

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SAMSUNG ELECTRONICS CO., LTD. AND SAMSUNG ELECTRONICS
AMERICA, INC.,
Petitioner,

v.

RADIAN MEMORY SYSTEMS LLC,
Patent Owner.

U.S. Patent No. 11,681,614

“Storage Device With Subdivisions, Subdivision Query, And Write Operations”

EXHIBIT 1003
DECLARATION OF DR. HEINER LITZ , PH.D

I.	SUMMARY OF OPINIONS	1
II.	QUALIFICATIONS	1
III.	LEGAL PRINCIPLES AND UNDERSTANDINGS	4
	A. Person of Ordinary Skill in the Art.....	4
	B. Claim Construction.....	5
	C. Anticipation	5
	D. Obviousness.....	6
IV.	LEVEL OF ORDINARY SKILL IN THE ART	9
V.	TECHNOLOGY BACKGROUND.....	9
VI.	OVERVIEW OF THE '614 PATENT	14
	A. Priority.....	14
	B. Specification	14
	C. Prosecution History.....	15
VII.	LEVEL OF ORDINARY SKILL IN THE ART	17
VIII.	CLAIM CONSTRUCTION	18
IX.	GROUND 1: CLAIMS 1-2, 4-8 ARE OBVIOUS IN VIEW OF ELLIS (EX1004) AND SON (EX1006).....	18
	A. Overview of the Prior Art	18
	1. Ellis (EX1004).....	18
	2. Son (EX1006).....	21
	B. Claim Analysis	22
	1. Claim 1	22
	2. Claim 2	45

3.	Claim 4	46
4.	Claim 5	47
5.	Claim 6	47
6.	Claim 7	48
7.	Claim 8	49
X.	GROUND 2: CLAIMS 3, 10-14, 16-19, 21-29 ARE OBVIOUS IN VIEW OF ELLIS (EX1004), SON (EX1006) AND OLBRICH (EX1010)	50
A.	Overview of the Prior Art	50
1.	Olbrich (EX1010)	50
2.	Claim 3	52
3.	Claim 10	54
4.	Claim 11	58
5.	Claim 12	58
6.	Claim 13	58
7.	Claim 14	59
8.	Claim 16	60
9.	Claim 17	61
10.	Claim 18	62
11.	Claim 19	62
12.	Claim 21	62
13.	Claim 22	64
14.	Claim 23	64
15.	Claim 24	64

16.	Claim 25	65
17.	Claim 26	65
18.	Claim 27	66
19.	Claim 28	66
20.	Claim 29	66
XI.	GROUND 3: CLAIMS 9, 15, 20, 30, ARE OBVIOUS IN VIEW OF ELLIS (EX1004), SON (EX1006), OLBRICH (EX1010), AND YAMADA (EX1015).....	67
A.	Overview of the Prior Art	67
1.	Yamada (EX1015)	67
B.	Claim Analysis	68
1.	Claim 9	68
2.	Claim 15	75
3.	Claim 20	77
4.	Claim 30	77
XII.	GROUND 4: CLAIMS 1-2, 4-8 ARE OBVIOUS IN VIEW OF TAKEO (EX1012), SON (EX1006), AND SINCLAIR (EX1005).....	78
A.	Overview of the Prior Art	78
1.	Takeo (EX1012)	78
2.	Sinclair (EX1005)	80
B.	Claim Analysis	83
1.	Claim 1	83
2.	Claim 2	100
3.	Claim 4	101

4.	Claim 5	102
5.	Claim 6	102
6.	Claim 7	103
7.	Claim 8	104
XIII.	GROUND 5: CLAIMS 3, 10-13, 16-19, 21-29 ARE OBVIOUS IN VIEW OF TAKEO, SON (EX1006), SINCLAIR (EX1005), AND OLBRICH (EX1010).....	105
A.	Claim Analysis	105
1.	Claim 3	105
2.	Claim 10	107
3.	Claim 11	110
4.	Claim 12	110
5.	Claim 13	111
6.	Claim 14	112
7.	Claim 16	114
8.	Claim 17	115
9.	Claim 18	115
10.	Claim 19	115
11.	Claim 21	115
12.	Claim 22	117
13.	Claim 23	117
14.	Claim 24	117
15.	Claim 25	118
16.	Claim 26	118

17.	Claim 27	119
18.	Claim 28	119
19.	Claim 29	119
XIV.	GROUND 6: CLAIMS 9, 15, 20, AND 30 ARE OBVIOUS IN VIEW OF TAKEO, SON (EC), SINCLAIR (EX1005), OLBRICH (EX1010) AND YAMADA (EX1015)	121
A.	Claim Analysis	121
1.	Claim 9	121
2.	Claim 15	123
3.	Claim 20	125
4.	Claim 30	125
XV.	CONCLUSION	125

EXHIBIT LIST

EX1001	U.S. Patent No. 11,681,614 to Andrey V. Kuzmin et al. (“the ’614 patent”)
EX1002	Prosecution history of the ’614 patent (“’614 Prosecution History”)
EX1003	Expert Declaration of Dr. Heiner Litz.
EX1004	U.S. Patent 9,239,781 (“Ellis”)
EX1005	International Publication No. WO 2008/082996 (“Sinclair”)
EX1006	U.S. Pub. No. 2012/0096217 (“Son”)
EX1007	U.S. Pub. No. 2008/0189485 (“Jung”)
EX1008	Hu et al., Exploring and Exploiting Multilevel Parallelism Inside SSDs for Improved Performance and Endurance (2011)
EX1009	U.S. Pub. No. 2012/0303931 (“Asnaashari”)
EX1010	U.S. Patent 8,621,137 (“Olbrich”)
EX1011	Radian ’614 Infringement Contentions
EX1012	Certified Translation of JP2008176606A (“Takeo”)
EX1013	JP2008176606A
EX1014	Yang Hu et al., Performance Impact and Interplay of SSD Parallelism through Advanced Commands, Allocation Strategy and Data Granularity (2013)
EX1015	U.S. Pat. Pub. No. 2011/0271032A1 (“Yamada”)
EX1016	U.S. Pat. Pub. 2012/0284587 (“Yu”)
EX1017	U.S. Patent No. 9,423,970 (“Darragh”)

I, HEINER LITZ, Ph.D. hereby declare as follows:

1. My name is Heiner Litz. I have been retained to testify as an expert in this *Inter Partes* Review (“IPR”) on behalf of Samsung Electronics Co., Ltd. (“SEC”) and Samsung Electronics America, Inc. (“SEA”) (collectively, “Samsung”).

2. I have prepared this declaration in connection with Samsung’s related petition for IPR of U.S. Patent No. 11,681,614, which I am informed is being filed concurrently with this declaration. I have been asked to provide certain opinions relating to the patentability of the ’614 patent. Specifically, I have been asked to provide my opinion regarding whether claims 1-30 of the ’614 patent (“the Challenged Claims”) are invalid as obvious in view of the prior art.

I. SUMMARY OF OPINIONS

3. It is my opinion that the Challenged Claims are rendered obvious pursuant to the Grounds and the reasons discussed in this Declaration.

II. QUALIFICATIONS

4. My education, experience, and qualifications are detailed in my résumé, included as **Appendix 1** to this declaration. While I incorporate **Appendix 1** by reference, below is a brief summary of my background, including my background and experience relevant to this case.

5. I earned a degree in Computer Engineering from the University of

Mannheim, Germany, in 2006. In 2012, I completed a Ph.D. in Computer Engineering from the same institution, graduating Summa Cum Laude. Between 2006 and 2012, my graduate research focused on computer architecture, hardware chip design, scalable storage systems, and high-performance computing networks.

6. In 2013, I began my academic career as a Postdoctoral Researcher at Stanford University, working in the research groups of Professors David Cheriton and Christos Kozyrakis. From 2012 to 2016, my research centered on programmable “Open-Channel” solid-state drives (SSDs). I introduced techniques to reduce access latency by shifting performance-critical tasks such as data placement from the SSD to the host processor. This work led to “RAIL: Predictable, Low Tail Latency for NVMe Flash,” which achieved a $7\times$ reduction in read access latency through host-orchestrated SSD data placement.

7. In 2017, I served as Visiting Faculty at Google, working on programmable storage devices within the Platforms Infrastructure Group.

8. In 2018, I joined the Computer Science Department at the University of California, Santa Cruz (UCSC), as an Assistant Professor. I was promoted to Associate Professor with tenure in 2023. In 2024, I was named the Kumar Malavalli Endowed Professor for Storage Systems Research, a position supported by a \$1 million gift from Kumar Malavalli.

9. Since 2023, I have served as Director of the Center for Research in

Storage Systems (CRSS) at UCSC, a center comprised of five faculty and 15 Ph.D. students. CRSS advances research in storage systems and is supported by the National Science Foundation (NSF) and industry sponsors. Our work spans a range of topics, including programmable storage and emerging technologies such as Open-Channel, Zoned Namespace (ZNS), and Flexible Data Placement (FDP) SSDs. In the context of ZNS, my group proposed the group append feature to optimize ZNS SSD integration with file systems (e.g., “Append is Near: Log-based Data Management on ZNS SSDs.”

10. In 2024, I was a Visiting Professor at the Massachusetts Institute of Technology (MIT) in Cambridge.

11. Over the course of my academic career, I have published more than 80 peer-reviewed research papers, journal articles, and patents in the areas of storage systems and computer hardware. I have advised over 15 Ph.D. students and secured more than \$9 million in research funding. My honors include the NSF CAREER Award, Intel Outstanding Researcher Award, and multiple Faculty Awards from Google and Facebook. I received the Best Paper Award at MICRO 2023 and several IEEE MICRO Top Picks. I have also served as Program Chair for HotChips 2023 and NVMW 2024, and as Associate Editor for IEEE Computer Architecture Letters (CAL).

III. LEGAL PRINCIPLES AND UNDERSTANDINGS

12. Counsel for Samsung has informed me of the following legal standards. I am not an attorney and I am relying only on instructions from Samsung's attorneys for these legal standards.

A. Person of Ordinary Skill in the Art

13. I understand that issues of claim construction and invalidity are analyzed from the perspective of a person of ordinary skill in the art ("POSITA").

14. I understand that a POSITA is a hypothetical person who is used to analyzing the prior art without the benefit of hindsight. A POSITA is presumed to be one who thinks along the lines of conventional wisdom in the art.

15. I understand that the hypothetical POSITA is presumed to have knowledge of all references that are sufficiently related to one another and to the pertinent art, and to have knowledge of all arts reasonably pertinent to the particular problem that the claimed invention addresses.

16. I understand that factors considered in determining the level of ordinary skill in the art at the time of an invention include (1) the type of problems encountered in the art; (2) the prior art solutions to those problems; (3) the educational level of active workers in the field; (4) the rapidity with which innovations are made; and (5) the sophistication of the technology. I further understand that the inventors' specific levels of skill are not relevant to this inquiry.

B. Claim Construction

17. I have been informed by counsel that the Patent Trial and Appeal Board (“PTAB”) applies the same claim construction standard used in district courts, where the claims are given their ordinary meaning as understood by one skilled in the art at the time of the invention, informed by the claim language itself, the specification, and the prosecution history. I also understand that “extrinsic evidence”—i.e., evidence other than the patent and prosecution history—can be relevant in determining how a POSITA would understand terms of art used in the claims. I have been informed, however, that extrinsic evidence may not be used to contradict the meaning of the claims as described in the intrinsic evidence—i.e., evidence in the claim language itself, the specification, and the prosecution history.

C. Anticipation

18. I understand that a patent claim is invalid based on anticipation if a single prior art reference discloses all of the features of that claim, and does so in a way that enables one of ordinary skill in the art to make and use the invention.

19. I understand that a prior art reference describes the claimed invention if it either expressly or inherently describes each and every feature set forth in the claim; *i.e.*, in determining whether a single item of prior art anticipates a patent claim, one should take into consideration not only what is expressly disclosed in that item, but also what is inherently present as a natural result of the practice of the

system or method disclosed in that item.

D. Obviousness

20. I understand that even if a patent claim is not anticipated, it is still invalid if the differences between the claimed subject matter and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person of ordinary skill in the pertinent art.

21. I understand that a person of ordinary skill in the art provides a reference point from which the prior art and claimed invention should be viewed. This reference point prevents one from using his or her own insight or hindsight in deciding whether a claim is obvious.

22. I also understand that an obviousness determination includes the consideration of various factors such as (1) the scope and content of the prior art,(2) the differences between the prior art and the asserted claims, (3) the level of ordinary skill in the pertinent art, and (4) the existence of secondary considerations or objective indicia of obviousness or non-obviousness.

23. I understand that an obviousness determination can be based on a single prior art reference or a combination of multiple prior art references or the knowledge of one of ordinary skill in the art. I understand that the prior art itself may provide a suggestion, motivation, or reason to combine or modify the teachings of the prior art, or that such a reason may come from other sources, such as the knowledge of a

person having ordinary skill in the art, common sense, and market forces. I understand that the following rationales may support a finding of obviousness:

- combining prior art elements according to known methods to yield predictable results;
- simple substitution of one known element for another to obtain predictable results;
- use of known technique to improve similar devices (methods, or products) in the same way;
- applying a known technique to a known device (method or product) ready for improvement to yield predictable results;
- “obvious to try” - choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success;
- known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art;
- some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

24. I understand that an obviousness determination requires a reasonable expectation of success in achieving the claimed invention.

25. I understand that secondary considerations or objective indicia of obviousness or non-obviousness may include (1) a long felt but unmet need in the prior art that was satisfied by the invention of the patent; (2) commercial success or lack of commercial success of processes covered by the patent; (3) unexpected

results achieved by the invention; (4) praise of the invention by others skilled in the art; (5) the taking of licenses under the patent by others; (6) deliberate copying of the invention; (7) teaching away; and (8) the simultaneous invention of the claimed subject matter. I also understand that there must be a relationship between any such secondary considerations and the invention. I further understand that contemporaneous and independent invention by others is a secondary consideration supporting an obviousness determination.

26. I understand that any secondary considerations or objective indicia must bear a nexus to the claimed invention. Where the offered secondary consideration actually results from something other than what is both claimed and novel in the claim, there is no nexus to the merits of the claimed invention. I further understand that the patentee bears the burden of demonstrating that the relevant commercial success is attributable to the claimed invention, as opposed to other economic and commercial factors unrelated to the technical quality of the patented subject matter.

27. I understand that whether a prior art reference “teaches away” from the claimed invention is a fact to be considered in determining obviousness. I understand that it is improper to combine references where the references teach away from their combination. However, I understand that the nature of the teaching is highly relevant and must be weighed in substance – a known or obvious composition does not

become patentable simply because it has been described as somewhat inferior to some other product for the same use. Furthermore, I understand that the prior art's mere disclosure of more than one alternative does not constitute a teaching away from any of these alternatives because such disclosure does not criticize, discredit, or otherwise discourage the solution claimed.

IV. LEVEL OF ORDINARY SKILL IN THE ART

28. In my opinion, a POSITA would possess at least a Bachelor of Science degree in electrical engineering, or an equivalent field, and at least three years of experience with SSDs or other flash memory devices. More education can supplement practical experience and vice versa. *Id.*

29. I am familiar with the understanding and knowledge of persons of ordinary skill in the art as of the alleged priority date of the '614 patent in 2013 and am at least as qualified as the POSITA based on the qualifications I have identified above.

V. TECHNOLOGY BACKGROUND

30. Solid-state drives (SSDs) are semiconductor-based data storage devices that provide large storage capacities, non-volatility, and high read and write performance. SSDs are a mature and widely adopted storage technology, present in nearly all modern smartphones, laptops, and servers. SSDs store data on NAND flash memory.

31. Flash memory is organized into blocks. Blocks are comprised of pages, which represent the smallest unit that may be written. To increase the efficiency of flash memory, SSDs may subdivide the flash memory into groupings of blocks that are operated together, termed, *e.g.*, “superblocks,” EX1004, 6:43-65, “metablocks,” EX1006, 3:43-46, or “zones,” EX1012, ¶¶41-12. Moreover, to increase parallelism, blocks of data in a grouping may be written or erased together, including blocks that are on different planes. EX1014, §2.2 (“Multi-plane command activates multiple read, program or erase operations in all planes of the same die”), Fig. 4.

32. A key characteristic of SSDs is that data cannot be modified in place without first performing an explicit erase operation. This constraint means each block of data can only be written once, unless it is being explicitly erased. To enable data modification, SSDs employ an out-of-place update strategy: when data is updated, the new version is written to a different location instead of overwriting the old one. This results in two versions of the same data element coexisting—the outdated version and the newly written version.

33. To ensure that read operations retrieve the most recent data, SSDs utilize a flash translation layer (FTL), which manages a logical-to-physical (L2P) mapping table. Applications issue I/O operations using logical block addresses (LBAs), while the SSD stores data at physical block addresses (PBAs). The L2P

table ensures that each LBA always maps to the PBA containing the most recently written data.

34. To increase the longevity of flash memory, SSDs perform management operations including wear-leveling and garbage collection.

35. For instance, due to the limited endurance of NAND flash cells (e.g., a floating-gate transistor may only withstand ~10,000 P/E cycles), wear-leveling is essential. Wear-leveling enables blocks to be erased and reused, thereby balancing wear across all units in the SSD.

36. Further, since each write consumes a previously unwritten block, an SSD will eventually run out of such blocks. To reclaim space, SSDs support an erase operation. For instance, to safely erase a grouping of blocks (e.g., a superblock, metablock, or zone) it must not contain any *valid* blocks. A valid block represents the latest version of a data element. If no such grouping of blocks is entirely free of valid blocks, the SSD performs *garbage collection*. During this process, the SSD rewrites valid blocks to new locations, updates the L2P accordingly, and invalidates the old PBAs. Once all valid blocks have been relocated, the grouping of blocks can be safely erased.

37. To identify data for maintenance, SSDs may track and evaluate metadata describing the history of the blocks or groups of blocks. For instance, when determining whether to perform wear leveling an SSD may consider, among other

things, an erase count (number of time a unit has been erased). When moving the data the SSD may also consider, for instance, the temperature of the data (*e.g.*, when it was last accessed and if it is hot or cold). To optimize garbage collection, the SSD may maintain metadata to track the number of valid blocks in the grouping of blocks (*e.g.*, the superblock, metablock, or zone). Selecting a grouping of blocks with the fewest valid blocks minimizes the number of costly copy operations. POSITAs would understand that various other types of metadata may also be evaluated in connection with write and maintenance operations. EX1004, 10:21-32, EX1006, ¶¶47, 65, 80.

38. In summary, the FTL performs several key tasks:

- L2P Management: Mapping LBAs to the most recent PBAs for servicing reads.
- Data Placement: Determining whether hot or cold data should be written to a given block or group of blocks.
- Wear-Leveling: Ensuring uniform wear across blocks or groups of blocks by migrating cold data.
- Garbage Collection: Reclaiming space by relocating valid data and erasing blocks.
- Erase Management: Erasing groups of blocks that no longer contain valid data.

39. In many traditional SSDs, the FTL is implemented entirely within the SSD controller. However, prior to the earliest possible priority date of the '614 patent in 2013, an alternative approach was known in which the FTL was

implemented—fully or partially—on the *host*. Host-based FTLs can leverage application-specific knowledge (e.g., whether data is hot or cold) and device conditions (e.g., battery life) to make more informed decisions about data placement, garbage collection, and wear-leveling. For instance, garbage collection can be deferred to periods when the application is idle, thereby improving performance and efficiency.

40. The sharing of memory management responsibilities between the host and the controller has been termed, *i.e.*, “Cooperative memory management.” For example, Samsung’s U.S. Pat. Pub. 2008/0189485A1, describes “method[s] of cooperative memory management between a host and a memory card device includes selecting at least one of several memory management functions to be performed by the host for the device.” EX1007, Abstract.

41. The operations implemented by the FTL are functionally equivalent regardless of whether they are executed on the SSD or on the host processor. In fact, in most commercial SSDs, the FTL is implemented in software (firmware) and executed on an embedded processor within the SSD. To enable FTL functionality on the host side, the SSD must expose additional interfaces to the host—such as support for explicit erase operations and fine-grained control over data placement for instance by allowing host software to directly read and write PBAs and access meta data from the SSD controller.

VI. OVERVIEW OF THE '614 PATENT

A. Priority

42. The '614 patent issued on June 20, 2023, and claims the benefit of Provisional Application No. 61/757,464, filed on January 28, 2013. I thus understand that January 28, 2013 is the earliest possible priority date. The prior art that I discuss in this Declaration predates January 28, 2013, and I express no view on whether or not the '614 patent is entitled to a priority date before its filing.

B. Specification

43. I have reviewed the '614 patent. The '614 patent describes a memory system, including a memory controller (in SSD) and a host, where responsibility for memory management is shared between the controller and host. EX1001, 4:12-20. The '614 patent discloses that in some embodiments the flash translation on the storage device can be “eliminate[d],” but that in other embodiments “a conventional translation layer” is used. *Id.* at 61:52-67, 53:19-40.

44. The memory is organized by “sub-divisions,” where “each subdivision is an EU [erase unit] or page of NAND flash memory.” *Id.*, 13:36-49. The '614 patent teaches that information about the subdivisions may be provided by the controller to use “asynchronously,” *e.g.*, upon “reach[ing] a predetermined threshold,” or based on a host-initiated “query.” EX1001, 14:48-67. This information is used for maintenance operations such as, *e.g.*, “garbage collection,” *id.*, Figs. 11A-11B, or wear leveling, *id.*, Figs. 12A-12B.

45. The '614 patent purports to improve upon the prior art by permitting the host to “choose physical addresses,” which the patents states allows the “host to direct a ‘true’ multi-plane write or read operation.” EX1001, 54:7-65, 53:20-40; *see also id.*, Abstract (“host directly assigns physical addresses”). As discussed in the next sub-section, the Examiner pointed to the patent’s disclosures regarding direct physical addressing as the reason for allowance. **However**, I understand that in the co-pending litigation PO has interpreted the claim language to encompass systems that do not include direct physical addressing by the host. EX1011. For instance, PO interprets the phrase “host accessible pages,” which appears in Limitations 1[pre]-1[a.ii], to broadly encompass SSDs that includes pages that may store data originating from a host. EX1011, 8-9. PO does not interpret the term to require that the host select physical addresses (*id.*) and accuses conventional “multi-plane writes” of satisfying the claim language (*id.* at 29-30). I understand that PO must interpret the claims consistently for purposes of infringement and invalidity. Thus, for purposes of this Declaration I have applied PO’s interpretation and interpreted the claim to not require direct physical addressing.

C. Prosecution History

46. I have also reviewed the prosecution history of the '614 patent. The application that issued as the '614 patent was filed on September 12, 2022.

47. In a Notice of Allowance dated January 10, 2023, the Examiner

identified Limitations 1[pre]-1[a.ii] as the reason for allowance. EX1002, 196-197; *see also* Appendix A, Claim Listing. The Examiner then went on to summarize certain prior art and conclude with the following purported distinction: “However, none of the prior art teaches or renders obvious the features as recited in the claims, taken in light of the disclosure, specifically as detailed on Par. 0183-0184.” EX1002, 198.

48. The paragraphs (183-184) cited by the Examiner concede that multi-plane writes were known in the prior art, but claim that the alleged invention allows the host to select physical addresses in order to provide a “true” multi-plane write:

The ability to choose physical addresses permits the host to effectively direct multi-plane writes and reads as long as the assigned addresses (for related, respective data to be obtained from the planes) are consistent with any device addressing limitations for multiple plane use.... **Host-selection of physical addressing thus permits the host to direct a “true” multi-plane write or read operation without being affected by legacy multi-plane device addressing restrictions....**

EX1002, 998-999.

49. The challenged claims, however, do not include a limitation reciting “host-selection of physical addressing.” Nor has PO interpreted the claims to require host-selection of physical addressing in its infringement contentions. EX1011. For instance, PO interprets the phrase “host accessible pages,” which appears in Limitations 1[pre]-1[a.ii], to encompass SSDs that include pages that may store data originating from a host. EX1011, 8-9. PO does not interpret the term to require that

the host select physical addresses (*id.*) and accuses conventional “multi-plane writes” of satisfying the claim language (*id.* at 29-30). As I mentioned above, I understand that PO must interpret the claims consistently for purposes of infringement and invalidity. Thus, for purposes of this Declaration I have applied PO’s interpretation and interpreted the claim to not require direct physical addressing. As discussed below, under this interpretation, the prior art in this Declaration discloses or renders obvious each limitation of the challenged claims, including the limitations (Limitations 1[pre]-1[a.ii]) that the Examiner found to be distinguishing over the prior art of record during prosecution.

VII. LEVEL OF ORDINARY SKILL IN THE ART

50. A person of ordinary skill in the art (“POSITA”) at the time of the ’614 patent would have had at least a Bachelor of Science degree in electrical engineering, or an equivalent field, and at least three years of experience with SSDs or other flash memory devices. More education can supplement practical experience and vice versa.

51. I am at least as qualified as the POSITA based on the qualifications I identified above. See Section II. For instance, by the year 2013, I had been awarded a Ph.D. in computer science and was working as a Postdoctoral Researcher at Stanford University on research that centered on SSDs. And in the year that have followed, I have continued working on SSD technology, have published numerous

papers on the topic, and have contributed to standards relating to the same.

VIII. CLAIM CONSTRUCTION

52. I understand that in the co-pending district court action the parties have not yet submitted claim construction proposals.

53. I have been informed that, for the purposes of my analysis in this IPR proceeding, the terms appearing in the claims of the '614 patent should be interpreted according to how a POSITA would have understood the claim terms at the time of the earliest priority date, in light of the patent specification, the prosecution history, and any other relevant intrinsic and extrinsic evidence. I have used the earliest possible priority date of the '614 patent (January 28, 2013) as the point in time for claim interpretation purposes. However, the interpretations that I provide below would have also been correct if the date of alleged invention was earlier or later.

IX. GROUND 1: CLAIMS 1-2, 4-8 ARE OBVIOUS IN VIEW OF ELLIS (EX1004) AND SON (EX1006)

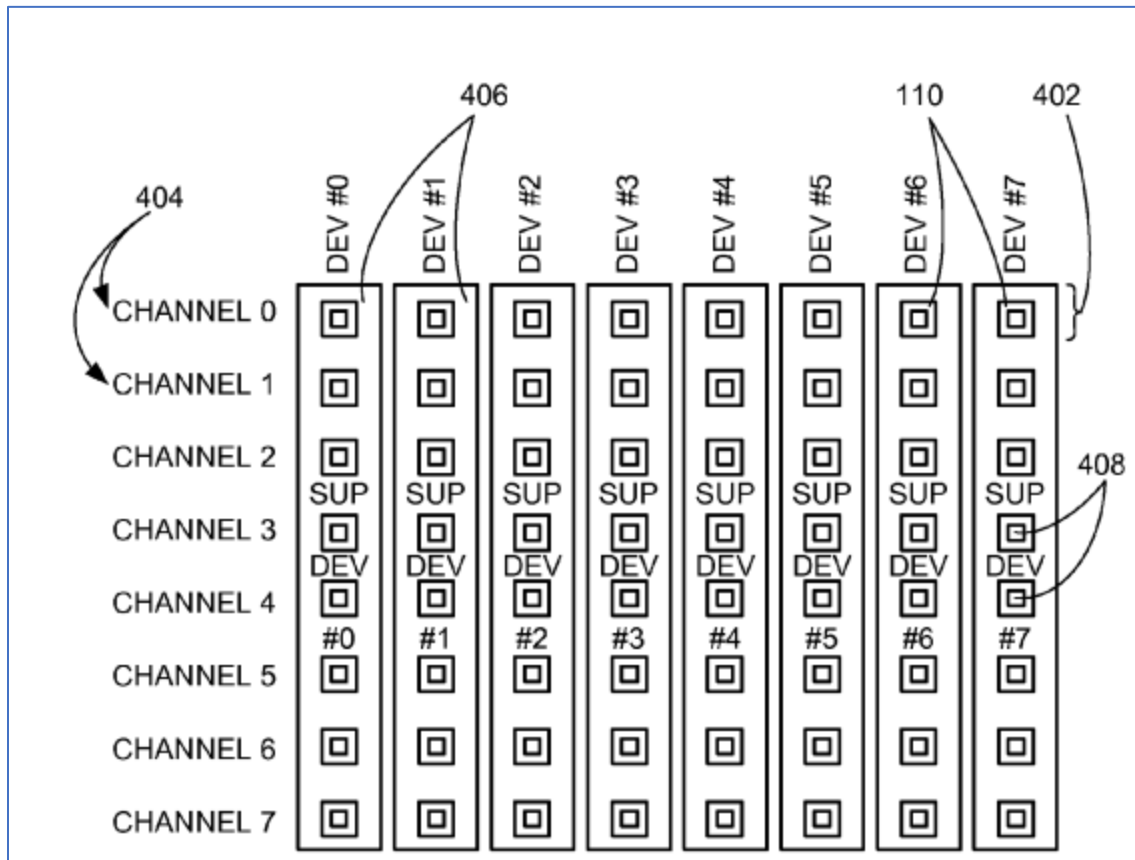
A. Overview of the Prior Art

1. Ellis (EX1004)

54. U.S. Patent 9,239,781 ("Ellis") was filed October 10, 2012. I understand that it is prior art because its filing date pre-dates the earliest possible priority date of the '614 patent.

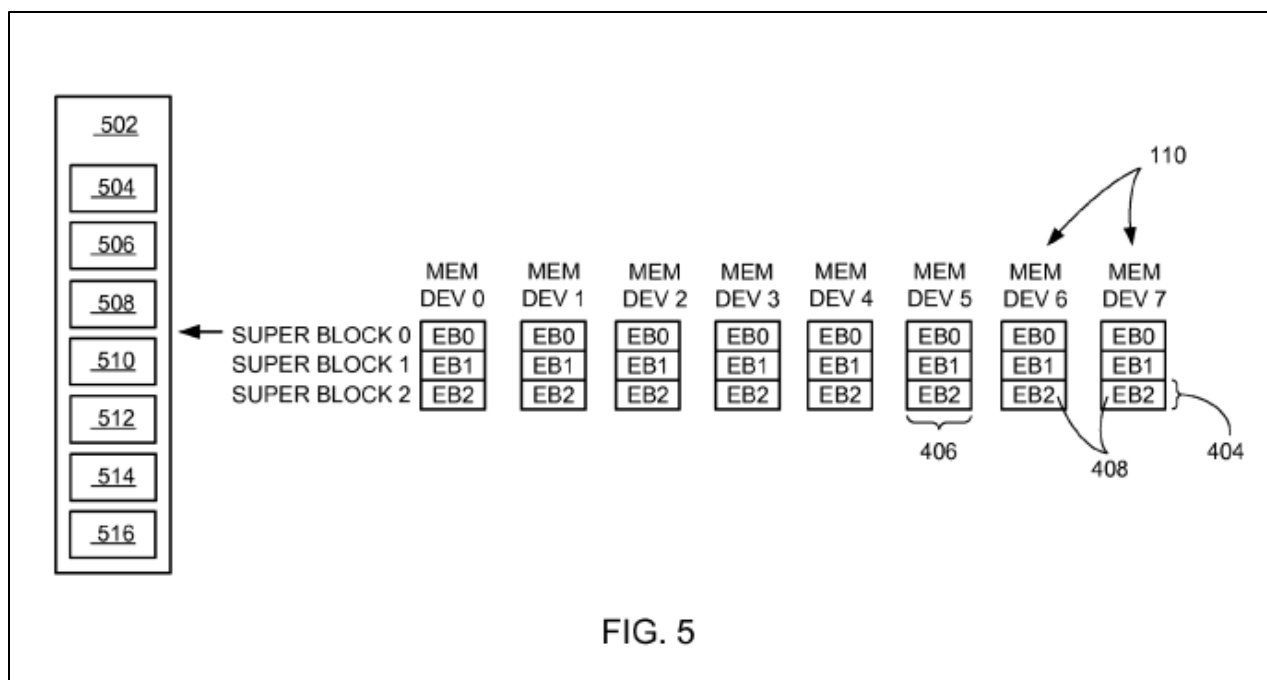
55. Ellis teaches a storage system with erase blocks spanning memory channels. Figure 4 illustrates erase blocks 408 within memory devices 110 that span

memory channels 404:



EX1004, Fig. 4; 6:42-65.

56. The memory is sub-divided into superblocks—each superblock is a group of erase blocks. EX1004, 6:43-65. Figure 5 illustrates three superblocks which comprise one erase block from each of the eight channels, *e.g.*, superblock 1, includes eight erase blocks (EB1 (erase block 1) from 8 different channels).



Id., Fig. 5; 6:43-56.

57. Ellis discloses that superblocks can be operated together: “host writes, recycling operations, or erasing of blocks can be performed on a set of the erase blocks 408 included within one of the superblocks **as a single unit.**” *Id.*, 8:4-7. Further, Ellis stores information (metadata) for each of the superblocks. *Id.*, 11:58-12:2; 12:37-41, Fig. 7. For example, metadata can include “logical block address (LBA) information, superblock identification (ID) values, program/erase (PE) cycles, aging data, and other NAND statistics that is collected as a consequence of normal operations.” *Id.*, 10:21-32.

58. Ellis also teaches that memory operations may be shared: either “the memory controller [or] the host system ... can operate any of the modules and functions of the memory controller.” *Id.*, EX1004, 4:58-65.

2. Son (EX1006)

59. U.S. Patent Application No. 2012/0096217 (“Son”) was filed on Oct. 14, 2011 and was published on April 19, 2012. I understand that it is prior art because its filing date and publication date pre-date the earliest possible priority date of the ’614 patent.

60. Son describes the “shortcomings of conventional systems” where management operations are accomplished primarily using a flash translation layer on the controller, and describes the benefits of shared management operations where the host “incorporates both file system information and information related to the underlying physical solid-state storage media into its operations.” EX1006, ¶14.

61. Son’s system is subdivided into superblocks which are “a grouping of... blocks.” EX1006, ¶29. Son teaches storing metadata for the superblocks in management tables, such as the superblock table below:

Superblock Table ³⁵⁰										
	335a	335b	335c	335d	335e	335f	335g	335h	335j	335k
	Super Block ID	Wear Count	# Valid Pages	# Invalid Pages	State	Next Superpage	Next Page	Page Detail	Last Mod. Time	Data Type
340a	0	115	0	0	CLEAN	0	0	Pagelist ₁ []	5/5/11 09:11:01	Metadata
340b	1	462	856	741	IN_USE	68	5	Pagelist ₂ []	1/1/09 22:08:11	User Data
	...	...	...	...	...	...	...	...	...	...

FIG. 3C

EX1006, Fig. 3C, ¶¶80, 29, 38.

B. Claim Analysis

1. Claim 1

(a) [1pre] A storage device comprising:

62. In my opinion, to the extent the preamble is limiting, Ellis discloses a storage device. Figure 1 shows a host and a connected non-volatile memory system:

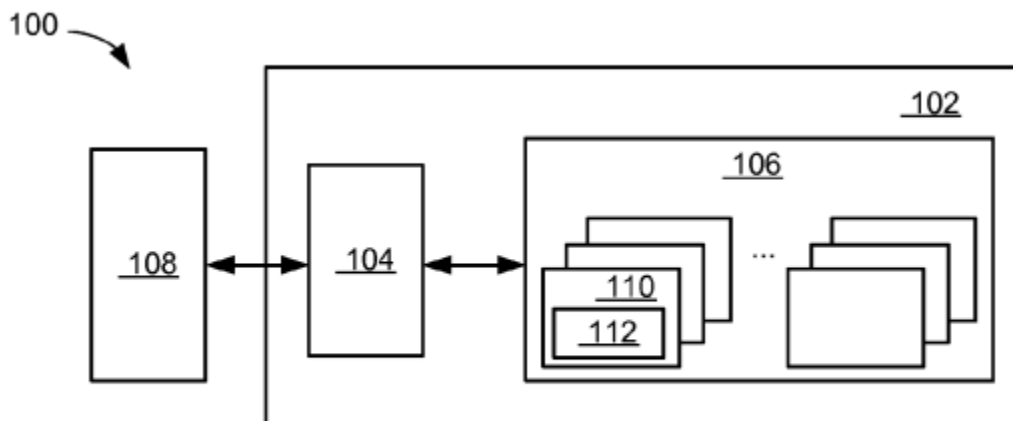


FIG. 1

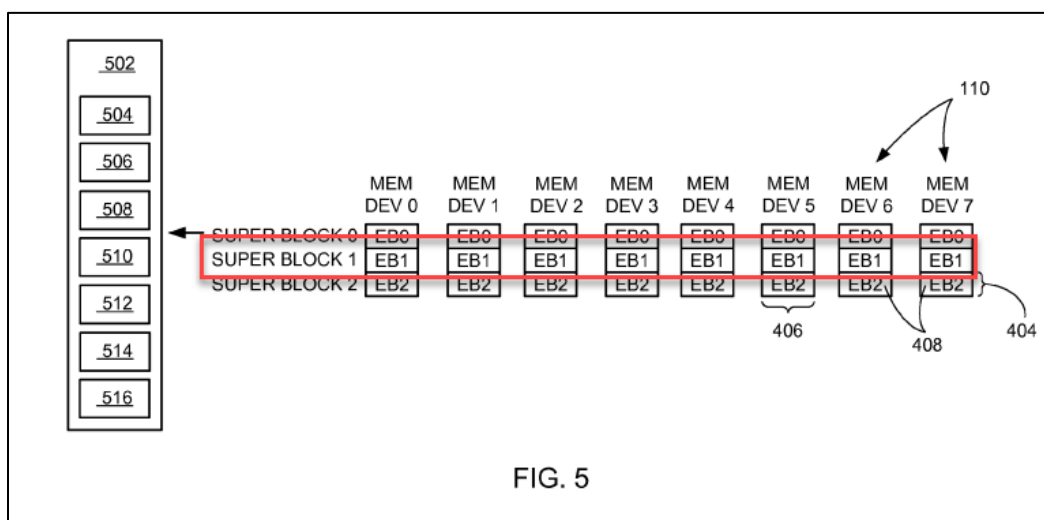
EX1004, Fig. 1; *id.* 3:10-21. The host 108 communicates with the memory system 102, which includes memory controller 104. *Id.* A POSITA would understand the memory system 102 is a storage device.

(b) [1a] flash memory divided into zones, each zone of the zones being a subdivision comprising a set of erase units, and each erase unit of the set of erase units comprising host accessible pages, wherein each zone:

63. In my opinion, Ellis discloses that the flash memory is divided into superblocks [**zones**], where each superblock is comprised of erase blocks [**each zone of the zones being a subdivision comprising a set of erase units**], and each

erase block of the set of erase blocks in the superblock comprises “pages” that are accessible by the host [**host accessible pages**].

64. Ellis teaches that memory array 106 (shown above) “includes an array of memory devices 110 including flash memory devices.” EX1004, 3:22-25, Fig. 1. The flash memory is divided into “superblocks” [**zones**], that are each “defined as logical groupings of the erase blocks” [**each zone of the zones being a subdivision comprising a set of erase units**]. EX1004, 7:32-39, 6:43-65, 3:10-45. For instance, Figure 5 illustrates superblocks that are comprised of eight erase blocks.



EX1004, 18:5-10, 7:7-12, Fig. 5.

65. A POSITA would understand that the superblocks are “zones” and “subdivisions” because they are a logical grouping of several physical blocks. Superblocks are used to manage larger units of data more efficiently. *Id.*

66. Ellis also discloses that “each of the erase blocks” includes “256 pages.” EX1004, 7:7-12, 18:5-10. Ellis’s erase blocks are “erase units.” An erase

block represents a unit of memory that can be erased at one time, and the '614 patent uses the terms erase block and erase unit interchangeably: “In embodiments where the memory is flash memory and the memory controller a flash memory controller, each subdivision can include one or more **erase blocks or units (EUs)**.” EX1001, 13:22-30. As I explained above, I understand that Patent Owner has interpreted the term “host accessible pages” such that it does not require host-selection of physical addressing. See Section VII.C. Thus, a POSITA would understand Ellis’s disclosure of pages that may include data sourced from the host to be “**host accessible pages**.”

(c) [1a.i] is associated with a logical address range,

67. Ellis discloses that the superblocks [**zones**] are associated with “logical block address (LBA) information” [**a logical address range**].

68. For instance, Ellis discloses that the host can read and write data “from or to a logical address space” of the memory system. EX1004, 3:25-28. Ellis also discloses that the memory system stores “logical block address (LBA) information” along with “superblock identification (ID) values.” EX1004, 10:28-32. A POSITA would understand this disclosure to teach that the blocks and pages of a superblock are associated with a logical address so that the host may read and write data from and to the superblock utilizing the logical addresses. Indeed, by the time of the '614 patent superblocks were typically associated with a range of logical block addresses. EX1009, ¶35 (“In this example, an LBA range of 4,096 addresses can be mapped to

a particular 4,096 sector physical Super Block. However, embodiments of the present disclosure are not limited to a particular page size, sector size, and/or physical block size.”)

- (d) [1a.ii] has a zone size for which the storage device is operable to receive a management command from a host that results in the storage device performing an erasure of one or more erase units of the set of erase units,

69. Ellis discloses a superblocks size [**zone size**] for which the memory controller may issue wear leveling and garbage collection commands [**management command**] that result in the memory system 102 [**storage device**] performing erasure of one or more erase blocks that form the superblock [i.e., **erasure of one or more erase units of the set of erase units**].

70. For instance, Ellis discloses that the memory controller 104 can include a “flash translation layer (FTL).” EX1004, 4:66-5:3, 16:7-13. The FTL performs maintenance operations on “the superblocks,” such as “wear leveling.” *Id.*, 16:7-13. A POSITA would understand that a command for wear leveling is a management command and that wear leveling results in erasure of the erase blocks that form the superblock. In particular, wear leveling involves the flash memory moving data to less worn superblocks to balance wear, and, after moving the data, erasing the original superblock to reclaim them for future writes. *Id.*

71. Ellis states that either “the memory controller [or] the host system ... can operate any of the modules and functions of the memory controller.” *Id.*,

EX1004, 4:58-65. A POSITA would understand this disclosure to teach, or at least suggest, that commands for wear leveling and garbage collection [management commands] may be implemented within the host, such that the memory system 102 [storage device] is operable to receive a management “**from a host.**”

72. To the extent Ellis does not expressly disclose that the memory system is operable to receive a management command from a host, it would have at least been obvious to modify Ellis’s storage system so that the storage device is operable to receive a management command from a host device, as taught in Son.

73. Son discloses an SSD Management Module 150 of a host device (orange box) that is in communication with the flash memory (red):

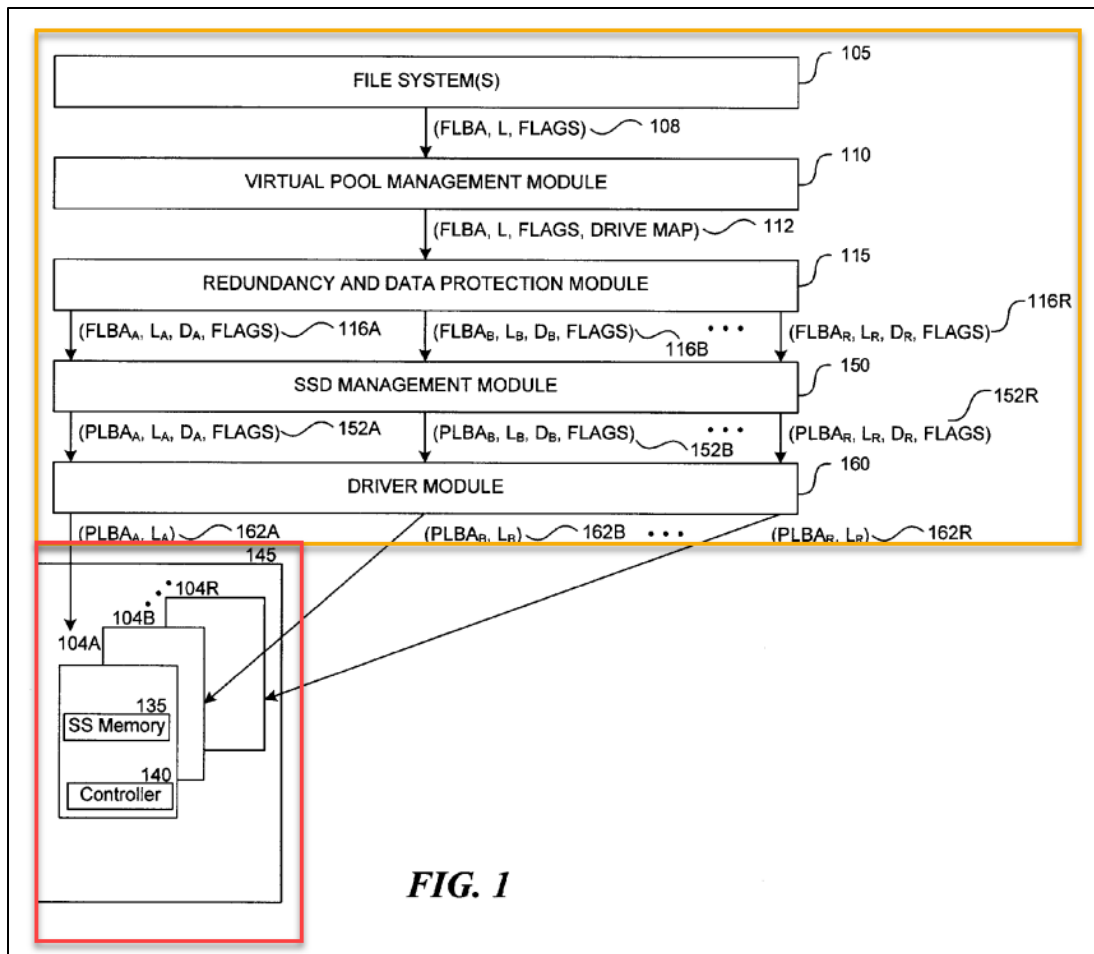


FIG. 1

EX1006, Fig. 1. Like Ellis, Son discloses that the flash memory is subdivided into superblocks that are comprised of erase blocks, which are in turn comprised of pages. EX1006, ¶¶28-29. The SSD Management Module of the host initiates “various SSD maintenance operations including garbage collection and wear leveling, which can relieve the SSD controller and firmware of these responsibilities.” EX1006, ¶¶16, 63-64.

74. A POSITA would understand a host-issued command for garbage collection or wear leveling is a “management command from a host,” and that this

command results in the storage device erasing a superblock. For instance, Figure 6 of Son discloses that with garbage collection the “SSD [management] module [within the host] instruct[s] the SSD controller to copy data from the source page and move it to the selected target superblock” and then after it is moved the “SSD [management] module instructs the SSD controller to **erase** the source superblock” that is no longer valid. EX1006, ¶¶68-70, Fig. 6 (blocks 620 and 650). Similarly, Fig. 7 discloses that when the host initiates wear leveling it “instructs the controller to copy the contents of all pages in the worn superblock over to the target superblock” and then “erases the worn superblock.” EX1006, ¶¶78-83, Fig. 7.

75. **Motivation to Combine:** A POSITA would have been motivated to combine these references because they cite to one another. For instance, Ellis cites to and references Son. EX1004, pg. 3. And both references concern the optimization of NAND Flash SSD memory management, such that they are in the same field of endeavor. EX1004, 7:32-45; EX1006, ¶2, 14, EX1010, 1:56-2:26.

76. The prior art also includes express teachings, suggestions and motivations to combine the teachings of Ellis and Son such that the storage device is operable to receive a management command from a host.

77. For instance, Son teaches that a system like Ellis may be improved by configuring the host to issue management commands so that the system may consider host information. Son explains that, as taught in Ellis, an SSD can include

a “controller and firmware” for performing “data management operations,” such as “wear leveling”—and other operations, such as “address remapping,” “garbage collection,” etc. EX1006, ¶13. Son states that “the inventors have discovered that the failure of conventional SSDs to incorporate file system information” (from the host) “into their operational logic often results in inefficiencies.” EX1006, ¶13. Son also discloses that moving some or all of the management operations to the host helps “relieve the SSD controller and firmware of these responsibilities.” EX1006, ¶63. Thus, Son discloses that Ellis can be improved by configuring an “SDD module” of a host to perform management operations to “help improve the performance of the SSD array.” EX1006, ¶¶17, 21.

78. Other references also teach that a system like Ellis may be improved by sharing the management functions between the host and controller. For instance, Jung discloses a system for “cooperative memory management” and teaches that moving some or all “FTL functions” (*e.g.*, “wear-leveling”) to the host can increase “execution speed” and avoid the controller becoming “overloaded.” EX1007, ¶¶3-5, 7, 41. Similarly, Yamada discloses that “data maintenance” operations “require[] a certain amount of power consumption” and that “conventional techniques fail to consider the device conditions” when maintenance is performed. EX1015, ¶10. Yamada teaches that to “solve this problem” it moves certain management operations to the host, such as wear leveling. EX1015, ¶¶11, 30. Thus, Son, Jung,

and Yamada all provide an express teaching, suggestion, and motivation for modifying Ellis's device so that one or more management commands for the superblocks are issued by the host.

79. A POSITA would also have had a reasonable expectation of success in combining Ellis and Son as it represents a straightforward implementation of shared management and metadata management to similar flash devices organized into superblocks.

(e) [1a.iii] is in a full state when there are no host accessible pages available for writing, and

80. Ellis discloses that each superblock [**zone**] may be in a closed state [**full state**] when there are no host accessible pages available for writing or an open state [**empty state**] when all host accessible pages are available for writing.

81. Ellis discloses that “when the **state** of the superblocks 502 is set to ‘closed,’ no more writing is allowed to take place.” EX1004, 15:27-38. A superblock that is completely logically written is transitioned to a “closed” state because there are no host accessible pages for writing. *Id.*

82. Ellis also discloses that a superblock may be in an “open” state when it has “unwritten pages” and that a “valid-page count” that defines the “number of pages that are valid in each of the superblocks” is maintained. EX1004, 15:27-38. A POSITA would understand an open superblock with a valid-page count of zero is in an empty state.

83. A POSITA would have also found it obvious for Ellis’s storage device to track whether superblocks are in a full or empty state in view of Son and would have had a reasonable expectation of success in combining the reference.

84. Son includes a similar disclosure as Ellis that the system maintains whether superblocks [zones] are in one of three states: “CLEAN” (“meaning the superblock has been erased and is available to take new writes.”), “IN_USE” (“there are pages within the superblock that are available for new writes”), or “SEALED” (“there are no free pages within the superblocks because all pages have been written.”). EX1006, ¶40, The “CLEAN” state is an **empty state** and the “SEALED” state is a **full state**. EX1006, ¶40, Fig 3C.

85. **Motivation to Combine:** A POSITA would have been motivated to combine Son’s teaching to Ellis’s storage device, such that the combined system would maintain a “CLEAN” and “SEALED” state for each superblock. Both references are directed at tracking superblock level metadata to optimize memory management, writing and erasing. EX1004, 12:26-29; EX1006, ¶¶78-80; EX1010, 47:62-49:19.

86. Moreover, Son provides express motivations for maintaining the state of each superblock—*e.g.*, that this information should be evaluated during management operations when selecting a superblock to write to. EX1006, ¶47. A POSITA would have had a reasonable expectation of success in combining these

references because maintaining a binary state (full/empty) would have been a straightforward modification that would improve a storage controller by simplifying decision-making logic.

- (f) [1a.iv] is in an empty state when all host accessible pages are available for writing; and

87. Ellis discloses, or at least suggests, an empty state. See Limitation 1.a.iii. For instance, I explained above that a POSITA would understand an open superblock with a valid-page count of zero in Ellis is in an empty state.

88. This limitation is also obvious in view of Son. *Id.* For instance, I showed above that Son discloses a “CLEAN” state [**empty state**].

- (g) [1b] logic operable to:

89. Ellis discloses that the memory system 102 includes a memory controller with hardware and software [**logic**]. EX1004, 3:38-45, 3:1-5.

- (h) [1b.i] return information about the zones in response to a query command from the host, the information about the zones comprising a list of the zones accessible by the host, a zone state, the zone size, logical addresses associated with respective zones, and information representing an extent to which the respective zones include host accessible pages available for writing;

90. Ellis teaches that control and status information about the superblocks [**information about the zones**] is stored as “metadata.” EX1004, 12:35-41, 10:27-32; 12:26-29, 7:45-52. For instance, the information can include “logical block address (LBA) information,” *id.*, 10:21-32, a “valid-page count” for each

superblock, *id.*, 11:58-60, a superblock state (“open” or “closed”), *id.*, 15:27-38. Ellis also discloses that the host and storage device exchange information about the superblocks. EX1004, 3:10-16, Fig. 1.

91. To the extent Ellis does not expressly disclose returning the information recited in this limitation in response to a query from the host, this limitation is at least obvious in view of Son, which also identifies superblock metadata and expressly discloses that it may be returned to the host in response to a query so that it may be evaluated when issuing maintenance commands. I address each portion of this limitation in greater detail below.

92. **return information about the zones in response to a query command from the host:** Son discloses that management tables containing information about the superblocks [**information about the zones**] may be stored in “persistent storage,” for example the “SSD.” EX1006, ¶¶34 (“The SSD [management] module may store some or all of the data structures shown in FIGS. 3A-3D related to a particular SSD in persistent storage such as the same SSD...”), 65, 80, Fig. 3A-3D. An example “superblock table” is reproduced below:

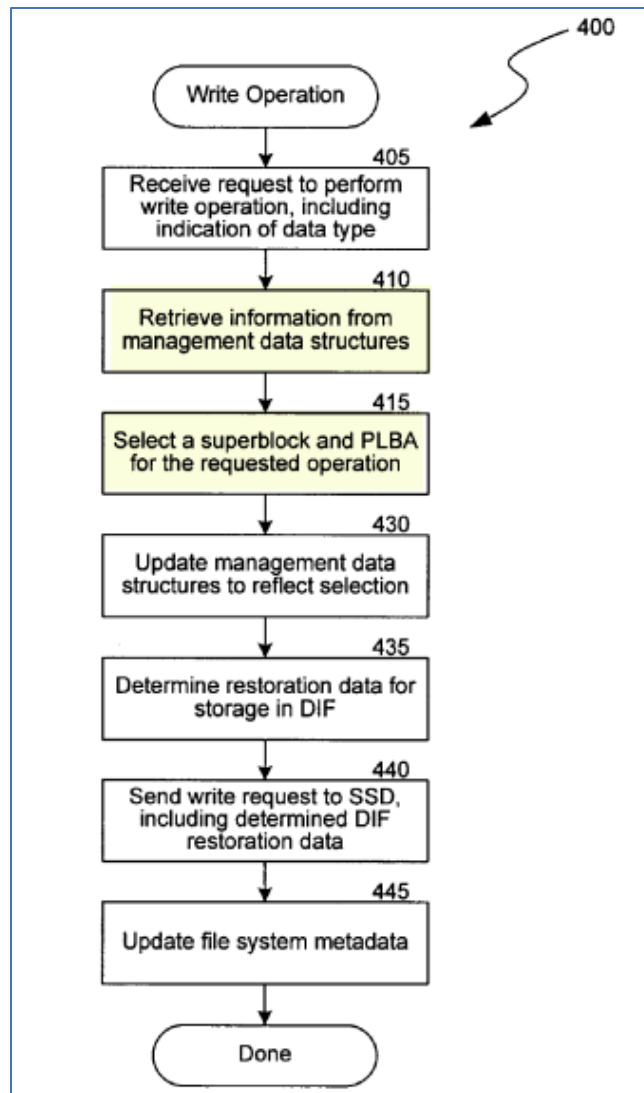
Superblock Table ³⁵⁰										
	335a	335b	335c	335d	335e	335f	335g	335h	335j	335k
	Super Block ID	Wear Count	# Valid Pages	# Invalid Pages	State	Next Superpage	Next Page	Page Detail	Last Mod. Time	Data Type
340a	0	115	0	0	CLEAN	0	0	Pagelist ₁ []	5/5/11 09:11:01	Metadata
340b	1	462	856	741	IN_USE	68	5	Pagelist ₂ []	1/1/09 22:08:11	User Data
	...	...	...	...	...	...	...	...	...	...

FIG. 3C

EX1006, Fig. 3C; *see also id.*, Figs. 3A-3B, 3D.

93. Son also discloses that the SSD module of the host may retrieve the information about the superblocks by “querying the SSD controller” [*e.g.* **query command from the host**]. EX1006, ¶47. Indeed, Son may query the SSD module for the claimed information in various scenarios, including as part of write operations, wear leveling, and garbage collection.

94. For instance, Figure 4 illustrates a write operation. EX1006, Fig. 4. At step 410, the SSD module “retrieve[s] information from the management data structures” (EX1006, ¶46) by “querying the controller” (EX1006, ¶47) and then uses that information to “select a superblock and PLBA for the requested operation” at step 415:



EX1006, Fig. 4. Similarly, the garbage collection process in Figure 6 involves “identifying source superblocks and target superblocks” at step 603. EX1006, Fig. 6. Son discloses that to perform step 603 the SSD module utilizes “information obtained from the controller,” such as “information shown in the superblock table, other management data structures, or information obtained from the controller.” EX1006, ¶65. A POSITA would understand that information may be obtained using

a host query, particularly in view of Son's earlier disclosure of a host query for obtaining this type of information. EX1006, ¶47.

95. As another example, Son discloses a wear leveling process in Figure 7 that selects superblocks at step 710. EX1006, Fig. 7. Once again, Son confirms that the SSD module may utilize "any" of the superblock information in this step and that the superblock information is obtained from the controller. EX1006, ¶80.

96. Additionally, Son discloses that "the SSD module may manipulate the values in a data structure by reading a portion of the data structure into system memory." EX1006, ¶34. A POSITA would understand this disclosure to teach that the SSD module may send a query to the storage device to read the management tables from the persistent memory. A POSITA would have also found it obvious to retrieve the entirety of the management tables from persistent memory, as for performance reasons, it is common to have the I/O device (SSD controller) copy the data into system memory via direct memory access (DMA) first, so that the software can then access the data directly from system memory.

97. **the information about the zones comprising a list of the zones accessible by the host:** Son discloses that the information about the zone may include a list of the superblocks [**list of the zones**] accessible by the host. For example, Figure 3C of Son discloses an example management table that includes a list of superblocks each of which is identified by a superblock ID:

Superblock Table ³⁵⁰										
	335a	335b	335c	335d	335e	335f	335g	335h	335j	335k
	Super Block ID	Wear Count	# Valid Pages	# Invalid Pages	State	Next Superpage	Next Page	Page Detail	Last Mod. Time	Data Type
340a	0	115	0	0	CLEAN	0	0	Pagelist ₁ []	5/5/11 09:11:01	Metadata
340b	1	462	856	741	IN_USE	68	5	Pagelist ₂ []	1/1/09 22:08:11	User Data
	...	...	...	...	...	...	...	...	...	...

FIG. 3C

EX1006, Fig. 3C. Additionally, Son teaches, or at least suggests, that as part of the write operation shown in Figure 4, the host will “access” the “superblock table” at step 410 to “retrieve” information about the list of superblocks and their state, so that it can “identify which superblocks in the SSD are currently in use or clean and therefore available as candidate superblocks to store the source data.” EX1006, ¶¶46-47. Similarly, Son teaches, or at least suggests, that a list of superblocks may be “obtained from the controller” for garbage collection and wear leveling. EX1006, ¶¶65 (SSD module may obtain the “management data structures”), 80 (same).

98. **a zone state:** Son teaches that the superblock table also includes a “state”. There are three states: “CLEAN” (“meaning the superblock has been erased and is available to take new writes.”), “IN_USE” (“there are pages within the superblock that are available for new writes”), or “SEALED” (“there are no free pages within the superblocks because all pages have been written.”). EX1006, ¶40, Fig 3C. The “state” column is shown below:

Superblock Table ³⁵⁰										
	335a	335b	335c	335d	335e	335f	335g	335h	335j	335k
	Super Block ID	Wear Count	# Valid Pages	# Invalid Pages	State	Next Superpage	Next Page	Page Detail	Last Mod. Time	Data Type
340a	0	115	0	0	CLEAN	0	0	Pagelist ₁ []	5/5/11 09:11:01	Metadata
340b	1	462	856	741	IN_USE	68	5	Pagelist ₂ []	1/1/09 22:08:11	User Data
	...	...	...	...	...	...	...	...	...	...

FIG. 3C

EX1006, Fig. 3C. As discussed above, as part of the write operation the SSD module of the host identifies a list of “candidate superblocks” by, among other things, retrieving the “state of each candidate superblock” and determining if it is in an “in use or clean” state. EX1006, ¶¶46, 47 (“When selecting the superblock, the SSD module may evaluate any combination of several factors including, as non-exhaustive examples... **the state of each candidate superblock**... The SSD module may determine some or all of these various factors... **by querying the... controller.**”). Similarly, the garbage collection and wear leveling operations also retrieve and evaluate the superblock state. EX1006, ¶¶65, 80.

99. **the zone size:** Son discloses that in response to a query from the SSD module in the host, the memory controller may return the “number of free, valid and/or invalid pages in each candidate superblock” [**zone size**]. EX1006, ¶¶47 (“When selecting the superblock, the SSD module may evaluate any combination of several factors including, as non-exhaustive examples: the volume of source data to

be written, **the number of free, valid and/or invalid pages in each candidate superblock.**”), 65 (same), 80 (same). A POSITA would understand that a page is a unit of storage and that returning the number of free, valid, and invalid pages in a superblock discloses, or at least fairly suggest, returning the size of the superblock. Moreover, Son discloses that superblocks comprise pages that are each “8KB of data” in size. EX1006, ¶28. Thus, it would have been obvious to also return an explicit value (*e.g.*, in bytes) for the size of the superblock.

100. **logical addresses associated with respective zones:** Son discloses that the management information may include a “mapping table” (*e.g.*, as shown in Figure 3B) that associates each FLBA, a file system logical block address, (column 320 *a*) with a data type (column 320 *b*), superblock address (column 320 *c*), a superpage address (column 320 *d*), a page address (column 320 *e*), and a sector address (column 320 *f*).”). EX1006, ¶38, Fig. 3B. A POSITA would understand that a FLBA is a **logical block address** and associating superblocks with logical block addresses discloses logical addresses associated with respective zones.

101. Son also discloses a “page detail array table 375” (*e.g.*, as shown in Figure 3D) that may include a row for “or all pages within a superblock.” EX1006., ¶43. Son “associates each page with an FLBA.” *Id.* Since the superblocks are comprised of pages, Son’s disclosure of logical addresses associated with each page

in a superblock also discloses logical address for the respective zones [**logical addresses associated with respective superblocks**].

102. Further, the SSD module may retrieve the mapping table and page detail array table in connection with step 410 of the write operation shown in Figure 4. EX1006, ¶¶46 (“At block 410, the SSD module retrieves information from management data structures, such as those illustrated by Figs. 3A-3D.”), 65 (same), 80 (same). For instance, Son teaches that the SSD module may retrieve and then consult the mapping table to identify whether an FLBA that the host seeks to write data to “is already associated with a previously-stored PLBA.” EX1006, ¶45.

103. **information representing an extent to which the respective zones include host accessible pages available for writing:** As discussed above, Son discloses that in response to a query from the SSD module in the host, the memory controller may return the “number of free, valid and/or invalid pages in each candidate superblock.” EX1006, ¶¶47, 65, 80; *see also id.*, ¶48 (SSD module considers whether a superblock has “an insufficient number of pages available”). At least the free page count is information representing the extent to which the specified zone includes host accessible pages available for writing because it indicates how many free pages are available for writing.

104. **Motivation to Combine:** A POSITA would be motivated to combine Son’s teachings of a host query for returning information about zones [superblocks]

with Ellis’s storage device. Ellis already teaches that “metadata can include ... [any] NAND statistics that [are] collected as a consequence of normal operatives,” and “any additional information ... that can aid in retrieval, organization, health of, and current condition or state of the memory devices.” EX1004, 10:21-32, 12:26-29. And, as discussed above (see Limitation [1.a.ii]), Son discloses the benefits of returning the information about zones to the host so that the host may utilize the information to issue management commands. *E.g.*, EX1006, ¶¶13-14.

105. A POSITA would have had a reasonable expectation of success in combining Son’s teachings and Ellis’s storage device as it merely represents a known technique of returning metadata to the host via a host query. By the time of the ‘614 patent sharing management operations with the host was well-known, as discussed above. See Ground 1, Limitation [1a.ii]. And it was also well-understood that the information could be returned in response to a host query, such as by defining a vendor specific command. See EX1005, ¶101 (“There are many ways in which these parameters may be read by the host. One example is to define a vendor specific command that is issued by the host 211 to the memory device 207”), EX1006, ¶¶46-47.

- (i) [1b.ii] write data to the set of erase units of a specified zone using a multi-plane write command such that the data is written concurrently to two or more erase units of the set of erase units; and

106. Ellis discloses writing data to the set of erase blocks **[sets of erase**

units] of a specified superblock [**specified zone]** using a multi-plane write command such that the data is written concurrently to two or more erase blocks of the set of erase blocks.

107. For instance, Ellis teaches that a superblock is comprised of a set of erase blocks that can be operated together as a “single unit” (EX1004, 8:4-8) and that a superblock can be formed from erase blocks of multiple memory devices that are comprised of multiple dies. EX1004, 7:25-31, 6:66-7:5. A POSITA would understand that erase blocks that are on different dies are also on different planes because a plane is a subdivision within a die. Thus, Ellis discloses that a superblock may comprise erase blocks from multiple planes.

108. Ellis also teaches that superblocks can provide “improved performance” by allowing “multiple pages within each of the superblocks 502” to be “written **concurrently**,” wherein the pages are “not just within one of the erase blocks.” EX1004, 10:49-55, 13:57-61, 14:39-44 (“Because data pages are in different instances of the memory channels 404 and physically different instances of the erase block 408, this virtual erase block [superblock] can be ordered such that multiple of the data pages can be written at the same time”). A POSITA would thus understand Ellis to disclose, or at least suggest, a multi-plane write command that concurrently writes to erase units in the superblock that are on different planes.

109. To the extent Ellis does not explicitly disclose the use of a multi-plane

command, it was at least obvious in view of the general knowledge of a POSITA. Multi-plane commands were well-known in the art. EX1014, §1.3 (“SSD manufacturers have provided advanced commands, such as... multi-plane...”), §2.2 (“Multi-plane command activates multiple read, program or erase operations in all planes of the same die”), Fig. 4. The prior art discloses that writing to multiple-planes concurrently improves performance of a storage devices by allowing “parallel” operations and that “parallelism has been regarded as a key to achieving the peak performance.” EX1004, 5:32-52; EX1010, 2:41-43; EX1014, §1.4. Thus, the prior art includes express teachings, suggestions and motivations to utilize a multi-plane command with the Ellis-Son combination.

- (j) [1b.iii] update the information representing the extent to which the specified zone includes host accessible pages available for writing.

110. Ellis discloses updating the information representing the extent to which the specified superblock [**zone**] includes host accessible pages available for writing. For instance, Ellis teaches that the controller may perform “a write operation ... for host write data,” EX1004, 14:25-34, where the “host interface” is the “communication mechanism” for the “host system,” *id.*, 4:46-50. Following the write execution, Ellis teaches tracking and “updating metadata” such as “valid-page count in ... the superblocks.” *Id.*, 20:11-20; Claim 1 (“...performing an operation on the superblock; and updating metadata corresponding to the superblock...”). Ellis

also discloses that the controller tracks the number of “stale or invalid pages” and that the controller tracks “unwritten pages in any of the erase blocks” of the superblock, to determine whether it is in an “open” (empty) stage or a “closed” state (full). *Id.* at 19:11-15, 15:27-38. A POSITA would, therefore, understand that Ellis discloses, or at least suggests, that after data is written to the flash memory, to ensure that the “unwritten pages in any of the erase blocks” counter is consistent, the storage device updates the information reflecting the extent to which a superblock [*e.g.*, **zone**] includes unwritten pages in any of the erase blocks [**host accessible pages available for writing**].

111. To the extent Ellis does not expressly teach this limitation, it was obvious in view of Son. Son discloses additional metadata that should be tracked and evaluated in connection with management operations, such as the “free page count,” “valid page count,” and “invalid page count” of a superblock. EX1006, ¶¶65, 80; *see also id.*, ¶47. At least the free page count is **information representing the extent to which the specified zone includes host accessible pages available for writing** because it indicates many free pages are available for writing. Son further teaches that following the write execution, the superblock “valid page count, invalid page count, [and] next page,” are “update[d].” EX1006, ¶¶50-52. A POSITA would understand Son to disclose, or at least suggest, that the storage device is operable to update the information about the superblocks that is stored in the SSD

module.

112. A POSITA would have also been motivated to apply Son's teaching to Ellis's storage device to update the number of free pages in a super block after a write operation. As discussed above, Son discloses that it is useful to evaluate this information in connection with write operations and management operations such as garbage collection and wear leveling. See Ground 1, Limitations [1.a.ii]. A POSITA would have also recognized that the information must be updated after a write operation so that it remains accurate.

- (k) [1c] wherein the logic comprises at least one of hardware logic or instructions stored on non-transitory machine-readable media that, when executed, control operation of hardware circuitry.

113. In my opinion, Ellis discloses this limitation. For instance, Ellis discloses a memory controller [**logic**] that comprises software instructions that control operation of hardware. See Ground 1, Limitation 1b.

2. Claim 2

- (a) The storage device of claim 1, wherein when the specified zone has no host accessible pages available for writing after writing data to the set of erase units of the specified zone using the multi-plane write command, the storage device is operable to update a state of the specified zone to the full state.

114. In my opinion, Ellis discloses that the storage device is operable to update a state of a superblock [**zone state**] to the closed state [**full state**] when the specified zone has no host accessible pages available for writing. See Ground 1,

Limitation 1a.iii. For instance, Ellis discloses a superblock is in a full state when it has no host accessible pages and a POSITA would have understood that this information needs to be updated so that it remains accurate after a write command.

115. I also showed above that it was at least obvious to transition the superblock to a full state in response to a “multi-plane” write command. See Limitation 1b.ii.

3. Claim 4

- (a) The storage device of claim 1, wherein the zone size is a minimum memory size associated with the management command.

116. I showed above that it would have been obvious in view of Son to modify Ellis’s storage device so that the host issues wear leveling and garbage collection commands [**management commands**] to the storage device. See Ground 1, Limitation 1.a.ii.

117. These management commands are associated with the memory size of at least a superblock. For instance, Figure 7 of Son discloses management commands for wear leveling that selects, for instance, a worn superblock and target superblock (step 710), and copies the contents of the worn superblock to a target superblock and erases the worn superblock (step 715). EX1006, Fig. 7, ¶80-83. Thus, a POSITA would understand that the management command specifies at least a superblock and is associated with the memory size of that superblock [**a memory**

size of a zone].

4. Claim 5

- (a) The storage device of claim 1, wherein the management command from the host results in the logic issuing erase commands to the set of erase units of the specified zone that are on two or more different planes.

118. As discussed above, the prior art includes express teachings, suggestions and motivations to modify Ellis's memory system so that the memory controller is operable to receive a management command from the host, as taught in Son. See Ground 1, Limitation 1.a.ii. For instance, this type of shared management relieves the controller of certain responsibilities and allows the host device to incorporate device conditions when scheduling management commands.

119. Son also discloses that the management commands may be commands for wear leveling and garbage collection that result in the memory controller **[logic]** issuing erase commands to the set of erase blocks **[erase units]** of the specified superblock **[specified zone]**. See Ground 1, Limitation 1.a.ii.

120. Further, the combined Ellis-Son system would continue to organize superblocks as a set of erase blocks that are on different planes, as taught in Ellis. See Ground 1, Limitation 1.b.ii.

5. Claim 6

- (a) The storage device of claim 1, wherein the information about the zones is stored on non-volatile memory of the storage device.

121. Son discloses that the information about the zones is stored in “persistent storage such as the same SSD.” *See* Ground 1, Limitation 1b.i.; *see also* EX1006, ¶34. “Persistent storage” discloses non-volatile memory.

6. Claim 7

- (a) The storage device of claim 1, wherein the information about the zones is stored on a die that includes the logic.

122. Figure 2 of Ellis illustrates memory controller 104 that includes a storage unit 204 for the “control firmware” [**the logic**]. EX1004, 4:4-12, Fig. 2. A POSITA would understand that the memory controller and its storage unit is included on a die.

123. Ellis also discloses that the memory controller 104 includes a host interface. EX1004, 4:4-12, Fig. 2. The host interface “allows the host system 108 of FIG. 1 to interface” and exchange information “with the memory controller 104.” *Id.* 4:46-50, Fig. 1. As discussed above, in the combined Ellis-Son system, the information transmitted to the host would include information about the superblocks [**information about the zones**] that is stored in “persistent storage.” Ground 1, Limitation 1b.i.; *see also* EX1006, ¶34. A POSITA would understand that the transmission process would involve the memory controller retrieving the information from persistent storage and buffering it for transmission to the host. At

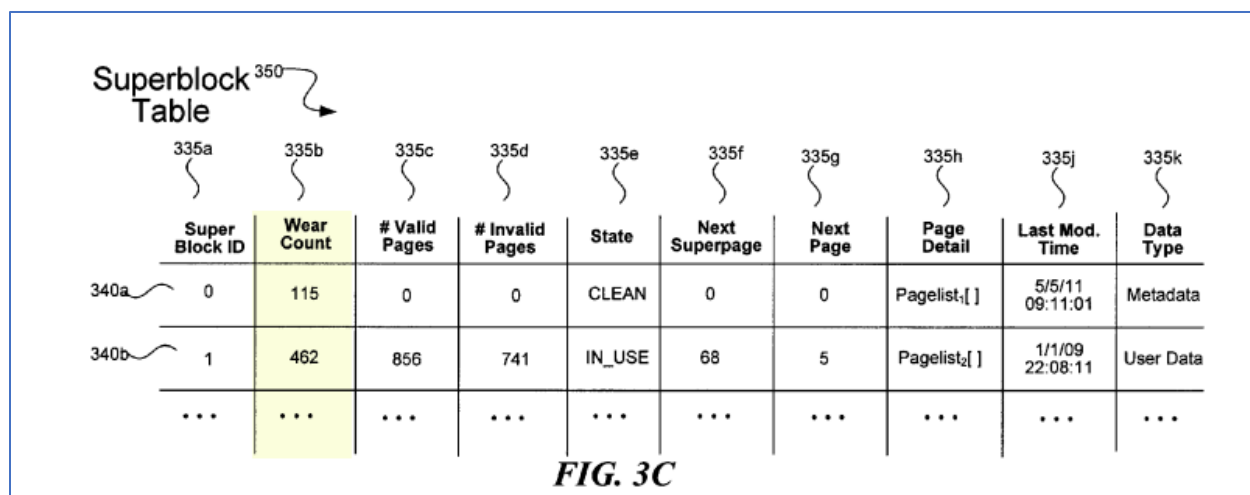
minimum, a POSITA would have found it obvious to buffer and queue the information before transmission, for example, to handle bursts of data or the need to batch data together. EX1012, ¶32, Fig. 1. Thus, a POSITA would understand that the information about the zones is buffered on the die for the memory controller that includes the logic.

7. Claim 8

- (a) The storage device of claim 1, wherein the information about the zones comprises information indicative of a number of erase operations performed on a particular zone of the zones.

124. In my opinion, Ellis discloses this information. For instance, Ellis discloses that the memory controller tracks the program erase-count [**number of erase operations**] performed on a particular superblock [*e.g.*, **a particular zone**]. EX1004, 12:60-12 (“metadata pages need to be managed by the block control module” and “can include a program-erase count”), 13:15-24, 14:15-20.

125. In my opinion, Son also discloses that the memory controller tracks the wear count. EX1006, ¶39, Fig. 3C. Son states that the “wear count column indicates how many times a particular superblock has been erased.” [**number of erase operations**]. EX1006, ¶39. The wear count column is illustrated in the superblock table of Figure 3C of Son:



EX1006, Fig. 3C. Son teaches that the superblock table (including the wear count) may be returned in response to the host query. *See* Limitation 1b.i; *see also* EX1006, ¶¶47, 65, 80.

126. As I discussed above, in the combined Ellis-Son system metadata in the superblock table is returned to the host in response to a host-query so that the host may issue maintenance commands. *See* Ground 1, Limitation 1a.ii.

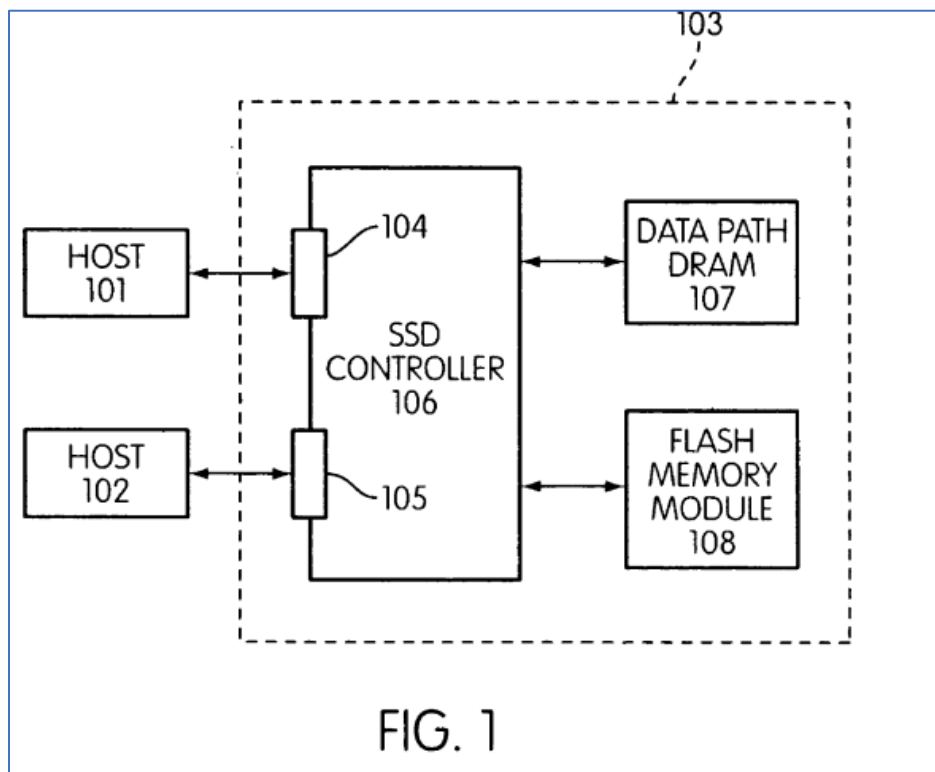
X. GROUND 2: CLAIMS 3, 10-14, 16-19, 21-29 ARE OBVIOUS IN VIEW OF ELLIS (EX1004), SON (EX1006) AND OLBRICH (EX1010)

A. Overview of the Prior Art

1. Olbrich (EX1010)

127. U.S. Patent 8,621,137 (“Olbrich”) was filed on April 8, 2008, was published on July 2, 2009, and issued on December 31, 2013. I understand that it is prior art because its filing date and publication date pre-date the earliest possible priority date of the ’614 patent.

128. Olbrich discloses a storage system comprising an SSD Controller in communication with one or more hosts and a flash memory module:



EX1010, Fig. 1.

129. The storage device is subdivided into superblocks that are comprised of superpages, and the system maintains information about the superblocks in a “superblock metadata table.” EX1010, 47:55-62, Fig. 35. An exemplary superblock metadata table is shown below and includes the time when the super page was last written (3505), whether a superpage is defective (3510), the logical block address of a superpage (3511), an open flag (3508) indicating the superblock is open for writing, a closed flag (3509) and time stamp (3506) that indicates when the superblock was closed for writing, and an erase count for the superblock (3507).

3502		3505 SUPER PAGE TIME STAMP	3510 DEFECT	3511 LBA
3503 BANK 0	SUPER PAGE 0			
	SUPER PAGE 1			
	⋮			
	SUPER PAGE 63			
3504 BANK 1	SUPER PAGE 0			
	SUPER PAGE 1			
	⋮			
	SUPER PAGE 63			
⋮				
BANK n	SUPER PAGE 0			
	SUPER PAGE 1			
	⋮			
	SUPER PAGE 63			
3506 — SUPER BLOCK TIME STAMP _____ 3507 — ERASE COUNT _____ 3508 — OPEN FLAG _____ 3509 — CLOSED FLAG _____				

SUPER BLOCK METADATA TABLE 3501

EX1010, Fig. 35, 47:55-48:46.

130. Olbrich also discloses that superpages of a superblock consist of pages spread across multiple dies (or planes). EX1010, 15:58-67. The host can issue a multi-plane write to these pages using the “same address location.” *Id.*

2. Claim 3

- (a) The storage device of claim 1, wherein the logic is operable to write the data to the two or more erase units of the set of erase units of the specified zone using a common page address.

131. In my opinion, Ellis discloses, or at least renders obvious, a multi-plane write command that is operable to write the data to the two or more erase units of

the set of erase units of the specified zone for the reasons I discussed above. *See* Limitation 1b.ii.

132. In my opinion, a POSITA would have also understood that multi-plane commands utilize a common page address. As Hu explains, SSD manufacturers at the time had “multi-plane commands” and these were defined as command in which the “blocks executing a multi-plane” command have the “same... page addresses.” EX1014, §2.2.

133. To the extent Ellis alone does not expressly disclose or suggest a multi-plane command that uses a common page address, it was at least obvious that the multi-plane command would use a common page address in view of Olbrich, which discloses a multi-plane write command and expressly states that it uses the “same address location” [**common page address**] for writing to pages that are on different dies. EX1010, 15:60-67, Fig. 12. For instance, Figure 12 shows “SuperPage 30 is spread across four separate dies” and consists of “four flash Pages” that have the “same address location” across planes. *Id.*, Fig. 12. A POSITA would understand that pages “spread across four separate dies” are on multiple planes because a plane is a subdivision of a die, such that the combined system would write concurrently to pages within erase blocks on different planes.

134. The prior art also includes express teachings, suggestions and motivations to combine Olbrich’s teachings and the Ellis-Son storage device such

that the storage device performs a multi-plane write command using the “same address location” across planes. For instance, Hu discloses that multi-plane commands in which the “blocks executing a multi-plane” command have the “same... page addresses” (EX1014, §§1.3, 2.2) “improve the performance of SSDs by handling read, write and erase operations more efficiently” (EX1014, §1.3, “Advanced commands”).

135. Further, the combination reflects the application of a known technique (multi-plane commands which use a common page address) to a known device (Ellis-Son’s flash memory system) ready for improvement (improving parallelism of Ellis-Son’s base device) to yield predictable results (a device that supports multi-plane commands).

136. A POSITA would also have had a reasonable expectation of success in combining the Ellis-Son storage device with Olbrich’s teachings because it represents a straightforward implementation of multi-plane write commands to similar flash devices. As I explained earlier, by the time of the ’614 patent SSD manufacturers had already implemented multi-plane commands using common page addresses in their SSDs.

3. Claim 10

(a) [10pre] A storage device comprising:

137. This limitation is disclosed for the reasons I discussed above. *See*

Ground 1, Limitation 1[pre].

- (b) [10a] flash memory divided into zones, each zone of the zones being a subdivision comprising a set of erase units, and each erase unit of the set of erase units comprising host accessible pages, wherein each zone:

138. This limitation is disclosed for the reasons I discussed above. *See*

Ground 1, Limitation 1[a].

- (c) [10a.i] is associated with a logical address range,

139. This limitation is disclosed for the reasons I discussed above. *See*

Ground 1, Limitation 1[a.i].

- (d) [10a.ii] has a zone size associated with a management command from a host,

140. This limitation is disclosed for the reasons I discussed above. *See*

Ground 1, Limitation 1[a.ii].

- (e) [10a.iii] transitions to a full state in response to no longer having host accessible pages available for writing, and

141. This limitation is disclosed for the reasons I discussed above. *See*

Ground 1, Limitation 1a.iii.

- (f) [10a.iv] in response to the storage device receiving the management command associated therewith, transitions to an empty state such that all host accessible pages are available for writing; and

142. In my opinion, Ellis alone, or in further view of Son, discloses or renders obvious that a superblock [**zone**] may be in an empty state. *See* Ground 1, Limitation 1a.iv.

143. Further, in the combined Ellis-Son system the storage device would receive a management command that specifies a superblock for erasure from the host, as I discussed above. See Ground 1, Limitation 1a.ii. A POSITA would understand that the combined Ellis-Son system would transition the superblock to the empty state after the erase operation so that the state of the superblocks remains accurate. Otherwise, the state information would be inaccurate after an erasure.

(g) [10b] logic operable to:

144. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1b.

(h) [10b.i] return information about zones stored by the storage device in response to a query command from the host, the information about the zones comprising:

145. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1b.

(i) [10b.ii] a list of the zones accessible by the host,

146. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1b.

(j) [10b.iii] a respective logical address associated with a zone of the zones stored,

147. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1b.

(k) [10b.iv] the zone size,

148. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1b.

(l) [10b.v] information indicating, for respective zones in which there are no host accessible pages available for writing the respective zone, that the respective zones are in the full state, and

149. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1b.

(m) [10b.vi] information representing, for respective zones, an extent to which the respective zones include host accessible pages available for writing;

150. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1b.

(n) [10b.vii] write data to the set of erase units of a specified zone using a common page address;

151. This limitation is disclosed for the reasons I discussed above. See Ground 1, Claim 3.

(o) [10b.viii] update the information representing the extent to which the specified zone includes host accessible pages available for writing; and

152. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1[b.iii].

(p) [10b.ix] when the specified zone has no host accessible pages available for writing the zone after writing data to the set of erase units of the specified zone, update the

information indicating that the specified zone is in the full state,

153. This limitation is disclosed for the reasons I discussed above. See Ground 1, Claim 2.

- (q) [10c] wherein the logic comprises at least one of hardware logic or instructions stored on non-transitory machine-readable media that, when executed, control operation of hardware circuitry.

154. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1[c].

4. Claim 11

- (a) The storage device of claim 10, wherein the logic is operable to write the data to the set of erase units of the specified zone using a multi-plane write command such that the data is concurrently written to two or more erase units of the set of erase units.

155. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1b.ii.

5. Claim 12

- (a) The storage device of claim 10, wherein the management command from the host results in the logic issuing at least one erase command to erase one or more erase units of the set of erase units of the specified zone.

156. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1a.ii, 10a.iv.

6. Claim 13

- (a) The storage device of claim 10, wherein the management command from the host results in the logic issuing an erase command associated with erase units of the specified zone that are on two or more different planes.

157. I showed above that in the Ellis-Son combination the host instructs the memory controller [**logic**] to issue an erase command associated with erase blocks [**erase units**] of the specified superblock [*e.g.*, **zone**]. See Ground 1, Limitation 1a.ii.

158. I also showed that Ellis discloses or render obvious that the erase blocks [**erase units**] of the specified superblock [**specified zone**] are on two or more different planes. See Ground 1, Limitation 1b.ii.

7. Claim 14

- (a) The storage device of claim 13, wherein, as a result of issuing the erase command, the storage device is operable to detect a bad erase unit and mark the bad erase unit as unusable for storage of new write data.

159. Ellis discloses that bad blocks are “determined to be bad during normal operation of the storage device.” EX1004, 11:41-45. For instance, Ellis explains that the memory system may include “factory marked bad erase blocks and erase blocks that have been determined to be bad during normal operation of the storage device.” EX1004, 11:41-45. A POSITA would understand that operations occurring during normal operation include erases and thus a system can detect bad blocks as a result of an erase operation. Furthermore, a POSITA would understand from this disclosure that a block that is determined to be bad is marked so that it is not used

for storage of new write data. At minimum, it would have been obvious to mark a bad block as unusable because the system would need to know not to write data to blocks that are bad and can no longer reliably store data.

160. A POSITA would also understand Ellis's disclosure of detecting bad blocks "during normal operation" to teach or suggest that the bad blocks may be determined as a result of an erasure command. A POSITA understood at the time that erasure is part of the normal operation of the device during which bad blocks may be determined. EX1016, ¶¶241-243(disclosing "a process for detection and handling of bad pages and bad blocks" where "*[a]fter the block is erased, the status of each page in that block is checked" to determine if it is bad.*"). In fact, a POSITA would have recognized that erasure performance can help identify bad blocks, such that integrating bad block detection with erasure improves efficiency and accuracy. See, e.g., EX1017, 6:21-7:7. For instance, a "block needs more than predetermined number of erase cycles applied to fully erase the block," it is an indication that the block has failed and should be marked bad. *Id.*

8. Claim 16

- (a) The storage device of claim 10, further comprising a host interface that comprises buffers to receive the information about the zones and queue the information about the zones for transmission to the host.

161. As I discussed above, the Ellis-Son system includes a host interface that transmits information about the zones to the host in response to a query from the

host. See Ground 1, Limitation 1b.i. A POSITA would have understood that the information about the zones may need to be buffered and queued at the host interface prior to transmission, for example to handle bursts of data or the need to batch data together.

162. To the extent the Ellis-Son system does not render this limitation obvious, it was at least obvious in view of Olbrich. Olbrich discloses that the “host interface” comprises, among other things, a “transmit stage buffer 1709” that is used to queue outgoing data for transmission to the host. EX1010, 20:31-54, Fig. 17.

163. A POSITA would have also been motivated to combine Olbrich’s teaching with the Ellis-Son storage device because using a buffer to queue information for transmission from flash memory to a host is a common design that optimizes performance and error handling.

164. A POSITA would have had a reasonable expectation of success because the use of buffers for queuing transmission is well-known and its implementation is straightforward.

9. Claim 17

- (a) The storage device of claim 10, wherein the information about the zones is stored on non-volatile memory of the storage device.

165. This limitation is disclosed for the reasons I discussed above. See Ground 1, Claim 6.

10. Claim 18

- (a) The storage device of claim 10, wherein the information about the zones is stored on a die that includes the logic.

166. This limitation is disclosed for the reasons I discussed above. See Ground 1, Claim 7.

11. Claim 19

- (a) The storage device of claim 10, wherein the information about the zones comprises information indicative of a number of erase operations performed on a particular zone of the zones.

167. This limitation is disclosed for the reasons I discussed above. See Ground 1, Claim 8.

12. Claim 21

- (a) [21pre] A storage device comprising:

168. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1pre.

- (b) [21a] flash memory divided into zones, each zone of the zones being a subdivision comprising a set of erase units, each erase unit of the set of erase units comprising host accessible pages, and the zones are each associated with a respective logical address range; and

169. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1a.i.

- (c) [21b] a memory controller in communication with the flash memory, the memory controller comprising:

170. In my opinion, Ellis discloses a memory controller [**memory**

controller 104] in communication with the flash memory [**flash memory 106**]. See Ground 1, Limitation 1pre.

- (d) [21b.i] storage arranged to retain information about the zones, the information about the zones comprising zone size, a logical address associated with a zone of the zones, and information representing an extent to which the zone of the zones comprises host accessible pages available for writing; and

171. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1b.i.

- (e) [21b.ii] logic operable to cause the storage device to output the information about the zones in response to a query command from a host, write to the set of erase units of the zone of the zones using a common page address, and update the information representing the extent to which the zone comprises host accessible pages available for writing in response to writing to the set of erase units of the zone.

172. As I discussed above, in the Ellis-Son combination the memory controller contains software and hardware [**logic**] operable to cause the storage device to output the information about the superblocks [**zones**] in response to a query command from a host. See Ground 1, Limitation 1b.i.

173. As I also discussed above, Olbrich further renders obvious writing to the set of erase blocks [**erase units**] of a superblock [**zone**] using a common page address. See Ground 2, Claim 3.

174. In my opinion, the Ellis-Son-Olbrich combination further discloses

updating the information representing the extent to which the zone comprises host accessible pages available for writing in response to writing to the set of erase units of the zone. *See* Ground 1, Limitation 1b.iii. Ellis and Son both disclose maintaining metadata regarding the number of free pages in a superblock, as I discussed above, and a POSITA would have been motivated to update this information after a write command so that it remains accurate.

13. Claim 22

- (a) The storage device of claim 21, wherein the information about the zones comprises a state of the zone.

175. This limitation is disclosed for the reasons I discussed above. *See* Ground 1, Limitation 1b.i.

14. Claim 23

- (a) The storage device of claim 22, wherein the state of the zone is a full state when the zone has no host accessible pages available for writing the zone.

176. This limitation is disclosed for the reasons I discussed above. *See* Ground 1, Limitation a.iii.

15. Claim 24

- (a) The storage device of claim 23, wherein the state of the zone is updated to the full state in response to the zone having no host accessible pages available for writing the zone after writing to the set of erase units of the zone of the zones using the common page address.

177. In my opinion, the Ellis-Son combination discloses tracking whether a

superblock is in a full state. *See* Ground 1, Limitation 1a.iii.

178. I also showed that in view of Olbrich it was obvious to write to the set of erase block [**erase units**] of the superblock [**zone**] using a common page address. *See* Ground 1, Claim 3.

179. A POSITA would have understood that when a superblock has no host accessible pages available for writing, for instance at the completion of a write operation, the state of the superblock should be updated to reflect the change so that the superblock status remains accurate.

16. Claim 25

- (a) The storage device of claim 22, wherein the state of the zone is an empty state when all host accessible pages of the zone are available for writing.

180. This limitation is disclosed for the reasons I discussed above. *See* Ground 1, Limitation 1a.iv.

17. Claim 26

- (a) The storage device of claim 21, wherein the information about the zones comprises information indicative of an amount of time since data was written to the set of erase units of the zone.

181. As I discussed above, the combined Ellis-Son system returns information about the superblocks [**zones**] in response to a host query. *See* Limitation 1b.i; *see also* EX1006, ¶¶47, 65, 80. Son discloses that the information may include a last modification time for the superblock [**information indicative of**

an amount of time since data was written to the set of erase units of the zone]:

	335a Super Block ID	335b Wear Count	335c # Valid Pages	335d # Invalid Pages	335e State	335f Next Superpage	335g Next Page	335h Page Detail	335j Last Mod. Time	335k Data Type
340a	0	115	0	0	CLEAN	0	0	Pagelist ₁ []	5/5/11 09:11:01	Metadata
340b	1	462	856	741	IN_USE	68	5	Pagelist ₂ []	1/1/09 22:08:11	User Data
...	...	...	...	...	...	...	...	...	...	...

FIG. 3C

EX1006, ¶39, Fig. 3C.

18. Claim 27

- (a) The storage device of claim 21, wherein the information about the zones comprises logical addresses associated with respective ones of a plurality of zones of the zones, and information representing an extent to which each of the plurality of zones comprises host accessible pages available for writing.

182. This limitation is disclosed for the reasons I discussed above. See Ground 1, Limitation 1b.i

19. Claim 28

- (a) The storage device of claim 21, wherein the zone is a minimum memory space associated with a management command from the host.

183. This limitation is disclosed for the reasons I discussed above. See Ground 1, Claim 4.

20. Claim 29

- (a) The storage device of claim 21, wherein the logic is operable to cause the storage device to concurrently erase the set of erase units of the zone.

184. In my opinion, Ellis discloses that the memory controller is operable to cause the memory system to concurrently erase the superblock. EX1004, 8:4-8, 14:35-36, 17:31-36. For instance, Ellis states that all the erase blocks 408 in each of the superblocks 502 can be erased at the same time.” *Id.*, 17:31-36.

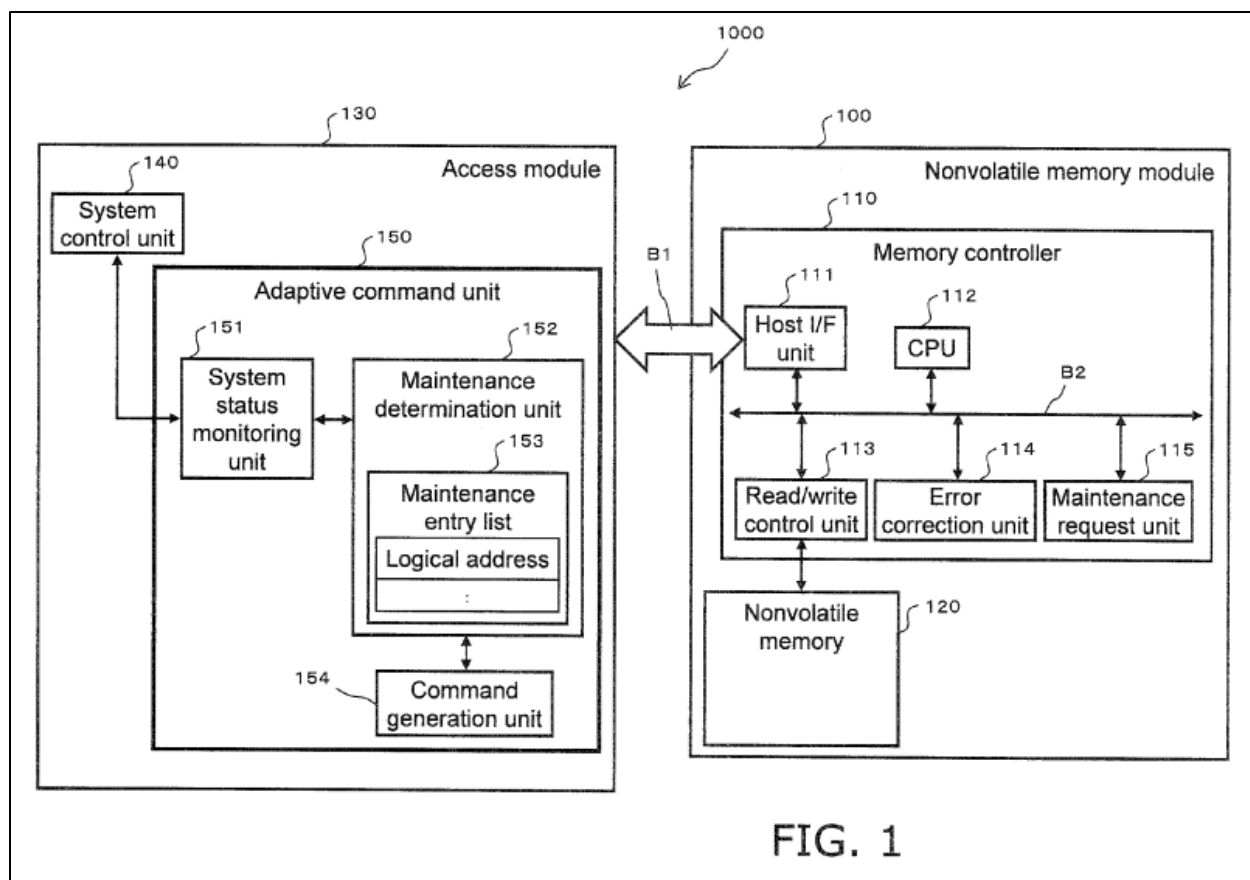
XI. GROUND 3: CLAIMS 9, 15, 20, 30, ARE OBVIOUS IN VIEW OF ELLIS (EX1004), SON (EX1006), OLBRICH (EX1010), AND YAMADA (EX1015)

A. Overview of the Prior Art

1. Yamada (EX1015)

185. U.S. Pat. Pub. No. 2011/0271032A1 (“Yamada”) was filed on July 27, 2010 and was published November 3, 2011. I understand that it is prior art because its filing date and publication date pre-date the earliest possible priority date of the ’614 patent.

186. Yamada teaches an SSD system where memory management is shared between the host and controller. Figure 1 illustrates a nonvolatile memory module that includes a “memory controller” comprising a “maintenance request unit,” and an access module (host) that includes a “maintenance command unit.” EX1015, ¶12.



187. The controller identifies a LBA (or group of LBAs) that need maintenance based on comparison of metadata against thresholds. *Id.*, ¶¶13, 32, 80-81. The controller then transmits the LBA (or groups of LBAs) to the host via a maintenance request, and the host may then issue a maintenance command. *Id.*, ¶14, 109-111, 116-117. “A single maintenance command may be generated for a plurality of addresses of a plurality of blocks in the nonvolatile memory system,” for instance a grouping of blocks like a superblock. *Id.*, ¶117.

B. Claim Analysis

1. Claim 9

- (a) The storage device of claim 1, wherein the storage device is operable to: generate metadata indicating an amount of time since the specified zone was written to, and

188. Ellis teaches a management command referred to as “recycle operation” which rewrites data from one superblock to another. To identify a superblock to copy, Ellis teaches that the controller tracks time since data was programmed and identifies “*superblocks 502 that are older or written a longer time in the past.*” EX1004, 19:4-15. The information indicating whether a superblock was written “a longer time in the past” is metadata indicating an amount of time since the specified superblock [**zone**] was written to.

189. A POSITA would have also found it obvious to generate metadata indicating the “last modification time” for the superblock. References like Son and Yamada disclose the need to track this metadata. EX1006, ¶39, Fig. 3C (superblock table with “last mod. time”); EX1015, ¶164 (maintaining metadata indicating the “elapsed time from when the data has been written previously based on the write history”). Indeed, a POSITA would understand that charge in the memory cells will leak over time such that data may become corrupted if it is not written for a long time. EX1015, ¶9 (“Data loss can occur in an area that has not been refreshed for a sufficiently long period of time.”). Thus, a POSITA would understand that data needs to be “refreshed” (rewritten) after a block has not been written to for a threshold amount of time to avoid this type of data loss. *Id.* A POSITA would have,

therefore, found it obvious to modify Ellis’s storage device to maintain this type of metadata.

- (b) in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a second management command directed to the specified zone.

190. As I explained above, a POSITA would have found it obvious to modify Ellis’s storage device so that metadata regarding last write times is maintained and a maintenance command is issued to “refresh” (rewrite) data if the last write time exceeds a threshold amount of time to avoid data loss. See Ground 3, Limitation 9[a]. I also showed that in the combined Ellis-Son system the management commands would be issued by the host. See Ground 1, Limitation 1a.ii.

191. To the extent the combined Ellis-Son system does not render this limitation obvious, it is obvious in further view of Yamada. Yamada discloses a memory module [**storage device**] that includes a “maintenance request unit 115” and a access module [**host**] that includes a “maintenance determination unit 152”:

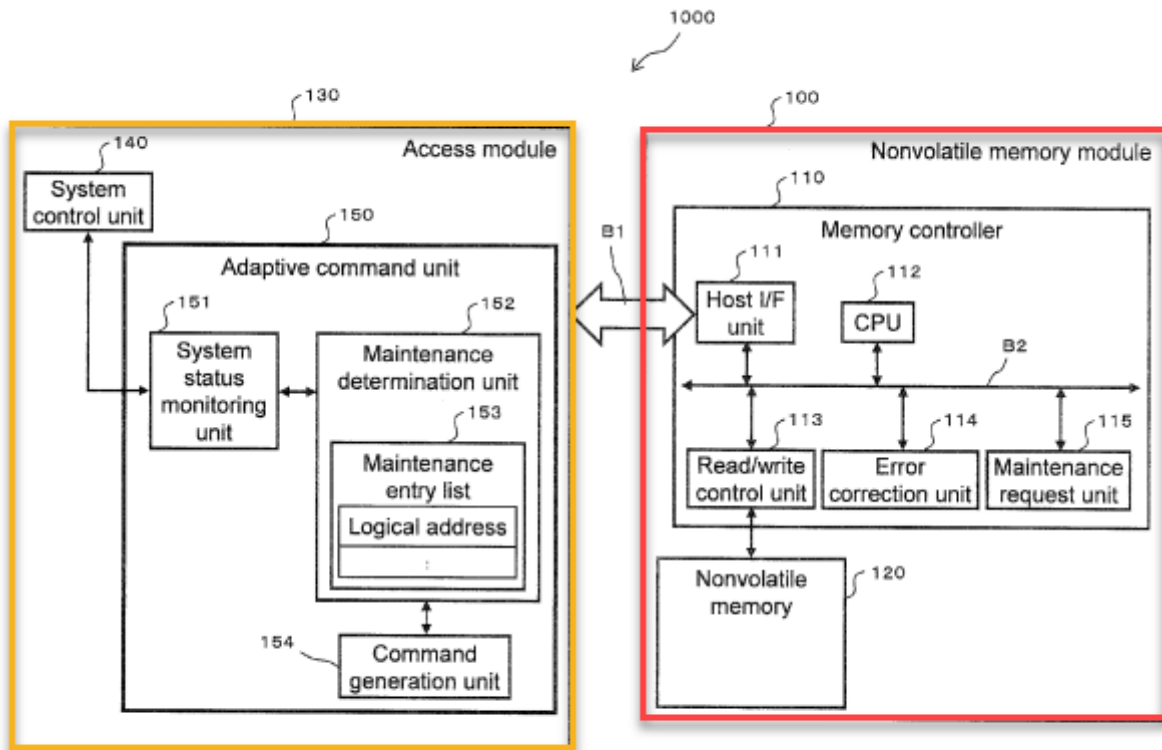


FIG. 1

EX1015, Fig. 1. The maintenance request unit of the memory module compares metadata to a “criterion value” [**threshold**] and sends “a maintenance request indicating that the nonvolatile memory needs maintenance to the access device” when the criterion value is exceeded [**recommends that the host send a second management command**]. EX1015, ¶¶32-33, 81-82, Figs. 1-2. The maintenance determination unit of the access device then maintenance request that “includes refresh and/or wear leveling of the nonvolatile memory.” EX1015, ¶30. Yamada further discloses that the maintenance request “may include a plurality of logical addresses of a plurality of blocks,” such that the “area that needs maintenance

consists of... a plurality of blocks.” EX1015, ¶116; *see also id.*, ¶¶117 (“A single maintenance command may be generated for a plurality of addresses of a plurality of blocks in the nonvolatile memory system”), 118 (“The maintenance may be performed for a plurality of logical blocks”). Thus, in view of Yamada, it was obvious for the host to send a second management command directed at a plurality of logical blocks, such as Ellis’s superblocks [**second management command directed to the specified zone**].

192. Although in the preferred embodiment of Yamada the maintenance request is generated based on the number of errors occurring in the data exceeding a threshold, a POSITA would understand that the criterion used to determine maintenance may be any metadata that would indicate a need for a maintenance operation. For instance, as explained above, “data loss can occur in an area that has not been refreshed for a sufficiently long period of time.” EX1015, ¶9. Thus, it would have been obvious for the maintenance request to be generated based on the last modification time of a superblock exceeding a threshold value.

193. **Motivation to Combine.** A POSITA would have been motivated to incorporate Yamada’s teaching regarding notifying a host to schedule maintenance commands when a metadata threshold is exceeded.

194. First, Ellis, Son, Olbrich and Yamada involve similar techniques to solve the same problem, *i.e.*, collecting, storing and using metadata for maintenance

activities to improve the efficiency, performance and “integrity of data” in NAND flash memory systems. EX1015, ¶¶5-7, 11. They are thus in the same field of endeavor.

195. Second, the prior art includes express teachings, suggestions and motivations for combining the references in this manner. Initially, Ellis, Son, and Yamada each disclose the need to maintain metadata that indicates the amount of time since a superblock was last written and the references also teach the need to perform maintenance when the last written time exceeds a threshold. EX1004, 19:4-15; EX1006, ¶81; EX1015, ¶164.

196. Yamada further discloses that it is desirable for the memory controller to perform the comparison and then recommend that the host send the management command. EX1015, ¶¶32-33, 81-82. Yamada explains that this type of shared management in which the memory controller performs the comparison and the host sends the management command is beneficial for a number of reasons. First, it “enables data maintenance to be performed in accordance with the status of the nonvolatile memory system, without requiring the data maintenance to be performed together with normal processing of the system.” EX1015, ¶36. Second, it “enables the maintenance to be performed with a sufficiently long processing time allocated to the maintenance, and improves the data retention properties.” *Id.* Third, “enables the system management status, such as the battery level, to be controlled in a manner

that the maintenance will be completed in a reliable manner.” *Id.*; *see also id.*, ¶10 (“conventional techniques” in which memory controller sends the management command “fails to consider the device conditions”). In light of the teaching in Ellis that memory management responsibility may be shared between the controller and the host, EX1004, 4:58-65; 20:40-44, Son’s teaching of the benefits of sharing responsibility between the host and controller, *see, e.g.*, EX1006, ¶¶13-14, and the benefits described in Yamada, a POSITA would be motivated to incorporate Yamada’s teaching regarding host-initiated maintenance commands which are based on metadata generated by the controller, EX1015, ¶148.

197. Third, a POSITA would have been motivated to implement this type of shared memory management because it avoids the storage device having to expose its performance characteristics to the host. Different memory devices have different performance characteristics such that the optimal threshold (*e.g.*, the last modified time or erase count) for triggering a maintenance command may differ across devices. A vendor may not want to expose performance characteristics to the host, such that a POSITA would have been motivated to have the memory controller of the memory device determine whether a threshold has been exceeded internally and then simply notify the host of the need for a management operation.

198. A POSITA would have also had a reasonable expectation of success moving the management command responsibility from the controller firmware to

the host-side software. By the time of the ‘614 patent the sharing of management operations between the controller and host was well-understood by those in the art and a simple design choice. See Ground 1, Limitation 1a.ii.

2. Claim 15

- (a) [15a] The storage device of claim 13, wherein, the storage device is operable to: generate metadata indicating a count of erasures of the erase units, and

199. Ellis teaches that the controller is configured to store a “**program-erase count**” for each superblock, which is “defined as a numerical value **indicating a number of times** or cycles the memory devices 110 associated with the corresponding instances of the **superblocks 502 are** programmed or **erased**.” EX1004, 11:63-67. See also *id.*, 13:14-24, claim 1. Thus, Ellis teaches a controller with logic to generate “program-erase count,” i.e., “**metadata indicating a count of erasures of the erase units**.”

200. Son also teaches the maintenance of “wear count” for each superblock. EX1006, ¶80. The superblock table includes a “**wear count column indicat[ing] how many times a particular superblock has been erased**.” EX1006, ¶39. Ellis teaches that its memory controller includes a block control module 306 that can perform wear leveling (EX1004, 4:66-5:15, 16:7-13, Fig. 3) and a POSITA would understand that implementation of wear leveling involves maintaining the wear count for each superblock.

- (b) [15b] in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a second management command directed to the specified zone.

201. As mentioned above, Ellis teaches that its memory controller includes a block control module 306 that can perform wear leveling. EX1004, 4:66-5:15, 16:7-13, Fig. 3. The block control module can use the “program-erase count” [**the metadata**] to “determine if wear begins to deteriorate the integrity of the memory devices.” EX1004, 13:15-25. A POSITA would understand this disclosure to teach, or at least fairly suggest, that the block control module determines deterioration by comparing whether the program-erase count has exceeded a threshold. At minimum, it would have been obvious for Ellis’s storage device to perform this threshold comparison in view of Son, which discloses that wear leveling involves comparing a wear count to a threshold value. EX1006, Fig. 7, ¶¶64, 79. And, as discussed above, in the combined Ellis-Son system the host would be responsible for sending the wear leveling command when the threshold is exceeded. See Ground 1, Limitation 1a.ii.

202. To the extent Ellis-Son do not render obvious this limitation, it was obvious in further view of Yamada. As I discussed in connection with Claim 9, Yamada discloses a maintenance request unit within a memory controller that sends a “maintenance request” to the host when a criterion value is exceeded. EX1015, ¶32; see also *id.* ¶33. While in the preferred embodiment of Yamada the criterion

value used is the number of errors, a POSITA would understand that the criterion value could include other information, such as wear count, that indicates the need for maintenance.

203. A POSITA would have been motivated to incorporate Yamada's teaching regarding notifying a host to initiate maintenance commands upon exceeding a threshold and had a reasonable expectation of success for the reasons discussed in connection with Claim 9. *See* Ground 2, Claim 9.

3. Claim 20

- (a) [20a] The storage device of claim 10, wherein the storage device is operable to: generate metadata indicating an amount of time since the specified zone was written to, and

204. This limitation is disclosed for the reasons I discussed above. *See* Ground 3, Claim 9[a].

- (b) [20b] in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a second management command directed to the specified zone.

205. This limitation is disclosed for the reasons I discussed above. *See* Ground 3, Claim 9[b].

4. Claim 30

- (a) [30a] The storage device of claim 21, wherein the storage device is operable to: generate metadata indicating an amount of time since the zone was written to, and

206. This limitation is disclosed for the reasons I discussed above. *See*

Ground 3, Claim 9[a].

- (b) [30b] in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a management command directed to the zone.

207. This limitation is disclosed for the reasons I discussed above. *See*

Ground 3, Claim 9[b].

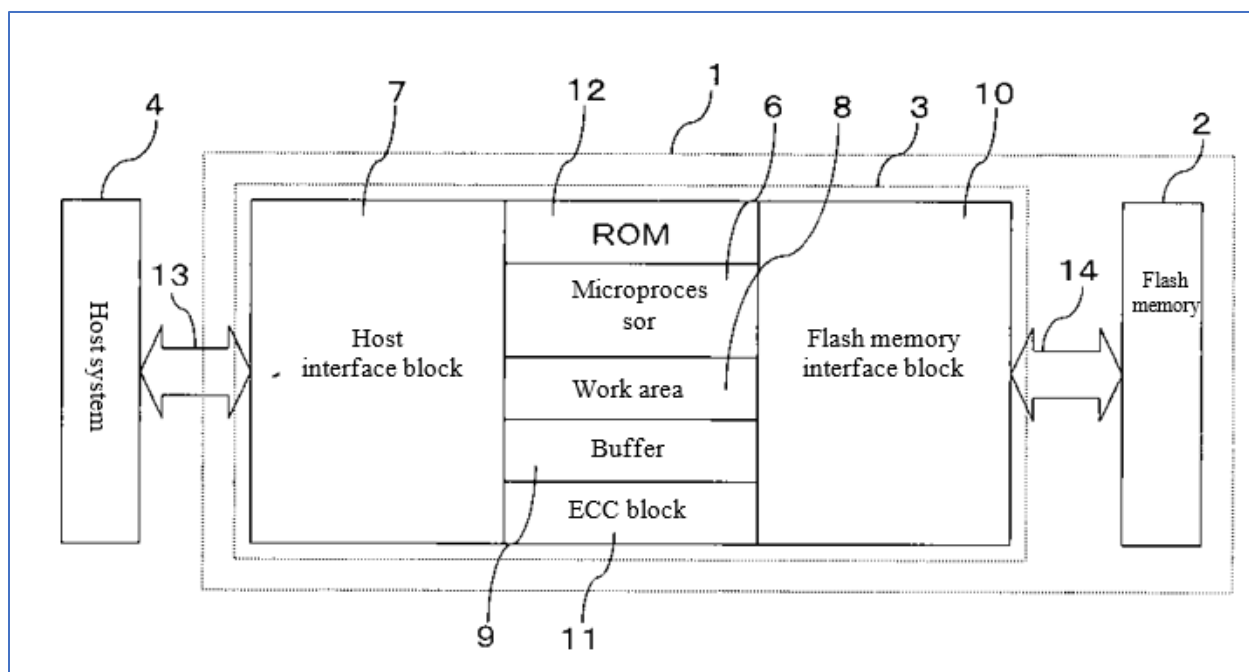
XII. GROUND 4: CLAIMS 1-2, 4-8 ARE OBVIOUS IN VIEW OF TAKEO (EX1012), SON (EX1006), AND SINCLAIR (EX1005)

A. Overview of the Prior Art

1. Takeo (EX1012)

208. Japanese Patent Publication No. JP2008-176606 (“Takeo”) was filed on January 19, 2007 and was published on July 31, 2008. I understand that it is prior art because its filing date and publication date pre-date the earliest possible priority date of the ’614 patent.

209. Takeo discloses a storage system designed to “efficiently transfer internal information within a flash memory system... to the host system side in response to a request from the host system side.” EX1012, Abstract. The flash memory of Takeo is illustrated in Figure 1.



EX1012, Fig. 1, ¶¶41-43. The memory is subdivided into “zones,” which Takeo explains are comprised of blocks that are in turn comprised of pages. *Id.*

210. In response to receiving a host query, referred to as “a vendor command,” the memory controller will write internal information regarding the flash memory to buffer 9 and then return that information to the host system 4 via the host interface 7. EX1012, Abstract; *see also id.*, ¶¶44-50.

211. The information returned to the host may include, for example, the information in Figure 4, such as the number of bad and empty blocks per zone:

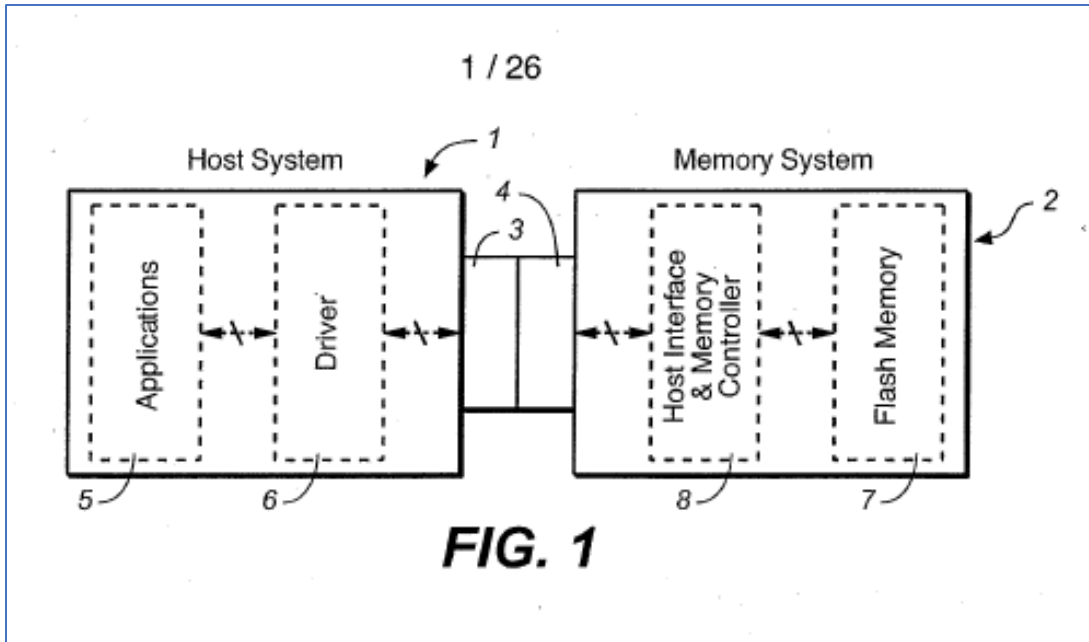
PZN	Number of bad blocks	Number of empty blocks
#0	3	10
#1	2	17
#2	4	12
#3	1	11
#4	3	15
#5	2	16
#6	3	14
#7	2	22

EX1012, Fig. 4; ¶¶44-50.

2. Sinclair (EX1005)

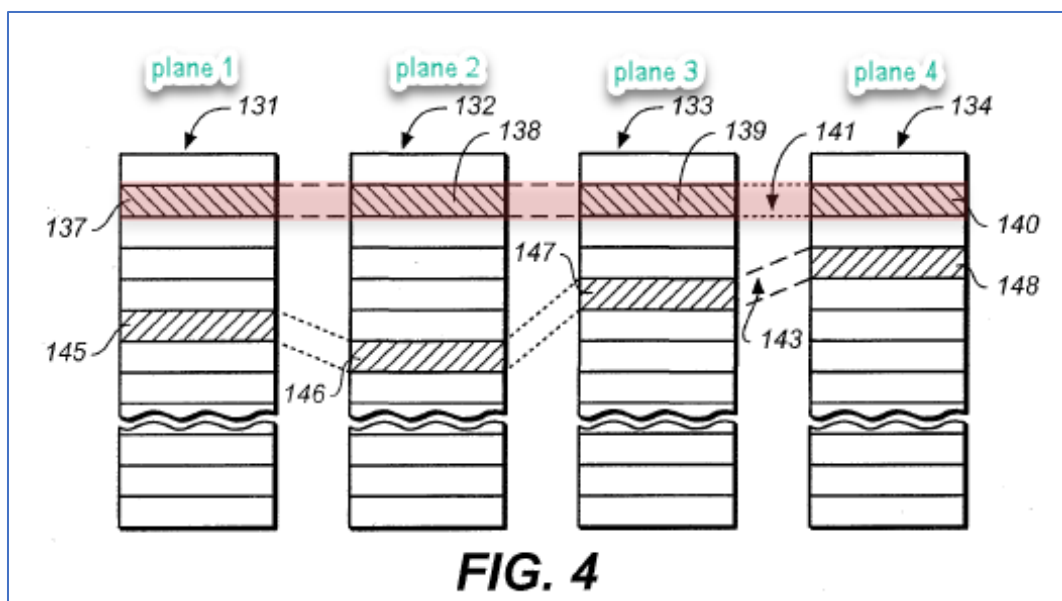
212. International Publication No. WO 2008/082996 (“Sinclair”) was filed on December 19, 2007 and was published on July 2008. I understand that it is prior art because its filing date and publication date pre-date the earliest possible priority date of the ’614 patent.

213. Sinclair discloses a host system that communicates with a memory system:



EX1005, Fig. 1.

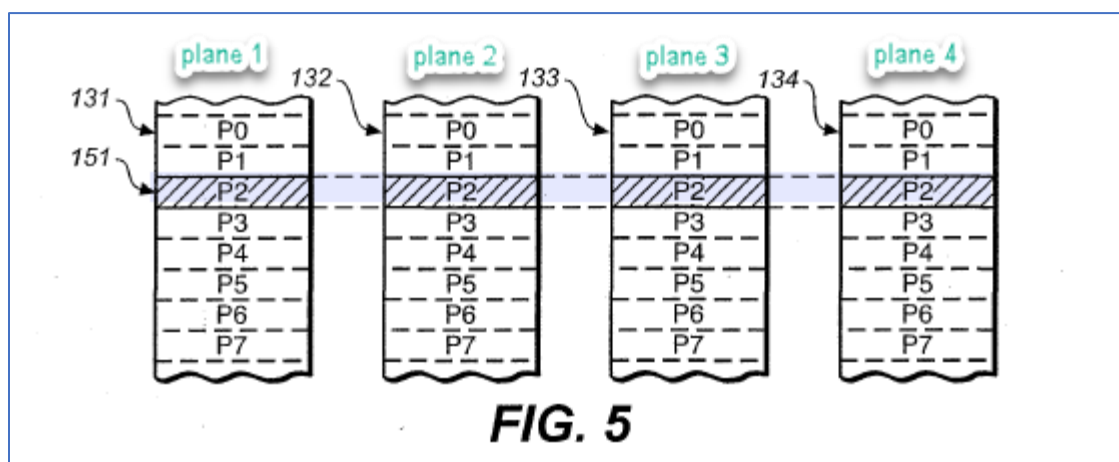
214. The memory system includes a flash memory. To increase parallelism, the flash memory may be organized into a “metablock,” which comprises a group of blocks across multiple planes that may be operated together (*e.g.*, written and erased together). EX1005, Fig. 4, ¶62. For instance, Figure 4 illustrates a metablock comprising four individual blocks, 137-140:



EX1005, Fig. 4, ¶62.

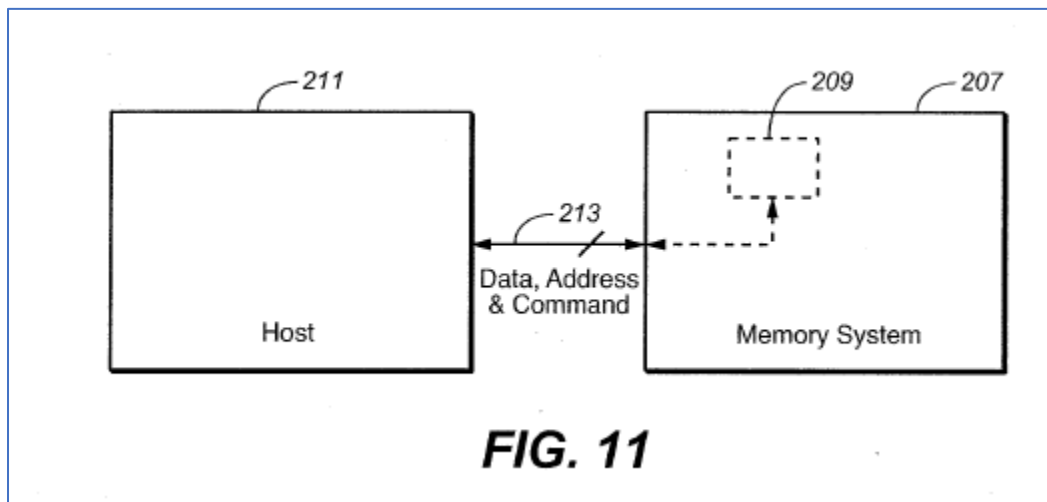
215. The individual blocks are in turn divided into pages of memory cells.

EX1005, Fig. 4, ¶63. Sinclair discloses that to further increase parallelism the pages within two block may be logically linked to form a “metapage.” EX1005, Fig. 5, ¶63. For instance, Figure 5 show each of blocks 137-140 comprises eight pages (P0-P7), where a metapage (purple) is formed by linking pages of four blocks together:



EX1005, Fig. 5, ¶63.

216. Sinclair further discloses that when a host write data files to the memory system they may become “scattered among a larger number of metablocks when file boundaries are not known.” EX1005, ¶86. Sinclair thus discloses that it is desirable for the host to “know the physical characteristics of the memory with which it is operating” and teaches that the host may query the memory for parameters that allow it to adapt to the different physical characteristics of different memories. EX1005, ¶¶99-100. “Figure 11 illustrates a memory device 207 containing these parameter data in non-volatile storage space 209 that is accessible by the host.” EX1005, ¶101.

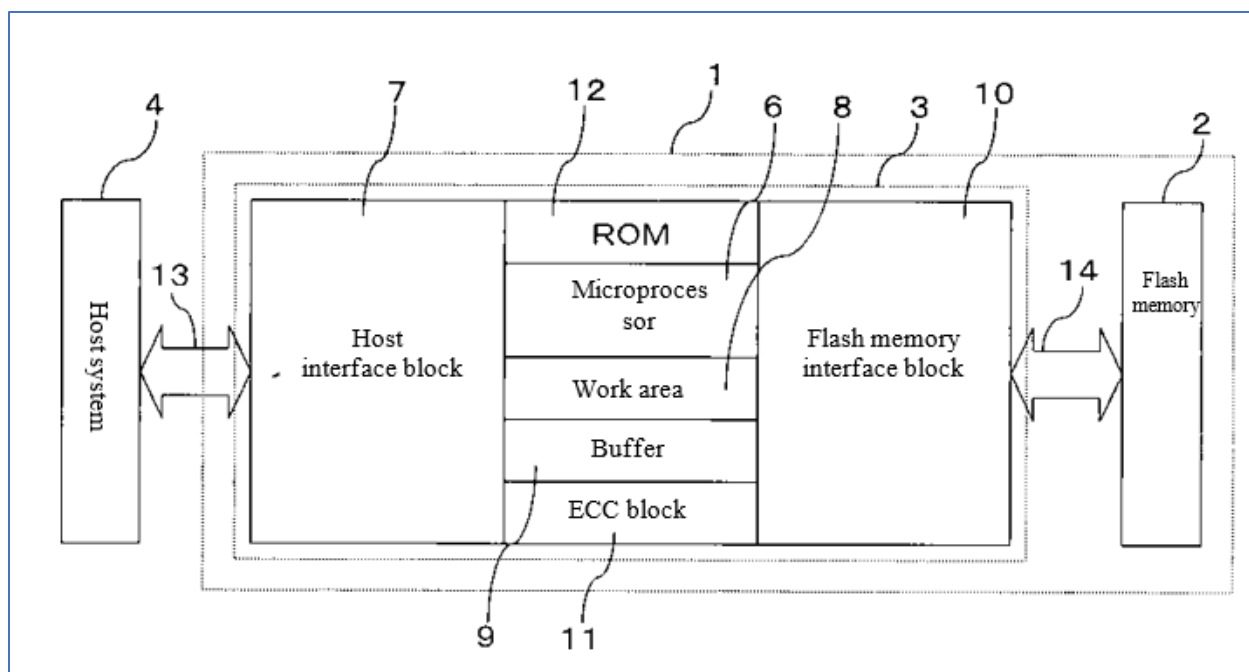


B. Claim Analysis

1. Claim 1

(a) [1pre] A storage device comprising:

217. To the extent the preamble is limiting, Takeo discloses a storage device. For instance, Figure 1 shows a flash memory system 1 [**storage system**] in communication with a host system 4:

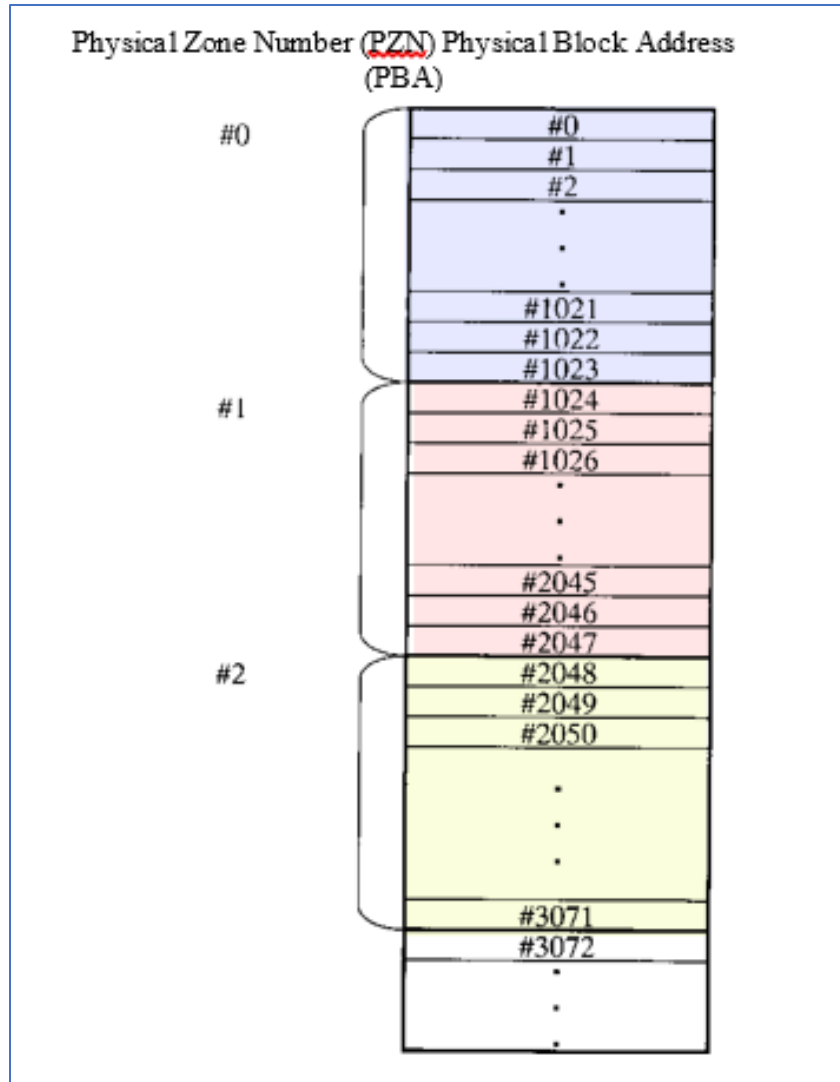


EX1012, Fig. 1, ¶17.

- (b) [1a] flash memory divided into zones, each zone of the zones being a subdivision comprising a set of erase units, and each erase unit of the set of erase units comprising host accessible pages, wherein each zone:

218. Takeo discloses that the flash memory is divided into “zones.”

EX1012, ¶41 (“Figure 3 is an example of the management of the storage area of flash memory 2 following division into multiple zones (physical zones).”). Each zone is assigned with a physical zone number (PZN) and is comprised of a set of blocks [**erase units**]. *Id.* (“The number of physical blocks included in each physical zone is called the Physical Zone Internal Block Number (PZIBN).”). Takeo expressly refers to the blocks as “erase units” (EX1012, ¶3) and Figure 3 illustrates an example where each physicals zone (*e.g.*, PZN #0-2) has “1024 physical blocks”:



EX1012, Fig. 3, ¶41. Each physical zone is assigned to a logical zone, wherein “the number of logical blocks in the logical zone will be set to be less than the number of physical blocks in the physical zone” to account for “bad blocks” or blocks that “became bad over time.” EX1012, ¶42. Each physical block is comprised of “a plurality of pages.” EX1012, ¶¶3, 22-24.

(c) [1a.i] is associated with a logical address range,

219. As described above, Takeo discloses that every zone is assigned with a

PZN and that every writable physical block is assigned with a PZIBN. Furthermore, Takeo discloses that the zones are associated with “logical address information” [**a logical address range**]. For instance, Takeo discloses that “the address space on the host” is divided into “multiple zones (logical zones), and a single physical zone is assigned to a single logical zone.” EX1012, ¶42. Takeo also discloses “logical address information” for each “logical block” that is “contained in each logical zone.” EX1012, ¶43. A POSITA would understand that the “logical address information” for each “logical block” for a “logical zone” forms a “logical address range” associated with the zone.

- (d) [1a.ii] has a zone size for which the storage device is operable to receive a management command from a host that results in the storage device performing an erasure of one or more erase units of the set of erase units,

220. Takeo discloses a zone size. For instance, Takeo teaches that a physical zone may comprise 1024 physical blocks and that the host may set the number of logical blocks in the logical zone to a number less than this, such as 1000 logical blocks. EX1012, ¶¶41-42 (“logical zone containing 1000 logical blocks will be assigned to a physical zone containing 1024 physical blocks.”); *see also id.*, ¶24 (disclosing the size of each block in “bytes”).

221. Takeo does not expressly discuss management operations. However, Son discloses that a storage device may receive a management command from a host that results in the storage device performing an erasure of the erase units in a zone.

See Ground 1, Limitation 1a.ii. As discussed above, Son issues commands for wear leveling and garbage collection [**management command**] for a superblock [**zone**]. *Id.* Son's superblocks are a "grouping of one or more flash blocks," like Takeo's "zones." EX1006, ¶29. Applying Son's teaching to Takeo, the flash memory system of Takeo would receive a management command from the host system 4 that results in the erasure of erase units in a zone.

222. The prior art includes express teachings, suggestions and motivations for combining Son's teachings regarding host-issued management operations. For, instance, a POSITA would understand that a flash memory system should perform management operations such as wear leveling and garbage collection. Because Takeo is silent regarding these management operations, a POSITA would look to other references, like Son, that describe these management operations. As discussed above (see Ground 1, Limitation 1a.ii), Son disclose that a storage device may be improved by having the host issue management commands. For instance, host-issued management commands allow the host to consider device conditions, improve execution speed, and avoid the controller becoming overloaded.

- (e) [1a.iii] is in a full state when there are no host accessible pages available for writing, and

223. Takeo discloses that a zone is in a full state when there are no host accessible pages available for writing [**the zone has zero empty blocks**] and an empty state when all host accessible pages are available for writing [*e.g.*, **all the**

blocks in the zone are empty].

224. For instance, Takeo discloses that the memory controller tracks “internal information” about the zones, including “the number of bad blocks and the number of empty blocks per physical zone as shown in Figure 4.” EX1012, ¶44. Figure 4 shows eight physical zones (PZN#0-PZN#7) and the “number of empty blocks” in each zone. EX1012, Fig. 4. When the number of empty blocks is zero, the zone is in a “full state” and “there are no host accessible pages available for writing.”

PZN	Number of bad blocks	Number of empty blocks
#0	3	10
#1	2	17
#2	4	12
#3	1	11
#4	3	15
#5	2	16
#6	3	14
#7	2	22

EX1012, Fig. 4.

225. Takeo also discloses that the host is aware of “logical zones” and the number of blocks contained therein. EX1012, ¶42 (“the address space on the host

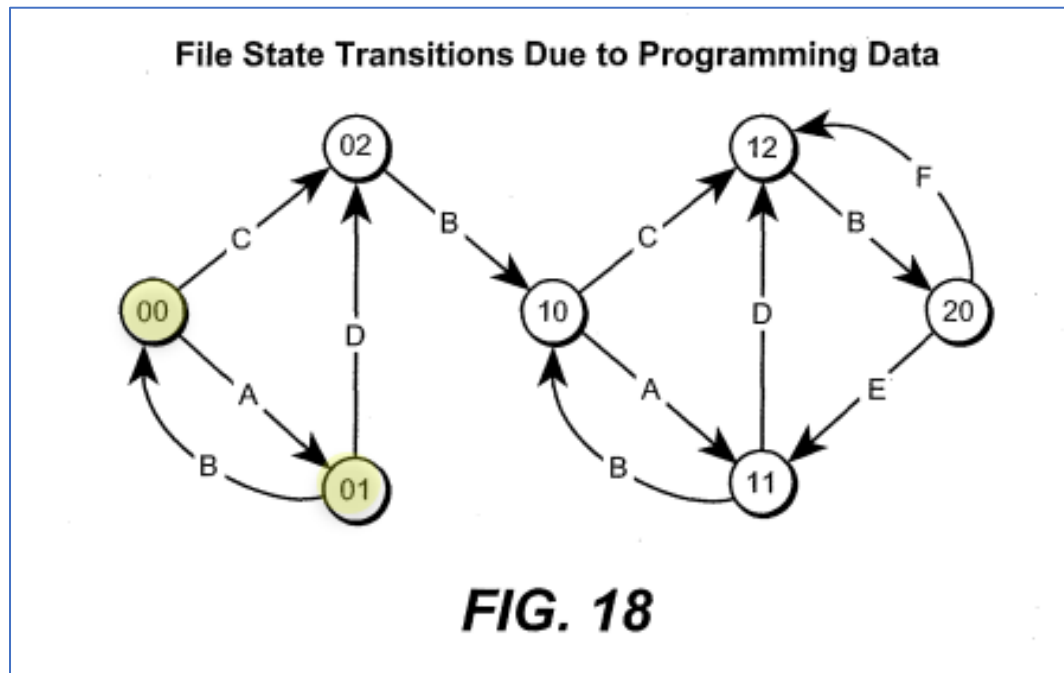
system 4 side is also divided into multiple zones (logical zones)... The number of logical blocks in the logical zone will be set to be less than the number of physical blocks in the physical zone.”). Thus, Takeo also discloses, or fairly suggests, to a POSITA that a zone is in the empty state when the number of empty blocks in the table for a given zone is equal to the number of logical blocks in that zone.

226. To the extent this limitation requires that the memory controller explicitly state whether the zone is “full” or “empty” (rather than indicating, for instance, a full state by returning zero as the number of empty blocks) the limitation is obvious in view of Son and/or Sinclair.

227. For instance, as discussed above, Son discloses maintaining whether superblocks [**zones**] are in a “CLEAN” state [**empty state**] or “SEALED” state [**full state**]. EX1006, ¶40, Fig 3C; *see also* Ground 1, Limitation 1a.iii.

228. Similarly, Sinclair teaches that “logical blocks” (a term Sinclair uses to refer to a grouping of blocks called a “metablock”) may be “in one of three states.” EX1005, ¶¶105-106. In one state the metablock is “storing valid file data without reclaimable capacity,” such as “reclaimable capacity from unprogrammed (erased) pages and/or obsolete (invalid data).” EX1005, ¶106. A POSITA would understand Sinclair’s disclosure of a metablock with no reclaimable capacity to disclose a **full state** because the metablock [**zone**] is filled with valid data and has no host accessible pages for writing. In Figure 18 the arrows labeled “B” indicate that a

“partial block has been filled,” such that the block is transitioned from a partially filled states to a full state after a write operation:



EX1005, Fig. 18 (yellow added), ¶143. For instance, Sinclair states that the transition from “01 to 00” (yellow) reflects the “program block for a file” entering a “full” state. EX1005, Fig. 19.

229. Sinclair also discloses an empty state. For instance, Sinclair discloses that “a logical block is said to be ‘erased’ when no data are allocated to it. An ‘erased’ logical block is one that... is fully available to have addresses of data assigned to it.” EX1005, ¶104. Because in the erased state the metablock [**zone**] is “fully available to have addresses of data assigned to it,” the erased state is an **empty state** where all host accessible pages are available for writing.

230. A POSITA would have been motivated to combine Son and Sinclair’s

teachings to maintain an explicit state value (empty/full) such that the zone is transitioned to an empty state after being erased and a full state after a write operation leaves no more host accessible pages for writing. A POSITA would have understood that maintaining an explicit state value (empty/full) allows the system to quickly determine where data can and cannot be written. Further, an explicit state value helps a system decide which blocks to target for garbage collection and to avoid wasting time attempting to reclaim space from zones that are already empty. Indeed, Sinclair teaches that it is advantageous for a “reclaim operation” to maintain information about the “amount of valid data in each block and the amount of erased space in each block” (EX1005, ¶169) and Takeo already discloses that its system tracks the number of empty blocks per zone (EX1012, Fig. 4). Thus, a POSITA would understand that maintaining a binary state value (full/empty) would have been a straightforward modification that would improve Takeo by simplifying decision-making logic.

(f) [1a.iv] is in an empty state when all host accessible pages are available for writing; and

231. Takeo discloses an empty state for the reasons I discussed above. See Ground 4 Limitation 1a.iii. At minimum, this limitation is obvious in further view of Son and/or Sinclair which I showed both disclose a full and empty state. *Id.*

(g) [1b] logic operable to:

232. Takeo discloses this limitation. Figure 1 illustrate memory controller

3. EX1012, Fig. 1, ¶¶17, 44, 46. The memory controller includes hardware and software, which are “**logic operable to**” perform the claimed functions. EX1012, ¶33 (“Microprocessor 6 controls the overall operation of controller 3 according to the program that may be stored in ROM 12.”).

- (h) [1b.i] return information about the zones in response to a query command from the host, the information about the zones comprising a list of the zones accessible by the host, a zone state, the zone size, logical addresses associated with respective zones, and information representing an extent to which the respective zones include host accessible pages available for writing;

233. Son renders obvious returning the claimed information about the zones in response to a query command from the host for the reasons discussed in connection with Ground 1, Limitation 1b.i. Alternatively, the limitation is obvious in view of Takeo, Son and Sinclair.

234. **return information about the zones in response to a query command from the host:** Takeo discloses that “[h]ost interface block 7 exchanges data, address information, status information, and external commands, etc., to and from host system 4.” EX1012, ¶34. For instance, Takeo discloses a “vendor internal information acquisition command” [**query command from the host**] for returning information about the zones. EX1012, ¶45 (“vendor command to acquire the number of bad blocks and the number of empty blocks in each physical zone is referred to as the internal information acquisition command.”). The host queries the

memory controller by sending it the internal information acquisition command: “In the first example of embodiment, first, host system 4 will write the command code of the internal information acquisition command to the command register in memory controller 3.” EX1012, ¶46. “Memory controller 3 will [then] initiate the preparation of information regarding the number of bad and the number of empty blocks in each physical zone” (EX1012, ¶46), which is then “written to buffer 9” and “transferred to the host system 4 side” (EX1012, ¶¶49-50). The internal information returned to the host includes the information shown in Figure 4 below:

PZN	Number of bad blocks	Number of empty blocks
#0	3	10
#1	2	17
#2	4	12
#3	1	11
#4	3	15
#5	2	16
#6	3	14
#7	2	22

EX1012, Fig. 4¶44. Takeo also discloses that the information above is an “example” and that other information may be returned to the host in response to the query

command. *Id.*, ¶35. As discussed below, it was at least obvious to return each of the claimed information as part of the vendor acquisition command.

235. **the information about the zones comprising a list of the zones accessible by the host:** For instance, Figure 4 of Takeo discloses information returned to the host by the vendor internal information acquisition command [**query command from the host**], which includes a list of physical zone numbers (PZN#0-7) [**list of the zones accessible by the host**]. EX1012, ¶¶44-51, Fig. 4.

PZN	Number of bad blocks	Number of empty blocks
#0	3	10
#1	2	17
#2	4	12
#3	1	11
#4	3	15
#5	2	16
#6	3	14
#7	2	22

236. **a zone state:** Takeo discloses the information returned to the host includes the number of empty blocks for each zone. EX1012, ¶¶45-51, Fig. 4. A POSITA would understand that zones in which the number of empty bocks is zero

are in a “full state” because there are no host accessible pages available for writing. Similarly, a POSITA would understand that where the number of empty blocks equals the number of blocks in the zone, the zone is in the empty state. *See also* Ground 4, Limitation 1a.iii-iv.

237. To the extent this limitation requires that the memory controller explicitly state whether the zone is “full” or “empty” (rather than indicating, for instance, a full state by returning zero as the number of empty blocks) this limitation is at least obvious. For instance, I showed above that both Son and Sinclair disclose maintaining an explicit empty and full state. *See* Ground 4, Limitation 1a.iii, 1a.iv. Because Takeo already discloses that its device returns the number of empty blocks in each zone, it would have been obvious for Takeo to also expressly indicate whether the zone is empty or full, as taught in Son and Sinclair. The combination reflects the use of a known technique (returning state information to the host) to improve similar devices (Takeo’s storage device would be improved by explicitly stating whether the number of blocks reflects an empty or full state) in the same way (Takeo already sends the number of empty blocks such that the improvement would not alter Takeo’s principles of operation).

238. **information representing an extent to which the respective zones include host accessible pages available for writing:** Takeo discloses the information returned to the host includes the number of empty blocks in a zone

[information representing the extent to which the specified zone includes host accessible pages available for writing]. EX1012, ¶¶45-51, Fig. 4.

PZN	Number of bad blocks	Number of empty blocks
#0	3	10
#1	2	17
#2	4	12
#3	1	11
#4	3	15
#5	2	16
#6	3	14
#7	2	22

239. the zone size and logical addresses associated with respective zones:

Takeo does not expressly disclose that the vendor internal information acquisition command returns a zone size and logical addresses associated with respective zones. However, a POSITA would have found it obvious to return this information in view of Sinclair.

240. Sinclair discloses that the host may query the memory for information about the physical characteristics of the memory system using a “vendor specific command that is issued by the host 211 to the memory device 207.” EX1005, ¶¶99-

100. “Figure 11 illustrates a memory device 207 containing these parameter data in non-volatile storage space 209 that is accessible by the host.” EX1005, ¶101. The data may include for each metablock [**a zone**] the “number of pages per metablock” and the “size of a physical page” [**zone size**] and the “the lowest logical address that is mapped to a first page of a physical metablock [**logical addresses associated with respective zones**].” EX1005, ¶99.

241. A POSITA would have been motivated to combine Sinclair’s teaching with Takeo-Son such that this information would be returned as part of Takeo’s vendor internal information acquisition command. For instance, Sinclair discloses that when writing data files they may become “scattered among a larger number of metablocks when file boundaries are not known.” EX1005, ¶86. Sinclair further discloses that the physical characteristics of the memory connected to the host may change such that it is desirable for the host to “know the physical characteristics of the memory with which it is operating” (*e.g.*, the zone size and logical addressing information discussed above). EX1005, ¶¶99-100. Knowing this information allows the host to optimize storage decision. For instance, if the host seeks to write a file that is 1Mb in size, knowing the metablock [**zone**] size and its starting logical address allows the host to ensure that the file is stored within a single metablock, rather than being scattered among multiple metablocks.

242. A POSITA would have had a reasonable expectation of success in

making the combination. Takeo and Sinclair both disclose that the zone information is returned via a vendor acquisition command. EX1012, ¶46 (“vendor command”), EX1005, ¶100 (“vendor specific command”). Thus, the combination represents a straightforward implementation of a host query for zone information using existing command types.

- (i) [1b.ii] write data to the set of erase units of a specified zone using a multi-plane write command such that the data is written concurrently to two or more erase units of the set of erase units; and

243. Takeo in view of Sinclair renders obvious Limitation 1b.ii.

244. Sinclair discloses that erase blocks that span multiple planes may be grouped together into a metablock [**zone**] and that data may be written to the set of erase blocks [**sets of erase units**] of a specified metablock [**specified zone**] using a multi-plane write command such that the data is written concurrently to two or more erase blocks of the set of erase blocks. See Ground 1, Limitation 1b.ii.

245. In my opinion, a POSITA would have been motivated to combine Sinclair’s teaching with Takeo such that Takeo would write data to the blocks of a specified zone concurrently using a multi-plane write command.

246. The prior art includes express teachings, suggestions and motivations to combine the teachings of Sinclair and Takeo to achieve increased parallelism. For instance, Sinclair discloses that multi-plane commands “increase[] parallelism.” EX1005, ¶¶62. Sinclair also explains that in flash memory it is “preferabl[e]” to

read and write data with the “maximum degree of parallelism” across a grouping of blocks [**zone**]. *Id.*, ¶76. *See also* EX1014, § 1 (“Priority order of SSD parallelism levels. Parallelism has been regarded as a key factor in achieving the peak SSD performance.”); EX1008, § (“the way in which parallelism in SSD is exploited can be a key performance-impact factor.”)

247. In my opinion, the combination of Takeo and Sinclair also reflects the application of a known technique, namely multi-plane writes, to Takeo’s known flash memory. EX1014, §2.4 (“[m]ost flash manufacturers provide advance commands, such as ... multiplane... to further improve the performance of SSD.”). The combination improves Takeo’s flash memory by increasing its parallelism and yields predictable results, the ability to write to blocks of the zone in parallel.

248. A POSITA would have had a reasonable expectation of success because the combination represents a straightforward implementation of multi-plane write commands to similar flash devices. As explained in Hu (EX1014, §1.3), SSD manufacturers at the time already included “multi-plane” commands in their SSDs.

- (j) [1b.iii] update the information representing the extent to which the specified zone includes host accessible pages available for writing,

249. In my opinion, Takeo discloses update the information representing the extent to which the specified zone includes host accessible pages available for writing [**updating the number of empty blocks in a zone**]. For instance, Takeo

explains that “the number of empty blocks... will change over time, so it will be necessary to **update the internal information with each change**. For example, **if user data has been written** to an empty block, **the number of empty blocks** in the corresponding physical zone **will be decremented** (decreased by one), and **if the block** to which user data was written in the past **is erased** and has become empty, the **number of empty blocks in the corresponding physical zone will be incremented (increased by one).**” EX1012, ¶58.

- (k) [1c] wherein the logic comprises at least one of hardware logic or instructions stored on non-transitory machine-readable media that, when executed, control operation of hardware circuitry.

250. Takeo discloses a controller that include instructions for controlling operation of the hardware **[logic]**. EX1012, ¶33; *see also* Limitation 1b.

2. Claim 2

- (a) The storage device of claim 1, wherein when the specified zone has no host accessible pages available for writing after writing data to the set of erase units of the specified zone using the multi-plane write command, the storage device is operable to update a state of the specified zone to the full state.

251. In my opinion, Takeo in view of Sinclair renders obvious Claim 2.

252. As shown above, Takeo discloses that when the specified zone has no host accessible pages available for writing after writing data to the set of erase units of the specified zone, the storage device is operable to update a state of the specified zone to the full state. *See* Limitation 1a.iii, 1b.iii. For instance, Takeo teaches that

“if user data has been written to an empty block,” the “number of empty blocks in the corresponding physical zone will be decremented (decreased by one).” EX1012, ¶58. A POSITA would understand that when the number of empty blocks is decremented to zero, the zone is transitioned into a full state because there are no host accessible pages available for writing.

253. Takeo does not, however, expressly disclose a multi-plane write command. However, Sinclair renders obvious that data may be written to a zone using a multi-plane write command for the reasons I discussed above. See Limitation 1b.ii.

3. Claim 4

- (a) The storage device of claim 1, wherein the zone size is a minimum memory size associated with the management command.

254. As I discussed above, in view of Son a POSITA would have been motivated to modify Takeo so that the storage device receives a management command from a host. See Limitation 1.a.ii. The management commands in the combined Takeo-Son system are associated with the memory size of at least a zone (grouping of blocks). For instance, Figure 7 of Son discloses a wear leveling command that selects a worn superblock [**zone**] and target superblock [**zone**] (step 710), copies the contents of the worn superblock to a target superblock, and then erases the worn superblock (step 715). EX1006, Fig. 7, ¶80-83.

4. Claim 5

- (a) The storage device of claim 1, wherein the management command from the host results in the logic issuing erase commands to the set of erase units of the specified zone that are on two or more different planes.

255. In view of Son, a POSITA would have been motivated to modify Takeo so that the storage device receives a management command from a host. See Limitation 1.a.ii. Son's management command from the host results in the logic issuing erase commands to the set of erase units of the specified zone. See Ground 1, Claim 5.

256. While Takeo and Son do not expressly disclose that the erase units of the zone are on "different planes," this limitation is obvious in view of Sinclair which discloses that the groups of blocks [**erase units**] that comprise a metablock [**zone**] may be on different planes and "erased together." EX1005, ¶¶62, 105. As I discussed above, a POSITA would have been motivated to implement the multi-plane commands described in Sinclair to increase parallelism of the SSD. See Ground 4, Limitation 1b.ii.

5. Claim 6

- (a) The storage device of claim 1, wherein the information about the zones is stored on non-volatile memory of the storage device.

257. Takeo does not expressly disclose that the information about the zones is stored in non-volatile memory. However, in my opinion, this limitation is obvious

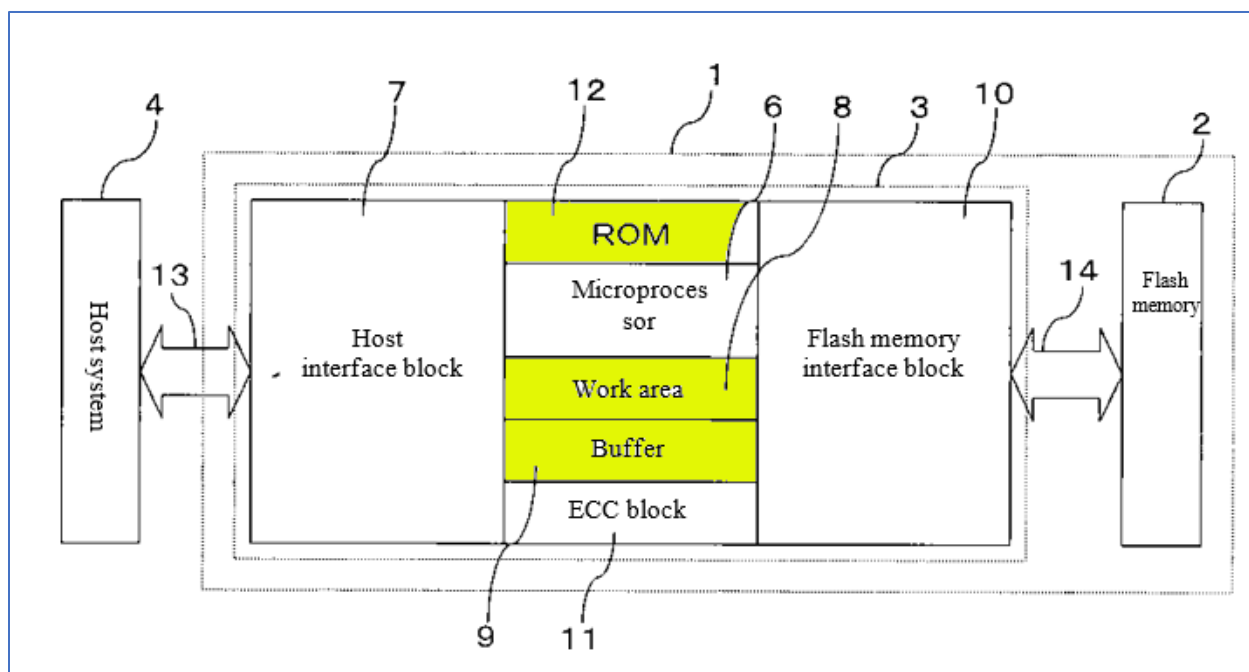
in view of Sinclair, which teaches that the information about the metablocks [**zones**] may be “stored in the memory device itself, in a manner that they may be read by a host.” EX1006, ¶100, 111 (“The memory system... maintains a write pointer”). For instance, Sinclair discloses that the information may be stored in a “**non-volatile storage** space 209 that is accessible by a host.” EX1006, ¶101, Fig. 11.

258. A POSITA would have found it obvious for apply Sinclair’s teaching about storing the information about the zones in non-volatile memory to Takeo’s storage device. For instance, a POSITA would have understood that storing the information in non-volatile (*i.e.*, persistent) memory avoids losing the data when power is lost or reset.

6. Claim 7

- (a) The storage device of claim 1, wherein the information about the zones is stored on a die that includes the logic.

259. Takeo discloses that memory controller 3 includes, in part, a ROM, a work area, and a buffer that are “integrated on a single semiconductor chip” [**die**]:



EX1012, Fig. 1, ¶¶17, 32. Takeo further discloses that the “internal information” about the zones [e.g., **information about the zones**] is temporarily “held in work area 8” along with the software logic [] and outputted to buffer 9 for transmission to the host system. EX1012, ¶¶34, 36, 49, 54. ROM 12 also stores the program logic [logic]. EX1012, ¶33. Thus, the information about the zones is stored on a single semiconductor chip [die] that includes the logic.

7. Claim 8

- (a) The storage device of claim 1, wherein the information about the zones comprises information indicative of a number of erase operations performed on a particular zone of the zones.

260. Son discloses that the information about the superblocks [**zones**] comprises a “wear count,” which “indicates how many times a particular superblock has been erased” [**number of erase operations**]. EX1006, ¶39, Fig. 7.

261. In my opinion, a POSITA would have been motivated to combine the teachings of Son to Takeo's storage device for the reasons I discussed above. See Ground 1, Claim 8. For instance, a POSITA would be motivated to implement wear leveling to extend the lifespan of the flash memory by evenly distributing write and erase operations across the memory. A POSITA would understand that wear leveling involves tracking wear count, as disclosed by Son.

XIII. GROUND 5: CLAIMS 3, 10-13, 16-19, 21-29 ARE OBVIOUS IN VIEW OF TAKEO, SON (EX1006), SINCLAIR (EX1005), AND OLBRICH (EX1010)

A. Claim Analysis

1. Claim 3

- (a) The storage device of claim 1, wherein the logic is operable to write the data to the two or more erase units of the set of erase units of the specified zone using a common page address.

262. As discussed above, Sinclair renders obvious the logic is operable to write the data to the two or more erase units of the set of erase units of the specified zone. See Ground 4, Limitation 1b.ii.

263. To the extent Sinclair does not disclose writing data to the two or more erase units using a "common page address" this limitation is at least obvious in view of Olbrich. Olbrich discloses flash memory that may be divided into superblocks which are comprised of superpages. EX1010, 15:60-67, Fig. 12. Olbrich's superblocks and super pages are similar to Sinclair's metablocks and metapages.

Olbrich further teaches the use of the “same address location” across dies (and planes). EX1010, 15:60-67. For instance, in the example of Figure 12, “SuperPage 30 is spread across four separate dies” and consists of “four flash Pages” that have the “same address location.” A POSITA would understand that having the same address location for pages on different dies discloses **a common page address**.

264. The prior art includes express teachings, suggestions and motivations to combine the teachings of Olbrich to Takeo’s storage device, such that the storage device performs a multi-plane write command using a common page address. For instance, Olbrich disclose that its multi-plane write command that uses a common page address allows for “parallel” operations. EX1010, 2:41-43 (“The use of parallel pipelines allows the controller to effectively mask the write latencies inherent in the use of flash memory.”). Hu similarly discloses that a “multi-plane command” where the “blocks executing a multi-plane” command have the “same ... page addresses” (EX1014, §2.2) “improves the performance of SSDs by handling read, write and erase operations more efficiently” (EX1014, §1.3, “Advanced commands”) and that “parallelism has been regarded as a key to achieving the peak performance” (EX1014, §1.4, “SSD parallelism”).

265. Moreover, the combination is obvious because it reflects the application of a known technique (multi-plane commands with a common page address) to a known device (Takeo flash memory system) ready for improvement (improvement

by allowing parallel writes using a common page address) to yield predictable results (a device with a multi-plane write command).

266. A POSITA would also have had a reasonable expectation of success in combining Olbrich because it represents a straightforward implementation of multi-plane write commands to similar flash devices. Indeed, as Hu explains, SSD manufacturers at the time provided multi-plane commands with a common page address with their SSDs. EX1014, §1.3.

2. Claim 10

(a) [10pre] A storage device comprising:

267. This limitation is disclosed for the reasons I discussed above. See Ground 4, Limitation 1[pre].

(b) [10a] flash memory divided into zones, each zone of the zones being a subdivision comprising a set of erase units, and each erase unit of the set of erase units comprising host accessible pages, wherein each zone:

268. This limitation is disclosed for the reasons I discussed above. See Ground 4, Limitation 1[a].

(c) [10a.i] is associated with a logical address range,

269. This limitation is disclosed for the reasons I discussed above. See Ground 4, Limitation 1[a.i].

(d) [10a.ii] has a zone size associated with a management command from a host,

270. This limitation is disclosed for the reasons I discussed above. See

Ground 4, Limitation 1[a.ii].

- (e) [10a.iii] transitions to a full state in response to no longer having host accessible pages available for writing, and

271. This limitation is disclosed for the reasons I discussed above. *See*

Ground 4, Limitation 1a.iii.

- (f) [10a.iv] in response to the storage device receiving the management command associated therewith, transitions to an empty state such that all host accessible pages are available for writing; and

272. This limitation is disclosed for the reasons I discussed above. *See*

Ground 4, Limitation 1a.iv.

- (g) 10b] logic operable to:

273. This limitation is disclosed for the reasons I discussed above. *See*

Ground 4, Limitation 1b.

- (h) [10b.i] return information about zones stored by the storage device in response to a query command from the host, the information about the zones comprising:

274. This limitation is disclosed for the reasons I discussed above. *See*

Ground 4, Limitation 1b.i.

- (i) [10b.ii] a list of the zones accessible by the host,

275. This limitation is disclosed for the reasons I discussed above. *See*

Ground 4, Limitation 1b.i.

- (j) [10b.iii] a respective logical address associated with a zone of the zones stored,

276. This limitation is disclosed for the reasons I discussed above. *See* Ground 4, Limitation 1b.i.

(k) [10b.iv] the zone size,

277. This limitation is disclosed for the reasons I discussed above. *See* Ground 4, Limitation 1b.i.

(l) [10b.v] information indicating, for respective zones in which there are no host accessible pages available for writing the respective zone, that the respective zones are in the full state, and

278. This limitation is disclosed for the reasons I discussed above. *See* Ground 4, Limitation 1b.i.

(m) [10b.vi] information representing, for respective zones, an extent to which the respective zones include host accessible pages available for writing;

279. This limitation is disclosed for the reasons I discussed above. *See* Ground 4, Limitation 1b.i.

(n) [10b.vii] write data to the set of erase units of a specified zone using a common page address;

280. This limitation is disclosed for the reasons I discussed above. *See* Ground 4, Claim 3.

(o) [10b.viii] update the information representing the extent to which the specified zone includes host accessible pages available for writing; and

281. This limitation is disclosed for the reasons I discussed above. *See* Ground 4, Limitation 1[b.iii].

- (p) [10b.ix] when the specified zone has no host accessible pages available for writing the zone after writing data to the set of erase units of the specified zone, update the information indicating that the specified zone is in the full state,

282. This limitation is disclosed for the reasons I discussed above. See Ground 4, Claim 2.

- (q) [10c] wherein the logic comprises at least one of hardware logic or instructions stored on non-transitory machine-readable media that, when executed, control operation of hardware circuitry.

283. This limitation is disclosed for the reasons I discussed above. See Ground 4, Limitation 1[c].

3. Claim 11

- (a) The storage device of claim 10, wherein the logic is operable to write the data to the set of erase units of the specified zone using a multi-plane write command such that the data is concurrently written to two or more erase units of the set of erase units.

284. This limitation is disclosed for the reasons I discussed above. See Ground 4, Limitation 1b.ii.

4. Claim 12

- (a) The storage device of claim 10, wherein the management command from the host results in the logic issuing at least one erase command to erase one or more erase units of the set of erase units of the specified zone.

285. In my opinion, this limitation is disclosed for the reasons I discussed above. See Ground 4, Limitation 1a.ii. For instance, as I showed above, the

combination of Takeo and Son renders obvious a management command from the host [*e.g.*, **wear leveling or garbage collection command**]. See Ground 4, Limitation 1a.ii. Son expressly discloses that these management commands results in the memory controller logic issuing an erase command for the specified superblock. *Id.*; EX1006, ¶¶68-70, ¶¶78-83, Figs. 6-7.

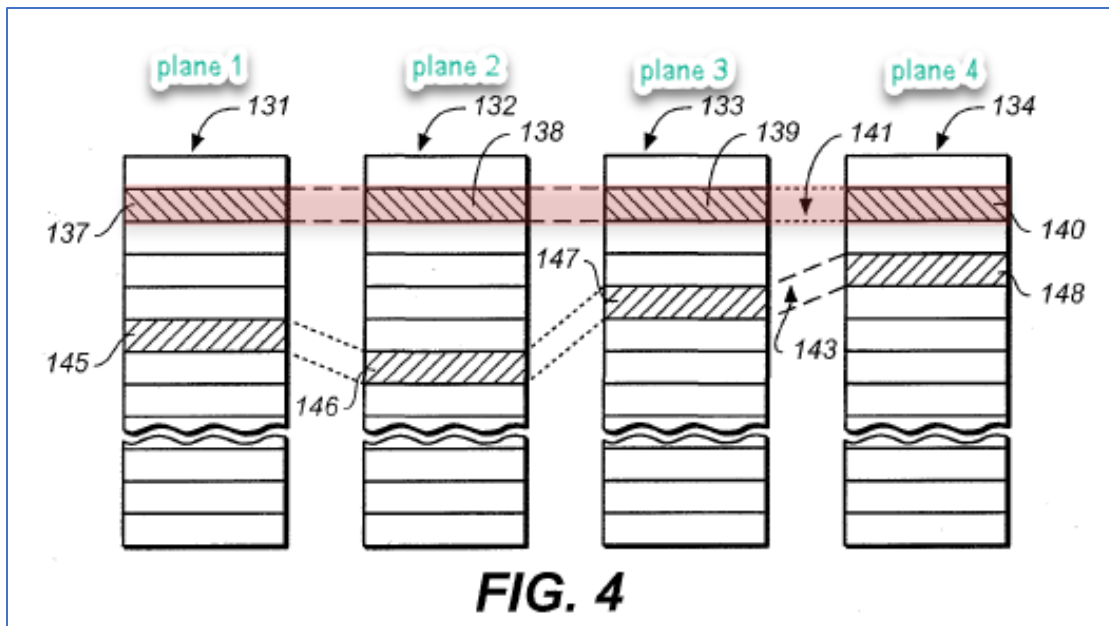
286. As I discussed above, the combination of Takeo and Son renders obvious a management command from the host [*e.g.*, **wear leveling or garbage collection command**] that results in the logic issuing an erase command to erase one or more erase units of the specified zone [*e.g.*, **erase blocks of the specified superblock**]. See Ground 4, Limitation 1a.ii.

5. Claim 13

- (a) The storage device of claim 10, wherein the management command from the host results in the logic issuing an erase command associated with erase units of the specified zone that are on two or more different planes.

287. As I discussed above, in the combined Takeo-Son system the management command from the host results in the logic issuing an erase command associated with erase units of the specified zone. See Ground 4, Limitation 1a.ii.

288. Sinclair further discloses that the erase units of the specified metablock [**specified zone**] are on two or more different planes and erased together:



EX1005, Fig. 4, ¶62 (“all of the cells within a metablock are typically erased together.”).

289. As discussed above, a POSITA would have been motivated to apply Sinclair’s teachings regarding a multi-plane architecture to the Takeo-Son system to improve parallelism. See Ground 4, Limitation 1b.ii. A POSITA would have also had a reasonable expectation of success because multi-plane commands were well-known in the art by this time. EX1014, §1.3 (“SSD manufacturers have provided advanced commands, such as... multi-plane...”), §2.2 (“Multi-plane command activates multiple read, program or erase operations in all planes of the same die”), Fig. 4.

6. Claim 14

- (a) The storage device of claim 13, wherein, as a result of issuing the erase command, the storage device is operable

to detect a bad erase unit and mark the bad erase unit as unusable for storage of new write data.

290. In my opinion, this limitation is obvious in view of Sinclair. Sinclair teaches that if a bad block [**bad erase unit**] is detected in a metablock a different block is responsively substituted:

One or more of these extra blocks may be provided as redundant blocks for ***substitution for other blocks that may become defective during the lifetime of the memory***. The logical grouping of blocks contained within individual metablocks may usually be changed for various reasons, including the ***substitution of a redundant block for a defective block originally assigned to the metablock***.

EX1005, ¶77. A POSITA would thus understand Sinclair to teach detecting a bad erase unit and marking it as unusable for storage of new write data. A POSITA would also understand that detection of a defective block “during the lifetime of the memory” discloses, or at least suggests, that the detection may be “***as a result of issuing the erase command.***” Indeed, detecting bad blocks during erasure was well-known by the time of the ’614 patent. EX1016, ¶¶241-243 (“disclosing a process for detection and handling of bad pages and bad blocks” where “***[a]fter the block is erased, the status of each page in that block is checked to determine if it is bad.***”).

291. A POSITA would have been motivated to apply Sinclair’s teachings to Takeo in this manner and would have had a reasonable expectation of success. Takeo already recognizes the importance of detecting blocks and that the number of bad blocks “will change over time.” EX1012, ¶¶30 (“The block status is information

that indicates whether the physical block is a bad block”), 58 (“if a new bad block has occurred...”). And a POSITA would have understood that erasure performance can help identify bad blocks, such that integrating bad block detection with erasure improves efficiency and accuracy. *See, e.g.*, EX1017, 6:21-7:7. For instance, where a “block needs more than predetermined number of erase cycles applied to fully erase the block,” it is an indication that the block has failed and should be marked bad. *Id.*

7. Claim 16

- (a) The storage device of claim 10, further comprising a host interface that comprises buffers to receive the information about the zones and queue the information about the zones for transmission to the host.

292. Takeo discloses that the host interface comprises a work area 8 and buffer 9 that are used to temporarily store the information about the zones and queue the information about the zones for transmission to the host. EX1012, ¶¶35-36, 47-50.

293. Similarly, Sinclair also discloses that the host interface comprises “random-access memory (RAM) 31 used primarily to buffer data being transferred between the memory and a host.” EX1005, ¶53, Fig. 1 (illustrating host interface & memory controller 8 communicating with host), Fig. 2 (illustrating host interface 35 coupled to RAM 31). A POSITA would have been motivated to combine Sinclair’s teaching with Takeo. Using a buffer to queue information for transmission from

flash memory to a host is a common design. The buffer allows data to be batched together, optimizes performance, and improves error handling.

8. Claim 17

- (a) The storage device of claim 10, wherein the information about the zones is stored on non-volatile memory of the storage device.

294. This limitation is disclosed for the reasons I discussed above. See Ground 4, Claim 6.

9. Claim 18

- (a) The storage device of claim 10, wherein the information about the zones is stored on a die that includes the logic.

295. This limitation is disclosed for the reasons I discussed above. See Ground 4, Claim 7.

10. Claim 19

- (a) The storage device of claim 10, wherein the information about the zones comprises information indicative of a number of erase operations performed on a particular zone of the zones.

296. This limitation is disclosed for the reasons I discussed above. See Ground 4, Claim 8.

11. Claim 21

- (a) [21pre] A storage device comprising:

297. This limitation is disclosed for the reasons I discussed above. See Ground 4, Limitation 1pre.

- (b) [21a] flash memory divided into zones, each zone of the zones being a subdivision comprising a set of erase units, each erase unit of the set of erase units comprising host accessible pages, and the zones are each associated with a respective logical address range; and

298. This limitation is disclosed for the reasons I discussed above. See Ground 4, Limitation 1a and 1a.i.

- (c) [21b] a memory controller in communication with the flash memory, the memory controller comprising:

299. Takeo discloses **memory controller 3** in communication with the **flash memory 2**. EX1012, Fig. 1, ¶¶17, 44, 46.

- (d) [21b.i] storage arranged to retain information about the zones, the information about the zones comprising zone size, a logical address associated with a zone of the zones, and information representing an extent to which the zone of the zones comprises host accessible pages available for writing; and

300. This limitation is disclosed for the reasons I discussed above. See Ground 4, Limitation 1b.i.

- (e) [21b.ii] logic operable to cause the storage device to output the information about the zones in response to a query command from a host, write to the set of erase units of the zone of the zones using a common page address, and update the information representing the extent to which the zone comprises host accessible pages available for writing in response to writing to the set of erase units of the zone.

301. Takeo in combination with Son and/or Sinclair discloses logic operable to cause the storage device to output the information about the zones in response to

a query command from a host. *See* Ground 4, Limitation 1b.i.

302. Takeo also discloses updating the information representing the extent to which the zone comprises host accessible pages available for writing in response to writing to the set of erase units of the zone. *See* Ground 4, Limitation 1b.iii.

303. Olbrich further renders obvious that the write operation to the set of erase units of the zone may be a multi-plane command that uses a common page address. *See* Ground 5, Claim 3.

12. Claim 22

- (a) The storage device of claim 21, wherein the information about the zones comprises a state of the zone.

304. This limitation is disclosed for the reasons I discussed above. *See* Ground 4, Limitation 1b.i.

13. Claim 23

- (a) The storage device of claim 22, wherein the state of the zone is a full state when the zone has no host accessible pages available for writing the zone.

305. This limitation is disclosed for the reasons I discussed above. *See* Ground 4, Limitation a.iii.

14. Claim 24

- (a) The storage device of claim 23, wherein the state of the zone is updated to the full state in response to the zone having no host accessible pages available for writing the zone after writing to the set of erase units of the zone of the zones using the common page address.

306. This limitation is disclosed for the reasons I discussed above. *See* Ground 4, Limitation a.iii, Ground 5, Claim 3.

15. Claim 25

- (a) The storage device of claim 22, wherein the state of the zone is an empty state when all host accessible pages of the zone are available for writing.

307. This limitation is disclosed for the reasons I discussed above. *See* Ground 4, Limitation a.iv.

16. Claim 26

- (a) The storage device of claim 21, wherein the information about the zones comprises information indicative of an amount of time since data was written to the set of erase units of the zone.

308. Son discloses that the information about the zones comprises information indicative of an amount of time since data was written to the set of erase units of the zone [**last modification time**]. EX1006, ¶39, Fig. 3C. Sinclair also discloses that information indicating the “oldest” block is maintained in a block list. EX1005, ¶¶182, 169, Fig. 31

309. A POSITA would have been motivated to apply the teachings of Son and/or Sinclair to Takeo. For instance, a POSITA would understand that flash cells leak charge over time such that data that sits too long without being rewritten risks bit errors. By tracking the elapsed time since a zone was last written to, aging data can be refreshed to avoid corruption. Moreover, as disclosed in Son (EX1006) and

Yamada (EX1015), it is beneficial to evaluate last written time or frequency of data writes when performing memory maintenance, such as wear leveling or garbage collection. EX1006, ¶¶80-81, EX1015, ¶¶155, 163-165. For instance, data that has not been written for a long time can be written to blocks that have a high write count.

17. Claim 27

- (a) The storage device of claim 21, wherein the information about the zones comprises logical addresses associated with respective ones of a plurality of zones of the zones, and information representing an extent to which each of the plurality of zones comprises host accessible pages available for writing.

310. See Ground 4, Limitation b.i

18. Claim 28

- (a) The storage device of claim 21, wherein the zone is a minimum memory space associated with a management command from the host.

311. See Ground 4, Claim 4.

19. Claim 29

- (a) The storage device of claim 21, wherein the logic is operable to cause the storage device to concurrently erase the set of erase units of the zone.

312. Sinclair discloses that erase blocks of a metablock are “erased together.” EX1005, ¶¶62, 105. Thus, in the Takeo-Sinclair combination, the erase blocks of the zones are erased together.

313. A POSITA would have been motivated to apply Sinclair’s teachings to

Takeo in this manner for the reason discussed above. See Ground 4, Limitation 1b.ii. For instance, the prior art includes express teachings, suggestions and motivations to apply Sinclair's teachings regarding multi-plane architectures and commands to Takeo's storage system to achieve increased parallelism. *Id.*

314. The combination of Takeo and Sinclair also reflects the application of a known technique, namely parallel erasure, to Takeo's known flash memory. By the time of the '614 patent, multi-plane erase commands were well-known. EX1014, §1 ("Advanced Commands": explaining that "SSD manufacturers" provide "advanced commands," including "multi-plane" read and write commands), §2.4.2 (explaining that multi-plane commands provided by SSD manufacturers cause the storage device to concurrently "read, program or erase" the erase units on different planes); EX1008, §2.2 ("Multi-plane command activates multiple read, program or erase operations in all planes of the same die. It only costs the time of one read, write or erase operation, while executing multiple such operations."). Thus, it would have been obvious to perform a multi-plane erase command that concurrently erases the erase units in a zone. The combination improves Takeo's flash memory by increasing its parallelism and yields predictable results, the ability to erase blocks of the zone in parallel.

315. In my opinion, a POSITA would have had a reasonable expectation of success because the combination represents a straightforward implementation of an

erase command to similar flash devices.

XIV. GROUND 6: CLAIMS 9, 15, 20, AND 30 ARE OBVIOUS IN VIEW OF TAKEO, SON (EC), SINCLAIR (EX1005), OLBRICH (EX1010) AND YAMADA (EX1015)

A. Claim Analysis

1. Claim 9

- (a) The storage device of claim 1, wherein the storage device is operable to: generate metadata indicating an amount of time since the specified zone was written to, and

316. In my opinion, Yamada in view of the general knowledge of a POSITA, renders it obvious for Takeo's flash device [**storage device**] to generate and track the elapsed time since data was last written to a zone [**generate metadata indicating an amount of time since the specified zone was written to**] and indicate to the host that maintenance is needed when a threshold value is exceeded.

317. Yamada discloses a maintenance request unit of a memory module that compares metadata to a "criterion value" and sends "a maintenance request indicating that the nonvolatile memory needs maintenance to the access device" when the criterion value is exceeded. EX1015, ¶¶32-33, 81-82, Figs. 1-2. A maintenance determination unit of the access device then schedules a maintenance request that "includes refresh and/or wear leveling of the nonvolatile memory." EX1015, ¶30.

318. In the preferred embodiment of Yamada the maintenance request is sent when metadata indicating the number of errors occurring in the data exceeds a

threshold. However, a POSITA would understand that the criterion used to trigger a maintenance request may be any metadata that would indicate a need for a maintenance operation. For instance, POSITAs understood that memory cells will leak over time such that data may become corrupted if it is not written for a long time. EX1015, ¶9. Thus, a POSITA would have been motivated to track “the elapsed time from when the data has been written previously” and refresh data “when the elapsed time is greater than or equal to a criterion value.” EX1015, ¶164. Similarly, a POSITA would understand that metadata indicating the elapsed time has exceeded a threshold may also be used to initiate other maintenance operations, such as wear leveling. EX1006, ¶¶47, 69; EX1004, 19:4-15. Thus, it would have been obvious for Takeo’s flash device to generate metadata indicating the elapsed time since data was last written to a zone.

- (b) in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a second management command directed to the specified zone.

319. As I discussed in the prior limitation, in view of Yamada and the general knowledge of a POSITA it would have been obvious for Takeo’s flash device [**storage device**] to track the elapsed time since data was last written to a zone and then indicate that the storage device recommends that the host send a second management command directed to the specified zone when a threshold value is exceeded [**the storage device recommends that the host send a second**

management command directed to the specified zone]. *See also* EX1015, ¶164.

320. A POSITA would have been motivated to compare the metadata to a threshold on the storage device and then recommend that the host issue a management command and had a reasonable expectation of success for the reasons discussed above. *See* Ground 3, Claim 9. For instance, the prior art teaches that this type of shared management “enables data maintenance to be performed in accordance with the status of the nonvolatile memory system, without requiring the data maintenance to be performed together with normal processing of the system” and “enables the system management status, such as the battery level, to be controlled in a manner that the maintenance will be completed in a reliable manner.” EX1015, ¶¶10, 36. A POSITA would have also been motivated to implement this type of shared memory management because it avoid the storage device having to expose its performance characteristics to the host.

2. Claim 15

321. Claim 15 is similar to Claim 9. The difference between these claims is that in Claim 15 the metadata at issue is “metadata indicating a count of erasures of the erase units,” while in Claim 9 the metadata is “metadata indicating an amount of time since the specified zone was written to.” Claim 15 is the obvious for substantially the same reasons discussed in connection with Claim 9.

322. As discussed in connection with Claim 9, Yamada discloses

transmitting a maintenance request to the host when metadata indicating the number of errors occurring in the data exceeds a threshold. EX1015, ¶¶32-33, 81-82, Figs. 1-2. However, a POSITA would have understood that the criterion used to determine maintenance may be any metadata that would indicate a need for a maintenance operation. For instance, a POSITA would have understood the benefit of implementing wear leveling in a flash memory device to distribute the wear across zones evenly and prevent premature failure of high-usage areas. A POSITA would understand that wear leveling involved maintaining a wear count for a zone **[metadata indicating a count of erasures of the erase units]** and indicating the need for wear leveling when the wear count exceeds a threshold value. EX1006, Fig. 7, ¶¶64, 79; EX1004, 11:63-67. *See also id.*, 13:14-24, claim 1. And for the reasons discussed in connection with Claim 9, a POSITA would have been motivated to implement shared management in which the memory controller performs the comparison and the host sends the management command. *See* Ground 6, Claim 9. Thus, a POSITA would have found it obvious to for Takeo's flash device **[storage device]** to track the wear count of a zone **[metadata indicating a count of erasures of the erase units]** and then indicate that the storage device recommends that the host send a second management command directed to the specified zone when a threshold value is exceeded **[recommends that the host send a second management command directed to the specified zone]**.

3. Claim 20

- (a) [20a] The storage device of claim 10, wherein the storage device is operable to: generate metadata indicating an amount of time since the specified zone was written to, and

See Ground 6, Claim 9[a].

- (b) [20b] in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a second management command directed to the specified zone.

See Ground 6, Claim 9[b].

4. Claim 30

- (a) [30a] The storage device of claim 21, wherein the storage device is operable to: generate metadata indicating an amount of time since the zone was written to, and

See Ground 6, Claim 9[a].

- (b) [30b] in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a management command directed to the zone.

See Ground 6, Claim 9[b].

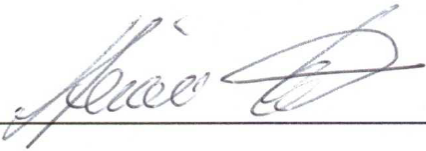
XV. CONCLUSION

323. I understand that this Declaration will be filed as evidence in a contested case before the PTAB. I acknowledge that I may be subject to cross-examination in the case and that cross-examination will take place in the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

324. I declare that all statements made in this Declaration of my own knowledge are true to the best of my knowledge and that all statements made on information and belief are believed to be true. These statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: July 11, 2025

By:

A handwritten signature in blue ink, appearing to read "Alice D.", is written over a horizontal line.

CLAIM LISTING

Claim 1	
1pre	A storage device comprising:
1a	flash memory divided into zones, each zone of the zones being a subdivision comprising a set of erase units, and each erase unit of the set of erase units comprising host accessible pages, wherein each zone:
1a.i	is associated with a logical address range,
1a.ii	has a zone size for which the storage device is operable to receive a management command from a host that results in the storage device performing an erasure of one or more erase units of the set of erase units,
1a.iii	is in a full state when there are no host accessible pages available for writing, and
1a.iv	is in an empty state when all host accessible pages are available for writing; and
1b	logic operable to:
1b.i	return information about the zones in response to a query command from the host, the information about the zones comprising a list of the zones accessible by the host, a zone state, the zone size, logical addresses associated with respective zones, and information representing an extent to which the respective zones include host accessible pages available for writing;
1b.ii	write data to the set of erase units of a specified zone using a multi-plane write command such that the data is written concurrently to two or more erase units of the set of erase units; and
1b.iii	update the information representing the extent to which the specified zone includes host accessible pages available for writing,
1c	wherein the logic comprises at least one of hardware logic or instructions stored on non-transitory machine-readable media that, when executed, control operation of hardware circuitry.
Claim 2	

2.	The storage device of claim 1, wherein when the specified zone has no host accessible pages available for writing after writing data to the set of erase units of the specified zone using the multi-plane write command, the storage device is operable to update a state of the specified zone to the full state.
Claim 3	
3	The storage device of claim 1, wherein the logic is operable to write the data to the two or more erase units of the set of erase units of the specified zone using a common page address.
Claim 4	
4	The storage device of claim 1, wherein the zone size is a minimum memory size associated with the management command.
Claim 5	
5	The storage device of claim 1, wherein the management command from the host results in the logic issuing erase commands to the set of erase units of the specified zone that are on two or more different planes.
Claim 6	
6	The storage device of claim 1, wherein the information about the zones is stored on non-volatile memory of the storage device.
Claim 7	
7	The storage device of claim 1, wherein the information about the zones is stored on a die that includes the logic.
Claim 8	
8	The storage device of claim 1, wherein the information about the zones comprises information indicative of a number of erase operations performed on a particular zone of the zones.
Claim 9	
9[a]	The storage device of claim 1, wherein the storage device is operable to: generate metadata indicating an amount of time since the specified zone was written to, and

9[b]	in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a second management command directed to the specified zone.
Claim 10	
10[pre]	A storage device comprising:
10a	flash memory divided into zones, each zone of the zones being a subdivision comprising a set of erase units, and each erase unit of the set of erase units comprising host accessible pages, wherein each zone:
10a.i	is associated with a logical address range,
10a.ii	has a zone size associated with a management command from a host,
10a.iii	transitions to a full state in response to no longer having host accessible pages available for writing, and
10a.iv	in response to the storage device receiving the management command associated therewith, transitions to an empty state such that all host accessible pages are available for writing; and
10b	logic operable to:
10b.i	return information about zones stored by the storage device in response to a query command from the host, the information about the zones comprising:
10b.ii	a list of the zones accessible by the host,
10b.iii	a respective logical address associated with a zone of the zones stored,
10b.iv	the zone size,
10bv	information indicating, for respective zones in which there are no host accessible pages available for writing the respective zone, that the respective zones are in the full state, and
10b.vi	information representing, for respective zones, an extent to which the respective zones include host accessible pages available for writing;
10b.vii	write data to the set of erase units of a specified zone using a common page address;
10b.viii	update the information representing the extent to which the specified zone includes host accessible pages available for writing; an

10b.ix	when the specified zone has no host accessible pages available for writing the zone after writing data to the set of erase units of the specified zone, update the information indicating that the specified zone is in the full state,
10c	wherein the logic comprises at least one of hardware logic or instructions stored on non-transitory machine-readable media that, when executed, control operation of hardware circuitry.
Claim 11	
11	The storage device of claim 10, wherein the logic is operable to write the data to the set of erase units of the specified zone using a multi-plane write command such that the data is concurrently written to two or more erase units of the set of erase units.
Claim 12	
12	The storage device of claim 10, wherein the management command from the host results in the logic issuing at least one erase command to erase one or more erase units of the set of erase units of the specified zone.
Claim 13	
13	The storage device of claim 10, wherein the management command from the host results in the logic issuing an erase command associated with erase units of the specified zone that are on two or more different planes.
Claim 14	
	The storage device of claim 13, wherein, as a result of issuing the erase command, the storage device is operable to detect a bad erase unit and mark the bad erase unit as unusable for storage of new write data.
Claim 15	
15[a]	The storage device of claim 13, wherein, the storage device is operable to: generate metadata indicating a count of erasures of the erase units, and
15[b]	in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a second management command directed to the specified zone.
Claim 16	

16	The storage device of claim 10, further comprising a host interface that comprises buffers to receive the information about the zones and queue the information about the zones for transmission to the host.
Claim 17	
17	The storage device of claim 10, wherein the information about the zones is stored on non-volatile memory of the storage device.
Claim 18	
18	The storage device of claim 10, wherein the information about the zones is stored on a die that includes the logic.
Claim 19	
19	The storage device of claim 10, wherein the information about the zones comprises information indicative of a number of erase operations performed on a particular zone of the zones.
Claim 20	
20[a]	The storage device of claim 10, wherein the storage device is operable to: generate metadata indicating an amount of time since the specified zone was written to, and
20[b]	in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a second management command directed to the specified zone.
Claim 21	
21pre	A storage device comprising:
21a	flash memory divided into zones, each zone of the zones being a subdivision comprising a set of erase units, each erase unit of the set of erase units comprising host accessible pages, and the zones are each associated with a respective logical address range; and
21b	a memory controller in communication with the flash memory, the memory controller comprising:
21b.i	storage arranged to retain information about the zones, the information about the zones comprising zone size, a logical address associated with a zone of the zones, and information representing an extent to which the zone of the zones comprises host accessible pages available for writing; and

21b.ii	logic operable to cause the storage device to output the information about the zones in response to a query command from a host, write to the set of erase units of the zone of the zones using a common page address, and update the information representing the extent to which the zone comprises host accessible pages available for writing in response to writing to the set of erase units of the zone.
Claim 22	
22	The storage device of claim 21, wherein the information about the zones comprises a state of the zone.
Claim 23	
23	The storage device of claim 22, wherein the state of the zone is a full state when the zone has no host accessible pages available for writing the zone.
Claim 24	
24	The storage device of claim 23, wherein the state of the zone is updated to the full state in response to the zone having no host accessible pages available for writing the zone after writing to the set of erase units of the zone of the zones using the common page address.
Claim 25	
25	The storage device of claim 22, wherein the state of the zone is an empty state when all host accessible pages of the zone are available for writing.
Claim 26	
26	The storage device of claim 21, wherein the information about the zones comprises information indicative of an amount of time since data was written to the set of erase units of the zone.
Claim 27	
27	The storage device of claim 21, wherein the information about the zones comprises logical addresses associated with respective ones of a plurality of zones of the zones, and information representing an extent to which each of the plurality of zones comprises host accessible pages available for writing.
Claim 28	

28	The storage device of claim 21, wherein the zone is a minimum memory space associated with a management command from the host.
Claim 29	
29	The storage device of claim 21, wherein the logic is operable to cause the storage device to concurrently erase the set of erase units of the zone.
Claim 30	
30[a]	The storage device of claim 21, wherein the storage device is operable to: generate metadata indicating an amount of time since the zone was written to, and
30[b]	in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a management command directed to the zone.

Appendix A

Heiner Litz

Curriculum Vitae

January 2025

Address: Department of Computer Engineering
1156 High St, Santa Cruz, CA 95064
Email: heiner.litz@gmail.com
WWW: <https://people.ucsc.edu/~hlitz/>

Overview

I am an Associate Professor at the University of California Santa Cruz and hold the Kumar Malavalli Endowed Chair of Storage Systems Research. I serve as the Director of the Center for Research in Systems and Storage (CRSS). My research spans computer architecture, storage, and systems to design next-generation microprocessors for scale-out data centers. My current interests revolve around re-designing the processor frontend (Icache, BTB, branch prediction) and the memory system, enabling efficient execution of warehouse-scale computing workloads. I am building full-stack prototypes, including applications, firmware, FPGAs, and ASICs. I have contributed to the Linux kernel. I am currently advising 8 Ph.D. students, and my research is supported by grants totaling \$9.1M from NSF, Samsung, Google, Facebook, Intel, Meta, Nutanix, and ARM.

Work Experience

2024 Visiting Professor at the Massachusetts Institute of Technology (MIT), Cambridge
2024- Kumar Malavalli Endowed Professor for Storage Systems Research, UCSC
2023- Associate Professor at the University of California, Santa Cruz
2023- Director of the Center for Research in Systems and Storage at UCSC, Santa Cruz
2022- Consulting Performance Exploration Engineer, ARM
2018-2023 Assistant Professor at the University of California, Santa Cruz
2018-2020 Consulting Researcher at Google, Mountain View
2017 Visiting Faculty at Google, Mountain View
2015-2016 Lecturer and Postdoctoral Researcher, Stanford University, with Prof. C. Kozyrakis
2012-2015 Postdoctoral Researcher, Stanford University, with Prof. D. Cheriton
2008 Hardware Engineering Intern, SUN Microsystems
2006-2012 Research Assistant, University of Heidelberg and Mannheim University

Education

2015 Postdoc in Electrical Engineering, Stanford University
Advisor: David Cheriton and Christos Kozyrakis
2011 PhD. in Computer Engineering, Mannheim University
Advisor: Professor Ulrich Bruening
2010 Visiting Researcher, University of Valencia
Advisor: Professor Jose Duato
2006 Diplom Informatik (M.Sc), Mannheim University

Honors and Awards

- IEEE MICRO Top Pick 2023 Award
- Best Paper Award, MICRO'22
- Intel Outstanding Researcher Award '21
- NSF CAREER Award
- Google Faculty Award
- Facebook Faculty Award
- IEEE MICRO Top Pick 2019 Award
- Memorable Paper Award, NVMW'17
- Dissertation with honors, Summa Cum Laude

Publications

Refereed Conference Papers

1. Liu, Y., M. Xie, S. Shi, Y. Xu, H. Litz, and C. Qian (2025). Outback: Fast and Communication-efficient Index for Key-Value Store on Disaggregated Memory. In: *Proceedings of the 51st International Conference on Very Large Data Bases (VLDB)*.
2. Zhu, K., Y. Zhao, Y. Gao, P. Braun, T. A. Khan, H. Litz, B. Kasikci, and S. Deng (2025). From Optimal to Practical: Efficient Micro-op Cache Replacement Policies for Data Center Applications. In: *Proceedings of the International Symposium on Computer Architecture (HPCA)*.
3. Chakraborty, J., M. Dorier, P. Carns, R. Ross, C. Maltzahn, and H. Litz (2024). Thallus: An RDMA-based Columnar Data Transport Protocol. In: *Proceedings of the 2nd Workshop on Hot Topics in System Infrastructure (HotInfra)*.
4. Oh, S., M. Xu, T. A. Khan, B. Kasikci, and **Litz, Heiner** (2024). UDP: Utility-Driven Fetch Directed Instruction Prefetching. In: *51st International Symposium on Computer Architecture (ISCA)*. ISCA.
5. Xie, M., C. Qian, and H. Litz (2024). En4S: Enabling SLOs in Serverless Storage Systems. In: *Proceedings of the Symposium on Cloud Computing (SoCC)*.
6. Zhang, Y., N. Sobotka, S. Park, S. Jamilan, T. A. Khan, B. Kasikci, G. A. Pokam, **Litz, Heiner**, and J. Devietti (2024). RPG 2: Robust Profile-Guided Runtime Prefetch Generation. In: *27th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'24)*.
7. Jaliminche, L. N., C. N. Chakrabortii, C. Choi, and **Litz, Heiner** (2023). Enabling Multi-tenancy on SSDs with Accurate IO Interference Modeling. In: *2023 ACM Symposium on Cloud Computing (SoCC'23)*.
8. Liu, Y., S. Shi, M. Xie, **Litz, Heiner**, and C. Qian (2023). Smash: Flexible, Fast, and Resource-efficient Placement and Lookup of Distributed Storage. *ACM on Measurement and Analysis of Computing Systems (SIGMETRICS)*.
9. Ni, Y., P. Mehra, E. Miller, and **Litz, Heiner** (2023). TMC: Near-Optimal Resource Allocation for Tiered-Memory Systems. In: *2023 ACM Symposium on Cloud Computing (SoCC'23)*.
10. Zhang, Y., T. A. Khan, G. Pokam, B. Kasikci, **Litz, Heiner**, and J. Devietti (2023). Online COde Layout OptimizationS via OCOLOS. *IEEE Micro Top Pick*.
11. Jamilan, S., T. A. Khan, G. Ayers, B. Kasikci, and **Heiner Litz** (2022). APT-GET: profile-guided *timely* software prefetching. In: *EuroSys '22: Seventeenth European Conference on Computer Systems (EuroSys'22)*.
12. Khan, T. A., M. Ugur, K. Nathella, D. Sunwoo, **Litz, Heiner**, D. Jiménez, and B. Kasikci (2022). Whisper: Profile-Guided Branch Misprediction Elimination for Data Center Applications. In: *55nd Annual International Symposium on Microarchitecture (MICRO'22)*.
13. **Litz, Heiner**, G. Ayers, and P. Ranganathan (2022). CRISP: Critical Slice Prefetching. In: *Twenty-Seventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'22)*.
14. Song, S., T. A. Khan, S. Mahdizadeh-Shahri, A. Sriraman, N. K. Soundararajan, S. Subramoney, D. A. Jiménez, **Heiner Litz**, and B. Kasikci (2022). Thermometer: profile-guided btb replacement for data center applications. In: *ISCA '22: The 49th Annual International Symposium on Computer Architecture (ISCA'22)*.
15. Yuxuan, Z., T. A. Khan, G. Pokam, B. Kasikci, **Litz, Heiner**, and J. Devietti (2022). OCOLOS: On-line COde Layout OptimizationS. In: *55nd Annual International Symposium on Microarchitecture (MICRO'22)*.
16. Chakrabortii, C. and **Litz, Heiner** (2021). Reducing write amplification in flash by death-time prediction of logical block addresses. In: *14th ACM International Conference on Systems and Storage (SYSTOR)*.

17. Kargar, S., **Litz, Heiner**, and F. Nawab (2021). Predict and Write: Using K-Means Clustering to Extend the Lifetime of NVM Storage. In: *37th IEEE International Conference on Data Engineering (ICDE)*.
18. Khan, T. A., D. Zhang, A. Sriraman, J. Devietti, G. Pokam, **Litz, Heiner**, and B. Kasikci (2021). Ripple: Profile-Guided Instruction Cache Replacement for Data Center Applications. In: *48th International Symposium on Computer Architecture (ISCA'21)*.
19. Ayers, G., **Litz, Heiner**, C. Kozyrakis, and P. Ranganathan (2020). Classifying Memory Access Patterns for Prefetching. In: *Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'20)*.
20. Chakrabortii, C. and **Litz, Heiner** (2020a). Improving the accuracy, adaptability, and interpretability of SSD failure prediction models. In: *11th ACM Symposium on Cloud Computing (SoCC)*.
21. Chakrabortii, C. and **Litz, Heiner** (2020b). Learning I/O Access Patterns to Improve Prefetching in SSDs. In: *European Conference on Machine Learning (ECML-PKDD'20)*.
22. Khan, T. A., A. Sriraman, J. Devietti, G. Pokam, **Litz, Heiner**, and B. Kasikci (2020). I-SPY: Context-Driven Conditional Instruction Prefetching with Coalescing. In: *53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'20)*.
23. Ayers, G., N. P. Nagendra, D. I. August, H. K. Cho, S. Kanev, C. Kozyrakis, T. Krishnamurthy, **Litz, Heiner**, T. Moseley, and P. Ranganathan (2019). AsmDB: understanding and mitigating front-end stalls in warehouse-scale computers. In: *46th International Symposium on Computer Architecture (ISCA'19)*.
24. Ni, Y., J. Zhao, **Litz, Heiner**, D. Bittman, and E. L. Miller (2019). SSP: Eliminating Redundant Writes in Failure-Atomic NVRAMs via Shadow Sub-Paging. In: *52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'19)*.
25. Grossman, S., **Litz, Heiner**, and C. Kozyrakis (2018). Making pull-based graph processing performant. In: *23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*.
26. Hashemi, M., K. Swersky, J. Smith, G. Ayers, **Litz, Heiner**, J. Chang, C. Kozyrakis, and P. Ranganathan (2018). Learning Memory Access Patterns. In: *35th International Conference on Machine Learning (ICML)*.
27. Klimovic, A., **Litz, Heiner**, and C. Kozyrakis (2018b). Selecta: Learning Heterogeneous Cloud Storage Configuration for Data Analytics. In: *USENIX Annual Technical Conference (ATC'18)*.
28. **Litz, Heiner**, A. Klimovic, and C. Kozyrakis (2017). ReFlex: Remote Flash == Local Flash. In: *22nd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'17)*.
29. **Litz, Heiner**, B. Braun, and D. R. Cheriton (2016). EXCITE-VM: Extending the Virtual Memory System to Support Snapshot Isolation Transactions. In: *25th International Conference on Parallel Architectures and Compilation Techniques (PACT'25)*.
30. **Litz, Heiner**, D. Cheriton, A. Firoozshahian, O. Azizi, and J. P. Stevenson (2014). SI-TM: reducing transactional memory abort rates through snapshot isolation. In: *19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'14)*.
31. Fröning, H., M. Nüssle, **Litz, Heiner**, C. Leber, and U. Brünig (2013). On achieving high message rates. In: *13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'13)*.
32. Leber, C., B. Geib, and **Litz, Heiner** (2011). High Frequency Trading Acceleration using FPGAs. In: *21st International Conference on Field Programmable Logic and Applications (FPL'21)*.
33. Fröning, H., M. Nüssle, **Litz, Heiner**, and U. Brünig (2010). A Case for FPGA based Accelerated Communication. In: *9th International Conference on Networking (ICN'9)*.
34. **Litz, Heiner**, M. Thürmer, and U. Brünig (2010). TCcluster: A Cluster Architecture Utilizing the Processor Host Interface as a Network Interconnect. In: *2010 IEEE International Conference on Cluster Computing (CLUSTER)*.

35. Fröning, H., **Litz, Heiner**, and U. Brüning (2009). Efficient Virtualization of High-Performance Network Interfaces. In: *8th International Conference on Networks (ICN'8)*.
36. **Litz, Heiner**, H. Fröning, M. Thürmer, and U. Brüning (2009). An FPGA based Verification Platform for HyperTransport 3.x. In: *19th International Conference on Field Programmable Logic and Applications (FPL'19)*.
37. **Litz, Heiner**, H. Fröning, M. Nüssle, and U. Brüning (2008). VELO: A Novel Communication Engine for Ultra-low Latency Message Transfers. In: *37th International Conference on Parallel Processing (ICPP'08)*.
38. Fröning, H., M. Nüssle, D. Slognat, **Litz, Heiner**, and U. Brüning (2006). The HTX-Board: A Rapid Prototyping Station. In: *3rd annual FPGAWorld Conference*.
39. Ding, Y. and **Litz, Heiner** (2005). Creating Multiplatform User Interfaces by Annotation and Adaption. In: *11th International Conference on Intelligent User Interfaces (IUI'11)*.
40. Ding, Y., **Litz, Heiner**, and D. Pfisterer (2004). A graphical single-authoring framework for building multi-platform user interfaces. In: *9th international conference on Intelligent user interface (IUI'9)*.

Journal Papers

1. **Litz, Heiner**, J. Gonzalez, A. Klimovic, and C. Kozyrakis (2021). RAIL: Predictable, Low Tail Latency for NVMe Flash. In: *ACM Transactions on Storage (TOS)*.
2. Nagendra, N. P., G. Ayers, D. I. August, H. K. Cho, S. Kanev, C. Kozyrakis, T. Krishnamurthy, **Litz, Heiner**, T. Moseley, and P. Ranganathan (2020). AsmDB: Understanding and Mitigating Front-End Stalls in Warehouse-Scale Computers. *IEEE Micro*.
3. **Litz, Heiner**, R. J. Dias, and D. Cheriton (2015). Efficient Correction of Anomalies in Snapshot Isolation Transactions. *ACM Transactions on Architecture and Code Optimization (TACO)*.

Refereed Workshop Papers

1. Klimovic, A., **Litz, Heiner**, and C. Kozyrakis (2018a). Learning Heterogeneous Cloud Storage Configuration for Data Analytics. In: *SysML Conference*.
2. **Litz, Heiner**, A. Klimovic, and C. Kozyrakis (2018). ReFlex: Remote Flash == Local Flash. In: *9th Annual Non-Volatile Memories Workshop (NVMW)*.
3. Wang, B., **Litz, Heiner**, and D. R. Cheriton (2014). HICAMP bitmap: space-efficient updatable bitmap index for in-memory databases. In: *10th International Workshop on Data Management on New Hardware (DAMON'10)*.
4. Chan, M., **Litz, Heiner**, and D. R. Cheriton (2013). Rethinking network stack design with memory snapshots. In: *14th USENIX Conference on Hot Topics in Operating Systems (HotOs'14)*.
5. **Litz, Heiner**, C. Leber, and B. Geib (Nov. 2011). DSL Programmable Engine for High Frequency Trading Acceleration. In: *4th Workshop on High Performance Computational Finance (WHPCF'11)*.
6. Fröning, H. and **Litz, Heiner** (2010). Efficient hardware support for the partitioned global address space. In: *24th IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*,
7. **Litz, Heiner**, H. Fröning, and U. Brüning (2010). HTAX: A Novel Framework for Flexible and High Performance Networks-on-Chip. In: *4th Workshop on Interconnection Network Architectures: On-Chip, Multi-Chip (INA-OCMC'4)*.
8. Kalisch, B., A. Giese, **Litz, Heiner**, and U. Brüning (2009). HyperTransport 3 Core: A Next Generation Host Interface with Extremely High Bandwidth. In: *1st International Workshop on HyperTransport Research and Applications (WHTRA'09)*.
9. **Litz, Heiner**, H. Fröning, and U. Brüning (2009). A HyperTransport 3 Physical Layer Interface for FPGAs. In: *6th International Symposium on Applied Reconfigurable Computing (ARC'6)*.

10. Nüssle, M., H. Fröning, A. Giese, **Litz, Heiner**, D. Slogsnat, and U. Brüning (2007). A Hyper-transport based low-latency reconfigurable testbed for message-passing developments. In: *2nd Workshop Kommunikation in Clusterrechnern und Clusterverbundsystemen (KiCC 2007)*.

Teaching Experience

Instructor, CSE120: "Computer Architecture"	'23, '22, '21, '20, '19, '18
Instructor, CSE226: "Advanced Parallel Computing"	'21, '20, '19
Instructor, CSE125: "Logic Design with Verilog"	'22, '21, '20, '19
Instructor, CSE220: "Advanced Computer Architecture"	'23, '22

Grants

I have acquired as PI or Co-PI \$9.2 million in external research grants since 2017.

2023	H. Litz "Google gift"	\$90,000
2022	H. Litz "Google gift"	\$90,000
2022	R. Sanfelice, H. Litz, A. Halder "CPS Frontier NSF Large"	\$5,758,472
2022	H. Litz "Intel research contract"	\$150,000
2022	H. Litz "Samsung gift"	\$75,000
2021	H. Litz "Google gift"	\$60,000
2021	H. Litz "Facebook gift"	\$60,000
2020	H. Litz "NSF CAREER award"	\$529,995
2020	D. Long and H. Litz "NSF IUCRC Phase II grant"	\$500,000
2020	H. Litz "Samsung gift"	\$70,000
2020	H. Litz "Samsung research contract"	\$120,000
2020	R. Sanfelice, H. Litz, A. Halder "UCSC center-scale seed funding award"	\$75,000
2018	H. Litz and B. Paten "Accelerating Genomics Workloads with Smart SSDs". WD	\$960,000
2018	H. Litz "FoMR: Improving Microprocessor IPC for Data Center Workloads". NSF/Intel	\$360,000
2018	H. Litz and R. Felice "Optimized Trajectory planning for Automotive Vehicles", NXP	\$50,000
2017	H. Litz "NVMe Device Level support for Quality of Service". Samsung	\$100,000
2017	H. Litz "Data Center Flash Storage Disaggregation". Broadcom	\$100,000

Patents

Benjamin Geib, Heiner Litz, Mondrian Nuessle. "Circuit arrangement for connection interface". DE102011009518B4, EP2668763A2, US20140207881, filed 2011, assigned 2014.

Graduated Students

Ph.D. Students

- 2022 Yuanjian Ni (Meta)
- 2021 Chandranil Chakrabortii (Trinity College)

MS Students

- 2024 Peter Braun (ARM)
- 2022 Karthik Rajesh Kain (Intel)
- 2021 Maxwell Bradley (ScaleUpStream)
- 2020 Sachet Mittal (Amazon AWS)

Current Students

Saba Jamilan (PhD)
 Minghao Xie (PhD), co-advised with Chen Qian
 Lokesh Jaliminche (PhD)
 Surim Oh (PhD)
 Inje Kim (PhD)
 Yinyuan Zhao (PhD)
 Jayjeet Chakraborti (PhD)
 Mingsheng Xue (MS)

Service

Chair

NVMW'24 - Program Chair
 HOTCHIPS'23 - Program Chair

ASPLOS'23 - Sponsorship Co-Chair

Editor

IEEE MICRO - Associate Editor

Technical Program Comittee

HPCA'25
ASPLOS'25
ISCA'24
ASPLOS'24
MICRO'24
HPCA'24
ISCA'23
HPCA'23
ASPLOS'23
IISWC'22
MICRO'22
ISCA'22
HOTCHIPS'22
NSF'22 Panelist
NVMW'22
Eurosys'22
MICRO'21
ISCA'21
MICRO'20
Usenix ATC'21
Systor'20
Usenix ATC'20
SOCC'19
MICRO'19
HotOS'19
ISCA'19
HPCA-IS'19
PPoPP'19
SOCC'18
Micro'17
TACO'16
HPCA'15
ReCoSoC'11-16
HUCAA'13-15

Panelist

NSF'22 Panel DoE Basic Research Needs Panel NSF'19 Panel

Invited Talks

Datacenter Microprocessors

September 2022	Texas A&M University
May 2022	UC San Diego
May 2022	Stanford University
May 2022	UC Santa Barbara
May 2022	UIUC
May 2022	Industry Academia Partnership (IAP) Workshop
April 2022	UT Austin
April 2022	UC Berkeley
March 2022	Google
Jan 2022	Intel
Nov 2021	ARM

Data center Operating Systems

October 2020	Intel
August 2020	IBM
August 2020	Nutanix
August 2020	VMWare
October 2019	Intel
August 2019	Samsung
August 2019	VMWare
March 2019	Facebook
December 2018	IAP Industry Academia Partnership
August 2018	ITU University Copenhagen
June 2018	Heidelberg University
June 2018	SAP, Walldorf
March 2017	UCSC
Dec 2016	Facebook
July 2016	Amazon AWS
July 2016	CNEX
July 2016	Google
July 2016	Cavium
Apr 2016	Ericsson
Feb 2016	Broadcom
Feb 2016	Stanford Platform Lab

Snapshot Isolation based Transactional Memory

Sep 2015	Samsung
July 2015	VMWare
July 2014	Heidelberg University
Jun 2013	HP Labs
Nov 2012	Stanford CIS

In Memory Deduplication

Nov 2013	SAP Labs
Oct 2013	High Performance Transaction Systems (HPTS)

High Frequency Trading with FPGAs

May 2012	Oracle Labs
----------	-------------

HyperTransport Cache Coherent Interconnect and its Applications

Aug 2011 University of Valencia

Aug 2011 University of Albacete

Mar 2011 KLA-Tencor

Mar 2008 SUN Microsystems

Status

US, German Dual-Citizenship