IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Silicon Motion Inc.

Petitioner,

v.

Unification Technologies LLC,

Patent Owner.

Case No. IPR2024-00199
U.S. Patent No. 11,640,359

PETITION FOR *INTER PARTES* REVIEW OF CLAIMS 7 AND 10-17 OF U.S. PATENT NO. 11,640,359

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

Cases

Statutes

Other Authorities

PETITIONER'S EXHIBIT LIST

Ex. No.	Brief Description
1001	U.S. Pat. No. 11,640,359 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. 11,640,359.
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	U.S. Pat. Pub. 2003/0079078, titled "CONFIRMATION OF SECURE DATA FILE ERASURE" to Zipprich et al.
1007	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.
1008	Original Complaint for Patent Infringement, *Unification Techs. LLC v. Silicon Motion Inc.,* No. 2:23-cv-266 (E.D. Tex. 2023).
1009	U.S. Provisional Pat. No. 60/873,111 titled "Elemental Blade System," to Flynn et al.
1010	Docket Report for *Unification Techs. LLC v. Silicon Motion Inc.,* No. 2:23-cv-266 (E.D. Tex. 2023).
1011	Frank Shu, *Notification of Deleted Data Proposal for ATA8-*

Ex. No.	Brief Description
	ACS2, T13 (rev. 0 Apr. 21, 2007).
1012	Frank Shu & Nathan Obr, *Notification of Deleted Data Proposal for ATA8-ACS2 Revision 1*, T13 (July 26, 2007).
1013	Frank Shu & Nathan Obr, *Notification of Deleted Data Proposal for ATA8-ACS2 Revision 2*, T13 (September 5, 2007).
1014	Public file history of U.S. Pat. No. 11,640,359 B2, titled "SYSTEMS AND METHODS FOR IDENTIFYING STORAGE RESOURCES THAT ARE NOT IN USE" to Flynn et al.
1015	Curriculum vitae of Jacob Baker, Ph.D., P.E.
1016	U.S. Pat. No. 5,404,485A, titled "FLASH FILE SYSTEM" to Ban.
1017	Expert Report of Sylvia Hall-Ellis, Ph.D. in Support of Public Availability of the Gal Publication.
1018	Docket Control Order, *Unification Techs. LLC v. Silicon Motion Inc.,* No. 2:23-cv-266 (E.D. Tex. 2023), ECF No. 47
1019	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).
1020	Serial ATA (SATA) Revision 2.5, Serial ATA International Organization (Oct. 27, 2005).
1021	WILLIAM D. BROWN & JOE E. BREWER, NONVOLATILE SEMICONDUCTOR MEMORY TECHNOLOGY (IEEE 1998).
1022	BRIAN DIPERT & MARKUS LEVY, DESIGNING WITH FLASH MEMORY (Annabooks 1994).
1023	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.

Ex. No.	Brief Description
1024	U.S. Pat. No. 6,766,432 B2, titled "MEMORY MANAGEMENT SYSTEM SUPPORTING OBJECT DELETION IN NON-VOLATILE MEMORY" to Saltz et al.
1025	*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).
1026	*Mapping Structures for Flash Memories: Techniques and Open Problems*, IEEE *XPLORE*, https://ieeexplore.ieee.org/document/1421068 (last visited Dec. 18, 2020).

I. Introduction

U.S. Patent 11,640,359 (the "'359 patent") should never have issued. For example, claim 7 generally recites NAND flash storage that receives an empty-block identifier and in turn updates an index of mappings to indicate that data associated identified need to be preserved. A person of ordinary skill in the art ("POSITA") would have known of these concepts long before the alleged effective filing date.

For example, the Board previously found that patents in this family, with very similar claim scope, are invalid in view of one of the primary references, Suda (Ex. 1003), that is relied upon herein. In particular, the Board found claims in U.S. Patent No. 9,632,727 ("'727 Patent") obvious in view of Suda in IPR2021-00345 (Paper 41). A comparison of the '727 Patent claim 12 and the '359 Patent claim 7 is included below in § XV. As shown, both claims recite a non-volatile flash storage system that receives an empty-block indication from a host and in return updates a logical to physical translation mapping in order to indicate that the data stored in the physical locations need not be preserved. The subject matter of the challenged claims was known and the Board should invalidate the challenged claims.

II. Petitioner Meets Standing and Eligibility Requirements for *Inter Partes* Review.

Petitioner certifies under 37 C.F.R. § 42.104(a) that the '359 patent "is available for *inter partes* review and that the Petitioner is not barred or estopped from requesting an *inter partes* review challenging the patent claims on the grounds

identified in the petition." UTL sued Petitioner less than one year ago on June 2, 2023. Exs. 1008, 1010.

III. Prosecution History of the '359 Patent

The '359 patent application was filed on August 16, 2019. Ex. 1001, cover. The Examiner rejected the claims on various grounds, but did not cite the references relied upon herein. Ex. 1014. To overcome the rejections, claim 21 (issued claim 7) was amended to recite that the method comprises of "maintaining an index of mapping between logical identifiers and physical NAND flash storage locations" and "updating the index to indicate that the data stored in one or more of the physical NAND flash storage locations associated with the empty-block identifier does not need to be preserved. Ex. 1014.

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"), ¶ 53. Unlike traditional platter hard drives, flash memory cannot be directly overwritten—a block must be erased before written to again. *Id.*, ¶ 62. Erase commands for flash memory were well known and standardized before the earliest provisional for the '359 patent. Ex. 1005, §§ 6.16, 8.1.

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

Baker, ¶¶ 69-72. 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.*, ¶ 72.

V. Summary of the '359 Patent

The '359 patent acknowledges that erase commands for file systems were known. *See, e.g.,* Ex. 1001, 1:34-36 ("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, 1:38-40. 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.*, 1:40-43.

A. Effective Filing Date and Date of Invention

The '359 patent claims priority to provisional application no. 60/873111, filed December 6, 2006. Ex. 1001, cover. Solely for purposes of this IPR, Petitioner assumes, but does not concede, an effective filing date of December 6, 2006, for the

'359 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 in computer engineering, electrical engineering, computer science, or a closely related field, along with at least two years of experience in the design, development, implementation, or management of memory devices and systems. Baker, ¶ 45. 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, ¶ 50.

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. 1010. Petitioner asserts that no construction is

necessary for the purposes of the present Petitioner as the challenged claims are invalid under any reasonable construction.

VII. Precise Relief Requested

A. Proposed Grounds

a) Ground 1

Claims 7 and 10-17 are invalid under 35 U.S.C. § 103 over Suda (Ex. 1003) in view of a POSITA's knowledge.

b) Ground 2

Claims 7 and 10-17 are invalid under 35 U.S.C. § 103 over Bennett (Ex. 1002) in view of a POSITA's knowledge.

c) Ground 3

Claim 11 is invalid under 35 U.S.C. § 103 over Suda and/or Bennett in further view of Zipprich (Ex. 1006) and a POSITA's knowledge.

d) Ground 4

Claims 7 and 10-13 are invalid under 35 U.S.C. § 103 over Bennett in view of Suda and a POSITA's knowledge.

e) Ground 5

Claims 13-15 is invalid under U.S.C. § 103 over Suda and/or Bennett in view of SwSTE'05 (Ex. 1007) and a POSITA's knowledge.

B. Qualifying Prior Art

Bennett, Suda, Zipprich, and SwSTE'05 are prior art to the '359 patent.

Petitioner is unaware of any assertion that the '359 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. Zipprich (published on April 24, 2003) is § 102(a), (b), and (e) prior art. Ex. 1006, cover. SwSTE'05 (presented February 23, 2005 and published by IEEE on May 23, 2005) is § 102(a) and (b) prior art. Ex. 1007, 1-2; Baker, ¶¶ 48-49.

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 are identified on the face of the patent, but were not substantively discussed during prosecution. Furthermore, the Board recently invalidated very similar claims in light of the Suda reference relied upon herein. *See infra* Section XV.

VIII. The Prior Art

A. 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.

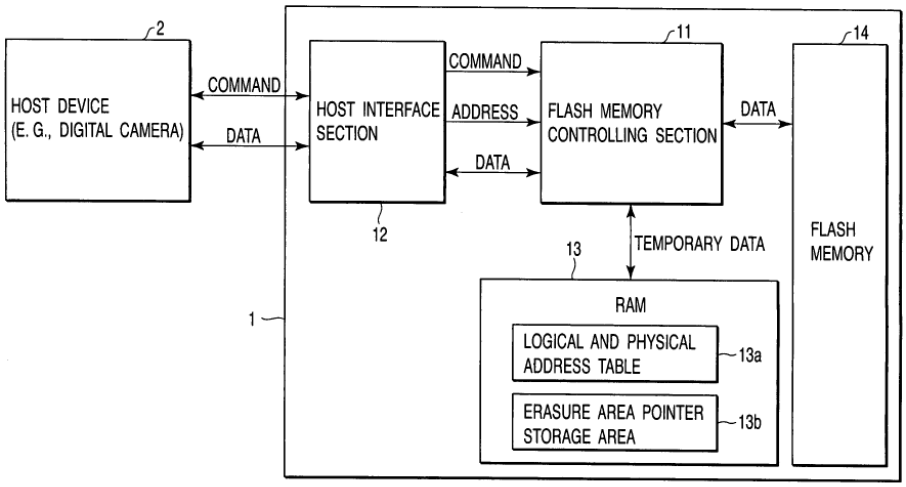


FIG. 1

Id., Fig. 1.

Like the '359 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).

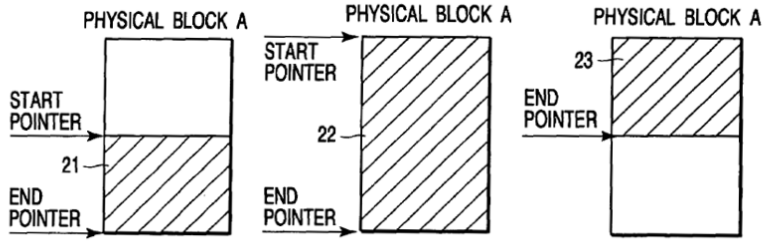


FIG. 3 FIG. 4 FIG. 5

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.

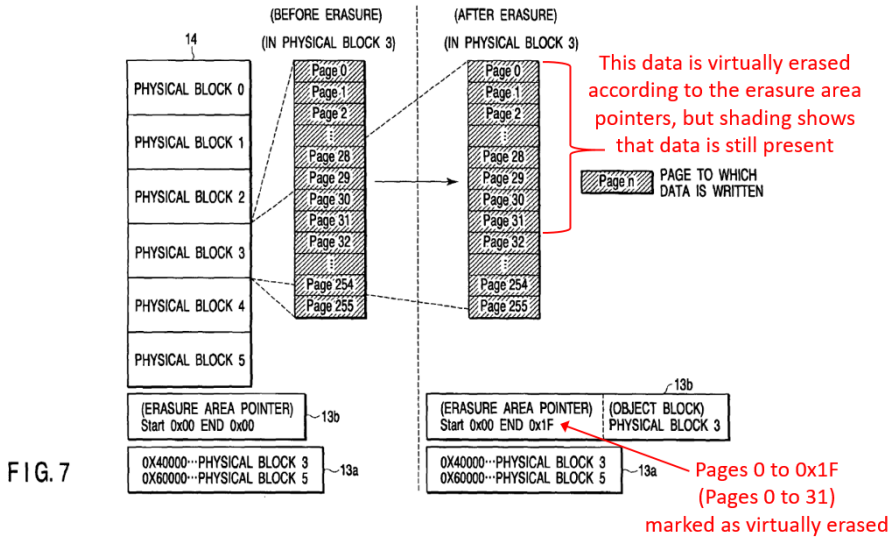


FIG. 7

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.

B. Summary of Bennett

Like the '359 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, ¶ 87-88, 163. The GAT is stored in non-volatile flash memory as highlighted in Bennett's Fig. 6 below:

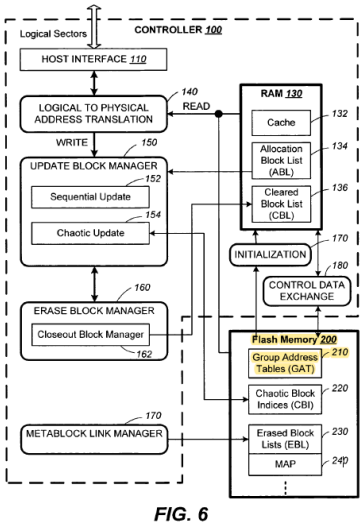


FIG. 6

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

The GAT is recorded as an index of sectors:

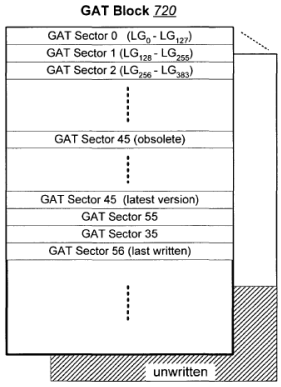


FIG. 8B

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, ¶¶ 123, 149, 151, 156. 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").

C. Summary of Zipprich

Zipprich provides a process that generates a status report following an overwrite of a data file according to a predetermined secure erase method. Ex. 1006, [0017]. The status report can provide immediate feedback and logging/tracking of the overwrite events. Ex. 1006, [0018]. Zipprich explains that the status report can be of various types and can be sent to various locations depending on the particular arrangement and desire of the user and/or administrator. *Id*.

D. Summary of SwSTE'05

SwSTE'05 provides a survey of flash memory technologies. Ex. 1007, Abstract. Two teachings of SwSTE'05 are relevant here.

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

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

E. Motivation to Combine Bennett, Suda, and Zipprich

A POSITA would have been motivated to combine teachings from Bennett, Suda, and Zipprich. Baker, ¶ 167. All references discuss data erasure and propose techniques for improving indications of data erasure in flash memory. Ex. 1002, *passim*; Ex. 1003, *passim;* Ex. 1006, *passim*.

Bennett and Suda do not explain every underlying technological concept in flash memory. Zipprich provides additional discussions of technological concepts that underlie Bennett's and Suda's method of erasure known to a POSITA. *See* § VIII.C, *supra*. These underlying concepts from Zipprich would have been easily and predictably implemented in Bennett's and Suda's systems because flash memory devices generally implemented these technological concepts. Baker, §§ VI.A.4. Section XI, *infra*, provides additional motivations for specific combinations.

F. Motivation to Combine Bennett and Suda

A POSITA would have been motivated to combine teachings from Bennett and Suda. Baker, ¶ 169. Both references discuss the management of flash storage devices. Ex. 1002, *passim*; Ex. 1003, *passim*. Both references propose techniques for improving performance given the same limitations of flash memory. Namely, that flash memory only supports erasing blocks, not erasure of individual pages within a block (Ex. 1002, Abstract; Ex. 1003, 4:33-38), and blocks cannot be directly overwritten without erasure (Ex. 1007, 3:11-17; Ex. 1003, 1:19-22, 1:54-55).

Bennett does not explain every underlying technological concept in flash memory. Suda provides additional discussions of the technological concepts that underlie Suda's system known to a POSITA. See §§ VIII.A-B, *supra.* These underlying technological concepts from Suda would have been easily and predictably implemented in Bennett's system because flash memory devices generally implemented these technological concepts. Baker, §§ VI.A.1-9 (describing background concepts). Section XII, infra, provides additional motivations for specific combinations.

G. Motivation to Combine Bennett, Suda, and SwSTE'05

A POSITA would have been motivated to combine teachings from Bennett, Suda, and SwSTE'05. Baker, ¶ 171. All references discuss the management of flash storage devices and garbage collection. Ex. 1002, *passim*; Ex. 1003, *passim*; Ex. 1007, *passim*. All references propose techniques for improving performance given the same limitations of flash memory. Namely, that flash memory only supports erasing blocks, not erasure of individual pages within a block (Ex. 1002, 3:43-50; Ex. 1003, 4:33-38; Ex. 1007, Abstract, § 1), and blocks cannot be directly overwritten without erasure (Ex. 1002, 3:10-17; Ex. 1003, 1:19-22, 1:54-55; Ex. 1007 § 1).

Bennett and Suda do not explain every underlying technological concept in flash memory. SwSTE'05 provides additional discussions of the technological

concepts that underlie Bennett's and Suda's systems known to a POSITA. See § VIII.D, *supra* (summarizing technological concepts). These underlying technological concepts from SwSTE'05 would have been easily and predictably implemented in Bennett's and Suda's systems because flash memory devices generally implemented these technological concepts. Baker, §§ VI.A.5, VI.A.7 (describing background concepts). Section XIII, infra, provides additional motivations for specific combinations.

IX. Ground 1: Obvious Over Suda and POSITA Knowledge

A. Claim 7[1]

a) Element 7[a]

If the preamble is limiting, Suda teaches "[a] method for managing data in a system having a host processor and a block-based NAND flash storage system, the NAND flash storage system having physical NAND flash storage locations controlled by a NAND controller." Suda teaches "managing a nonvolatile semiconductor memory which comprises a plurality of blocks" (the claimed "method for managing data in a system having… a block-based flash storage system"). Ex. 1003, Abstract. Suda's system includes flash memory provided as NAND type nonvolatile memory. Ex. 1003, Fig. 1, 2:57-66. The following

[1] *See* attached Claim Listing.

illustration annotates Figure 1 of Suda to show various claim elements, including this one:

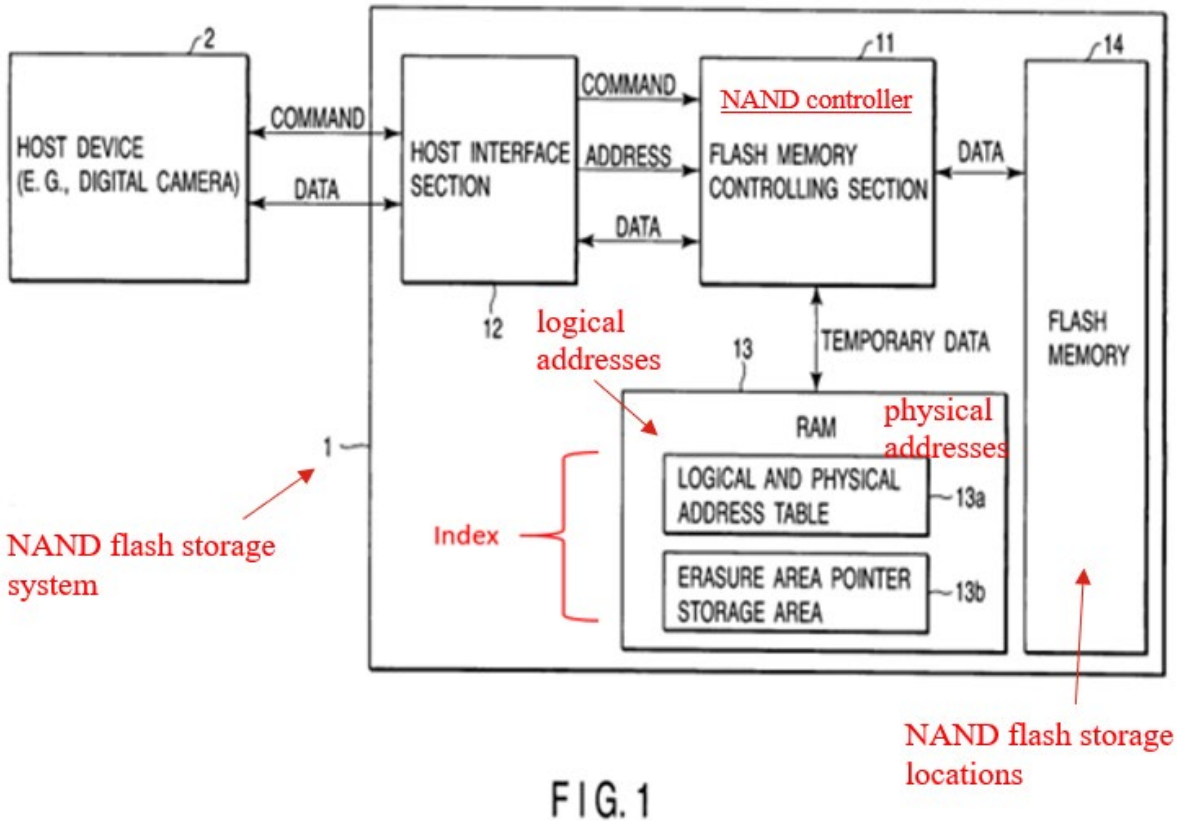


FIG. 1

Ex. 1003, Fig. 1 (annotated). Figure 1 shows a NAND flash storage system (1) which has physical NAND flash storage locations (14) controlled by a NAND controller (11). A POSITA would understand that a host device (2) has a host processor. Baker, ¶ 111.

Thus, a POSITA would have understood that Suda teaches this element. Baker, ¶¶ 110-111.

b) Element 7[b]

A POSITA would have understood Suda to teach "receiving, via the host processor, an indication of deletion of a file at the host processor." 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, ¶ 112. 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 block address information described for claim 7[f] below, in order to free up cache memory. Baker, ¶ 112. A POSITA would have known this process to have been performed by the digital camera's host processor. Baker, *Id*. Thus, a POSITA would have understood Suda to teach this element. Baker, *Id.*

c) Element 7[c]

A POSITA would understand that Suda teaches "creating, via the host processor, an empty-block identifier in response to the indication of deletion." A POSITA would have understood that when the erase command was issued by the

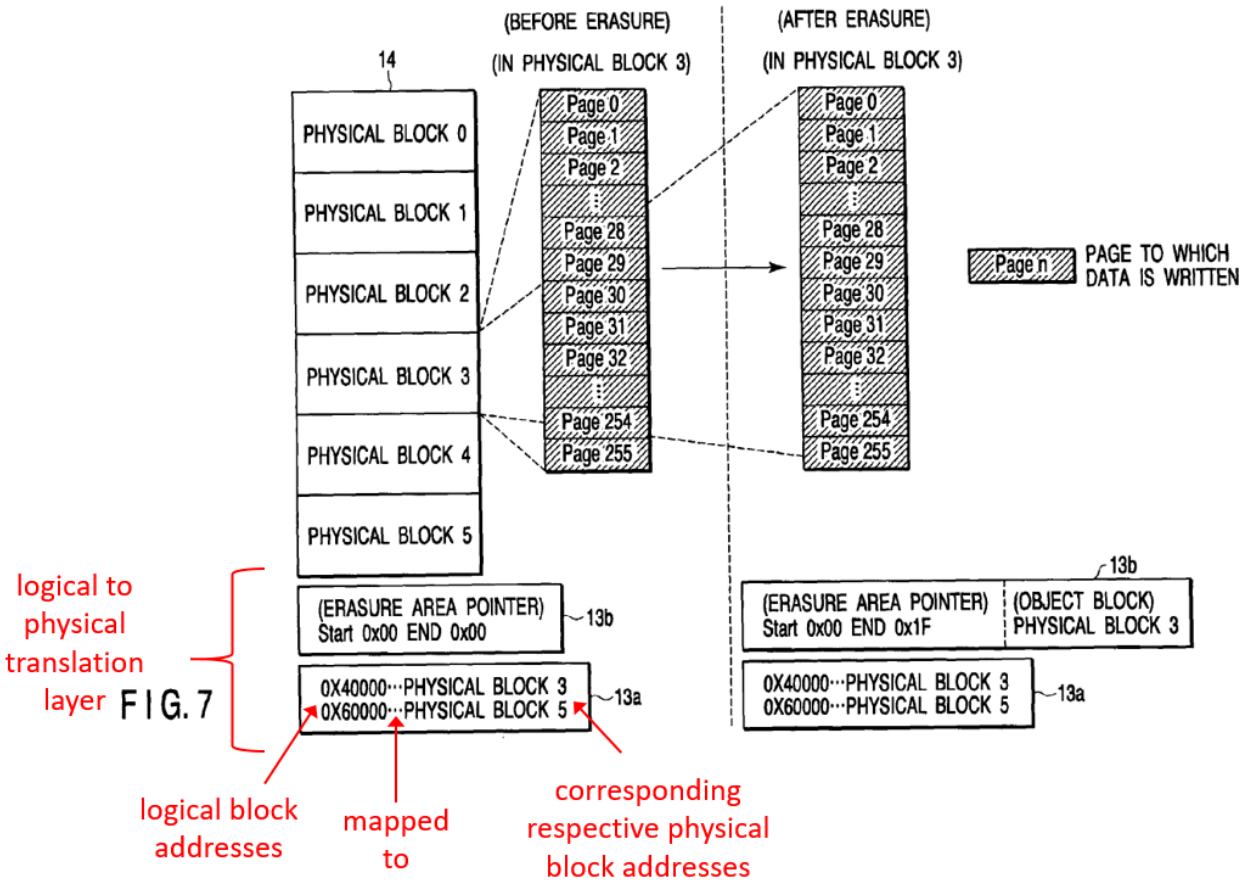
camera's host processor, the erase command was created in response to the file being deleted by a user of the digital camera. Baker, ¶ 113. Suda teaches a logical and physical address table that maps logical addresses to corresponding physical block addresses. Ex. 1003, 3:41-55. Thus, a POSITA would have understood that when the erase command was issued, the erase command creates an "empty-block identifier" in the memory of the digital camera. Baker, ¶ 113.

d) Element 7[d]

A POSITA would understand that Suda teaches this claim element. Suda teaches "receiving, via the NAND controller, an empty-block identifier in response to the indication of deletion" in the form of erase commands. 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). Thus, the indexer of Suda's flash memory controlling section (the claimed "NAND controller") receives an empty-block identifier as claimed. Baker, ¶ 114.

e) Element 7[e]

A POSITA would have understood that Suda teaches "maintaining, via the NAND controller, an index of mappings between logical identifiers and physical NAND flash storage locations." 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 "NAND controller"). Ex. 1003, 3:13-15. An annotated version of Suda's Fig. 7 below shows how the logical and physical 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, ¶ 115-116.

f) Element 7[f]

A POSITA would understand that Suda teaches "updating the index of mappings, in response to the receiving the empty-block identifier, to indicate that data associated with the file on one or more of the physical NAND flash storage locations identified by the empty-block identifier does not need to be preserved."

First, Suda teaches "updating the index of mappings" as claimed in the form of removing assignments of logical addresses to physical addresses. In the following two examples where entire blocks are erased, Suda 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, ¶ 117.

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, ¶ 117. A POSITA would have understood both of these examples as Suda's flash memory controlling system "updating the index of mappings" by 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 (the claimed "empty-block identifier"). Baker, *Id*.

Second, a POSITA would have understood Suda to teach updating the erasure area pointers storage area to indicate that the data associated with the file on one or more of the physical NAND flash storage locations does not need to be preserved. Ex. 1003, Fig. 8 (updating at step S4); Baker, ¶ 118. 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, ¶ 118.

B. Claim 10

A POSITA would have understood Suda to teach that "the empty-block identifier indicates that the data associated with the file on the one or more physical NAND flash storage locations identified by the empty-block identifier should be overwritten." Suda teaches that upon receiving an erase command designating a logical address, start and end erasure area pointers will collectively designate a range of addresses as virtually erased and "to be erased" from the flash memory. *Id.*, 5:19-27, 5:36-53, 8:66-9:3. Specifically Suda teaches:

> Then, the flash memory controlling section 11 determines whether or not the address range indicated by the erasure area pointer

> designated in step S4 is coincident with the size of a physical block to be subjected to data erasure, i.e., whether or not a start address indicated by the erasure area pointer is a first page address of the physical block to be subjected to data erasure, and an end address indicated by the erasure area pointer is a last page address of the physical block to be subjected to data erasure (step S5).
>
> When the flash memory controlling section 11 determines in step S5 that the above address range is coincident with the size of the physical block subjected to data erasure, it erases a physical block address given to the above physical block and data indicating a logical block address related to the physical block address (step S6).

Ex. 1003, 5:56-6:2. As stated by Suda: when the Erase Command indicates an "address range [that] is coincident with the size of the physical block subjected to data erasure, it erases a physical block address" and therefore the Erase Command "indicates that the data associated with the file on the one or more physical NAND flash storage locations identified by the empty-block identifier ***should be "overwritten***" as claimed. *Id*. Thus, a POSITA would have understood Suda to teach this element. Baker, ¶ 120.

C. Claim 11

Suda teaches "transmitting a message indicating that the data associated with the file on one or more physical NAND flash storage locations identified by the empty-block identifier has been overwritten." Suda teaches that "[a]fter the step S6 or S7 are carried out, the flash memory controlling section 11 ***outputs*** a control signal ***indicating that data erasure processing is completed***, to the host device 2

through the host interface section 12 (step S8)." Ex. 1003, 8:17-20. Thus, a POSITA would have understood that Suda teaches this claim limitation. Baker, ¶ 121-122.

D. Claim 12

A POSITA would have understood Suda to teach this element. Suda discloses erasing a physical block:

> When the flash memory controlling section 11 determines in step S5 that the above address range is coincident with the size of the physical block subjected to data erasure, it erases a physical block address given to the above physical block and data indicating a logical block address related to the physical block address (step S6).

Ex. 1003, 7:64-8:2. A POSITA would know that erasing data in flash memory requires writing what Suda refers to as "initial-value data" to set the memory in an unused state, which would be a string of empty values (typically all 1's). *Id.,* 3:58-59, 4:4-6; Baker, ¶ 123. A POSITA would understand, for example as evidenced by the Bennett prior art discussed herein, that writing FFs or 00s to the sector flag erases data by overwriting the data to an initial value state so that it is unrecoverable. *See* § X.A.f. A POSITA would also have understood that during the typical process of garbage collection in these systems, data is overwritten with a series of characters such that the overwritten data is non-recoverable. Baker, ¶¶ 123. These processes are admitted by the Challenged Patent in in the background where it states: "[a]nother method of erasing data is to write zeros, ones, or some other null data character to the data storage device." Ex. 1001, 1:38-40, 1:57-61. For these reasons,

a POSITA would have understood that Suda teaches this element as claimed. Baker, ¶¶ 123-124.

E. Claim 13

Suda teaches "performing garbage collection, via the NAND controller, in the NAND flash-storage system." First, 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, ¶ 125; Ex. 1003, 1:54-55. Instead, Suda discloses garbage collection in terms of designating that the physical address previously assigned to the identified logical address comprises data suitable for removal from the solid-state storage medium. Specifically, 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." *E.g., id.*, 5:40-48, 5:57-61, 6:9-14, 6:29-45, 6:60-64, 7:38-41; Baker, ¶ 126. 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. Baker, ¶ 126. Suda's blocks will be physically erased 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 the reasons discussed in claim element 7[e], it is the NAND controller that performs these tasks, and the storage system at issue is the NAND flash storage system. *See* § IX.A.e. For these reasons, a POSITA would have understood that Suda teaches the garbage collector as claimed. Baker, ¶¶ 125-127.

F. Claim 14

a) Element 14[a]

Suda teaches "copying a block containing data that should be preserved from an erase block." Suda provides an example in Fig. 6 where "an erase command to erase data items written to physical blocks A, B, and C is issued from the host device **2**." Ex. 1003, 6:22-26, Fig. 6. With respect to block A in Fig. 6, Suda instructs, "the flash memory controlling section **11** performs such data erasing and copying processing as shown in FIG. 2 on an area **24** of the physical block A, the data items of which are to be erased." Ex. 1003, 6:29-32. In Fig. 2, Suda provides an example where data items written to pages 0 to 31 are to be erased while following pages are to be preserved. Suda discloses, "[w]hen the data items of the pages 'Page0' to 'Page31' are erased, data items written to the pages following 'Page32' in the physical block having physical block address '3' are also erased along with

the data items of the pages 'Page0' to 'Page31'" as "data erasure is carried out in units of one physical block." Ex. 1003, 4:33-38. Suda further instructs that the data items written to the pages following "Page31" in the physical block having physical block address "3" are copied to another physical block given physical block address "4." Ex. 1003, 4:39-43. Thus, a POSITA would have understood Suda to teach this element. Baker, ¶¶ 128-129.

b) Element 14[b]

Suda teaches "erasing the erase block" in a garbage collection process whenever the erasure area pointers indicate an entire block contains virtually erased data. For example, Suda instructs to perform "[e]rasure processing on entire block" in step S6 of Fig. 8 after the erasure area pointer is updated (S4) such that it equals a block size (S5). In a second example, with respect to the block illustrated in Fig. 4, Suda instructs that virtually erased, physical block (area 22) is "to be erased" and to "be set in an unused state." Ex. 1003, 5:65-6:3, 3:56-67 (explaining that blocks in an unused state have "initial-value" data and "can be used" again). In a third example, with respect to physical block B in Fig. 6, Suda discloses, "the data items written to the entire area of the physical block B are to be erased," and further instructs "setting the entire area (area 25) of the physical block B in an unused state." Ex. 1003, 6:33-41. A POSITA would have understood these parts of Suda to teach erasing the erase block, as claimed. Baker, ¶¶ 130-131.

G. Claim 15

Suda teaches "writing the copied data, via the NAND controller, to the erased erase block. Suda's system requires 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, ¶ 132. A POSITA would understand physical blocks to become unused following an erasure as discussed in claim 14[b]. Thus, a POSITA would have understood that Suda teaches this element. Baker, ¶ 132.

H. Claim 16

A POSITA would understand that Suda teaches that "the NAND flash storage system is an append-only storage system." As discussed for claim 13, 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." Baker, ¶ 133; Ex. 1003, 1:54-55. In addition, as described for claim 7[e], Suda's process of maintaining an index of mappings leaves data unaltered. *Id.*, 221-22; Ex. 1003, Fig. 1. Thus, a POSITA would have understood Suda to teach this element. Baker, ¶ 133.

I. Claim 17

A POSITA would understand that Suda teaches that "each of the one or more physical NAND flash storage locations are smaller than the erase block." A POSITA would understand that flash memories are erased in blocks. Baker, ¶ 134. Each block is made up of multiple physical NAND flash storage locations smaller than the erased block. Ex. 1003, 8:47-48; Baker, ¶ 134. Thus, a POSITA would understand that each of the physical NAND flash storage locations are smaller than the erase block. Baker, ¶ 134.

X. Ground 2: Obvious Over Bennett and POSITA Knowledge

A. Claim 7

a) Element 7[a] [2]

Bennett is titled "Methods For The Management Of Erase Operations in Non-Volatile Memory" (the claimed "method for managing data in a system"). Ex. 1002, Title. Within a host 10, Bennett discloses a "host-side memory manager" (the claimed "host processor"). *See* Ex. 1002, Fig. 2 (copied below); *id.*, 7:27-29 ("In some host systems, an optional host-side memory manager may exist to perform lower level memory management at the host").

[2] *See* attached Claim Listing.

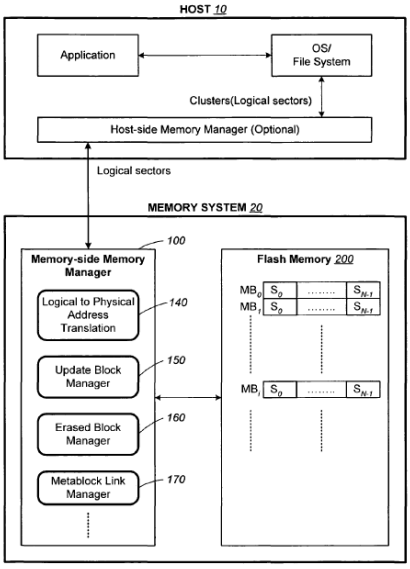


FIG. 2

Id., Fig. 2. Bennett discloses "Flash Memory 200," and discloses using "NAND" (the claimed "block-based NAND flash storage system"). Ex. 1002, Fig. 2; *id*., 2:44-46. Bennett discloses that this memory can be block-based. Ex. 1002, 5:15-18. A POSITA would understand that NAND flash has physical NAND storage locations controlled by a NAND processor. Baker, ¶ 136. Bennett discloses this NAND controller as "controller 100" as depicted in Figure 2 copied above. Ex. 1002, Fig. 2. Additionally, Bennett teaches that the memory system's operations are "controlled by a controller 100." *Id.,* 6:62-63; Baker, ¶ 137. A POSITA would understand that Bennett's controller controls "NAND physical flash storage locations" as claimed because it implements storage operations on the flash memory 200. Baker, ¶ 138. 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[3] … [and] maintains 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, ¶¶ 135-139.

b) Element 7[b]

A POSITA would understand that Bennett renders obvious receiving, via the host processor, an indication of deletion of a file at the host processor.

Bennett teaches "[t]he host 10 accesses the memory 200 when running an application under a file system or operating system." Ex. 1002, 7:21-22. A POSITA would have understood that file systems store logical address and memory systems attached to hosts store data at physical addresses. Baker, ¶ 140; *see* Ex. 1002, 7:22-23 ("Typically, the host system addresses data in units of logical sectors"). A POSITA would also understand that, as the '359 Patent itself recognized in its "Description of the Related Art," "[i]n many file systems, an erase command deletes a directory entry in the file system." Ex. 1001, 1:34-36. Bennett teaches "a host

[3] Bennett's "FIG. 2 illustrates the memory being organized into physical groups of sectors (or metablocks) and managed by a memory manager of the controller, according to a preferred embodiment of the invention." Ex. 1002, 5:15-18.

issu[ing] a command to erase a portion of the memory, such as an Erase Sectors command that specifies the logical sectors to erase." Ex. 1002, 12:15-17, 4:22 ("a host issues an Erase Sectors command"), Fig. 1 (showing host computer 10). From these disclosures, a POSITA would understand that Bennett teaches "receiving, via the host processor, an indication of deletion of a file at the host processor" as claimed. Baker, ¶ 140-141.

Alternatively, a POSITA would also know that the host processor will receive an indication of deletion of a file. A POSITA would have 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.* A POSITA would also understand that upon a user specifying a file name to be deleted, operating systems access the OS/file system to delete mapping of file names being deleted. Baker, ¶ 141. As shown in Benett's Figure 2, the file system is connected to the "host-side memory manager." Bennett Fig 2. As a result of these connections, a POSITA would understand the process of Bennett to be 1) the application notifies the OS/File system of a file being deleted; 2) an indication is transferred to the host-side memory manager of that deletion in order to generate an erase command sent to memory. Baker, ¶ 141. A POSITA would have understood these disclosures to teach that an indication of deletion is received at the host-side memory manager (the claimed "host processor"). Baker, ¶¶ 140-141.

c) Element 7[c]

A POSITA would understand that Bennett renders obvious creating, via the host processor, an empty-block identifier in response to the indication of deletion. Specifically, a POSITA would have understood that Bennett's erase command is sent by the host-side memory manager in response to user interactions with an "application" to delete a document. Baker, ¶ 142; Ex. 1002, Fig. 2 (depicting components). 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. A POSITA would understand that Bennett's erase commands indicates that the file at the host processor is deleted. Baker, ¶ 143. Bennett teaches that the data will be "physically erased or otherwise subjected as a whole to an erase operation." Ex. 1002, 4:12-18. Thus, it would have been obvious to a POSITA that Bennett's erase commands, by identifying logical block addresses to be erased, creates an empty-block identifier as claimed. Baker, ¶¶ 142-144.

d) Element 7[d]

A POSITA would understand that Bennett teaches this claim element. Bennett teaches "receiving, via the NAND controller, an empty-block identifier" 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"). As depicted in Bennett's Fig. 2, the erase command is received from the host through the controller's memory-side memory manager (the claimed "controller").

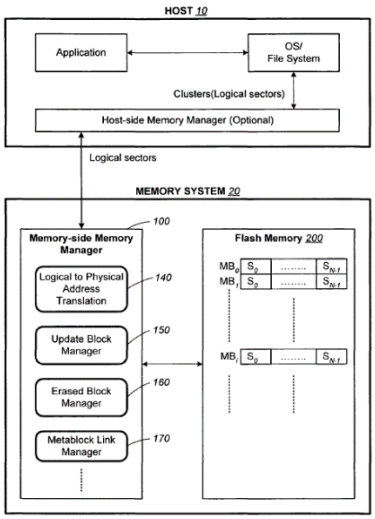


FIG. 2

Ex. 1002, Fig. 2. As discussed for claim 7[c], Bennett's erase command is an empty-block identifier. *See* § IX.A.c. Thus, a POSITA would understand that the empty-block identifier is received via the NAND controller as claimed. Baker, ¶¶ 145-147.

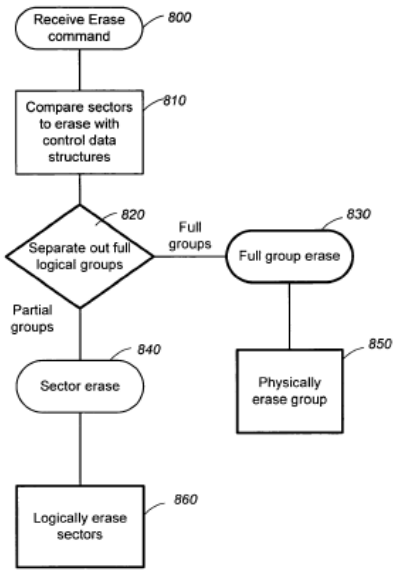
e) Element 7[e]

Bennett teaches "maintaining, via the NAND controller, an index of mappings between logical identifiers and physical NAND flash storage locations" as claimed. 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 140" module,

analogous to the claimed index of mappings between logical identifiers and physical NAND flash storage locations. *Id.* This module "is responsible for relating a host's logical 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, ¶ 148.

f) Element 7[f]

Bennett teaches "updating the index of mappings, in response to the receiving the empty-block identifier, to indicate that data associated with the file on one or more of the physical NAND flash storage locations identified by the empty-block identifier does not need to be preserved." Bennett teaches storing an index mapping of logical-to-physical addresses in a GAT stored in flash memory. Ex. 1002, 10:37-39. Bennett further teaches "marking sectors as erased" in response to an erase command. *Id.*, 20:20-61. For logical erasures: Bennett teaches 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, ¶ 149. Bennett further teaches "indicat[ing] that data associated with the file on one or more of the physical NAND flash storage locations identified by the empty-block identifier does not need to be preserved." 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 need not be preserved and can be later "physically erased when convenient, for example in a background process." *Id.*, 3:39-41.

Alternatively, a POSITA would understand that "updating the index of mappings" is obvious. Baker, ¶ 150. 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."). Thus, a POSITA would understand that Bennett teaches this claim element. Baker, ¶¶ 149-151.

B. Claim 10

A POSITA would have understood Bennett to teach "wherein the empty-block identifier indicates that the data associated with the file on the one or more physical NAND flash storage locations identified by the empty-block identifier should be overwritten." Specifically, Bennett teaches:

> When an erase command is received, the specified sectors are checked against the memory system's control data. If the specified sectors span any full logical grouping, the full logical groupings can each be treated as a whole and erased according to one process (such as performing a true, physical erase), while other sectors are "logically" erased at the sector level by standard techniques.

Ex. 1002, Abstract. As such, when the Bennett's Erase Command specifies a "full logical group" it is indicating that the data stored in the flash storage locations should be overwritten. The overwriting process involves writing FFs or 00s to the sector and setting an erased flag. *Id.*, 20:20-28. A POSITA would understand that Bennett's disclosure of responding to erase commands, specifying a full logical group, by actually erasing the physical memory, returns the memory to a point where it can be written again. Baker, ¶ 152. Bennett further teaches keeping a cleared block list that contains a list of physical block addresses that have been erased and are thus available for storing new data. *Id.*, 10:27-33. Thus, a POSITA would understand that Bennett teaches this element. Baker, ¶¶ 152-153.

C. Claim 11

A POSITA would have understood Bennett to teach this element. For example, Bennett teaches marking sectors as erased by "marking full logical groups as erased, where the logical group has all sectors written with FF or 00 data and an 'erased' flag set in corresponding headers. Ex. 1002, 20:20-28 (The logical group can again be associated with an MS block.) Bennett discloses returning status information regarding "whether the sector being read was erased by an Erase Sectors command or not." Ex. 1002, 4:37-40. Bennett further teaches that "an erased block (or erased sector) status can be returned if a sector belonging to a block (or the sector itself) is erased"). *Id.*, 4:40-42. Thus, a POSITA would understand that Bennett teaches this element. Baker, ¶ 154-155.

D. Claim 12

A POSITA would have understood that Bennett teaches this element. 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, ¶ 156. A POSITA would understand that the data associated with the file on the deleted physical address locations would be "non-recoverable" as claimed. Baker, *Id*.

E. Claim 13

Bennett teaches "performing garbage collection, via the NAND controller, in the NAND flash storage system." *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, ¶¶ 64-66, 156; *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"). Bennett teaches that erasure is managed by the memory-side memory manager (the claimed "NAND controller"). *Id.* 7:37-39 ("the memory manager contains a number of software modules for managing erase of the metablocks"). In addition, a POSITA would have understood that Bennett discloses a garbage collector for recovering space for a system that does not support update-in-place of data in the storage medium. Baker, ¶ 158. 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 an erase command. For example, a POSITA 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, ¶ 159. 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, ¶ 159. 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 14

a) Element 14[a]

Bennett teaches the claimed "copying a block containing data from an erase block" by rewriting sectors within a GAT block to a new block location. For example, Bennett discloses that "a new GAT block is allocated, and valid GAT sectors as defined by the GAT index are copied in sequential order from the full GAT block. *Id.* 12:6-11. Bennett teaches that this occurs while performing garbage collection:

> The following sequence executed by the media management layer can provide a safe mechanism for erasing a logical group:
> 1. Open a new Update Block for the Logical Group. If the current number of Update Blocks is up to the limit, then close (garbage collect) another Update block. If the Logical Group has an open Update block, then force closure of the Update block which belongs to the Logical group.
> 2. Write at least one sector to the new Update block with any data and "erased" flag set in the header.
> 3. Erase the Original block of the Logical Group.
> 4. Update GAT marking the Logical Group as physically erased. Update erase block management (EBM) 760 (FIG. 18 a) by adding the erased block to the erase pool. Perform EBM-MAP exchange if EBL get full as the erase pool increases.
> 5. Erase the new Update Block of the Logical Group. There is no need to maintain it in control update, as this block is not referenced

> by any data structure and is meant to be erased.
> 6. As a result, all the block associated with the Logical Group are erased. If power loss happens in the middle of this sequence, the "erased" sector can be found and the operation can be completed.

See, e.g., *id.* 19:5-28. Thus, A POSITA would understand that Bennett teaches this element. Baker, ¶ 160.

b) Element 14[b]

Bennett teaches "erasing the erase block" during the garbage collection process. Bennett's Fig. 10 illustrates the process flow for erase commands. Ex. 1002, Fig. 10. 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. Bennett further teaches erasing blocks during garbage collection in steps 3-6 of the sequence described for Claim 14[a]. *See* § X.F.a (explaining sequence executed by the media management layer for erasing logical group). A POSITA would have understood the erasing as a process to reclaim blocks to be written to again. Baker, ¶ 161. Thus, a POSITA would understand that Bennett teaches this element. Baker, ¶ *Id.*

G. Claim 15

Bennett teaches "writing copied data to the erased erase block" during the garbage collection process. *See* § X.F. Bennett further discloses writing to erased

storage. Ex. 1002, 3:15-17 ("the entire erase block contain that physical location will have to be first erased and then rewritten with the updated data"). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶ 162.

H. Claim 16

A POSITA would understand that Bennett teaches this claim element. Provisional application no. 60/873111 discloses that the "append-only methodology of storing data is well suited for NAND flash, since it is not feasible to update (change) data in-place". Ex. 1009, pg. 63. Similarly, as discussed for claim 13, Bennett does not support update-in-place of data in the storage medium. Ex. 1002, 3:11-17 ("If data of a certain logical address from a host is to be updated, one way is rewrite the update data in the same physical memory location… this will mean the entire erase block contain that physical location will have to be first erased and then rewritten with the updated data. This method of update is inefficient"). In addition, as described for claim 7[e], Bennett's process of maintaining an index of mappings leaves data unaltered. Ex. 1002, Fig. 6, 9:50-52, 10:36-38. Thus, a POSITA would have understood Bennett to teach this element. Baker, ¶ 163-164.

I. Claim 17

Bennett teaches that entire erase blocks "contain" physical locations. *Id*. 3:14-17. Further, it teaches:

> FIG. 4 illustrates the alignment of a metablock with structures in physical memory. Flash memory comprises blocks of memory cells

> which are erasable together as a unit. Such erase blocks are the minimum unit of erasure of flash memory or minimum erasable unit (MEU) of the memory. The minimum erase unit is a hardware design parameter of the memory, although in some memory systems that supports multiple MEUs erase, it is possible to configure a "super MEU" comprising more than one MEU. For flash EEPROM, a MEU may comprise one sector but preferably multiple sectors. In the example shown, it has M sectors. In the preferred embodiment, each sector can store 512 bytes of data and has a user data portion and a header portion for storing system or overhead data. If the metablock is constituted from P MEUs, and each MEU contains M sectors, then, each metablock will have N=P*M sectors.
>
> The metablock represents, at the system level, a group of memory locations, e.g., sectors that are erasable together. The physical address space of the flash memory is treated as a set of metablocks, with a metablock being the minimum unit of erasure. Within this specification, the terms "metablock" and "block" are used synonymously to define the minimum unit of erasure at the system level for media management, and the term "minimum erase unit" or MEU is used to denote the minimum unit of erasure of flash memory.

*Id.*at 8:31-55.

Further, Fig. 4 demonstrates NAND flash storage locations smaller than the erase blocks.

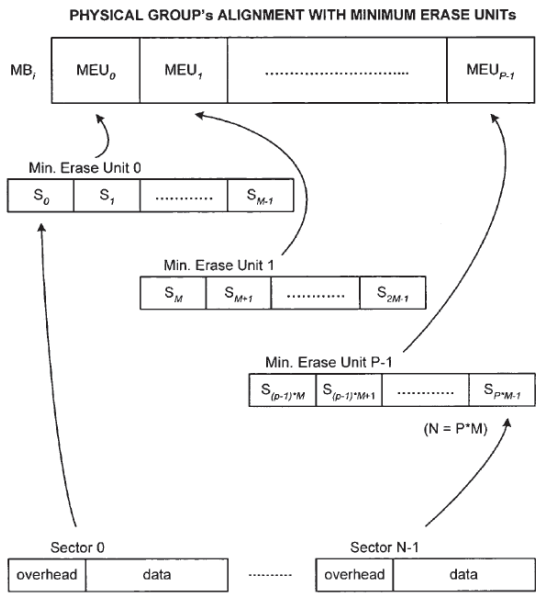


FIG. 4

Ex. 1002, Fig. 4. For example, "sector 0" (a "NAND flash storage location" as claimed) is grouped among other sectors within a larger "Min. Erase Unit 0" (an "erase block" as claimed). Thus, a POSITA would have understood that Bennett teaches this element. Baker, ¶ 165-166.

XI. Ground 3: A POSITA Would Have Found Claim 11 Obvious Over Suda and/or Bennett In Further View Of Zipprich and Knowledge of a POSITA.

Claim 11 depends on Claim 7, which would have been obvious to a POSITA for the reasons discussed under Ground 1. *See* § IX.A. To the extent that the Board does not find that Bennett and/or Suda (as discussed in Grounds 1 and 2) do not explicitly show transmitting a message indicating that the data associated with the file on one or more physical NAND flash storage locations identified by the empty-

block identifier has been overwritten, either of these references in view of Zipprich, render claim 11 obvious.

Zipprich teaches a "status report" that is generated to indicate the status of an overwrite. Ex.1006, [0018]. The report provides immediate feedback and logging/tracking of the overwrite events. *Id*. Zipprich teaches a report in the form of an e-mail message sent to an email address specified by a user or administrator. *Id*., [0040]. Zipprich also teaches a report in the form of an entry in a log file. *Id*., [0041].

A POSITA would have found it obvious to combine Zipprich's teachings with either Bennett or Suda. Baker, ¶ 167. Modification of Bennett and/or Suda to incorporate transmitting a report was within the skillset of one of ordinary skill and would improve performance by providing additional information about the system erasures. Baker, *Id*. Including a Zipprich message would have informed the host-side memory manager in Bennett or the digital camera in Suda that a physical block address is available for storing new data. Baker, ¶ 168. This message would have further increased the ability of these devices to control physical flash storage locations. Baker, *Id*. Thus, Claim 11 would have been obvious to a POSITA in view of Bennett and Zipprich.

XII. Ground 4: A POSITA Would Have Found Claims 7, 10-13 Obvious Over Bennett In View of Suda

To the extent That Bennett is not found to disclose Claim 7[b], a POSITA would still have found Claim 7[b] obvious based on Bennett and Suda.

Suda discloses an example of a host device, a digital camera, for which users can delete photos. Ex. 1003, 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, ¶ 169. Suda teaches that when a user selects a photo to delete, the digital camera looks to this cache for the corresponding logical identifier and designates this logical identifier in an erase command. Ex. 1003, 8:66-9:3.

It would have been obvious to a POSITA that the host processor in Bennett's system would have operated similarly to the host processor in the digital camera described in Suda, or that the host processor could operate similarly to the host processor taught in Suda. Baker, ¶ 170. This operation would have enabled Bennett's host processor to identify a logical address when issuing an erase command in a manner that is simple, efficient and commonplace in these types of devices. Ex. 1003, 8:66-9:3. Thus, claims 7 and 10-13 would have been obvious to a POSITA in view of Suda and Bennett.

XIII. Ground 5: A POSITA Would Have Found Claims 13-15 Obvious Over Bennett and/or Suda in View of SwSTE'05

To the extent that Suda and/or Bennett is not found to disclose Claims 13-15, a POSITA would still have found claims 13-15 obvious in view of SwSTE'05.

SwSTE'05 explicitly teaches a garbage collection process in its bulleted steps in Section 2.3. Ex. 1007. One of these steps is copying valid sectors to newly allocated free space. *Id.* § 2.3. A POSITA would have found it obvious to apply this step to Suda and/or Bennett's system, recognizing that the "valid sectors" are Suda and/or Bennett's "data," and that the "newly allocated free space" are the "erased blocks" that have been restored to a point where they can be written to again. Baker, ¶ 171-172. A POSITA would have been motivated to combine SWSTE'05's garbage collection teachings, and had a reasonable expectation of success, because garbage collection was efficient, well-known and had been a standard part of flash memory management since the mid-1990s. Baker, *Id*. Thus, claims 13-15 would have been obvious to a POSITA in view of Bennett and SwSTE'05.

XIV. 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, the FTL and garbage collection process were already well-known in the art. *See, e.g.,* Ex. 1007 § 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 1011-1013 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.

XV. The Board Should Reach the Merits of This Petition

A. Section 325(d) Does Not Apply Because the Petition Relies on New Art and Arguments

The Board should not deny institution under 35 U.S.C. § 325(d). The Board has recently provided a two-step framework that asks: (1) whether the same or substantially the same art or arguments previously were presented to the Office; and (2) if part (1) of the framework is satisfied, whether the Petitioner has demonstrated that the Patent Office erred in a manner material to the patentability of the claims. *Advanced Bionics, LLC v. Med-El Elektromedizinische Gerate GmbH*, IPR2019-

01469, Paper 6, pp 6-11 (PTAB Feb. 12, 2020). This two-step process considers the six factors set forth in *Becton, Dickinson & Co. v. B. Braun Melsungen AG*, IPR2017-01586, Paper 8 (PTAB Dec. 15, 2017) (precedential as to § III.C.5, first paragraph). The *Beckton Dickinson* factors (a), (b), and (d) are relevant to step (1), while factors (c), (e), and (f) are relevant to step (2). *Advanced Bionics* at 10.

In *Advanced Bionics*, the Board noted that misapprehending or overlooking specific teachings that impact patentability is an example of a material error by the Patent Office. *Id.* at 8 (fn. 9). In relating the analysis to the *Becton Dickinson* factors, the Board stated that "if the record of the Office's previous consideration of the art is not well developed or silent, then a petitioner may show the Office erred by overlooking something persuasive under factors (e) and (f)." *Id.* at 10.

Step 1 Analysis: Suda and Bennett were cited in information disclosure statements as two out of hundreds prior art references cited on the face of the '359 Patent. However, these references were not substantively cited by the Examiner, did not form the basis for any rejection, and were not discussed during prosecution. "The Board has frequently held that, when a reference "was neither applied against the claims nor discussed by the Examiner," merely citing the reference during prosecution does not weigh in favor of exercising the Board's discretion under § 325(d) to deny a petition." *Weber, Inc. v. Provisur Technologies, Inc.*, IPR2019-01467, Paper 7 at 10 (PTAB Feb. 14, 2020) citing *Zip-Top LLC v. Stasher, Inc.*,

IPR2018-01216, Paper 14 at 35–36 (PTAB Jan. 17, 2019); *see also Apple Inc. v. Qualcomm Inc.*, IPR2018-01316, Paper 7 at 25 (PTAB Jan. 18, 2019) ("The fact that neither [Applicant Admitted Prior Art] nor Majcherczak was the basis of rejection weighs strongly against exercising our discretion to deny under 35 U.S.C. § 325(d).").

Step 2 Analysis: With respect to factor (c), there is no evidence that Suda or Bennett were substantively evaluated during prosecution as three out of numerous cited references disclosed on the face of the patent. *Becton*, IPR201701586, Paper 8 at 17-18; *see also Solaredge Techs. Ltd. v. SMA Solar Tech. AG*, IPR2020-00021, Paper 8 at 12 (PTAB Apr. 10, 2020); *Fasteners for Retail, Inc. v. RTC Industries, Inc.*, IPR2019-00994, Paper 9 at 9 (PTAB Nov. 5, 2019) (instituting review, explaining that "the examiner was left considering over 1,000 prior art documents . . . we are skeptical that the examiner was able to devote sufficient time to evaluate all of the asserted art in detail during prosecution"). The Examiner either overlooked these disclosures or erred in interpreting them because, as detailed above, Bennett or Suda alone and in view of a POSITA's knowledge, or Bennett in view of Zipprich and a POSITA's knowledge, or Suda in view of Zipprich or Bennett and a POSITA's knowledge, render obvious the challenged claims of the '359 patent, and thus materially impacts patentability of the claims of the '359 patent. Petitioner's

detailed analysis of Suda and Bennett warrants reconsideration of the patentability of the challenged claims of the ’359 patent.

Further underscoring the error of the Patent Office is this Board’s determination in *Micron Technologies Inc. v. Unification Technologies LLC*, IPR2021-00345 that claims that are substantively identical to at least one of the claims challenged here were unpatentable over Suda or a combination of Suda and Bennett.[4] Paper 41, Final Written Decision. For example, this Board held claim 12 of U.S. Patent No. 9,632,727 (the “’727 Patent”) was unpatentable over Suda alone. Claim 12 of the ’727 maps closely to claim 7 of the ’359 patent, as demonstrated below:

[4] The Petitioners also argued alternatively that the claims were unpatentable over Bennett alone, but the Board did not reach that argument. *Micron*, IPR2021-00345, Paper 41 at 10.

Claim 12 of the ’727 Patent	**Claim 7 of the ’359 Patent**
A non-volatile solid-state storage system, comprising: a storage interface configured to communicate with a storage client; a **storage processor** coupled to the storage interface; a **flash memory device** coupled to the storage processor; and **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, 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, **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.	A method for managing data in a system having a host processor and a block-based **NAND flash storage system**, the NAND flash storage system having physical NAND flash storage locations controlled by a **NAND controller**, the method comprising: receiving, via the host processor, an indication of deletion of a file at the host processor; creating, via the host processor, an **empty-block identifier** in response to the indication of deletion; receiving, via the NAND controller, the empty-block identifier; maintaining, via the NAND controller, **an index of mappings between logical identifiers and physical** NAND flash storage locations; and **updating the index of mappings**, in response to the receiving the empty-block identifier, to indicate that data associated with the file on one or more of the physical NAND flash storage locations identified by the empty-block identifier **does not need to be preserved**.

The fact that this Board held invalid such closely related claims over the same references presented here demonstrates that the PTO erred when granting the '359 Patent. Consequently, weighing the *Becton Dickinson* factors, the petition thus satisfies Step 2 of the Advanced Bionics analysis and Petitioner respectfully asks the Board to reach the merits and institute the Petition.

B. The Board Should Not Discretionarily Deny Institution Under § 314(a)

Under the Director's interim procedure for evaluating requests for discretionary denial under 35 U.S.C. § 314(a), regardless of any factors that otherwise apply, "the PTAB will not discretionarily deny institution of an IPR or PGR in view of parallel district court litigation where a petitioner stipulates not to pursue in a parallel district court proceeding the same grounds as in the petition or any grounds that could have reasonably been raised in the petition." Interim Procedure for Discretionary Denials in AIA Post-Grant Proceedings with Parallel District Court Litigation, 7 (USPTO Dir. June 21, 2022).

Petitioner stipulates that, should the Board institute review, Petitioner will not pursue in parallel district court proceedings the same grounds as in the Petition or any grounds that could have reasonably been raised in the petition. Thus, the Board should not exercise discretion to deny institution under §314(a).

XVI. Mandatory Notices

A. Real Parties-in-Interest

The named Petitioner is the only entity who is funding and controlling this Petition and is therefore named as a real party-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.

B. Related Matters

No previous IPR or post-grant proceeding has been filed challenging the '359 Patent. The Patent Owner has asserted the '359 Patent in *Unification Technologies LLC v. Phison Electronics Corporation*, 2-23-cv-00266 (lead case), *Unification Technologies LLC v. Silicon Motion Inc.*, 2-23-cv-0267 (member case).

C. Lead and Backup Counsel

The following lead and backup counsel represent Petitioner:

Lead Counsel for Petitioner	Backup Counsel for Petitioner
Ya-Chiao Chang Winston & Strawn LLP C/O Yuanda China Law Offices 2801 No. 88 Century Avenue Shanghai 200121 China ychang@winston.com T: 86.21.2208.2628, F: 86.21.5298.5262 *Lead Counsel for Petitioner* *Silicon Motion Inc.*	Michael Rueckheim Winston & Strawn LLP 255 Shoreline Drive, Suite 520 Redwood City, California 94065 mrueckheim@winston.com T: 650.858.6433, F: 650.858.6433 (to seek *pro hac vice* admission) ********** Evan D. Lewis Winston & Strawn LLP 800 Capitol Street, Suite 2400 Houston, TX 77002

	edlewis@winston.com T: 713.651.2785, F: 713.651.2700 (to seek *pro hac vice* admission)

XVII. Electronic Service

Petitioner consents to electronic service at:

Winston-IPR-Unification@winston.com

XVIII. Fees

Petitioner has paid the required fee electronically through P.T.A.B. E2E.

XIX. Conclusion

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

Dated: December 18, 2023 *Respectfully submitted,*

Ya-Chiao Chang
Winston & Strawn LLP
C/O Yuanda China Law Offices
2801 No. 88 Century Avenue
Shanghai 200121
China
ychang@winston.com
T: 86.21.2208.2628,
F: 86.21.5298.5262
Lead Counsel for Petitioner
Silicon Motion Inc.

Michael Rueckheim
Winston & Strawn LLP
255 Shoreline Drive, Suite 520
Redwood City, California 94065
mrueckheim@winston.com
T: 650.858.6433, F: 650.858.6433
Back-up Counsel for Petitioner Silicon Motion Inc.

Evan D. Lewis
Winston & Strawn LLP
800 Capitol Street, Suite 2400
Houston, TX 77002
edlewis@winston.com
T: 713.651.2785, F: 713.651.2700
Back-up Counsel for Petitioner Silicon Motion Inc.

CLAIM LISTING

Claim 7	
Element	**Language**
7[a]	A method for managing data in a system having a host processor and a block-based NAND flash storage system, the NAND flash storage system having physical NAND flash storage locations controlled by a NAND controller, the method comprising:
7[b]	receiving, via the host processor, an indication of deletion of a file at the host processor;
7[c]	creating, via the host processor, an empty-block identifier in response to the indication of deletion;
7[d]	receiving, via the NAND controller, the empty-block identifier;
7[e]	maintaining, via the NAND controller, an index of mappings between logical identifiers and physical NAND flash storage locations; and
7[f]	updating the index of mappings, in response to the

	receiving the empty-block identifier, to indicate that data associated with the file on one or more of the physical NAND flash storage locations identified by the empty-block identifier does not need to be preserved.

Claim 10	
Element	**Language**
10	The method of claim 7, wherein the empty-block identifier indicates that the data associated with the file on the one or more physical NAND flash storage locations identified by the empty-block identifier should be overwritten.

Claim 11	
Element	**Language**
11	The method of claim 7, further comprising transmitting a message indicating that the data associated with the file on one or more physical NAND flash storage locations identified by the empty-block identifier has been

	overwritten.

Claim 12	
Element	**Language**
12	The method of claim 7, further comprising overwriting the data associated with the file on the one or more physical NAND flash storage locations identified by the empty-block identifier with a series of characters such that the overwritten data is non-recoverable.

Claim 13	
Element	**Language**
13	The method of claim 7, further comprising performing garbage collection, via the NAND controller, in the NAND flash storage system.

Claim 14	
Element	**Language**
14[a]	The method of claim 13, wherein performing garbage

	collection comprises: copying a block containing data that should be preserved from an erase block comprising blocks containing data that does not need to be preserved; and
14[b]	erasing the erase block.

Claim 15	
Element	**Language**
15	The method of claim 14, further comprising writing the copied data, via the NAND controller, to the erased erase block.

Claim 16	
Element	**Language**
16	The method of claim 7, wherein the NAND flash storage system is an append-only storage system.

Claim 17	
Element	**Language**

17	The method of claim 14, wherein each of the one or more physical NAND flash storage locations are smaller than the erase block.

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 11,315 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 18, 2023

Respectfully submitted,

/ Ya-Chiao Chang /
Ya-Chiao Chang
Winston & Strawn LLP
C/O Yuanda China Law Offices
2801 No. 88 Century Avenue
Shanghai 200121
China
ychang@winston.com
T: 86.21.2208.2628,
F: 86.21.5298.5262
Lead Counsel for Petitioner
Silicon Motion Inc.

CERTIFICATE OF SERVICE

Under 37 C.F.R. §§ 42.6(e) and 42.105(a), this is to certify that on December 18, 2023, I caused to be served a true and correct copy of the foregoing "**PETITION FOR *INTER PARTES* REVIEW OF CLAIMS 7 AND 10-17 OF U.S. PATENT NO. 11,640,359,**" Petitioner Power of Attorney and Exhibits 1001-1026 by FedEx on the Patent Owner at the correspondence address of record for U.S. Patent No. 11,640,359:

Unification Technologies LLC
6136 Frisco Square Boulevard
Suite 400
Frisco, TX

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

[David T. DeZern]
Nelson Bumgardner Conroy PC
3131 W. 7th Street, Suite 300
Fort Worth, TX 76107
Email: david@nelbum.com

WINSTON & STRAWN LLP

/ Ya-Chiao Chang /
Ya-Chiao Chang
Winston & Strawn LLP
C/O Yuanda China Law Offices
2801 No. 88 Century Avenue
Shanghai 200121
China
ychang@winston.com
T: 86.21.2208.2628,

F: 86.21.5298.5262
Lead Counsel for Petitioner
Silicon Motion Inc.