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. IPR2021-00344

U.S. Patent No. 8,762,658

PETITION FOR *INTER PARTES* REVIEW OF CLAIMS 1-5, 8-12, 22-26 OF U.S. PATENT NO. 8,762,658

TABLE OF CONTENTS

Page

I. Introduction....1
II. Petitioners Meet Standing and Eligibility Requirements for *Inter Partes* Review....2
III. Prosecution History of the '658 Patent....2
IV. Background....3
V. Summary of the '658 Patent....4
A. Effective Filing Date and Date of Invention....4
B. Level of Ordinary Skill in the Art....4
VI. Claim Construction....5
VII. Precise Relief Requested....8
A. Proposed Grounds....8
B. Qualifying Prior Art....9
C. The Proposed Grounds Are Not Cumulative or Redundant....9
VIII. The Prior Art....10
A. Summary of Bennett....10
B. Summary of Suda....14
C. Summary of SwSTE'05....16
D. Motivation to Combine Suda and SwSTE'05....17
IX. Ground 1: Obvious Over Bennett and POSITA Knowledge....18
A. Claim 1....18
B. Claim 2....25
C. Claim 3....26
D. Claim 4....27
E. Claim 5....27
F. Claim 8....28
G. Claim 9....28
H. Claim 10....29
I. Claim 11....29

J. Claim 12 30
K. Claim 22 31
L. Claim 23 32
M. Claim 24 32
N. Claim 25 33
O. Claim 26 34
X. Ground 2: Obvious Over Suda and POSITA Knowledge 34
A. Claim 1 34
B. Claim 2 40
C. Claim 3 42
D. Claim 4 43
E. Claim 5 43
F. Claim 8 44
G. Claim 9 45
H. Claim 10 46
I. Claim 11 47
J. Claim 12 48
K. Claim 22 48
L. Claim 23 50
M. Claim 24 51
N. Claim 25 52
O. Claim 26 53
XI. Ground 3: Obvious Over Suda, SWSTE'05, and POSITA Knowledge 53
A. Claim 2 53
B. Claim 3 55
C. Claim 4 55
D. Claim 5 55
E. Claim 10 56
F. Claim 11 58
G. Claim 12 59

H. Claim 23 59
I. Claim 25 60
XII. Secondary Considerations 61
XIII. The Parallel District Court Litigations Do Not Warrant Denying Institution 62
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
CLAIM LISTING 70
CERTIFICATE OF COMPLIANCE 79
CERTIFICATE OF SERVICE 80

TABLE OF AUTHORITIES

Page(s)

Cases

Apple Inc. v. Fintiv, Inc.,
IPR2020-00019, Paper 11 (P.T.A.B. Mar. 20, 2020) 62, 64

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) 61

Globalfoundries Inc. v. UNM Rainforest Innovations,
IPR2020-00984, Paper 11 (P.T.A.B. Dec. 9, 2020) 64

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) 62

ZTE (USA) Inc. v. Evolved Wireless LLC,
No. IPR2016-00757, Paper 42 (P.T.A.B. Nov. 30, 2017).....61

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. 8,762,658 B2, titled "SYSTEMS AND METHODS FOR PERSISTENT DEALLOCATION" 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. 8,762,658.
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: 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 A to Plaintiff's First Amended Infringement Contentions: Unification Technologies' Allegations of Infringement with Respect to U.S. Patent No. 8,762,658, *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 (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 DELETION IN NON-VOLATILE MEMORY" to Saltz et al.
1021	Public File History of U.S. Pat. No. 8,762,658 B2, titled "SYSTEMS AND METHODS FOR PERSISTENT DEALLOCATION" 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. 8,762,658.
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).
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 8,762,658 (the "'658 patent") should never have issued. For example, claim 1 generally recites a non-volatile storage device that (i) receives a message indicating that data associated with a logical identifier has been erased; (ii) maps the logical identifier to a physical address space, and (iii) stores an indication the data is erased. In the related litigations the Patent Owner, Unification Technologies LLC ("UTL") generally asserts the claims encompass receiving a message indicating "data has been erased" from a user's perspective, e.g., by a computer attached to storage, even though data remains on the storage device. *See* § VI, *infra*. 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 '658 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 '658 Patent

The '658 patent application was filed on August 3, 2012. Ex. 1001, cover.

The Examiner rejected the claims on various grounds, but did not cite the references relied upon herein. *Id.*, pp. 1-5 (listing cited references); Ex. 1021, 169-83. To overcome the rejections, the independent claims were amended to recite that the received message/hint/indication comprises "a logical identifier." Ex. 1021, 932-41. The Examiner allowed the amended claims, stating, "the art of record fails to teach or suggest receiving a message/hint/identifier comprising a logical identifier that indicates data associated with the logical identifier has been erase[d] and also storing data on a storage medium to indicate data associated with the logical identifier has been erased." *Id.*, 1263.

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.*, ¶ 73. Erase commands for flash memory were well known and standardized before the earliest provisional for the '658 patent. Ex. 1005, §§ 6.16, 8.1.

Flash memory uses an FTL to map logical addresses to physical addresses. Baker, ¶ 80. 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.*, ¶ 83.

V. Summary of the '658 Patent

The '658 patent acknowledges that erase commands for file systems were known. *See, e.g.*, Ex. 1001, 1:29-32 ("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.*, 1:33-36. 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.*, 44:44-47.

A. Effective Filing Date and Date of Invention

The '658 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 '658 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, ¶ 56. 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, ¶¶ 57, 61.

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"	1-5, 8-10,	"an identifier	"information that

Claim Term	Claim Nos.	Petitioners	UTL
	22-26	maintained by a computer attached to the storage medium used to identify the logical location of data stored on the storage medium"	identifies a particular set of data that is not the physical address of the data"
"data associated with the logical identifier [has been/is] erased"	1	Indefinite	Plain and ordinary meaning
"storage module" "marking module"	1 9-10	Indefinite under 112(f) with no corresponding structure	Only [module] needs to be construed. "Module" should be construed as "a hardware circuit and/or programmable hardware and/or software implemented within a storage controller"
"persistent data"	1-4, 22, 25	Plain and ordinary meaning	"data that is retained in the absence of power, such as data stored in a non-volatile storage medium like NAND flash memory"
"instructions configured to … recording persistent data … in response the indication"	22	Indefinite under 112(f) with no corresponding structure	Not indefinite and not subject to 112(f)
"logical identifier [in the index] is	22, 25-26	Indefinite	"data identified by the [logical

Claim Term	Claim Nos.	Petitioners	UTL
empty"			identifier] in the index does not need to be preserved"

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.").

Although not offering a construction other than "plain and ordinary meaning,"

UTL appears to contend that "data associated with the logical identifier [has been/is] erased" refers to "data associated with the logical identifier has been erased by a device connected to the [accused] Product (e.g., by a computer)" and, as a result, "[f]rom the user's perspective, this data has been deleted from a document." Ex. 1013, 2. Further, UTL contends that this term does ***not*** refer to data on a non-volatile device being already physically erased. *See, e.g.*, *id.* ("The TRIM command tells the SSD that specific areas contain data that is no longer in use").

VII. Precise Relief Requested

A. Proposed Grounds

a) Ground 1

Claims 1-5, 8-12, and 22-26 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 1-5, 8-12, and 22-26 are rendered obvious by Suda (Ex. 1003) in view of a POSITA's knowledge.

c) Ground 3

Claims 2-5, 10-12, 23 and 25 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 '658 patent. Petitioners are unaware of any assertion that the '658 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, ¶¶ 59-60.

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: (i) "a message/hint/identifier comprising a logical identifier that indicates data associated with the logical

identifier has been erase[d]"; and (ii) "storing data on a storage medium to indicate data associated with the logical identifier has been erased." *See* § III*, supra*. As shown herein: Bennett and Suda do.

VIII. The Prior Art

A. Summary of Bennett

Like the '658 patent, Bennett recognizes that flash-memory erase operations take a (relatively) long time. *Compare* Ex. 1001, 41:35-36 (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 is "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, 51:33-35 ("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, … S_{N-1} that are erasable together." *See id.*, 7:14-20, Figs. 3A(i)-3A(ii); Baker, ¶ 98. The GAT is stored in non-volatile flash memory as highlighted in Bennett's Fig. 6 below:

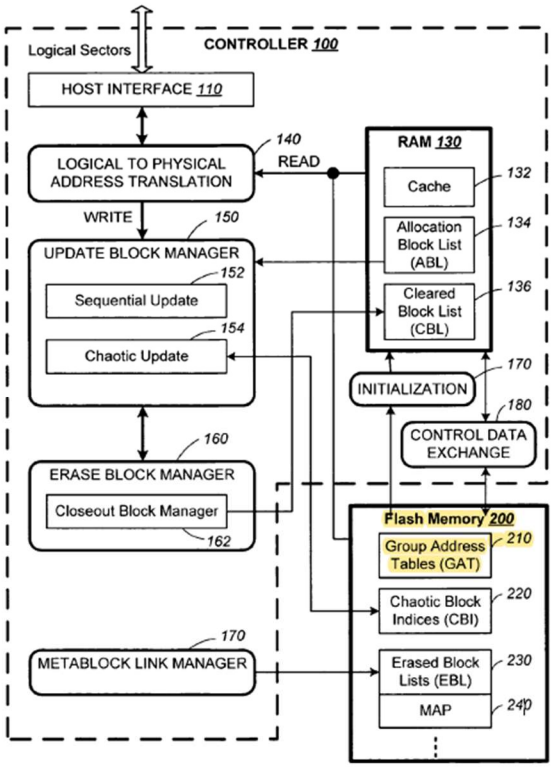


FIG. 6

Id., 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:

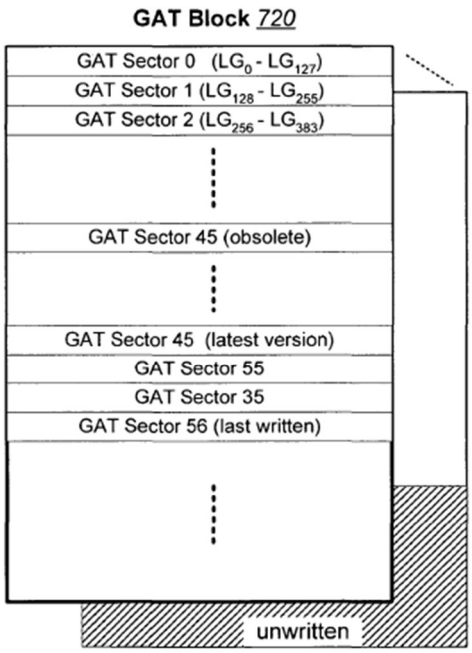


FIG. 8B

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, ¶ 73. 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.

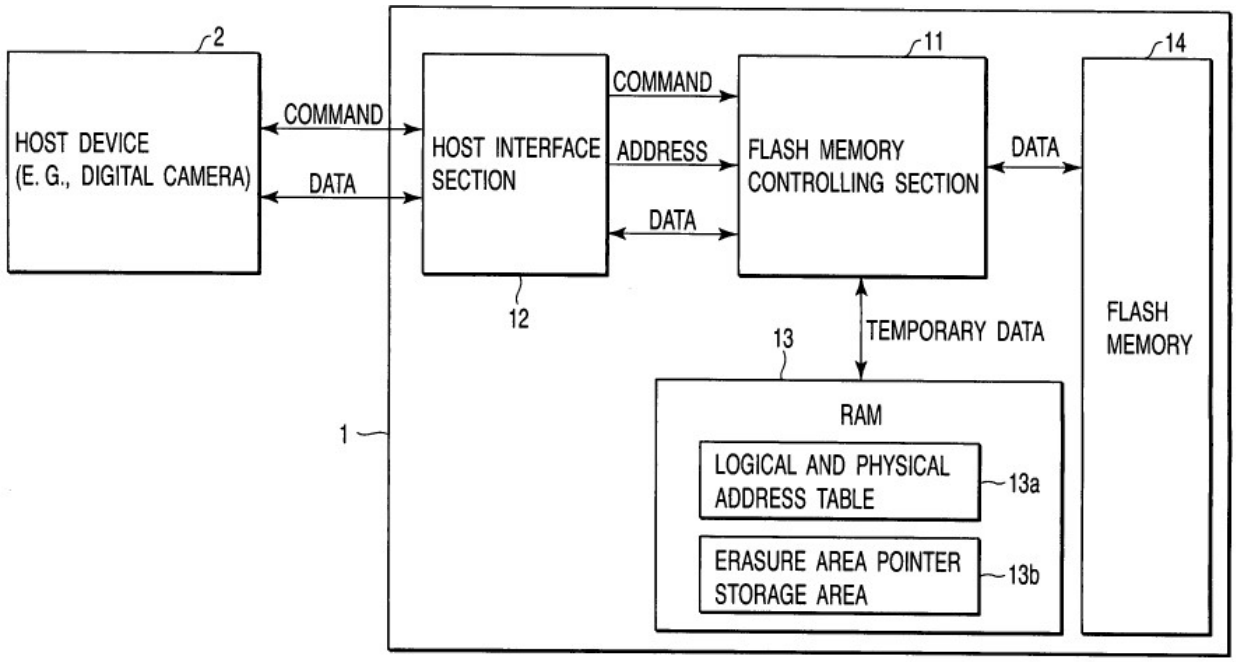


FIG. 1

Id., Fig. 1.

Like the '658 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).

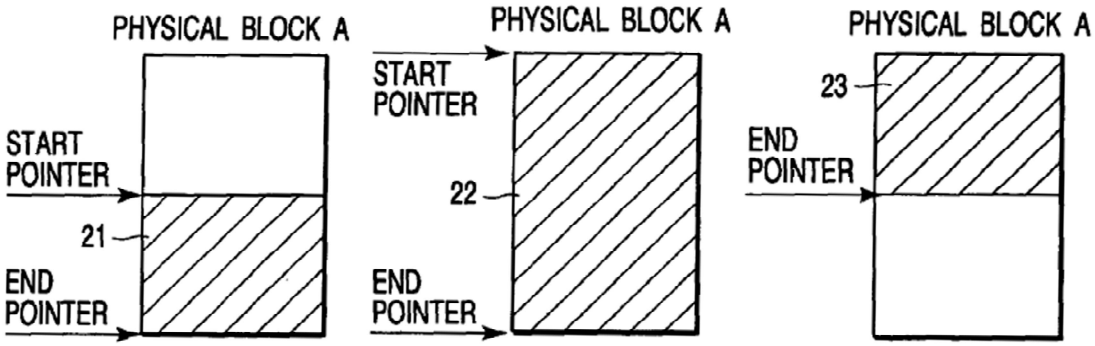


FIG. 3 FIG. 4 FIG. 5

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.

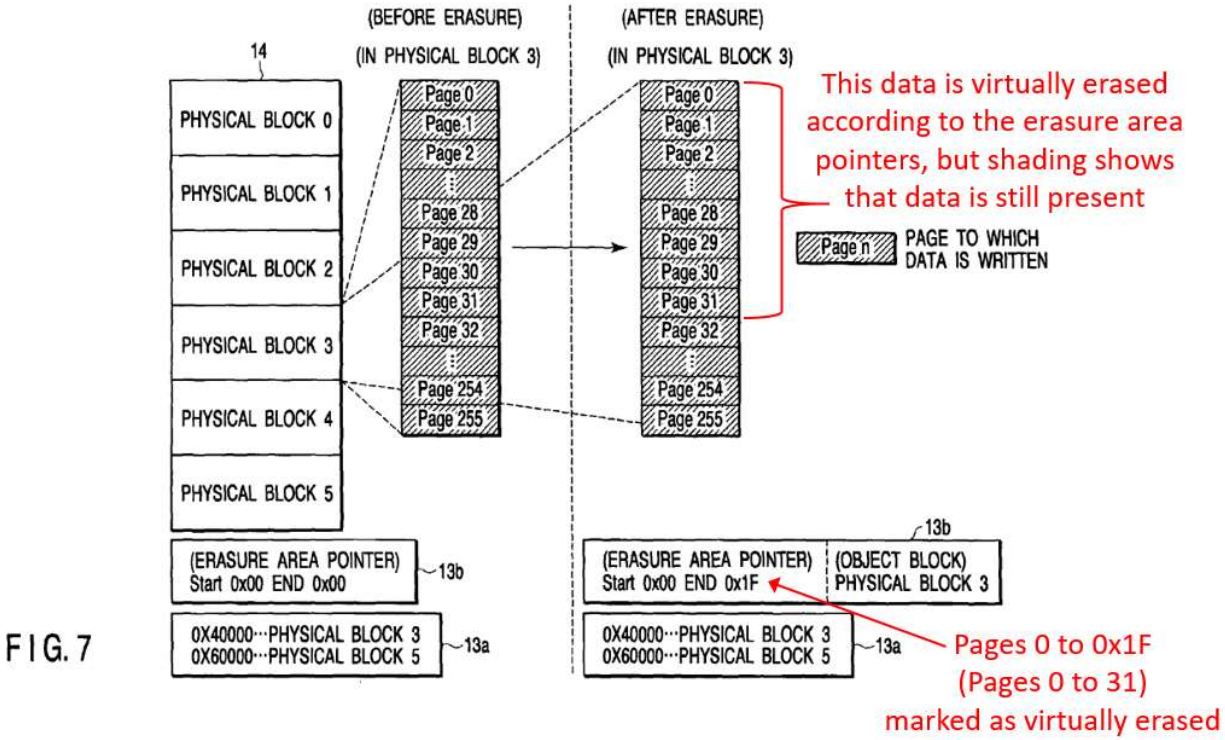


FIG. 7

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. Three teachings of SwSTE'05 are relevant here.

First, SwSTE'05 explains that memory devices used an 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, ¶ 118 (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 storing the inverse maps and direct maps in a two-level caching scheme. Ex. 1010, § 2.2. "Inverse maps are stored on the flash device itself." *Id.* Whereas, "[d]irect maps are stored at least partially in RAM" to allow for "fast" and "quick" lookups. *Id.* The direct map in RAM "is reconstructed during device initialization." *Id.*

Third, 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, ¶¶ 221, 245-46. 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*. 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 1

a) Element 1[a][1]

Bennett is titled "Methods For The Management Of Erase Operations In Non-Volatile Memory." Ex. 1002, Title. Further, Bennett teaches memory "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.

[1] *See* attached Claim Listing.

b) Element 1[b]

Bennett discloses "Flash Memory 200":

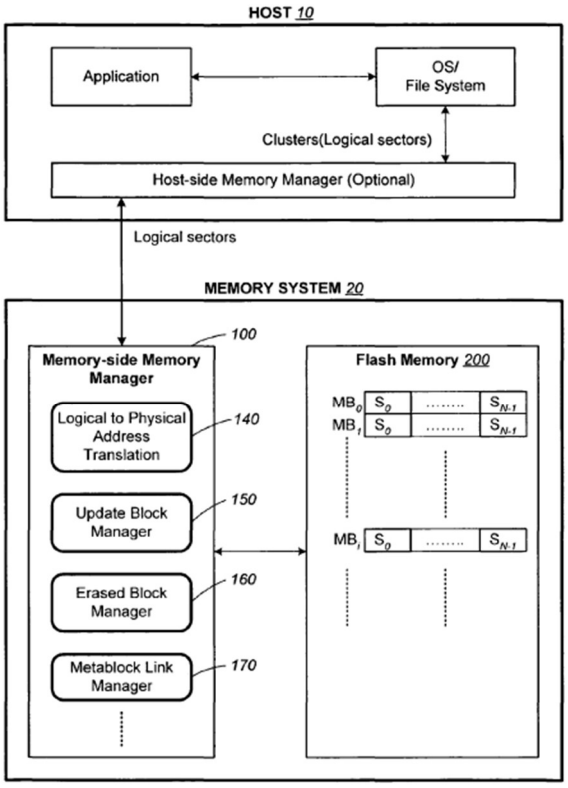


FIG. 2

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 1[c]

Bennett teaches the recited "request receiver module" in the form of a host interface that receives commands from a host. Ex. 1002, 5:56-58. Figs. 1 and 6 depict the host interface 110:

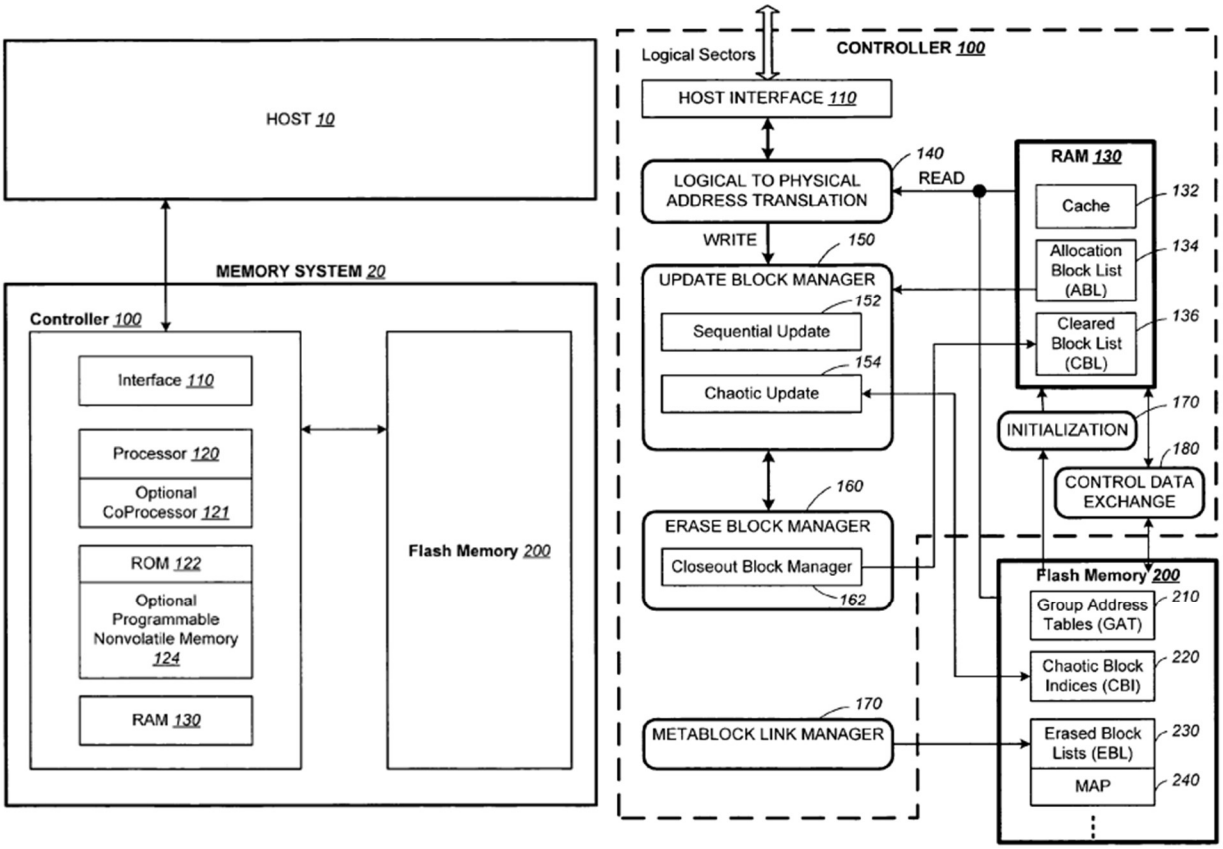


FIG. 1 FIG. 6

Ex. 1002, Figs. 1 and 6; *see also* 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 further teaches receiving a message comprising a logical identifier. 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). 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). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 125-29.

d) Element 1[d]

UTL contends "data associated with the logical identifier [has been/is] erased" refers to data that appears deleted from a user's perspective, for example by a computer attached to a storage device. *See* § VI, *supra*. As a result, "[f]rom the user's perspective, this data has been deleted from a document," even though it remains on the storage device. *Id.*

Bennett teaches the same thing. Bennett discloses "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, 17:12-15; *see also id.*, 4:22 ("a host

issues an Erase Sectors command"), Fig. 1 (showing host computer 10). The command is sent as part of the host "running an application under a file system or operating system." *Id.*, 7:21-22. For example, a POSITA would have understood that the erase command could be sent when a user deletes a document associated with a word processing application. Baker, ¶ 132. Thus, from the perspective of a user, the designated data has been deleted, and a POSITA would have understood that these erase commands indicate that data has been erased by the host device, as claimed. Baker, ¶¶ 130-32.

e) Element 1[e]

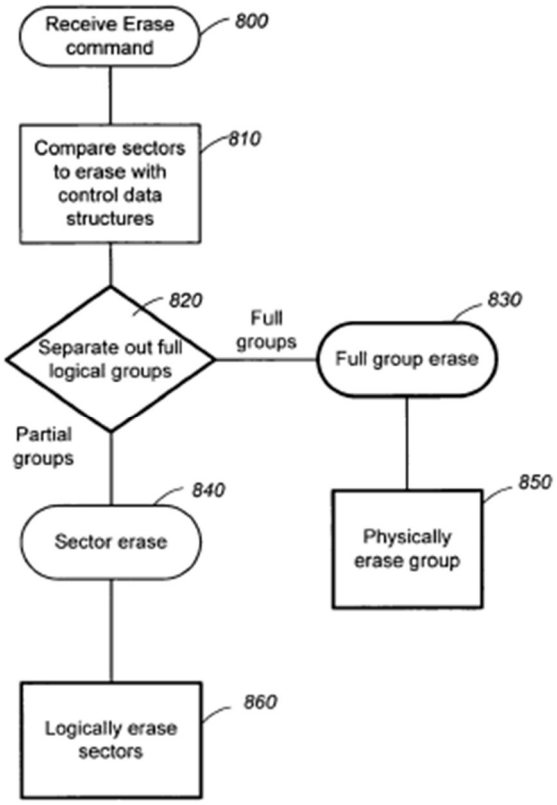
Bennett teaches a "logical to physical address translation module [that] maps the logical address from the host to a physical memory location." *Id.*, Fig. 6, 9:50-52, 10:36-38. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶ 133.

f) Element 1[f]

Bennett teaches "memory-side memory manager" with a "logical to physical address translation" module for "responsible for relating a host's logical address to a corresponding physical address in flash memory". Ex. 1002, 10:38-39. The logical-to-physical address mapping is stored in tables, such as a GAT, in flash memory ("non-volatile storage medium"). Ex. 1002, 1:29-30, Fig. 6, 10:65-11:8. The GAT is updated in response to receiving erase command messages to indicate

that the data associated with the logical identifier is erased (actually erased or logically erased). *Id.*, 5:60-61, 20:20-27, 20:45-47. A POSITA would understand that the GAT update is persistent data, because the GAT is stored in non-volatile memory. Baker, ¶ 139. The memory manager and GAT are a "storage module" under UTL's proposed construction of "module" because a POSITA would understand that Bennett implements this functionality through a hardware circuit and/or programmable hardware and/or software in a controller. *Id.*, ¶ 135.

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, erase commands specifying full groups are "physically

erased" (step 850); commands specifying partial groups "are logically erased" (step 860). *Id.*, 4:12-21.

For physical erasures, Bennett teaches storing persistent data in a variety of ways to indicate that the data associated with the logical identifier has been erased. For example, Bennett teaches marking physical sector headers with an "erased" flag in addition to writing FFs or 00s to the sector. *Id.*, 20:20-27. Bennett also teaches maintaining "a special record, maintained in the non-volatile memory, about the erase operation to be performed, where the record can be updated at the completion of the erase operation as a whole or as parts of it are completed." *Id.*, 18:28-32.

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

Thus, a POSITA would have recognized that Bennett teaches "a storage module configured to store persistent data on the non-volatile storage medium in response to the indication" for both full group actual erasures and partial group logical erasures. Baker, ¶¶ 134-40.

B. Claim 2

a) Element 2[a]

Bennett teaches a memory-side memory manager for managing a GAT stored in flash memory, and the logical-to-physical address mappings therein, as discussed above for claim 1[f]. Bennett further teaches that storing mapping information in flash memory allows for reconstruction of volatile mapping records, e.g., "when the system is initialized after power-up." Ex. 1002, 10:47-49. Bennett also teaches several erase methods that "set flag values so that the system can recover in the case of a power loss event." *Id.*, 18:23-27. Thus, a POSITA would have understood that memory manager and GAT are the claimed "index reconstruction module." Baker, ¶¶ 141-43. 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. Baker, ¶ 141; *see also* Ex. 1002, 4:50-52 (disclosing "circuitry and firmware, to execute an erase or erase equivalent"), 7:37-39 ("the memory manager contains a number of software modules for managing erase" operations), 9:40-44 (disclosing controller modules for managing flash tables).

b) Element 2[b]

As explained for claims 1[d] and [f], Bennett teaches marking the non-volatile GAT to indicate that the host data associated with the logical address has been erased from a user's perspective. Bennett also teaches a "GAT cache" stored in volatile

memory. *Id*., 12:12-17. Given Bennett's teaching of using non-volatile address mappings to update volatile address mappings, discussed in claim 2[a] above, a POSITA would have understood that Bennett teaches this claim 2[b] with respect to updating the volatile GAT cache with non-volatile GAT indications. Baker, ¶¶ 144-145. Bennett further provides a detailed description of storing non-volatile address mappings in non-volatile memory for updating volatile address mappings. *See, e.g.*, Ex. 1002, Fig. 9, 13:12-13, 13:65-14:7 ("These lists may be reconstructed during system initialization after a power-down, via information in the erased block lists and address translation tables stored in sectors in flash memory"). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 144-45.

C. Claim 3

a) Element 3[a]

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). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 146-47.

b) Element 3[b]

This claim element recites limitations indistinguishable from the limitations of claim 2[a]-[b] and would have been obvious to a POSITA for the same reasons. Baker, ¶¶ 148-51.

D. Claim 4

a) Element 4[a]

This claim element recites limitations similar to the limitations of claim 3[a] except for requiring "a plurality of index entries." For the reasons discussed for claim 3[a], a POSITA would have understood that Bennett teaches this element. Baker, ¶ 152 (Bennett teaches multiple index entries for its multiple addresses); *see also* Bennett's Figs. 2, 3A, 4, 8A, 8B (disclosing plurality of index entries).

b) Element 4[b]

This claim element recites limitations indistinguishable from the limitations of claim 2[a]-[b] and would have been obvious to a POSITA for the same reasons. Baker, ¶ 153.

E. Claim 5

As stated above for claim 2[b], Bennett teaches a "reconstruction module is configured [to] indicate that data of the logical identifier are erased." Bennett further teaches for full group erases, "a special record, maintained in the non-volatile memory, about the erase operation to be performed, where the record can be updated at the completion of the erase operation as a whole or as parts of it are completed." Ex. 1002, 18:28-32. For example, Bennett teaches updating the "GAT marking the Logical Group as physically erased." *Id.*, 19:16-17. A POSITA would have also understood that Bennett teaches using the non-volatile GAT to update volatile GAT caches after a physical erase. Baker, ¶ 155; *see* claim 2[b] ("Bennett teaches using

address mappings in non-volatile memory to reconstruct volatile address mappings). Thus, a POSITA would have understood that Bennett teaches this claim. Baker, ¶¶ 154-55.

F. Claim 8

As discussed above for claim 1, for logical erasures, Bennett teaches marking entries in a GAT as "logically erased" and "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." Ex. 1002, 20:49-50. A POSITA would have understood that the hardware and software for performing this functionality was a "read request module" as recited in the claims. Baker, ¶ 157. Thus, a POSITA would have understood that Bennett teaches this claim. *Id*., ¶¶ 156-57.

G. Claim 9

As discussed above in claim 1[f], for partial group erasures, Bennett teaches marking a GAT with a flag. This marking indicates that the contents of the physical storage location no longer need to be retained and "can then be physically erased when convenient, for example in a background process." Ex. 1002, 3:39-41.

Thus, a POSITA would have recognized that Bennett teaches "a marking module configured to record that contents of the physical storage location associated with the logical identifier no longer need to be retained on a non-volatile storage medium in response to the indication" for partial group erasures. Baker, ¶¶ 158-59.

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.*, ¶ 159.

H. Claim 10

As above in claim 9, Bennett teaches a "marking module" that sets a flag in response to an erase command. A POSITA would understand this marking to invalidate the logical-to-physical index entry. Baker, ¶ 161.

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, ¶¶ 160-61.

I. Claim 11

UTL asserts that this element is met by "circuitry and/or software/firmware" that implements "garbage collection" to "return[] 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 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, ¶ 163.

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, ¶ 163. A POSITA would have found it obvious to perform garbage collection in response to an erase command; for example, in order to determine whether there are other areas that need to be efficiently erased at the same time. Baker, ¶¶ 162-64. 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, ¶¶ 162-64.

J. Claim 12

For full erasures, as identified in the claim 1[f] discussion above, Bennett teaches erasing the physical storage location in response to an indication of a full

logical group erasure. A POSITA would have recognized the erasure is implemented by circuitry and/or software/firmware in a controller and thus encompasses an "erase module." *See, e.g.*, § VI, *supra* (construing "module"); Baker, ¶¶ 165-66. Thus, a POSITA would have understood that Bennett teaches this claim. *Id.*

K. Claim 22

a) Element 22[a]

Bennett teaches a computing device in the form of a host and memory system with a "host-side memory manager" and a "memory-side memory manager." *See, e.g.*, Ex. 1002, Fig. 2. In the memory system, a processor executes instructions from the ROM or optional programmable nonvolatile memory. *Id.*, Fig. 1, 7:4-7. For these reasons, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 167-68.

b) Element 22[b]

This claim element recites limitations indistinguishable from the limitations of claim 3[a] and would have been obvious to a POSITA for the same reasons. Baker ¶¶ 169-70.

c) Element 22[c]

First, as discussed for claim 1[c], Bennett teaches receiving a message comprising a "logical identifier" under either proposed construction. The logical identifier is mapped to a physical storage location. *See* Ex. 1002, Fig. 3B (showing

mapping).

Second, UTL interprets "logical identifier in the index is empty" as "data identified by the [logical identifier] in the index does not need to be preserved." *See* § VI, *supra*. Bennett teaches an erase command indicating the same thing. *See, e.g.*, §§ IX.A.c-f, *supra*. For these reasons, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 171-73.

d) Element 22[d]

As discussed above for claim 1[f], Bennett teaches marking a GAT with a flag to indicate that data need not be preserved. *See also* Ex. 1002, 20:45-50, 5:60-61 ("actual, physical erase taking place at a later time"). These GAT markings are persistent data because the GAT is stored in non-volatile flash memory. *See id.*, Fig. 6. Thus, a POSITA would have found that Bennett teaches this claim under UTL's construction. Baker, ¶¶ 174-80.

L. Claim 23

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 notations. *See, e.g.*, §§ IX.A.c, f (setting flags in non-volatile GAT), § IX.H (invalidate index entries), *supra*. For these reasons, a POSITA would have understood that Bennett teaches this claim. *Id.*, ¶¶ 181-82.

M. Claim 24

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 a 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 "removing the mapping between the logical identifier and the physical storage location from the index." Baker, ¶ 183 (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.H, X.M, *infra* (Ground 2). For these reasons, a POSITA would have understood that Bennett teaches this claim. Baker, ¶ 183.

N. Claim 25

a) Element 25[a]

This claim element recites limitations indistinguishable from the limitations of claim 2[a] and would have been obvious to a POSITA for the same reasons. Baker ¶¶ 184-87.

b) Element 25[b]

UTL interprets "logical identifier is empty" as "data identified by the [logical identifier] in the index does not need to be preserved"." *See* § VI, *supra*. Under this construction, claim 25[b] is a combination of claims 2[b] (reconstruction module configured to indicate that logical identifier data are erased based on persistent data), 8 (while the data associated with the logical identifier remains on the physical

storage location), and 22[d] (persistent data configured to indicate that the logical identifier is empty). The "determining" in claim 25 occurs when data is read as recited in claim 8. Ex. 1013, 7 (UTL interpreting the determining step with "a read command that occurs"). Thus, for the same reasons as discussed for claims 2[b], 8 and 22[d], a POSITA would have found this element obvious. Baker, ¶¶ 188-92.

O. Claim 26

UTL appears to interpret "the data associated with the logical identifier" as relating to data on the non-volatile storage medium. Ex. 1013, 4, 8-9. Under this construction, claim 26 is indistinguishable from the limitations of claim 8 and would have been obvious to a POSITA for the same reasons. Baker ¶¶ 193-95.

X. Ground 2: Obvious Over Suda and POSITA Knowledge

A. Claim 1

a) Element 1[a][2]

Suda discloses a "memory management device for managing a nonvolatile semiconductor memory." Ex. 1003, 1:15-17, Abstract, Fig. 1 (annotated below). Thus, a POSITA would have understood that Suda teaches as the claimed apparatus. Baker, ¶¶ 196-98.

[2] *See* attached Claim Listing.

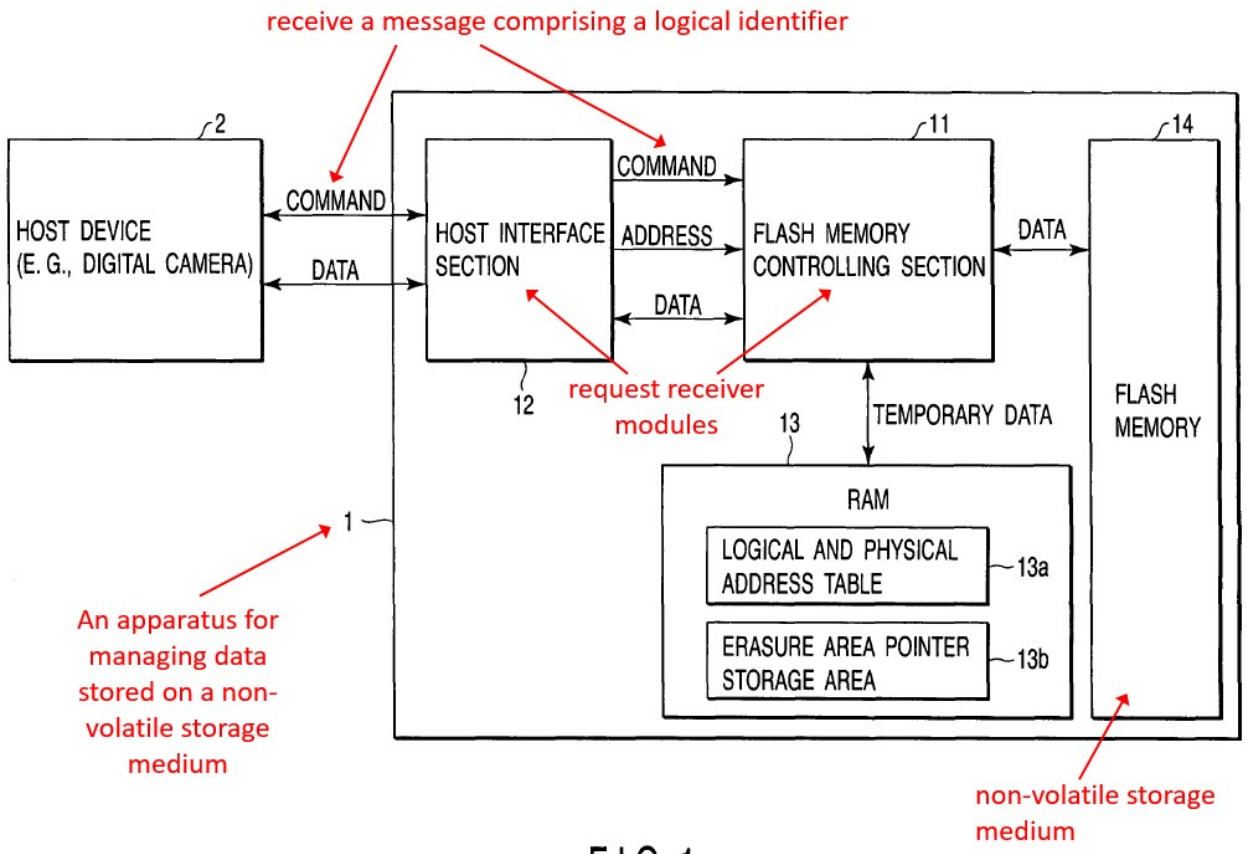


FIG. 1

Ex. 1003, Fig. 1 (annotated).

b) Element 1[b]

Suda discloses a memory card or memory device that comprises a NAND type non-volatile flash memory. Ex. 1003, 2:57-66; *see also id.*, Fig. 1 (annotated above). Thus, a POSITA would have understood that Suda teaches this element. Baker, ¶ 199.

c) Element 1[c]

Suda teaches the recited "request receiver module" in the form of a host interface section and/or the flash memory controlling section. Ex. 1003, Fig. 1 (annotated above at element 1[a]). Both of these components receive commands

originating from the host device. *Id.*, 2:63-3:11.

UTL interprets "module" as "a hardware circuit and/or programmable hardware and/or software implemented within a storage controller." *See* § VI, *supra*. A POSITA would have understood that Suda's host interface section and/or flash memory controlling section includes "a hardware circuit and/or programmable hardware and/or software implemented within a storage controller." Baker, ¶ 202.

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); *see* § VI, *supra* (construing term). Thus, a POSITA would have understood that Suda teaches this element. Baker, ¶¶ 200-04.

d) Element 1[d]

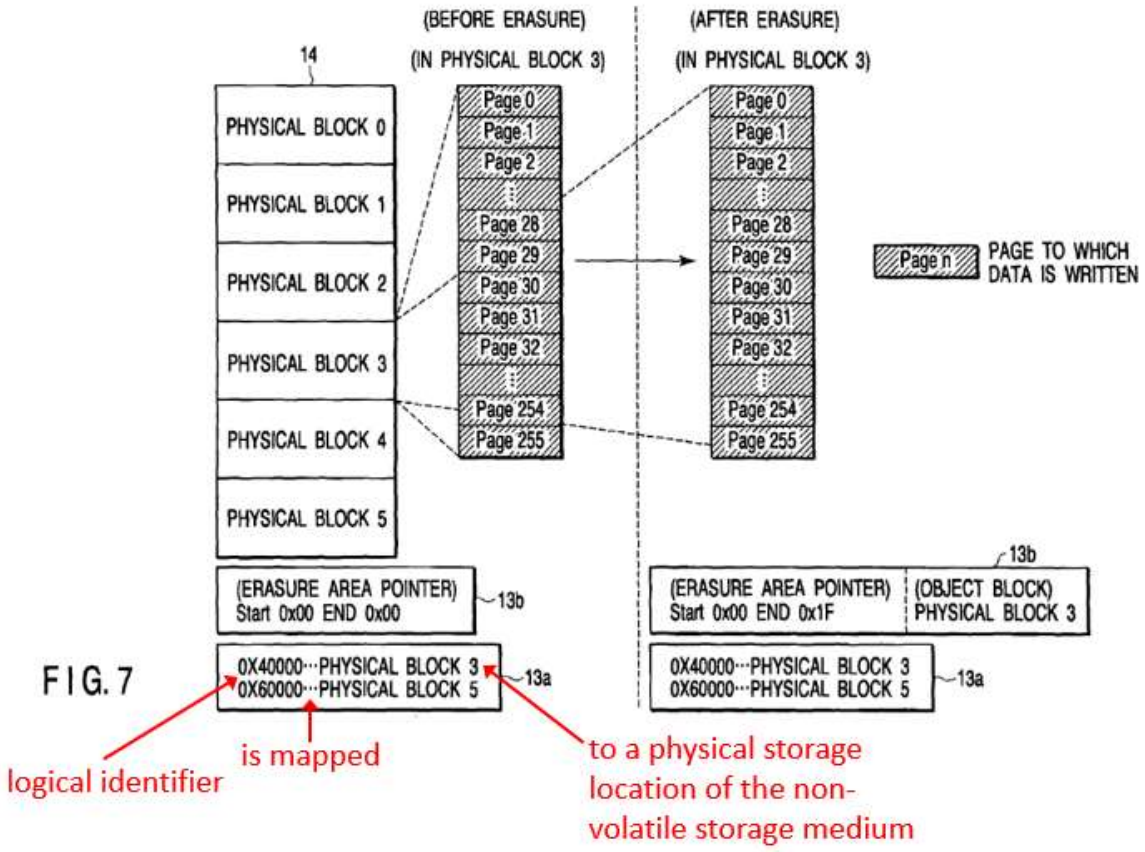
UTL contends "data associated with the logical identifier [has been/is] erased," refers to data that appears deleted from a user's perspective, for example by a computer attached to a storage device. *See* § VI, *supra*. As a result, "[f]rom the user's perspective, this data has been deleted from a document" even though it remains on the storage device. *Id.*

Suda teaches the same thing. A host device (e.g., a digital camera) sends an erase command that includes a logical identifier to the memory device. Ex. 1003,

8:66-9:3, Fig. 1. The erase command designates a logical block address that indicates "a block in which the data to be erased is written, when the erase command is issued." *Id.*, 1:66-67, 8:66-9:3. Suda's system then processes the erase command to prevent users from later reading this erased data. *Id.*, 8:21-50, Fig. 9. Thus, from the perspective of a user of Suda's digital camera, the designated data has been deleted, and a POSITA would have understood that these erase commands indicate that data has been erased by the host device, as claimed. Baker, ¶¶ 205-07.

e) Element 1[e]

As shown in annotated Fig. 7 below, Suda's logical and physical address translation table 13a maps logical addresses (the recited "logical identifier") to block numbers (the recited "physical storage locations") of the flash memory.



Ex. 1003, Fig. 7 (annotated), 3:42-55. Thus, a POSITA would have understood that Suda teaches this element. Baker, ¶¶ 208-09.

f) Element 1[f]

Suda teaches storing erasure area pointers in response to the erase command (the recited "indication"). Ex. 1003, 5:38-46, 6:60-63, 7:30-53, Fig. 8 (steps S1-S4). These erasure area pointers "indicate that the data associated with the logical identifier is erased," as recited, because the erasure area pointers indicate which storage locations are virtually erased. *Id.*, 5:9-61, 6:18-21, 7:5-55, Fig. 7 (table 13b), Fig. 8 (steps S1-S4); *see also* § X.A.d (explaining that data is "erased" from a user's perspective, as construed by UTL). Data located at addresses falling within the

erasure area pointers "are in a virtual erased state" and "subjected to virtual erasure." *Id.*, 5:20-26.

Suda teaches that the erasure area pointers are also stored as "persistent data," as recited, because both the RAM 13 and the non-volatile flash memory 14 (the claimed "non-volatile storage medium") store copies of the erasure area pointer storage area 13b. Ex. 1003, 3:41-43, 8:3-11, Fig. 1 (annotated below). Suda's system "writes, **in the flash memory 14 also**, the data items written to the erasure area pointer storage area 13*b*." *Id.*, 8:3-8 (emphasis added). The copy in Suda's nonvolatile NAND flash memory 14 is the recited "persistent data" that preserves the erasure area pointers through power-off events. *Id.*, 2:65-66, 8:3-16; *see* § VI, *supra* (construing term).

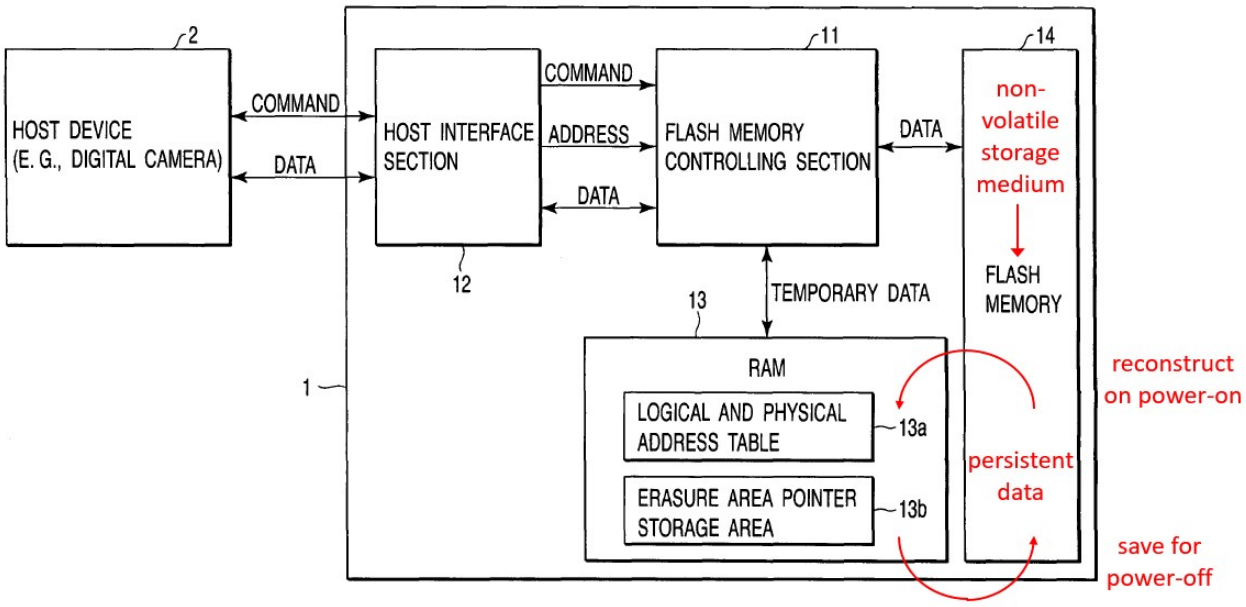


FIG. 1

Fig. 1003, Fig. 1 (annotated).

A POSITA would have understood that Suda's flash-memory controlling section controls storage and is the recited "storage module." Baker, ¶ 213 (citing Ex. 1003, 7:45-55). This is true even under UTL's construction of "module" because the flash-memory controlling section implements this functionality through a hardware circuit and/or programmable hardware and/or software in a controller. Baker, ¶ 213. Thus, a POSITA would have understood Suda to teach this element. *Id.* ¶¶ 210-14.

B. Claim 2

a) Element 2[a]

Suda's logical and physical address table 13a includes mappings between logical identifiers and physical storage locations of the non-volatile storage medium as discussed above for claim 1[e]. *See, e.g.*, Ex. 1003, 3:48-55. Suda's flash-memory controlling section manages data erasure and the tables; thus it operates as the index reconstruction module. Ex. 1003, 3:13-15.

Suda does not explicitly state to store information in the flash memory 14 for reconstructing the logical and physical address table 13a and then to reconstruct the logical and physical address table 13a in RAM 13. Even so, a POSITA would have known to store reconstruction information in the flash memory and reconstruct the logical and physical address table in RAM after power-on events. Baker, ¶ 220; Ex.

1003, Fig. 1 (annotated below). Suda teaches reconstructing the storage and reconstruction of the erasure area pointer area after power-on events. Ex. 1003, 8:6-16. A POSITA would have known that flash-memory devices also use this same technique to preserve the logical-to-physical mappings in the flash memory when powered off. Baker, ¶¶ 218-20. A POSITA would have understood that both tables need to be reconstructed for the same reasons. *Id.*, ¶¶ 215-20, 224. Otherwise, the logical-to-physical mappings would be lost every time the power is turned off. *Id.*, ¶ 220. Thus, a POSITA would have understood Suda to teach this element. Baker, ¶¶ 216-20, 222.

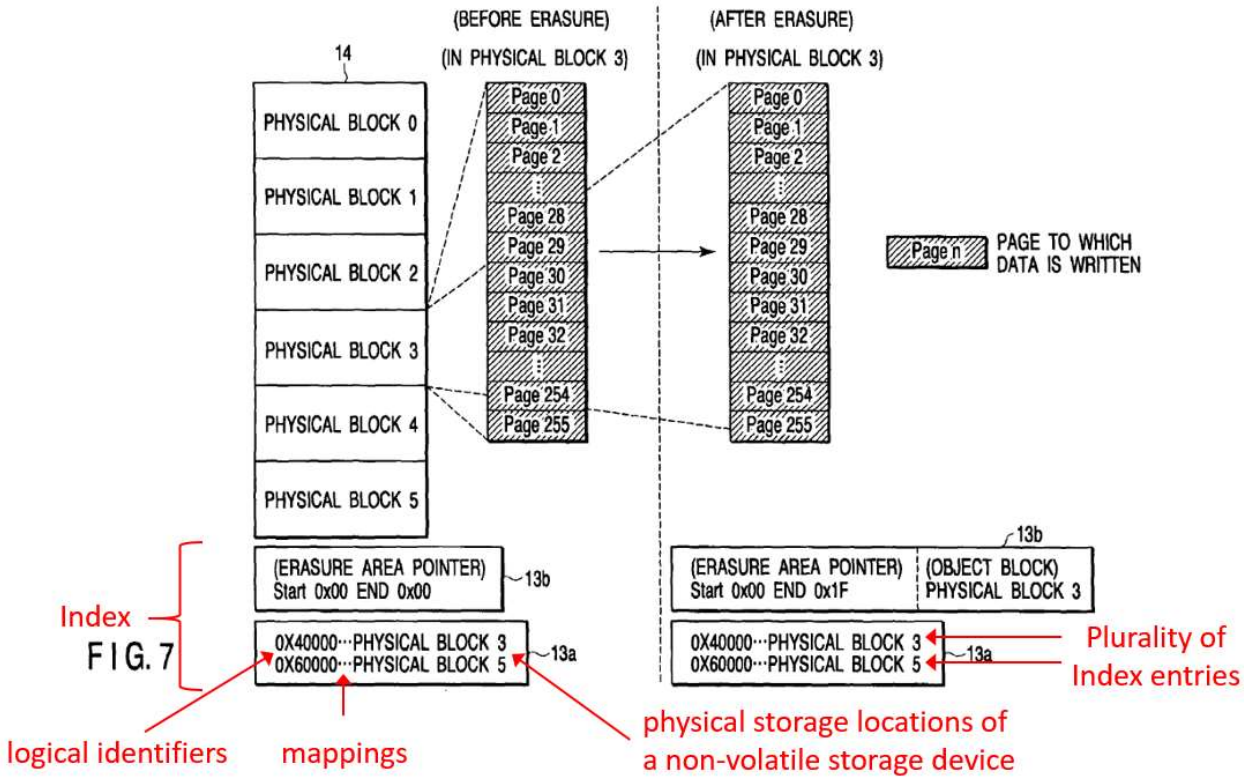
b) Element 2[b]

As explained with respect to Claim 1[d] and [f] above, Suda teaches storing persistent data in non-volatile memory to indicate that the host data associated with the logical address has been erased from a user's perspective. After each power-off-and-on cycle, the flash-memory controlling section reads this data from flash memory and writes the erasure area pointer storage area to RAM. Ex. 1003, 8:13-17, Fig. 1. Thus, by this writing, a POSITA would have understood that Suda's system indicates that data of the logical identifier are erased, based on the persistent data stored on the non-volatile storage medium, as recited, during this reconstruction process. Baker, ¶ 225.

C. Claim 3

a) Element 3[a]

Suda teaches an index that includes logical and physical address table 13a and erasure area pointer storage area 13b. *See, e.g.*, Ex. 1003, 3:41-55, Figs. 1, 7 (annotated below). The logical and physical address table 13a maps the "logical addresses and physical addresses allocated to physical blocks in which data items are written, of the physical blocks in the flash memory 14." *Id.,* 3:43-55 (discussing example), Figs. 1, 7 (element 13a). Thus, a POSITA would have understood that Suda teaches this element. Baker, ¶ 227.



Ex. 1003, Fig. 7 (annotated).

b) Element 3[b]

This claim element recites limitations indistinguishable from the limitations of claim 2[a]-[b] and would have been obvious to a POSITA for the same reasons. Baker, ¶ 228.

D. Claim 4

a) Element 4[a]

This claim element recites limitations similar to the limitations of claim 3[a], except for requiring "a plurality of index entries." For the reasons discussed for claim 3[a], Suda also teaches this element. *See* Baker, ¶¶ 229-32. Suda's index includes a plurality of index entries. *E.g.*, *id.*, 3:48-55, Figs. 7 (showing plurality of index entries in 13a, annotated above at § X.C.a), 10.

b) Element 4[b]

This claim element recites limitations indistinguishable from the limitations of claim 2[a]-[b] and would have been obvious to a POSITA for the same reasons. Baker, ¶ 233.

E. Claim 5

As discussed for claim 2[b], Suda teaches a "reconstruction module is configured [to] indicate that data of the logical identifier are erased." Suda teaches updating the index to indicate that data associated with the logical identifier are erased. *E.g.*, Ex. 1003, Fig. 7 (showing update to erasure area pointers 13b on "AFTER ERASURE" side), Fig. 8 (block S4), Fig. 10 (showing update to erasure

area pointers 13b on "AFTER ERASURE" side). These erasure area pointers indicate that data is "in a virtual erased state." *Id.*, 5:19-27, 5:36-53. Thus, a POSITA would have understood Suda to teach this claim. Baker, ¶¶ 234-37.

F. Claim 8

Suda teaches this claim. Suda teaches a flash-memory controlling section that, in response to a data-read command that includes a logical block address, determines whether a page is included in an erasure area and, if so, outputs initial-value data. Ex. 1003, 8:21-41; Fig. 9. "Initial-value data" means data initially written in an unused physical block where "user data is not written," or in other words, empty data. *Id.*, 3:58-59, 4:4-6; Baker, ¶ 110. This read process returns initial-value data even though the actual data remains stored in the physical storage location. Ex. 1003, Fig. 7 (annotated at § VIII.B, *supra*, showing, with shading, that the data still remains written in physical block 3 while the erasure area pointer 13b indicates the virtually erased state of data). Thus, Suda's disclosure of outputting initial-value data indicates that "the data of the logical identifier are erased," even "while the data associated with the logical identifier remains on the physical storage location of the non-volatile storage medium," as claimed. Baker, ¶¶ 238-40.

A POSITA would have understood that the hardware and software for performing this functionality was a "read request module" as recited in the claims. Baker, ¶ 239. In particular, Suda teaches that the flash-memory controlling section

processes the read command using a logical-to-physical address table and erasure area pointer storage area "to output initial-value data as data to be read in response to a data read command." Ex. 1003, 2:2-3, 8:24-34, Fig. 1. Thus, a POSITA would have understood these components to act as the claimed "read request module." Baker, ¶¶ 239-40.

G. Claim 9

Suda teaches that the erasure area pointers record physical storage locations "subjected to virtual erasure." Ex. 1003, 5:9-61. In other words, the erasure area pointers indicate "data items to be erased" later. *E.g., id.*, 5:40-48, 5:57-61, 6:9-14, 6:29-45, 6:60-64, 7:38-41. Suda's system will in fact erase these blocks later, once erasure pointers mark the entire block. *E.g., id.*, 6:34-41, Fig. 6 block B, 5:54-6:3, Fig. 4. A POSITA would have understood that data items "to be erased" means that the data items "no longer need to be retained," as claimed. Baker, ¶ 241.

The flash-memory controlling section updates the stored erasure area pointer storage in response to an erase command (the claimed "indication"). Ex. 1003, 3:13-15, 5:19-27, 5:38-43, 6:18-21, 7:5-55, Fig. 8 S1-S4. Thus, a POSITA would have recognized that Suda teaches "a marking module configured to record that contents of the physical storage location associated with the logical identifier no longer need to be retained on a non-volatile storage medium in response to the indication." Baker, ¶¶ 241-43. This is true under even UTL's construction of "module" because

Suda teaches implementing this functionality through a flash-memory controlling section that includes a hardware circuit and/or programmable hardware and/or software and are data that is retained in the absence of power. *Id*., ¶ 242. Therefore, a POSITA would have understood that Suda teaches this claim. *Id.*, ¶¶ 241-43.

H. Claim 10

As shown in claim 9, Suda teaches a "marking module" under UTL's construction. Suda further teaches, with respect to Fig. 4, "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, 6:35-41 ("the flash memory controlling section … erases address information of the physical block B"). These cancellation/erases occur in response to an erase command (the claimed indication). *Id.*, 5:37-46, 6:22-25. A POSITA would have also known that the cancellation/erasure of the logical block address and the physical address in the logical-and-physical address table invalidates the association between those addresses. Baker, ¶¶ 244-45.

Alternatively, a POSITA would have also reached this understanding from Suda's teaching of "erasure area pointers" to indicate that certain physical storage locations are "virtually erased." Ex. 1003, 5:9-61, Fig. 7 (element 13b), Figs. 3-6; Baker, ¶ 246. For example, Suda Fig. 4 shows erasure area pointers indicating that a physical block A is virtually erased. Ex. 1003. A POSITA would have understood

that a virtually erased state is an invalid state. Baker, ¶¶ 112, 211, 246.

Therefore, a POSITA would have found that Suda teaches this claim. *Id.*, ¶¶ 244-46.

I. Claim 11

UTL asserts that this element is met by "circuitry and/or software/firmware" that implements "garbage collection in response to the TRIM command." Ex. 1013, 5.

Suda teaches a storage recovery module including the flash-memory controlling section 11, the flash memory 14, and the RAM 13. Ex. 1003, Fig. 1. These components erase the contents of blocks in a storage-recovery process whenever an erase command triggers the erasure area pointers to indicate a block has filled with virtually erased data. *Id.*, 5:44-6:3, 6:18-21, Fig. 8 (steps S1, S6). For example, in Fig. 6, "an erase command to erase data items written to a number of physical blocks in the flash memory 14 is issued." *Id.*, 6:15-21. With respect to physical block B in Fig. 6, Suda instructs, "the data items written to the entire area of the physical block B are to be erased." *Id.*, 6:34-36. These blocks are "to be erased [to] be set in an unused state," where they contain "initial-value" data and "can be used" again. *Id.*, 3:56-67, 5:65-6:3; *see also id.*, 1:44-55 (otherwise, "flash memory cannot be overwritten"). A POSITA would have understood these disclosures to mean that block B will have its contents physically erased in a storage-

recovery process so that block B can then write new data again. Baker, 250-51. Therefore, a POSITA would have understood that Suda teaches this claim. *Id*.

J. Claim 12

As discussed for claim 11, Suda teaches a storage-recovery module and teaches to erase the contents of blocks in a storage-recovery process whenever the erasure area pointers indicate a block has filled with virtually erased data in response to an erase command. *See* § X.I, *supra* (citing Ex. 1003, 5:44-6:3, 6:18-21, 6:34-36, Fig. 8 (step S6)). A POSITA would have understood these disclosures to mean that Suda's block will have its contents physically erased in a storage-recovery process so that the block can then write new data. Baker ¶ 254. The erasing would be performed by an "erase module" made of circuitry and/or software/firmware, such as Suda's flash-memory controlling section. *Id*. Therefore, a POSITA would have understood that Suda teaches this claim. *Id*.

K. Claim 22

a) Element 22[a]

Suda teaches a computing device in the form of a memory card or memory device 1 that includes a flash-memory controlling section 11 that manages data erasure. Ex. 1003, 2:57-3:15, Fig. 1. A POSITA would have understood that Suda's flash-memory controlling section 11 performs the management by executing instructions from a non-transitory computer-readable storage medium located either within the flash-memory controlling section or in the flash memory. Baker, ¶ 257.

Therefore, a POSITA would have understood that Suda teaches this element. *Id*.

b) Element 22[b]

This claim element recites limitations indistinguishable from the limitations of claim 3[a] and would have been obvious to a POSITA for the same reasons. Baker, ¶ 258.

c) Element 22[c]

First, as discussed for claim 1[c], Suda teaches receiving a message comprising a logical identifier under either proposed construction of logical identifier. *See* Ex. 1003, Fig. 1 (annotated below); Baker, ¶¶ 259-61. This logical identifier is mapped to a physical storage location in Suda's logical and physical address table. Ex. 1003, Figs. 1, 7.

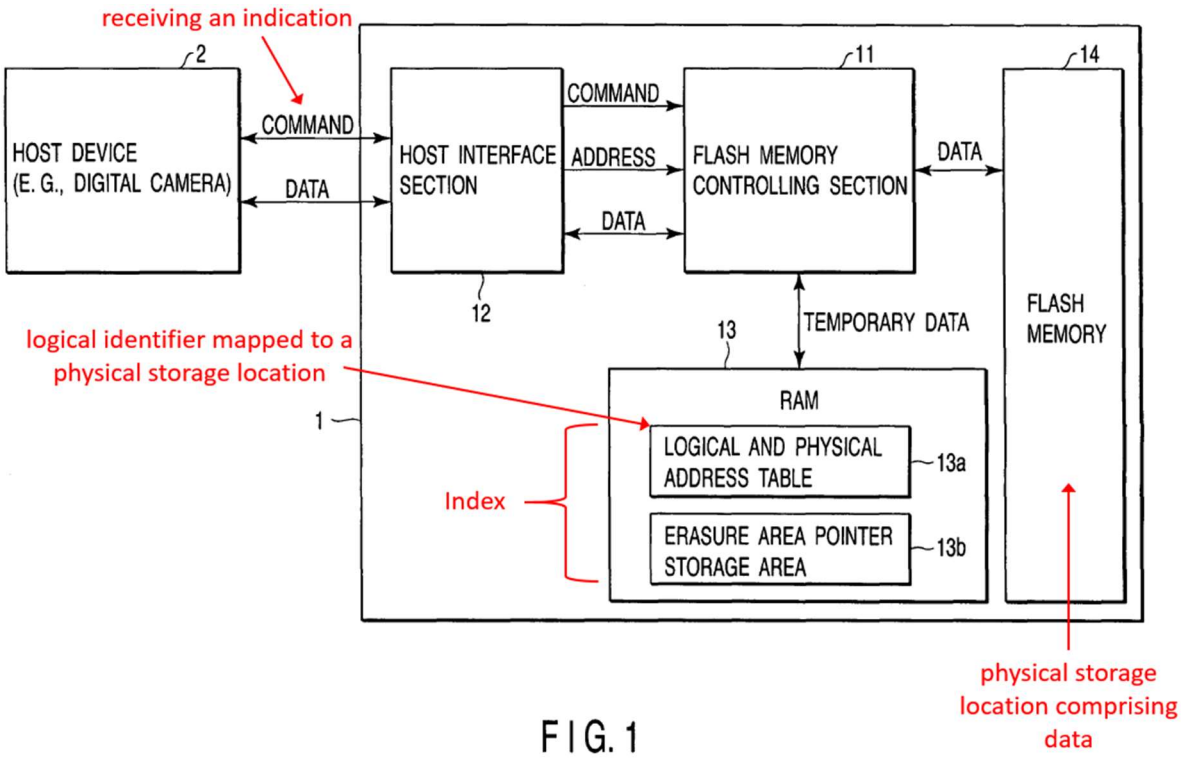


FIG. 1

Ex. 1003, Fig. 1.

Second, a POSITA would understand that Suda teaches that this message is an indication as claimed under UTL's interpretation of the claim. Baker, ¶ 261. UTL interprets "logical identifier [in the index] is empty" to mean "data identified by the [logical identifier] in the index does not need to be preserved." *See* § VI, *supra.* Suda teaches the same thing: receiving an erase command that designates a logical identifier and indicating that data identified by the logical identifier, as mapped by the logical and physical address table, does not need to be preserved because the data is "to be erased." Ex. 1003. at 5:40-48, 5:57-61, 6:9-14, 6:29-45, 6:60-64, 7:38-41; *see, e.g.*, §§ X.A.c-f, *supra*. For these reasons, a POSITA would have understood that Suda teaches this element. Baker, ¶¶ 259-61.

d) Element 22[d]

As discussed above for claim 1[f], Suda teaches storing erasure area pointers, indicating that the data is "to be erased" and not preserved. Ex. 1003, 5:35-43, 5:57-61, 6:9-14, 6:60-64, 7:38-41. Suda stores erasure area pointers as "persistent data on the non-volatile storage medium in response to the indication." *See* § X.A.f, *supra*. Thus, a POSITA would have found that Suda teaches this claim under UTL's construction. Baker, ¶¶ 262-66.

L. Claim 23

UTL appears to interpret "data packet" as "the area of the [memory] that

contains the data." Ex. 1013, 6. Suda Figs. 3-5 shows invalidating areas of memory that contain the data by setting erasure area pointers in response to an erase command:

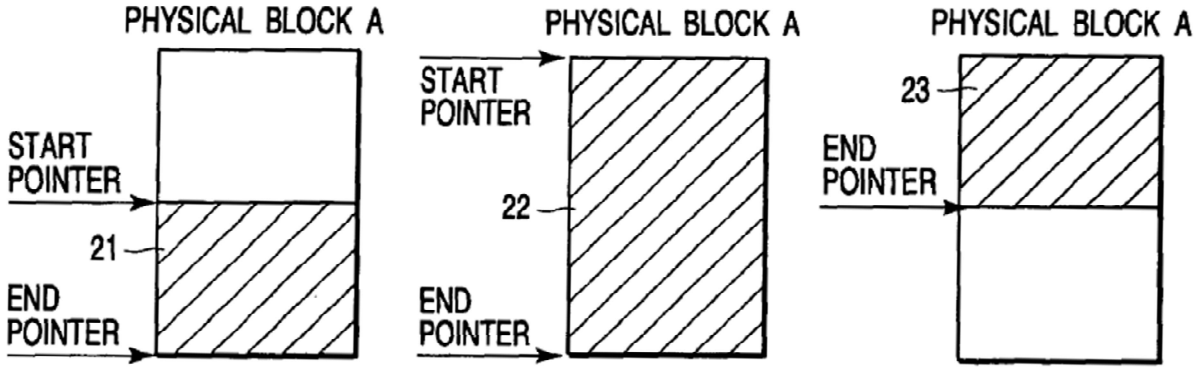


FIG. 3 FIG. 4 FIG. 5

Ex. 1003, Figs. 3-5, 5:38-43, 5:55-57.

Suda also teaches canceling/erasing relationships in the logical and physical address table in response to an erase command, as explained for claim 10. *See* X.H, *supra*. A POSITA would have also understood that these cancelations/erasures indicate that an area of memory located at the canceled/erased physical address is invalid. Baker, ¶ 268. For these reasons, a POSITA would have understood that Suda teaches this claim. *Id.*, ¶¶ 267-70.

M. Claim 24

Suda teaches to remove index mappings in the logical and physical address table when entire blocks are erased. Ex. 1003, 5:54-6:3, 7:29-8:2. For example, with respect to 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, in order that a physical block (area 22) to be erased be set in an unused state." *Id.*, Fig. 4, 5:62-6:3; *see also* 1:27-30, 6:18-21 (explaining that this occurs "in the case where an erase command to erase data items … is issued"), 6:35-41 ("the flash memory controlling section … erases address information of the physical block B"), 7:64-8:2, Fig. 8 (steps S5, S6). Therefore, a POSITA would have understood that Suda teaches this claim. Baker, ¶ 273.

N. Claim 25

a) Element 25[a]

This claim element recites limitations indistinguishable from the limitations of claim 2[a] and would have been obvious to a POSITA for the same reasons. Baker, ¶ 275.

b) Element 25[b]

UTL interprets "logical identifier is empty" as "data identified by the [logical identifier] in the index does not need to be preserved"." *See* § VI, *supra*. Under this construction, claim 25[b] is a combination of claims 2[b] (reconstruction module configured to indicate that logical identifier data are erased based on persistent data), 8 (while the data associated with the logical identifier remains on the physical storage location), and 22[d] (persistent data configured to indicate that the logical identifier is empty). The "determining" in claim 25 occurs when data is read as recited in claim 8. Ex. 1013, 7 (UTL interpreting the determining step with "a read

command that occurs"). Thus, for the same reasons as discussed for claims 2[b], 8 and 22[d], a POSITA would have found this element obvious. Baker, ¶ 278.

O. Claim 26

UTL appears to interpret "the data associated with the logical identifier" as relating to data on the non-volatile storage medium. Ex. 1013, 4, 8-9. Under this construction, claim 26 is indistinguishable from the limitations of claim 8 and would have been obvious to a POSITA for the same reasons. Baker, ¶ 279.

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

A. Claim 2

a) Element 2[a][3]

Claim 2 depends on claim 1, which Suda teaches under ground 2. *See* § X.A, *supra*. Suda's logical and physical address table includes mappings between logical identifiers and physical storage locations of the non-volatile storage medium as discussed above for claim 1[e]. *See, e.g.*, Ex. 1003, 3:48-55; *see* § X.A.e, *supra*. To the extent UTL argues that Suda does not obviate claim 2[a], Suda with SwSTE'05 renders claim 2[a] obvious.

SwSTE'05 explains that flash devices store an inverse map "on the flash device itself." Ex. 1010 § 2.2. "The main use of the inverse map is to reconstruct a direct map during device initialization (when the device is inserted into a system or

[3] *See* attached Claim Listing.

when the system boots)." *Id.* The "direct map" refers to a logical-to-physical address mapping, such as Suda's logical and physical address table 13a. *Id.*; Baker, ¶ 222. This "direct map in RAM . . . is reconstructed during initialization," which occurs when a device is powered back on. Ex. 1010 § 2.2.

A POSITA would have found it obvious to modify Suda's system with this teaching of SwSTE'05 so that Suda also stored an inverse map of the logical and physical address table in the flash memory, and then reconstructed the logical and physical address table (the direct map) in RAM 13 during device initialization (a power-on event). Baker, ¶ 223. The reason for this modification is the same reason Suda provided for storing and reconstructing the erasure area pointer area: to preserve the mappings through power-off events. *Id.*; Ex. 1003, 8:3-16. A POSITA would have had a reasonable expectation of successfully preserving the logical and physical mappings because this same technique works for preserving the erasure area pointers. Baker, ¶ 223. Thus, this element would have been obvious in view of the combination of Suda and SwSTE'05. *Id.*, ¶¶ 221-24.

Thus, a POSITA would have found it obvious to modify Suda's memory device to reconstruct mappings between logical identifiers and physical storage locations of the non-volatile storage medium (Suda's logical and physical address table 13a) from contents (an inverse map of Suda's logical and physical address table 13a) of the non-volatile storage medium in view of SwSTE'05. *Id.*

b) Element 2[b]

Suda teaches this element for the reasons discussed under ground 2. *See* §§ X.B.b, *supra*.

B. Claim 3

a) Element 3[a]

Suda teaches element for the reasons discussed under ground 2. *See* § X.C.a, *supra*.

b) Element 3[b]

This element is obvious over Suda and SwSTE'05 for the reasons discussed for claim 2 under ground 3. *See* §§ XI.A.a-b, *supra* (explaining inverse map on flash memory).

C. Claim 4

a) Element 4[a]

Suda teaches element for the reasons discussed under ground 2. *See* § X.D.a, *supra*.

b) Element 4[b]

This element is obvious over Suda and SwSTE'05 for the reasons discussed for claim 2 under ground 3. *See* §§ XI.A.a-b, *supra* (discussing reconsruction based on inverse map).

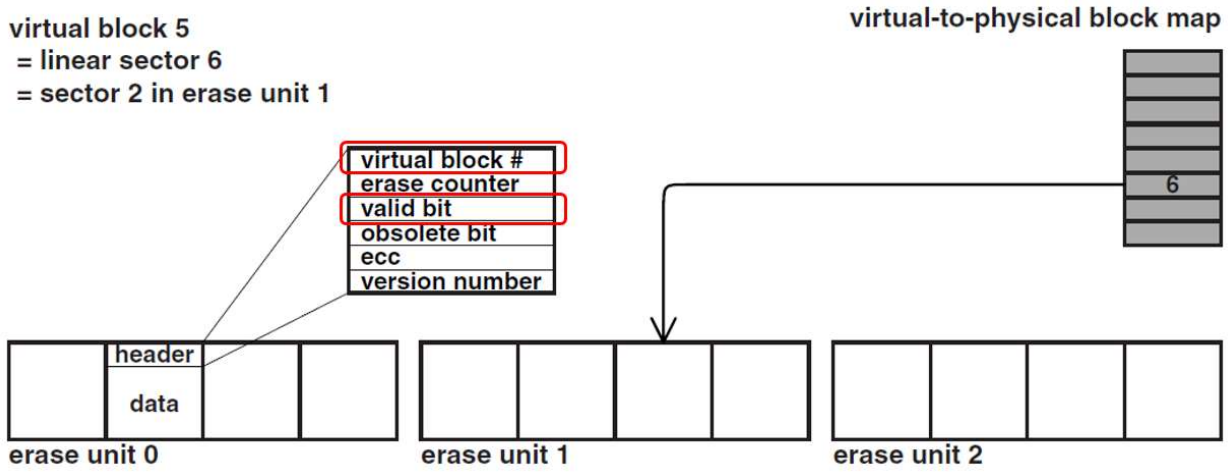
D. Claim 5

Claim 5 depends on Claim 4, which is obvious for the reasons discussed for

Ground 3. *See* § XI.C, *supra.* Suda teaches the rest of this claim for the reasons discussed for claim 5 under ground 2. *See* § X.E.a.

E. Claim 10

Claim 10 depends on claim 1, which Suda teaches as explained under ground 2. *See* § X.A, *supra*. To the extent that UTL argues that claim 10 is not obvious over Suda, Suda with SwSTE'05 renders claim 10 obvious. Fig. 1 (annotated below) of SwSTE'05 shows that the virtual-to-physical[4] block map points to blocks that have headers that explicitly include a "valid bit" associated with each "virtual block #." Ex. 1010 § 2.2. An index entry pointing to a block with this valid bit set as invalid indicates that the mapping is to an invalid logical address. Baker, ¶ 248.



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, ¶ 248. Suda's

[4] "Virtual" means "logical." Baker, ¶ 79.

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.*; Ex. 1003, 8:66-9:3. This would have been an obvious modification because both headers and valid bits were standard techniques. Baker, ¶ 247. A POSITA would have had a reasonable expectation of successfully using a small fraction of Suda's existing memory, whether RAM or flash memory or both, to store the valid bit associated with each logical block address. *Id.*, ¶ 248. These options are illustrated below. Therefore, a POSITA would have found this claim obvious based on Suda in combination with SwSTE'05. *Id.*, ¶¶ 247-49.

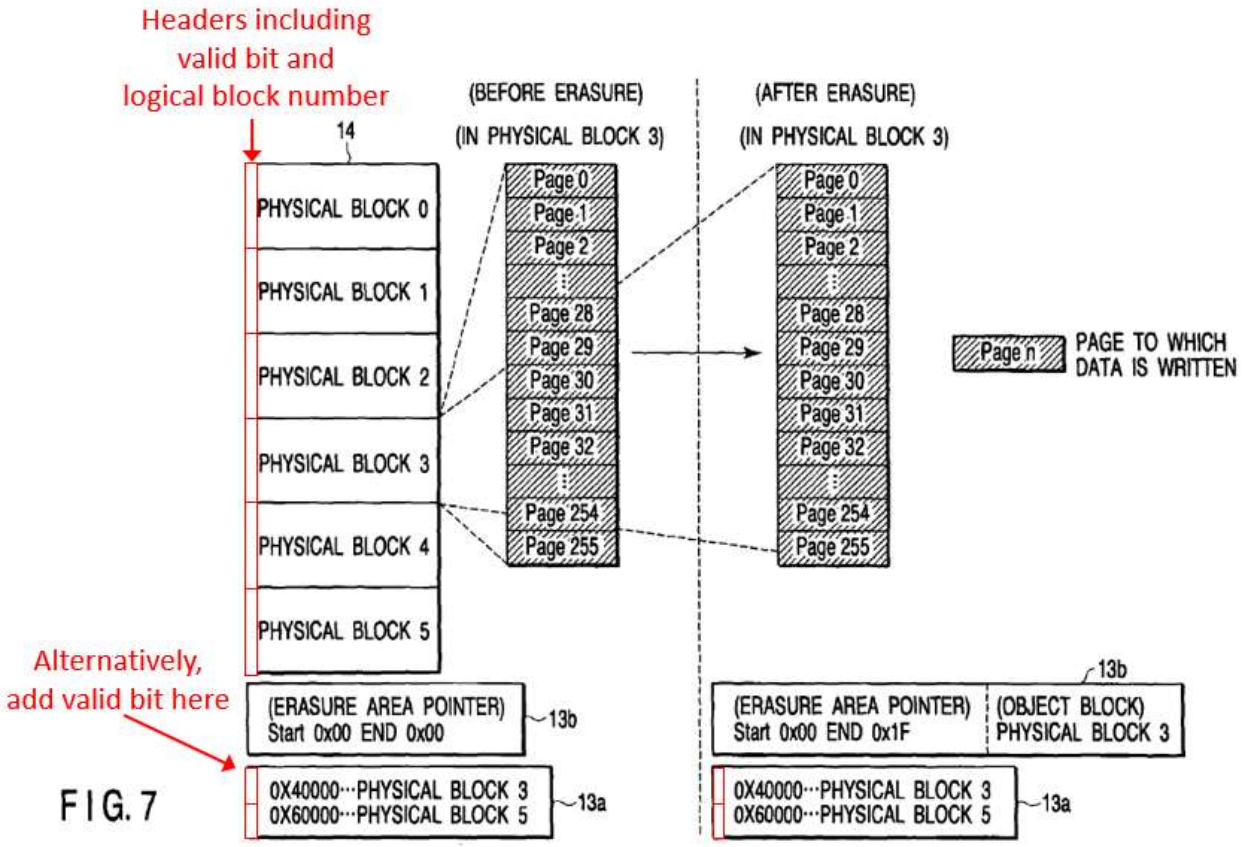


FIG. 7

Ex. 1003, Fig. 7 (annotated).

F. Claim 11

UTL asserts that the "storage recovery module" recited in claim 11 is met by "circuitry and/or software/firmware" that implements "garbage collection in response to the TRIM command." Ex. 1013, 5.

Claim 11 depends on claim 1, which Suda teaches as explained under ground 2. *See* § X.A, *supra*. Suda also teaches the storage-recovery process as described for claim 11, ground 2. *See* § X.I.a. Suda does not explicitly call its process "garbage collection," a term of art that the Patent Owner contends infringes the claimed "storage recovery." Ex. 1013, 4-5, 9-10. But even if this claim requires garbage collection, a POSITA would have still found the claim obvious in view of Suda and SwSTE'05 for two reasons.

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. Ex. 1010 § 2.3; Baker, ¶ 251. Suda's steps leading to "erasure processing" in step S6 of Fig. 8 implement a form of garbage collection. *Id*. Thus, a POSITA would have understood Suda's process to be a "garbage collection" process as accused by the Patent Owner. *Id.*

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. 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. *Id.* § 2.3; Baker, 252. Garbage collection allowed for much faster write speeds. Baker, ¶ 252. 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-1990's. *Id*. Thus, a POSTIA would have understood that SwSTE'05 teaches garbage collection and that it would have been obvious to modify Suda to include it. *Id.*, 252-53.

G. Claim 12

Claim 12 depends on claim 1, which Suda teaches as explained under ground 2. *See* § X.A, *supra*. For the reasons discussed with respect to claim 11 ground 3, a POSITA would have known that Suda's system could be modified to use a garbage collection system described in SwSTE'05. *See* § XI.F, *supra*. As part of this garbage collection process, "the reclaimed erase units are erased." Ex. 1010 § 2.3. Thus, it would have been obvious to "erase the physical storage location in response to the indication" as recited in claim 12 based on Suda and SwSTE'05. Baker, ¶¶ 255-56.

H. Claim 23

Claim 23 depends on claim 22, which Suda teaches as explained under ground

2. *See* § X.K, *supra*. To the extent that UTL argues that claim 23 is not obvious over Suda, the combination of Suda and SwSTE'05 discussed for claim 10 under ground 3 makes claim 23 obvious. *See* § XI.E, *supra*. A POSITA would have found it obvious to "invalidate an index entry configured to associate the logical identifier with the physical storage location in response to the indication" by setting a valid bit as invalid, where the valid bit is associated with a logical block address designated in an erase command. *See* § IX.E, *supra*. This valid bit would additionally indicate whether the data packet in the block is invalid. Baker, ¶ 272; *see* § X.L, *supra* (explaining "data packet"). Thus, a POSITA would have found this claim obvious in view of Suda and SwSTE'05. *Id.*, ¶¶ 271-72.

I. Claim 25

a) Element 25[a]

This claim element recites limitations indistinguishable from the limitations of claim 2[a] and would have been obvious to a POSITA for the same reasons. *See* § XI.A.a, *supra*; Baker, ¶ 276.

b) Element 25[b]

As identified in section VI, *supra*, UTL interprets "logical identifier is empty" as "data identified by the [logical identifier] in the index does not need to be preserved." Under this construction, claim 25[b] is a combination of claim 2[b] under ground 3 and claims 8 and 22[d] under ground 2. *See* §§ XI.A.b, X.F, *supra*. The "determining" in claim 25 occurs when data is read as recited in claim 8. Ex.

1013, 7 (alleging infringement of claim 25 because the determining step "refers to a read command that occurs"). Thus, for the same reasons as discussed for claim 2[b] and claim 8, a POSITA would have found this element obvious. Baker, ¶ 278.

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 fourth factor also favors institution because the Petitioners seek invalidation of claims 1-5, 8-12, and 22-26 while UTL asserted only claims 1, 4, 8-12, and 22-25 in court. Thus, claims 2-3, 5, and 26 do not overlap. The Board alone will decide these claims in IPR with no inefficient overlap with the court proceedings.

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. Exs. 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.[5] *See Globalfoundries Inc. v. UNM Rainforest*

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

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").[6]

XIV. Mandatory Notices

A. Real Parties-in-Interest

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,

[6] 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").

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 '658 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 1	
Element	**Language**
1[a]	An apparatus for managing data stored on a non-volatile storage medium, comprising:
1[b]	a non-volatile storage medium
1[c]	a request receiver module configured to receive a message comprising a logical identifier,
1[d]	the message indicating that data associated with the logical identifier has been erased,
1[e]	wherein the logical identifier is mapped to a physical storage location of the non-volatile storage medium; and
1[f]	a storage module configured to store persistent data on the non-volatile storage medium in response to the indication, wherein the persistent data is configured to indicate that the data associated with the logical identifier is erased.

Claim 2	
Element	**Language**
2[a]	The apparatus of claim 1, further comprising an index reconstruction module configured to reconstruct mappings between logical identifiers and physical storage locations of the non-volatile storage medium from contents of the non-volatile storage medium,
2[b]	wherein the reconstruction module is configured [to] indicate that data of the logical identifier are erased based on the persistent data stored on the non-volatile storage medium.

Claim 3	
Element	**Language**
3[a]	The apparatus of claim 1, further comprising: an index comprising mappings between logical identifiers and physical storage locations of the non-volatile storage medium; and
3[b]	an index reconstruction module configured to reconstruct

	the mappings using contents of the non-volatile storage medium, wherein the index reconstruction module is configured to indicate that data associated with the logical identifier are erased based on the persistent data stored on the non-volatile storage medium.

Claim 4	
Element	**Language**
4[a]	The apparatus of claim 1, further comprising: an index comprising a plurality of index entries comprising mappings between logical identifiers and physical storage locations of the non-volatile storage medium; and
4[b]	an index reconstruction module configured to reconstruct the index entries from data stored on the non-volatile storage medium, wherein the index reconstruction module is configured to record an indication that data associated with the logical identifier are erased based on the persistent data stored on the non-volatile storage medium.

Claim 5	
Element	**Language**
5	The apparatus of claim 4, wherein the index reconstruction module is configured to update the index to indicate that the data associated with the logical identifier are erased.

Claim 8	
Element	**Language**
8	The apparatus of claim 1, further comprising a read request module configured to return an indication that the data of the logical identifier are erased in response [to] a request pertaining to the logical identifier received while the data associated with the logical identifier remains on the physical storage location of the non-volatile storage medium.

Claim 9	
Element	**Language**

9	The apparatus of claim 1, further comprising a marking module configured to record that contents of the physical storage location associated with the logical identifier no longer need to be retained on a non-volatile storage medium in response to the indication.

Claim 10	
Element	**Language**
10	The apparatus of claim 1, further comprising a marking module configured to invalidate an index entry configured to associate the logical identifier with the physical storage location in response to the indication.

Claim 11	
Element	**Language**
11	The apparatus of claim 1, further comprising a storage recovery module configured to recover a storage division comprising the physical storage location in response to the indication.

Claim 12	
Element	**Language**
12	The apparatus of claim 1, further comprising an erase module configured to erase the physical storage location in response to the indication.

Claim 22	
Element	**Language**
22[a]	A non-transitory computer-readable storage medium comprising instructions configured to cause a computing device to perform a method, comprising:
22[b]	maintaining an index comprising mappings between logical identifiers and physical storage locations of a non-volatile storage device;
22[c]	receiving an indication, comprising a logical identifier, that the logical identifier mapped to a physical storage location comprising data associated with the logical

	identifier in the index is empty; and
22[d]	recording persistent data on the non-volatile storage device in response [to] the indication, wherein the persistent data is configured to indicate that the logical identifier is empty.

Claim 23	
Element	**Language**
23	The non-transitory computer-readable storage medium of claim 22, the method further comprising invalidating a data packet in response to the indication.

Claim 24	
Element	**Language**
24	The non-transitory computer-readable storage medium of claim 22, the method further comprising removing the mapping between the logical identifier and the physical storage location from the index in response to the indication.

Claim 25	
Element	**Language**
25[a]	The non-transitory computer-readable storage medium of claim 22, the method further comprising reconstructing the index based on the contents of the non-volatile storage medium,
25[b]	wherein reconstructing comprises determining that the logical identifier is empty in response to the persistent data recorded on the non-volatile storage device while the data associated with the logical identifier remains on the physical storage location.

Claim 26	
Element	**Language**
26	The non-transitory computer-readable storage medium of claim 22, further comprising responding to a request pertaining to the logical identifier with an indication the logical identifier is empty while the data associated with the logical identifier remains on the physical storage

	location.

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 12,350 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 above "**PETITION FOR *INTER PARTES* REVIEW OF CLAIMS 1-5, 8-12, and 22-26 OF U.S. PATENT NO. 8,762,658**" and Exhibits 1001-1036 by FedEx on Patent Owner at the correspondence address of record for U.S. Patent No. 8,762,658:

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

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