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. IPR2021-00345
U.S. Patent No. 9,632,727

PETITION FOR *INTER PARTES* REVIEW OF CLAIMS 1-6 AND 12-16 OF U.S. PATENT NO. 9,632,727

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

Cases

Statutes

Other Authorities

PETITIONERS' EXHIBIT LIST

Ex. No.	Brief Description
1001	U.S. Pat. No. 9,632,727 B2, titled "SYSTEMS AND METHODS FOR IDENTIFYING STORAGE RESOURCES THAT ARE NOT IN USE" to Flynn et al.
1002	U.S. Pat. No. 7,624,239 B2, titled "METHODS FOR THE MANAGEMENT OF ERASE OPERATIONS IN NON-VOLATILE MEMORIES" to Bennett et al.
1003	U.S. Pat. No. 7,057,942 B2, titled "MEMORY MANAGEMENT DEVICE AND MEMORY DEVICE" to Suda et al.
1004	Declaration of R. Jacob Baker, Ph.D., P.E., Regarding U.S. Patent No. 9,632,727.
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.

Ex. No.	Brief Description
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 C to Plaintiff's First Amended Infringement Contentions: Unification Technologies' Allegations of Infringement with Respect to U.S. Patent No. 9,632,727, *Unification Techs. LLC v. Micron Tech. Inc.*, No. 6:20-cv-500 (W.D. Tex. 2020).
1014	Eran Gal et al., *Mapping Structures for Flash Memories: Techniques and Open Problems*, PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON SOFTWARE—SCIENCE, TECHNOLOGY & ENGINEERING ("SwSTE'05") (print copy as scanned by Sylvia-Ellis Hall), Herzlia, Israel, 2005, pp. 83-92, doi: 10.1109/SWSTE.2005.14.
1015	U.S. Provisional Pat. No. 60/873,111 titled "Elemental Blade System," to Flynn et al.
1016	Docket Report for *Unification Techs. LLC v. Micron Tech. Inc.*, No. 6:20-cv-500 (W.D. Tex. 2020) (accessed Dec. 22, 2020).
1017	Frank Shu, *Notification of Deleted Data Proposal for ATA8-ACS2*, T13 (rev. 0 Apr. 21, 2007).
1018	Frank Shu & Nathan Obr, *Notification of Deleted Data Proposal for ATA8-ACS2 Revision 1*, T13 (July 26, 2007).
1019	Frank Shu & Nathan Obr, *Notification of Deleted Data Proposal for ATA8-ACS2 Revision 2*, T13 (September 5, 2007).
1020	U.S. Pat. No. 6,766,432 B2, titled "MEMORY MANAGEMENT SYSTEM SUPPORTING OBJECT DELETION IN NON-VOLATILE MEMORY" to Saltz et al.
1021	Public file history of U.S. Pat. No. 9,632,727 B2, titled "SYSTEMS AND METHODS FOR IDENTIFYING

Ex. No.	Brief Description
	STORAGE RESOURCES THAT ARE NOT IN USE" to Flynn et al.
1022	MARC record for the print digital version of the IEEE International Conference on Software--Science, Technology & Engineering Proceedings in the Linda Hall Library
1023	MARC record for the print version of the IEEE International Conference on Software--Science, Technology & Engineering Proceedings obtained from the OCLC bibliographic database.
1024	MARC record for Library of Congress.
1025	Curriculum vitae of Sylvia Hall-Ellis, Ph.D.
1026	Curriculum vitae of Jacob Baker, Ph.D., P.E.
1027	*Proceedings. IEEE International Conference on Software – Science, Technology and Engineering*, IEEE COMPUTER SOCIETY, www.computer.org/csdl/proceedings/swste/2005/12OmNC17hWm (last visited Oct. 30, 2020).
1028	Filed Stipulations of Petitioners for U.S. Patent No. 9,632,727.
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 (rev. 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).

Ex. No.	Brief Description
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 9,632,727 (the "'727 patent") should never have issued. For example, claim 1 generally recites solid-state storage drive with a controller and an indexer that (i) assign logical addresses to physical addresses; and (ii) removes logical-to-physical assignments in response to a message indicating that a logical address is erased. In the related litigations the Patent Owner, Unification Technologies LLC ("UTL") generally asserts the claims encompass receiving a message indicating that data in storage need not be preserved because it has been erased from a user's perspective, e.g., by a computer attached to storage. *See* § VI, *infra*. A person of ordinary skill in the art ("POSITA") would have known of 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 '727 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 '727 Patent

The '727 patent application was filed on June 19, 2014. Ex. 1001, cover. The

Examiner rejected the claims on various grounds, but did not cite the references relied upon herein. *Id.*, pp. 1-2 (listing cited references); Ex. 1021, 273-84. To overcome the rejections, claim 31 (issued claim 1) was amended to recite that the indexer is "comprised within the solid-state storage controller" and the message is "received from a host operating system." Ex. 1021, 316.

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 '727 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 '727 Patent

The '727 patent acknowledges that erase commands for file systems were known. *See, e.g.,* Ex. 1001, 1:33-35 ("In many file systems, an erase command deletes a directory entry in the file system while leaving the data in place in the storage device containing the data.").

Similarly, erasing data by overwriting with zeros, ones, or other null characters was also known. Ex. 1001, 1:37-39. The patent alleges, however, that these erase methods were "inefficient" because "valuable bandwidth is used while transmitting the data [that] is being overwritten" and "space in the storage device is taken up by the data used to overwrite invalid data." *Id.*, 1:39-42.

A. Effective Filing Date and Date of Invention

The '727 patent claims priority to provisional application no. 60/873111, 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 '727 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, ¶ 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
"indexer"	1-4	Indefinite under 112(f) for lack of structure and/or algorithm	Not indefinite and no construction is needed in light of the surrounding claim language.
"empty-block directive command"	12, 15	a command to empty data from a block	a command that indicates that certain blocks contain data that does not need to be preserved

Claim Term	Claim Nos.	Petitioners	UTL
garbage collector	5	Plain and ordinary meaning	hardware and/or software for recovering space for a system that does not support update-in-place of data in the storage medium
storage client	12	Plain and ordinary meaning	a computing system capable of being coupled to the non-volatile storage medium
persistent data	12	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
host operating system	1	Plain and ordinary meaning	operating system of the computing system that is capable of being coupled to the non-volatile storage medium
the identified logical address is erased	1	Plain and ordinary meaning, where the plain and ordinary meaning is that the identified logical address is erased, not the data associated with the identified logical address is erased	the data identified by the logical address does not need to be preserved
logical-to-physical	12	Indefinite under	Not indefinite and not

Claim Term	Claim Nos.	Petitioners	UTL
translation layer		112(f) for lack of structure and/or algorithm	subject to 112(f).
storage processor is configured to … update the logical-to-physical translation layer to indicate that data stored in physical block addresses corresponding to the received logical block addresses do not need to be preserved, and store persistent data on the flash memory device, the persistent data indicating that the data corresponding to the received logical block addresses is deleted at the storage client.	12	Indefinite under 112(f) for lack of structure and/or algorithm	Not indefinite and not subject to 112(f).

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.").

In infringement contentions, UTL provides examples that allegedly infringe certain claim elements. Ex. 1013. UTL contends that "**the indexer**" in claim 1 includes "circuitry, software, and/or firmware configured to assign LBAs of the logical address space to physical addresses on the NAND flash memory" and contains "a map or index of LBAs with their corresponding physical addresses. *Id.*, 3. Although not offering a construction of "**index entries**," in claim 2, UTL contends, "Logical block addresses and physical addresses are both forms of index entries." *Id*. Although not offering a construction of "**index metadata**" in claim 3, UTL contends, "Logical block addresses and physical addresses are both forms of index metadata." *Id.*, 4. Although not offering an affirmative construction of **"logical-to-physical translation layer**," in claim 12, UTL contends, "a logical-to-

physical translation layer [is] often referred to as the FTL of the NAND flash memory. The FTL is a structure/set of functions for mapping LBAs to Physical Blocks. … [T]ables are widely used in order to map sectors and pages from logical to physical (Flash Translation Layer or FTL)." Ex. 1013, 6.

VII. Precise Relief Requested

A. Proposed Grounds

a) Ground 1

Claims 1-6 and 12-16 are invalid under 35 U.S.C. § 103 over 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 1-3, 5-6, and 12-16 are invalid under 35 U.S.C. § 103 over Suda (Ex.

1003) in view of a POSITA's knowledge.

c) Ground 3

Claims 4 and 13 are invalid under 35 U.S.C. § 103 over Suda in view of Bennett and a POSTIA's knowledge.

B. Qualifying Prior Art

Bennett and Suda are prior art to the '727 patent. Petitioners are unaware of any assertion that the '727 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.

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, according to the applicant, the art of record during prosecution did not show "an indexer, comprised within the solid-state storage controller," because the "FAT table … is stored by the host." Ex. 1021, 321. The Examiner found this argument persuasive. *Id.*, 333. But, as shown herein, references such as Bennett and Suda teach this element along with all of the other elements of the claims.

VIII. The Prior Art

A. Summary of Bennett

Like the '727 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 would be "logically erased" by "the system's standard, logical erase method." *Id.*

Bennett recognizes that "logical erasing" was not inventive and "it was common" for advanced memory systems to erase data logically, with the actual erasure taking place at a later time. *Id.*, 3:26-32. For a logical erasure, the memory system will write a specific "data pattern to the memory portion, set a flag, or otherwise designate it as erased." *Id.*, 3:36-41. The logically erased "portion can then be physically erased when convenient, for example in a background process" such as a garbage collection process. *Id.*, 3:39-41; *compare* Ex. 1001, 51:33-35 ("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, ¶ 99. The GAT is stored in non-volatile flash memory as highlighted in Bennett's Fig. 6 below:

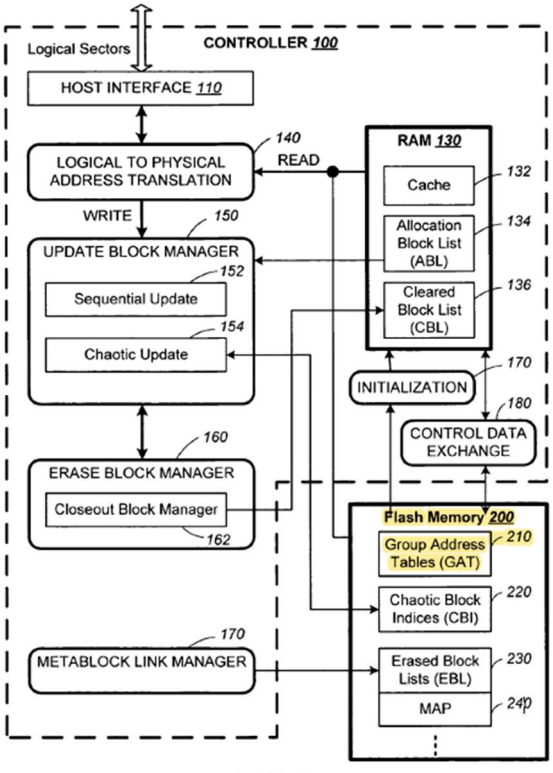


FIG. 6

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:

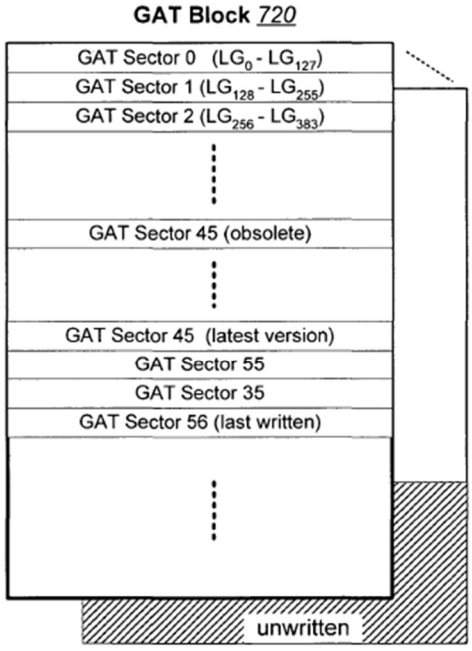


FIG. 8B

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.

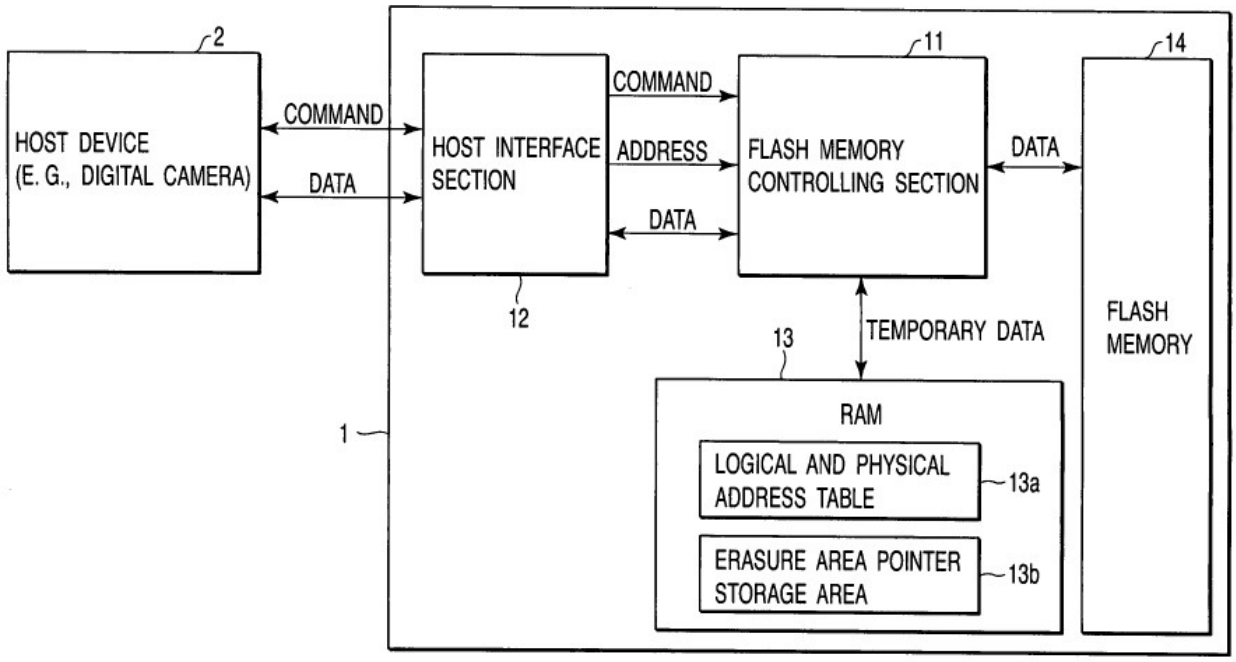


FIG. 1

Id., Fig. 1.

Like the '727 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).

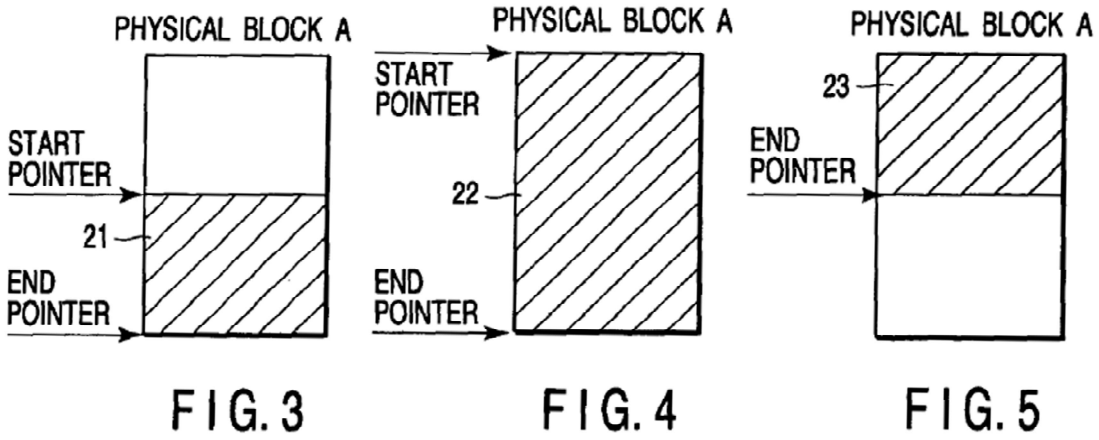


FIG. 3 FIG. 4 FIG. 5

Id., Figs. 3, 4, 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.

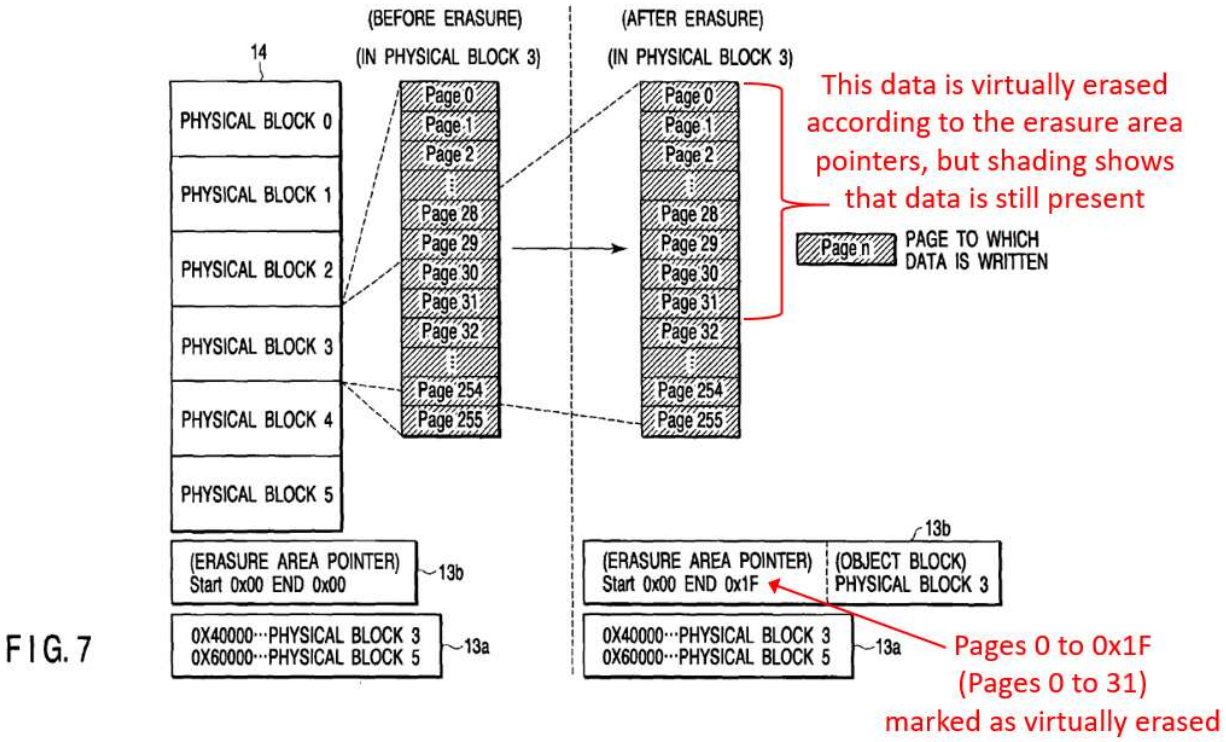


FIG. 7

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.

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-

[1] *See* attached Claim Listing.

Volatile Memory." Ex. 1002, Title. Bennett discloses a host and memory system. *See* Ex. 1002, Fig. 2.

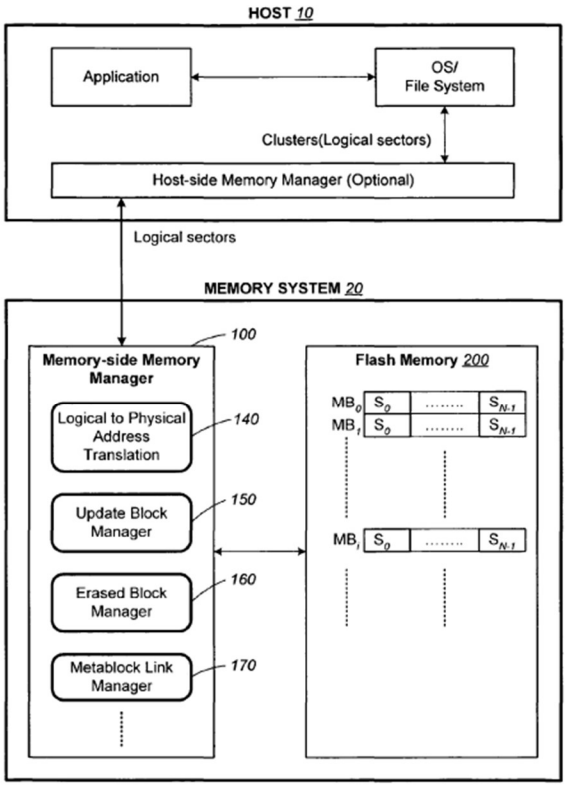


FIG. 2

Id., Fig. 2. Bennett discloses "Flash Memory 200." Ex. 1002, Fig. 2. "[F]lash memory is non-volatile" solid state memory. *Id.*, 1:29-30, 2:38-40 ("There are many commercially successful non-volatile solid-state memory devices being used today. These memory devices may be flash EEPROM or may employ other types of nonvolatile memory cells."). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 117-18.

b) Element 1[b]

Bennett teaches a solid-state storage medium. For example, Bennett's Fig. 2

(copied above in the prior element) discloses a "memory system 20" that includes a "flash memory 200." Ex. 1002, Fig. 2. Bennett also teaches that the "memory system is typically in the form of a memory card or an embedded memory system." *Id.*, 6:60-62. A POSITA would understand that a "memory card" is a solid-state storage medium. Baker, ¶ 119; *see also* Ex. 1002, 1:23-26 ("Solid-state memory capable of nonvolatile storage of charge, particularly in the form of EEPROM and flash EEPROM packaged as a small form factor card, has recently become the storage of choice), Fig. 6 (depicting Flash Memory 200).

c) Element 1[c]

Bennett teaches a "memory system 20 [that] includes a memory 200 whose operations are controlled by a controller 100." Ex. 1002, 6:62-63. The "controller 100 includes an interface 110" and the interface "has one component interfacing the controller to a host and another component interfacing to the memory 200." *Id.*, 6:66-7:4. A POSITA would understand that Bennett's controller is a "solid-state controller" as claimed because it implements storage operations on the flash memory 200. Baker, ¶ 120. For example, the controller includes a "memory-side memory manager" (*id.*, Fig. 2) which "contains a number of software modules for managing erase, read and write operations of the metablocks[2] … [and] maintains

[2] Bennett's "FIG. 2 illustrates the memory being organized into physical groups of

system control and directory data associated with its operations among the flash memory 200 and the controller RAM 130." *Id.*, 7:36-41. Bennett also teaches storing data pertaining to a computer system's logical address space at respective physical address of the flash memory. *Id.*, 7:29-36, Figs. 3A-3B. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 120-22.

d) Element 1[d]

A POSITA would understand that Bennett teaches an indexer as claimed under UTL's interpretation (*see* § VI, *supra*). For example, UTL appears to interpret this term to mean "circuitry, software, and/or firmware configured to assign LBAs[3] of the logical address space to physical addresses on the NAND flash memory" and contain "a map or index of LBAs with their corresponding physical addresses." *See* § VI, *supra.* While Bennett does not use the term "indexer", a POSITA would understand that Bennett teaches the same thing.

For example, Bennett teaches use of a "memory-side memory manager" comprised within the storage controller. Ex. 1002, Fig. 2. The memory-side memory manager includes components such as a "logical to physical address translation" module. *Id.* This module "is responsible for relating a host's logical

---

sectors (or metablocks) and managed by a memory manager of the controller, according to a preferred embodiment of the invention." Ex. 1002, 5:15-18.

[3] "Logical Block Addresses."

address to a corresponding physical address in flash memory." Ex. 1002, 10:37-39. Bennett further teaches storing an index mapping of logical-to-physical addresses in a GAT stored in flash memory. *Id.*, Figs. 6, 8B; Baker, ¶ 123. POSITA would have understood this logical to physical address translation module to include "circuitry, software, and/or firmware". *Id*. A POSITA would understand that physical addresses are used to identify blocks that stored data on the flash memory. *Id.* Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 123-24.

e) Element 1[e]

A POSITA would understand that Bennett teaches this claim element. First: Bennett teaches the claimed "message received from a host operating system … indicating that the logical address is erased" in the form of erase commands. Ex. 1002, 5:56-58 ("an erase command, originating either from the host or with the memory system itself"). Bennett's 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"). A "logical sector" refers to a logical address. *See* Ex. 1002, 7:22-24 ("[t]ypically, the host system addresses data in units of logical sectors where, for example, each sector may contain 512 bytes of data"); Baker, ¶ 126. As depicted in Bennett's Fig. 2, the erase command is received from the host through the controller's memory-side memory manager.

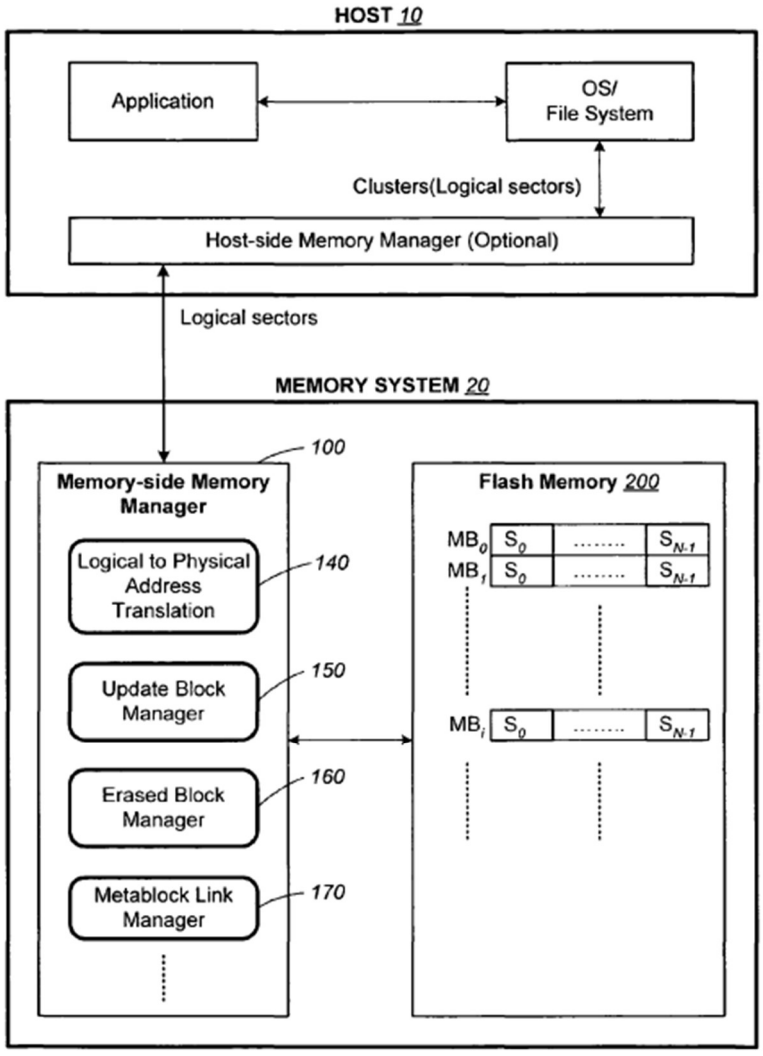
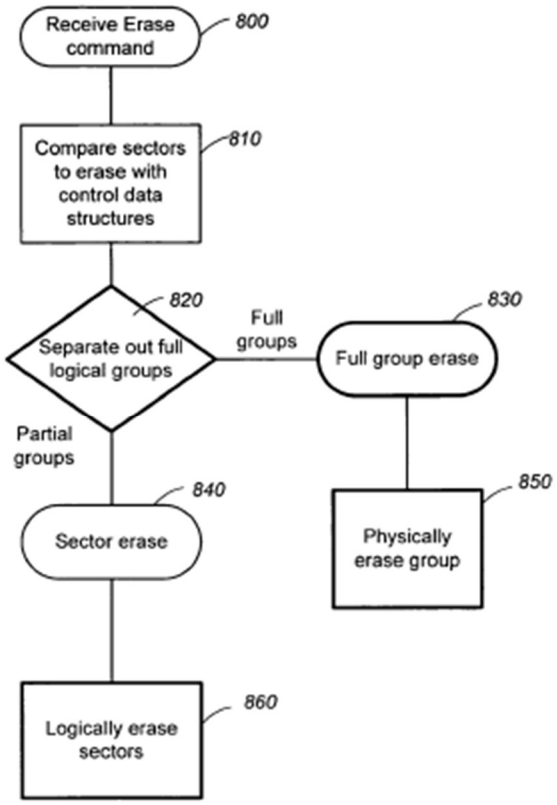


FIG. 2

Ex. 1002, Fig. 2.

Bennett's erase command indicates that a logical address is erased under UTL's proposed construction of "the identified logical address is erased" as "the data identified by the logical address does not need to be preserved". *See* § VI, *supra.* For example, Bennett's Fig. 10 illustrates the process flow for erase commands:



Ex. 1002, Fig. 10. As shown, the memory controller (at step 820) separates erase commands for "partial groups" and "full groups." *Id.* Full groups "can then be physically erased" (step 850) and the partial groups "are logically erased" (step 860). *Id.*, Fig. 10, 4:12-21. The "logically erased" data "can then be physically erased when convenient, for example in a background process." *Id.*, 3:39-41. As such, a POSITA would understand that an erase command (that results in a logical or actual erase) indicates that the data need not be preserved, ie., how UTL construes the term "the identified logical address is erased". Baker, ¶ 129.

Alternatively, a POSITA would also understand that Bennett renders obvious receiving a message indicating that a logical address is erased under Petitioners'

proposed construction of "the identified logical address is erased" as "plain and ordinary meaning[:] that the identified logical address is erased, not the data associated with the identified logical address is erased". *See* § VI, *supra*. Specifically, a POSITA would have understood that Bennett's erase command could be sent by an "OS/file system" in a host, for example, in response to user interactions with an "application" to delete a document. Baker, ¶¶ 130-33; Ex. 1002, Fig. 2 (depicting components). The POSITA would have also understood that the OS/file system keeps a mapping of file names (e.g., document.doc) and the associated logical addresses (e.g., 0x4000). *Id.* The OS/file system uses this mapping to set the correct logical address in the erase command in response to the user deletion of a file. *Id.*; Ex. 1002, 17:52-56. While Bennett does not explicitly disclose a host deleting a logical address in the mapping of an OS/file system, a POSITA would have known that hosts could delete logical addresses. Baker, ¶ 133. A POSITA would also understand that it would be a simple obvious matter to modify Bennett's erase command to indicate the address deletion. *Id.* Such a modification would be motivated in order to effectuate the same purpose as Bennett's disclosed erase commands: to initiate removal of obsolete data in storage. *Id.* Bennett itself teaches that the process of deleting obsolete data in storage is common:

> When the data in a portion of the memory becomes obsolete, or the memory receives a command to erase a particular portion, in more advanced memory systems it is common for the designated portions

> not to be erased immediately at that time, but to be "logically erased" by being marked for erase, with the actual, physical erase taking place at a later time.

Ex. 1002, 3:25-32.

Second, a POSITA would also understand that Bennett also teaches that the "indexer is further configured to remove an assignment between an identified logical address and a physical address of the solid-state storage medium" as claimed in response to receiving an erase command. Specifically, as identified in claim 1[d], Bennett discloses an indexer in the form of a memory-side memory manager that "is responsible for relating a host's logical address to a corresponding physical address in flash memory." As part of this process, Bennett teaches storing logical-to-physical address tables in a GAT. *See* Ex. 1002, 10:19-21, 10:65-11:60, Figs. 6, 8A-8B.

For physical erasures: Bennett teaches removing the assignment by writing FFs or 00s to the sector and setting an erased flag. *Id.*, 20:20-27. This process removes the assignment and results in a logical address that can be re-mapped to a new physical block address. *See id.* (the "logical group can again be associated with an MS block"); Baker, ¶ 137 ("MS block" refers to a physical block address).

Similarly, **for logical erasures:** Bennett teaches removing the assignment by setting an erased flag. *Id.*, 20:47-49. In this scenario, the previously assigned

physical address is no longer used. *Id.*, 20:47-50 ("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."); Baker, ¶ 138.

To the extent that UTL argues that Bennett's logical and physical erase disclosures are not "remov[ing] an assignment between an identified logical address and a physical address" as claimed, a POSITA would understand this limitation is obvious. Baker, ¶¶ 134-40. For example, Bennett teaches that "sectors are 'logically' erased at the sector level by standard techniques." Ex. 1002, Abstract; *see also id.*, 6:18-20 ("For the specified sectors not forming complete block, the system uses the system's standard, logical erase method."). A POSITA would understand that one known technique for recording a logical erase is "removing the assignment between an identified logical address and a physical address." Baker, ¶ 139 (citing examples of a POSITA's knowledge). For example, the Suda prior art discussed herein discloses "canceling the relation between the logical block addresses and the physical addresses" and to "erase[] address information of the physical block [] and a logical block address." *See* § X.A.e.

Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 125-40.

B. Claim 2

a) Element 2[a]

Bennett teaches that its memory-side memory manager (the claimed indexer) includes 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. The manager performs this mapping through use of index entries, e.g., in a GAT. *See id.*, Fig. 8B (depicting an index mapping GAT sectors to logical addresses). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 141-42 (Bennett teaches multiple addresses and each index entry maps a logical address to a physical address); *see also* Bennett's Figs. 2, 3A, 4, 8A, 8B (disclosing address mapping indexes with a plurality of index entries).

b) Element 2[b]

A POSITA would understand that Bennett teaches this element. 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 "remov[ing] an index entry corresponding to the identified logical address" as claimed. Baker, ¶ 143 (citing examples of a POSITA's knowledge). For example, the Suda prior art discussed herein discloses removing the index mapping by "canceling the relation between the logical block addresses

and the physical addresses." *See* §§ X.A.e, *infra* (Ground 2). A POSITA would have been motivated to modify Bennett's teaching of setting a flag to remove the assignment between logical to physical address with this known technique as a simple matter of design choice and no more than a substitution of one of a limited number of potential ways to indicate an erasure that is known in the field. Baker, ¶ 143. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 143-44.

C. Claim 3

In the related litigations, UTL alleges that "Logical block addresses and physical addresses are both forms of index metadata. ... [T]he logical block address [LBA] for that data – the location associated with it – is called metadata, which literally means data about data."). Ex. 1013, 4. Bennett similarly teaches index metadata. *See, e.g.,* Ex. 1002, Fig. 3B (mapping logical-to-physical addresses). Bennett further discloses an indexer (the memory-side memory manager) with a "logical to physical address translation module" for assigning logical and physical addresses. *See, e.g., id.*, Fig. 6, 9:50-52, 10:36-38, Fig. 8B (depicting a GAT index mapping physical sectors to logical addresses).

Bennett also teaches using a list of physical block addresses when making new logical-to-physical assignments. For example, Bennett keeps, in a RAM of the controller, a cleared block list (CBL). *Id.*, Fig. 6. The CBL contains a list of

physical block addresses that have been erased and are thus available for storing new data. *Id.*, 10:27-33. A POSITA would have understood that Bennett's system picks from, or uses, a physical block addresses from the CBL when assigning a physical block address to a new logical address. *Id.*, Fig. 6 (showing RAM data being read to inform the logical to physical address translation mapping); Baker, ¶ 145. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 145-46.

D. Claim 4

Bennett teaches "[f]irmware stored in nonvolatile ROM 122 and/or the optional nonvolatile memory 124 provides codes for the processor 120 to implement the functions of the controller 100." Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶ 147.

E. Claim 5

In the related litigations, UTL alleges that this element is met through a "garbage collection" process. Ex. 1013, 4-5. Bennett teaches garbage collection. *See* Ex. 1002, 19:10, 20:32. Indeed, "garbage collection" was a well-known process for erasing data in a background process at a convenient time. Baker, ¶¶ 75-77, 149; *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"). More specifically, a POSITA would have understood that Bennett discloses a garbage collector under either proposed construction. UTL proposes that the term "garbage collector" is construed as "hardware and/or software for recovering space for a system that does not support update-in-place of data in the storage medium." *See* § VI, *supra.* Bennett discloses the same thing. Baker, ¶ 149. For example, Bennett compares its solutions against other systems that support update-in-place. Ex. 1002, 3:12-17 ("one way is rewrite the update data in the same physical memory location … This method of update is inefficient").

A POSITA would also have found it obvious to determine whether to perform garbage collection in response to the message (an erase command) as claimed. For example, a POSTIA would have found it obvious to determine whether to initiate garbage collection in response to an erase command to effectuate Bennett's goal of erasure of a "substantial size" of data. *Id.*, 3:5-9; Baker, ¶ 150. A POSITA would recognize that there are a limited number of potential options as to when a garbage collection process should be initiated, and thus initiating a garbage collection process in response to an erase command of significant size would have been obvious. Baker, ¶ 150. Indeed, Bennett explicitly teaches that "it is desirable to have the erase block of substantial size … [to] amortize [the erase time] over a large aggregate of memory cells." Ex. 1002, 3:7-9.

F. Claim 6

A POSITA would have understood that Bennett teaches a "bus interface" to couple Bennett's memory system 20 (the claimed "solid state storage controller") to the host 10 (the claimed "computer system"). Ex. 1002, Fig. 1; Baker, ¶ 152.

Bennett does not expressly disclose the types of bus interfaces that may be supported. Even so, the claimed bus interfaces would have been obvious to a POSITA because they merely include industry standards that were well-known to a POSITA at the time. Baker, ¶ 153. For example, a POSITA would have known of the industry standard SATA interface. *Id.*; Ex. 1007 (SATA 2.5 Standard). Even the inventors admit that SCSI and ATA standards (which include SATA) were "ubiquitous." Ex. 1015, 92 (stating that object storage methodologies "are not ubiquitous like block device protocols (such as SCSI and ATA)"). The SATA standard enjoyed widespread adoption throughout the computer industry. Ex. 1007 at 1 (listing major companies as board members, including Dell, Hewlett-Packard, Hitachi, Intel, Maxtor, Seagate, and Vitesse); Baker, ¶ 153. The SATA standard was obvious because it defined a high-speed interface to ease integration and enable scalable performance with data rates of 1.5 Gbps and 3.0 Gbps. Ex. 1007 at 19. Flash drives commonly implemented this SATA interface to achieve these high speeds and to enjoy widespread compatibility across the industry. Baker, ¶ 153. Thus, it would have been obvious to a POSITA for Bennett's host interface section

to be a common interface such as SATA. *Id.*, ¶¶ 152-54.

G. Claim 12

a) Element 12[a]

This claim element recites limitations indistinguishable from the limitations of claim 1[a] and would have been obvious to a POSITA for the same reasons. *See* § IX.A.a, *supra*; Baker, ¶¶ 155-58.

b) Element 12[b]

Bennett teaches a storage interface with a storage client. *See* Ex. 1002, Fig. 1 (depicting host coupled to memory system 20). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶ 159; *see also* discussion for claim 6, *supra*.

c) Element 12[c]

Bennett teaches a "controller 100 includes an interface 110." The controller 100 is the claimed storage processor. Baker, ¶ 161. The controller includes an "interface 110" that "has one component interfacing the controller to a host and another component interfacing to the memory 200." Ex. 1002, 6:66-7:4, Figs. 2 and 6. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 160-61.

d) Element 12[d]

Bennett teaches a flash memory device coupled to the storage processor. *See* Ex. 1002, Figs. 2 and 6 (depicting "Flash Memory 200" coupled to controller 100).

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

e) Element 12[e]

According to UTL, the logical-to-physical translation layer is "often referred to as the FTL[4] of the NAND flash memory. The FTL is a structure/set of functions for mapping LBAs[5] to Physical Blocks. … [T]ables are widely used in order to map sectors and pages from logical to physical (Flash Translation Layer or FTL)." Ex. 1013, 6. A POSITA would have understood that Bennett teaches a logical-to-physical translation layer under UTL's example. Baker, ¶¶ 163-64.

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. The mapping is stored, e.g., in the GAT. *See id.*, 10:19-21, 10:65-11:60, Figs. 6, 8A-8B. A POSITA would understand that the GAT is a logical to physical translation layer under UTL's example. Baker, ¶¶ 163-64.

Bennett further teaches that the GAT is maintained by the storage processor as claimed. *See* Ex. 1002, Fig. 2 (controller's memory-side memory manager includes a "logical to physical translation" module coupled to flash memory 200), 10:37-39 (logical to physical translation module "is responsible for relating a host's

[4] "Flash Translation Layer."
[5] "Logical Block Addresses."

logical address to a corresponding physical address in flash memory"). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 163-65.

f) Element 12[f]

Bennett's controller is configured to receive, from a host through the host interface, an Erase Sectors command that includes a range of logical sectors (logical block addresses). Ex. 1002, Figs. 1-2, 17:52-56; Baker, ¶¶ 166-67.

Bennet's Erase Sectors command meets UTL's proposed construction of "empty block directive command" of "a command that indicates that certain blocks contain data that does not need to be preserved." *See* § VI*, supra.* For example, Bennett teaches treating erase commands for "partial groups" as a "logical" erase command. *See* § IX.A.e (explaining logical erasures of partial groups). Such logical erase commands indicate that certain blocks contain data that "can then be physically erased when convenient, for example in a background process." *Id.*; Ex. 1002, 3:39-41. Thus, a POSITA would have understood that Bennett teaches logical erase commands that indicate certain blocks contain data that do not need to be preserved. *Id.*; Baker, ¶ 166.

Bennet's Erase Sectors command also teaches Petitioners' proposed construction of "empty block directive command" as "a command to empty data from a block." *See* § VI*, supra*. Specifically, Bennett teaches treating erase

commands for full logical groups as commands to physically erase the data from that full logical group. *See* § IX.A.e (explaining physical erasures of full logical groups). The data will be "physically erased or otherwise subjected as a whole to an erase operation." Ex. 1002, 4:12-18. Thus, a POSITA would have understood that Bennett teaches an erase command to empty data from a block by physically erasing the data from the blocks in that full logical group. Baker, ¶ 167.

g) Element 12[g]

As discussed for claim 12[e], Bennett teaches the logical to physical translation layer as a GAT that maps logical to physical addresses. *See* IX.G.e, *supra*.

Bennett further teaches updating the GAT to indicate that the data stored in physical block addresses corresponding to the received logical block addresses do not need to be preserved. Specifically, Bennett teaches setting flags in the GAT in response to receiving erase commands. For example, for logical erasures, 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. A POSITA would understand that because these flags are sent in response to receiving an erase command, they indicate the data stored in physical

block addresses corresponding to the received logical block addresses do not need to be preserved as claimed. Baker, ¶¶ 169-70.

h) Element 12[h]

Bennett teaches storing persistent data on the flash memory device under either construction of "persistent data." *See* § VI, *supra* (construing term). Bennett teaches setting flags in a GAT. *See* § IX.G.g. The GAT is stored in flash memory. *See* Ex. 1002, Fig. 6 (GAT 210 is part of flash memory 200). Flash memory is nonvolatile; thus, any data stored in flash is "persistent data" under either proposed construction. *Id*., 1:29-30; Baker, ¶ 172. Because the flags are set in response to receiving an erase command from a host (*see* § IX.A.e) a POSITA would understand that they indicate that the data (stored at the physical block addresses) corresponding to the received logical block addresses is deleted at the storage client. Baker, ¶¶ 172-73.

Indeed, in the related litigations, UTL alleges that the Trim command infringes this element because it "tells the SSD that specific areas contain data that is no longer in use. From the user's perspective, this data has been deleted from a document." Ex. 1013, 7. A POSITA would understand that Bennett teaches the same thing. Specifically, in Bennett's system, users may use an "Application" in a "HOST" (the claimed "storage client") to delete data such as documents, which causes an erase command to be sent by the "OS/File System." Ex. 1002, Fig. 1. As

discussed in the preceding paragraph, Bennett's memory system responds to the erasure of the document by setting flags. As a result, the documents appear deleted from a user's perspective. Baker, ¶ 172. In fact, Bennett's flags make the deleted data or document appear deleted to the user because, in response to future reads of this data, Bennet's system will return only all ones or all zeros if the user tries to read this data. Ex. 1002, 20:45-50. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 172-73.

H. Claim 13

Bennett discloses that the GAT (the claimed logical-to-physical translation layer) is stored in non-volatile flash memory that maps logical-to-physical addresses. Ex. 1002, Fig. 6, 10:17-21, 10:65-11:60. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 174-75.

I. Claim 14

Bennett's Fig. 6 depicts a volatile memory device (RAM) coupled to the controller. Ex. 1002, Fig. 6; Baker, ¶ 62 (RAM is volatile memory). Bennett teaches that the RAM "acts as a cache for control data stored in flash memory 200." Ex. 1002, 10:33-34. Included in this RAM is a "GAT cache [that] is a copy … of entries in a subdivision of the 128 entries in a GAT sector." *Id.*, 12:12-13. A POSITA would understand that that the "GAT sector" being referred to is the GAT (stored in flash memory) that was previously identified as the claimed logical-to-physical

translation layer. *See* § IX.G.e; Baker, ¶¶ 176-78. As such, a POSITA would understand that the GAT cache is also logical-to-physical translation layer stored in volatile memory as claimed. *Id.*

J. Claim 15

Bennett teaches responding to read commands for data marked as "logically erased" by returning a predetermined data string. For example, Bennett discloses:

> [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.

Ex. 1002, 20:45-50. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 179-80.

K. Claim 16

In the related litigations, UTL alleges that a "string of ones or zeros are data bits that have a uniform logic level" as claimed. Ex. 1013, 8. Bennett teaches this. *See* discussion for claim 15, *supra* (sending host "FFs or 00s as if the sectors were erased"). In additional, a POSITA would have generally understood all of Bennett's references to "erasing" flash memory to mean setting all values to ones. Baker, ¶ 181. Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶¶ 181-82.

X. Ground 2: Obvious Over Suda and POSITA Knowledge

A. Claim 1

a) Element 1[a][6]

If the preamble is a claim limitation, Suda teaches a non-volatile solid-state storage system. Ex. 1003, Fig. 1, 2:59-65; Baker ¶ 184. The following illustration annotates Figure 1 of Suda to show various claim elements, including this one:

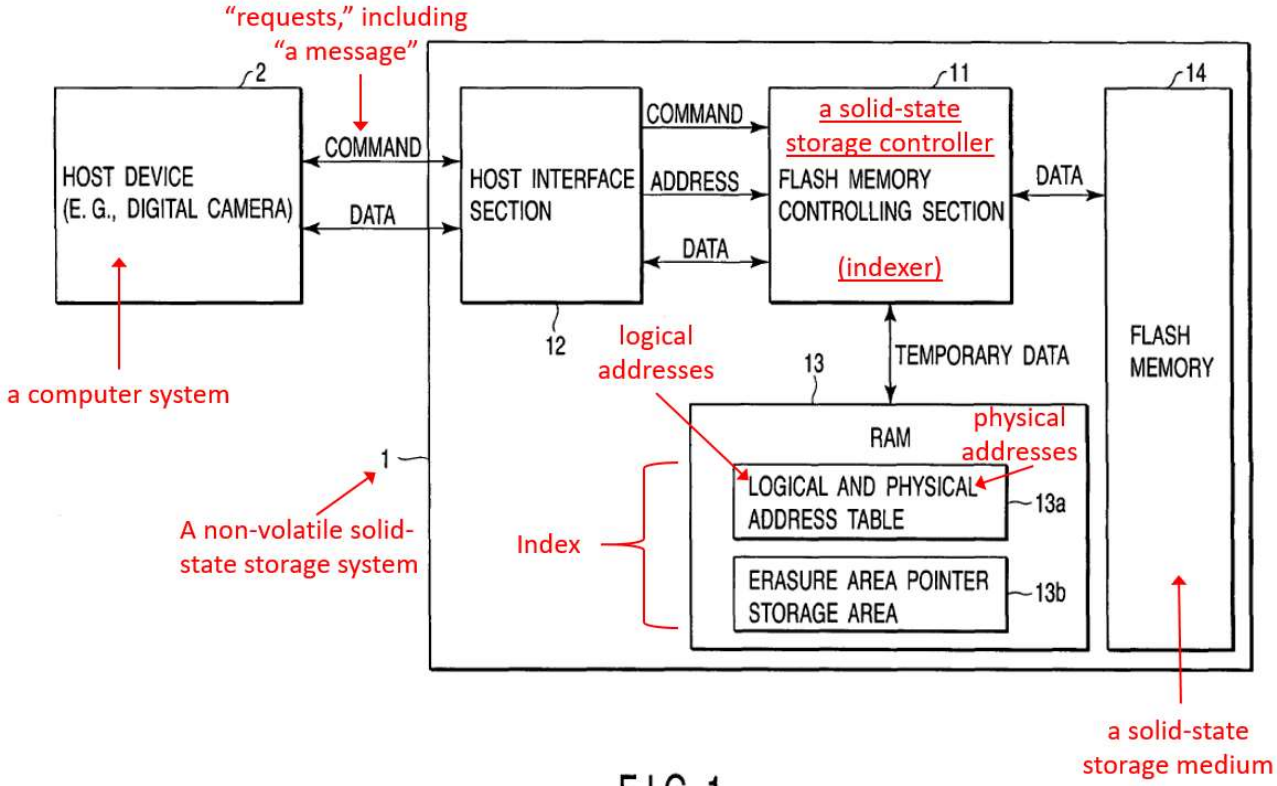


FIG. 1

Ex. 1003, Fig. 1 (annotated).

b) Element 1[b]

Suda teaches that the memory card includes a solid-state storage medium in the form of nonvolatile flash memory. Ex. 1003, Fig. 1 (annotated above), 2:63-66;

[6] *See* attached Claim Listing.

Baker ¶ 185.

c) Element 1[c]

Suda shows that the memory card has the recited "solid state storage controller" in the form of a "flash memory controlling section." Ex. 1003, Fig. 1, 2:63-65. This flash memory controlling section "operates based on the command information" ("in response to requests," as recited) that is "issued from the host device" ("from a computer system," as recited). *Id.*, 2:66-3:15, Fig. 1. The flash memory controlling section implements the recited "storage operations" including reading (*e.g.*, *id.*, 2:41-43), writing (*e.g.*, *id.*, Fig. 11, step B6), and erasing (*e.g.*, *id.*, 8:66-9:3). Thus, a POSITA would have understood Suda to teach "a solid state controller configured to implement storage operations on the solid state storage medium in response to requests from a computer system." Baker, ¶ 186.

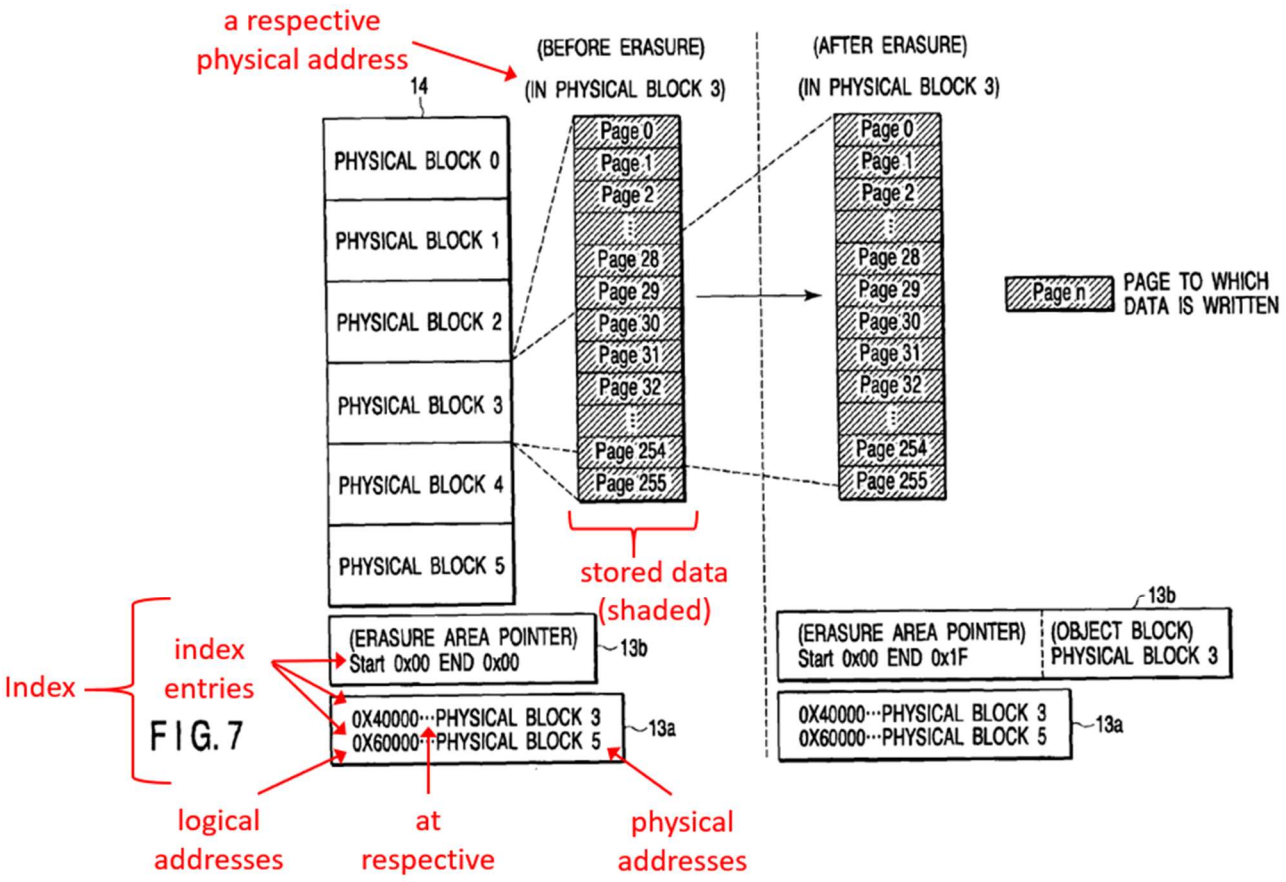
Suda also teaches that its flash memory controlling section stores data pertaining to logical addresses of a logical address space at respective physical addresses of the solid state storage medium, as claimed. Baker, ¶ 187. Using Fig. 2, Suda discusses a case of storing data, "where successive 256-Kbytes data items are written to two physical blocks in the memory card." *Id.*, 3:24-27. Half of the data is written to physical block 3, and the other 128 Kbytes of data items are written to physical block 5. *Id.*, 3:36-40. Logical addresses are associated with the data at each of these physical addresses. *Id.*, 3:43-55, Fig. 2 (table 13a), Fig. 7 (annotated

at § X.A.d, *infra*); Baker, ¶ 187. Thus, a POSITA would have understood Suda to teach this element. Baker, ¶¶ 186-88.

d) Element 1[d]

A POSITA would have understood the claimed "indexer" to be the circuitry, software, and/or firmware in Suda's "flash memory controlling section 11 [that] manages data erasure and a table indicating a relationship between logical blocks and physical blocks of the flash memory 14." Ex. 1003, 3:13-15; Baker, ¶ 189; *see* § VI, *supra* (construing indexer and citing UTL's example of "circuitry, software, and/or firmware" in Ex. 1013, 3).

Suda provides various examples of the flash memory controlling section assigning logical addresses to physical addresses. Ex. 1003, 3:33-40. The assigned physical addresses are used to identify empty blocks (*id.*, 3:64-67) or partially empty, partially used blocks "in which data items are written," (*id.*, 3:41-47, Fig. 2 (showing partially empty block)). The assigned logical-to-physical address mappings are stored in Suda's logical and physical address table. *E.g., id.*, 3:43-55, Figs. 1, 7 (annotated below). Thus, a POSITA would have understood Suda to teach an indexer configured to assign logical addresses of the logical address space to physical addresses in use to store data pertaining to the logical addresses on the solid-state storage medium. Baker, ¶¶ 190-91.



Ex. 1003, Fig. 7 (annotated).

e) Element 1[e]

Suda teaches that the flash memory controlling section responds to a message from the host operating system indicating that the identified logical address is erased. For example, when receiving an erase command specifying logical block address 0x4000, Suda discloses, "[w]hen an erase command is issued from the host device 2, the flash memory controlling section 11 refers to the logical-to-physical conversion table 13a, and detects physical block address '3' related to the logical block address '0x40000' designated in the erase command." Ex. 1003, 8:66-9:3, Fig. 7 (annotated above to show the related addresses). The message is sent by a

host device, such as a digital camera, which a POSITA would have known to be running an operating system. *Id.*, Fig. 1 (showing components), Fig. 8 (step S1); Baker, ¶ 192. Thus, the indexer of Suda's flash memory controlling section operates "in response to a message received from a host operating system," as claimed. Baker, ¶ 192.

In the following two examples where entire blocks are erased, Suda also teaches that the flash memory controller section is configured to remove an assignment between an identified logical address and a physical address in response to the erase command. In a first example, an erase command for an entire block is received, causing the entire block to be marked by erasure area pointers as shown in Fig. 4. Ex. 1003, Fig. 4. In response, Suda teaches "canceling the relation between the logical block addresses and the physical addresses." *Id.*, 5:65-6:3. Thus, the relation (the claimed "assignment") between the logical block address and the physical address are canceled ("removed," as claimed). Baker, ¶ 193.

In a second example of Fig. 6, an erase command is received to erase memory including block B. *Id.*, 6:15-21. In response, Suda again reiterates to "erase[] address information of the physical block B and a logical block address," from the logical and physical address table. *Id.*, 6:33-41. Again, the information (the claimed "assignment") of the logical block address and the physical address are erased ("removed," as claimed). Baker, ¶ 193. A POSITA would have understood both of

these examples as Suda's flash memory controlling system cancelling/erasing an assignment between an identified logical address and a physical address of the solid-state storage medium in response to an erase message. Baker, *Id*.

In addition, Suda's erase command designates a logical address. Ex. 1003, 6:66-9:3. And a POSITA would have understood that Suda's erase command indicates that "the identified logical address is erased" under either party's construction. Baker, ¶ 194; *see* § VI, *supra* (construing term).

Under UTL's construction, a POSITA would have understood Suda's erase command to indicate that "the data identified by the logical address does not need to be preserved." Baker, ¶ 194; *see* § VI, *supra* (construing term). Suda's system responds to the erase command by using erasure area pointers to mark data at the designated addresses as in a "virtual erased state." *E.g.,* Ex. 1003, 5:19-27. Such virtually erased data is "to be erased" later from the flash memory, meaning the data will not be preserved. *E.g., id.*, 5:40-48, 5:57-61, 6:9-14, 6:29-45, 6:60-64, 7:38-41; Baker, ¶ 194.

Under Petitioner's plain and ordinary meaning construction, a POSITA would have understood Suda's erase command to indicate that "the identified logical address is erased" at the host device. Suda shows the host device is a digital camera that lets users delete, for example, unwanted photos such as IMG001.jpg. *Id.*, Fig. 1, 2:61-62. A POSITA would have known that the digital camera keeps, in cache

memory, information about which logical identifiers are associated with which photos. Baker, ¶ 195. When a user selects to delete a photo, such as IMG001.jpg, the digital camera looks to this cache for the corresponding logical identifier and designates this logical identifier in an erase command. *Id.*; Ex. 1003, 8:66-9:3. As part of the deletion process, the camera will erase the cached entry of IMG001.jpg, along with the corresponding logical identifier, just like how Suda's system erases assignments of logical to physical address information described a few paragraphs above, in order to free up cache memory. Baker, ¶ 195. Thus, a POSITA would have understood that when the erase command was issued, the erase command indicates that "the identified logical address is erased" in the memory of the digital camera. *Id.*, ¶ 195. And, as discussed in the preceding paragraph, the memory device receiving the command further understands the erase command to mean that the memory device does not need to preserve the photo. *Id.*, ¶ 194.

B. Claim 2

a) Element 2[a]

As discussed for claim 1[d], the indexer in Suda's flash memory controlling section assigns logical addresses to physical addresses and stores the resulting assignments (the claimed "index entries") in the logical and physical address table (part of the claimed "index"). Ex. 1003, 3:13-15, 3:33-40, 3:64-67 ("physical block address is related to a logical block address in accordance with the control of the

flash memory controlling section"). Logical and physical address table 13a in Figure 7 shows an example "use of index entries" to assign the logical and physical addresses. *Id.*, Fig. 7 (annotated at § X.A.D, *supra*), 3:48-55; *see* § VI, *supra* (providing UTL's examples of "index entries"). Thus, a POSITA would have understood Suda to teach this element. Baker, ¶ 197.

To the extent UTL argues that claim 2[a] requires consulting the index entries when making a new assigning logical address to physical address assignment, this would have been obvious to a POSITA as well. Baker, ¶ 198-99. Suda's system does not assign physical block numbers that are already in use. Baker, ¶ 198. Suda uses this principle by requiring that an "**unused** physical block can be used when its physical block address is related to a logical block address in accordance with the control of the flash memory controlling section 11." Ex. 1003, 3:64-66 (emphasis added); *see also* Fig. 10 (showing assignment of unused physical block 4). During address assignment, the index entries would be consulted to avoid assigning a physical address already in use. Baker, ¶ 198.

b) Element 2[b]

As discussed for claim 1[e], Suda teaches to cancel/erase relationships between a logical block address and physical address in response to an erase command. Ex. 1003, 5:65-6:3; 6:15-25, 33-41; *see* § X.A.e, *supra* (discussing examples with respect to Fig. 4 and Fig. 6 of Suda). These cancellations/erasures

of index entries occur in response to an erase command that designates the logical address. Ex. 1003, 6:18-21, 7:32-35, 7:45-51, 8:66-9:3. As discussed for claim 1[d], Suda's flash memory controlling section is the claimed "indexer" that manages data erasure and the logical and physical address table. *Id.* 3:13-15; *see* § X.A.d, *supra*. Thus, a POSITA would have understood Suda to teach this claim. Baker, ¶¶ 200-01.

C. Claim 3

As discussed for claims 1[c]-[d], Suda teaches the claimed "storage controller" and "indexer." *See* §§ X.I.c-d, *supra*. This indexer maintains the logical and physical address table in a RAM (the claimed "memory") connected to and controlled by the flash memory controlling section ("of the storage controller," as claimed). Ex. 1003, Fig. 1, 3:1-2, 3:13-15, 3:41-43. As shown in Fig. 7 (annotated at claim 1[d], *see* § X.A.d, *supra*), the logical and physical address table includes assignment of logical addresses to physical addresses. UTL contends that logical addresses and physical addresses are examples of "index metadata." *See* § VI, *supra* (citing Ex. 1013, 3). For the reasons discussed for claim 2[a], a POSITA would have understood that Suda's indexer assigns logical addresses to physical addresses by use of at least the physical addresses (the claimed "index metadata") in the logical and physical address table maintained in a RAM of the flash memory controlling section, as claimed. Baker, ¶¶ 203-04.

D. Claim 5

Suda teaches this element under either construction of "garbage collector." *See* § VI, *supra* (construing term as "hardware and/or software for recovering space for a system that does not support update-in-place of data in the storage medium"). Suda's flash memory does not support direct rewriting/overwriting of data, meaning that it does not support "update-in-place of data in the storage medium." *Id.*; Baker, ¶ 209; Ex. 1003, 1:54-55. Thus, any hardware and/or software for recovering space on Suda's flash memory fits UTL's construction. For example, Suda's flash memory controlling section includes hardware/software is configured to manage erasures (recovery) of flash memory. Ex. 1003, 3:13-15.

Suda's garbage collector is "configured to designate that the physical address previously assigned to the identified logical address comprises data suitable for removal from the solid-state storage medium in response to the message," as claimed. In response to an erase command designating a logical address, Suda teaches to use start and end erasure area pointers to designate that identified data is in a "virtual erased state." Ex. 1003, 5:19-23, Figs. 4-6 (illustrating examples of erasure area pointers), Figs. 7, 10 (showing erasure area pointers in table 13a). Virtually erased data is not yet erased, but is "to be erased" later, meaning that it is "suitable for removal," as claimed. *E.g., id.*, 5:40-48, 5:57-61, 6:9-14, 6:29-45, 6:60-64, 7:38-41; Baker, ¶ 210. The areas designated by the erasure area pointers

are the physical addresses identified by logical addresses in an erase command. *E.g.,* Ex. 1003, 6:15-21, 7:29-34, 8:66-9:3. Suda's erasure area pointers are set as part of a process that includes recovering the virtually erased space, consistent with UTL's construction of "recovering space." Baker, ¶ 210. Suda's blocks will be physically erased (or "recovered") once the block fills up with virtually erased data. Ex. 1003, Figs. 4, 6, Fig. 8 (steps S5 and S6), 5:54-6:3 (in order that a physical block … to be erased be set in an unused state), 6:15-41 ("thereby setting the entire area (area 25) of the physical block B in an unused state."). For these reasons, a POSITA would have understood that Suda teaches the garbage collector as claimed, under either party's construction. Baker, ¶¶ 209-11.

E. Claim 6

A POSITA would have understood that Suda teaches a host interface section 12 as the "bus interface" that communicatively couples Suda's flash memory controlling section (the claimed "solid state storage controller") to the host device (the claimed "computer system"). Ex. 1003 at Fig. 1, 3:4-6; Baker, ¶ 212.

Suda does not expressly disclose the types of bus interfaces that may be supported by the host interface section 12. Even so, the claimed bus interfaces would have been obvious to a POSITA because they merely include industry standards that were well-known to a POSITA at the time. Baker, ¶ 213. For example, a POSITA would have known of the industry standard SATA interface.

Id.; Ex. 1007 (SATA 2.5 Standard). Even the inventors admit that SCSI and ATA standards (which include SATA) were "ubiquitous." Ex. 1015, 92 (stating that object storage methodologies "are not ubiquitous like block device protocols (such as SCSI and ATA)"). The SATA standard enjoyed widespread adoption throughout the computer industry. Ex. 1007 at 1 (listing major companies as board members, including Dell, Hewlett-Packard, Hitachi, Intel, Maxtor, Seagate, and Vitesse); Baker, ¶ 213. The SATA standard was obvious because it defined a high-speed interface to ease integration and enable scalable performance with data rates of 1.5 Gbps and 3.0 Gbps. Ex. 1007 at 19. Flash drives commonly implemented this SATA interface to achieve these high speeds and to enjoy widespread compatibility across the industry. Baker, ¶ 213. Thus, it would have been obvious to a POSITA for Suda's host interface section to be a common interface such as SATA. *Id.*, ¶¶ 213-14.

F. Claim 12

a) Element 12[a]

If the preamble is limiting, Suda teaches a non-volatile solid-state storage system. Ex. 1003, Fig. 1, 2:57-66; Baker, ¶ 216. This system includes flash memory provided as NAND type nonvolatile memory. Ex. 1003, Fig. 1, 2:57-66. The following annotated version of Ex. 1003, Fig. 1 (annotated below) shows various claimed components:

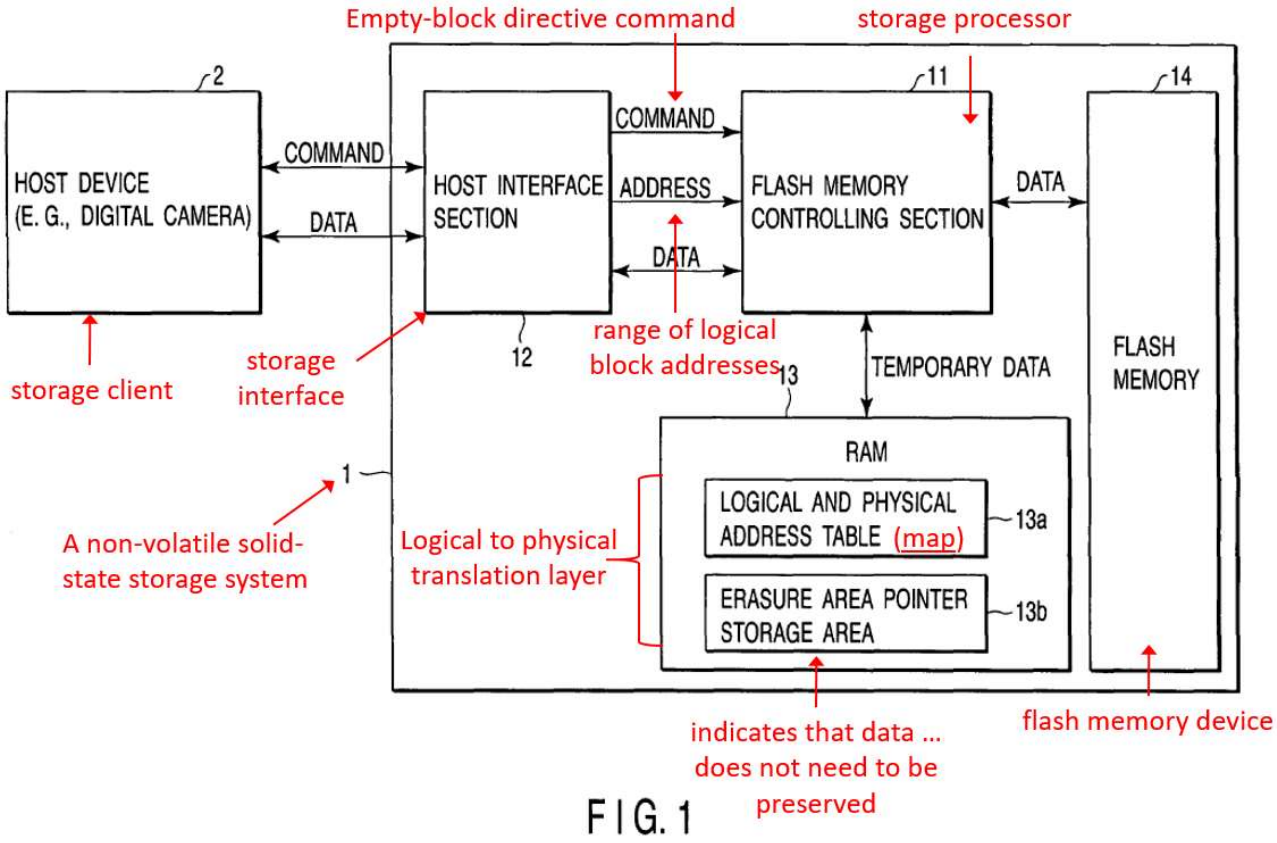


FIG. 1

b) Element 12[b]

A POSITA would have understood that Suda teaches that the memory device includes a host interface section (the claimed "storage interface") that communicates with a host device (the claimed "storage client"). Ex. 1003, 2:63-67, 3:4-6, Fig. 1 (annotated at § X.F.a, *supra*); Baker, ¶ 217.

c) Element 12[c]

Suda teaches that the memory device includes a flash memory controlling section, which is the claimed "storage processor." Ex. 1003, 2:63-64; Baker, ¶ 218. This flash memory controlling section is coupled to the host interface section. *Id.*, 2:66-3:1, Fig. 1 (annotated at § X.F.a, *supra*).

d) Element 12[d]

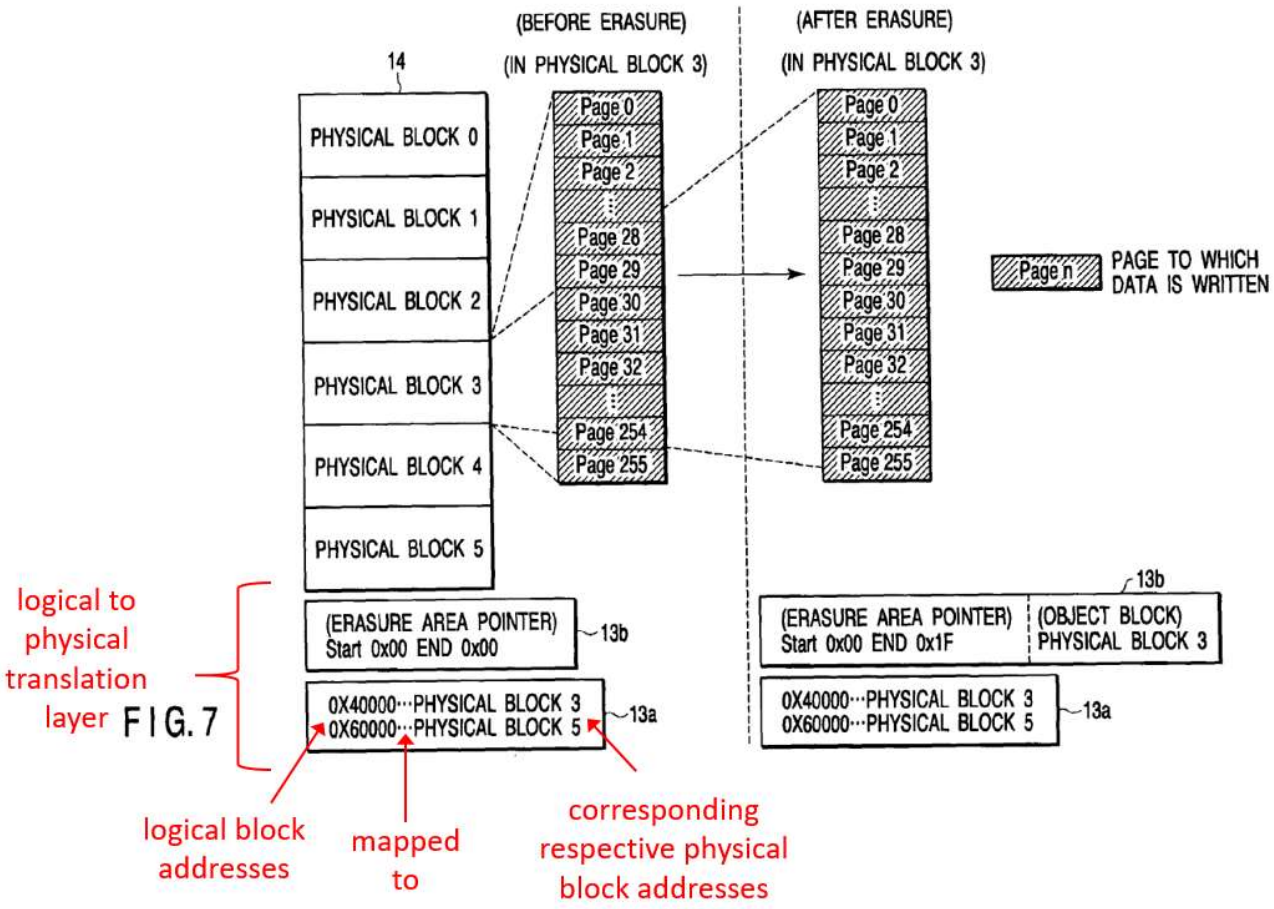
Suda teaches that "the flash memory controlling section 11 is connected to . . . the flash memory 14," which is the claimed "flash device." Ex. 1003, 3:1-3, Fig. 1 (annotated at § X.F.a, *supra*); Baker, ¶ 219.

e) Element 12[e]

According to UTL, the logical-to-physical translation layer is "often referred to as the FTL[7] of the NAND flash memory. The FTL is a structure/set of functions for mapping LBAs[8] to Physical Blocks. … [T]ables are widely used in order to map sectors and pages from logical to physical (Flash Translation Layer or FTL)." Ex. 1013, 6. A POSITA would have understood that Suda teaches a logical-to-physical translation layer under UTL's example. Baker, ¶ 221. Suda teaches a logical-to-physical translation layer that includes the logical and physical address table and the erasure area pointer storage area. *Id.*, 221-22; Ex. 1003, Fig. 1. Both of these tables are managed (or "maintained," as claimed) by the flash memory controlling section (the claimed "storage processor"). Ex. 1003, 3:13-15. These tables are structures that map logical block addresses to corresponding respective physical block addresses, thereby implementing the FTL as argued by UTL. Baker, ¶ 221. An annotated version of Suda's Fig. 7 below shows how the logical and physical

[7] "Flash Translation Layer."
[8] "Logical Block Addresses."

address table 13a maps logical block addresses to corresponding respective physical addresses:



Ex. 1003, Fig. 7 (annotated); *see also id.*, 3:41-55 (describing how the logical and physical address table maps logical block addresses to physical block addresses). Thus, a POSITA would have understood Suda to teach this element. Baker, ¶ 221-23.

f) Element 12[f]

Suda teaches, "A command issued from the host device 2 to the memory card 1 is input to the host interface section 12 …. The flash memory controlling section

11 operates based on the command information and address information from the host interface section 12." Ex. 1003 at Fig. 1 (annotated at § X.F.a, *supra*), 3:4-12. These commands include erase commands that designate a logical block address. *Id.*, 7:30-34, 8:66-9:3. In some of Suda's examples, the erase commands include a range of logical block addresses, as claimed. *Id.* at Fig. 6 (showing a range spanning three blocks), 6:15-26 ("an erase command to erase data items written to a number of physical blocks …. the case where an erase command to erase data items written to three physical blocks"). Thus, a POSITA would have understood that Suda taught that the flash memory controlling section (the claimed "storage processor") was configured to receive, from the host device (the claimed "storage client") through the host interface section (the claimed "storage interface"), an erase command and a range of logical block addresses. Baker, ¶¶ 224-25.

A POSITA would have understood that Suda's erase command was an empty-block directive command under either claim construction. *See* § VI, *supra* (construing term). Baker, ¶ 226. Under UTL's construction, Suda's erase command indicates that blocks contain data that does not need to be preserved for the reasons discussed for claim 1[e], namely that Suda responds to the command by marking the data in blocks as virtually erased and "to be erased" later. *See* § X.A.e, *supra*; Baker, ¶ 226. Under the Petitioner's construction, Suda's erase command is treated as a command to empty data from a block for similar reasons. Baker, ¶ 227. Suda

responds to the erase commands by marking data from a block "to be erased" later, thus treating the erase command as a command to empty data from a block (though not necessarily immediately). *See* § X.A.e; Baker, ¶ 227. Thus, a POSITA would have found that Suda teaches this claim element under either construction. Baker, ¶¶ 226-28.

g) Element 12[g]

A POSITA would have understood Suda to teach updating the erasure area pointers storage area in the logical-to-physical translation layer to indicate that the data stored in the corresponding physical blocks do not need to be preserved. Ex. 1003, Fig. 8 (updating at step S4); Baker, ¶ 230. As discussed for claim 1[e], Suda teaches to use erasure area pointers to mark data at the designated addresses as in a "virtual erased state," and is "to be deleted," meaning the data will not be preserved. *See* § X.A.e, *supra*; Ex. 1003, 5:40-48, 5:57-61, 6:9-14, 6:29-45, 6:60-64, 7:38-41; Baker, ¶ 230.

h) Element 12[h]

Suda teaches this element under either construction of "persistent data." *See* § VI, *supra* (construing term). Flash memory is nonvolatile; thus, any data stored in flash is "persistent data" under either party's construction. Ex. 1003, 2:65-66.

Suda stores a copy of its erasure area pointers in flash memory to avoid losing the erasure area pointers when the power supply is turned off. Ex. 1003, 8:3-16 ("it

writes, in the flash memory 14 also, the data items written to the erasure area pointer storage area …. even if a power supply to the memory card 1 is turned off … a virtual erased state is also maintained."). An annotated version of Ex. 1003, Fig. 1 shows this process:

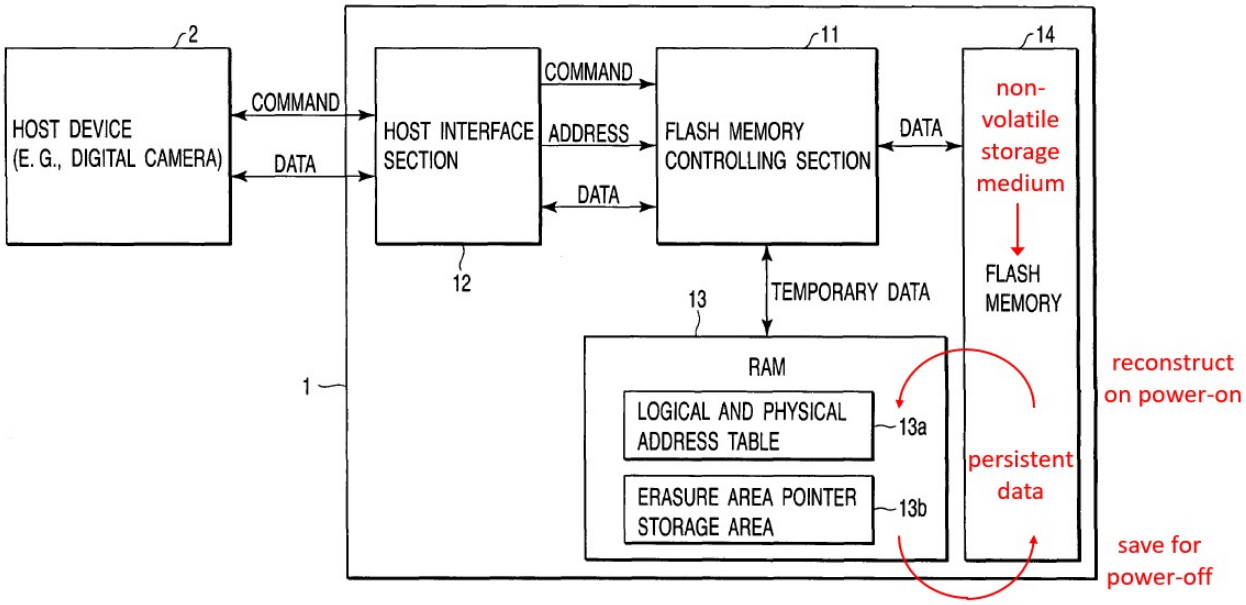


FIG. 1

These erasure area pointers are set in response to an erase command. Ex. 1003, 7:11-55. According to UTL, the accused Trim command infringes this element because "The TRIM command tells the SSD that specific areas contain data that is no longer in use. From the user's perspective, this data has been deleted from a document." Ex. 1013, 7.

Under UTL's reasoning, Suda's persistent erasure area pointers indicate "that the data corresponding to the received logical block addresses is deleted at the

storage client," as claimed. This because Suda teaches the same thing accused by UTL: erasure area pointers indicate data that is no longer in use, and, "From the user's perspective, this data has been deleted from a document." Ex. 1013, 7; Baker, ¶ 234-35. Suda shows the host device is a digital camera that lets users delete unwanted photos. Ex. 1003, Fig. 1, 2:61-62. A POSITA would have known that, when a user selects a photo to delete, the digital camera will look up the logical address corresponding to the digital photo selected for deletion and issue an erase command to the memory device, the erase command designating the logical address. Baker, ¶¶ 235-36; Ex. 1003, 8:66-9:3, Fig. 1. As a result, the photo will appear deleted from a user's perspective. Baker, ¶¶ 235-36. In fact, Suda's memory device uses erasure area pointers to mark this data as in a "virtual erased state," which indicates that this data is no longer in use should appear deleted from the user's perspective. Ex. 1003, 5:14-27, 8:21-41; Fig. 9; Baker, ¶¶ 235-37. This indicated data cannot be read by the user; Suda's system will instead return initial-value (or empty) data. Ex. 1003, 5:14-27, 8:21-41. Thus, a POSITA would have understood Suda to teach this element in the same way that UTL accuses the TRIM command of infringement. Baker, ¶¶ 232-38.

G. Claim 13

As discussed for claim 12[h], Suda teaches to store its erasure area pointers in the flash memory device in order to prevent data loss when powered off. *See*

§ X.F.h, *supra*. It would have been obvious to a POSITA that the rest of Suda's logical-to-physical translation layer, including the logical and physical address table, also be similarly stored in persistent memory for the same reason. Baker, ¶ 240. 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. *Id.* A POSITA would have understood that both tables can be preserved by storing their data in the nonvolatile flash memory. *Id.* Otherwise, the logical-to-physical mappings would be lost every time the power is turned off. *Id.* Thus, a POSITA would have found it obvious to store Suda's logical-to-physical translation layer, including the logical and physical address table, in the flash memory device. Baker, ¶ 239-41.

H. Claim 14

Suda illustrates RAM 13 (the claimed "volatile memory device") coupled to the flash memory controlling section (the claimed "storage processor"). Ex. 1003, Fig. 1 (annotated at § X.F.a, *supra*). For the reasons discussed for claim 12[e], a POSITA would have understood that the logical and physical address table and erasure area pointer storage area make up the logical-to-physical translation layer. *Id.,* Fig. 1, 2:63-3:3, 3:41-47, 4:1-3, 9:7-11; *see* § X.F.e, *supra*. Thus, a POSITA would have understood Suda to teach this element. Baker, ¶¶ 242-44.

I. Claim 15

Suda teaches this claim element in Fig. 9. Ex. 1003, 8:21-41 (explaining steps). The flash memory controlling section (the claimed "storage processor") is configured to perform these steps responsive to a data read command. *Id.*, 8:21-29. At step A2, the flash memory controlling section 11 determines that the read address is included in the erasure area specified by a previous erase command (the claimed "empty-block directive command"). *Id.*, 8:35-38. Then at step A3, the flash memory controlling section outputs "initial-value data" instead of user data. *Id.,* 8:35-41. A POSITA would have understood that initial-value data refers to the data initially written in an unused memory device, which would be a string of empty values (typically all 1's). *Id.,* 3:58-59, 4:4-6; Baker, ¶ 245. Thus, a POSITA would have understood Suda to teach this claim. Baker, ¶¶ 245-46.

Aside from Suda's disclosure, the background of the '727 patent admits that this claim element was known. The inventors admitted in the background, "Another method of erasing data is to write zeros, ones, or some other null data character to the data storage device." Ex. 1001, 1:37-9, 1:56-60.

J. Claim 16

As discussed for claim 15, a POSITA would have understood initial value data as the data initially written in an unused physical block, in other words, a string of empty values (typically all 1's). *Id.,* 3:58-59, 4:4-6; Baker, ¶ 247; *see* § X.I, *supra*.

Thus, a POSITA would have understood Suda to teach this claim. Baker, ¶¶ 247-48. Additionally, the inventors admit in the background of the '727 patent that this claim was known. Ex. 1001, 1:56-60 ("The storage device may also receive a command to read the erased file so the storage device may transmit a stream of zeros, ones, or a null character to the requesting device.").

XI. Ground 3: A POSITA Would Have Found Claims 4 and 13 Obvious In View of Suda, Bennett and the Knowledge of a POSITA.

A. Claim 4[9]

Claim 4 depends on Claim 1, which would have been obvious to a POSITA for the reasons discussed under Ground 2. *See* § X.A, *supra*. Suda does not explicitly show the internal components of its flash memory controlling section or state that the indexer in the flash memory controlling section includes "firmware." *See* Ex. 1003 at Fig. 1. However, as evidenced by Bennett, it would have been obvious to a POSITA for Suda's indexer to comprise firmware. Baker, ¶ 207.

Bennett shows typical internal components, including ROM, in a controller in Fig. 1. Ex. 1002 at Fig. 1. Bennett teaches, "Firmware stored in nonvolatile ROM 122 and/or the optional nonvolatile memory 124 provides codes for the processor 120 to implement the functions of the controller 100." *Id.*, 7:4-7, 24:19-20.

It would have been obvious to a POSITA that the indexer in Suda's "flash

[9] See attached claim listing

memory controlling section" would have had similar internal components to those described in Bennett, including firmware. Baker, ¶ 207. This firmware would have enabled Suda's flash memory controlling section 11 to "implement the functions of the controller" as Bennett describes. Ex. 1002, 7:4-7; Baker, ¶ 207. Thus, claim 4 would have been obvious to a POSITA in view of Suda and Bennett. Baker, ¶¶ 207-08.

B. Claim 13

Claim 13 depends on Claim 12, which would have been obvious to a POSITA for the reasons discussed under Ground 2. To the extent that UTL disagrees that claim 13 would not have been obvious to a POSITA based on Suda alone, claim 13 would have been obvious to a POSITA in further view of Bennett. Bennett teaches, "mappings between logical groups and physical groups … are stored in a set of table and lists distributed among the nonvolatile flash memory 200 and the volatile but more agile RAM." *Id.* at 10:40-43; *see also* §§ X.A.d, X.H, *supra* (discussing how Bennett teaches this element). This implements a common technique called "caching," where the logical to physical address table is stored in RAM for faster access and also stored in nonvolatile flash so it is not lost when powered off. Baker., ¶ 240. Suda already does this for its erasure area pointers. Ex. 1003, 8:3-16. A POSITA would have found it obvious that Suda's logical and physical address table could be cached in the same way, such that it is also stored in nonvolatile flash

memory in addition to RAM. Baker ¶ 240. Suda already taught storing the logical and physical address table in RAM, and Suda already recognize the desirability of not losing data when powered off. Ex. 1002, Fig. 1, 8:6-16. Thus, a POSITA would have found this claim obvious. Baker., ¶ 240.

XII. Secondary considerations

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

Here, exhibits 1017-1019 show that standard-setting group T13 began work on the Trim command proposal at least by April 21, 2007, only four months from the earliest possible (disputed) priority date. Baker, ¶ 91. UTL accuses this Trim command of infringing the claims. Ex. 1013, *passim*. Like the *ZTE* case, here a

standard-setting group worked on the same technology around the same time. Exs. 1017-1019. Also, Suda and Bennett teach similar commands. Ex. 1002, 17:52-56 (an erase command "specifies the (logical) sectors to be erased"); Ex. 1003, 9:2-3 ("logical block address ... designated in the erase command"). Furthermore, many claim elements were already well-known in the art. *See, e.g.,* Ex. 1010 § 2.2 (Ban patented the FTL in 1995, and the FTL became part of an industry standard), § 2.3 (explaining the garbage collection process). Thus, Exhibits 1002-1003 and 1017-1019 all serve as evidence of simultaneous invention by others, and the Board should find the challenged claims obvious for being only the product of ordinary mechanical or engineering skill.

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

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

The fourth factor (overlap) strongly favors institution. Petitioners have stipulated that they will not pursue invalidity on the same grounds—or even the same references—if the Board institutes trial in this proceeding. Ex. 1028. Petitioners modeled this stipulation on the stipulation that the Board found to

"mitigate any concerns" in *VMware, Inc. v. Intellectual Ventures I LLC*, IPR2020-00470, Paper 13 at 20 (P.T.A.B. August 18, 2020). This fourth factor favors institution here even more so than in *Apple, Inc. v. SEVEN Networks, LLC*, IPR2020-00156, Paper 10 (P.T.A.B. June 15, 2020). There, the petitioner provided no stipulation. *Id*. pp. 16-19. Nevertheless, the fourth factor "strongly favored" petitioner. *Id*.

The third factor (investment in parallel proceeding) also favors institution. The District Court has not issued any substantive opinions regarding the scope or validity of the challenged claim, and given Petitioners' stipulations, the Court is unlikely to invest any resources on the grounds raised in this petition, either before or after the scheduled institution date. Furthermore, the parallel proceeding is in an early stage: the Court is deciding Petitioners' motions to dismiss, 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.[10] *See Globalfoundries Inc. v. UNM Rainforest Innovations*, IPR2020-00984, Paper 11 at concurrence 3 (P.T.A.B. Dec. 9, 2020) (weighing large number of trials and proportion of reschedule trials and noting "as the period of time remaining before trial increases, the certainty that the trial date

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

will remain unchanged decreases").[11]

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, 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.

[11] 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").

B. Related Proceedings

In three related lawsuits, UTL asserted the '727 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 23, 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]	A non-volatile solid-state storage system, comprising:
1[b]	a solid-state storage medium;
1[c]	a solid-state storage controller configured to implement storage operations on the solid-state storage medium in response to requests from a computer system, including storing data pertaining to logical addresses of a logical address space at respective physical addresses of the solid-state storage medium; and
1[d]	an indexer, comprised within the solid-state storage controller, wherein the indexer is configured to assign logical addresses of the logical address space to physical addresses in use to store data pertaining to the logical addresses on the solid-state storage medium;
1[e]	wherein the indexer is further configured to remove an assignment between an identified logical address and a

	physical address of the solid-state storage medium in response to a message received from a host operating system, the message indicating that the identified logical address is erased.

Claim 2	
Element	**Language**
2[a]	The apparatus of claim 1, wherein the indexer assigns logical addresses to physical addresses by use of index entries,
2[b]	and wherein the indexer removes an index entry corresponding to the identified logical address in response to the message.

Claim 3	
Element	**Language**
3	The apparatus of claim 1, wherein the indexer assigns logical addresses to physical addresses by use of index metadata maintained in a memory of the storage

	controller.

Claim 4	
Element	**Language**
4	The apparatus of claim 1, wherein the indexer comprises firmware of the solid-state storage controller.

Claim 5	
Element	**Language**
5	The apparatus of claim 1, further comprising a garbage collector configured to designate that the physical address previously assigned to the identified logical address comprises data suitable for removal from the solid-state storage medium in response to the message.

Claim 6	
Element	**Language**
6	The apparatus of claim 1, wherein the solid-state storage controller comprises a bus interface configured to

	communicatively couple the solid-state storage controller to the computer system, wherein the bus interface comprises one of a universal serial bus interface, an Institute of Electrical and Electronics Engineers 1394 bus interface, an external Serial Advanced Technology Attachment bus interface, a Peripheral Component Interconnect (PCI) bus interface, a PCI Express bus interface, an InfiniBand interface, an Integrated Drive Electronics (IDE) bus interface, an AT Attachment (ATA) interface, a Parallel ATA (PATA) interface, a Serial ATA (SATA) bus interface, an external SATA bus interface, a Small Computer System Interface (SCSI) bus interface, an internet SCSI interface, and a Fibre Channel interface.

Claim 12	
Element	**Language**
12[a]	A non-volatile solid-state storage system, comprising:
12[b]	a storage interface configured to communicate with a

	storage client;
12[c]	a storage processor coupled to the storage interface;
12[d]	a flash memory device coupled to the storage processor; and
12[e]	a logical-to-physical translation layer maintained by the storage processor, wherein the logical-to-physical translation layer maps logical block addresses to corresponding respective physical block addresses of the flash memory device,
12[f]	wherein the storage processor is configured to: receive, from the storage client through the storage interface, an empty-block directive command and a range of logical block addresses,
12[g]	update the logical-to-physical translation layer to indicate that data stored in physical block addresses corresponding to the received logical block addresses do not need to be preserved, and

12[h]	store persistent data on the flash memory device, the persistent data indicating that the data corresponding to the received logical block addresses is deleted at the storage client.

Claim 13	
Element	**Language**
13	The system of claim 12, wherein the logical-to-physical translation layer is stored in the flash memory device.

Claim 14	
Element	**Language**
14	The system of claim 12, further comprising a volatile memory device coupled to the storage processor, wherein the logical-to-physical translation layer is stored in the volatile memory device.

Claim 15	
Element	**Language**

15	The system of claim 12 wherein the storage processer is configured such that, responsive to receiving a read request specifying one or more logical addresses included in the empty-block directive command, the storage processor returns a predetermined data string.

Claim 16	
Element	**Language**
16	The system of claim 15, wherein data bits of the predetermined data string have a uniform logic level.

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 13,017 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 was used to prepare this Petition.

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

CERTIFICATE OF SERVICE

Under 37 C.F.R. §§ 42.6(e) and 42.105(a), this is to certify that on December 22, 2020, I caused to be served a true and correct copy of the foregoing "**PETITION FOR *INTER PARTES* REVIEW OF CLAIMS 1-6 AND 12-16 OF U.S. PATENT NO. 9,632,727,**" Petitioners Power of Attorney and Exhibits 1001 – 1036 by FedEx on the Patent Owner at the correspondence address of record for U.S. Patent No. 9,632,727:

Longitude Licensing Stoel Rives
201 South Main Street, Suite 1100
One Utah Center
Salt Lake City UT 84111

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

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

WINSTON & STRAWN LLP

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