



Mercedes's Hearing Demonstratives for *Mercedes vs. Daedalus*

IPR2023-01344

U.S. Patent No. 9,575,895

Mercedes EX1021
Mercedes-Benz USA, LLC
v. Daedalus Prime LLC

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

IPR2023-01344

Table of Contents



Introduction

Dispute 1 – Whether Sinharoy discloses the “single caching agent” limitation

Dispute 2 – Whether Tang discloses the “single caching agent” limitation

Dispute 3 – Whether Tang discloses the “distributed portions” limitation

Dispute 4 – Whether Xu was publicly available before December 13, 2011

Introduction

Mercedes EX1021
Mercedes-Benz USA, LLC
v. Daedalus Prime LLC

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

IPR2023-01344

- The '895 Patent relates to a multicore processor computer system having a plurality of cores and an integrated input/out (I/O) module.
- The I/O module interfaces between the multicore processor and an I/O device.
- The multicore processor further includes a caching agent to perform cache coherency operations for the plurality of cores and the I/O module.

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

4

Overview of the '895 Patent



1. A processor comprising:
a plurality of cores, a shared cache memory, a memory controller to interface with a memory coupled to the processor, an integrated input/output (IIO) module to interface between the processor and an IO device coupled to the processor and a caching agent to perform cache coherency operations for the plurality of cores and the IIO module, wherein the processor is to receive an allocation transaction from the IO device and directly store data of the allocation transaction into the shared cache memory, wherein the caching agent is a single caching agent for the processor and includes a plurality of distributed portions each associated with a corresponding one of the plurality of cores.

Claim 1 generally recites a processor including:

- A plurality of cores
- A shared cache memory
- A memory controller
- An integrated input/output (IIO) module to interface between the processor and an IO device
- A caching agent to perform cache coherency operations for the plurality of cores and the IIO module
- The caching agent is a single caching agent for the processor
- The caching agent includes a plurality of distributed portions each associated with a corresponding core

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

“Single Caching Agent” in the '895 Patent



- The '895 Patent broadly describes a “single caching agent” for the processor.
- It does not discuss a particular structure or circuit implementation for the “single caching agent.”
- It does, however, indicate the caching agent can be formed from distributed portions and that it “may be logic.”

Referring now to FIG. 2, shown is a block diagram of a caching agent in accordance with an embodiment of the present invention. As shown in FIG. 2, caching agent 200 may be logic interposed between one or more cores of a processor and a LLC 230. In addition, caching agent 200

'895 Patent at 2:41-45

Note that although a single structure is shown in FIG. 2 for ease of illustration, understand that a caching agent can be distributed such that each of different portions of the caching agent can be associated with a corresponding core and LLC bank or slice.

Id. at 3:6-10

"Single Caching Agent" in the '895 Patent



- In FIG. 6, the processor 700 includes a "distributed configuration" having "partitions or slices" each including a core 710 and a partition of a caching agent 715 and a[n] LLC 720.
- "Note that while distributed caching agents are shown, understand that these distributed portions **form** a single cache agent, and which is configured to handle cache coherency operations **both for the cores as well as an I/O module 750.**"

'895 Patent at 6:30-35

Id. at 6:35-40 (emphasis added)

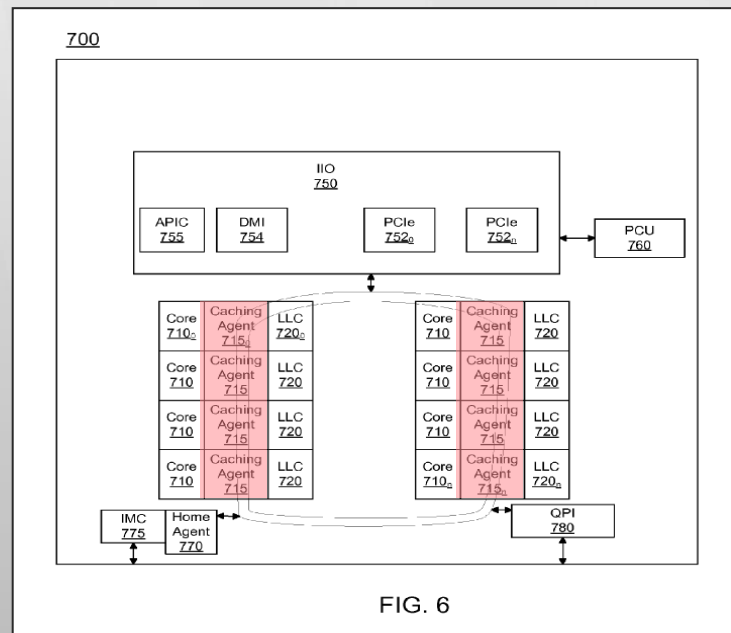


FIG. 6

'895 Patent at Fig. 6 (annotated)

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Dispute 1 – Whether Sinharoy Discloses the “Single Caching Agent” Limitation

Mercedes EX1021
Mercedes-Benz USA, LLC
v. Daedalus Prime LLC

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE IPR2023-01344

Overview of Sinharoy

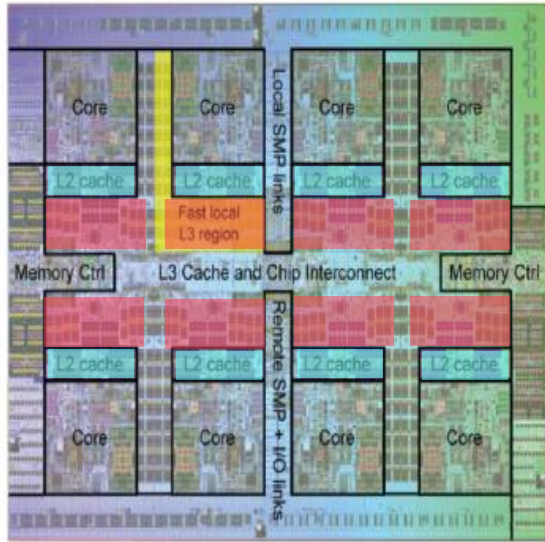


Figure 1

Die photo of the IBM POWER7 chip.

Sinharoy at Fig. 1 (annotated)

- Sinharoy discloses a “POWER7” multicore server processor.
- Fig. 1 depicts a die photo of the IBM POWER7 processor, which includes:
 - Eight cores (Sinharoy at 1:2)
 - A “shared 32-MB L3 cache” comprising “eight 4-MB L3 regions” (*id.* at 1:18)
 - Two memory controllers that interface with DDR memory chips (*id.* at 1:2, 21)
 - Two “integrated I/O controllers” that interface between the processor and “IO devices” including I/O hub chips (*id.* at 1:22)
 - Centralized controllers for the L2 and L3 caches, which form a “single caching agent” performing “cache coherency operations” (*id.* at 1:18-19)

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Overview of Sinharoy



IBM POWER7 multicore server processor

The IBM POWER[®] processor is the dominant reduced instruction set computing microprocessor in the world today, with a rich history of implementation and innovation over the last 20 years. In this paper, we describe the key features of the POWER7 processor chip. 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 cache 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.

B. Sinharoy
R. Kalla
W. J. Starke
H. Q. Lu
R. Cognigni
J. A. Van Norstrand
B. J. Rinchetti
J. Stuechli
J. Luenstra
G. L. Guthrie
D. G. Nguyen
B. Elmer
C. F. Marino
E. Rether
P. Williams

Introduction

Over the years, IBM POWER[®] processors [1–4] have introduced reduced instruction set computing architecture, advanced branch prediction, out-of-order execution, data prefetching, multithreading, dual-core chip, core accelerators [5], on-chip high-bandwidth memory controller, and highly scalable symmetric multiprocessing (SMP) interconnect. In this seventh-generation POWER processor [6–9], IBM introduces a balanced, eight-core multiprocessor design, with large on-chip embedded dynamic random access memory (eDRAM) caches, and high-performance four-way multithreaded cores, implementing IBM PowerPC[®] architecture version 2.06 [10].

Starting in 2001 with POWER4[®] in 180-nm technology up to POWER6[®] in 65-nm, spanning four technology generations, all POWER processors were designed with two cores on the processor chip. The emphasis had been to stay with two cores per chip and significantly improve per-core performance by improving the core and cache microarchitectures, as well as exploiting the socket resources by only two cores on the chip. The socket resources that have an impact on performance include chip power, various off-chip bandwidths such as input/output (I/O), memory, and SMP bandwidths, as well as socket-level cache and

memory capacity. As we add more cores to share the existing socket resources in a given technology generation, per-core performance can be negatively affected. Unlike previous generation POWER processors, in POWER7[®], four times as many cores share the socket resources. Along with significant slowdown in the performance improvements from the silicon technology, this makes it more challenging to significantly improve the per-core performance of POWER7 over POWER6.

The goal of POWER7 was to improve the socket-level, core-level, and thread-level performance significantly over POWER6 while achieving all of the following in one technology generation.

- Reduce the core area and power to such an extent that four times as many cores can be placed on the processor chip in the same power envelope as that of POWER6. To reduce power, processor frequency is reduced in POWER7, while higher performance is achieved through much more emphasis on microarchitecture improvements, such as aggressive out-of-order execution, advanced four-way simultaneous multithreading (SMT), advanced branch prediction, advanced prefetching, and cache and memory latency reduction and bandwidth improvements.
- Fit the POWER7 chip in the same socket as POWER6 and utilize the same SMP and I/O buses as in POWER6

Digital Object Identifier: 10.1145/1203120.1212736

©Copyright 2011 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the journal reference and IBM copyright notice are included on the first page. This title and abstract, however, may not be copied for any purpose or disclosed publicly without written permission by International Business Machines Corporation. Permission to reproduce this document may be obtained from the publisher.

0378-0064/11/0600 0000 © 2011 IBM

IBM J. RES. & DEV., VOL. 55, NO. 3, PAPER 3, MAY/JUNE 2011

B. SINHAROY ET AL.: 1:1

- The POWER7 multicore processor supports “cache coherency operations” such as:

- Fetches

- Read-write operations

- Core prefetch operations (Sinharoy at 1:18-19)

- It also supports “cache injection” as a “coherent operation,” which “enables DMA write traffic to be written directly into a cache, instead of to memory.” (Id. at 1:17, 22)

- Sinharoy also discloses a system which combines up to four POWER7 chips. (Id. at 1:23)

Mercedes EX1021

Mercedes-Benz USA, LLC

v. Daedalus Prime LLC

IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

The “Single Caching Agent” Limitation



Claim 1 recites “a caching agent to perform cache coherency operations for the plurality of cores and the I/O module . . ., wherein the caching agent is a single caching agent for the processor . . .”

Sinharoy Discloses a Single Caching Agent



- The L2 cache includes a “single centralized controller” and each L3 cache region includes a “single centralized controller.” (Sinharoy at 1:18-19)
- Both L2 and L3 cache controllers have read/claim (RC) machines, which manage “cache coherence operations,” such as:
 - core fetches
 - read-modify-write operations
 - core prefetch operations

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.

Sinharoy at 1:18

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Sinharoy Discloses a Single Caching Agent



The L2 and L3 cache controllers in Sinharoy work with each other.

sockets.” (*Id.*, 1:18-19.) These RC machines are shared across all L3 regions and hence are part of a logical single cache controller. In my opinion, if the L3 regions had separate independent cache controllers, they would each need their own RC machines and there would be no opportunity to globally reorder the storage operations. The RC machines in both the L2 and L3 cache controllers also further support that the L2 and L3 cache controllers work with each other and share information and logic across cores in certain scenarios. Thus, there is not an independent cache controller per core and the shared and portioned parts of the cache controllers form a logical “single caching agent.”

EX1017 (Horst Supp. Decl.)
at ¶18

Mercedes EX1021

Sinharoy Discloses a Single Caching Agent



- Apart from RC machines, cache controllers for L3 cache regions also include write machines, which manage “cache coherence operations” such as:
 - Cast-outs and
 - Cache injections
- “Cache injection” operation is a “cache coherency operation” for the I/O module.
- The cache controllers for the L2 and L3 caches, including RC and write machines, form a logical “single caching agent.”

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.

Sinharoy at 1:19

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Knowledge by Other L3 Caches Shows Cache Controllers Work Together



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 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. When the associated core and L2 are enabled, a number of heuristics cooperate to give precedence to the associated L2 cache while allowing load and capacity balancing among all eight L3 regions on the chip, thereby causing them to behave as a highly set-associative (64-way) adaptive 32-MB L3 cache.

Sinharoy at 1:19-20

to the L3 region.” (*Id.* (emphasis added).) A POSITA would have understood that the reason why the remaining L3 regions still enjoy full access to the disabled L3 region is that the other cache controllers still have knowledge about the disabled L3 cache region. Thus, they do not operate independently from each other.

Following that, Sinharoy discloses “[w]hen the associated core and L2 are enabled, a number of heuristics² cooperate to give precedence to the associated L2 cache while allowing load and capacity balancing among all eight L3 regions on the chip, thereby causing them to behave as a highly set-associative (64-way) adaptive 32-MB L3 cache.” (*Id.* at 1:20.) As I have explained during my deposition, the load and capacity balancing means that there is a single “centralized controller for all the caches.” (*See* EX2001, 17:5-8.) The single cache controller includes the distributed portions of cache controllers of the L2 and L3 caches. (*See* EX1007, ¶¶55-58.)

Mercedes EX1021

EX1017 (Horst Supp. Decl.) at ¶16

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Sinharoy Discloses a Single Caching Agent



Patent Owner argues:

Because Sinharoy's L2 and L3 caches have structurally individual cache controllers, there is no "single cache controller."

But:

- The Petition states that "the caching agent, comprising the centralized cache controllers for the L2 and L3 caches including RC and write machines, is formed as a single caching agent for the processor."
- These cache controllers for the L2 caches and L3 cache regions, including RC and write machines, form one logical "single caching agent."

Sinharoy Discloses a Single Caching Agent



Patent Owner also argues:

Because there are some snooping operations in the L2 and L3 caches in Sinharoy, the cache controllers for the L2 and L3 caches work independently and do not form a “single caching agent.”

But:

- There are at least three scenarios where snooping operations are avoided in Sinharoy, such as:
 - the difference between the cache states for T and TEN (Sinharoy at 1:17)
 - “when the core and L2 associated with a given L3 region are disabled, the remaining L3 regions enjoy full access to the L3 region . . .” (*id.* at 1:19)
 - “snooped by . . . possibly other L3 caches” (*id.*)
- Because snooping operations can be avoided in these instances, the cache controllers for the L2 and L3 caches work together and form a logical “single caching agent.”

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Snooping Can Be Avoided in at Least the TEN State, Showing the Cache Controllers Work Together



Table 2 explains 13 cache-state protocol used in Sinharoy's cache "coheren[ce] operations"

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	Yes, probably global		
SL	Shared, local data source	Read	Yes, scope unknown			
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	Yes, local		
TN	Formerly MU, now shared	Update	Yes, local			
TEN	Formerly ME, now shared	Update	Yes, local	If notified	No	None

the "T" state (Description, "Formerly MU, now shared"), Sinharoy specifies "Yes, probably global." (*Id.*, Table 2.) In my opinion, the "probably global" means that when a cache line is in this state and is potentially shared with other L3 cache regions, accesses to the local L3 regions need to snoop the remote L3 regions. Meanwhile, for the cache state "TEN," (Description, "Formerly ME, now shared"), it specifies "Yes, local." (*Id.*)³ In my opinion, the "local" means that the targeted cache line is shared but it is known that the other non-local cores have not modified that cache line, so the local L3 cache region does not need to snoop other regions. The difference between the cache states "T" and "TEN" shows that snooping can be avoided in certain scenarios. At least because a L3 cache region does not need to snoop other L2 caches or L3 cache regions every single time, a POSITA would have understood that the cache controllers for them work together with each other and form a single logical caching agent.

Sinharoy at 1:17 (Table 2)

Mercedes EX1021 EX1017 (Horst Supp. Decl.) at ¶17

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Snooping Is Not Necessary for All of the Other L3 Caches



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.

Sinharoy at 1:19

the term “possibly other L3 caches” in this sentence. The term “possibly” means that not all of the other L3 cache regions need to be snooped. As I have opined in my deposition, this disclosure means the cache controllers for the L2 and L3 caches do not work independently from each other. (See EX2001, 18:3-16.) The caches in Sinharoy cannot independently handle read and write misses without coordinating with each other. (See *id.*) To avoid the need of snooping all of the other L3 regions, there has to be “some high-level logic that does that coordination.” (*Id.*, 17:11-14.) A POSITA would have understood that the cache controller for a L3 region has knowledge about whether the lines in its local L3 region are held in any other L2 and L3 caches as well. (See *id.*, 18:3-9.) Such knowledge means that each of them has to have knowledge of what is in the other caches to be able to avoid snooping in other regions. (See *id.*, 18:3-16.)

EX1017 (Horst Supp. Decl.) at ¶15

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Dispute 2 – Whether Tang Discloses the “Single Caching Agent” Limitation

Mercedes EX1021
Mercedes-Benz USA, LLC
v. Daedalus Prime LLC



DMA Cache: Using On-Chip Storage to Architecturally Separate I/O Data from CPU Data for Improving I/O Performance

Dan Tang^{1,2,3}, Yungang Bao¹, Weiwu Hu^{1,2}, Mingyu Chen¹

¹Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences

²Loongson Technologies Corporation Limited

³Graduate School of Chinese Academy of Sciences
{tangdan, baoyg, hww, cmty}@ict.ac.cn

Abstract

As technology advances both in increasing bandwidth and in reducing latency for I/O buses and devices, moving I/O data in/out memory has become critical. In this paper, we have observed the different characteristics of I/O and CPU memory reference behavior, and found the potential benefits of separating I/O data from CPU data. We propose a DMA cache technique to store I/O data in dedicated on-chip storage and present two DMA cache designs. The first design, Decoupled DMA Cache (DDC), adopts additional on-chip storage as the DMA cache to buffer I/O data. The second design, Partition-Based DMA Cache (PBDC), does not require additional on-chip storage, but can dynamically use some ways of the processor's last level cache (LLC) as the DMA cache. We have implemented and evaluated the two DMA cache designs by using an FPGA-based emulation platform and the memory reference traces of real-world applications. Experimental results show that, compared with the existing snooping-cache scheme, DDC can reduce memory access latency (in bus cycles) by 34.8% on average (up to 58.4%), while PBDC can achieve about 80% of DDC's performance improvements despite no additional on-chip storage.

1. Introduction

DMA technology provides special channels for CPU and I/O devices to exchange I/O data, and the memory is used for buffering the I/O data. When the CPU wants to handle I/O data, it triggers the DMA write operations that transfer the I/O data from I/O devices to the memory. On the opposite direction of the CPU writing data to I/O devices, the DMA read operations, i.e., transferring I/O data from the memory to I/O devices, are performed.

In the past decade, the performance of I/O bus and device has improved dramatically. New I/O bus technologies, such as PCI-Express 2.0, AMD's HyperTransport 3.0 and Intel's QPI, can provide bus bandwidths of 10-20GB/s that are close to the bandwidth of a DDR2/3 DRAM memory system. I/O device's bandwidth and latency have also substantially improved.

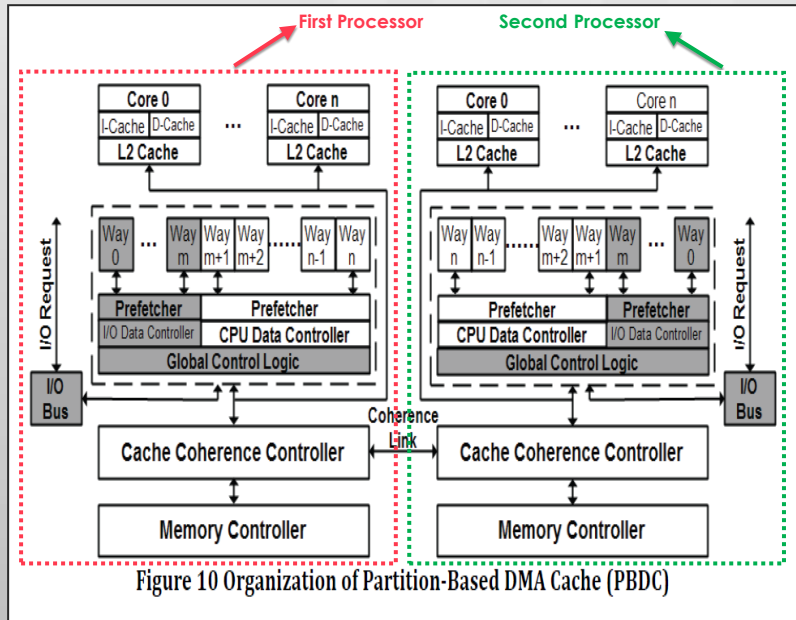
Corresponding author: Weiwu Hu (hww@ict.ac.cn).

For example, using a single server, Fusion-io Company recently adopted PCI Express and SSD technology to exceed one million IOPS (I/O Operations Per Second) and 8GB/s sustained throughput [1]. Unlike traditional hard disks, SSD avoids mechanical delays so that it usually has low access latency, e.g., experiments on Intel® X25-E SSD demonstrate that the latency of reading 4KB data is only 75us [2]. Additionally, the latency of Infiniband's RDMA operations can even be less than 1us [5]. With these substantial technology advances, moving I/O data in/out memory has become critical for DMA scheme, because it exhausts a substantial portion of memory bandwidth as well as accounts for a considerable part of I/O operation latency. Kumar et al. have profiled the execution of handling 4KB network I/O data and found that buffer copy (i.e., I/O data movement) is the largest single contributor to the overall processing time per packet, accounting for over 25% [18].

To address this problem, researchers have proposed several schemes to reduce the memory access overhead of the DMA operations. Some studies proposed the cache injection techniques that directly inject I/O data into a processor's cache [21-23]. Iyer proposed a chipset cache to improve performance of Web Servers [15]. Intel proposed a Direct Cache Access (DCA) scheme [14] that has been implemented in the Intel 82599 10 Gigabit Ethernet (GbE) controller [4]. Although these schemes have demonstrated significant I/O performance improvement for a few specific workloads under some specific environments, their evaluation results are limited because of the three reasons: (1) Little previous work investigated the inherent characteristics of DMA memory references, such as DMA request size and reuse distance (the definition is in Section 2). The characteristic differences between DMA and CPU memory reference patterns, however, are significant for studying effective optimization schemes. (2) Their studies mainly focus on specific processors that are designed for some specific applications, such as networking processors and embedded processors. (3) Their conclusions are usually drawn from studying one type of I/O such as network. In real systems, e.g., Video-on-Demand servers and data centers, however, various types of I/O operations can

- Tang discloses a system and method to store I/O data into a shared last level cache (LLC).
- One design, "Partition-Based DMA Cache (PBDC)," includes a dual-processor system, each using the processor's LLC as a "dedicated DMA cache." (Tang at § 3.2.1)
- Tang discloses I/O data from an I/O device can be written directly into the dedicated DMA cache. (Id. at § 2, § 3.2.2)

Tang Overview



Tang at Fig. 10 (annotated)

- The PBDC design of Tang is a dual-processor system. Each processor includes:

- n Cores (Core 0, ..., Core n)
- An I/O bus handling I/O requests
- A memory controller
- An LLC partitioned into n ways (Way 0, ..., Way n)
- A cache Coherence Controller
- An LLC Controller (comprising two prefetchers, I/O data controller, CPU data controller, global control logic, and configuration module) (Tang at § 3.2)

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Tang Discloses a Single Caching Agent in Each Processor



Single Caching Agent for First Processor

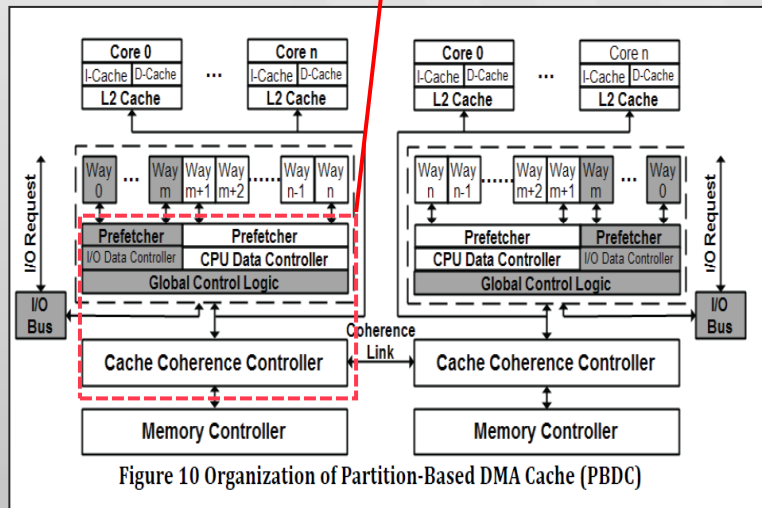


Figure 10 Organization of Partition-Based DMA Cache (PBDC)

Tang at Fig. 10 (annotated)

The first processor on the left includes a cache coherence controller and an LLC-controller, which form a logical “single caching agent”

- The I/O data controller and CPU data controller of the LLC-controller performs cache coherency operations for the cores (Core 0 to Core n) (Tang at § 3.2.2)
- The prefetchers of the LLC-controller perform cache coherency operations for the I/O module (*id.* at §§ 3.1.5, 3.2.1)
- The cache coherence controller performs cache coherency operations for the cores and the I/O module (*id.* at §§ 3.1.5, 3.2.1)

Thus, Tang's “single caching agent” performs “cache coherency operations for the plurality of cores and the I/O module.”

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

LLC and Cache Coherence Controllers Manage Cache Coherence Operations for the Processor



can use several ways of the processor's LLC as the dedicated DMA Cache. Because some important design issues of PBDC, such as write policy, cache coherence protocol, replacement policy, can adopt the same design of DDC, we mainly focus on the design of the LLC's controller.

Tang at § 3.2.1 (adopting some of same designs from DDC design)

I/O Data Controller (IOD-Ctrler) and CPU Data Controller (CPUD-Ctrler) are responsible for write policy, maintaining cache coherence, replacement policy and tagging data for the I/O data ways ($\text{Way}_0 \sim \text{Way}_m$) and the CPU data ways ($\text{Way}_{m+1} \sim \text{Way}_n$) respectively. CPUD-Ctrler still adopts the original cache management policies. IOD-Ctrler can use different policies for the I/O data, e.g., the WT policy, the IO-ESI protocol and the replacement policy of the DDC.

Id. at § 3.2.2

In the CPDC direction, the DMA read request on the DMA Cache can trigger a prefetching operation. As mentioned in Subsection 2.2, the I/O data has very regular reference patterns, most being linear within a DMA buffer. Therefore, the DMA Cache could benefit from adopting even a straightforward sequential prefetching technique. In our design, the prefetcher is triggered to generate four sequential prefetching requests upon the state transitions from Invalid-state to Shared-state or Exclusive-state. The prefetching requests are forwarded to the cache coherence controller as a Probe-Read request and the desired data can be fetched from either some processor's caches or the memory of the system.

The L1/L2 caches add one bit to each block tag to indicate whether the block stores I/O data or CPU data. The tag of the block can be propagated among the L1/L2 caches. The replacement policy of the L1/L2 caches requires a slight modification in which the block with the "IO_Data" tag is evicted directly without being swapped into the victim buffer or the LLC. To propagate the tag, the data paths also need to add one bit. When forwarding blocks to other caches, the DMA Cache and LLC tag the data copy as "IO_Data" and "CPU_Data", respectively. In other words, the DMA Cache is the source of "IO_Data" tag and the LLC is the source of "CPU_Data" tag.

Id. at § 3.1.5 (DDC Design)

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Tang Discloses a Single Caching Agent



Patent Owner argues:

Because Tang includes a pair of cache controllers which communicate with one another via a “coherence link,” it does not have a “single caching agent.”

But:

- As previously explained, Figure 10 of Tang discloses two processors, each with the same structure and each including its own cache controller and LLC-controller.
- The “coherence link” in Figure 10 connects the two processors together to form a cluster.
- Patent Owner raised this in their Response, but appears to have dropped the argument in the Sur-reply.

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Dispute 3 – Whether Tang Discloses “Distributed Portions Each Associated with a Corresponding” Core Limitation

Mercedes EX1021
Mercedes-Benz USA, LLC
v. Daedalus Prime LLC

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE IPR2023-01344

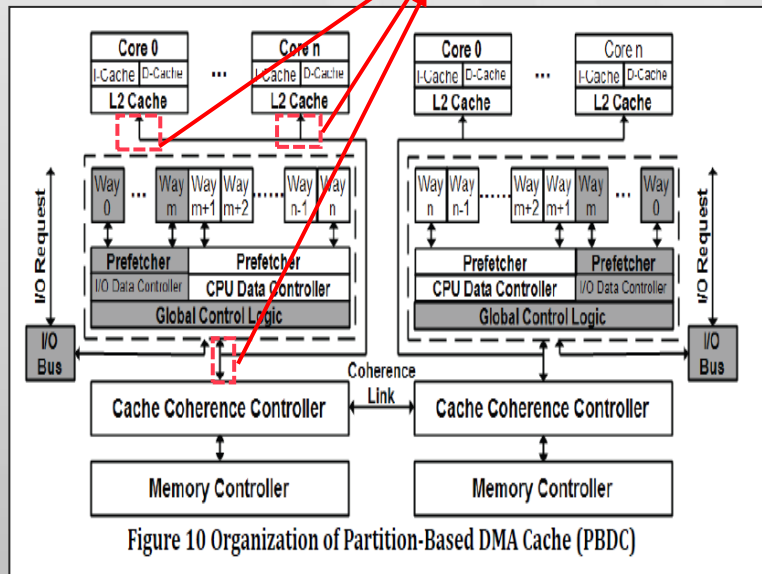


Claim 1 recites “wherein the caching agent . . . includes a plurality of distributed portions each associated with a corresponding one of the plurality of cores”

Tang Discloses a Caching Agent with Distributed Portions



Each arrow is connected to a “distributed portion” of logic associated with a corresponding core



Tang at Fig. 10 (annotated)

29. In addition, according to Figure 10 of Tang, the LLC controller and the cache coherence controller (i.e., the “single caching agent”) include signals and logic (depicted as arrows in Figure 10) that are each connected to core 0 to core n. (EX1004, Figure 10 (annotated below).) The separate arrows to each of the cores 0-n shows that the shared logic must have signals to individual cores for handling, e.g., L2 cache misses, and those signals are sent or received by logic associated with individual cores. As shown in Figure 10, each signal (depicted as a single arrow going into a given core in Figure 10) is associated with logic in a single given core, which is a “distributed portion” of the caching agent. (EX1004, Figure 10 (annotated below).) Thus, because the cache coherence controller sends different signals to the different cores, it has “distributed portions” of logic that connect to these signals. In addition, the cache coherence controller has “distributed portions” corresponding to each core, allowing it to separately address and send signals to the different cores. (EX1004, Figure 10 (annotated below).)

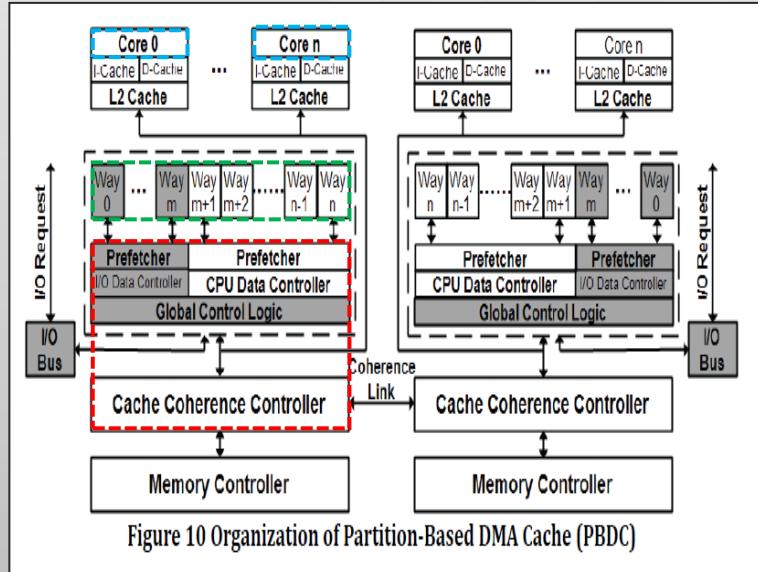
EX1017 (Horst Supp. Decl.) at ¶29

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Tang Discloses a Caching Agent with Distributed Portions



Tang at Fig. 10 (annotated)

- The left processor of the PBDC design includes an LLC partitioned into n ways (Way 0 to Way n) and n cores. (Tang at § 3.2.2)
- Because the caching agent manages cache coherence protocol for n cores and the n -way shared LLC, it has “distributed portions” of logic that each associated with a given core.

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Tang Discloses a Caching Agent with Distributed Portions



Patent Owner argues:

“Even if Tang can be read as disclosing a single caching agent for the processor, it is evidence that caching agent does not include a plurality of distributed portions each associated with a corresponding one of the plurality of cores . . .”

“[W]hen it comes to the cache controller in Tang, all of the cores of the processor share the same cache controller.”

But:

- As shown previously, the cache coherence controller (as part of the “single caching agent”) sends different signals to different cores, and thus has “distributed portions” of its logic that connect to the signals.
- Each arrow is connected to a “distributed portion” of logic associated with a corresponding core from Core 0 to Core n.
- Further, the cache coherence controllers has “distributed portions” corresponding to each core, allowing it to separately address and send signals to the different cores.

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Dispute 4 – Whether Xu Was Publicly Available Before December 13, 2011

Ground 1b: Sinharoy in view of Xu Renders Obvious Dependent Claims 2 and 10

Mercedes EX1021
Mercedes-Benz USA, LLC
v. Daedalus Prime LLC

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE IPR2023-01344



Xu's Publicity Available Before December 13, 2011

Explorations of Optimal Core and Cache Placements for Chip Multiprocessor

Thomas Canhao Xu, Pasi Liljeberg, Hannu Tenhunen
Department of Information Technology, University of Turku, 20014 Turku, Finland
Turku Centre for Computer Science (TUCS), Joukahaisenkatu 3-5 B, 20520 Turku, Finland
xcanx, pasi.liljeberg, hannu.tenhunen@utu.fi

Abstract—In this paper, we study and analyze optimal core and cache placements for modern Chip Multiprocessors (CMPs). As the number of cores increases, traditional on-chip interconnects such as bus and crossbar suffer from poor scalability and low efficiency. Ring based design has been proposed and implemented to mitigate these problems. However, the continuous growth of number of cores will render the ring interconnect infeasible. Network based designs are therefore proposed for future CMPs for better scalability. In this paper, we explore the interconnect of a state-of-the-art CMP. We analyze and compare the implementation of the ring-based and the network-based interconnect. The placement of cores and caches in a network is proved crucial for system performance. We investigate optimal core/cache placement for CMPs. The benchmark results are presented by using a cycle-accurate full system simulator. Results show that, by using the optimal network interconnect, compared with the ring interconnect, the average network latency and execution time are reduced by 11.5% and 12.5% respectively, for four configurations and two applications.

1. INTRODUCTION

We have witnessed multi-core processors emerging with increasing number of cores and complex on-chip interconnect in the past few years. The constraints of chip clock frequency, power consumption and heat dissipation have pushed the chip designers to integrate multiple cores rather than to improve single-core performance. A prevalent approach to increase chip performance is to add more cores. Normally, each core has its own computational units, registers and private caches. A last level cache will be shared by all cores. Intel Pentium D¹ is one of the first CMP which embedded two dies on a processor package. Nowadays, quad-core CMPs such as Intel Core i5, i7 [1] and AMD Phenom II [2] are mainstream for desktop systems. The integration of more cores on a CMP is under intensive research. For example, AMD has announced its twelve-core Opteron CMP with two dies on a chip, each of which has six cores [3]. The newest Xeon, based on Westmere-EX architecture, has been fabricated by Intel.

[4]. Sun and IBM have their own CMPs [5], [6]. It is predictable that in the future, the integration of more cores on a single chip will move on.

The current communication schemes in CMPs are based on the shared bus and crossbar architectures. For large scale CMPs, crossbar based architectures are

since they have a simple topology and are easy to implement. Besides, the latencies are usually fixed and messages can be broadcasted. However, as the system size increases, crossbar based communication architectures are no longer feasible due to the circuit complexity and energy efficiency. For example, crossbar interconnect is used in Intel Nehalem and Westmere architectures. Each core has its own data path (around 1,000 wires) to the central interconnect. This provides high bandwidth for a small number of agents (e.g. cores, caches and memory controllers). The latency grows as the number of agents grows, e.g. the latency to the last level cache with six cores is 22% higher than that in four cores [7]. The crossbar has to be redesigned and it becomes more complex. Another research shows that, high-radii crossbars are not feasible due to the area overhead and the frequency limitations [8]. Obviously, we cannot design a crossbar based CMP by simply adding more core/caches.

Due to the limited scalability of crossbar switches, academics and industries have moved to new approaches. Sun SPARC T3 applied a hierarchical crossbar to alleviate the scaling problem. While IBM Power 7, Intel Sandy Bridge, Nehalem-EX and Westmere-EX rely on a ring interconnect. Previous researches have focused on the performance and efficiency of crossbar/mesh interconnects. For example, Cheshmin Park et al. [9] presented a 1.2 Tbit/s ring interconnect implementation, providing on-chip communication for 8 cores, 8-port parallel access 24 MB cache, and 2 system interfaces. Hyper ring, hierarchical ring and hybrid ring have been proposed to improve efficiency further [10], [11]. Jean Canache Villanueva et al. [12] compared the performance differences between crossbar, ring and mesh interconnects. In this paper, we investigate the placement of cores and caches in a 2D mesh network interconnect. We model a CMP based on state-of-the-art processors. We analyze the impact of different placements of core/caches in a mesh network. Optimal placements of

- Xu has a “2011 IEEE” copyright notice. (EX1005 at 1)
- Gordon MacPherson (Director, Board Governance & Policy Development at IEEE) indicated in his declaration that “Xu . . . , published in 2011 NORCHIP, date of conference November 14-15, 2011. Copies of the conference proceedings were made available no later than the last day of the conference . . . ” (EX1013 at ¶13)
- According to the record of the copyright catalog from the Copyright Office, 2011 NORCHIP was published at least by December 22, 2011, and thus would have been distributed before then. (EX1019)

978-1-4577-0516-8/11/\$26.00 ©2011 IEEE

978-1-4577-0516-8/11/\$26.00 ©2011 IEEE

Mercedes EX1005, Page 1 of 6
U.S. Patent No. 9,575,895

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Xu Was Publicly Available Before December 13, 2011



11. Thomas Canhao Xu, et al.; “Explorations of optimal core and cache placements for Chip Multiprocessor”, published in 2011 NORCHIP, date of conference November 14-15, 2011. Copies of the conference proceedings were made available no later than the last day of the conference. The article is currently available for public download from the IEEE digital library, IEEE Xplore.

EX1013 (MacPherson Decl.) at ¶11

The screenshot shows the Copyright Public Records System (CPRS) search results for the entry '2011 NORCHIP (11EX5488)'. The entry is highlighted with a red dashed box. The search results display the following information:

- Public Catalog**
- Copyright Catalog (1978 to present) at DC4
- Search Request: Builder = (2011 AND NORCHIP)[in Title (TKE
- Search Results: Displaying 1 of 1 entries
- 2011 NORCHIP (11EX5488)** (highlighted with a red dashed box)
- Type of Work:** Text
- Registration Number / Date:** TX0007522416 / 2012-04-23
- Application Title:** 2011 NORCHIP (11EX5488)
- Title:** 2011 NORCHIP (11EX5488)
- Description:** Book, 221 p.
- Copyright Claimant:** IEEE, Transfer: By written agreement, United States.
- Date of Creation:** 2011
- Date of Publication:** 2011-12-22 (highlighted with a red dashed box)
- Nation of First Publication:** United States
- Authorship on Application:** IEEE, employer for hire; Domicile: United States. Authorship: Collective Work.
- Rights and Permissions:** William Hagen, IEEE, 445 Hoes Lane, Piscataway, NJ, 08854, United States, (732) 562-3966; w.hagen@ieee.org
- Names:** IEEE
- Save, Print and Email (Help Page)**
- Select Download Format:** Full Record | Format for Print/Save
- Enter your email address:** | Email
- Help Search History Titles Start Over**
- Contact Us | Request Copies | Get a Search Estimate | Frequently Asked Questions (FAQs) about Copyright | Copyright Office Home Page | Library of Congress Home Page**

EX1019 (copyright record of 2011 NORCHIP)

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Xu Was Publicly Available Before December 13, 2011



Patent Owner argues:

"[Mr. MacPherson's] declaration offers no indication that he had personal knowledge of any publication of Xu at the 2011 NORCHIP conference . . ."

But:

- Patent Owner has not put forth any contradictory evidence regarding Xu's date of public availability.
- The declaration by Mr. MacPherson, who is Director, Board Governance & Policy Development at IEEE, explicitly states that it was available at least by Nov. 15, 2011.
- Record from the Copyright Office shows that 2011 NORCHIP was published at least by Dec. 22, 2011, supporting the position it would have been distributed before then.

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

EX1019 is admissible under FRE 803(8) and 902



Patent Owner argues:

“ . . . Ex. 1019 is being offered to prove the truth of the alleged public availability of Xu . . . , is hearsay.”

“ . . . Ex. 1019, that record has not been authenticated.”

But:

- In *Smart Microwave Sensors GMBH v. Wavetronix LLC*, the Board found that printed copies of “Copyright Office Catalog information” are “public records and, therefore, not hearsay.” IPR2016-00488, Paper 57, at 29 (PTAB July 17, 2017).
- Patent Owner appears to have dropped the hearsay objection in the Sur-reply.
- In *Jackson v. Netflix, Inc.*, 506 F. Supp. 3d 1007, 1013 (C.D. Cal. 2020), the court took judicial notice of copyright catalog entries from the Copyright Office’s website under FRE 902, because they were “indisputably authentic, relevant to [plaintiff’s] claims, and on government websites.”

Mercedes EX1021

Mercedes-Benz USA, LLC
v. Daedalus Prime LLC
IPR2023-01344

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE