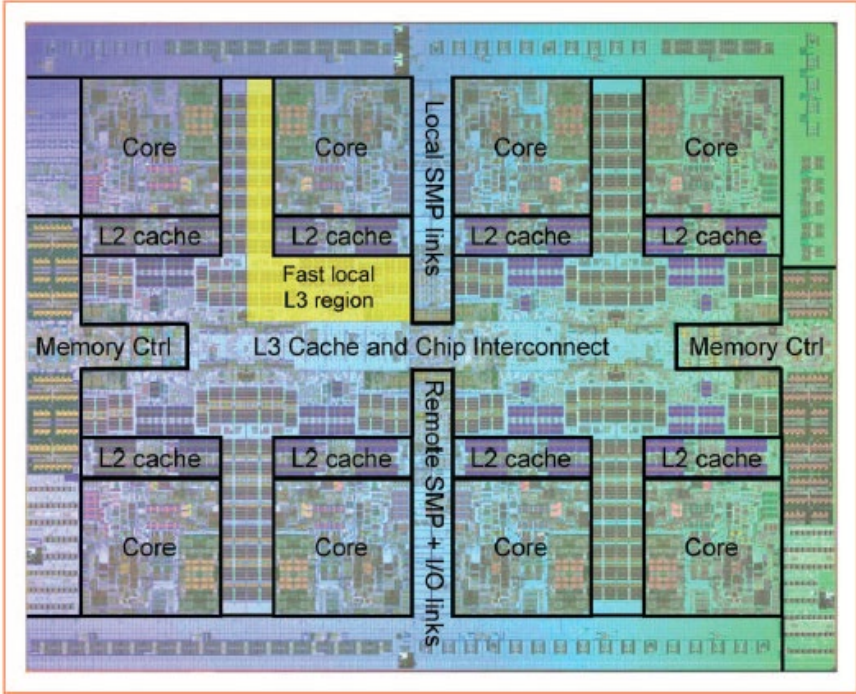
Sinharoy discloses the following at 23:

For standard commercial systems, up to 32 POWER7 chips may be connected to form a single 256-way SMP system. Figure 11 depicts a first-level nodal structure, which combines up to four POWER7 chips. Each chip has four 10-B/s on-node SMP links associated with the vertical spine that emanates from the center of the chip toward the top edge.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following, at Fig. 11: 
[1a] a multicore processor including a plurality of cores,	<i>Sinharoy</i> discloses or suggests “a multicore processor including a plurality of cores.” <i>Sinharoy</i> discloses a multicore server processor that includes eight cores. <i>See Sinharoy</i> at 1-2, Fig. 1. <i>Sinharoy</i> discloses the following at 1: On the chip is an eight-core processor, with each core capable of four-way simultaneous multithreaded operation. <i>Sinharoy</i> discloses the following at 2: Figure 1 shows the POWER7 chip, which has eight processor cores, each with 12 execution units capable of simultaneously running four threads.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following at Fig. 1:  The image is a die photo of the IBM POWER7 chip, showing its internal architecture. It features eight cores arranged in a 2x4 grid. Each core is connected to a local L2 cache. A central L3 Cache and Chip Interconnect connects all L2 caches. Memory controllers are located on the left and right sides. Local SMP links connect the cores within a node, while Remote SMP + I/O links connect the nodes. A Fast local L3 region is highlighted in yellow. Figure 1 Die photo of the IBM POWER7 chip. <i>Sinharoy</i> further discloses the following at 2: The eight cores on the processor chip can provide 32 concurrently executing threads (8× more than POWER6), yielding very high throughput per socket.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>										
[1b] a shared cache memory,	<i>Sinharoy</i> discloses or suggests “a shared cache memory.” <i>Sinharoy</i> discloses a shared 32-MB L3 cache within the multicore server processor. <i>See Sinharoy</i> at 2, 17-21, Table 1. <i>Sinharoy</i> discloses the following at 2 (citations omitted): eDRAM is a key technology innovation on the POWER7 chip. This allows the large 32-MB-shared-L3 cache to be on-chip, which provides several advantages such as improved L3 latency by elimination of off-chip drivers and receivers and wire-length reduction, improved L3 bandwidth per core (on-chip interconnects provide each core with 32-byte buses to and from the L3), and less area and power, compared with SRAM. <i>Sinharoy</i> discloses the following at Table 1: Table 1 Comparison of POWER6 and POWER7 cache hierarchies. <table> <tr> <th><i>POWER6 (assuming 5-GHz core)</i></th><th><i>POWER7 (assuming 4-GHz core)</i></th></tr> <tr> <td>32 KB store-through L1 D-cache 0.5ns latency, 192 GB/s private</td><td>32 KB store-through L1 D-cache 0.5ns latency, 192 GB/s private</td></tr> <tr> <td>64 KB store-through L1 D-cache 0.8ns latency, 80 GB/s private</td><td>256 KB store-in L2 cache 2.0-ns latency, 256 GB/s private</td></tr> <tr> <td>4 MB store-in L2 cache ~5.0-ns latency, 160 GB/s private</td><td>4 MB partial victim local L3 region ~6.0-ns latency, 128 GB/s private</td></tr> <tr> <td>32 MB victim L3 cache ~35-ns latency, 80 GB/s shared by 2</td><td>32 MB adaptive victim L3 cache ~30-ns latency, 512 GB/s shared by 8</td></tr> </table>	<i>POWER6 (assuming 5-GHz core)</i>	<i>POWER7 (assuming 4-GHz core)</i>	32 KB store-through L1 D-cache 0.5ns latency, 192 GB/s private	32 KB store-through L1 D-cache 0.5ns latency, 192 GB/s private	64 KB store-through L1 D-cache 0.8ns latency, 80 GB/s private	256 KB store-in L2 cache 2.0-ns latency, 256 GB/s private	4 MB store-in L2 cache ~5.0-ns latency, 160 GB/s private	4 MB partial victim local L3 region ~6.0-ns latency, 128 GB/s private	32 MB victim L3 cache ~35-ns latency, 80 GB/s shared by 2	32 MB adaptive victim L3 cache ~30-ns latency, 512 GB/s shared by 8
<i>POWER6 (assuming 5-GHz core)</i>	<i>POWER7 (assuming 4-GHz core)</i>										
32 KB store-through L1 D-cache 0.5ns latency, 192 GB/s private	32 KB store-through L1 D-cache 0.5ns latency, 192 GB/s private										
64 KB store-through L1 D-cache 0.8ns latency, 80 GB/s private	256 KB store-in L2 cache 2.0-ns latency, 256 GB/s private										
4 MB store-in L2 cache ~5.0-ns latency, 160 GB/s private	4 MB partial victim local L3 region ~6.0-ns latency, 128 GB/s private										
32 MB victim L3 cache ~35-ns latency, 80 GB/s shared by 2	32 MB adaptive victim L3 cache ~30-ns latency, 512 GB/s shared by 8										

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> also discloses the following at 17: [T]he POWER7 cache hierarchy includes a shared 32-MB L3 cache comprised of the 4-MB local L3 regions from the eight cores that reside on the processor chip. This enables the signaling resources that would have been used for communication with an off-chip L3 cache to be redeployed for other purposes, such as the memory and I/O subsystem interfaces. <i>Sinharoy</i> further discloses the following at 19: The shared 32-MB POWER7 L3 cache is composed of eight 4-MB L3 regions. Each L3 region utilizes 128-byte lines and is eight-way set associative. A given L3 region comprises a single centralized controller, with four address hashed eDRAM cache data banks and four address hashed SRAM directory banks. The local L3 region is tightly coupled to the L2 cache associated with a given core. All L3 constructs operate in 2:1 cycles (defined in the L2 cache section).
[1c] an integrated input/output (IIO) module to interface between the multicore processor and at least one IO device coupled to the multicore processor, and	<i>Sinharoy</i>, alone or in combination with <i>Agarwal, Berg, Jain, and/or Guthrie</i>, discloses or renders obvious “an integrated input/output (IIO) module to interface between the multicore processor and at least one IO device coupled to the multicore processor.” <i>Sinharoy</i> discloses two integrated I/O controllers. <i>See Sinharoy</i> at 22. These integrated I/O controllers connect (i.e., “interface between”) one or more input/output devices (e.g., PCI device) and the multicore processor via an I/O hub. <i>See id.</i> at 17, 22, Fig. 14. <i>Sinharoy</i> discloses the following at 17: [N]ew coherent operations are supported. One such operation is cache injection. An I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory. If a given core’s L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME,

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region. Further, <i>Sinharoy</i> discloses the following at 22: The POWER7 chip supports two integrated I/O controllers. . . . Each of the controllers 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. These interfaces operate at frequencies ranging from 1.25 to 2.5 GHz, depending on the I/O hub chips to which they are connected. In addition to supporting the interrupt presentation function, pipelined I/O high-throughput mode, and advanced partial DMA write management pioneered in the POWER6 design, the POWER7 I/O controllers also support cache injection, which enables DMA write traffic to be written directly into a cache, instead of to memory. <i>Sinharoy</i> also discloses the following at 22: As shown in Figure 1, the POWER7 chip contains eight cores, each with an associated L2 cache and local L3 region, two memory controllers, two I/O controllers, and a multichip SMP interconnect capable of scaling up to 32 POWER7 chips into a single large SMP system. Consequently, each POWER7 chip contains several coherence request, snoop, response, and notification interfaces and several data interfaces. <i>Berg</i> discloses the following at 1: Take, for instance, a typical STB, which does much more than tune and decode television signals. Figure 1 shows the architecture of a modern STB controlled by a system on chip (SoC) with many I/O interface units, specialized hardware accelerators for processing data, and one or more general-purpose CPUs to process the data and control various other on-chip components. This multifunction STB adds several complex subsystems: hard disks, high-speed network support, graphic

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	processors, and expansion peripheral ports. These I/O subsystems are a critical part of the STB functionality, and their importance will continue to grow as new technologies, such as Internet Protocol television (IPTV), are implemented. <i>Berg</i> also discloses the following at 2: Most mechanisms for passing data between I/O devices and a CPU use the classic producer-consumer model.¹ In this model, the producer (either an I/O device or CPU) writes data into memory and then sets a flag by writing to a memory location or asserting an interrupt. When the consumer recognizes that the flag is set, either by polling or by receiving the interrupt, it reads the data from memory. The hardware must function such that once the consumer recognizes that the flag is set, subsequent reads of the data return updated values. <i>Berg</i> also discloses the following at 2: <i>I/O data coherence</i> refers to maintaining coherency and consistency of data passed between an I/O peripheral device and one or more CPUs. The data can flow in either direction: an I/O device produces data and the CPUs consume it, or the CPUs produce the data and an I/O device consumes it. <i>Berg</i> also discloses the following at 3: Systems with <i>hardware I/O coherence</i> implement circuitry that reduces the burden on software to maintain I/O operations' consistency and coherency. For example, hardware can automatically invalidate L1 cache lines when an I/O device writes to memory.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Berg</i> also discloses the following at 4: Figure 3 shows a typical embedded hardware I/O coherence architecture. Reads and writes from an I/O device can be treated in a coherent manner as they're routed through the I/O interconnect to the coherence manager and into the optional L2 cache or to main memory. The coherence manager might or might not keep I/O transactions coherent with L1 caches. Without L1 cache coherency, the coherence manager steers I/O transactions directly to the L2 cache, and software must manage the L1 cache coherence as described previously. However, because both CPUs and I/O devices access the L2 cache, coherence is implicitly maintained for data stored there. <i>Berg</i> also discloses the following at 7: An alternative to traditional hardware or software I/O coherence architectures involves passing data between CPUs and I/O devices via specialized hardware or software mechanisms. One alternative I/O coherence scheme uses one or more private memories, as Figure 5 shows. In this architecture, a Scratchpad RAM (SPRAM) is directly attached to each CPU. <i>Berg</i> also discloses the following at 8: Figure 6 shows a hybrid architecture that simultaneously supports all the described methods of passing data between I/O devices and CPUs. Data can flow from I/O devices to or from memory, one of the L1 caches, the L2 cache, or an SPRAM, if implemented. The architecture's flexibility is well suited for embedded devices running diverse applications, such as STBs. In these devices, specialized tuner and decoder hardware process the high-bandwidth video stream. Because the video data isn't touched by a CPU, it can flow directly to memory, bypassing the coherence manager and L2 cache. However, because the CPU processes Ethernet traffic, having the data flow through the coherence manager into the L2 cache leads to better performance.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Berg</i> also discloses the following at 9: The key to making full use of the hybrid architecture is having software intelligent enough to determine which I/O coherence method to implement for a given portion of an application, as well as hardware that can route transactions appropriately. One approach is to implement an I/O memory management unit (IOMMU) in the I/O subsystem and to associate a cache-ability attribute with each IOMMU entry. IOMMUs are commonly used in servers for address space isolation and protection, and to enable I/O devices to access larger address spaces. To some degree, embedded systems might be able to exploit the existing operating system and driver support. <i>Berg</i> also discloses the following at 9: When the software schedules an I/O operation, it allocates and initializes an IOMMU entry supplying the buffer's address range, routing information, and associated cache-ability attribute. The I/O interconnect routes transactions to main memory, the SPRAM, or the coherence manager based on the IOMMU entry for the corresponding address. When the I/O transfer is complete, the software deallocates the IOMMU entry.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Berg also discloses the following in Fig. 1:

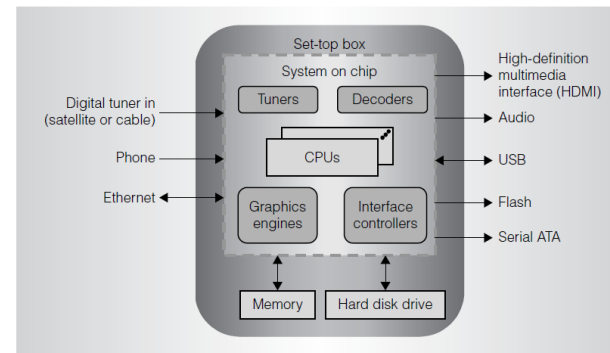


Figure 1. A typical set-top box block diagram. Modern STBs process many input and output data streams with multiple dedicated subsystems and general purpose CPUs.

Berg also discloses the following in Fig. 2:

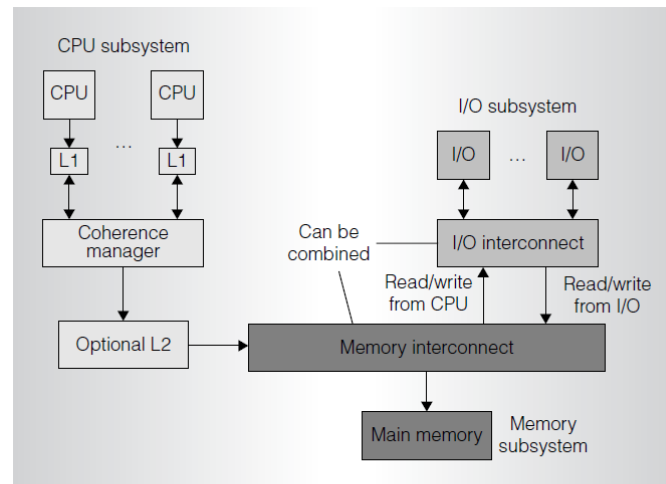


Figure 2. A software I/O coherence architecture. Although the coherence manager maintains coherence between the L1 caches of multiple CPUs, software is responsible for keeping I/O data coherent with the L1 caches.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Berg also discloses the following in Fig. 3:

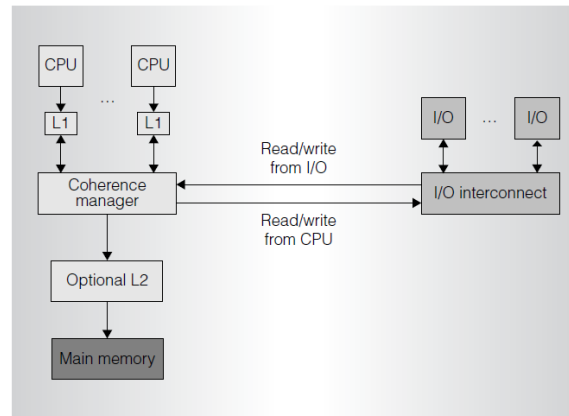


Figure 3. A hardware I/O coherence architecture. In these architectures, the coherence manager maintains coherence between the I/O data and L1 caches. The CPUs and I/O devices can both access the L2 cache.

Berg also discloses the following in Fig. 5:

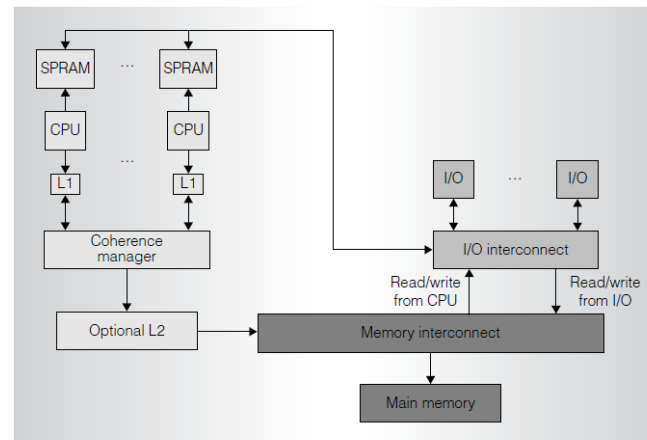
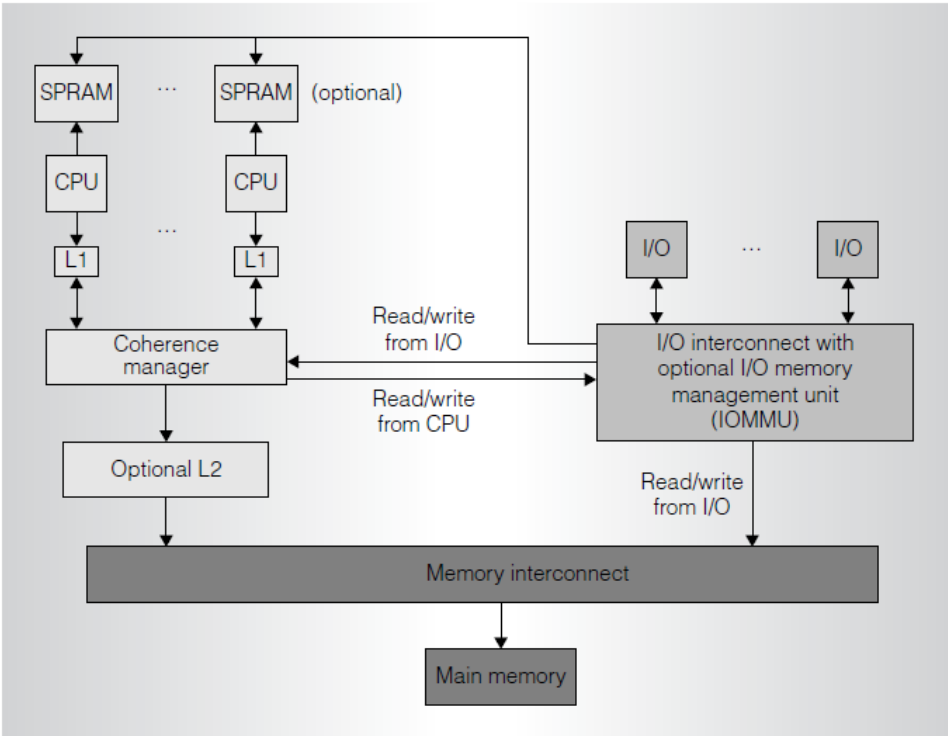


Figure 5. The Scratchpad RAM architecture. In this scheme, each CPU has its own private memory, or SPRAM, that I/O devices can also access.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Berg</i> also discloses the following in Fig. 6:  Figure 6. Hybrid architecture combining software and hardware I/O coherence. This architecture supports data flowing between I/O devices and memory, an L1 cache, the L2 cache, or SPRAM. <i>See also</i> Exs. D-1; D-3, D-4 at [1c].

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

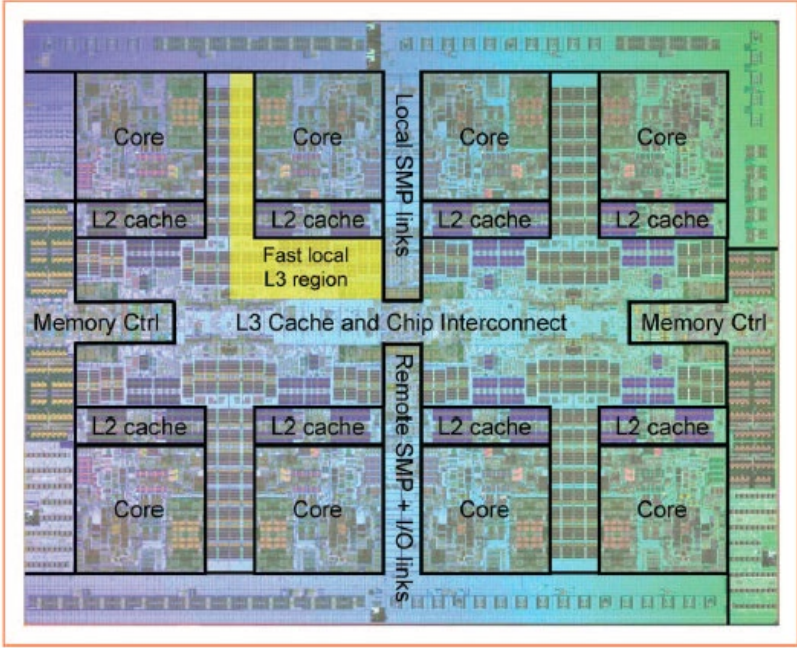
Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[1d] a caching agent to perform cache coherency operations for the plurality of cores and the IIO module,	<i>Sinharoy</i> discloses or suggests “a caching agent to perform cache coherency operations for the plurality of cores and the IIO module.” <i>Sinharoy</i> discloses a distributed “coherence fabric” comprising eight L2 cache controllers and eight L3 cache controllers. <i>Sinharoy</i> at 19; <i>see also id.</i> at 17-19. This distributed coherence fabric is a “caching agent” as claimed. <i>Sinharoy</i>’s multicore processor includes three layers of caches. <i>See Sinharoy</i> at Table 1. The L1 and L2 caches are private to each core, and the L3 cache is shared by all cores. <i>See id.</i> at 19, Table 1. The 32-MB L3 cache is organized as a collection of eight distributed 4-MB local L3 regions, and each of the local L3 regions is located next to a core/L2 to provide faster access to the corresponding core/L2. <i>See id.</i> at 19, Fig. 1. Each of <i>Sinharoy</i>’s eight L2 caches (one per core) and each of the local L3 cache regions (also one per core) includes a controller to help maintain cache coherency across all eight cores.⁴ <i>See id.</i> at 18-19. For example, each of the controllers for the L2 and L3 caches supports the “13-cache-state protocol” defined in Table 2. <i>See id.</i> at 17, Table 2. To maintain cache coherency for the cores, each L2 and L3 controller also includes a pool of read/claim (RC) machines and write machines that perform cache coherency operations, such as fetches, read-write operations, and core prefetch operations. <i>See id.</i> at 18-19. These cache states support “coherent operations.” <i>See id.</i> at 17. To maintain cache coherency, each L2 and L3 controller includes a pool of read/claim (RC) machines and write machines that perform cache coherency operations, such as fetches, read-write operations, and core prefetch operations. <i>Sinharoy</i> at 18-19.

⁴ Note that *Sinharoy*’s L1 cache does not have to be separately managed for cache coherency because (i) the L1 I-cache (instruction cache) is a read-only cache, so it does not have cache coherency problem by definition, and (ii) the L1 D-cache (data cache) uses the store-through cache policy (also known as write-through policy), which means all writes are sent to the next-level memory hierarchy (L2 cache) without delay, so the L2 cache always has up-to-date data. Thus, managing L2 cache coherency suffices to provide all cores a coherent view of the L1 cache memory space. *See Sinharoy* at 11.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> further explains, for example, that “[s]torage accesses that miss both the L2 cache and the 4-MB local L3 region are broadcast to the coherence fabric” where they are snooped by “other L2 caches, possibly other L3 caches, and by the seven remote 4-MB L3 regions that comprise the remaining 28 MB of the on-chip L3 cache” to check whether another core contains modified data for the same address in the cache (e.g., a core’s L2 cache or the local L3 cache region). <i>Id.</i> at 19. This enables checking whether another core contains modified data for the same address in the cache (e.g., a core’s L2 cache or the local L3 cache region). <i>See Sinharoy</i> at 19. <i>Sinharoy</i>’s cache architecture also supports “coherent operations” like “cache injection” that allows an I/O device to perform coherency operations (e.g., data read or write) targeted directly at the cache. <i>See id.</i> at 17, 22. For example, when an I/O device sends a write command to the CPU caches via the integrated I/O controller, “[i]f a given core’s L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME, MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region.” <i>Sinharoy</i> at 17. The multicore processor maintains cache coherency for the cores and integrated I/O controllers through an on-chip coherence interconnect that includes “coherent request, snoop, response, and notification interfaces and several data interfaces.” <i>See id.</i> at 22.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following at Fig. 1:  The image is a die photo of an IBM POWER7 chip. It shows a symmetrical layout of eight processing cores arranged in a 2x4 grid. Each core is connected to a local L2 cache. A central horizontal band is labeled 'L3 Cache and Chip Interconnect'. On the left and right sides of this band are 'Memory Ctrl' blocks. Vertical lines of connections are labeled 'Local SMP links' on the left and 'Remote SMP + I/O links' on the right. A yellow highlighted area in the center is labeled 'Fast local L3 region'. Figure 1 Die photo of the IBM POWER7 chip. <i>Sinharoy</i> discloses the following at 11: The L1 D-cache is a store-through design; all stores are sent to the L2 cache, and no L1 cast-outs are required. The L1 D-cache is not allocated on a store miss; the store is just sent to the L2 cache. The L1 D-cache is inclusive of the L2 cache. The L1 D-cache has byte-write capability of up to 16 bytes within a given 32-byte sector in support of store instructions.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following at 17: [T]he POWER7 cache hierarchy includes a shared 32-MB L3 cache comprised of the 4-MB local L3 regions from the eight cores that reside on the processor chip. This enables the signaling resources that would have been used for communication with an off-chip L3 cache to be redeployed for other purposes, such as the memory and I/O subsystem interfaces. <i>Sinharoy</i> discloses the following at 17 (citation omitted): As shown in Table 2, the POWER7 L2 and L3 caches support the same 13-cache-state protocol as the POWER6 design point. While no new cache states have been added for POWER7, new coherent operations are supported. One such operation is cache injection. An I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory. If a given core's L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME, MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region. <i>Sinharoy</i> also discloses the following at 19: The shared 32-MB POWER7 L3 cache is composed of eight 4-MB L3 regions. Each L3 region utilizes 128-byte lines and is eight-way set associative. A given L3 region comprises a single centralized controller, with four address hashed eDRAM cache data banks and four address hashed SRAM directory banks. The local L3 region is tightly coupled to the L2 cache associated with a given core. All L3 constructs operate in 2:1 cycles (defined in the L2 cache section).

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Further, *Sinharoy* discloses the following at Table 2:

Table 2 IBM POWER7 cache states.

State	Description	Authority	Sharers and scope	Source data	Data cast-out	Scope cast-out
I	Invalid	None	N/A	N/A	N/A	None
ID	Deleted, do not allocate	None	N/A	N/A	N/A	None
S	Shared	Read	Yes, scope unknown	No	No	None
SL	Shared, local data source	Read	Yes, scope unknown	At request	No	None
T	Formerly MU, now shared	Update	Yes, probably global	If notified	Yes	Required, global
TE	Formerly ME, now shared	Update	Yes, probably global	If notified	No	Required, global
M	Modified, avoid sharing	Update	No	At request	Yes	Optional, local
ME	Exclusive	Update	No	At request	No	None
MU	Modified, bias toward sharing	Update	No	At request	Yes	Optional, local
IG	Invalid, cached scope-state	None	N/A, probably global copies	N/A	N/A	Required, global
IN	Invalid, scope predictor	None	N/A, probably local copies	N/A	N/A	None
TN	Formerly MU, now shared	Update	Yes, local	If notified	Yes	Optional, local
TEN	Formerly ME, now shared	Update	Yes, local	If notified	No	None

Sinharoy further discloses the following at 19:

The centralized L3 region controller provides a single core/L2 dispatch port, a single lateral L3 region dispatch port, dual snoop dispatch ports (for even and odd cache lines), and a single pool of operational resources. Storage accesses that miss the L2 cache access the 4-MB local L3 region via the core/L2 dispatch port. Those that hit in the 4-MB local L3 region are managed by a pool of eight read machines. Prefetches and some L2 store misses also access the L3 region via the core/L2 dispatch port and are managed by a pool of 24 lightweight RC machines. When prefetched data is staged to the L3 cache, it is managed by a pool of ten write

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	machines. Each write machine has an associated cast-out machine to manage the eviction of lines displaced by writes. Note that these write machines are also utilized by other operations such as cast-outs and cache-injections. 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 cache. Therefore, operations that miss the 4-MB local L3 region but hit in the remaining 28 MB of the L3 cache access the cache via the snoop dispatch ports. Further, <i>Sinharoy</i> discloses the following at 22: These interfaces operate at frequencies ranging from 1.25 to 2.5 GHz, depending on the I/O hub chips to which they are connected. In addition to supporting the interrupt presentation function, pipelined I/O high-throughput mode, and advanced partial DMA write management pioneered in the POWER6 design, the POWER7 I/O controllers also support cache injection, which enables DMA write traffic to be written directly into a cache, instead of to memory. <i>Sinharoy</i> also discloses the following at 22: As shown in Figure 1, the POWER7 chip contains eight cores, each with an associated L2 cache and local L3 region, two memory controllers, two I/O controllers, and a multichip SMP interconnect capable of scaling up to 32 POWER7 chips into a single large SMP system. Consequently, each POWER7 chip contains several coherence request, snoop, response, and notification interfaces and several data interfaces. <i>See also supra</i> at [1c]. <i>See also</i> Exs. D-1; D-3, D-4 at [1d].

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[1e] the caching agent a single caching agent for the multicore processor and including a plurality of distributed portions each associated with a corresponding one of the plurality of cores.	<i>Sinharoy</i> teaches or suggests “the caching agent a single caching agent for the multicore processor and including a plurality of distributed portions each associated with a corresponding one of the plurality of cores.” <i>Sinharoy</i> teaches that each core has a corresponding L2 cache and 4-MB L3 cache region managed by a “controller” that form a distributed “coherence fabric” across the cores (i.e., “a single caching agent”). <i>See Sinharoy</i> at 17-19, Fig. 1. Each L2 cache and each L3 cache region includes its own controller (i.e., a distributed portion of the caching agent) responsible for maintaining cache coherency among the cores. <i>See id.</i> at 17-19. Since each L2 cache is private to its corresponding core, it is associated with a corresponding core. And although the entire 32-MB L3 cache is shared by all eight cores, each local 4-MB L3 cache region is tightly coupled to the associated core and L2 cache to provide faster access time. <i>See id.</i> Accordingly, each local 4-MB L3 cache region is also associated with a corresponding core. <i>Sinharoy</i> further teaches that the multicore processor utilizes a “nonblocking-broadcast-based coherence-transport mechanism” that provides a distributed cache coherency management based on the “13-cache-state protocol” shown in Table 2. <i>See id.</i> at 17, 23, Table 2. For example, each local L3 cache region controller provides “a single core/L2 dispatch port, a single lateral L3 region dispatch port, dual snoop dispatch ports (for even and odd cache lines), and a single pool of operational resources.” <i>Id.</i> at 19. Working together with the eight L2 cache controllers, the eight L2 controllers and the eight L3 controllers, collectively, implement the broadcast-based distributed cache coherency functionality for all eight cores. For example, when a core memory access misses both its L2 cache and its 4-MB local L3 cache region, snoop messages are broadcasted throughout this distributed coherence fabric to the other cache regions, and the snooped cache regions send snoop responses and also potentially forward the local data to the requester, if needed. <i>See id.</i> at 19. Additionally, the multicore processor supports “coherent operations” like “cache injection” that allows an I/O device to perform coherency operations (e.g., data read or write) targeted directly to the core caches. <i>See id.</i> at 17, 22. For example, “[i]f a given core’s L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME, MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region.” <i>Id.</i> at 17.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	The multicore processor maintains cache coherency for the eight cores and two integrated I/O controllers through an on-chip coherence interconnect that includes “coherent request, snoop, response, and notification interfaces and several data interfaces.” <i>Id.</i> at 22. <i>Sinharoy</i> thus discloses or at least suggests a single caching agent (a “coherence fabric,” including eight L2 controllers and eight L3 controllers distributed among the cores) that performs cache coherency operations for all cores and integrated I/O controllers. For example, <i>Sinharoy</i> discloses the following at Fig. 1: Figure 1 Die photo of the IBM POWER7 chip.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following at 17: [T]he POWER7 cache hierarchy includes a shared 32-MB L3 cache comprised of the 4-MB local L3 regions from the eight cores that reside on the processor chip. This enables the signaling resources that would have been used for communication with an off-chip L3 cache to be redeployed for other purposes, such as the memory and I/O subsystem interfaces. <i>Sinharoy</i> discloses the following at 17 (citation omitted): As shown in Table 2, the POWER7 L2 and L3 caches support the same 13-cache-state protocol as the POWER6 design point. While no new cache states have been added for POWER7, new coherent operations are supported. One such operation is cache injection. An I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory. If a given core's L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME, MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region. <i>Sinharoy</i> also discloses the following at 19: The shared 32-MB POWER7 L3 cache is composed of eight 4-MB L3 regions. Each L3 region utilizes 128-byte lines and is eight-way set associative. A given L3 region comprises a single centralized controller, with four address hashed eDRAM cache data banks and four address hashed SRAM directory banks. The local L3 region is tightly coupled to the L2 cache associated with a given core. All L3 constructs operate in 2:1 cycles (defined in the L2 cache section).

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> further discloses the following at 19: The centralized L3 region controller provides a single core/L2 dispatch port, a single lateral L3 region dispatch port, dual snoop dispatch ports (for even and odd cache lines), and a single pool of operational resources. Storage accesses that miss the L2 cache access the 4-MB local L3 region via the core/L2 dispatch port. Those that hit in the 4-MB local L3 region are managed by a pool of eight read machines. Prefetches and some L2 store misses also access the L3 region via the core/L2 dispatch port and are managed by a pool of 24 lightweight RC machines. When prefetched data is staged to the L3 cache, it is managed by a pool of ten write machines. Each write machine has an associated cast-out machine to manage the eviction of lines displaced by writes. Note that these write machines are also utilized by other operations such as cast-outs and cache-injections. 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 cache. Therefore, operations that miss the 4-MB local L3 region but hit in the remaining 28 MB of the L3 cache access the cache via the snoop dispatch ports. <i>Sinharoy</i> discloses the following at 22: These interfaces operate at frequencies ranging from 1.25 to 2.5 GHz, depending on the I/O hub chips to which they are connected. In addition to supporting the interrupt presentation function, pipelined I/O high-throughput mode, and advanced partial DMA write management pioneered in the POWER6 design, the POWER7 I/O controllers also support cache injection, which enables DMA write traffic to be written directly into a cache, instead of to memory.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 1	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following at 22: As shown in Figure 1, the POWER7 chip contains eight cores, each with an associated L2 cache and local L3 region, two memory controllers, two I/O controllers, and a multichip SMP interconnect capable of scaling up to 32 POWER7 chips into a single large SMP system. Consequently, each POWER7 chip contains several coherence request, snoop, response, and notification interfaces and several data interfaces.

Claim 2	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[2] The apparatus of claim 1, wherein the caching agent includes a table of requests to store pending requests for the plurality of cores and the IIO module.	<i>Sinharoy</i>, alone or in combination with <i>Blumrich</i>, discloses and/or renders obvious “the caching agent includes a table of requests to store pending requests for the plurality of cores and the IIO module.” The specification of the '228 Patent defines a table of requests (TOR) as “a buffer or other temporary storage for holding incoming requests.” '228 Patent at 2:46-48. <i>Sinharoy</i> discloses this feature as explained below. <i>Sinharoy</i> explains that the controller in each of the eight 4-MB local L3 cache regions (part of the distributed “coherence fabric”) “provides a single core/L2 dispatch port, a single lateral L3 region dispatch port, dual snoop dispatch ports (for even and odd cache lines), and a single pool of operational resources.” <i>See Sinharoy</i> at 19. Accordingly, all incoming requests to the L3 controller must be arbitrated and processed using this single pool of resources. <i>Sinharoy</i> further explains that this single pool of resources includes “a pool of eight read machines,” “a pool of 24 lightweight RC [read/claim] machines,” and “a pool of ten write machines. <i>Id.</i> Read machines manage read hits; RC machines manage store misses and prefetches; and write machines manage data writes. <i>See id.</i> Write machines also perform core prefetch operations, L2 cast-out operations, and I/O cache injection operations. <i>See id.</i> at 17-20. Because the pool of resources has limited entries (e.g., capable of handling 8 outstanding reads, 24 outstanding store misses, and 10 outstanding data writes), the controller must use

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 2	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	some sort of storage element (e.g., queues or buffers) to track outstanding requests and avoid overflowing the limited pool of resources. <i>See, e.g., id.</i> at 10 (explaining the core includes various buffers/queues to keep track of the outstanding requests, such as providing a 32-entry SRQ for 32 outstanding stores and a 32-entry LRQ for total 32 outstanding loads). Thus, <i>Sinharoy</i> discloses “a buffer or other temporary storage for holding incoming requests,” which according to the '228 Patent, constitutes a “table of requests” as claimed. <i>See</i> '228 Patent at 2:46-48. <i>Sinharoy</i> discloses the following at 17 (citation omitted): As shown in Table 2, the POWER7 L2 and L3 caches support the same 13-cache-state protocol as the POWER6 design point. While no new cache states have been added for POWER7, new coherent operations are supported. One such operation is cache injection. An I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory. If a given core’s L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME, MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region. <i>Sinharoy</i> discloses the following at 18: To further reduce energy usage, the read/claim (RC) machines, which have been utilized in prior designs to manage core fetches, read–modify–write operations on behalf of core stores, and core prefetch operations, have been divided into two classes: 1) a set of eight fully functional RC machines that are incorporated into the L2 cache controller and manage core fetches and stores that hit the L2 cache and 2) a set of 24 more energy-efficient ‘lightweight’ RC machines that are incorporated into the L3 cache controller and manage prefetches and stores that miss the L2 cache. These lightweight RC machines stage data through the L3 cache and will be described more fully in the section on L3 cache.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 2	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> also discloses the following at 19: The shared 32-MB POWER7 L3 cache is composed of eight 4-MB L3 regions. Each L3 region utilizes 128-byte lines and is eight-way set associative. A given L3 region comprises a single centralized controller, with four address hashed eDRAM cache data banks and four address hashed SRAM directory banks. The local L3 region is tightly coupled to the L2 cache associated with a given core. All L3 constructs operate in 2:1 cycles (defined in the L2 cache section). <i>Sinharoy</i> also discloses the following at 19: The centralized L3 region controller provides a single core/L2 dispatch port, a single lateral L3 region dispatch port, dual snoop dispatch ports (for even and odd cache lines), and a single pool of operational resources. <i>Sinharoy</i> also discloses the following at 19: Storage accesses that miss the L2 cache access the 4-MB local L3 region via the core/L2 dispatch port. Those that hit in the 4-MB local L3 region are managed by a pool of eight read machines. Prefetches and some L2 store misses also access the L3 region via the core/L2 dispatch port and are managed by a pool of 24 lightweight RC machines. When prefetched data is staged to the L3 cache, it is managed by a pool of ten write machines. Each write machine has an associated cast-out machine to manage the eviction of lines displaced by writes. Note that these write machines are also utilized by other operations such as cast-outs and cache-injections. <i>Sinharoy</i> also discloses the following at 20: L2 cast-out operations are managed in the L3 region by a pool of ten L3 write machines and ten associated L3 cast-out machines.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 2	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Blumrich</i> discloses the following at [0128]. The request queue 302 buffers incoming read and write requests. In this embodiment, it is 16 entries deep, though other request buffers might have more or less entries. The addresses of incoming requests are matched against all pending requests to determine ordering restrictions. The queue presents the requests to the directory pipeline 308 based on ordering requirements. <i>Blumrich</i> discloses the following at Fig. 3. FIG. 3 <i>See also supra</i> 1[c].

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 4	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[4] The apparatus of claim 1, wherein the shared cache memory comprises a last level cache (LLC) including a plurality of distributed portions each associated with a corresponding one of the plurality of cores.	<i>Sinharoy</i> discloses or suggests “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.” <i>Sinharoy</i> discloses a shared 32-MB L3 cache that comprises eight local regions (the 4-MB L3 regions) distributed across the die among the eight cores in the processor. <i>See Sinharoy</i> at 17, 19, Fig. 1. The L3 cache is the last level cache. <i>See id.</i> at Table 1. Each of the 4-MB L3 regions is associated with the coupled L2 cache and an associated core. For example, the 4-MB L3 region and the coupled L2 share the same data and control flow logic, and the “associated L2 cache utilizes the 4-MB local L3 region directly as a victim cache.” <i>See id.</i> at 16-17, 19, Fig. 1. The yellow highlighting in <i>Sinharoy</i>’s Figure 1 below indicates the local L3 region associated with its neighboring core, and each of the other seven cores similarly has an associated “Fast local L3 region” neighboring it. <i>Sinharoy</i> discloses the following at Fig. 1: Figure 1 Die photo of the IBM POWER7 chip.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 4	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>										
	<i>Sinharoy</i> discloses the following at Table 1: Table 1 Comparison of POWER6 and POWER7 cache hierarchies. <table> <tr> <th><i>POWER6 (assuming 5-GHz core)</i></th><th><i>POWER7 (assuming 4-GHz core)</i></th></tr> <tr> <td>64 KB store-through L1 D-cache 0.8ns latency, 80 GB/s private</td><td>32 KB store-through L1 D-cache 0.5ns latency, 192 GB/s private</td></tr> <tr> <td>4 MB store-in L2 cache ~5.0-ns latency, 160 GB/s private</td><td>256 KB store-in L2 cache 2.0-ns latency, 256 GB/s private</td></tr> <tr> <td>32 MB victim L3 cache ~35-ns latency, 80 GB/s shared by 2</td><td>4 MB partial victim local L3 region ~6.0-ns latency, 128 GB/s private</td></tr> <tr> <td></td><td>32 MB adaptive victim L3 cache ~30-ns latency, 512 GB/s shared by 8</td></tr> </table> <i>Sinharoy</i> discloses the following at 16-17: Developed for this express purpose, it has been exploited by the POWER7 cache microarchitects to create a low-latency 4-MB local L3 region, tightly coupled to the private L2 associated with each core. <i>Sinharoy</i> also discloses the following at 19: The shared 32-MB POWER7 L3 cache is composed of eight 4-MB L3 regions. Each L3 region utilizes 128-byte lines and is eight-way set associative. A given L3 region comprises a single centralized controller, with four address hashed eDRAM cache data banks and four address hashed SRAM directory banks. The local L3 region is tightly coupled to the L2 cache associated with a given core. All L3 constructs operate in 2:1 cycles (defined in the L2 cache section). <i>Sinharoy</i> further discloses the following at 19: As illustrated in Figure 10, the capacity, data flow logic, and control flow logic of a 4-MB local L3 region are shared between the L2 cache to which the local L3 region is coupled, and the other seven 4-MB L3 regions on the chip. The associated	<i>POWER6 (assuming 5-GHz core)</i>	<i>POWER7 (assuming 4-GHz core)</i>	64 KB store-through L1 D-cache 0.8ns latency, 80 GB/s private	32 KB store-through L1 D-cache 0.5ns latency, 192 GB/s private	4 MB store-in L2 cache ~5.0-ns latency, 160 GB/s private	256 KB store-in L2 cache 2.0-ns latency, 256 GB/s private	32 MB victim L3 cache ~35-ns latency, 80 GB/s shared by 2	4 MB partial victim local L3 region ~6.0-ns latency, 128 GB/s private		32 MB adaptive victim L3 cache ~30-ns latency, 512 GB/s shared by 8
<i>POWER6 (assuming 5-GHz core)</i>	<i>POWER7 (assuming 4-GHz core)</i>										
64 KB store-through L1 D-cache 0.8ns latency, 80 GB/s private	32 KB store-through L1 D-cache 0.5ns latency, 192 GB/s private										
4 MB store-in L2 cache ~5.0-ns latency, 160 GB/s private	256 KB store-in L2 cache 2.0-ns latency, 256 GB/s private										
32 MB victim L3 cache ~35-ns latency, 80 GB/s shared by 2	4 MB partial victim local L3 region ~6.0-ns latency, 128 GB/s private										
	32 MB adaptive victim L3 cache ~30-ns latency, 512 GB/s shared by 8										

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 4	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	L2 cache utilizes the 4-MB local L3 region directly as a victim cache. The other seven 4-MB L3 regions on the chip also use the 4-MB local L3 region as a victim cache under certain circumstances. Note that when the core and L2 associated with a given L3 region are disabled, the remaining L3 regions enjoy full access to the L3 region.

Claim 5	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[5] The apparatus of claim 4, wherein the plurality of distributed portions of the caching agent are coupled via a ring interconnect.	<i>Sinharoy</i>, alone or in combination with <i>Agarwal</i>, <i>Jain</i>, <i>Sistla</i>, <i>Herrero</i>, and/or <i>Chrysos</i>, teaches or renders obvious “the plurality of distributed portions of the caching agent are coupled via a ring interconnect.” <i>Sinharoy</i> explains its L3 local regions are connected through an on-chip interconnect that are centralized at the center of the chip. <i>See Sinharoy</i> at Fig. 1, 22-23 (“These are tied together by 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”). <i>Sinharoy</i>’s on-chip interconnect uses an “arbitration logic located at the center of the chip.” <i>See id.</i> at 23. Each of the L3 cache controllers (part of <i>Sinharoy</i>’s distributed “coherence fabric” that corresponds to the claimed “caching agent”), for example, receives snoops through two sets (odd, even) of coherent interconnect routes from the cores or other instances of the distributed L3 cache controllers. <i>See id.</i> at 19, 22-23, Fig. 1. Although <i>Sinharoy</i> may not expressly mention using a ring interconnect, a POSITA would have found it obvious to use a ring interconnect in a multicore processor architecture with all the relevant features in <i>Sinharoy</i>’s architecture. <i>Sistla</i>, for example, teaches a multicore processor with a distributed cache much like <i>Sinharoy</i>’s. <i>See Sistla</i> at [0001], Figs. 3-4. <i>Sistla</i>’s multicore processor comprises “a plurality of core-cache clusters 310-310 in communication with each other over an on-die interconnect 320,” and each of the core-cache cluster includes a scalability agent (a distributed portion of a caching agent) that maintains memory coherency. <i>See id.</i> at [0018], [0030], [0031]. Further, <i>Sistla</i> explains the on-die interconnect in such an architecture may be a ring interconnect. <i>See id.</i> at [0030]. <i>Sistla</i> thus

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 5	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	demonstrates that the relevant features in <i>Sinharoy</i>'s architecture are compatible with a ring interconnect. <i>See id.</i> <i>Sinharoy</i> discloses the following at Fig. 1: Figure 1 Die photo of the IBM POWER7 chip. <i>Sinharoy</i> discloses the following at 19: The centralized L3 region controller provides a single core/L2 dispatch port, a single lateral L3 region dispatch port, dual snoop dispatch ports (for even and odd cache lines), and a single pool of operational resources.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 5	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following at 22-23: As shown in Figure 1, the POWER7 chip contains eight cores, each with an associated L2 cache and local L3 region, two memory controllers, two I/O controllers, and a multichip SMP interconnect capable of scaling up to 32 POWER7 chips into a single large SMP system. Consequently, each POWER7 chip contains several coherence request, snoop, response, and notification interfaces and several data interfaces. These are tied together by 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. <i>Sinharoy</i> discloses the following at 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. Up to one even request and one odd request may be granted each on-chip bus cycle. Once granted, the requests are broadcast within the chip on the even/odd snoop buses outward toward the left and right edges of the chip. <i>Agarwal</i> discloses the following at 3:45-47: The processor cores are interconnected over a network that includes at least one of a bus, a ring, a mesh, a torus, a 3D cube, or a hypercube. <i>Agarwal</i> disclose the following at 11:51-12:2: The SCC scheme applies to a wide range of multicore processor architectures as well as to multiprocessors in general. An exemplary multicore processor 200 having a tiled processor architecture is shown in FIG. 2. The figure shows 9 tiles (corresponding to the “processor cores” or simply “cores”) numbered T0-T8, each tile 202 containing a processor 204 (or “processing engine”), a cache 206 and a switch 208. Also shown are data paths 209 of a mesh network connecting the tiles

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 5	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	through the Switches. In some tiled architectures the switch 208 can include multiple switching modules for different independent networks connecting the tiles, such as static networks (in which routing is controlled by Statically programmed Switch instructions) or dynamic networks (in which messages are dynamically routed at runtime). A memory dynamic network can be dedicated to routing messages associated with memory access. The memory coherence techniques can be applied to architectures with any type of network including a bus, a ring, a mesh, a torus, a 3D cube, a hypercube, or any other director indirect network. <i>Agarwal</i> discloses the following at 20:63-67: As before, this protocol maintains cache coherence in a multicore (also referred to as a chip multiprocessor or CMP) processor, wherein the multiple processors each having private caches are connected together in a mesh, ring, bus or other network (direct or indirect) network. <i>Jain</i> discloses the following at [0067]: Like memory interface 130 discussed above, memory interfaces 430a-b may be configured to manage the transfer of data between L2 cache 420 and system memory, via coherence units 450. Inter-processor links 460a-b may be configured to provide direct communication between instances of processor 10, e.g., for the exchange of memory data, I/O data, coherence control information, or other inter processor communication. For example, these links may be implemented as high-speed serial point-to-point interconnects, such that each processor 10 may have one or more direct communication paths to the other processors 10. Alternative interconnect topologies such as shared buses, rings, cubes, etc. may also be employed.

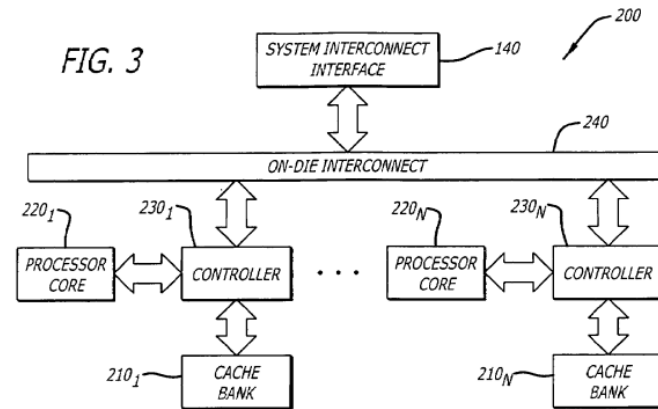
Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 5	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sistla</i> discloses the following at [0018]-[0019]: According to one embodiment of the invention, the “scalability agent protocol” is a communication scheme that enables the aggregating of core-cache clusters operating independently from each other, and provides “graceful” scalability where the responsibility for maintaining memory coherency is approximately shared equally among the core cache clusters despite any increase or decrease in number. This is accomplished because the scalability agent protocol is partitionable among address regions in the system memory address space. The latencies to the cache in the core-cache cluster substantially remains unchanged as the number of cores increases. A “scalability agent” is hardware and/or software that manages the flow of outgoing and incoming transactions into a socket associated with the core-cache cluster and supports the scalability agent protocol described above. According to one embodiment of the invention, the scalability agent (i) aggregates core-cache clusters to appear as one caching agent, (ii) handles local cache coherence between core-cache clusters on the same integrated circuit (IC), and (iii) support scalability so that the operations of a core-cache cluster are not significantly effected [sic] if other core-cache clusters are added. <i>Sistla</i> discloses the following at [0030]: Multi core processor 300 comprises a plurality of core-cache clusters 310-310 in communication with each other over an on-die interconnect 320. Multi-core processor 300 is in communication with externally located devices over a system interconnect interface 140. According to one embodiment of the invention, on-die interconnect 320 is configured as a ring interconnect, but may be configured as an interconnect mesh (e.g., 2D mesh).

Claim 5

Exemplary Disclosures Relevant to *Sinharoy* and Related Combinations

Sistla discloses the following at Fig. 3:



Sistla discloses the following at Fig. 4:

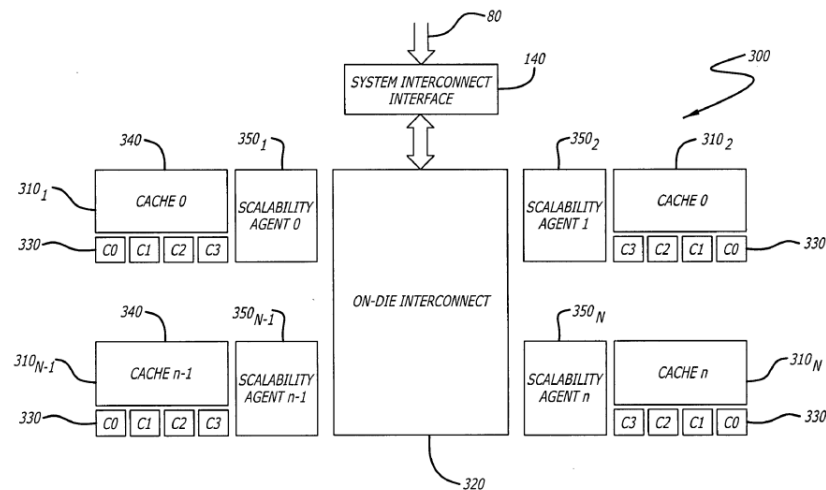


FIG. 4

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 5	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Herrero</i> discloses the following at § 3.1, p. 4: If random spilling is used, data from one CPU in a corner of the chip may end up in the opposite corner and then is probably going to be reused by the original node. In a 4x4 mesh, this means traversing 6 hops per message. In recent architectures like the Intel Larrabee chip multiprocessor [16] this effect is even increased since a ring topology is used. In this case the maximum number of hops would be 8 for a 16 core configuration. This data transfers are going to increase the network traffic unnecessarily, increasing also the energy consumption and access latency. <i>Herrero</i> also discloses the following at § 3.1, p. 4: Figure 5 shows the assignment for a ring network. In this case, the benefit of the proposed technique is much higher since the average node distance is 4.27 for the random distribution (as shown in Figure 6). Our proposed policies, however, have the same average destination distance that we showed for the mesh; 1 hop for the 2 destinations configuration (DAS2n) and 1.5 hops for the 4 destinations configuration (DAS4n). <i>Herrero</i> also discloses the following in Fig. 5: Fig. 5. Distance-Aware Spilling node assignment in a ring network

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 5	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Herrero</i> also notes that “we present the results obtained for a network with the same configuration parameters stated in Table 1 but changing the network topology to a bidirectional ring (as in the Larrabee chip multiprocessor).” <i>Herrero</i> at § 5.2, p. 8. <i>Chrysos</i> discloses at [0045]: Embodiments of the present invention may use any well-known cache coherence protocol for communication and maintaining memory consistency. Many protocols may be layered upon a bidirectional ring interconnect. <i>Chrysos</i> discloses the following at [0018]: Embodiments of the present invention advantageously provide reduced latency and increased bandwidth for an on-chip integration of multiple processors. This may be particularly beneficial in parallel shared-memory applications, Such as transaction processing, data mining, managed run-time environments Such as lava or .net, and web or email serving. <i>See also Chrysos</i> at Abstract. <i>See also Exs. D-1 and D-3</i> at [5].

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 6	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[6] The apparatus of claim 1, wherein the caching agent is to receive an allocation transaction from the IIO module, the allocation transaction received in the IIO module from an IO device coupled to the multicore processor.	<i>Sinharoy</i>, alone or in combination with <i>Agarwal</i>, <i>Berg</i>, <i>Jain</i>, and/or <i>Guthrie</i>, discloses or renders obvious “the caching agent is to receive an allocation transaction from the IIO module, the allocation transaction received in the IIO module from an IO device coupled to the multicore processor.” According to the '228 Patent specification, an allocation transaction is generated by an I/O device to “lodge data directly into the LLC.” '228 Patent at 3:22-25; <i>see also id.</i> at 3:47-51 (“For example, this allocation transaction can be a request to write data directly into a cache memory, e.g., a LLC of the processor, and it can be received from an IO device such as a peripheral device coupled to the processor via a PCIe™ link, for example.”) <i>Sinharoy</i> discloses this feature. <i>Sinharoy</i> discloses a “cache injection” operation that allows an I/O device coupled to the processor to send “a direct memory access (DMA) write operation” (an allocation transaction) that writes data directly into the local L3 cache region. <i>Sinharoy</i> at 17, 22. That is, “[a]n I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory.” <i>Id.</i> at 17. Accordingly, <i>Sinharoy</i>’s “coherence fabric” will receive an allocation transaction (e.g., write request) from one of the integrated I/O controllers when an I/O device seeks to perform a cache injection. <i>Sinharoy</i> discloses the following at 17: While no new cache states have been added for POWER7, new coherent operations are supported. One such operation is cache injection. An I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory. If a given core’s L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME, MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 6	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following at 22: In addition to supporting the interrupt presentation function, pipelined I/O high-throughput mode, and advanced partial DMA write management pioneered in the POWER6 design, the POWER7 I/O controllers also support cache injection, which enables DMA write traffic to be written directly into a cache, instead of to memory. <i>See also supra</i> [1c]. <i>See also</i> Exs. D-1, D-3, and D-4 at [1c] and [6].

Claim 7	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[7] The apparatus of claim 6, wherein the caching agent is to directly store data of the allocation transaction into the shared cache memory and to not initiate a memory transaction to a system memory coupled to the multicore processor.	<i>Sinharoy</i>, alone or in combination with <i>Agarwal</i>, <i>Berg</i>, <i>Jain</i>, and/or <i>Guthrie</i>, discloses or renders obvious “the caching agent is to directly store data of the allocation transaction into the shared cache memory and to not initiate a memory transaction to a system memory coupled to the multicore processor.” As discussed above in [6], <i>Sinharoy</i> discloses a “cache injection” operation that allows an I/O device coupled to the processor to send “a direct memory access (DMA) write operation” (an allocation transaction) that writes data directly into a local L3 cache region “instead of to memory.” <i>See Sinharoy</i> at 17. Thus, <i>Sinharoy</i>’s cache injection operation utilizes the L3 cache controller’s write-back policy to handle memory transactions. <i>See id.</i> at 20-21. <i>Sinharoy</i>’s L3 cache protocol supports a “modified” state, which means the L3 cache uses a write-back policy. <i>See id.</i> at 20-21, Table 2. When a core or I/O device writes new data to a memory address, the L3 controller installs the new data into the cache as a “modified” (“dirty”) state, which indicates the cache has the latest data and the memory has a stale data. The cache writes the modified data back to the memory later, when the line is evicted/cast out, to create a room for another cache line. Such a policy is commonly known as a “write-back” policy. Thus, when an L3 controller receives the I/O “cache injection” write allocation

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 7	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	transaction, it writes the data to the L3 cache but does not send the data to the system memory at that time. A “write-back” cache policy differs from a “write-through” cache policy (e.g., used by <i>Sinharoy</i>’s L1 data caches, <i>see supra</i> [1d]). A “write-through” policy forces the write data out of the cache to the higher-level memory (e.g., system memory) right away. But because <i>Sinharoy</i>’s L3 cache uses a write-back policy, and because data written by the I/O device via cache injection is written directly to the L3 cache, the I/O write data is installed into the L3 cache as a “modified” (“dirty”) state. Subsequent core accesses to the same data find a “hit” (i.e., locate the data) in the L3 cache, such that those core accesses do not need to access the main memory. <i>See Sinharoy</i> at 17-21, Table 2, Fig. 10. Thus, for I/O write transactions (allocation transactions), <i>Sinharoy</i>’s caching agent (the “coherence fabric,” including the eight L2 controllers and eight L3 controllers distributed among the cores) directly stores the write data in the shared L3 cache and does not initiate system memory transaction. <i>Sinharoy</i> discloses the following at 17 (citation omitted): As shown in Table 2, the POWER7 L2 and L3 caches support the same 13-cache-state protocol as the POWER6 design point. While no new cache states have been added for POWER7, new coherent operations are supported. One such operation is cache injection. An I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory. If a given core’s L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME, MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 7	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following at 20: The replacement algorithm serves as a primary heuristic in creating this balance. It categorizes all lines in the L3 cache into the following two classes: 1. Those installed as victims via a cast-out operation by the associated L2 cache. 2. Those installed as victims via a lateral cast-out operation by one of the other seven L3 regions, those that are residual shared copies of lines that have been moved to the local L2 cache, and those that are invalid. <i>Sinharoy</i> discloses the following at 22: In addition to supporting the interrupt presentation function, pipelined I/O high-throughput mode, and advanced partial DMA write management pioneered in the POWER6 design, the I/O controllers also support cache injection, which enables DMA write traffic to be written directly into a cache, instead of to memory.
Claim 11	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[11pre] A method comprising:	<i>Sinharoy</i> discloses or suggests “a method.” For example, <i>Sinharoy</i> discloses an I/O cache injection operation that allows an I/O device to write data directly into the L3 cache. <i>See Sinharoy</i> at 17, 22. <i>Sinharoy</i> discloses the following at 17: Therefore, the POWER7 cache hierarchy includes a shared 32-MB L3 cache comprised of the 4-MB local L3 regions from the eight cores that reside on the processor chip. This enables the signaling resources that would have been used for communication with an off-chip L3 cache to be redeployed for other purposes, such as the memory and I/O subsystem interfaces.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 11	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following at 17: As shown in Table 2, the POWER7 L2 and L3 caches support the same 13-cache-state protocol as the POWER6 design point [4]. While no new cache states have been added for POWER7, new coherent operations are supported. One such operation is cache injection. An I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory. If a given core's L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME, MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region. <i>Sinharoy</i> discloses the following at 22: These interfaces operate at frequencies ranging from 1.25 to 2.5 GHz, depending on the I/O hub chips to which they are connected. In addition to supporting the interrupt presentation function, pipelined I/O high-throughput mode, and advanced partial DMA write management pioneered in the POWER6 design, the POWER7 I/O controllers also support cache injection, which enables DMA write traffic to be written directly into a cache, instead of to memory.
[11a] receiving an allocation transaction in an integrated input/output (IIO) module of a multicore processor from an IO device coupled to the multicore processor, the allocation transaction comprising a request to directly store data of the allocation transaction into a shared cache memory of the multicore processor;	<i>Sinharoy</i>, alone or in combination with <i>Agarwal</i>, <i>Berg</i>, <i>Jain</i>, and/or <i>Guthrie</i>, discloses or renders obvious “receiving an allocation transaction in an integrated input/output (IIO) module of a multicore processor from an IO device coupled to the multicore processor, the allocation transaction comprising a request to directly store data of the allocation transaction into a shared cache memory of the multicore processor.” <i>See supra</i> [1c] (discussing the IIO module); <i>see also supra</i> [6] (“receiv[ing] an allocation transaction from the IIO module, the allocation transaction received in the IIO module from an IO device coupled to the multicore processor”); <i>see also supra</i> [7] (“directly store data of the allocation transaction into the shared cache memory”). <i>See also</i> Exs. D-1, D-3, and D-4 at [11a].

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 11	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[11b] forwarding the allocation transaction to a caching agent of the multicore processor, the caching agent shared by a plurality of cores of the multicore processor and the IIO module; and	<i>Sinharoy</i>, alone or in combination with <i>Agarwal</i>, <i>Berg</i>, <i>Jain</i>, and/or <i>Guthrie</i>, discloses or renders obvious “forwarding the allocation transaction to a caching agent of the multicore processor, the caching agent shared by a plurality of cores of the multicore processor and the IIO module.” As discussed above in [6], <i>Sinharoy</i>’s multicore processor supports the “cache injection” operation that allows an I/O device coupled to the processor to send “a direct memory access (DMA) write operation” (an allocation transaction) to an L3 cache controller (part of the distributed “coherence fabric” that maps to the claimed “caching agent”) that writes data directly into the local L3 cache region. That is, <i>Sinharoy</i>’s coherence fabric (i.e., “caching agent”) receives a cache injection write operation (i.e., “allocation transaction”), which has been forwarded by an integrated I/O controller (i.e., “IIO module”) to the coherence fabric so it can be received. <i>See Sinharoy</i> at 17, 22; <i>see also supra</i> [6]. Further, as discussed for [1d] and [1e] above, <i>Sinharoy</i> discloses or suggests a single caching agent with distributed portions (i.e., a “coherence fabric,” including eight L2 controllers and eight L3 controllers distributed among the cores) that collectively maintain cache coherency for the plurality of cores and IIO module. <i>See also</i> Exs. D-1, D-3, and D-4 at [11b].
[11c] directly storing the data of the allocation transaction into the shared cache memory of the multicore processor.	<i>Sinharoy</i> discloses or suggests “directly storing the data of the allocation transaction into the shared cache memory of the multicore processor.” As discussed above in [7], <i>Sinharoy</i>’s “cache injection” operation allows an I/O device coupled to the processor to send “a direct memory access (DMA) write operation” (an allocation transaction) that writes data directly into the local L3 cache region “instead of to memory.” <i>See Sinharoy</i> at 17; <i>see also supra</i> at [7].

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 12	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[12] The method of claim 11, further comprising accessing the data in the shared cache memory via at least one of the plurality of cores of the multicore processor a number of times, without sending a transaction to a system memory coupled to the multicore processor.	<i>Sinharoy</i> discloses or suggests “accessing the data in the shared cache memory via at least one of the plurality of cores of the multicore processor a number of times, without sending a transaction to a system memory coupled to the multicore processor.” As discussed above in [7] and [11], <i>Sinharoy</i>’s “cache injection” operation allows an I/O device to write data directly into the local L3 cache region, instead of to memory. (<i>See Sinharoy</i> at 17; <i>see also supra</i> [7], [11].) Thus, <i>Sinharoy</i>’s cache injection operation utilizes the L3 cache controller’s write-back policy to handle memory transactions. <i>See Sinharoy</i> at 20-21. <i>Sinharoy</i> also discloses (consistent with the cache operation), that when a core needs to access a particular data location, the core checks first in its L2 cache, then in its local L3 cache, and then in the “remaining 28 MB of the on-chip L3 cache” – all without accessing system memory. <i>See Sinharoy</i> at 19, Fig. 10. <i>Sinharoy</i>’s L3 cache protocol supports a “modified” state, which means the L3 cache uses a write-back policy. <i>See id.</i> at 20-21, Table 2. When a core or I/O device writes new data to a memory address, the L3 controller installs the new data into the cache as a “modified” (“dirty”) state, which indicates the cache has the latest data and the memory has a stale data. The cache writes the modified data back to the memory later, when the line is evicted/cast out, to create a room for another cache line. Such a policy is commonly known as a “write-back” policy. Thus, when an L3 controller receives the I/O “cache injection” write allocation transaction, it writes the data to the L3 cache but does not send the data to the system memory at that time. A “write-back” cache policy differs from a “write-through” cache policy (e.g., used by <i>Sinharoy</i>’s L1 data caches, <i>see supra</i> [1d]). A “write-through” policy forces the write data out of the cache to the higher-level memory (e.g., system memory) right away. But because <i>Sinharoy</i>’s L3 cache uses a write-back policy, and because data written by the I/O device via cache injection is written directly to the L3 cache, the I/O write data is installed into the L3 cache as a “modified” (“dirty”) state. Subsequent core accesses to the same data find a “hit” (i.e., locate the data) in the L3 cache, such that those core accesses do not need to access the main memory. <i>See Sinharoy</i> at 17-21, Table 2, Fig. 10.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 12	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	Accordingly, a core that wants to access the I/O data injected into <i>Sinharoy</i>'s shared L3 cache will access that data without sending a transaction to system memory. In operation, this will happen many times, not just once. <i>Sinharoy</i> discloses the following at 17: [N]ew coherent operations are supported. One such operation is cache injection. An I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory. If a given core's L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME, MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region. <i>Sinharoy</i> discloses the following at 17 (citation omitted): As shown in Table 2, the POWER7 L2 and L3 caches support the same 13-cache-state protocol as the POWER6 design point.

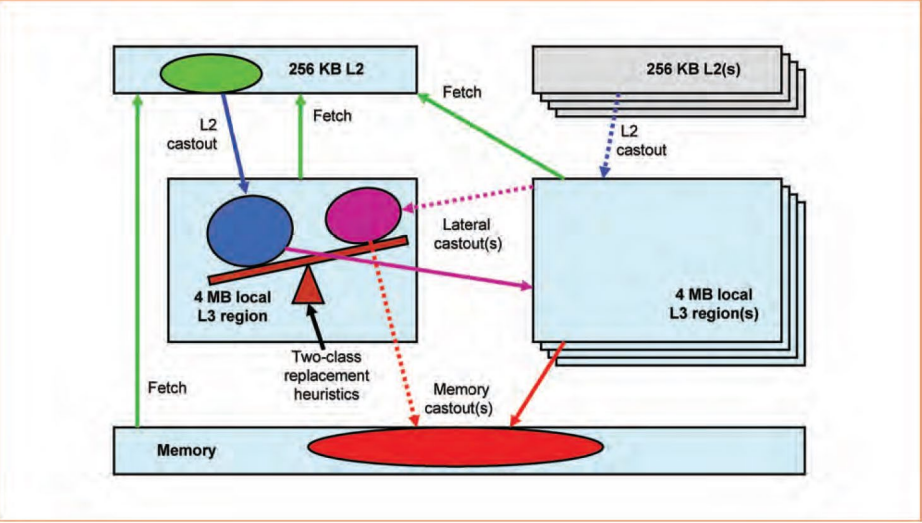
Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 12	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>																																																																																																		
	<i>Sinharoy</i> discloses the following at Table 2: Table 2 IBM POWER7 cache states. <table><tr><th>State</th><th>Description</th><th>Authority</th><th>Sharers and scope</th><th>Source data</th><th>Data cast-out</th><th>Scope cast-out</th></tr><tr><td>I</td><td>Invalid</td><td>None</td><td>N/A</td><td>N/A</td><td>N/A</td><td>None</td></tr><tr><td>ID</td><td>Deleted, do not allocate</td><td>None</td><td>N/A</td><td>N/A</td><td>N/A</td><td>None</td></tr><tr><td>S</td><td>Shared</td><td>Read</td><td>Yes, scope unknown</td><td>No</td><td>No</td><td>None</td></tr><tr><td>SL</td><td>Shared, local data source</td><td>Read</td><td>Yes, scope unknown</td><td>At request</td><td>No</td><td>None</td></tr><tr><td>T</td><td>Formerly MU, now shared</td><td>Update</td><td>Yes, probably global</td><td>If notified</td><td>Yes</td><td>Required, global</td></tr><tr><td>TE</td><td>Formerly ME, now shared</td><td>Update</td><td>Yes, probably global</td><td>If notified</td><td>No</td><td>Required, global</td></tr><tr><td>M</td><td>Modified, avoid sharing</td><td>Update</td><td>No</td><td>At request</td><td>Yes</td><td>Optional, local</td></tr><tr><td>ME</td><td>Exclusive</td><td>Update</td><td>No</td><td>At request</td><td>No</td><td>None</td></tr><tr><td>MU</td><td>Modified, bias toward sharing</td><td>Update</td><td>No</td><td>At request</td><td>Yes</td><td>Optional, local</td></tr><tr><td>IG</td><td>Invalid, cached scope-state</td><td>None</td><td>N/A, probably global copies</td><td>N/A</td><td>N/A</td><td>Required, global</td></tr><tr><td>IN</td><td>Invalid, scope predictor</td><td>None</td><td>N/A, probably local copies</td><td>N/A</td><td>N/A</td><td>None</td></tr><tr><td>TN</td><td>Formerly MU, now shared</td><td>Update</td><td>Yes, local</td><td>If notified</td><td>Yes</td><td>Optional, local</td></tr><tr><td>TEN</td><td>Formerly ME, now shared</td><td>Update</td><td>Yes, local</td><td>If notified</td><td>No</td><td>None</td></tr></table>	State	Description	Authority	Sharers and scope	Source data	Data cast-out	Scope cast-out	I	Invalid	None	N/A	N/A	N/A	None	ID	Deleted, do not allocate	None	N/A	N/A	N/A	None	S	Shared	Read	Yes, scope unknown	No	No	None	SL	Shared, local data source	Read	Yes, scope unknown	At request	No	None	T	Formerly MU, now shared	Update	Yes, probably global	If notified	Yes	Required, global	TE	Formerly ME, now shared	Update	Yes, probably global	If notified	No	Required, global	M	Modified, avoid sharing	Update	No	At request	Yes	Optional, local	ME	Exclusive	Update	No	At request	No	None	MU	Modified, bias toward sharing	Update	No	At request	Yes	Optional, local	IG	Invalid, cached scope-state	None	N/A, probably global copies	N/A	N/A	Required, global	IN	Invalid, scope predictor	None	N/A, probably local copies	N/A	N/A	None	TN	Formerly MU, now shared	Update	Yes, local	If notified	Yes	Optional, local	TEN	Formerly ME, now shared	Update	Yes, local	If notified	No	None
State	Description	Authority	Sharers and scope	Source data	Data cast-out	Scope cast-out																																																																																													
I	Invalid	None	N/A	N/A	N/A	None																																																																																													
ID	Deleted, do not allocate	None	N/A	N/A	N/A	None																																																																																													
S	Shared	Read	Yes, scope unknown	No	No	None																																																																																													
SL	Shared, local data source	Read	Yes, scope unknown	At request	No	None																																																																																													
T	Formerly MU, now shared	Update	Yes, probably global	If notified	Yes	Required, global																																																																																													
TE	Formerly ME, now shared	Update	Yes, probably global	If notified	No	Required, global																																																																																													
M	Modified, avoid sharing	Update	No	At request	Yes	Optional, local																																																																																													
ME	Exclusive	Update	No	At request	No	None																																																																																													
MU	Modified, bias toward sharing	Update	No	At request	Yes	Optional, local																																																																																													
IG	Invalid, cached scope-state	None	N/A, probably global copies	N/A	N/A	Required, global																																																																																													
IN	Invalid, scope predictor	None	N/A, probably local copies	N/A	N/A	None																																																																																													
TN	Formerly MU, now shared	Update	Yes, local	If notified	Yes	Optional, local																																																																																													
TEN	Formerly ME, now shared	Update	Yes, local	If notified	No	None																																																																																													
	<i>Sinharoy</i> discloses the following at 18: Another such enhancement involves the L3 cache management policy. While the POWER7 L2 replacement policy is similar to the POWER6 policy, employing an enhanced pseudo-LRU replacement algorithm, with biasing toward various invalid locations, the POWER7 L3 replacement policy incorporates significant enhancements and exploits the semantic content of the 13 cache states to manage the aggregation of the 4-MB regions. This feature is described in further detail in the L3 cache section.																																																																																																		

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 12	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following at 19: Each write machine has an associated cast-out machine to manage the eviction of lines displaced by writes. Note that these write machines are also utilized by other operations such as cast-outs and cache-injections. <i>Sinharoy</i> discloses the following at 19: 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 cache. Therefore, operations that miss the 4-MB local L3 region but hit in the remaining 28 MB of the L3 cache access the cache via the snoop dispatch ports. L2 cast-out operations access the 4-MB local L3 region via the core/L2 dispatch port, whereas lateral cast-outs access a given 4-MB L3 region via the lateral L3 region dispatch port (see targeted cases below) or via the snoop dispatch ports (see state merge cases below). All of the cast-out scenarios are described in detail in the subsequent paragraphs. <i>Sinharoy</i> discloses the following at 21: 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. If the evicted line is first class, a set of heuristics weigh the cache state and the usage pattern (capacity) of the evicting cache to determine whether to laterally cast-out the line to another L3 region or to immediately write it back to memory (if it is “dirty”). If the line is laterally cast-out, another set of heuristics determine which of the other seven L3 regions to target based upon past history, reference pattern, and system firmware control. L3 cast-out operations are managed in the L3 region by a pool of ten L3 cast-out machines.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 13	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[13] The method of claim 12, further comprising sending data of the cache line to the system memory from the shared cache memory if the cache line is evicted from the shared cache memory, when the cache line includes modified data.	<i>Sinharoy</i> discloses or suggests “sending data of the cache line to the system memory from the shared cache memory if the cache line is evicted from the shared cache memory, when the cache line includes modified data.” As discussed above in [12], <i>Sinharoy</i>’s L3 cache uses write-back policy and marks the modified lines as “modified” state (also known as “dirty” state). <i>See Sinharoy</i> at 20-21, Table 2. According to <i>Sinharoy</i>, if the evicted (cast-out) cache line is dirty (i.e., in the “modified” state), the data of the cache line is written back to system memory. <i>See id.</i> at 19, 21; <i>see also supra</i> [12]. <i>Sinharoy</i>’s Figure 10 shows the cast-out (evicted / modified) lines being written to system memory:  Figure 10 L3 region behavior.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 13	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>																																																																																																		
	<i>Sinharoy</i> discloses the following at 17: [N]ew coherent operations are supported. One such operation is cache injection. An I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory. If a given core’s L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME, MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region. <i>Sinharoy</i> discloses the following at Table 2: Table 2 IBM POWER7 cache states. <table><tr><th>State</th><th>Description</th><th>Authority</th><th>Sharers and scope</th><th>Source data</th><th>Data cast-out</th><th>Scope cast-out</th></tr><tr><td>I</td><td>Invalid</td><td>None</td><td>N/A</td><td>N/A</td><td>N/A</td><td>None</td></tr><tr><td>ID</td><td>Deleted, do not allocate</td><td>None</td><td>N/A</td><td>N/A</td><td>N/A</td><td>None</td></tr><tr><td>S</td><td>Shared</td><td>Read</td><td>Yes, scope unknown</td><td>No</td><td>No</td><td>None</td></tr><tr><td>SL</td><td>Shared, local data source</td><td>Read</td><td>Yes, scope unknown</td><td>At request</td><td>No</td><td>None</td></tr><tr><td>T</td><td>Formerly MU, now shared</td><td>Update</td><td>Yes, probably global</td><td>If notified</td><td>Yes</td><td>Required, global</td></tr><tr><td>TE</td><td>Formerly ME, now shared</td><td>Update</td><td>Yes, probably global</td><td>If notified</td><td>No</td><td>Required, global</td></tr><tr><td>M</td><td>Modified, avoid sharing</td><td>Update</td><td>No</td><td>At request</td><td>Yes</td><td>Optional, local</td></tr><tr><td>ME</td><td>Exclusive</td><td>Update</td><td>No</td><td>At request</td><td>No</td><td>None</td></tr><tr><td>MU</td><td>Modified, bias toward sharing</td><td>Update</td><td>No</td><td>At request</td><td>Yes</td><td>Optional, local</td></tr><tr><td>IG</td><td>Invalid, cached scope-state</td><td>None</td><td>N/A, probably global copies</td><td>N/A</td><td>N/A</td><td>Required, global</td></tr><tr><td>IN</td><td>Invalid, scope predictor</td><td>None</td><td>N/A, probably local copies</td><td>N/A</td><td>N/A</td><td>None</td></tr><tr><td>TN</td><td>Formerly MU, now shared</td><td>Update</td><td>Yes, local</td><td>If notified</td><td>Yes</td><td>Optional, local</td></tr><tr><td>TEN</td><td>Formerly ME, now shared</td><td>Update</td><td>Yes, local</td><td>If notified</td><td>No</td><td>None</td></tr></table> <i>Sinharoy</i> discloses the following at 19: Each write machine has an associated cast-out machine to manage the eviction of lines displaced by writes. Note that these write machines are also utilized by other operations such as cast-outs and cache-injections.	State	Description	Authority	Sharers and scope	Source data	Data cast-out	Scope cast-out	I	Invalid	None	N/A	N/A	N/A	None	ID	Deleted, do not allocate	None	N/A	N/A	N/A	None	S	Shared	Read	Yes, scope unknown	No	No	None	SL	Shared, local data source	Read	Yes, scope unknown	At request	No	None	T	Formerly MU, now shared	Update	Yes, probably global	If notified	Yes	Required, global	TE	Formerly ME, now shared	Update	Yes, probably global	If notified	No	Required, global	M	Modified, avoid sharing	Update	No	At request	Yes	Optional, local	ME	Exclusive	Update	No	At request	No	None	MU	Modified, bias toward sharing	Update	No	At request	Yes	Optional, local	IG	Invalid, cached scope-state	None	N/A, probably global copies	N/A	N/A	Required, global	IN	Invalid, scope predictor	None	N/A, probably local copies	N/A	N/A	None	TN	Formerly MU, now shared	Update	Yes, local	If notified	Yes	Optional, local	TEN	Formerly ME, now shared	Update	Yes, local	If notified	No	None
State	Description	Authority	Sharers and scope	Source data	Data cast-out	Scope cast-out																																																																																													
I	Invalid	None	N/A	N/A	N/A	None																																																																																													
ID	Deleted, do not allocate	None	N/A	N/A	N/A	None																																																																																													
S	Shared	Read	Yes, scope unknown	No	No	None																																																																																													
SL	Shared, local data source	Read	Yes, scope unknown	At request	No	None																																																																																													
T	Formerly MU, now shared	Update	Yes, probably global	If notified	Yes	Required, global																																																																																													
TE	Formerly ME, now shared	Update	Yes, probably global	If notified	No	Required, global																																																																																													
M	Modified, avoid sharing	Update	No	At request	Yes	Optional, local																																																																																													
ME	Exclusive	Update	No	At request	No	None																																																																																													
MU	Modified, bias toward sharing	Update	No	At request	Yes	Optional, local																																																																																													
IG	Invalid, cached scope-state	None	N/A, probably global copies	N/A	N/A	Required, global																																																																																													
IN	Invalid, scope predictor	None	N/A, probably local copies	N/A	N/A	None																																																																																													
TN	Formerly MU, now shared	Update	Yes, local	If notified	Yes	Optional, local																																																																																													
TEN	Formerly ME, now shared	Update	Yes, local	If notified	No	None																																																																																													

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 13	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> also discloses the following at 21: 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. If the evicted line is first class, a set of heuristics weigh the cache state and the usage pattern (capacity) of the evicting cache to determine whether to laterally cast-out the line to another L3 region or to immediately write it back to memory (if it is “dirty”).

Claim 14	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
[14] The method of claim 12, further comprising directly storing the data of the allocation transaction into the shared cache memory without communicating with the system memory.	<i>Sinharoy</i> discloses or suggests “directly storing the data of the allocation transaction into the shared cache memory without communicating with the system memory.” As discussed above in [11] and [12], <i>Sinharoy</i>’s “cache injection” operation allows an I/O device to write data directly into the local L3 cache region, instead of to memory. <i>See Sinharoy</i> at 17, 22; <i>see also supra</i> [11], [12]. <i>Sinharoy</i> discloses the following at 17: [N]ew coherent operations are supported. One such operation is cache injection. An I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory. If a given core’s L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME, MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 14	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following at 19: Each write machine has an associated cast-out machine to manage the eviction of lines displaced by writes. Note that these write machines are also utilized by other operations such as cast-outs and cache-injections. <i>Sinharoy</i> discloses the following at 19: 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 cache. Therefore, operations that miss the 4-MB local L3 region but hit in the remaining 28 MB of the L3 cache access the cache via the snoop dispatch ports. L2 cast-out operations access the 4-MB local L3 region via the core/L2 dispatch port, whereas lateral cast-outs access a given 4-MB L3 region via the lateral L3 region dispatch port (see targeted cases below) or via the snoop dispatch ports (see state merge cases below). All of the cast-out scenarios are described in detail in the subsequent paragraphs. <i>Sinharoy</i> discloses the following at 21: 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. If the evicted line is first class, a set of heuristics weigh the cache state and the usage pattern (capacity) of the evicting cache to determine whether to laterally cast-out the line to another L3 region or to immediately write it back to memory (if it is “dirty”). If the line is laterally cast-out, another set of heuristics determine which of the other seven L3 regions to target based upon past history, reference pattern, and system firmware control. L3 cast-out operations are managed in the L3 region by a pool of ten L3 cast-out machines.

Appendix 228-02: Invalidity of U.S. Patent No. US 8,984,228 (the '228 Patent) Based on *Sinharoy*

Claim 14	<u>Exemplary Disclosures Relevant to <i>Sinharoy</i> and Related Combinations</u>
	<i>Sinharoy</i> discloses the following at 22: In addition to supporting the interrupt presentation function, pipelined I/O high-throughput mode, and advanced partial DMA write management pioneered in the POWER6 design, the POWER7 I/O controllers also support cache injection, which enables DMA write traffic to be written directly into a cache, instead of to memory.