IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Micron Technology, Inc.; Micron Semiconductor Products, Inc.; Micron Technology Texas LLC; Dell Technologies Inc.; Dell Inc.; and HP Inc.

Petitioners,

v.

Unification Technologies LLC,

Patent Owner.

Case No. IPR2021-00343

U.S. Patent No. 8,533,406

PETITION FOR *INTER PARTES* REVIEW OF CLAIMS 15-21 AND 26-30 OF U.S. PATENT NO. 8,533,406

TABLE OF CONTENTS

Page

I. Introduction 1

II. Petitioners Meet Standing and Eligibility Requirements for *Inter Partes* Review 2

III. Prosecution History of the ’406 Patent 3

IV. Background 3

V. Summary of the ’406 Patent 4

- A. Effective Filing Date and Date of Invention 4
- B. Level of Ordinary Skill in the Art 5

VI. Claim Construction 5

VII. Precise Relief Requested 7

- A. Proposed Grounds 7
- B. Qualifying Prior Art 8
- C. The Proposed Grounds Are Not Cumulative or Redundant 9

VIII. The Prior Art 9

- A. Summary of Bennett 9
- B. Summary of Suda 13
- C. Summary of SwSTE’05 15
- D. Motivation to Combine Suda and SwSTE’05 16

IX. Ground 1: Obvious Over Bennett and POSITA Knowledge 17

- A. Claim 15 17
- B. Claim 16 23
- C. Claim 17 24
- D. Claim 18 25
- E. Claim 19 25
- F. Claim 20 27
- G. Claim 21 27
- H. Claim 26 28
- I. Claim 27 29

J. Claim 28 33
K. Claim 29 35
L. Claim 30 36
X. Ground 2: Obvious Over Suda and POSITA Knowledge 36
A. Claim 15 36
B. Claim 16 41
C. Claim 17 42
D. Claim 18 43
E. Claim 19 43
F. Claim 20 44
G. Claim 21 45
H. Claim 26 45
I. Claim 27 48
J. Claim 28 54
K. Claim 29 55
L. Claim 30 56
XI. Ground 3: Obvious Over Suda, SwSTE'05, and POSITA Knowledge 57
A. Claim 21 57
B. Claim 26 59
C. Claim 28 61
XII. Secondary considerations 61
XIII. The Parallel District Court Litigations Do Not Warrant Denying Institution 63
XIV. Mandatory Notices..65
A. Real Parties-in-Interest 65
B. Related Proceedings 66
C. Lead and Backup Counsel..66
D. Electronic Service...67
XV. Fees 67
XVI. Conclusion 67

TABLE OF AUTHORITIES

Page(s)

Cases

Apple Inc. v. Fintiv, Inc.,
IPR2020-00019, Paper 11 (P.T.A.B. Mar. 20, 2020) 63, 64, 65

Apple, Inc. v. SEVEN Networks, LLC,
IPR2020-00156, Paper 10 (P.T.A.B. June 15, 2020) 63, 64

Geo. M. Martin Co. v. All. Mach. Sys. Int'l LLC,
618 F.3d 1294 (Fed. Cir. 2010) 62

Globalfoundries Inc. v. UNM Rainforest Innovations,
IPR2020-00984, Paper 11 (P.T.A.B. Dec. 9, 2020) 65

Intel Corp. v. Alacritech, Inc.,
IPR2017-01391, Paper 8 (P.T.A.B. Nov. 28, 2017) 7

Koninklijke Philips N.V. v. Google LLC,
948 F.3d 1330 (Fed. Cir. 2020) 8

Micron Tech., Inc. v. Godo Kaisha IP Bridge 1,
IPR2020-01007, Paper 15 (P.T.A.B. December 7, 2020) 64

Precision Planting LLC v. Deere & Co.,
IPR2019-01048, Paper 17 (P.T.A.B. Dec. 4, 2019) 65

Samsung Elecs. Am., Inc. v. Prisua Eng'g Corp.,
948 F.3d 1342 (Fed. Cir. 2020) 7

Spherix Inc. v. Matal,
703 F. App'x 982 (Fed. Cir. 2017) 7

Target Corp. v. Proxicom Wireless, LLC,
IPR2020-00904, Paper 11 (P.T.A.B. Nov. 10, 2020) 7

Vibrant Media v. Gen. Elec. Co.,
No. IPR2013-00172, Paper 50, 10 (P.T.A.B. July 28, 2014) 7

VMware, Inc. v. Intellectual Ventures I LLC,
IPR2020-00470, Paper 13 (P.T.A.B. August 18, 2020) 63

ZTE (USA) Inc. v. Evolved Wireless LLC,
No. IPR2016-00757, Paper 42 (P.T.A.B. Nov. 30, 2017) 62

Statutes

35 U.S.C. § 102 4

35 U.S.C. § 103 4

Other Authorities

37 C.F.R. § 42.104(a) 2

PETITIONERS' EXHIBIT LIST

Ex. No.	Brief Description
1001	U.S. Pat. No. 8,533,406 B2, titled "APPARATUS, SYSTEM, AND METHOD FOR IDENTIFYING DATA THAT IS NO LONGER IN USE" to Flynn et al.
1002	U.S. Pat. No. 7,624,239 B2, titled "METHODS FOR THE MANAGEMENT OF ERASE OPERATIONS IN NON-VOLATILE MEMORIES" to Bennett et al.
1003	U.S. Pat. No. 7,057,942 B2, titled "MEMORY MANAGEMENT DEVICE AND MEMORY DEVICE" to Suda et al.
1004	Expert Declaration of Jacob Baker, Ph.D., P.E., Regarding U.S. Patent No. 8,533,406.
1005	American National Standard for Information Technology—AT Attachment with Packet Interface – 6 (ATA/ATAPI-6), ANSI INCITS 361-2002 (Sept. 2002) (excerpts filed with permission).
1006	American National Standard for Information Technology—AT Attachment with Packet Interface – 7 Volume 1 – Register Delivered Command Set, Logical Register Set (ATA/ATAPI-7 V1), ANSI INCITS 397-2005 (Feb. 7, 2005) (excerpts filed with permission).
1007	Serial ATA (SATA) Revision 2.5, Serial ATA International Organization (Oct. 27, 2005).
1008	WILLIAM D. BROWN & JOE E. BREWER, NONVOLATILE SEMICONDUCTOR MEMORY TECHNOLOGY (IEEE 1998).
1009	BRIAN DIPERT & MARKUS LEVY, DESIGNING WITH FLASH MEMORY (Annabooks 1994).
1010	Eran Gal et al., *Mapping Structures for Flash Memories: Techniques and Open Problems*, PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON SOFTWARE—SCIENCE, TECHNOLOGY & ENGINEERING ("SwSTE'05") (digital version), Herzlia, Israel, 2005, pp. 83-92, doi:

Ex. No.	**Brief Description**
	10.1109/SWSTE.2005.14.
1011	H. Niijima, *Design of a Solid-State File Using Flash EEPROM*, IBM JOURNAL OF RESEARCH AND DEVELOPMENT, vol. 39, no. 5, pp. 531-545, Sep. 1995.
1012	Original Complaint for Patent Infringement, *Unification Techs. LLC v. Micron Tech. Inc.*, No. 6:20-cv-500 (W.D. Tex. 2020), ECF No. 1.
1013	Exhibit B to Plaintiff's First Amended Infringement Contentions: Unification Technologies' Allegations of Infringement with Respect to U.S. Patent No. 8,533,406, *Unification Techs. LLC v. Micron Tech. Inc.*, No. 6:20-cv-500 (W.D. Tex. 2020).
1014	Eran Gal et al., *Mapping Structures for Flash Memories: Techniques and Open Problems*, PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON SOFTWARE—SCIENCE, TECHNOLOGY & ENGINEERING ("SwSTE'05") (print copy as scanned by Sylvia-Ellis Hall), Herzlia, Israel, 2005, pp. 83-92, doi: 10.1109/SWSTE.2005.14.
1015	U.S. Provisional Pat. No. 60/873,111 titled "Elemental Blade System," to Flynn et al.
1016	Docket Report for *Unification Techs. LLC v. Micron Tech. Inc.*, No. 6:20-cv-500 (W.D. Tex. 2020) (accessed Dec. 22, 2020).
1017	Frank Shu, *Notification of Deleted Data Proposal for ATA8-ACS2*, T13 (rev. 0, Apr. 21, 2007).
1018	Frank Shu & Nathan Obr, *Notification of Deleted Data Proposal for ATA8-ACS2 Revision 1*, T13 (July 26, 2007).
1019	Frank Shu & Nathan Obr, *Notification of Deleted Data Proposal for ATA8-ACS2 Revision 2*, T13 (September 5, 2007).
1020	U.S. Pat. No. 6,766,432 B2, titled "MEMORY MANAGEMENT SYSTEM SUPPORTING OBJECT

Ex. No.	Brief Description
	DELETION IN NON-VOLATILE MEMORY" to Saltz et al.
1021	Public file history of U.S. Pat. No. 8,533,406, titled "APPARATUS, SYSTEM, AND METHOD FOR IDENTIFYING DATA THAT IS NO LONGER IN USE" to Flynn et al.
1022	MARC record for the print digital version of the IEEE International Conference on Software--Science, Technology & Engineering Proceedings in the Linda Hall Library.
1023	MARC record for the print version of the IEEE International Conference on Software--Science, Technology & Engineering Proceedings obtained from the OCLC bibliographic database.
1024	MARC record for Library of Congress.
1025	Curriculum vitae of Sylvia Hall-Ellis, Ph.D.
1026	Curriculum vitae of Jacob Baker, Ph.D., P.E.
1027	*Proceedings. IEEE International Conference on Software – Science, Technology and Engineering*, IEEE COMPUTER SOCIETY, www.computer.org/csdl/proceedings/swste/2005/12OmNC17hWm (last visited Oct. 30, 2020).
1028	Filed Stipulations of Petitioners for U.S. Patent No. 8,533,406.
1029	Amended Scheduling Order, *Unification Techs. LLC v. Micron Tech. Inc.*, No. 6:20-cv-500 (W.D. Tex. 2020), ECF No. 48.
1030	Expert Report of Sylvia Hall-Ellis, Ph.D. in Support of Public Availability of the Gal Publication.
1031	Judge Albright, ORDER GOVERNING PROCEEDINGS – PATENT CASE (Ver. 3.2).

Ex. No.	**Brief Description**
1032	Micron's Preliminary Identification of Extrinsic Evidence, *Unification Techs. LLC v. Micron Tech. Inc.*, No. 6:20-cv-500 (W.D. Tex. 2020).
1033	Plaintiff's Revised Claim Constructions, *Unification Techs. LLC v. Micron Tech. Inc.*, No. 6:20-cv-500 (W.D. Tex. 2020).
1034	*Mapping Structures for Flash Memories: techniques and open problems*, IEEE *XPLORE*, https://ieeexplore.ieee.org/document/1421068 (last visited Dec. 18, 2020).
1035	U.S. Pat. No. 5,404,485A, titled "FLASH FILE SYSTEM" to Ban.
1036	INSTITUTE FOR THE ADVANCEMENT OF THE AMERICAN LEGAL SYSTEM, CIVIL CASE PROCESSING IN THE FEDERAL DISTRICT COURTS (2009).

I. Introduction

U.S. Patent No. 8,533,406 ("the '406 patent") should never have issued. For example, Claim 15 generally recites (i) receiving an indication that a data structure, corresponding to data stored on a non-volatile storage medium, has been deleted; (ii) where the indication includes a logical identifier, associated with the data structure by a storage client, that is mapped to a physical storage address; and (iii) recording that the data can be erased in response to receiving the indication. In the related litigations, the Patent Owner, Unification Technologies LLC ("UTL") generally asserts the claims encompass receiving a command indicating that "data has been erased" from a user's perspective, e.g., by a computer attached to storage, even though the data remains on the storage device. A person of ordinary skill in the art ("POSITA") would have known these concepts long before the alleged effective filing date in 2006.

For example, erase commands specifying logical addresses were part of the Advance Technology Attachment ("ATA") industry standard by 2002. Ex. 1005, § 8.1. Additionally, in 1995, Ban patented updating logical-to-physical address mappings when data is deleted. *See* Ex. 1035, 5:61-65; Ex. 1010, § 2.2 (Ban patented the Flash Translation Layer ("FTL"), to perform "block-to-sector mapping" within flash memory, which was adopted as an industry standard); Ex. 1013, 3 (UTL accusing FTL of infringing mapping and storing elements).

With this background knowledge, a POSITA would have found the claims obvious. The primary references Bennett (Ex. 1002) and Suda (Ex. 1003) provide concrete examples of the claimed technology. For example, Bennett discloses responding to erase commands, that specify logical addresses, by storing a flag in a logical-to-physical index mapping to indicate that the data (i) is erased or (ii) is "logically erased" so that an actual erase can take place at a later time. Ex. 1002, 5:60-61, 20:20-27, 20:45-47. Indeed, Bennett teaches that logically erasing data is "common" and can be performed using a system's "standard logical erase method." *Id.*, 5:57-61, 6:18-20. Suda similarly teaches responding to erase commands that specify logical addresses by marking those addresses as in a "virtual erased" state, which is similar to Bennett's logically erased state. Ex. 1003, 5:19-23, 5:38-46, 7:11-19. Suda also teaches maintaining and updating a logical-to-physical address mapping table. *Id.*, 3:13-15, Figs. 1, 7.

The Board should invalidate the challenged claims.

II. Petitioners Meet Standing and Eligibility Requirements for *Inter Partes* Review.

Petitioners certify under 37 C.F.R. § 42.104(a) that the '406 patent "is available for *inter partes* review and that the Petitioners are not barred or estopped from requesting an *inter partes* review challenging the patent claims on the grounds identified in the petition." UTL, sued Petitioners less than one year ago on June 5, 2020. Exs. 1012, 1016.

III. Prosecution History of the '406 Patent

The '406 patent application was filed on September 7, 2012. Ex. 1001, cover. The Examiner rejected the claims on various grounds but did not cite the references relied upon herein. *Id.*, pp. 1-4 (listing cited references); Ex. 1021, 962-72. To overcome the rejections, independent claim 45 (issued as claim 15) was amended to recite an indication that "a data structure corresponding to data stored on the non-volatile storage medium has been deleted," and that "the logical identifier is mapped to a physical address." *Id.*, 999. The Examiner allowed the application, noting, "the art of record fails to teach or suggest receiving a message at a storage controller (or storage layer) comprising a logical identifier, where the message indicates that a client has deleted a blocks [sic] associated with the logical identifier." *Id.,* 1267.

IV. Background

Flash memory is a form of solid-state non-volatile computer memory. Flash memory is organized in erasable units called "blocks," which are made up of smaller "pages." Ex. 1004 ("Baker"), ¶ 63. Unlike traditional platter hard drives, flash memory cannot be directly overwritten—a block must be erased before written to again. *Id.*, ¶ 72. Erase commands for flash memory were well known and standardized before the earliest provisional for the '406 patent. Ex. 1005, §§ 6.16, 8.1.

Flash memory uses an FTL to map logical addresses to physical addresses.

Baker, ¶ 81. A "logical address" is generated by a user's operating system; a "physical address" is the actual storage location on flash memory. *Id.* The FTL allows computer systems to operate and address data in a logical address space (e.g., logical address 0x0000 through 0xFFFF) without concern for where a solid-state storage device physically saves the data (e.g., in which particular block/page). *Id.*, ¶ 82.

V. Summary of the '406 Patent

The '406 patent acknowledges that erase commands for file systems were known. *See, e.g.,* Ex. 1001, 1:32-35 ("In many file systems, an erase command deletes a directory entry in the file system while leaving the data in place in the storage device containing the data.").

Similarly, erasing data by overwriting with zeros, ones, or other null characters was also known. *Id.*, 1:55-60. The patent alleges, however, that these erase methods were "inefficient" because "valuable bandwidth is used while transmitting the data [that] is being overwritten" and "space in the storage device is taken up by the data used to overwrite invalid data." *Id.*, 1:39-42.

A. Effective Filing Date and Date of Invention

The '406 patent claims priority to provisional application no. 60/873,111, filed December 6, 2006. Ex. 1001, cover. Solely for purposes of this IPR, Petitioners assume, but do not concede, an effective filing date of December 6, 2006, for the

’406 patent. Pre-AIA 35 U.S.C. §§ 102 and 103 apply.

B. Level of Ordinary Skill in the Art

A POSITA as of December 2006 would have a Bachelor of Science degree in computer science or electrical engineering and at least two years of experience in the design, development, implementation, or management of solid-state memory devices. Baker, ¶ 55. The references cited in this Petition, the state of the art, and the experience of Dr. Jacob Baker as described in his expert declaration (Ex. 1004) reflect this level of skill in the art. In this Petition, reference to a POSITA refers to a person with these or similar qualifications.

A POSITA would have known, as background information: how flash memory erases data, how flash memory programs or writes data, how memory is used in a cache hierarchy, relative speeds of flash memory compared to other memory, how garbage collection is used with flash memory, how to use wear leveling to combat endurance limits of flash memory, how the FTL works, and industry standards affecting flash memory including the ATA standard. Baker, ¶¶ 55-56.

VI. Claim Construction

The Board construes claims under the same construction standard as civil actions in federal district court. The District Court for the related litigations has not yet construed the claim terms. Ex. 1016.

The parties' proposed constructions from the related litigations are set forth below. Exs. 1032, 1033.

Claim Term	Claim Nos.	Petitioners	UTL
"logical identifier"	15-19, 26-27, 30	"an identifier maintained by a computer attached to the storage medium used to identify the logical location of data stored on the storage medium"	"information that identifies a particular set of data that is not the physical address of the data"
"storage client"	15, 27, 30	Plain and ordinary meaning	"a computing system capable of being coupled to the non-volatile storage medium"
"a logical identifier that is empty"	27, 30	Indefinite	"data identified by the [logical identifier] that does not need to be preserved"
"marking module" "index module" "storage module"	15-19 27, 30 26	Indefinite 112(f) term with no corresponding structure or algorithm	Only [module] needs to be construed. No further construction is needed in light of the surrounding claim language. "Module" should be construed as "a hardware circuit and/or programmable hardware and/or software implemented within

			a storage controller.”

These construction disputes do not affect the outcome of this Petition with respect to any claim. For the terms that Petitioners allege are indefinite, for the purposes of this Petition, Petitioners use UTL’s proposed constructions and have addressed them in the claim analysis below. The Board and Federal Circuit have approved of this procedure in several matters. *See, e.g., Spherix Inc. v. Matal*, 703 F. App’x 982, 983 (Fed. Cir. 2017) (approving petitioner’s proposal of patent owner’s claim interpretations); *Target Corp. v. Proxicom Wireless, LLC*, IPR2020-00904, Paper 11 at 12 (P.T.A.B. Nov. 10, 2020) (“Petitioner’s alternative pleading before a district court is common practice, especially where it concerns issues outside the scope of *inter partes* review.”); *Samsung Elecs. Am., Inc. v. Prisua Eng’g Corp.*, 948 F.3d 1342, 1355 (Fed. Cir. 2020) (indefinite claims may also be found invalid as anticipated or obvious); *Intel Corp. v. Alacritech, Inc.*, IPR2017-01391, Paper 8 at 7 (P.T.A.B. Nov. 28, 2017) (instituting trial even where petitioner argued claim was indefinite); *Vibrant Media v. Gen. Elec. Co.*, No. IPR2013-00172, Paper 50, 10 (P.T.A.B. July 28, 2014) (“an indefiniteness determination in this proceeding would not have prevented us from deciding whether the claims would have been obvious over the cited prior art.”).

VII. Precise Relief Requested

A. Proposed Grounds

a) Ground 1

Claims 15-21 and 26-30 are rendered obvious by Bennett (Ex. 1002) in view of a POSITA's knowledge. The Federal Circuit has affirmed prior obviousness determinations where the claims were found obvious over prior art "in light of the general knowledge" of a POSITA. *Koninklijke Philips N.V. v. Google LLC*, 948 F.3d 1330, 1337-38 (Fed. Cir. 2020). In *Philips*, the Federal Circuit agreed that expert testimony and other references corroborated that "pipelining" in the challenged claims was part of the "general knowledge" of a POSITA. *Id.*, 1338. Although the asserted prior art reference did not expressly teach the "pipelining" claim limitations, a POSITA "would have known about pipelining" and would have "been motivated to combine" this knowledge with the reference. *Id.*, 1338. As in *Philips*, the challenged claims here are obvious over Bennett in light of the general knowledge of a POSITA.

b) Ground 2

Claims 15-21 and 26-30 are rendered obvious by Suda (Ex. 1003) in view of a POSITA's knowledge.

c) Ground 3

Claims 21, 26, and 28 are rendered obvious by Suda (Ex. 1003) in view of (i) a POSITA's knowledge and (ii) SwSTE'05 (Ex. 1010).

B. Qualifying Prior Art

Bennett, Suda, and SwSTE'05 are prior art to the '406 patent. Petitioners are unaware of any assertion that the ’406 patent is entitled to an invention date earlier than the assumed effective filing date. Bennett (filed November 14, 2005) is § 102(e) prior art and Suda (filed December 28, 2004; published March 16, 2006) is § 102(a) and (e) prior art. Ex. 1002, cover; Ex. 1003, cover. SwSTE’05 (presented February 23, 2005 and published by IEEE on May 23, 2005) is § 102(a) and (b) prior art. Ex. 1027, 1-2; Ex. 1034; Ex. 1030, ¶¶ 46-49, 54; Baker, ¶¶ 58-59.

C. The Proposed Grounds Are Not Cumulative or Redundant

The grounds for trial presented in this Petition are not cumulative to issues already examined during prosecution. The references raised in this proceeding were not cited during prosecution. Furthermore, the references relied on by the Examiner during prosecution allegedly did not disclose: “receiving a message at a storage controller (or storage layer) comprising a logical identifier, where the message indicates that a client has deleted a blocks associated with the logical identifier.” *See* § III*, supra*. As shown herein: Bennett and Suda do.

VIII. The Prior Art

A. Summary of Bennett

Like the ’406 patent, Bennett recognizes that flash memory erase operations take a (relatively) long time. *Compare* Ex. 1001, 40:50-52 (erasing flash memory “is a lengthy process”) *with* Ex. 1002, 3:5-8 (“In flash memory systems, erase operation may take as much as an order of magnitude longer”).

Bennett addresses lengthy erase times by treating an erase command differently for "specified sectors not forming [a] complete block." *Id.*, 6:13-20. If the erase command specifies a complete block, the block is erased. *Id.* If the command specifies less than a complete block, the sector would be "logically erased" by "the system's standard, logical erase method." *Id.*

Bennett recognizes that "logical erasing" was not inventive and "it was common" for advanced memory systems to erase data logically, with the actual erasure taking place at a later time. *Id.*, 3:26-32. For a logical erasure, the memory system will write a specific "data pattern to the memory portion, set a flag, or otherwise designate it as erased." *Id.*, 3:36-41. The logically erased "portion can then be physically erased when convenient, for example in a background process" such as a garbage collection process. *Id.*, 3:39-41; *compare* Ex. 1001, 51:49-51 ("The data may be later recovered in a storage recovery operation, garbage collection operation, etc.").

Bennett "keeps track of the mapping between logical groups of sectors and their corresponding metablocks" with a Group Address Table ("GAT"). Ex. 1002, 10:19-21, 10:65-11:8 (GAT provides a "list of metablock addresses for all logical groups of host data in the memory system"). Bennett explains that typically, "the host system addresses data in units of logical sectors where, for example, each sector may contain 512 bytes of data." *Id.*, 7:22-24. Bennett further explains that the memory storage "is organized into meta blocks, where each metablock is a group of physical sectors $S_0, \ldots S_{N-1}$ that are erasable together." *See id.*, 7:14-20, Figs. 3A(i)-3A(ii); Baker, ¶ 99. The GAT is stored in non-volatile flash memory as highlighted in Bennett's Fig 6 below:

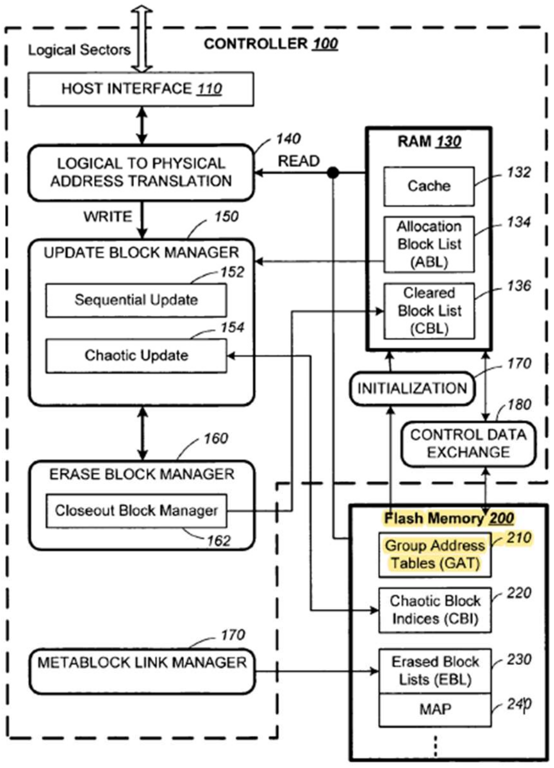


FIG. 6

Ex. 1002, Fig. 6; *see also id.*, 10:16-26. By storing address tables in non-volatile memory, Bennett's system can reconstruct volatile records, such as "when the system is initialized after power-up." *Id.*, 10:47-49.

The GAT is recorded as an index of sectors:

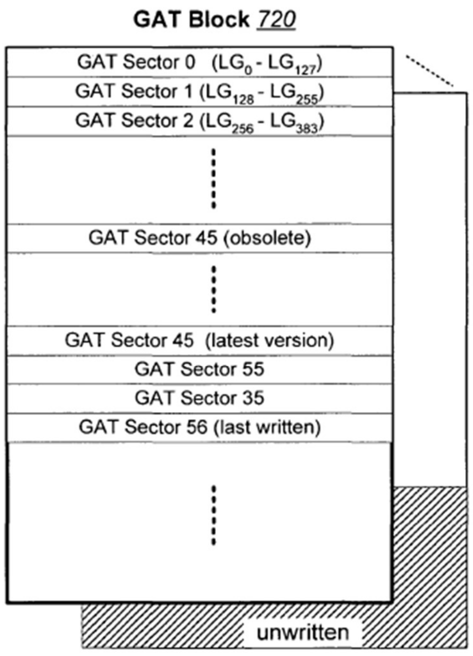

GAT Block 720
GAT Sector 0 (LG0 - LG127)
GAT Sector 1 (LG128 - LG255)
GAT Sector 2 (LG256 - LG383)
GAT Sector 45 (obsolete)
GAT Sector 45 (latest version)
GAT Sector 55
GAT Sector 35
GAT Sector 56 (last written)
unwritten

FIG. 8B

Ex. 1002, 11:4-5, Fig. 8B. Each GAT sector includes two components: "a set of GAT entries for the metablock address of each logical group within a range, and a GAT sector index." *Id.*, 11:13-14. The GAT sector index "contains information for locating all valid GAT sectors within the GAT block." *Id.*, 11:17-18.

Bennett uses flags for marking sector headers as "erased" or "logically erased." *Id.*, 20:20-61 ("Marking Sectors as Erased"). For an actual "erase," Bennett's system marks "sector headers with the 'erased' flag in addition to writing FFs or 00s" to the non-volatile memory. *Id.*, 20:25-27. Writing FFs or 00s to physical memory causes the erasure of flash memory. Baker, ¶ 72. Alternatively, a flag can mark locations as "logically" erased. Ex. 1002, 20:45-47. Unlike the "erased" blocks, logically erased blocks "will not be changed" in the underlying physical memory, but any read attempts will result in the return of "FFs or 00s as if the sectors were erased." *Id.*, 20:47-50, 4:50-54 ("an erased data pattern can be sent to the host if it reads a sector from the erased logical grouping").

B. Summary of Suda

Suda Fig. 1 shows a memory device 1 including a controller 11 and flash memory 14. The controller manages "data erasure," a logical and physical address table 13a, and an erasure area pointer storage area 13b. Ex. 1003, 3:13-15, 5:19-23, Fig. 1. The logical and physical address table 13a maps logical addresses to physical addresses of physical storage locations within the flash memory. *Id.*, 3:43-55.

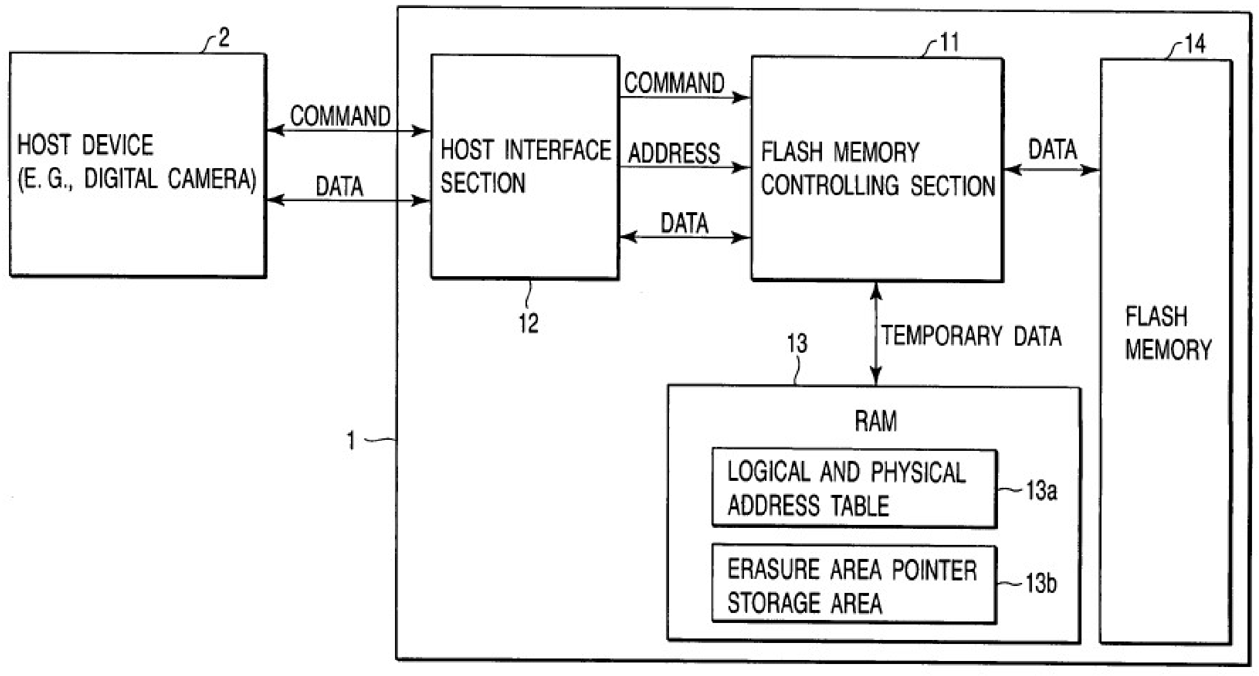

2
HOST DEVICE (E. G., DIGITAL CAMERA)
COMMAND
DATA
HOST INTERFACE SECTION
12
COMMAND
ADDRESS
DATA
11
FLASH MEMORY CONTROLLING SECTION
DATA
14
FLASH MEMORY
TEMPORARY DATA
13
RAM
LOGICAL AND PHYSICAL ADDRESS TABLE
13a
ERASURE AREA POINTER STORAGE AREA
13b
1

FIG. 1

Id., Fig. 1.

Like the ’406 patent, Suda recognizes that “the time required for data erasure is long.” Ex. 1003, 1:19-23, 4:60-67. Suda avoids the lengthy physical erasure process by writing “erasure area pointers” that indicate data ranges to treat as in a “virtual erased” state. *Id.*, 5:9-46. Suda describes this virtual erasure process where, upon receiving an erase command that designates a logical address, start and end erasure area pointers will collectively designate a range of addresses “to be erased.” *Id.*, 5:19-27, 5:36-53, 8:66-9:3, Fig. 8; *see* Figs. 3-5 (reproduced below, showing examples).

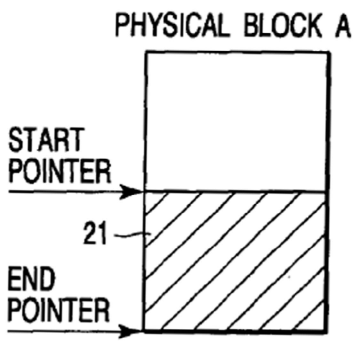


FIG. 3

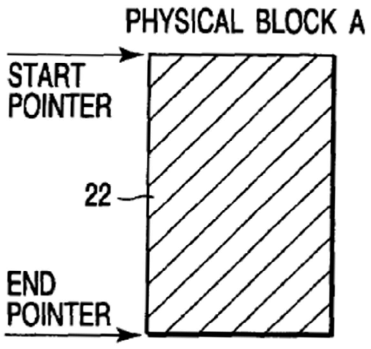


FIG. 4

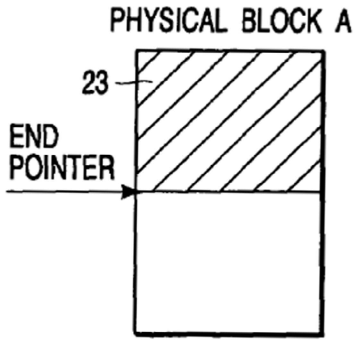
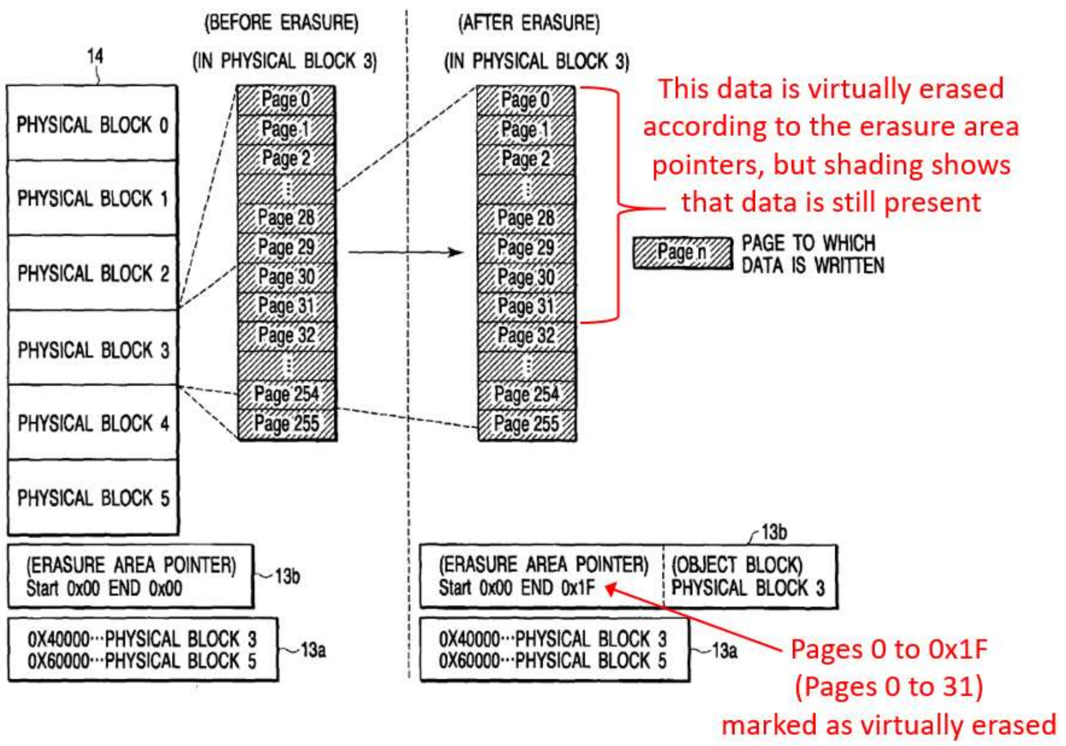


FIG. 5

Virtually erased data may remain stored in memory. Fig. 7 (annotated below) shows that data remains in pages 0-31 despite being marked as virtually erased. Reading data in a virtually erased address range will return “initial-value” (empty) data rather than stored data. Ex. 1003, 9:53-62. The system will physically erase a block once it fills up with virtually erased data, returning the block to an unused state. *Id.*, 5:54-6:3, 5:33-41. When erasing the block, the corresponding logical and physical address entry is removed. *Id.*, 5:54-67, 7:64-8:2.

FIG. 7



(BEFORE ERASURE)
(IN PHYSICAL BLOCK 3)
(AFTER ERASURE)
(IN PHYSICAL BLOCK 3)
14
PHYSICAL BLOCK 0
PHYSICAL BLOCK 1
PHYSICAL BLOCK 2
PHYSICAL BLOCK 3
PHYSICAL BLOCK 4
PHYSICAL BLOCK 5
Page 0
Page 1
Page 2
Page 28
Page 29
Page 30
Page 31
Page 32
Page 254
Page 255
This data is virtually erased according to the erasure area pointers, but shading shows that data is still present
Page n PAGE TO WHICH DATA IS WRITTEN
(ERASURE AREA POINTER) Start 0x00 END 0x00
13b
0X40000···PHYSICAL BLOCK 3
0X60000···PHYSICAL BLOCK 5
13a
(ERASURE AREA POINTER) Start 0x00 END 0x1F
(OBJECT BLOCK) PHYSICAL BLOCK 3
Pages 0 to 0x1F (Pages 0 to 31) marked as virtually erased

Id., Fig. 7 (annotated).

The erasure area pointers are stored both in volatile RAM (*e.g., id.*, Fig. 1) and in non-volatile (persistent) flash memory to preserve the information through power-off events. Ex. 1003, 8:6-16. The flash memory preserves the address information when the memory card is powered off so that RAM can load and cache the address information after power-on. *Id.*, 8:12-16.

C. Summary of SwSTE'05

SwSTE'05 provides a survey of flash memory technologies. Ex. 1010, Abstract. Two aspects of SwSTE'05 are relevant here.

First, SwSTE'05 explains that memory devices used a FTL to remap the same virtual block number (a logical address) to different physical sectors (physical addresses) to implement wear-leveling by distributing the writes/erases in different physical locations. *Id.,* §§ 2-2.1; *see also* Baker, ¶ 119 (explaining terminology). The FTL operates by storing a logical-to-physical address mapping on the flash device itself. Ex. 1010, § 2.2. The FTL may include two forms of address mappings: "direct maps [which] allow efficient mapping of blocks to sectors, and inverse maps [which] allow efficient mapping of sectors to blocks." *Id.*

Second, SwSTE'05 teaches that a sector is reclaimed using a "garbage collection" process to make more space available. *Id.*, § 2.3.

D. Motivation to Combine Suda and SwSTE'05

A POSITA would have been motivated to combine teachings from Suda and SwSTE'05. Baker, ¶¶ 235, 241, 269. Both references discuss the management of flash storage devices. Ex. 1003, *passim*; Ex. 1010, *passim*. Both references propose techniques for improving performance given the same limitations of flash memory. Namely, that flash memory only supports erasing blocks, not erasure of individual pages within a block (Ex. 1003, 4:33-38; Ex. 1010, Abstract, § 1), and blocks cannot be directly overwritten without erasure (Ex. 1003, 1:19-22, 1:54-55; Ex. 1010 § 1).

Suda does not explain every underlying technological concept in flash memory. SwSTE'05 provides additional discussions of the technological concepts that underlie Suda's system known to a POSITA. *See* § VIII.C, *supra* (summarizing technological concepts). These underlying technological concepts from SwSTE'05 would have been easily and predictably implemented in Suda's system because flash memory devices generally implemented these technological concepts. Baker, §§ VI.A.5, VI.A.7 (describing background concepts). Section XI, *infra*, provides additional motivations for specific combinations.

IX. Ground 1: Obvious Over Bennett and POSITA Knowledge

A. Claim 15

a) Element 15[a].[1]

Bennett discloses an apparatus in the form of a host and memory system. *See* Ex. 1002, Fig. 2.

[1] *See* attached Claim Listing.

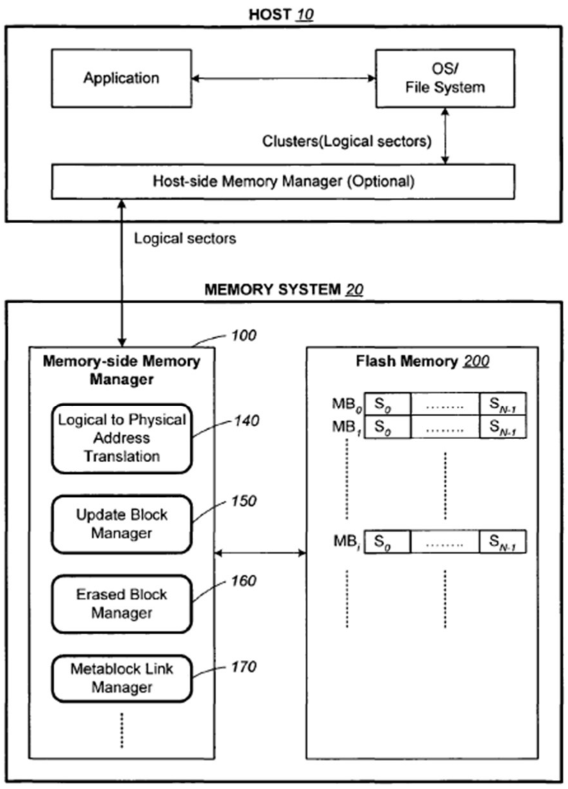


FIG. 2

Id., Fig. 2. Fig. 2 "illustrates the memory being organized into physical groups of sectors (or metablocks) and managed by a memory manager of the controller, according to a preferred embodiment of the invention." *Id.*, 5:15-18. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 121-23.

b) Element 15[b].

Bennett discloses "Flash Memory 200." Ex. 1002, Fig. 2. "[F]lash memory is non-volatile." *Id.*, 1:29-30. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶ 124.

c) Element 15[c].

Bennett teaches the recited "request receiver module" in the form of a host interface in a memory storage layer which receives commands from a host. Ex. 1002, 5:56-58. Figs. 1 and 6 illustrate the host interface:

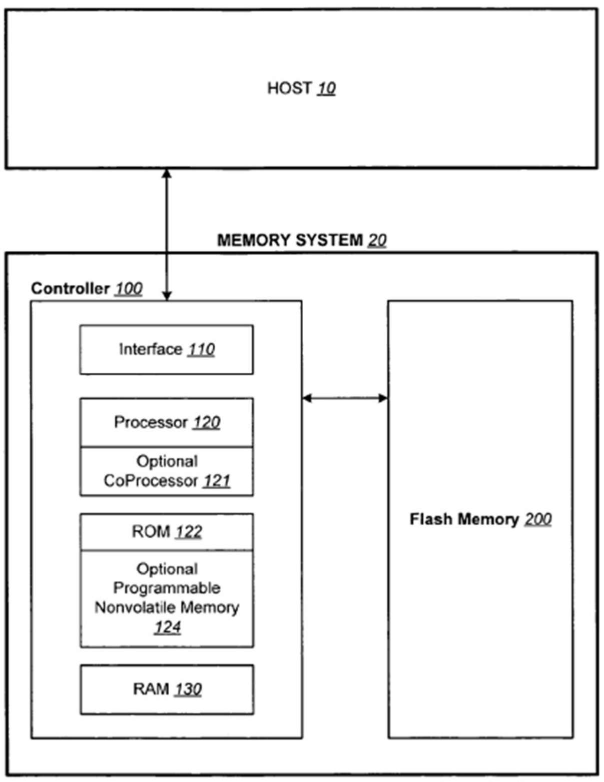


FIG. 1

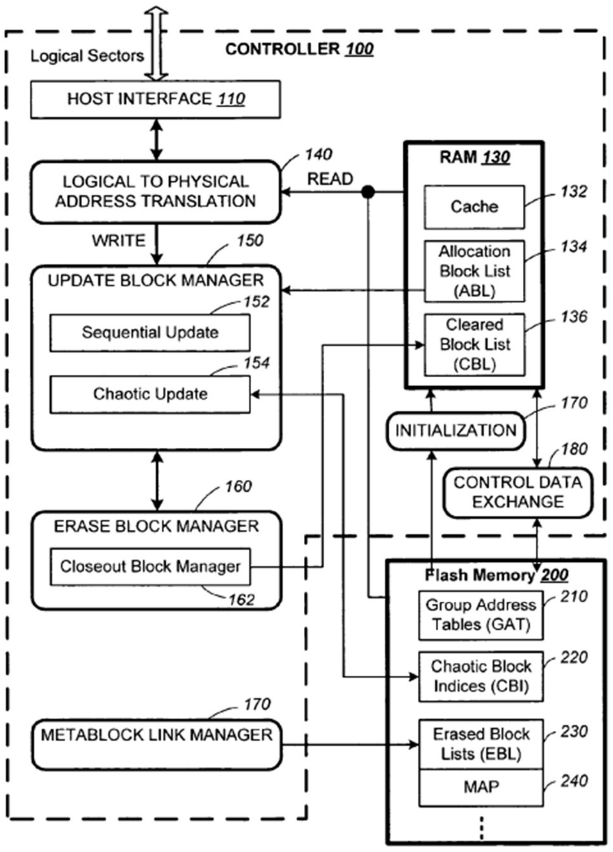


FIG. 6

Ex. 1002, Figs. 1 and 6; *see also id.* 7:2-4 ("interface 110 has one component interfacing the controller to a host and another component interfacing to the memory 200"). UTL interprets "module" as "a hardware circuit and/or programmable hardware and/or software implemented within a storage controller." *See* § VI, *supra*. Bennett's host interface includes a hardware circuit and/or programmable hardware and/or software and is in the controller. Ex. 1002, 4:50-52 (disclosing "circuitry and firmware, to execute an erase or erase equivalent").

Bennett also teaches receiving an indication that a data structure, corresponding to data stored on the non-volatile medium, has been deleted. UTL accuses receiving an indication that data associated with a host logical block address has been deleted as infringing. Ex. 1013, 2. Bennett similarly teaches "a host issu[ing] a command to erase a portion of the memory, such as an Erase Sectors command that specifies the logical sectors to erase." Ex. 1002, 12:17-15, 4:22 ("a host issues an Erase Sectors command"), Fig. 1 (showing host computer 10). A POSITA would understand that a "logical sector" is associated with a data structure. Baker, ¶ 127 (a "sector" is a specific area of memory). Bennett also teaches receiving the erase commands from a host "running an application under a file system or operating system." *Id.*, 7:21-22. Thus, from the perspective of a user, the designated data structure has been deleted and Bennett teaches this element. Baker, ¶¶ 125-29.

d) Element 15[d].

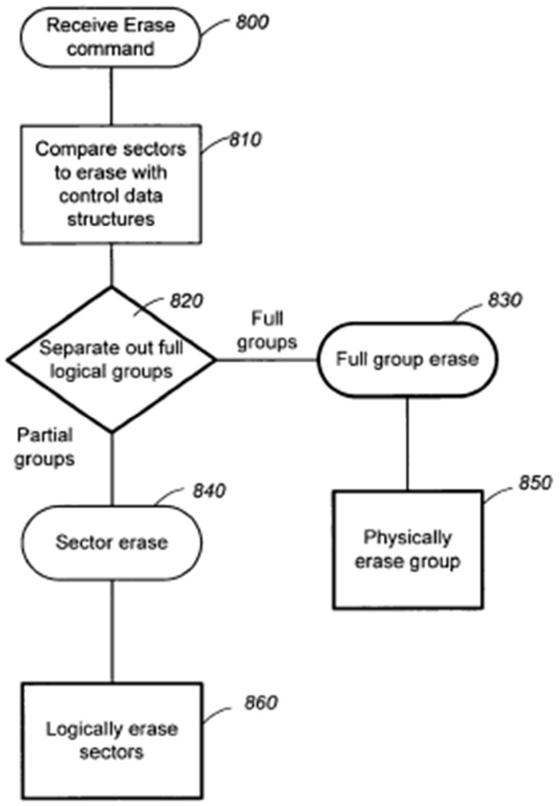
Bennett teaches that the host interface can receive an erase command message from a host. *See id.*, 5:56-58 ("an erase command, originating either from the host or with the memory system itself"). The erase command is a message that includes reference to a logical sector. *See id.*, 17:52-56 (an erase command "specifies the (logical) sectors to be erased"). The reference to a logical sector is a "logical identifier" under either proposed construction of the term. *See* § VI, *supra* (construing term); Baker, ¶ 131. For example, Bennett teaches that the logical sectors are maintained by the host computer and used to identify the logical location of data. *See* Ex. 1002, 7:22-24 ("[t]ypically, the host system addresses data in units of logical sectors"). Bennett further teaches that the logical sectors identify a particular set of data that is not the physical address of the data. *See id.*, Fig. 3B (showing mapping). *See id.*, 8:16-19 ("groups of logical sectors throughout the logical address range of the memory system are treated identically"). Thus, a POSITA would understand that the logical sector is associated with the data structure that has been deleted by a storage client under either proposed construction. Baker, ¶ 131 (host, when coupled to storage, is a storage client under either proposed construction).

Bennett further teaches a "logical to physical address translation module [that] maps the logical address from the host to a physical memory location." *Id.*, Fig. 6, 9:50-52, 10:36-38. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 130-33.

e) Element 15[e].

Bennett teaches this element under UTL's proposed construction of "marking module". Bennett teaches tables stored in flash memory ("non-volatile storage medium"). Ex. 1002, 1:29-30. These tables are marked in response to receiving erase command messages to indicate that the data associated with the logical identifier can be erased. *Id.*, 5:60-61, 20:45-47.

More specifically, Bennett's Fig. 10 illustrates the process flow for the memory controller responding to an erase message:



Ex. 1002, Fig. 10. As shown, the memory controller (at step 820) separates erase commands for "partial groups" and "full groups". *Id.* Full groups "can be then be physically erased" (step 850) and the partial groups "are logically erased" (step 860). *Id.*, 4:12-21.

For logical erasures, Bennett teaches marking sectors as logically erased, with "actual, physical erase taking place at a later time." *Id.*, 5:60-61. Bennett teaches:

> [T]he logical group can be marked as 'logically' erased in the GAT, if there is room there for an extra flag. In this case, all the data of the Logical Group will not be changed, but will not be read, as the host will be sent FFs or 00s as if the sectors were erased.

Id., 20:45-50. These GAT markings are stored in non-volatile flash memory. *See id.*, Fig. 6. The GAT provides a "list of metablock addresses for all logical groups of host data in the memory system." *Id.*, 10:65-11:8; *see also id.* 11:13-14 (GAT records "a set of GAT entries for the metablock address of each logical group within a range, and a GAT sector index"), Fig. 8A-B (depicting GAT table entries).

Thus, a POSITA would have recognized that Bennett teaches "a marking module configured to record that the data stored at the physical address mapped to the logical identifier can be erased from the non-volatile storage medium in response to receiving the indication" for partial group erasures. Baker, ¶¶ 134-39 This is true under UTL's construction of "module" because Bennett teaches implementing this functionality through a hardware circuit and/or programmable hardware and/or software in a controller. *Id.*, ¶ 139; *see also* Ex. 1002, 18:5-30.

B. Claim 16

Bennett teaches that "sectors are 'logically' erased at the sector level by standard techniques." Ex. 1002, Abstract, 6:18-20. One way that a POSITA would have known to invalidate the claimed index entry is by using a flag. Baker, ¶ 142. Bennett discloses such flags for responding to an erase command. *See, e.g.,* claim

15[e].

To the extent URL argues that setting a flag does not invalidate the entry, Bennett renders this claim obvious. Bennett teaches that "sectors are 'logically' erased at the sector level by standard techniques." Ex. 1002, Abstract, 6:18-20. A POSITA would have known standard techniques for invalidating logical-to-physical entries generally, e.g., upon receiving write commands and using superseding sector indexes to invalidate old sectors. *See, e.g.*, Ex. 1002, 11:21-25, 11:30-39. A POSITA would understand the same technique is used, or could be used, in response to an erase command. Baker, ¶ 142.

Thus, a POSITA would have understood that the marking module that Bennett teaches also performs the functionality recited in this element. Baker, ¶¶ 140-42.

C. Claim 17

Bennett teaches that "sectors are 'logically' erased at the sector level by standard techniques." Ex. 1002, Abstract; *see also id.*, 6:18-20 ("For the specified sectors not forming complete block, the system uses the system's standard, logical erase method."). A POSITA would have understood that one known technique for recording a logical erase is "deleting a mapping between the logical identifier and the physical address." Baker, ¶ 143 (citing examples of a POSITA's knowledge). For example, the Suda prior art discussed herein discloses removing the index mapping by "canceling the relation between the logical block addresses and the physical addresses." *See* §§ X.B, C, *infra* (Ground 2).

Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶ 143.

D. Claim 18

Bennett teaches a "logical to physical address translation module [that] maps the logical address from the host to a physical memory location." Ex. 1002, Fig. 6, 9:50-52, 10:36-38; *see also id.*, Fig. 8B (depicting an index mapping GAT sectors to logical addresses). The GAT index is stored on non-volatile storage. *Id.*, Fig 6.

The remaining limitations of this claim element ("marking module is configured to remove a mapping between the logical identifier and the physical addresses of the data from the index") are indistinguishable from the limitations of claim 17 and obvious for the same reasons.

Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 144-46.

E. Claim 19

Bennett teaches that "sectors are 'logically' erased at the sector level by standard techniques." Ex. 1002, Abstract; *see also id.*, 6:18-20 ("For the specified sectors not forming complete block, the system uses the system's standard, logical erase method."). A POSITA would have understood that one known technique for recording a logical erase is "deleting a reference to the physical address from an index entry of the logical identifier". Baker, ¶ 147-48 (citing examples of a POSITA's knowledge). For example, the Suda prior art discussed herein discloses erasing "address information of the physical block B and a logical address related to the address information of the physical block B from the logical and physical address table 13a." *See* § X.E, *infra* (Ground 2).

Deleting reference to physical addresses in logical-to-physical indexes was a well-known process. Many non-volatile storage devices delete the reference to an old physical address and reassign new physical addresses, after an updated write command, to point to the new data location. Baker, ¶ 148. Indeed, Bennett discloses mapping to new physical addresses after a logical group update. Ex. 1002, 8:4-6. Thus, to the extent UTL argues that Bennett does not teach this element, a POSITA would have found it obvious to indicate logical erasures by deleting reference to the old physical address and assigning a new physical address. Baker, ¶ 148. For example, Bennett's process of returning FFs or 00s (Ex. 1002, *id.*, 20:25-27) could be improved by this modification if the physical address was reassigned to point to an area of storage that already includes FFs or 00s. Baker, ¶ 148 (reassigning addresses could improve Bennett's process of writing FFs or 00s by allowing for an efficient memory read without additional write step).

Thus, a POSITA would have understood that Bennett teaches this element or found it obvious. Baker, ¶¶ 147-49.

F. Claim 20

Bennett teaches that "sectors are 'logically' erased at the sector level by standard techniques." Ex. 1002, Abstract; *see also id.*, 6:18-20 ("For the specified sectors not forming complete block, the system uses the system's standard, logical erase method."). For the reasons discussed with respect to claim 17, a POSITA would have understood that one known technique for recording a logical erase is deleting/removing "a mapping between the logical identifier and the physical address." Furthermore, for the reasons discussed with respect to claim 19, a POSITA would have found it obvious to indicate logical erasures by deleting reference to the old physical address and assigning a new physical address to an area of storage reciting FFs or 00s. Thus, a POSITA would have understood that Bennett teaches this element or found it obvious. Baker, ¶¶ 150-51.

G. Claim 21

Bennett teaches this claim. In the district court litigation, UTL appears to interpret "data packet" as "the area of the [memory] that contains the data." Ex. 1013 6. Bennett similarly teaches invalidating areas of memory through index marking. *See, e.g.,* § IX.A.e (setting flags in non-volatile GAT), *supra*. A POSITA would understand that the use of flags is one way to invalidate a data packet in response to the erase command indication. Baker, ¶ 152.

Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 152-53.

H. Claim 26

a) Element 26[a].

Bennett discloses "Flash Memory 200." Ex. 1002, Fig. 2. "[F]lash memory is non-volatile." *Id.*, 1:29-30. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 154-55.

b) Element 26[b].

UTL asserts that this element is met by "circuitry and/or software/firmware" that implements "garbage collection … by returning the physical memory to the point where it can be written again." Ex. 1013, 5. Similarly, a POSITA would understand that Bennett's disclosure of responding to erase commands, specifying a full logical group, by actually erasing the physical memory, returns the memory to a point where it can be written again. Baker, ¶ 156.

Bennett also teaches garbage collection. Ex. 1002, 19:10, 20:32. "Garbage collection" was a well-known process for erasing data in a background process at a convenient time. Baker, ¶ 157; compare Ex. 1002, 3:26-32 ("in more advanced memory systems it is common for the designated portions not to be erased immediately at that time, but to be 'logically erased' by being marked for erase, with the actual, physical erase taking place at a later time"). A POSITA would have understood that a garbage collection process is implemented by circuitry and/or software/firmware in a controller and thus encompasses a "storage recovery module". *See, e.g.*, § VI, *supra* (construing "module"); Baker, ¶ 158. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 156-59.

c) Element 26[c].

As described above for claim 26[b], Bennett discloses recovering the physical storage location by erasing the data in response to an erase command or through garbage collection process. After either process, the memory is at a point where it can be written again. Baker, *¶ 160. Bennett discloses a storage module for writing to the erased storage. *See, e.g.,* Ex. 1002 at 2:33-35 ("peripheral circuits such as decoders and erase, write and read circuits"), 3:15-17 ("the entire erase block contain that physical location will have to be first erased and then rewritten with the updated data"). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 160-61.

I. Claim 27

a) Element 27[a].

Bennett discloses a host and memory system. *See* Ex. 1002, Fig. 2.

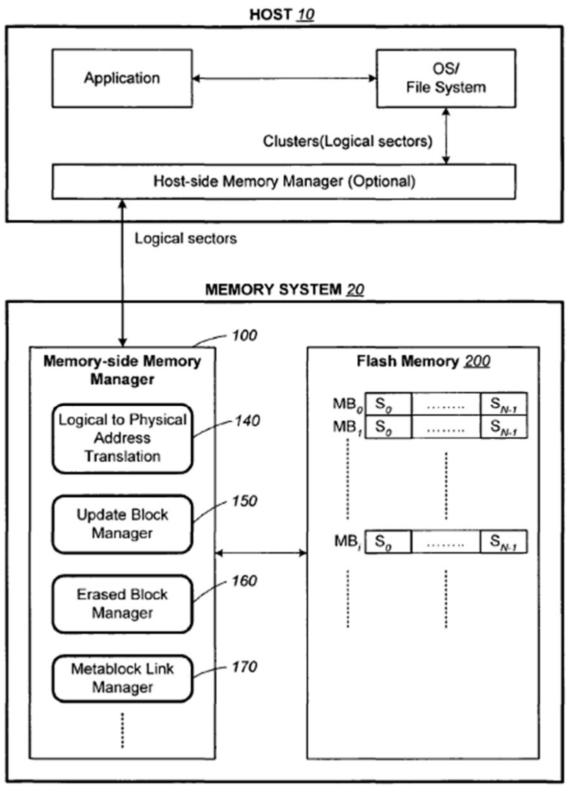


FIG. 2

Id., Fig. 2. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶ 162.

b) Element 27[b].

Bennett discloses "Flash Memory 200." Ex. 1002, Fig. 2. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 163-64.

c) Element 27[c].

Bennett discloses a storage layer for the flash storage medium by its disclosure of modules for performing storage operations, such as the flash memory cells controlled by the memory side manager 100 depicted in Bennett's Fig. 2. *See* Ex. 1002, Fig. 2, 6:63-66 ("The memory 200 comprises of one or more array of non-volatile memory cells distributed over one or more integrated circuit chip."). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 165-66.

d) Element 27[d].

Bennett teaches an "index module" in the form of a "logical to physical address translation module [that] maps the logical address from the host to a physical memory location." Ex. 1002, Fig. 6, 9:50-52, 10:36-38; *see also id.* Fig. 8B (depicting an index mapping GAT sectors to logical addresses). A POSITA would understand that Bennett teaches an index module (under UTL's proposed construction of "module") for maintaining these indexes. *See, e.g.,* Ex. 1002, 4:50-52 (disclosing "circuitry and firmware, to execute an erase or erase equivalent"); Baker, ¶ 167. A POSITA would also understand that these indexes include a plurality of entries as claimed. *Id.* (Bennett teaches multiple address and each index entry maps a logical address to a physical address); *see also* Bennett's Figs. 2, 3A, 4, 8A, 8B (disclosing address mapping indexes with a plurality of index entries). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 167-68.

e) Element 27[e].

First, as discussed for claim 15[c]-[d], Bennett teaches a request receiver module receiving an indication (an erase command) comprising a logical identifier (under either proposed construction) that a data structure, corresponding to data stored on the non-volatile medium, has been deleted. *See* § IX.A.c-d, *supra*.

Second, a POSITA would understand that Bennett's erase command indicates "a logical identifier that is empty" under UTL's proposed construction as "data identified by the [logical identifier] that does not need to be preserved." *See supra* § VI, (construing term). Specifically, Bennett's erase command indicates that data has been erased by a host (from a user's perspective) and thus the associated data in storage does not need to be preserved. *See* § IX.A.c-e, *supra*. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 169-74.

f) Element 27[f].

A POSITA would understand that Bennett teaches a logical identifier associated with a data identifier of a storage client. Baker, Decl., ¶ 176. For example, A POSITA would understand that Bennett describes a host computer that processes application files. *See, e.g.,* Ex. 1002, Fig. 2 (host 10 has OS/File System that run applications). A POSITA would also understand that a file name is a data identifier. Baker, ¶ 178. A POSITA would also understand that logical sectors (the logical identifier) are associated with files. *See, e.g.,* Ex. 1002, 20-25 ("The host 10 accesses the memory 200 when running an application under a file system or operating system. Typically, the host system addresses data in units of logical sectors ...."); Baker, ¶ 176.

The remaining limitations are indistinguishable from the limitations of claim 15[d] and would have been obvious to a POSITA for the same reasons. *See* § IX.A.d, *supra*.

g) Element 27[g].

Bennett teaches this element under UTL's proposed construction of "index module". Bennett teaches tables stored in flash memory ("non-volatile storage medium"). Ex. 1002, 1:29-30. These tables are marked in response to receiving erase command messages to indicate that the data associated with the logical identifier can be erased. *Id.*, 5:60-61, 20:45-47. *See also* § IX.A.e, *supra.* Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 179-80.

h) Element 27[h].

Bennett teaches updating the index by "removing an entry from the index that associates the specified logical identifier with the physical storage location." *See* § IX.C-F, *supra.* Bennett also teaches adding an entry to the index either by adding a flag (*see* § IX.B, *supra*) or by otherwise updating the GAT. *See* Ex. 1002, 11:64-67 ("during a GAT update operation, one GAT sector has entries updated with information from corresponding entries in the closed update block list."). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 181-82.

J. Claim 28

UTL asserts that this element is met by "circuitry and/or software/firmware" that implements "garbage collection … by returning the physical memory to the point where it can be written again." Ex. 1013, 6. Similarly, a POSITA would understand that Bennett's disclosure of actually erasing data in response to an erase command specifying a full logical group returns the memory to a point where it can be written again. Baker, ¶ 183.

Bennett also teaches garbage collection. Ex. 1002, 19:10, 20:32. "Garbage collection" was a well-known process for erasing data in a background process at a convenient time. Baker, § VI.A.5; *compare* Ex. 1002, 3:26-32 ("in more advanced memory systems it is common for the designated portions not to be erased immediately at that time, but to be 'logically erased' by being marked for erase, with the actual, physical erase taking place at a later time"). A POSITA would have understood that a garbage collection process is implemented by circuitry and/or software/firmware in a controller and thus encompasses a "storage recovery module." *See, e.g.*, § VI, *supra* (construing "module"); Baker, ¶ 184. A POSITA would have found it obvious to perform garbage collection in response to removing an entry, for example, in order to determine whether there are other areas that need to be efficiently erased at the same time. *Id.* Indeed, Bennett teaches "it is desirable to have the erase block of substantial size … [so the] erase time is amortized over a large aggregate of memory cells." Ex. 1002, 3:7-9. Thus, a POSITA would have understood that Bennett teaches this claim. Baker, ¶¶ 183-185.

K. Claim 29

a) Element 29[a].

Bennett teaches that the indication is an erase command. Ex. 1002, 4:22 ("When a host issues an Erase Sectors command"). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶ 186.

b) Element 29[b].

Bennett teaches receiving erase commands via a host interface. *See* § IX.A.c, *supra*. Bennett teaches "execut[ing] the command by the means of physical block erases" for full logical groups or using "the system's standard, logical erase method" for partial groups. Ex. 1002, 6:14-20. Bennett further discloses storage in terms of blocks throughout. *See, e.g.,* Abstract. From these teachings, a POSITA would have understood the host interface to be the claimed a "request receiver module" that is "configured to receive the command through one of a block device interface and via a block storage protocol." Baker, ¶¶ 187-92.

c) Element 29[c].

A POSITA would understand that Bennett teaches erase commands that mark data as invalid, e.g., through use of setting a flag in a GAT table. *See* § IX.B, *supra*. A POSITA would also understand that Bennett teaches a storage division recovery module as claimed. *See* § IX.H.b, *supra*; Ex. 1002, 3:27-32 (when "the memory receives a command to erase a particular portion, in more advanced memory systems it is common for the designated portions not to be erased immediately at that time, but to be "logically erased" by being marked for erase, with the actual, physical erase taking place at a later time"), 20:32-34 ("garbage collection"). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 193-96.

L. Claim 30

a) Elements 30[a]-30[f].

A POSITA would have found these claim elements obvious in view of Bennett for the same reasons discussed with respect to the identically worded elements in claim 27[a]-[f].

b) Element 30[g]

Bennett teaches that logically erased data "will not be changed" in the underlying memory, but any read attempts of the logically erased data will result in the return of "FFs or 00s as if the sectors were erased." Ex. 1002, 20:47-50, 4:50-54 ("an erased data pattern can be sent to the host if it reads a sector from the erased logical grouping"). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶ 203.

X. Ground 2: Obvious Over Suda and POSITA Knowledge

A. Claim 15

a) Element 15[a].[2]

Suda discloses an apparatus in the form of a "memory card" or "memory

[2] *See* attached Claim Listing.

device" in Fig. 1. Ex. 1003, 2:38-62. Baker, ¶¶ 205-206.

b) Element 15[b].

"The memory card 1 comprises . . . a flash memory 14. The flash memory 14 is provided as, e.g., a NAND type nonvolatile memory." Ex. 1003, 2:63-66, Fig. 1 (annotated below). Baker, ¶ 207.

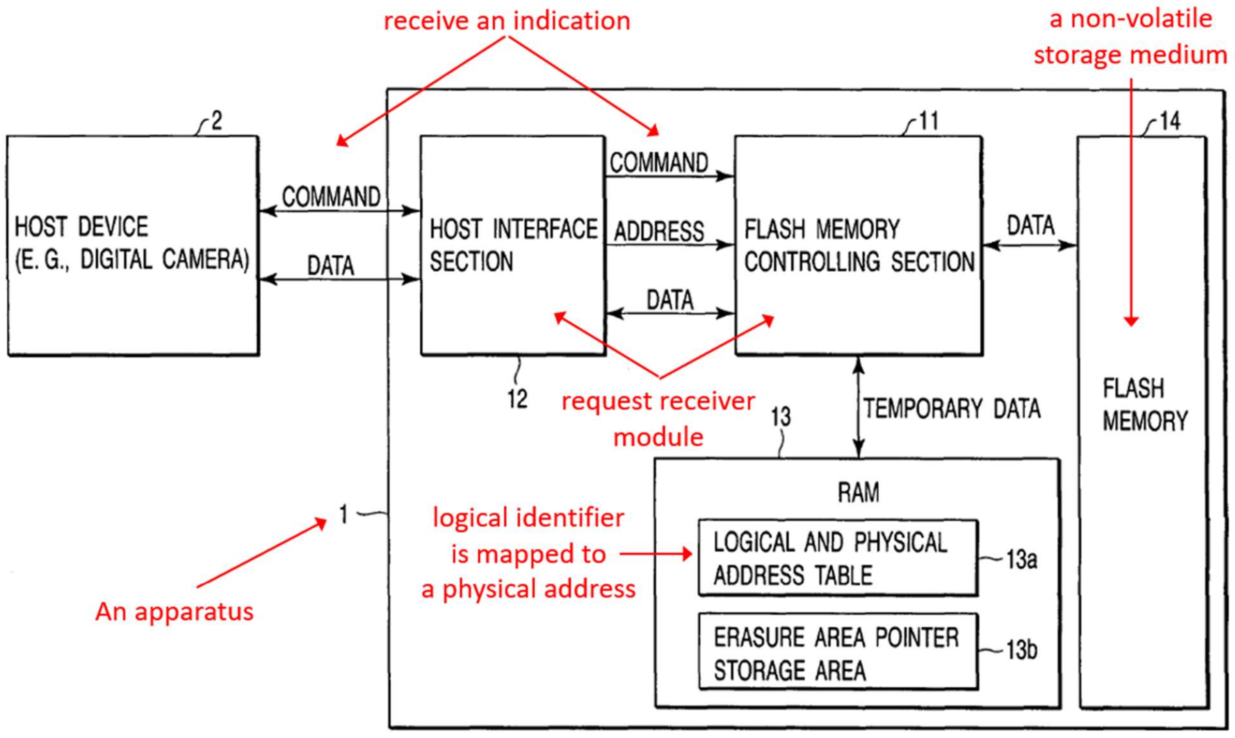

receive an indication
a non-volatile storage medium
2
11
14
HOST DEVICE (E. G., DIGITAL CAMERA)
COMMAND
DATA
HOST INTERFACE SECTION
COMMAND
ADDRESS
DATA
FLASH MEMORY CONTROLLING SECTION
DATA
FLASH MEMORY
12
request receiver module
13
TEMPORARY DATA
RAM
LOGICAL AND PHYSICAL ADDRESS TABLE
13a
ERASURE AREA POINTER STORAGE AREA
13b
1
logical identifier is mapped to a physical address
An apparatus

FIG. 1

Ex. 1003, Fig. 1 (annotated).

c) Element 15[c].

Suda's Fig. 1 shows that the memory card has a "host interface section," "flash memory controlling section," and "logical and physical address table." Ex. 1003, Fig. 1. A POSITA would have recognized Suda's flash memory controlling section or host interface sections, both of which receives commands originating from a host device, as the “request receiver module.” Ex. 1003, Fig. 1 (annotated above at element 15[b]), 2:63-39. Of these, a POSITA would have understood that the flash memory controlling section fits UTL’s interpretation of “module” as “a hardware circuit and/or programmable hardware and/or software implemented within a storage controller.” Baker, ¶ 209; *see* § VI, supra.

The received commands include erase commands that designate a logical identifier. Ex. 1003, 7:11-18, 7:30-34, 8:66-9:3 (“the logical block address ‘0x40000’ designated in the erase command.”). Both constructions of “logical identifier” encompass a logical block address. Ex. 1013, 2 (UTL accusing a “logical block address” of infringement); Baker, ¶ 210; *see* § VI, *supra* (construing term).

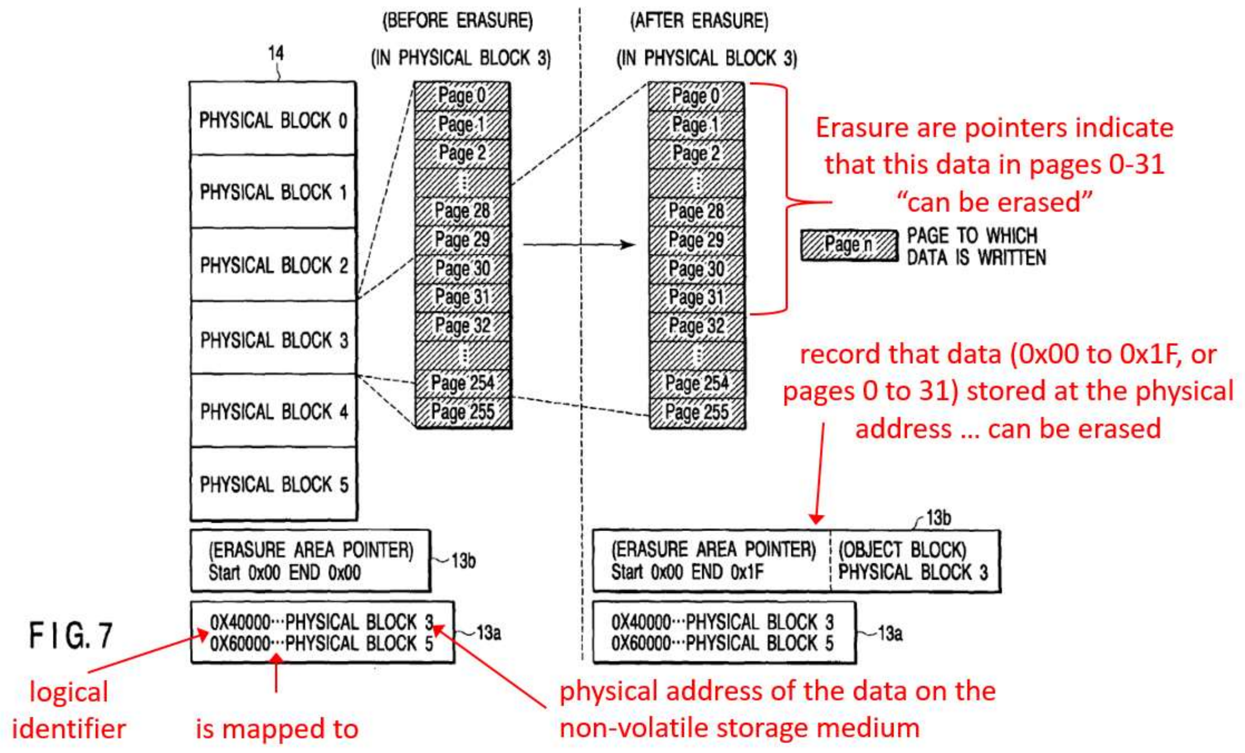
A POSITA would have understood that, in the context of Fig. 1 where Suda’s host device is a digital camera connected to the memory device, the erase command (the recited “indication”) indicates that a digital photo (the recited “data structure”) has been selected for deletion by a user. Baker, ¶ 211. This digital photo, located at logical block address 0x4000, would have a data structure in the form of a .jpg, .gif, .raw, or any file system data structure, and this digital photo would be stored as data on the flash memory of the memory device. *Id.* Thus, a POSITA would have understood Suda’s erase command to be indication that a picture file, corresponding to digital photo data stored on the flash memory, has been deleted by a user. *Id.*, ¶

208-12; Baker, ¶¶ 208-12.

d) Element 15[d].

As discussed for preceding claim 15[c], Suda provides a digital camera example where "an erase command is issued from the host device 2" with "the logical block address '0x40000' designated in the erase command." Ex. 1003, 8:66-9:3, 7:11-18. This logical block address 0x4000 is an example of the claimed "logical identifier." *See* Ex. 1013, 2 (accusing LBA of infringement). Because the digital camera (the claimed "storage client") designates this logical identifier in the erase command when sending the command in response to a photo deletion, this logical identifier was associated with the data structure by the storage client. Baker, ¶¶ 213-14; Ex. 1003, 8:66-9:3, 7:11-18, 7:30-34.

Suda uses a logical and physical address table to map block numbers (the claimed "physical addresses of data…") with logical block addresses. Ex. 1003, 3:42-55, Fig. 1 (annotated above at § X.A.b), Fig. 7 (annotated below).



(BEFORE ERASURE)
(IN PHYSICAL BLOCK 3)
(AFTER ERASURE)
(IN PHYSICAL BLOCK 3)
14
PHYSICAL BLOCK 0
PHYSICAL BLOCK 1
PHYSICAL BLOCK 2
PHYSICAL BLOCK 3
PHYSICAL BLOCK 4
PHYSICAL BLOCK 5
Page 0
Page 1
Page 2
Page 28
Page 29
Page 30
Page 31
Page 32
Page 254
Page 255
Erasure are pointers indicate that this data in pages 0-31 "can be erased"
Page n PAGE TO WHICH DATA IS WRITTEN
record that data (0x00 to 0x1F, or pages 0 to 31) stored at the physical address … can be erased
(ERASURE AREA POINTER) Start 0x00 END 0x00
13b
(ERASURE AREA POINTER) Start 0x00 END 0x1F
(OBJECT BLOCK) PHYSICAL BLOCK 3
0X40000···PHYSICAL BLOCK 3
0X60000···PHYSICAL BLOCK 5
13a
FIG. 7
logical identifier
is mapped to
physical address of the data on the non-volatile storage medium

Ex. 1003, Fig. 7 (annotated).

e) Element 15[e].

Suda uses "erasure area pointers" to record which physical storage locations are in a "virtual erased state" and "subjected to virtual erasure" in response to an erase command. Ex. 1003, 5:9-61, Fig. 7 (element 13b), Figs. 3-6. In other words, the erasure area pointers indicate data "to be erased" later from the flash memory. *E.g., id.*, 5:40-48, 5:57-61, 6:9-14, 6:29-45, 6:60-64, 7:38-41. In fact, once an entire block is designated by erasure pointers, the block will be erased and set to an unused state. *Id.*, 5:54-6:3, 6:34-41, Fig. 4, Fig. 6 (block B). Thus, a POSITA would have understood that Suda teaches to use erasure area pointers to record that the data stored at the physical address (e.g., physical block 3 in Suda Fig. 7) mapped to the logical identifier (e.g., logical identifier 0x4000 in Suda Fig. 7) "can be erased" from the flash memory, as claimed. *Id.*, Fig. 7 (annotated above at § X.A.d); Baker, ¶ 217. The erasure area pointers are set in response to an erase command. Ex. 1003, 5:38-43, 6:18-21, 7:5-55, Fig. 8 S1-S4.

Suda discloses a flash memory controlling section 11 that operates in response commands, manages data erasure and a table indicating a relationship between logical blocks and physical blocks, and determines the address ranges of erasure area pointers. Ex. 1003, Fig. 1, 3:9-15, 5:19-23, 5:38-43. A POSITA would have understood Suda's flash memory controlling section 11 to be the "marking module" configured to perform the operations as recited in claim 15, even under UTL's construction of "module." Baker, ¶¶ 218-19; *see* § VI (construing "module").

B. Claim 16

Suda teaches that "the marking module is configured to record that data stored at a physical address on the non-volatile storage medium can be erased from the non-volatile storage medium" for the reasons discussed for claim 15.e. *See* § X.A.e, *supra*. Using the examples of Figures 4 and 6, Suda teaches that the recording further includes "invalidating an association between the logical identifier and the physical address," as claimed. In these scenarios, an erase command is received to delete at least an entire block. Ex. 1003, 6:15-21, Figs. 4, 6.

Using the example in Fig. 4, Suda teaches "canceling the relation between the logical block addresses and the physical addresses ('A'), which is indicated by the logical and physical address table 13a." Ex. 1003, 5:65-6:3. Using the example in Fig. 6, Suda teaches that the flash memory controller "erases address information of the physical block B and a logical address related to the address information of the physical block B from the logical and physical address table 13a." *Id.*, 6:35-41. A POSITA would have understood both the act of "canceling the relation" and the act that "erases address information" of the logical block address and the physical address are acts that invalidate the association between those addresses, as claimed. Baker, ¶ 222.

C. Claim 17

Suda teaches that "the marking module is configured to record that data stored at a physical address on the non-volatile storage medium can be erased from the non-volatile storage medium" for the reasons discussed for claim 15[e]. *See* § X.A.e, *supra*. For the reasons discussed for claim 16, a POSITA would have understood Suda to teach that the recording further includes "deleting a mapping between the logical identifier and the physical address," as recited in claim 17 for cases where at least an entire block is being deleted. *See* § X.B, *supra*; Baker, ¶¶ 224-26.

Using the example in Fig. 4, Suda teaches "canceling the relation between the logical block addresses and the physical addresses ('A'), which is indicated by the logical and physical address table 13a." Ex. 1003, 5:65-6:3. Using the example in

Fig. 6, Suda teaches that the flash memory controller "erases address information of the physical block B and a logical address related to the address information of the physical block B from the logical and physical address table 13a." *Id.*, 6:35-41. A POSITA would have understood both the act of "canceling the relation" and the act that "erases address information" of the logical block address and the physical address are acts of "deleting a mapping between the logical identifier and the physical address," as claimed. Baker, ¶¶ 224-26.

D. Claim 18

Suda's controller maintains a logical and physical address table as part of an index of mappings between logical addresses (the recited "logical identifiers") and physical addresses. *E.g.*, Ex. 1003, 3:33-55, Fig. 1 (element 13a), Fig. 7 (annotated at § X.A.d, *supra*.

The remaining limitations of claim 18 ("marking module is configured to remove a mapping between the logical identifier and the physical addresses of the data from the index") are indistinguishable from the limitations of claim 17 and are obvious for the same reasons. Baker, ¶¶ 227-28; *see* § X.C, *supra*.

E. Claim 19

Suda teaches this claim for the reasons given in the discussion of claim 17. *See* § X.C, *supra* (explaining that Suda teaches "canceling the relation between the logical block addresses and the physical addresses" and "erases address information of the physical block B and a logical address related to the address information of the physical block B from the logical and physical address table 13a."). For the same reasons, a POSITA would have understood Suda to teach claim 19. Baker, ¶¶ 229-30.

F. Claim 20

For the reasons discussed with respect to Claim 18, Suda teaches to "remove a mapping between the logical identifier and the physical addresses of the data from the index." For the following reasons, this removal indicates that the data stored at the physical address can be erased from the non-volatile storage medium.

Suda teaches, with respect to the example of Fig. 4, to cancel (the claimed "removal") the relation (the claimed "mapping") between the logical block address and a physical address indicated by the logical and physical address table 13a. Ex. 1003, 5:65-6:3. This cancellation in performed "in order that a physical block … be set in an unused state," which refers to an erased state. *Id.*, 5:65-6:3. This means that the cancellation is performed to indicate that the block (including the data therein) can and will be erased. Baker, ¶ 231.

In the example of Fig. 6, an erase command designates more than one block (block B) for erasure. Ex. 1003, 6:15-21. Then, block B has its physical address information and logical address information erased from logical to physical address table 13a, "thereby setting the entire area … of the physical block B in an unused state." *Id.*, 6:33-41. This means that the erasure of the mapping of block B indicates that block B (including the data therein) will be erased and set to an unused state, as claimed. Baker, ¶ 231.

G. Claim 21

As discussed for claim 15[e], Suda teaches the claimed "marking module." *See* § X.A.e, *supra*.

Suda also teaches to use "erasure area pointers" to mark data packets at a physical address as in a "virtual erased state." Ex. 1003, 5:9-61, Fig. 7 (element 13b), Figs. 3-6 (showing examples of erasure area pointers). Virtual erased data is "subjected to virtual erasure" and can no longer be read by a user because the system will return initial-value (empty) data instead of the actual data stored therein. *Id.*, 5:23-27, 8:21-38, Fig. 9. A POSITA would have understood that a virtually erased state is an invalid state. Baker, ¶ 223. Thus, a POSITA would have understood Suda teaches to use erasure area pointers to mark data packets at the physical address invalid, as claimed. *Id.* For example, a POSITA would have understood data packets in the shaded areas illustrated in Suda Figs. 4-6 to be marked as invalid.

H. Claim 26

a) Element 26[a].

Suda teaches this element because Suda specifies, "The flash memory 14 is provided as, e.g., a NAND type nonvolatile memory." Ex. 1003, 2:65-66, Fig. 1. Baker, ¶¶ 237-38.

b) Element 26[b].

Suda teaches to erase the contents of blocks in a storage recovery process whenever the erasure area pointers indicate an entire block contains virtually erased data. For example, Suda instructs to perform "Erasure processing on entire block" in step S6 of Fig. 8 after the erasure area pointer is updated (S4) such that it equals a block size (S5). In a second example, with respect to the block illustrated in Fig. 4, Suda instructs that virtually erased, physical block (area 22) is "to be erased" and to "be set in an unused state." Ex. 1003, 5:65-6:3, 3:56-67 (explaining that blocks in an unused state have "initial-value" data and "can be used" again). In a third example, with respect to physical block B in Fig. 6, Suda discloses, "the data items written to the entire area of the physical block B are to be erased," and further instructs "setting the entire area (area 25) of the physical block B in an unused state." Ex. 1003, 6:33-41. A POSITA would have understood these parts of Suda to teach recovering the physical storage locations at the physical address, as claimed. Baker, ¶ 240.

A POSITA would also have understood that Suda's system is configured to perform the storage recovery process using a hardware circuit and/or programmable hardware and/or software implemented within a storage controller in the form of Suda's flash memory controlling section. Baker, ¶ 239; Ex. 1003, 3:13-15, 6:33-41; *see* § VI, *supra* (construing "module").

c) **Element 26[c].**

As discussed above for claim 26[b], Suda teaches a process to erase and recover blocks (physical storage locations) to set them in an "unused state." Ex. 1003, 5:65-6:3, 6:35-41. Later, the recovered block ("the physical storage location") will be mapped with a new logical identifier and then used to store data associated with the new logical identifier. *Id.*, 3:64-67. This mapping occurs when a write command is received and the recovered block's "physical block address is related to a logical block address" by a flash memory controlling section. *Id.*, 3:48-55 (describing address assignment process); Baker, ¶ 243. The flash memory controlling section 11 will assign or relate this new logical address with a physical block address of this unused block and store the new data there. *Id.*, 3:48-55 (describing address assignment process); Baker, ¶ 243.

A POSITA would also have understood that Suda taught the claimed "storage module" because Suda taught a hardware circuit and/or programmable hardware and/or software implemented within a storage controller in Suda's flash memory controlling section. Baker, ¶ 244; Ex. 1003, 3:13-15. For these reasons, a POSITA would have understood that Suda taught that "a storage module configured to store data associated with another logical identifier on the physical storage location in response to recovering the physical storage location" as recited in claim 26. Baker, ¶¶ 243-44.

I. Claim 27

a) Element 27[a].

Suda teaches the claimed system in the form of a memory card or memory device. Ex. 1003, 2:61-3:3, Fig. 1; Baker Decl, ¶ 246. An annotated version of Suda Fig. 1 shows various claimed components:

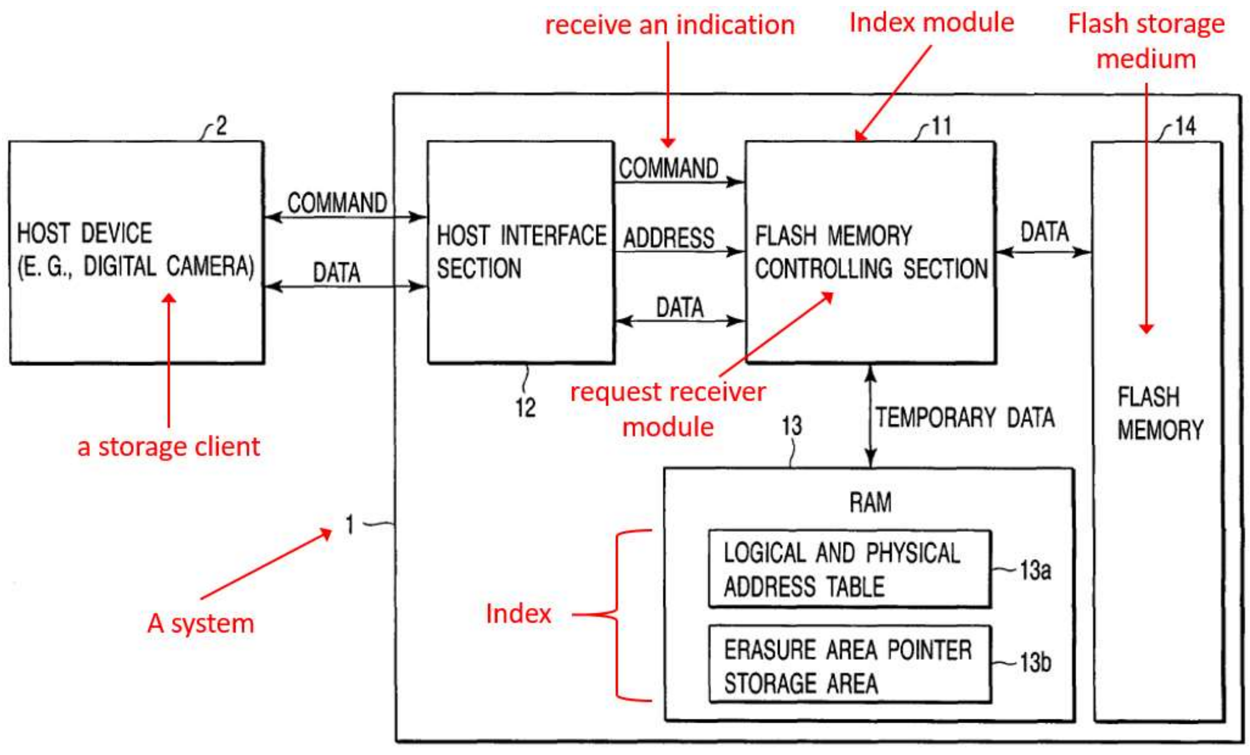


FIG. 1

Ex. 1003, Fig. 1 (annotated).

b) Element 27[b].

Suda teaches, "The memory card 1 comprises . . . a flash memory 14. The flash memory 14 is provided as, e.g., a NAND type nonvolatile memory." Ex. 1003, 2:63-66.

c) Element 27[c].

Suda's Fig. 1 shows that the memory card has a "host interface section," a "flash memory controlling section," a logical and physical address table, an erasure area pointer storage area, and flash memory. These components implement storage operations. Ex. 1003, *passim*. UTL contends that the storage layer refers to the parts "where storage operations are implemented." Ex. 1013, 5. A POSITA would have understood these parts of Suda's system to make up the storage layer under UTL's interpretation because these components implement storage operations. Baker, ¶ 249.

d) Element 27[d].

Suda maintains the logical to physical address table 13a and erasure area pointers 13b as the claimed "index." Ex. 1003, Fig. 1 (annotated at § X.I.a, *supra*), Fig. 7 (annotated below). This index includes a plurality of entries. *E.g., id.*, 3:43-55, Fig. 7. The entries associate a logical identifier (a logical address, such as 0x40000) with a physical storage location (a block address, such as block 3). *E.g.*, *id*.

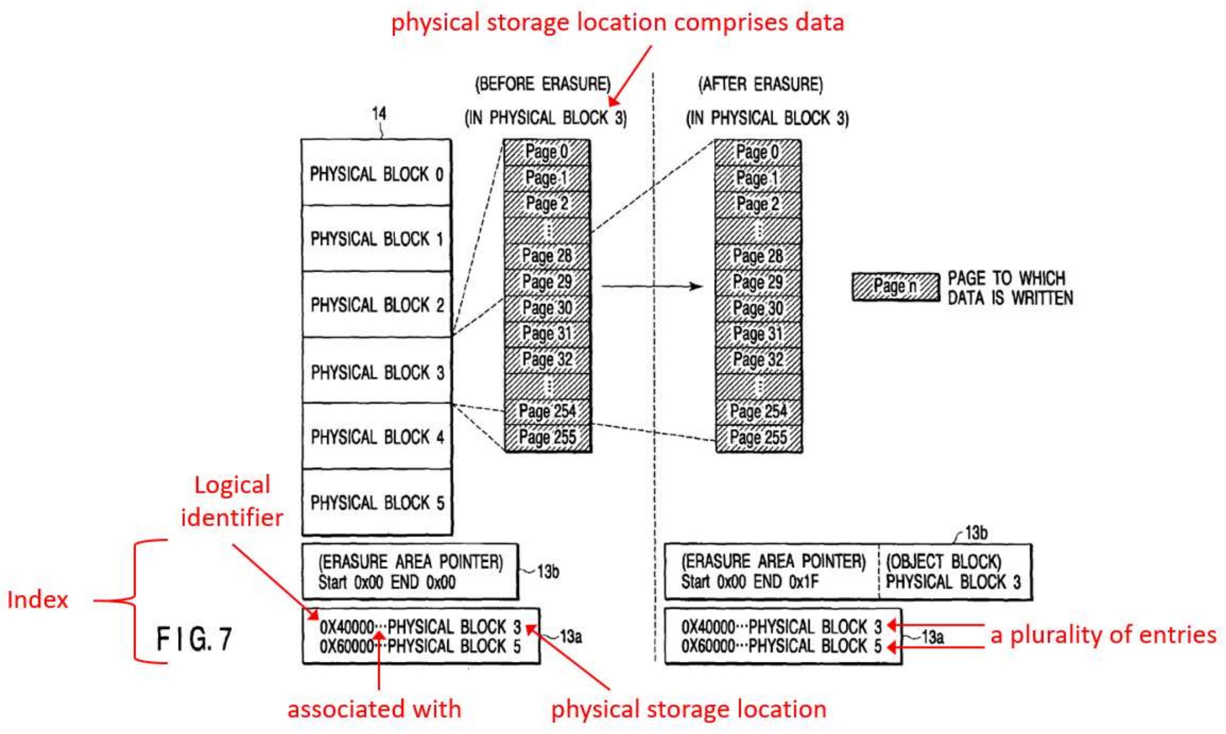


FIG. 7

The flash memory controlling section "manages" this "table indicating a relationship between logical blocks and physical blocks of the flash memory." *Id.*, 3:13-15. A POISTA would have understood this flash memory controlling section to be the claimed "index module," under the patent owner's construction because the flash memory controlling section includes "a hardware circuit and/or programmable hardware and/or software implemented within a storage controller for managing the claimed index. Baker, ¶ 251; *see* § VI, *supra* (construing module). Thus, a POSITA would have understood Suda to teach this claim. Baker, ¶¶ 250-52.

e) Element 27[e].

For the reasons discussed for claim 15[c], a POSITA would have understood

Suda to teach the request receiver module configured to receive an indication in the form of an erase command. *See* §§ X.A.c, *supra* (discussing element), X.I.a, *supra* (showing annotated system picture). For the reasons discussed for claim 15[d], a POSITA would have understood Suda to teach the indication comprising a logical identifier. *See* § X.A.d, *supra*; Ex. 1003, 7:11-18, 7:30-34, 8:66-9:3 ("the logical block address '0x40000' designated in the erase command.").

UTL contends that "a logical identifier that is empty" means "data identified by the [logical identifier] that does not need to be preserved." *See supra* § VI, (construing term). This is what Suda's erase command indicates. Baker, ¶ 255. As explained for claim 15[c], Suda teaches a system where the host device is a digital camera that issues an erase command, for example, when a user deletes a digital photo. *See* § X.A.c, *supra*. From the perspective of the memory device, this erase command indicates that the data identified by the logical identifier in the erase command are "to be erased," therefore, the contents do not need to be preserved. Ex. 1003, abstract, 1:65-67, 5:19-27, 6:34-36, 6:60-62, 9:5, 9:12-13; Baker, ¶¶ 253-56.

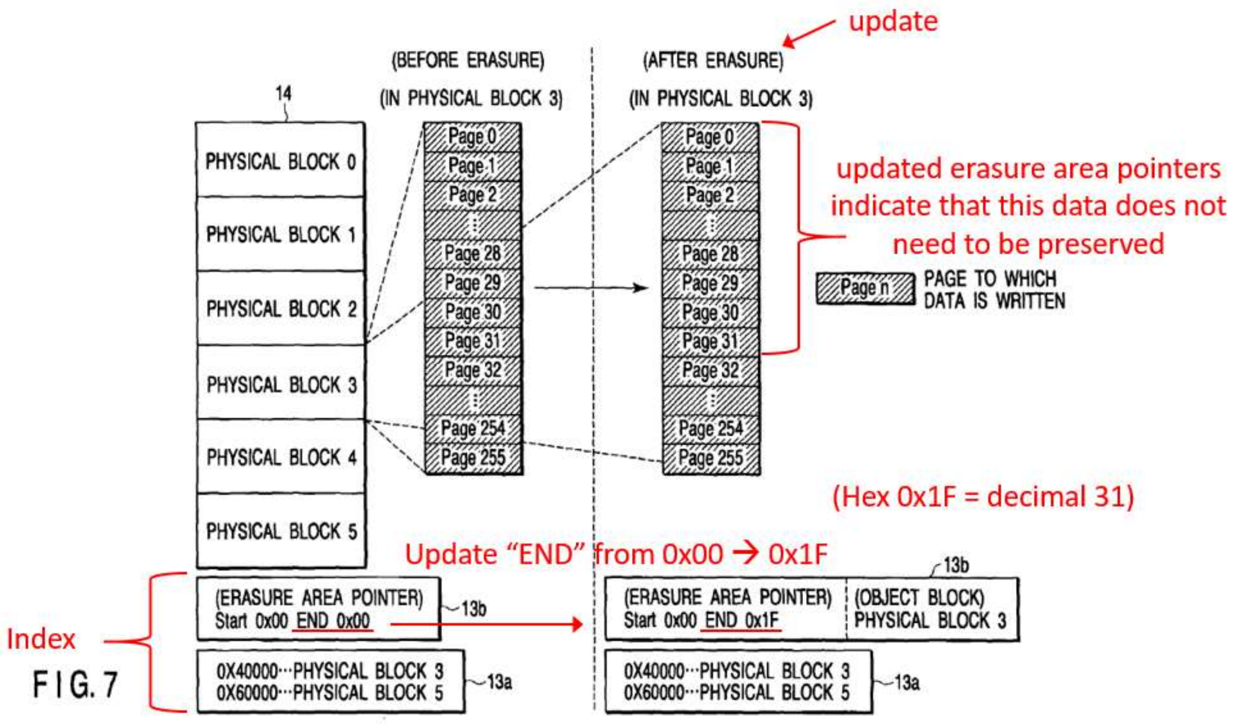
f) Element 27[f].

A POSITA would have understood Suda to teach that the logical identifier, which is designated by the host (the claimed "storage client") in an erase command, is associated with some type of data identifier. Baker, ¶ 257; *see* § X.I.a, *supra* (showing annotated Fig. 1). As discussed for claim 15[c], Suda provides an example where a digital camera issues erase commands. *See* § X.A.c, *supra*; Ex. 1003, 2:61-62. A POSITA would have known that digital cameras associate file names with the pictures that they take, such as IMG001.jpg or named in the format yyyyMMdd.raw. These file names are examples of the claimed "data identifiers." Baker, ¶ 257. The POSITA would have understood that when a user selects one of these named files for deletion, the digital camera then sends an erase command designating the logical identifier that is associated with the file name, as described in Suda. *Id.*; Ex. Baker, ¶ 257; Ex. 1003, 2:61-62, 7:11-18, 7:30-34, 8:66-9:3.

Suda's system also uses the logical and physical address table to associates block numbers (the claimed "physical storage location") with logical identifiers. Ex. 1003, Fig. 1, Fig. 7 (annotated at § X.I.d, *supra*), 3:42-55. Thus, a POSITA would have found this claim obvious in view of Suda. Baker, ¶¶ 257-59.

g) Element 27[g].

As discussed for claim 27[d], Suda teaches the "index module" under UTL's proposed construction. *See* § X.I.d, *supra*. Suda teaches to update its erasure area pointers in response to the erase command (the claimed indication), as illustrated in the following annotated Fig. 7:


update
(BEFORE ERASURE)
(IN PHYSICAL BLOCK 3)
(AFTER ERASURE)
(IN PHYSICAL BLOCK 3)
14
PHYSICAL BLOCK 0
PHYSICAL BLOCK 1
PHYSICAL BLOCK 2
PHYSICAL BLOCK 3
PHYSICAL BLOCK 4
PHYSICAL BLOCK 5
Page 0
Page 1
Page 2
Page 28
Page 29
Page 30
Page 31
Page 32
Page 254
Page 255
updated erasure area pointers indicate that this data does not need to be preserved
Page n PAGE TO WHICH DATA IS WRITTEN
(Hex 0x1F = decimal 31)
Update "END" from 0x00 → 0x1F
(ERASURE AREA POINTER) Start 0x00 END 0x00
13b
(ERASURE AREA POINTER) Start 0x00 END 0x1F
(OBJECT BLOCK) PHYSICAL BLOCK 3
0X40000···PHYSICAL BLOCK 3
0X60000···PHYSICAL BLOCK 5
13a
Index
FIG. 7

Suda discusses this update numerous times. *E.g.*, Ex. 1003, Fig. 7, Fig. 10, Fig. 8 ("update erasure area pointer" in S4). The update occurs in response to the erase command. *Id.*, Fig. 8 (S1, S4), 7:30-55. Updating these erasure area pointers indicates that data is in a "virtual erased state" and "to be erased" later. *Id.*, 5:19-27, 5:40-48, 5:57-61, 6:9-14, 6:29-45, 6:60-64, 7:38-41. A POSITA would have understood that virtually erased data that is "to be erased" means that the data "does not need to be preserved on the flash storage medium," as claimed. Baker, ¶ 260.

Additionally, or alternatively, Suda teaches to cancel/erase a relationship between the designated logical block address and the corresponding physical address in response to the erase command. *Id.*, 5:62-6:3; 6:33-41. This cancelation/erasing in the logical to physical address table indicates that a physical block is "to be erased" and "set in an unused state," also indicating that the block will not be preserved. *Id.* Thus, a POSITA would have understood that Suda teaches claim 27[f] for this alternative reason as well. Baker, ¶¶ 260-62.

h) Element 27[h].

As discussed for claim 27[g] above, Suda teaches both (a) to cancel/erase an entry from the logical and physical address table in the index (b) to add an entry to the erasure area pointer storage area 13b in the index as shown in annotated Figure 7. *See* § X.I.h, *supra*. Thus, a POSITA would have understood Suda to teach claim 27[h] for either of these reasons. Baker, ¶¶ 263-66.

J. Claim 28

A POSITA would have understood that Suda taught the claimed "storage recovery module" because Suda taught a hardware circuit and/or programmable hardware and/or software implemented within a storage controller in Suda's flash memory controlling section. Baker, ¶ 267; Ex. 1003, 3:13-15.

As explained for claim elements 27[g]-[h], Suda teaches to cancel/erase a relationship between the designated logical block address and the corresponding physical address in response to the erase command, and this will cause the block "to be erased" and "set in an unused state." *Id.*, 5:62-6:3. When Suda refers to setting the block in an unused state, this refers to recovering or "erasing" the block so that the block "can be used" (e.g., written to) again. *Id.*, 3:64-67, 1:44-55 (recognizing that otherwise, "flash memory cannot be overwritten"). A POSITA would have understood this process to be the claimed erasing of a storage division (a block) comprising the physical storage location in response to removing the entry, as recited in the claim. Baker, ¶ 268.

K. Claim 29

a) Element 29[a].

Suda Fig. 1 explicitly shows that the indication from host device 2 is a "COMMAND." *E.g.*, Ex. 1003, 1:65-67, 3:4-11, 5:43, 6:18-25, 6:60-64, 7:11-18, 7:30, 7:40 (teaching "command" or "erase command"); Baker, ¶ 271.

b) Element 29[b].

Suda teaches the request receiver module in the form of flash memory controlling section, as discussed for claim 27[e]. Ex. 1003, Fig. 1; *see* § X.I.e, *supra*. Suda's memory device also includes a host interface section through which the flash memory controlling section receives commands. Ex. 1003, Fig. 1, 3:7-12. The commands received designate a logical block address, such as 0x40000. *Id.*, 6:66-9:3. Thus, because the host interface section handles commands with block addresses, a POSITA would have understood the host interface section to be the claimed "block device interface" and that the host interface section also uses a block storage protocol. Baker, ¶ 272.

c) Element 29[c].

For the reasons discussed for claim 26[b], Suda's system includes "a storage recovery module configured to recover the physical storage location at the physical address." *See* § X.H.b, *supra*. As explained, Suda's system erases the contents of blocks in a storage recovery process whenever the erasure area pointers indicate a block contains entirely deleted or virtually erased data. *See id.* A POSITA would have understood this erasing to be the recovery of physical storage locations comprising invalid data, as claimed. Baker, ¶ 274. A POSITA would also have understood that the recovered blocks comprised "invalid data," as claimed, because those blocks were filled with "virtual erased" data that cannot be read. Ex. 1003, 5:20-26, 5:54-61, 6:34-41, Figs. 4, 6, 9; Baker, ¶ 274; *see also* §§ VIII.B, X.A.e, X.I.g (explaining virtual erasure).

L. Claim 30

a) Elements 30[a]-30[f].

A POSITA would have found these elements obvious in view of Suda for the same reasons discussed with respect to the identically worded elements in claim 27. Baker, ¶¶ 276-81.

b) Element 30[g].

Suda discusses "the procedure for reading data" with respect to Fig. 9. Ex. 1003, 8:21-41. "[T]he flash memory controlling section 11 … determines whether a page range in which data items to be read are written is included in the area indicated by the erasure area pointer (step A2)." *Id.*, 8:30-35. If so, "it outputs an initial-value data as data to be read," in step A3 of Fig. 9. *Id.*, 8:35-38. "Initial-value" data refers the initial, empty state of data in a memory device. Baker, ¶ 282. This process for returning empty data may occur while the pages of a block are marked using the erasure area pointer 13b but still contain data, as indicated by shading in Fig. 7. *Id.*, Fig. 7 (annotated in § VIII.B, *supra*). A POSITA would have understood that such events would have been executed by at least Suda's flash memory controlling section 11, which would be the claimed "module" (circuitry and/or software/firmware). Thus, a POSITA would have understood Suda to teach this element. Baker, ¶¶ 282-83.

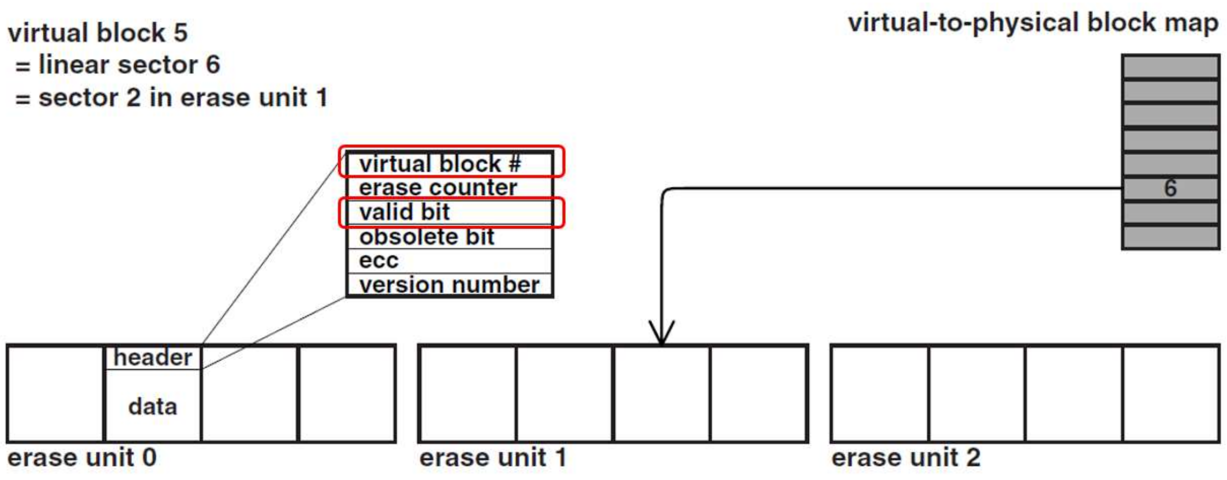
XI. Ground 3: Obvious Over Suda, SwSTE'05, and POSITA Knowledge

A. Claim 21[3]

Claim 21 depends on claim 15, which a POSITA would have found obvious over Suda as discussed under ground 2. *See* § X.A, *supra*. To the extent that UTL argues that Suda's erasure area pointers do not mark data invalid, a POSITA would still have found claim 21 obvious based on Suda and SwSTE'05.

SwSTE'05 explicitly teaches a "valid bit" associated with each block. Ex. 1010, Fig. 1 (annotated below). A POSITA would have understood that this valid bit indicates that the data packets stored in the block are invalid. Baker, ¶ 234.

[3] *See* attached Claim Listing.



A POSITA would have found it obvious for blocks in Suda's system to include these valid bits in either headers of blocks or the index. Baker, ¶¶ 234-35. Suda's system would set these bits as invalid once a block is designated for erasure, in addition to or instead of using erasure area pointers. *Id.*, ¶ 235; Ex. 1003, 8:66-9:3. This would have been an obvious modification because both headers and valid bits were standard techniques, and these bits provide useful information for memory systems. Baker, ¶ 235. A POSITA would have had a reasonable expectation of successfully using a small fraction of Suda's existing memory, whether RAM and/or flash memory, to store the valid bit associated with each logical block address. *Id.*, ¶ 235. These options are illustrated below. Thus, based on SwSTE'05, a POSITA would have found it obvious for Suda's system to use a valid bit to mark a data packet at the physical address invalid, as claimed. *Id.*, ¶¶ 233-36.

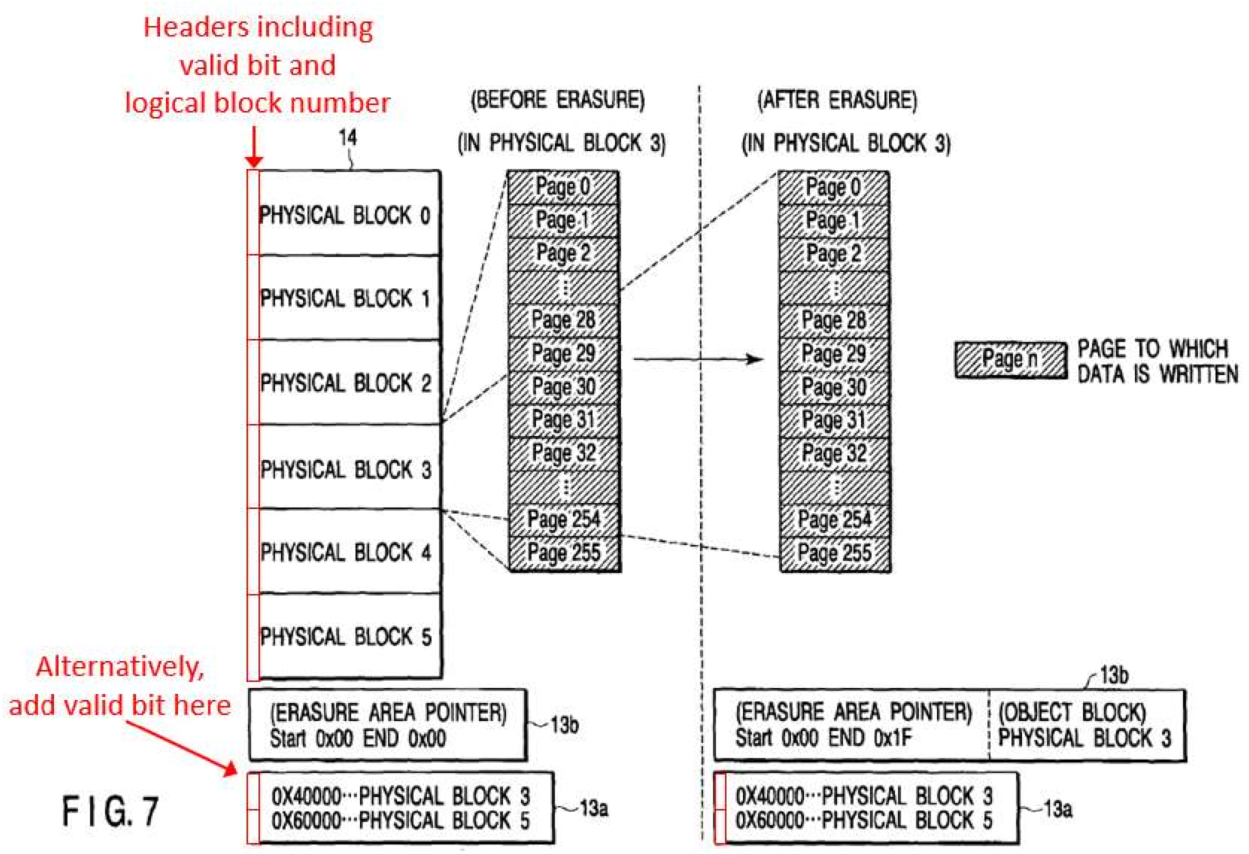

Headers including valid bit and logical block number
14
(BEFORE ERASURE)
(IN PHYSICAL BLOCK 3)
(AFTER ERASURE)
(IN PHYSICAL BLOCK 3)
PHYSICAL BLOCK 0
PHYSICAL BLOCK 1
PHYSICAL BLOCK 2
PHYSICAL BLOCK 3
PHYSICAL BLOCK 4
PHYSICAL BLOCK 5
Page 0
Page 1
Page 2
Page 28
Page 29
Page 30
Page 31
Page 32
Page 254
Page 255
Page n
PAGE TO WHICH DATA IS WRITTEN
Alternatively, add valid bit here
(ERASURE AREA POINTER) Start 0x00 END 0x00
13b
(ERASURE AREA POINTER) Start 0x00 END 0x1F
(OBJECT BLOCK) PHYSICAL BLOCK 3
0X40000···PHYSICAL BLOCK 3
0X60000···PHYSICAL BLOCK 5
13a

FIG. 7

Ex. 1003, Fig. 7 (annotated).

B. Claim 26

a) Elements 26[a], [c].

A POSITA would have found Elements 26[a] and [c] obvious over Suda as discussed under ground 2. *See* § X.H.a, c, *supra*.

b) Element 26[b].

Claim 26[b] does not recite "garbage collection," a technique that UTL contends infringes this element. Ex. 1013, 5. To the extent that UTL contends "garbage collection" is required by element 26[b], this would have been obvious to a POSITA for two reasons. Baker, ¶¶ 239-42.

First, a POSITA would have understood that Suda's system operates as a "garbage collection" process to reclaim erase units (Suda's blocks) as expressly taught by SwSTE'05. Baker, ¶ 240; Ex. 1010 § 2.3. Suda's steps leading to "erasure processing" in step S6 of Fig. 8 implement a form of garbage collection. Baker, ¶ 240. Thus, a POSITA would have understood Suda's process to be a "garbage collection" process as accused by the Patent Owner.

Alternatively, SwSTE'05 explicitly teaches a garbage collection process in its bulleted steps in Section 2.3. Ex. 1010. These steps include selecting an erase unit (a block), copying valid sectors (pages) out of the block, updating a logical to sector (logical to physical) mapping, and physically erasing the erase unit. *Id.*, § 2.3. This last step of physically erasing the erase unit will "recover the physical storage location," as claimed. Baker, ¶ 241; Ex. 1010 ("the reclaimed erase units are erased"). A POSITA would have found it obvious to apply the bulleted steps disclosed in SwSTE'05 to Suda's system, recognizing that the "erase units" of SwSTE'05 are Suda's "blocks," and that the "data structures that map logical blocks to sectors" of SwSTE'05 are Suda's tables 13a and 13b. Ex. 1010, § 2.3; Baker, ¶ 241. A POSITA would have had a reasonable expectation of success because garbage collection was well-known and had been a standard part of flash memory management since the mid-1990s. Baker, ¶ 241.

Thus, a POSITA would have understood that SwSTE'05 teaches garbage collection or that it would have been obvious to modify Suda to include it. *Id.*, ¶ 241-42.

C. Claim 28

Claim 28 does not recite, "garbage collection," a technique that ULT contends infringes this element. Ex. 1013, 6. To the extent that UTL contends "garbage collection" is required by claim 28, this would have been obvious to a POSITA in view of Suda and SwSTE'05 for the two reasons discussed above at element 26[b]. Baker, ¶¶ 269-70; *see* § IX.B.b, *supra*.

First, a POSITA would have recognized that the steps of Suda's process leading to "erasure processing" in step S6 of Fig. 8 implement a form of garbage collection. Baker, ¶ 268.

Alternatively, a POSITA would have applied SwSTE'05 garbage collection process in its bulleted steps in Section 2.3 to Suda's system. Ex. 1010. These steps end with physically erasing a storage division, as claimed. *Id.*, § 2.3 ("Finally, the erase units are erased"). Thus, a POSITA would have understood that SwSTE'05 teaches garbage collection and that it would have been obvious to modify Suda to include it. Baker, ¶¶ 241, 269; *see* § IX.B.b, *supra*.

XII. Secondary considerations

Simultaneous invention by others shows that the claims fall within the level of the ordinary skill in the art. "Independently made, simultaneous inventions, made within a comparatively short space of time, are persuasive evidence that the claimed apparatus was the product only of ordinary mechanical or engineering skill." *Geo. M. Martin Co. v. All. Mach. Sys. Int'l LLC*, 618 F.3d 1294, 1305 (Fed. Cir. 2010). The Board has held that exhibits of a standard-setting group on a related standard "are evidence of simultaneous invention by others," support finding challenged claims obvious, and "are persuasive evidence that the claimed apparatus 'was the product only of ordinary mechanical or engineering skill.'" *ZTE (USA) Inc. v. Evolved Wireless LLC*, No. IPR2016-00757, Paper 42, at 29 (P.T.A.B. Nov. 30, 2017).

Here, exhibits 1017-1019 show that standard-setting group T13 began work on the Trim command proposal at least by April 21, 2007, only four months from the earliest possible (disputed) priority date. Baker, ¶ 91. UTL accuses this Trim command of infringing the claims. Ex. 1013, *passim*. Like the *ZTE* case, here a standard-setting group worked on the same technology around the same time. Exs. 1017-1019. Also, Suda and Bennett teach similar commands. Ex. 1002, 17:52-56 (an erase command "specifies the (logical) sectors to be erased"); Ex. 1003, 9:2-3 ("logical block address ... designated in the erase command"). Furthermore, many claim elements were already well-known in the art. *See, e.g.,* Ex. 1010 § 2.2 (Ban patented the FTL in 1995, and the FTL became part of an industry standard), § 2.3 (explaining the garbage collection process). Thus, Exhibits 1002-1003 and 1017-

1019 all serve as evidence of simultaneous invention by others, and the Board should find the challenged claims obvious for being only the product of ordinary mechanical or engineering skill.

XIII. The Parallel District Court Litigations Do Not Warrant Denying Institution

When considering a parallel proceeding, the PTAB "balances" considerations such as "system efficiency, fairness, and patent quality" using the six factors set forth by the Board in *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (P.T.A.B. Mar. 20, 2020) (precedential). These factors "overlap," and a "holistic view" should be taken. *Id.*, p. 6.

The fourth factor (overlap) strongly favors institution. Petitioners have stipulated that they will not pursue invalidity on the same grounds—or even the same references—if the Board institutes trial in this proceeding. Ex. 1028. Petitioners modeled this stipulation on the stipulation that the Board found to "mitigate any concerns" in *VMware, Inc. v. Intellectual Ventures I LLC*, IPR2020-00470, Paper 13 at 20 (P.T.A.B. August 18, 2020). This fourth factor favors institution here even more so than in *Apple, Inc. v. SEVEN Networks, LLC*, IPR2020-00156, Paper 10 (P.T.A.B. June 15, 2020). There, the petitioner provided no stipulation. *Id.* pp. 16-19. Nevertheless, the fourth factor "strongly favored" petitioner. *Id.*

The third factor (investment in parallel proceeding) also favors institution.

The District Court has not issued any substantive opinions regarding the scope or validity of the challenged claim, and given Petitioners' stipulations, the Court is unlikely to invest any resources on the grounds raised in this petition, either before or after the scheduled institution date. Furthermore, the parallel proceeding is in an early stage: the Court is deciding Petitioners' motions to dismiss, the Petitioners have not otherwise answered, fact discovery is not open and only initial contentions have been exchanged. Ex. 1016, 1029.

Regarding the sixth factor (merits, other circumstances), the merits strongly weigh in favor of instituting trial as shown through the strength of the grounds in this petition. Other circumstances also favor institution. Like in *Apple v. SEVEN*, the parallel litigations are complex, involving 3 patents, 34 asserted claims and hundreds of accused products, and the District Court requires reduction of claims pre-trial. *Apple v. SEVEN*, 21-22; Ex. 1031, 10 (weighing claim reductions). An IPR trial, in contrast, allows a focus on resolving all challenged claims in a single patent, thus "enhanc[ing] the integrity of the patent system." *Apple v. SEVEN* at 22.

Fintiv factors 1 (stay) and 2 (proximity of trial dates) do not significantly weigh for or against instituting IPR. Petitioners do not know if the District Court will stay the case if trial is instituted and the Court has not yet set the trial date. The District Court estimated a February 2022 date, but ordered the parties to only file a proposed schedule up to the *Markman* hearing stage. Ex. 1029; *compare Micron*

Tech., Inc. v. Godo Kaisha IP Bridge 1, IPR2020-01007, Paper 15 at 10-13 (P.T.A.B. December 7, 2020) (the "proximity factor in *Fintiv*, on its face, asks us to evaluate our discretion in light of trial dates that have been set in parallel litigations, not to speculate as to trial dates that are still to-be-determined"). Moreover, the estimated trial date is uncertain given that District Court has set about 99 cases for trial between now and February 2022. This equates to an average of about 1-2 trials per week. In addition, 28 cases have been set for trial for the eight-week period between January to February 2022.[4] *See Globalfoundries Inc. v. UNM Rainforest Innovations*, IPR2020-00984, Paper 11 at concurrence 3 (P.T.A.B. Dec. 9, 2020) (weighing large number of trials and proportion of reschedule trials and noting "as the period of time remaining before trial increases, the certainty that the trial date will remain unchanged decreases").[5]

XIV. Mandatory Notices

A. <u>Real Parties-in-Interest</u>

[4] Petitioners reviewed trial date data from Bloomberg Law Dockets.

[5] In many courts, more than 50% of cases have their initial trial date continued. Ex. 1036, 64 & Table 25 (average delay is three to six months); *see also Precision Planting LLC v. Deere & Co*., IPR2019-01048, Paper 17 at 15–19 (P.T.A.B. Dec. 4, 2019) (courts modify deadlines "for myriad reasons").

The named Petitioners are the only entities who are funding and controlling this Petition and are therefore all named as real parties-in-interest. No other entity is funding, controlling, or otherwise has an opportunity to control or direct this Petition or Petitioner's participation in any resulting IPR.

Out of an abundance of caution, Petitioners also identify Denali Intermediate Inc., which is a corporate parent entity of Dell Inc., as a real party-in-interest. Petitioners also identify that there are many entities such as suppliers, resellers, part providers, contractors, etc. who may have financial liabilities with respect to the hundreds of accused products in the related litigations. Petitioners do not believe that any of these entities, however, are real parties-in-interest. None of these other entities participated in the preparation or funding of this Petition or otherwise had an opportunity to control or direct this Petition. To Petitioners' best knowledge, no entity, other than Petitioners, has been served with a complaint alleging infringement of the patent at issue herein.

B. Related Proceedings

In three related lawsuits, UTL asserted the '406 patent against Petitioners in the Western District of Texas, Case Nos. 6:20-cv-499, -500, and -501. UTL filed each lawsuit on June 5, 2020.

C. Lead and Backup Counsel

The following lead and backup counsel represent Petitioners:

Lead Counsel for Petitioner	Backup Counsel for Petitioner
Katherine A. Vidal Winston & Strawn LLP 275 Middlefield Rd., Suite 205 Menlo Park, CA 94025 kvidal@winston.com T: 650.858.6500, F: 650.858.6550 USPTO Reg. No. 46,333	Michael Rueckheim Winston & Strawn LLP 275 Middlefield Rd., Suite 205 Menlo Park, CA 94025 mrueckheim@winston.com T: 650.858.6500, F: 650.858.6550 (to seek *pro hac vice* admission) *************** Qi (Peter) Tong Winston & Strawn LLP 2121 N Pearl St. Dallas, TX 75201 ptong@winston.com T: 214.453.6473, F: 214.453.6400 USPTO Reg. No. 74,292

D. Electronic Service

Petitioners consent to electronic service at:

Winston-IPR-Unification@winston.com

XV. Fees

Petitioners have paid the required fee electronically through P.T.A.B. E2E.

XVI. Conclusion

Petitioners respectfully request that the Board institute IPR and enter a final written decision finding the challenged claims unpatentable.

Dated: December 22, 2020

Respectfully submitted,
/ Katherine A. Vidal /

Katherine A. Vidal
Winston & Strawn LLP
275 Middlefield Rd, Suite 205
Menlo Park, California 94025
kvidal@winston.com
T: 650.858.6500, F: 650.858.6550
USPTO Reg. No. 46,333
Lead Counsel for Petitioners Micron Technology, Inc.; Micron Semiconductor Products, Inc.; Micron Technology Texas LLC; HP Inc.; Dell Inc.; and Dell Technologies Inc.

Michael Rueckheim
Winston & Strawn LLP
275 Middlefield Rd, Suite 205
Menlo Park, California 94025
mrueckheim@winston.com
T: 650.858.6500, F: 650.858.6550
Back-up Counsel for Petitioners Micron Technology, Inc.; Micron Semiconductor Products, Inc.; Micron Technology Texas LLC HP; Inc.; Dell Inc.; and Dell Technologies Inc.
(to seek *pro hac vice* admission)

Qi (Peter) Tong
Winston & Strawn LLP
2121 N Pearl St,
Dallas, TX 75201
ptong@winston.com
T: 214.453.6473, F: 214.453.6400
USPTO Reg. No. 74,292
Back-up Counsel for Petitioners Micron Technology, Inc.; Micron Semiconductor Products, Inc.; Micron Technology Texas LLC; HP Inc.; Dell Inc.; and Dell Technologies Inc.

CLAIM LISTING

Claim 15	
Element	**Language**
15[a]	An apparatus, comprising:
15[b]	a non-volatile storage medium;
15[c]	a request receiver module of a storage layer for the non-volatile storage medium configured to receive an indication that a data structure, corresponding to data stored on the non-volatile storage medium, has been deleted,
15[d]	wherein the indication comprises a logical identifier that is associated with the data structure by a storage client, and wherein the logical identifier is mapped to a physical address of the data on the non-volatile storage medium; and
15[e]	a marking module configured to record that the data stored at the physical address mapped to the logical identifier can be erased from the non-volatile storage

	medium in response to receiving the indication.

Claim 16	
Element	**Language**
16	The apparatus of claim 15, wherein the marking module is configured to record that data stored at a physical address on the non-volatile storage medium can be erased from the non-volatile storage medium by invalidating an association between the logical identifier and the physical address.

Claim 17	
Element	**Language**
17	The apparatus of claim 15, wherein the marking module is configured to record that data stored at a physical address on the non-volatile storage medium can be erased from the non-volatile storage medium by deleting a mapping between the logical identifier and the physical address.

Claim 18	
Element	**Language**
18	The apparatus of claim 15, further comprising an index comprising mappings between logical identifiers and physical addresses on the non-volatile storage medium, wherein the marking module is configured to remove a mapping between the logical identifier and the physical addresses of the data from the index.

Claim 19	
Element	**Language**
19	The apparatus of claim 18, wherein the marking module is configured to delete a reference to the physical address from an index entry of the logical identifier.

Claim 20	
Element	**Language**
20	The apparatus of claim 18, wherein removal of the mapping indicates that data stored at the physical address

	can be erased from the non-volatile storage medium.

Claim 21	
Element	**Language**
21	The apparatus of claim 15, wherein the marking module is configured to mark a data packet at the physical address invalid.

Claim 26	
Element	**Language**
26[a]	The apparatus of claim 15, wherein the non-volatile storage medium comprises a flash storage medium, the apparatus further comprising:
26[b]	a storage recovery module configured to recover the physical storage location at the physical address; and
26[c]	a storage module configured to store data associated with another logical identifier on the physical storage location in response to recovering the physical storage location.

Claim 27	
Element	**Language**
27[a]	A system comprising:
27[b]	a flash storage medium; and
27[c]	a storage layer for the flash storage medium, comprising;
27[d]	an index module configured to maintain an index comprising a plurality of entries, wherein the individual entries associate a logical identifier with a physical storage location of the flash storage medium that comprises data associated with the logical identifier; and
27[e]	a request receiver module configured to receive an indication comprising a logical identifier that is empty,
27[f]	wherein the logical identifier is associated with a data identifier of a storage client, and wherein the logical identifier is associated with a physical storage location in the index,
27[g]	wherein the index module is configured to update the

	index, in response to the indication, to indicate that contents of the physical storage location associated with the logical identifier do not need to be preserved on the flash storage medium,
27[h]	wherein the index module is configured to update the index by one or more of (a) removing an entry from the index that associates the specified logical identifier with the physical storage location, and (b) adding an entry to the index.

Claim 28	
Element	**Language**
28	The system of claim 27, further comprising a storage recovery module configured to erase a storage division comprising the physical storage location in response to removing the entry.

Claim 29	
Element	**Language**

29[a]	The system of claim 27, wherein the indication comprises a command,
29[b]	wherein the request receiver module is configured to receive the command through one of a block device interface and via a block storage protocol,
29[c]	and wherein the system further comprises a storage division recovery module configured to recover physical storage locations comprising invalid data.

Claim 30	
Element	**Language**
30[a]	A system, comprising:
30[b]	a flash storage medium;
30[c]	an index module configured to maintain an index comprising a plurality of entries, wherein the individual entries associate a logical identifier with a physical storage location of the flash storage medium that

	comprises data associated with the logical identifier;
30[d]	a request receiver module configured to receive an indication that a specified logical identifier is empty, wherein the logical identifier is associated with a data identifier of a storage client, and wherein the logical identifier is associated with a physical storage location in the index,
30[e]	wherein the index module is configured to update the index, in response to the indication, to indicate that contents of the physical storage location associated with the logical identifier do not need to be preserved on the flash storage medium,
30[f]	wherein the index module is configured to update the index by one or more of (a) removing an entry from the index that associates the specified logical identifier with the physical storage location, and (b) adding an entry to the index; and
30[g]	a read request response module configured to return an

	indication that the logical identifier is empty in response to a request to read data of the logical identifier while the data remains on the flash storage medium.

CERTIFICATE OF COMPLIANCE

This Petition complies with the word count limits set forth in 37 C.F.R. ¶ § 42.24(a)(1)(i), because this Petition contains 12,484 words, excluding the parts of the Petition exempted by 37 C.F.R. § 42.24(a)(1) and determined using the word count provided by Microsoft Word, which counsel used to prepare this Petition.

Dated: December 22, 2020

Respectfully submitted,
/ *Katherine A. Vidal* /
Katherine A. Vidal
kvidal@winston.com
USPTO Reg. No. 46,333
Winston & Strawn LLP
275 Middlefield Rd, Suite 205
Menlo Park, California 94025
T: 650.858.6500, F: 650.858.6550

CERTIFICATE OF SERVICE

Under 37 C.F.R. §§ 42.6(e) and 42.105(a), this is to certify that on December 22, 2020, I caused to be served a true and correct copy of the foregoing "**PETITION FOR *INTER PARTES* REVIEW OF CLAIMS 15-21 AND 26-30 OF U.S. PATENT NO. 8,533,406**" and Exhibits 1001 – 1036 by FedEx on the Patent Owner at the correspondence address of record for U.S. Patent No. 8,533,406:

Western Digital, c/o Longitude Licensing Stoel Rives LLP
201 South Main Street, Suite 1100
One Utah Center
Salt Lake City UT 84111

A courtesy copy of this Petition and supporting material was also served on litigation counsel for Patent Owner via email:

Barry J. Bumgardner
Nelson Bumgardner Albritton PC
3131 W. 7th Street, Suite 300
Fort Worth, TX 76107
Email: barry@nbafirm.com

WINSTON & STRAWN LLP

/ Katherine A. Vidal /
Katherine A. Vidal
Winston & Strawn LLP
275 Middlefield Rd, Suite 205
Menlo Park, California 94025
kvidal@winston.com
T: 650.858.6500, F: 650.858.6550
USPTO Reg. No. 46,333