

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.

Case IPR2025-01321

U.S. Patent No. 11,709,772

**DECLARATION OF DR. HEINER LITZ IN SUPPORT OF PETITION
FOR INTER PARTES REVIEW OF U.S. PATENT NO. 11,709,772**

TABLE OF CONTENTS

	<u>Page</u>
I. SUMMARY OF OPINIONS	1
II. QUALIFICATIONS	1
III. MATERIALS CONSIDERED	3
A. Person of Ordinary Skill in the Art	3
B. Claim Construction.....	4
C. Presumption of Validity	4
D. Obviousness.....	4
IV. DETAILED OPINIONS	7
A. Technology Background and State of the Art.....	7
B. Summary of Patent	11
1. The '772 Patent	11
2. Disclosure.....	11
3. The Challenged Claims.....	12
V. LEVEL OF ORDINARY SKILL IN THE ART	15
VI. OVERVIEW OF PRIOR ART REFERENCES.....	15
A. Olbrich (EX1004).....	15
B. Yamada (EX1005).....	17
C. NVMe1.0b (EX1006).....	18
D. Gonzalez (EX1007).....	19
E. Goss (EX1008)	19
F. Son (EX1009).....	20
G. Ban (EX1010).....	22
H. Sprouse (EX1011)	22
I. Cornwell (EX1012)	23
VII. INVALIDITY OF CLAIMS.....	23

A.	GROUND I: CLAIMS 1-6, 8, 11-13, 15-20, 22, 25-30 ARE RENDERED OBVIOUS BY OLBRICH IN LIGHT OF YAMADA, AND NVME STD	23
1.	Motivation to combine Olbrich, Yamada and NVMe1.0b	23
B.	Claim-by-Claim Analysis.....	27
1.	Claim 1[pre]: A method for providing a host with access to an SSD that contains logic and non-volatile memory comprising NAND flash memory, the logic comprising circuitry and/or a processor executing instructions, and the NAND flash memory comprising a plurality of erase units and each erase unit comprising a plurality of pages, the method comprising, at the SSD:	27
(a)	Claim 1[a]: dividing the flash memory into zones; wherein the zones are non-overlapping subdivisions of the flash memory, each zone comprises two or more erase units, the erase units in each zone are on two or more different planes,.....	27
(b)	Claim 1[b]: and each zone is associated with a zone start logical block address and a range of logical block addresses;	28
(c)	Claim 1[c]: in response to a write command from the host to write data to the range of logical block addresses of a specified zone, performing a multi-plane write of said data to erase units on different planes in the specified zone using a common page address for the respective erase units on the different planes;	29
(d)	Claim 1[d]: generating metadata based on accessing and maintenance of erase units in the specified zone; and;	30
(e)	Claim 1[e]: based on the metadata, making a notification available to the host that recommends performing maintenance on the specified zone, wherein the recommendation is associated with the zone start logical block address of the specified zone;.....	31

- (f) Claim 1[f]: wherein the SSD and the host communicate over a host interface, using commands from a Non-Volatile Memory Express (NVMe) standard 32
- 2. Claim 2: A solid state storage drive (SSD) comprising:.....33
 - (a) 2[a]: a host interface configured to communicate with a host using commands from a Non-Volatile Memory Express (NVMe) standard; 33
 - (b) 2[b]: non-volatile memory comprising NAND flash memory, the NAND flash memory comprising a plurality of erase units and each erase unit comprising a plurality of pages;; 33
 - (c) 2[c]: logic coupled to the host interface and to the flash memory, wherein the logic comprises at least one of hardware logic and software logic, and communicates with the host via the host interface using NVMe commands; 33
 - (d) 2[d]: wherein the logic operates to: divide the flash memory into zones; wherein the zones are non-overlapping subdivisions of the flash memory, each zone comprises two or more erase units, the erase units in at least one zone are on two or more different planes, ...34
 - (e) 2[e]: and each zone is associated with a zone start logical block address and a range of logical block addresses; 34
 - (f) 2[f]: in response to a write command from the host to write data within the range of logical block addresses of a specified zone, perform a multi-plane write of said data to erase units on different planes in the specified zone using a common page address for the respective erase units on the different planes; 34
 - (g) 2[g]: generate metadata based on accessing and maintenance of erase units in the specified zone; and;.....34

- (h) 2[h]: based on the metadata, make a notification available to the host that recommends performing maintenance on the specified zone, wherein the recommendation is associated with the zone start logical block address of the specified zone. 34
- 3. Claim 3: The SSD of claim 2 further comprising: additional non-volatile memory, wherein the logic further operates to store the generated metadata in the additional non-volatile memory. 34
- 4. Claim 4: The SSD of claim 2 wherein the logic further operates to compare the metadata with at least one threshold, and wherein making the notification available to the host is dependent on the comparison. 35
- 5. Claim 5: The SSD of claim 2 36
 - (a) wherein the logic further operates to determine, based on the metadata, that maintenance is recommended for the specified zone and to identify the zone start logical block address associated with the specified zone for which maintenance is recommended; 36
 - (b) wherein determining that maintenance is recommended for the specified zone, identifying the zone start logical block address associated with the specified zone, and making the notification available to the host are all unsolicited by the host. 37
- 6. Claim 6: The SSD of claim 2 wherein the logic further operates to receive a command from the host; and, in response to said command, transmit to the host (a) information representing list of host accessible zones including a zone start logical block address for each listed zone, and (b) a host accessible capacity for each listed zone. 39
- 7. Claim 8: The SSD of claim 2 wherein: 41
 - (a) the logic further operates to receive read commands and write commands from the host, and to execute the read commands and the write

	commands by transferring associated data between the host and the NAND flash memory;	41
(b)	and each read command directed to the SSD is accompanied by a logical read address, each logical read address has a first address portion which identifies one of the zones according to a value of the first portion, and a second address portion which identifies a physical memory location in the SSD in a manner that depends on the first address portion.	42
8.	Claim 11: The SSD of claim 2 wherein the flash memory comprises multiple dies of flash memory and the erase units on different planes are also in different dies., ...	43
9.	Claim 12: The SSD of claim 2 wherein the logic further operates to, upon the performance of the multi-plane write, update the metadata to indicate an amount of unerased host accessible space in the specified zone.	43
10.	Claim 13: The SSD of claim 12 wherein the logic further operates to: upon the performance of the multi-plane write, detect that the specified zone contains no erased host accessible space; and, in response to said detection, transition the specified zone to a full state.....	44
11.	Claim 15: The SSD of claim 2 wherein the logic further operates to receive a maintenance command directed to the specified zone and, in response to the maintenance command, erase host accessible space only within the specified zone and transition the specified zone to an empty state.	45
12.	Claim 16: The SSD of claim 2 wherein the metadata is dependent on an erase count for the specified zone; and the logic further operates to compare the metadata with at least one threshold for erase count, wherein making the notification available to the host is dependent on the comparison.	46
13.	Claim 17: The SSD of claim 2 wherein the logic further operates to: in response to a maintenance command from the host directed to the zone start logical block address of	

	the specified zone, perform an erase of erase units in the specified zone, and transition a state of the specified zone to an empty state whereby all host accessible pages of the specified zone are available for writing by the host.	47
14.	Claim 18: The SSD of claim 17 wherein the maintenance command from the host is in response to the notification that recommends performing maintenance on the specified zone.....	49
15.	Claim 19: The SSD of claim 17 wherein the metadata is dependent on erase counts for erase units in the specified zone; and the logic further operates to update the metadata responsive to performance of the erase.	50
16.	Claim 20: The SSD of claim 17 wherein performing the erase of erase units in the specified zone comprises performing the erase on erase units of different planes in the specified zone.....	50
17.	Claim 22: The SSD of claim 2.....	51
	(a) wherein the logic further operates to, in response to a write command from the host to the zone start logical block address of the specified zone:	51
	(b) assign a logical block address within the range of logical block addresses of the specified zone, transfer data from the host to the specified zone at the assigned logical block address,.....	51
	(c) and return the assigned logical block address to the host.....	53
18.	Claim 25: The SSD of claim 2.....	53
	(a) wherein the metadata is dependent on a time period since data was written to the specified zone;	53
	(b) and the logic further operates to compare the metadata with at least one threshold, wherein making the notification available to the host is dependent on the comparison	54
19.	Claim 26: The SSD of claim 25 wherein comparing the metadata with at least one threshold and making the	

	notification available to the host are unsolicited by the host	55
20.	Claim 27: The SSD of claim 2 wherein logic further operates to, in response to a command from the host, transmit to the host information dependent on a time period since data was written to the specified zone.....	55
21.	Claim 28: The SSD of claim 2 wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone and transmit to the host the zone start logical block address and a state of the specified zone.....	56
22.	Claim 29: The SSD of claim 2 wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone and, in response to a write command from the host within the range of logical block addresses of the specified zone, transmit to the host a notification of a write failure.	58
23.	Claim 30: The SSD of claim 2 wherein the metadata is dependent on read errors detected for the specified zone.....	59
VIII.	GROUND II: Claims 1-6 ARE RENDERED OBVIOUS BY GONZALEZ in light of GOSS and NVMe1.0b.....	60
A.	Motivation to Combine	61
B.	Claim-by-Claim Analysis.....	62
1.	Claim 1[pre]: A method for providing a host with access to an SSD that contains logic and non-volatile memory comprising NAND flash memory, the logic comprising circuitry and/or a processor executing instructions, and the NAND flash memory comprising a plurality of erase units and each erase unit comprising a plurality of pages, the method comprising, at the SSD:	62
(a)	Claim 1[a]: dividing the flash memory into zones; wherein the zones are non-overlapping subdivisions of the flash memory, each zone comprises two or more erase units, the erase units in each zone are on two or more different planes,.....	63

- (b) Claim 1[b]: and each zone is associated with a zone start logical block address and a range of logical block addresses; 63
- (c) Claim 1[c]: in response to a write command from the host to write data to the range of logical block addresses of a specified zone, performing a multi-plane write of said data to erase units on different planes in the specified zone using a common page address for the respective erase units on the different planes; 64
- (d) Claim 1[d]: generating metadata based on accessing and maintenance of erase units in the specified zone; and; 65
- (e) Claim 1[e]: based on the metadata, making a notification available to the host that recommends performing maintenance on the specified zone, wherein the recommendation is associated with the zone start logical block address of the specified zone;.....66
- (f) Claim 1[f]: wherein the SSD and the host communicate over a host interface, using commands from a Non-Volatile Memory Express (NVMe) standard.....67
- 2. Claim 2: A solid state storage drive (SSD) comprising:.....68
 - (b) 2[a]: a host interface configured to communicate with a host using commands from a Non-Volatile Memory Express (NVMe) standard; 68
 - (c) 2[b]: non-volatile memory comprising NAND flash memory, the NAND flash memory comprising a plurality of erase units and each erase unit comprising a plurality of pages;; 68
 - (d) 2[c]: logic coupled to the host interface and to the flash memory, wherein the logic comprises at least one of hardware logic and software logic, and communicates with the host via the host interface using NVMe commands; 68

(e)	2[d]: wherein the logic operates to: divide the flash memory into zones; wherein the zones are non-overlapping subdivisions of the flash memory, each zone comprises two or more erase units, the erase units in at least one zone are on two or more different planes, ...	69
(f)	2[e]: and each zone is associated with a zone start logical block address and a range of logical block addresses;	69
(g)	2[f]: in response to a write command from the host to write data within the range of logical block addresses of a specified zone, perform a multi-plane write of said data to erase units on different planes in the specified zone using a common page address for the respective erase units on the different planes;	69
(h)	2[g]: generate metadata based on accessing and maintenance of erase units in the specified zone; and;	69
(i)	2[h]: based on the metadata, make a notification available to the host that recommends performing maintenance on the specified zone, wherein the recommendation is associated with the zone start logical block address of the specified zone.	69
3.	Claim 3: The SSD of claim 2 further comprising: additional non-volatile memory, wherein the logic further operates to store the generated metadata in the additional non-volatile memory.	70
4.	Claim 4: The SSD of claim 2 wherein the logic further operates to compare the metadata with at least one threshold, and wherein making the notification available to the host is dependent on the comparison.	70
5.	Claim 5: The SSD of claim 2	72
(a)	wherein the logic further operates to determine, based on the metadata, that maintenance is recommended for the specified zone and to identify the zone start logical block address	

	associated with the specified zone for which maintenance is recommended;.....	72
	(b) wherein determining that maintenance is recommended for the specified zone, identifying the zone start logical block address associated with the specified zone, and making the notification available to the host are all unsolicited by the host.	72
6.	Claim 6: The SSD of claim 2 wherein the logic further operates to receive a command from the host; and, in response to said command, transmit to the host (a) information representing list of host accessible zones including a zone start logical block address for each listed zone, and (b) a host accessible capacity for each listed zone.	73
IX.	GROUND III – CLAIM 8 IS RENDERED OBVIOUS BY GONZALEZ IN VIEW OF GOSS, NVME1.0B AND BAN	74
A.	Motivation to Combine	74
B.	Claim-by-claim Analysis.....	75
1.	Claim 8: The SSD of claim 2 wherein:.....	75
	(a) the logic further operates to receive read commands and write commands from the host, and to execute the read commands and the write commands by transferring associated data between the host and the NAND flash memory;	75
	(b) and each read command directed to the SSD is accompanied by a logical read address, each logical read address has a first address portion which identifies one of the zones according to a value of the first portion, and a second address portion which identifies a physical memory location in the SSD in a manner that depends on the first address portion.	76
X.	GROUND IV - CLAIM 11 IS RENDERED OBVIOUS BY GONZALEZ IN VIEW OF GOSS, NVME1.0B, AND SPROUSE	78
A.	Motivation to Combine	78
B.	Claim-by-claim Analysis.....	79

1.	Claim 11: The SSD of claim 2 wherein the flash memory comprises multiple dies of flash memory and the erase units on different planes are also in different dies.	79
XI.	GROUND V – CLAIMS 12, 13, 15-20, 22, 25-27 ARE RENDERED OBVIOUS BY GONZALEZ IN VIEW OF GOSS, NVME1.0B, AND SON	79
A.	Motivation to Combine	80
B.	Claim-by-claim Analysis.....	81
1.	Claim 12: The SSD of claim 2 wherein the logic further operates to, upon the performance of the multi-plane write, update the metadata to indicate an amount of unerased host accessible space in the specified zone.	81
2.	Claim 13: The SSD of claim 12 wherein the logic further operates to: upon the performance of the multi-plane write, detect that the specified zone contains no erased host accessible space; and, in response to said detection, transition the specified zone to a full state.....	82
3.	Claim 15: The SSD of claim 2 wherein the logic further operates to receive a maintenance command directed to the specified zone and, in response to the maintenance command, erase host accessible space only within the specified zone and transition the specified zone to an empty state.	83
4.	Claim 16 The SSD of claim 2 wherein the metadata is dependent on an erase count for the specified zone; and the logic further operates to compare the metadata with at least one threshold for erase count, wherein making the notification available to the host is dependent on the comparison.	84
5.	Claim 17: The SSD of claim 2 wherein the logic further operates to: in response to a maintenance command from the host directed to the zone start logical block address of the specified zone, perform an erase of erase units in the specified zone, and transition a state of the specified zone to an empty state whereby all host accessible pages of the specified zone are available for writing by the host.	85

6.	Claim 18: The SSD of claim 17 wherein the maintenance command from the host is in response to the notification that recommends performing maintenance on the specified zone.....	86
7.	Claim 19: The SSD of claim 17 wherein the metadata is dependent on erase counts for erase units in the specified zone; and the logic further operates to update the metadata responsive to performance of the erase.	88
8.	Claim 20: The SSD of claim 17 wherein performing the erase of erase units in the specified zone comprises performing the erase on erase units of different planes in the specified zone.....	89
9.	Claim 22: The SSD of claim 2.....	89
	(a) wherein the logic further operates to, in response to a write command from the host to the zone start logical block address of the specified zone:	89
	(b) assign a logical block address within the range of logical block addresses of the specified zone, transfer data from the host to the specified zone at the assigned logical block address,.....	90
	(c) and return the assigned logical block address to the host.....	90
10.	Claim 25: The SSD of claim 2.....	91
	(a) wherein the metadata is dependent on a time period since data was written to the specified zone;	91
	(b) and the logic further operates to compare the metadata with at least one threshold, wherein making the notification available to the host is dependent on the comparison	91
11.	Claim 26: The SSD of claim 25 wherein comparing the metadata with at least one threshold and making the notification available to the host are unsolicited by the host	92
12.	Claim 27: The SSD of claim 2 wherein logic further operates to, in response to a command from the host,	

	transmit to the host information dependent on a time period since data was written to the specified zone.....	92
XII.	GROUND VI - GONZALEZ IN VIEW OF GOSS, SON, NVME1.0B, AND CORNWELL	93
A.	Motivation to Combine	93
B.	Claim-by-claim Analysis.....	94
1.	Claim 28: The SSD of claim 2 wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone and transmit to the host the zone start logical block address and a state of the specified zone.....	94
2.	Claim 29: The SSD of claim 2 wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone and, in response to a write command from the host within the range of logical block addresses of the specified zone, transmit to the host a notification of a write failure.	96
3.	Claim 30: The SSD of claim 2 wherein the metadata is dependent on read errors detected for the specified zone.....	97

DECLARATION OF DR. HEINER LITZ, Ph.D.

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 Petition on behalf of Samsung Electronics Co., Ltd. (“SEC”) and Samsung Electronics America, Inc. (“SEA”) (collectively, “Samsung” or “Petitioner”).

2. I have prepared this declaration in connection with Samsung’s related petition for inter partes review (“IPR”) of U.S. Patent No. 11,709,772 (the “’772 patent,” EX1001), 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 ’772 patent. Specifically, I have been asked to provide my opinion regarding whether claims 1-6, 8, 11-13, 15-20, 22, and 25-30 of the ’772 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 Grounds I-VI and the reasons discussed herein.

II. QUALIFICATIONS

4. My education, experience, and qualifications are detailed in my résumé, included as Exhibit 1 to this declaration. While I incorporate Exhibit 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× 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. MATERIALS CONSIDERED

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 ("POSA").

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

15. I understand that the hypothetical POSA 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 POSA 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 I described in the intrinsic evidence—i.e., evidence in the claim language itself, the specification, and the prosecution history.

C. Presumption of Validity

18. I understand that a granted patent is presumed to be valid. I understand that the presumption of validity can be overcome if clear and convincing evidence is presented that proves the patent is invalid.

D. Obviousness

19. 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.

20. 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.

21. 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.

22. 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.

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

24. 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.

25. 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.

26. 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. DETAILED OPINIONS

A. Technology Background and State of the Art

27. 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.

28. SSDs store data on NAND flash memory, which is composed of numerous floating-gate transistors or Charge Trap Flash (CTF) memory cells. To achieve high storage density, SSDs group many of these transistors into units of variable sizes. Data can only be read, written, and erased at the granularity of these units. Notably, the read unit size—often 4KB—can differ from the write unit size, which is generally larger (*e.g.*, 64KB or more), and the erase unit size, which may have a size of *e.g.* 256KB. These units may be referred to as “erase units,” “erase blocks,” or “blocks” interchangeably.

29. In light of the need to manage blocks efficiently, there were known methods in the art for organizing groupings of blocks as a single unit—termed, *i.e.*, SuperBlocks, Metablocks, Zones—where that grouping may be written and erased together, and metadata is tracked at that same level. As a further means to improve efficiency with respect to operational latency and performance, representing a solution for a known need for memory systems, there are various methods in the art for constructing the groupings of blocks to comprise blocks from different planes and/or dies in order to improve parallel operations.

30. 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.

31. 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.

32. For example, assume the SSD has a read granularity of 4KB and the application writes a 4KB block to LBA=2. The SSD stores it at PBA=12 and updates the L2P so that LBA=2 maps to PBA=12. Upon a subsequent overwrite of LBA=2, the SSD writes the new block to PBA=13, leaving PBA=12 unchanged. The L2P is updated to map LBA=2 to PBA=13. If the write unit size is larger (e.g., 64KB), a write buffer is used to coalesce multiple 4KB writes (e.g., 16 blocks) into a single 64KB write operation.

33. Since each write consumes a previously unwritten block, the SSD will eventually run out of such blocks. To reclaim space, SSDs support an *erase* operation, which may be directed to an erase block or erase unit. SSDs may also operate at another level of granularity—commonly 256KB—where erases, for example, are conducted for 64 consecutive 4KB blocks. This unit is also referred to as a *subdivision* or *zone*, which are functionally equivalent.

34. To safely erase a zone, it must not contain any *valid* blocks. A valid block is defined as a block that is still referenced by the L2P and thus represents the latest version of a data element. If no such zone is entirely free of valid blocks, the SSD performs *garbage collection*. During this process, the SSD itself rewrites valid blocks to new locations, updates the L2P accordingly, and invalidates the old PBAs. Once all valid blocks have been relocated, the blocks can be safely erased. Garbage collection involves additional reads and writes, and therefore, should be minimized to improve performance and endurance, as NAND flash has a limited number of program/erase (P/E) cycles.

35. To optimize garbage collection, the SSD maintains metadata to track the number of valid blocks per zone. Selecting a block with the fewest valid blocks minimizes the number of costly copy operations. Furthermore, it is beneficial to segregate frequently updated (*hot*) data from rarely updated (*cold*) data by placing them into separate blocks. One implementation approach is to maintain multiple open blocks to which writes can be applied: one for cold LBAs and one for hot LBAs. More than two blocks may be maintained, with write frequency tracked using finer-grained *temperature* metrics.

36. 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 moves cold data from infrequently used blocks, enabling those blocks to be erased and reused, thereby balancing wear across all blocks in the SSD.

37. 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 unit.
- Garbage Collection: Reclaiming space by relocating valid data and erasing blocks.

- Wear-Leveling: Ensuring uniform wear across blocks by migrating cold data.
- Erase Management: Erasing only those blocks that no longer contain valid data.

38. In traditional SSDs, the FTL is implemented entirely within the SSD controller. This architecture presents a simple hardware interface to software, exposing only read and write operations at fixed granularities (e.g., 4KB).

39. An alternative approach is to implement the FTL—fully or partially—on the *host processor*. Host-based FTLs can leverage application-specific knowledge (e.g., data temperature) 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.”

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 exposes 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.

B. Summary of Patent

1. The '772 Patent

42. The '772 patent issued on January 3, 2023, and is titled “Storage System with Multiplane Segments and Cooperative Flash Management.”

2. Disclosure

43. The '772 patent describes a memory system, including a memory controller (in an SSD) and a host, where responsibility for memory management is shared between the controller and host. EX1001, Abstract, 18:26-45.

44. The claims are directed to memory units termed, “zones” which are “non-overlapping subdivisions of the flash memory, ... compris[ing] two or more erase units ...on two or more different planes.” EX1001, Claim 1. The patent also describes the use of interleaving to allow memory to be organized across-dies, where the host has knowledge of this architecture and may issue multi-plane writes. *Id.*, 38:60-64, 50:44-47.

45. Memory management functions are based on metadata, where that metadata may be sent “to the host”—“the memory controller of FIG. 4 provides data to the host based on tracked, subdivision-specific (*e.g.*, EU-specific) information associated with a particular parameter or metric.” *Id.*, 25:2-64, 24:36-39. For example this tracked metadata includes, *e.g.*, an “age field,” “erase count,” and “bad” block identification. *Id.*, 25:26-64. One such memory management function facilitated by the metadata sent to the host is “host-initiated maintenance operations (*e.g.*, garbage collection, erase, maintenance operations, defect management and so forth), [where] the host is able to ensure that adequate free space is available in each memory array and to generally streamline operations.” *Id.*, 53:15-36.

3. The Challenged Claims

46. Below is a table described the Challenged Claims, including claims 1-6, 8, 11-13, 15-20, 22, and 25-30.

Independent Claims	
1[pre]	1. A method for providing a host with access to an SSD that contains logic and non-volatile memory comprising NAND flash memory, the logic comprising circuitry and/or a processor executing instructions, and the NAND flash memory comprising a plurality of erase units and each erase unit comprising a plurality of pages, the method comprising, at the SSD:
1[a]	dividing the flash memory into zones; wherein the zones are non-overlapping subdivisions of the flash memory, each zone comprises two or more erase units, the erase units in each zone are on two or more different planes,
1[b]	and each zone is associated with a zone start logical block address and a range of logical block addresses;
1[c]	in response to a write command from the host to write data to the range of logical block addresses of a specified zone, performing a multi-plane write of said data to erase units on different planes in the specified zone using a common page address for the respective erase units on the different planes
1[d]	generating metadata based on accessing and maintenance of erase units in the specified zone; and
1[e]	based on the metadata, making a notification available to the host that recommends performing maintenance on the specified zone, wherein the recommendation is associated with the zone start logical block address of the specified zone
1[f]	wherein the SSD and the host communicate over a host interface, using commands from a Non-Volatile Memory Express (NVMe) standard
2[pre]	A solid state storage drive (SSD) comprising:
2[a]	a host interface configured to communicate with a host using commands from a Non-Volatile Memory Express (NVMe) standard;
2[b]	non-volatile memory comprising NAND flash memory, the NAND flash memory comprising a plurality of erase units and each erase unit comprising a plurality of pages;
2[c]	logic coupled to the host interface and to the flash memory, wherein the logic comprises at least one of hardware logic and software logic, and communicates with the host via the host interface using NVMe commands;
2[d]	wherein the logic operates to:
2[e]	divide the flash memory into zones; wherein the zones are non-overlapping subdivisions of the flash memory, each zone comprises two or more erase units, the erase units in at least one zone are on two or more different planes,

2[f]	and each zone is associated with a zone start logical block address and a range of logical block addresses;
2[g]	in response to a write command from the host to write data within the range of logical block addresses of a specified zone, perform a multi-plane write of said data to erase units on different planes in the specified zone using a common page address for the respective erase units on the different planes;
2[h]	generate metadata based on accessing and maintenance of erase units in the specified zone; and
2[i]	based on the metadata, make a notification available to the host that recommends performing maintenance on the specified zone, wherein the recommendation is associated with the zone start logical block address of the specified zone.
Dependent Claims	
3	The SSD of claim 2 further comprising: additional non-volatile memory, wherein the logic further operates to store the generated metadata in the additional non-volatile memory.
4	The SSD of claim 2 wherein the logic further operates to compare the metadata with at least one threshold, and wherein making the notification available to the host is dependent on the comparison.
5	The SSD of claim 2 wherein the logic further operates to determine, based on the metadata, that maintenance is recommended for the specified zone and to identify the zone start logical block address associated with the specified zone for which maintenance is recommended; wherein determining that maintenance is recommended for the specified zone, identifying the zone start logical block address associated with the specified zone, and making the notification available to the host are all unsolicited by the host.
6	The SSD of claim 2 wherein the logic further operates to receive a command from the host; and, in response to said command, transmit to the host (a) information representing list of host accessible zones including a zone start logical block address for each listed zone, and (b) a host accessible capacity for each listed zone
8	The SSD of claim 2 wherein: the logic further operates to receive read commands and write commands from the host, and to execute the read commands and the write commands by transferring associated data between the host and the NAND flash memory; and each read command directed to the SSD is accompanied by a logical read address, each logical read address has a first address portion which identifies one of the zones according to a value of the first portion, and a second address portion which identifies a physical memory location in the SSD in a manner that depends on the first address portion.
11	The SSD of claim 2 wherein the flash memory comprises multiple dies of flash memory and the erase units on different planes are also in different dies.

12	The SSD of claim 2 wherein the logic further operates to, upon the performance of the multi-plane write, update the metadata to indicate an amount of unerased host accessible space in the specified zone.
13	The SSD of claim 12 wherein the logic further operates to: upon the performance of the multi-plane write, detect that the specified zone contains no erased host accessible space; and, in response to said detection, transition the specified zone to a full state.
15	The SSD of claim 2 wherein the logic further operates to receive a maintenance command directed to the specified zone and, in response to the maintenance command, erase host accessible space only within the specified zone and transition the specified zone to an empty state.
16	The SSD of claim 2 wherein the metadata is dependent on an erase count for the specified zone; and the logic further operates to compare the metadata with at least one threshold for erase count, wherein making the notification available to the host is dependent on the comparison.
17	The SSD of claim 2 wherein the logic further operates to: in response to a maintenance command from the host directed to the zone start logical block address of the specified zone, perform an erase of erase units in the specified zone, and transition a state of the specified zone to an empty state whereby all host accessible pages of the specified zone are available for writing by the host.
18	The SSD of claim 17 wherein the maintenance command from the host is in response to the notification that recommends performing maintenance on the specified zone
19	The SSD of claim 17 wherein the metadata is dependent on erase counts for erase units in the specified zone; and the logic further operates to update the metadata responsive to performance of the erase.
20	The SSD of claim 17 wherein performing the erase of erase units in the specified zone comprises performing the erase on erase units of different planes in the specified zone
22	The SSD of claim 2 wherein the logic further operates to, in response to a write command from the host to the zone start logical block address of the specified zone: assign a logical block address within the range of logical block addresses of the specified zone, transfer data from the host to the specified zone at the assigned logical block address, and return the assigned logical block address to the host.
25	The SSD of claim 2 wherein the metadata is dependent on a time period since data was written to the specified zone; and the logic further operates to compare the metadata with at least one threshold, wherein making the notification available to the host is dependent on the comparison.
26	The SSD of claim 25 wherein comparing the metadata with at least one threshold and making the notification available to the host are unsolicited by the host

27	The SSD of claim 2 wherein logic further operates to, in response to a command from the host, transmit to the host information dependent on a time period since data was written to the specified zone.
28	The SSD of claim 2 wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone and transmit to the host the zone start logical block address and a state of the specified zone.
29	The SSD of claim 2 wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone and, in response to a write command from the host within the range of logical block addresses of the specified zone, transmit to the host a notification of a write failure.
30	The SSD of claim 2 wherein the metadata is dependent on read errors detected for the specified zone.

V. LEVEL OF ORDINARY SKILL IN THE ART

47. The level of ordinary skill in the art would be a person with 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.

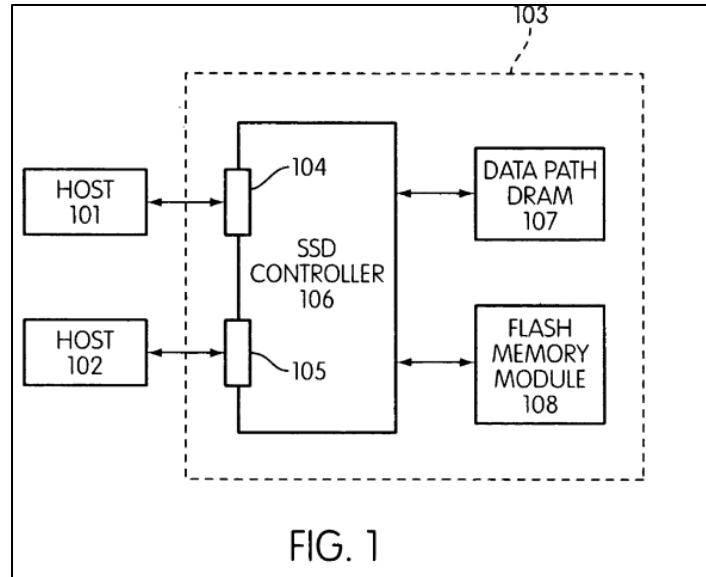
48. Under this formulation, my opinions expressed in this declaration apply.

VI. OVERVIEW OF PRIOR ART REFERENCES

A. Olbrich (EX1004)

49. U.S. Patent 8,386,700 (“Olbrich”) was filed on November 29, 2011, was published on March 22, 2012, and issued on February 26, 2013. I understand that it is prior art under at least AIA 35 U.S.C. § 102.

50. Olbrich discloses a storage system which includes a controller that communicates with a host and a flash memory module:



EX1004, Fig. 1.

51. Olbrich teaches dividing the flash memory into “SuperBlocks,” comprised of erase units on two or more different planes. *Id.*, 16:12-16. Olbrich further teaches that each SuperBlock includes SuperPages, where each SuperPage includes a “starting address,” *i.e.*, “a zone start logical block address” (“ZSLBA”) and “15 consecutive LBAs,” “a range of logical block addresses” (“LBAs”). *Id.*, 72:33-38. “[E]ach SuperPage consists of four flash Pages, each of which is stored on a different Flash Die” (across different planes). *Id.*, 6:23-29. Olbrich utilizes a common page address in which the pages of a SuperPage have the “same address location.” *Id.*, 15:60-67.

52. In light of this design, where memory is organized into SuperBlocks, Olbrich teaches the tracking of metadata at the SuperBlock-level to further improve efficiency and to streamline memory management. Specifically, the system maintains information about the SuperBlocks in a “super block metadata table.” *Id.*, 48:5-7. “SuperBlock Metadata Table 3501 stores three types of metadata for each SuperPage: a SuperPage Time Stamp, Defect Flag, and LBA.” *Id.* Further, the “SuperBlock Metadata Table 3501 also contains four fields that are not specific to particular SuperPages, but contain metadata relevant to the entire SuperBlock: Super

Block Time Stamp 3506, Erase Count Field 3507, Open Flag 3508 and Closed Flag 3509.” *Id.*, 48:30-34. The SuperBlock metadata table is shown below:

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

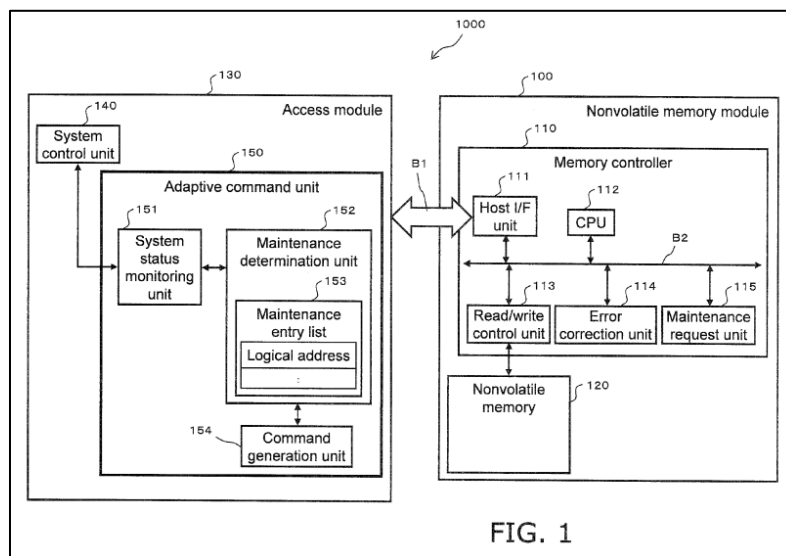
Id., Fig. 35, 48:5-44.

B. Yamada (EX1005)

53. 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 under at least AIA 35 U.S.C. § 102.

54. Yamada teaches that memory management responsibilities are shared between the host and controller. Yamada describes particular implementations of such shared responsibility, *e.g.*, where maintenance operations are recommended by the controller in notifications to the host, and maintenance is subsequently commanded by the host.

55. Specifically, Yamada describes a controller including a “maintenance request unit,” and a host (termed an “access module”) including a “maintenance determination unit.” EX1005, [0012], [0017], [0031].



Id., Fig. 1.

56. Yamada describes that, based on an analysis of the metadata of blocks or groups of blocks to determine if certain thresholds have been met, if so, the controller transmits the metadata and address information (a “single maintenance command may be generated for a plurality of addresses of a plurality of blocks in the nonvolatile memory system”) to the host in a “maintenance request” from the “maintenance request unit.” *Id.*, [0013]-[0014], [0032], [0080]-[0081], [0109]-[0111], [0117]. In response, the host may then issue a “maintenance command” from the “maintenance determination unit,” commanding the controller to copy data from the block or blocks that need maintenance to new block(s). *Id.*

C. NVMe1.0b (EX1006)

57. The NVM Express Base Specification Revision 1.0b (NVMe1.0b), on its face, establishes that it was published on July 12, 2011. I understand that it is prior art under at least

AIA 35 U.S.C. § 102.

58. The NVMe1.0b (along with the iterative specifications issued afterwards) represent widely adopted standards promulgated and used by the industry to promote interoperability and compatibility across systems. Specifically, it is “a register level interface that allows host software to communicate with a nonvolatile memory subsystem” which “is optimized for Enterprise and Client solid state drives, typically attached to the PCI Express interface.” EX1006, § 1.1.

59. The NVMe1.0b includes a unit of memory termed a “namespace,” which “is collection of logical blocks that range from 0 to the capacity of the namespace.” The aggregation of blocks into a namespace enables efficiencies in memory management. For example, metadata collected by the controller and provided to the host may be organized and maintained at the namespace-level. *E.g., id.*, § 5.10.1.2.

D. Gonzalez (EX1007)

60. U.S. Pub. Patent 2010/0049908 (“Gonzalez”) was filed on October 30, 2009, and was published on February 25, 2010. I understand that it is prior art under at least AIA 35 U.S.C. § 102.

61. Gonzalez teaches that “blocks may be linked together as metablocks; that is, each logically addressable metablock is defined to include one physical block from each plane,” in order to increase parallelism and performance. EX1007, [0011]-[0018]. Each metablock includes a “common logical address” and is comprised of a “range of consecutive [logical] addresses.” *Id.*, [0010]-[0018], [0044], [0057].

E. Goss (EX1008)

62. U.S. Pat. Pub. No. 2011/0302477 (“Goss”) was filed on June 3, 2010, and was published December 8, 2011. I understand that it is prior art under at least AIA 35 U.S.C. § 102.

63. Goss teaches a memory system where the controller maintains metadata regarding the status of data, including, *e.g.*, wear indication information relating to accessing and erase count, where this information is provided to the host in order to initiate maintenance activities such as data hardening. EX1008, [0021], [0024], [0048], [0085].

64. For example, Goss teaches that the controller conducts “[s]elf-monitoring routines [and] can provide real-time notifications to a host with regard to the status of the array and/or the data, allowing the user to take affirmative steps to alter the operation of the device to extend its useful life.” EX1008, [0024]. For further example, Goss teaches controller “ROUTINE DATA MAINTENANCE”—“steps carried out by the controller 102 to assess and improve the reliability of the array 104 in accordance with various embodiments ... which may [] be implemented by the device performing monitoring and providing a notification to the host that, without data hardening, the data may be below retention specifications.” *Id.*, [0085]. These routines resulting in notification to the host “can provide a number of benefits to the operational reliability of a memory.” *Id.*, [0091].

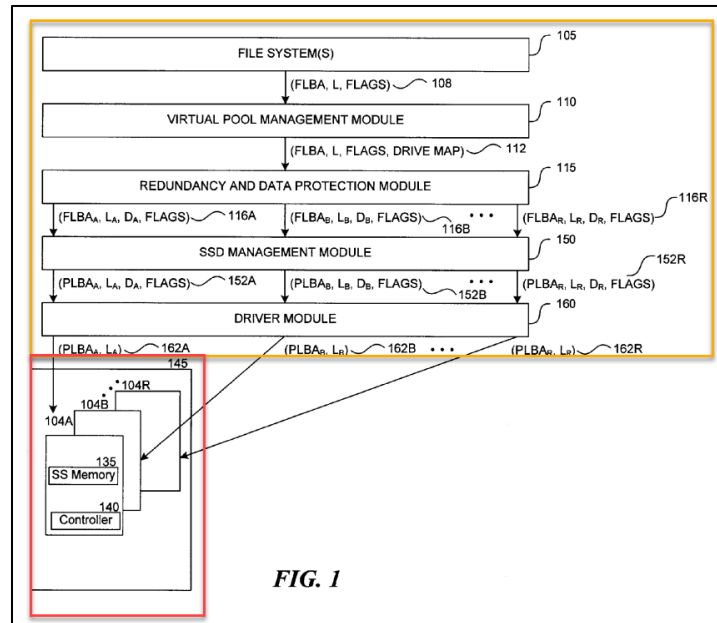
F. Son (EX1009)

65. U.S. Patent Application No. 2012/0096217 (“Son”) was filed Oct. 14, 2011, and was published April 19, 2012. I understand that it is prior art under at least AIA 35 U.S.C. § 102.

66. Son discusses the “shortcomings” of memory systems which rely primarily on the controller and teaches the benefits of sharing memory management between the host and controller where the host has access to “both file system information and information related to the underlying physical solid-state storage media.” EX1009, [0014]. By “moving solid-state management functions above the firmware level,” memory systems with shared memory

management responsibilities “may achieve various advantages over conventional systems, such as enhanced I/O performance, simplified SSD firmware, and extended SSD lifespan.” *Id.*

67. The memory system Son teaches relies on smart host system, which is comprised of “one or more file systems 105, ... an SSD management module (‘the SSD module’) 150, a driver module 160.” *Id.*, [0016]. The host is the yellow box in the annotated figure below.



68. Son’s memory architecture is comprised of SuperBlocks which are “a grouping of ... blocks.” *Id.*, [0029]. Son teaches that memory management is done at the SuperBlock level where commands are directed at SuperBlocks using a “SuperBlock address.” *Id.*, [0029], [0038]. Son describes tracking SuperBlock metadata. This metadata is stored in a “SuperBlock Table” and includes “free page count, [and] valid page count” of each individual SuperBlock. *Id.*, [0080].

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

Id., Fig. 3C, [0080], [0029], [0038].

G. Ban (EX1010)

69. U.S. Pat. No. 5,404,485 (“Ban”) was filed on March 8, 1993, and issued on April 4, 1995. I understand that it is prior art under at least AIA 35 U.S.C. § 102.

70. Ban teaches “a method (*i.e.*, software, firmware or hardware) to control and manage access to a flash memory so that the flash memory appears to the computer operating system as a data storage device in which it is possible to read data from, and write data to, any flash memory location,” through various techniques of mapping and address construction and derivation. EX1010, 1:51-2:55. For example, the patent teaches a “virtual mapping system that allows data to be continuously written to unwritten physical address locations,” *id.*, Abstract, based on “computer generated addresses [that] are comprised of a block number and a block offset,” *id.*, 4:51-66.

H. Sprouse (EX1011)

71. U.S. Pat. Pub. No. 2011/0153911 (“Sprouse”) was filed on December 18, 2009, published June 23, 2011 and issued on April 4, 1995. I understand that it is prior art under at least AIA 35 U.S.C. § 102.

72. Based on the “common practice with flash memory is to operate multiple memory die in parallel to increase write performance,” Sprouse describes a memory system comprised of multiple dies where the unit of memory, *e.g.*, a metablock, may be comprised of blocks spanning multiple dies, “achieving memory die parallelism at block level.” EX1011, [0003]-[0007]. Sprouse further teaches that its “approach to block interleaving is scalable to any number of planes per die.” *Id.*, [0036].

I. Cornwell (EX1012)

73. U.S. Pat. No. 2012/0072807 (“Cornwell”) was filed on November 28, 2011, and was published March 22, 2012. It is prior art under AIA 35 U.S.C. § 102.

74. Cornwell describes a memory system, termed a “host,” which includes a “nonvolatile memory” “a memory 111, a memory controller 112, a processor 113, a presentation interface 114, and/or a communication interface,” and an external host, “such as a computer.” EX1012, [0004], [0020]. Cornwell describes the utility of controller logic enabling “[m]etadate [to] be transmitted from a memory of a host to an external host,” where that information “may be analyzed” by the external host and “operations may be performed based on the analysis.” *Id.*, [0006]-[0008], [0043]. For example, the metadata transmitted to the host may include “time stamps,” “bad block data,” “erase count” and “mapping metadata.” *Id.*, [0006]-[0012], [0055].

VII. INVALIDITY OF CLAIMS

A. GROUND I: CLAIMS 1-6, 8, 11-13, 15-20, 22, 25-30 ARE RENDERED OBVIOUS BY OLBRICH IN LIGHT OF YAMADA, AND NVME STD

1. Motivation to combine Olbrich, Yamada and NVMe1.0b

75. A person of ordinary skill in the art would have been motivated to combine Olbrich, Yamada and NVMe1.0b as they all discuss methods and systems for the improvement and optimization of NAND Flash SSD memory management. Moreover, these references are

analogous because they all discuss various means of memory management relying on the generation and tracking of metadata where that metadata is updated over time and that metadata is used for activities to prolong the useful life of the memory. Moreover, the references all disclose techniques for improving the lifetime, data retention, and performance of solid state disk drives (SSDs) and therefore all relate to the same technological field.

76. Memory management systems in SSDs typically rely on and combine ideas from different references to iteratively achieve the superior functionality—and reflect discrete improvements in the field of memory management—based on known techniques in the art. Accordingly, a POSA would be motivated to combine the above-identified contributions, and variations thereof, to incorporate various metadata and the utility of said metadata, *e.g.*, the use of maintenance metadata for maintenance activities to increase efficiency and memory system longevity. These combinations would also have been based on aggregating elements of known art according to common known methods. In light of the finite number of identified, predictable solutions in the prior art, these combinations would be obvious to try in this field of endeavor.

77. Specifically, a POSA would have been motivated to (i) incorporate Yamada’s teaching regarding controller maintenance requests and host-initiated maintenance commands and (ii) implement various aspects of the NVMe1.0b standard.

78. First, 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. *E.g.*, EX1004, 1:45-52, 2:63-3:17, 11:18-28, Fig. 1; EX1005, [0005]-[0011], [0207]-[0211], [0029].

79. Second, Olbrich teaches various means to “maximize[] use of the host and flash interfaces” and through the “use of multiple microprocessors in the controller design, thereby

creating multiple parallel independent pipelines.” EX1004, 2:64-66. Due to this architecture, the system of Olbrich is capable of ready modification to include the discrete Maintenance Request Unit/Module disclosed in Yamada. A POSA would have had a reasonable expectation of success in incorporating this shared host functionality—adding a maintenance request module to the controller and translocating the maintenance command responsibility from SSD controller firmware to host-side software—because, as described in more detail above, the modification would have been straightforward and routine and yielded predictable results.

80. A POSA would be motivated to do so due to Yamada’s descriptions of the benefits of a particular form of shared host and controller responsibility, namely, host-initiated maintenance/copy commands. Yamada describes the benefits to data retention, longevity and performance of flash memory based on shared control with the host with respect to maintenance activities. *See, e.g.*, EX1005, [0013] (“The adaptive command unit determines whether data stored in the nonvolatile memory needs maintenance based on a data retention status of the nonvolatile memory obtained from the nonvolatile memory device, and determines whether maintenance is to be performed based on whether the access device is in a system status in which maintenance is enabled and based on whether the data stored in the nonvolatile memory needs maintenance, and generates a maintenance command to the nonvolatile memory device when determining that the maintenance is to be performed.”). First, incorporating the teaching of Yamada “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.” *Id.*, [0036]. Second, the techniques in Yamada “enable[] the maintenance to be performed with a sufficiently long processing time allocated to the maintenance, and improves the data retention properties,” and, third, they “enable[] 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.*

81. In light of the flexible hardware architecture in Olbrich and the benefits of the discrete shared host and control responsibility in Yamada, there is a clear motivation to incorporate Yamada’s teaching regarding host-initiated maintenance commands based on maintenance requests from the controller based on tracked metadata. A POSA could implement the additional discrete type of shared host and controller memory management disclosed in Yamada into the system of Olbrich with a reasonable expectation of success because the incorporation of discrete aspects of shared responsibility merely requires the transposing of certain data and logic from the firmware of the controller to the software of the host. Such relocation of functions from the controller to the host represents routine work that is well-understood and predictable combinations.

82. Third, Olbrich teaches that the claimed systems and methods are flexible and are “designed to operate with various enterprise-level mass storage protocols” such as “the Small Computer Systems Interface (‘SCSI’), and Serial ATA (‘SATA’) protocols” and “may be adapted for use with other similar protocols, either currently in use or not yet developed.” EX1004, 4:35-55. A POSA would therefore be motivated to incorporate the teachings of an NVMe express standard and could do so with a reasonable expectation of success. Indeed, the ’772 Patent explicitly calls for such incorporation in, *e.g.*, Claim 1, “wherein the SSD and the host communicate over a host interface, using commands from a Non-Volatile Memory Express (NVMe) standard.” EX1001, Claim 1.

83. Additionally, the reasons for combining various references in other grounds, as relevant to broadly applicable motivations to incorporate additional forms of metadata and discrete implementations of shared host functionality set out in the other grounds, are incorporated by

reference here. Further, my reasoning for motivations to combine as set out within the claim-by-claim analysis is also incorporated by reference.

B. Claim-by-Claim Analysis

- 1. Claim 1[pre]: A method for providing a host with access to an SSD that contains logic and non-volatile memory comprising NAND flash memory, the logic comprising circuitry and/or a processor executing instructions, and the NAND flash memory comprising a plurality of erase units and each erase unit comprising a plurality of pages, the method comprising, at the SSD:**

84. It is my opinion that, to the extent the preamble is limiting, it is disclosed by Olbrich.

85. Olbrich teaches a “solid state nonvolatile memory,” comprised of “NAND flash interface” where the nonvolatile memory includes erase “blocks, each made up of 64 pages,” where a host is provided access via a “controller” with logic comprising circuitry and/or a “processor” executing instruction. EX1004, 1:45-52, 2:59-3:21, 9:45-48, 11:18-28, Fig. 1, Claim 1.

- (a) Claim 1[a]: dividing the flash memory into zones; wherein the zones are non-overlapping subdivisions of the flash memory, each zone comprises two or more erase units, the erase units in each zone are on two or more different planes,**

86. It is my opinion that Olbrich teaches this limitation.

87. Olbrich teaches *dividing the flash memory into “SuperBlocks,” [zones]*, where each “are made up of the same Block on each Bank in the Flash Group (e.g., Block 0 1201 is part of SuperBlock 0 of the Flash Group).” EX1004, 16:12-16. Figure 16 shows a “simplified architecture” where “SuperBlock 0 of Flash Group 0 consists of Block 0 from each Bank, designated in FIG. 16 as 1602 (i.e., the entire Flash Group, in the simplified example),” *id.*, 19:11-19, and each SuperBlock may include SuperPages, where each SuperPage consists of four flash Pages, each of which is stored on a different Flash Die” and thus different plane, *id.*, 6:23-29. In the figure below, I have annotated, in a red box, one SuperBlock comprising two SuperPages. A

POSA would understand that blocks [*erase units*] on different banks and dies represent non-overlapping blocks on different planes. Olbrich thus teaches *erase units in each [SuperBlock] are on two or more different planes*.

1601 BANK	FLASH GROUP 0		FLASH GROUP 1		FLASH GROUP 2	
	SP0	SP1	SP0	SP1	SP0	SP1
0	SMT	285-299	SMT	375-389	SMT	465-479
1	0-14	300-314	90-104	390-404	180-194	480-494
2	15-29	315-329	105-119	405-419	195-209	495-509
3	30-44	330-344	120-134	420-434	210-224	510-524
4	45-59	345-359	135-149	435-449	225-239	525-539
5	60-74	540-554	150-164	585-599	240-254	630-644
6	75-89	555-569	165-179	600-614	255-269	645-659
7	270-284	570-584	360-374	615-629	450-464	660-674

1602

FIG. 16

Id., Fig. 16.

88. In all, Olbrich’s SuperBlocks and its SuperPages represent zones, which include pages from non-overlapping blocks on different planes.

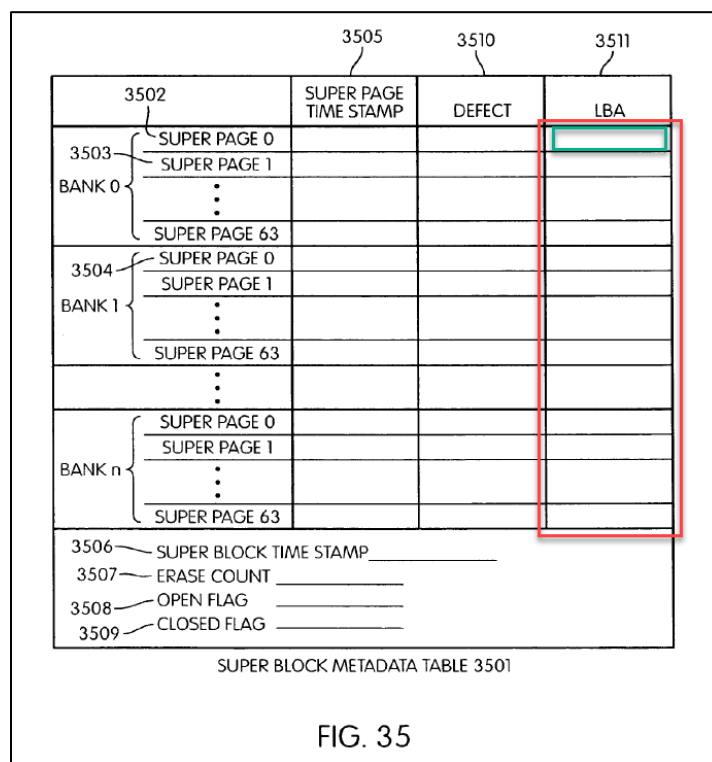
(b) Claim 1[b]: and each zone is associated with a zone start logical block address and a range of logical block addresses;

89. It is my opinion that Olbrich teaches this limitation.

90. Olbrich teaches that each SuperBlock includes SuperPages, each spanning four dies. EX1004, 6:23-29. The SuperPage is identified by “the same address, and this is the starting address of a page of data in each of the Dies.” *Id.*, 72:30-38. The SuperPage is “associated with 15 consecutive LBAs,” ending with the “highest LBA.” *Id.* 72:30-38, 15:37-49.

91. As shown in Figure 35 below, a SuperBlock may be comprised of SuperPages (SP1 – SP63) and includes “a zone start logical block address” (“ZSLBA”), the “*starting address*” of

SP1 (annotated in a green box below), and “a range of logical block addresses,” spanning “*consecutive LBAs*” (annotated in a red box below) *Id.*



Id., Fig. 35.

- (c) **Claim 1[c]: in response to a write command from the host to write data to the range of logical block addresses of a specified zone, performing a multi-plane write of said data to erase units on different planes in the specified zone using a common page address for the respective erase units on the different planes;**

92. It is my opinion that Olbrich teaches this limitation.

93. Olbrich teaches that the controller may “receive a write command for a particular block of LBAs,” or “a write command and an LBA” from a host that is directed to a SuperBlock—***in response to a write command from the host to write data to the range of logical block addresses of a specified zone.*** EX1004, 15:27-34, 25:21-41, 41:15-24.

94. Olbrich teaches that the write command may designate a multi-plane write across SuperPages on different dies, which a POSA would understand to mean a multi-plane write across

erase units on different planes (as each die comprises, *e.g.*, four planes), through the use of “the same address location”—***using a common page address for the respective erase units on the different planes.*** *Id.* 15:60-67 (“Pages 1202, 1203, 1204 and 1205 are stored at the same address location on four different Flash Memory Dies in the same Flash Bank. Thus, SuperPage 30 is spread across four separate Dies.”), 54:61-67 (“Flash Write Address Field 4007. This contains the Flash Group address to be used for a write by the Page Request, and is calculated based on the SuperPage to be used for the write.”).

(d) Claim 1[d]: generating metadata based on accessing and maintenance of erase units in the specified zone; and;

95. It is my opinion that Olbrich teaches this limitation.

96. Olbrich teaches the generation of “Time Stamp” SuperBlock ***maintenance*** metadata “indicating the time when each SuperBlock was originally closed for writing,” and ***accessing*** metadata showing “the number of times the associated LBAs have been accessed (read or written).” EX1004, 47:45-49, 46:19-29. Since the LBAs correspond to SuperBlocks, a POSA would understand that this metadata collectively describes access to SuperBlocks and all their constituent blocks and pages. Because maintenance operations to SuperBlocks are applied to all erase units in the SuperBlock, a POSA would understand that this metadata may be specific to the erase units with the Superblock. Olbrich therefore teaches time of writing and access count metadata for SuperBlocks—***generating metadata based on accessing and maintenance of erase units in the specified zone.***

97. Olbrich also teaches, for maintenance operations such as “wear-leveling” (*see infra* 1[e]) the generation of “Erase Count” SuperBlock metadata, representing “the number of times the SuperBlock has been erased,” *id.*, 48:39-41—***metadata based on ... maintenance of erase units in the specified zone.***

- (e) **Claim 1[e]: based on the metadata, making a notification available to the host that recommends performing maintenance on the specified zone, wherein the recommendation is associated with the zone start logical block address of the specified zone;**

98. It is my opinion that Olbrich and the combined system of Olbrich and Yamada teach this limitation.

99. Olbrich teaches “maintenance commands,” whose plain and ordinary meaning encompasses operations such as wear-leveling and garbage collection (EX1001, 9:48-52), including commands that manage erase-count metadata used for those functions (EX1004, 48:39-41; *see also* Claim 1 and Claim 24). For example, Olbrich teaches that a SuperBlock may be selected for garbage collection based on SuperBlock erase count metadata. EX1004, Claim 1, Claim 24 (“...the instructions that select the first memory area do so at least in part based on the number of times the first memory area has been erased in relation to the number of times at least one other memory area in the first subgroup has been erased; ...”), 92:18-22 (“SuperBlocks with an Erase Count value that exceeds a threshold could be identified so that they are preferentially not chosen for normal garbage collection, thereby reducing additional wear on such SuperBlocks.”). Thus, Olbrich teaches SuperBlock garbage collection, ***performing maintenance on the specified zone***, depending on SuperBlock erase count metadata, ***based on the metadata***.

100. Olbrich also teaches initiating maintenance based on metadata exceeding a threshold. For example, it teaches that where a “Counter value” (SuperBlock capacity) “is at or below the Critical Threshold, a Critical Flag is set and the [device] shifts into critical garbage collection mode, with garbage collection operations taking precedence.” *Id.*, 80:16-81:38.

101. To the extent Olbrich does not teach this limitation, a POSA would be motivated to incorporate Yamada’s teaching of host-initiated maintenance commands based on metadata received from the controller, due to the benefits over traditional methods, including (i) parallel

maintenance operations “without requiring the data maintenance to be performed together with normal processing,” (ii) “improv[ing] the data retention properties,” and (iii) ensuring “maintenance will be completed in a reliable manner.” EX1005, [0036]. A POSA would also be so motivated by Olbrich’s teaching that the “SSD Controller 106 could use explicit wear leveling techniques.” EX1004, 91:64-65.

102. Yamada teaches ***based on the metadata***: that the controller’s “maintenance request unit 115 determines whether an area (in units of blocks) in the nonvolatile memory 120 from which data has been read needs maintenance based on [metadata, *e.g.*,] the number of errors determined by the error correction unit.” EX1005, [0081]. Yamada teaches ***making a notification available to the host***: upon “determining that the area needs maintenance, the maintenance request unit 115 transmits a maintenance request to the access module” in the host, *id.*, [0081-0082]. Yamada also teaches ***recommends performing maintenance on the specified zone***: where that “maintenance request may include a plurality of logical addresses of a plurality of blocks,” *id.*, [0116], or “a logical address of the start of the maintenance-needed block and a logical address of the end of the maintenance-needed block,” *id.*, [0101].

103. When incorporating the teaching of Yamada into the system of Olbrich, where Yamada already discloses a maintenance request to a series of blocks, a POSA would understand that the maintenance request as applied to Olbrich’s SuperBlock would include a start logical block address corresponding to the SuperBlock (as described *supra*)—***is associated with the zone start logical block address of the specified zone***.

(f) **Claim 1[f]: wherein the SSD and the host communicate over a host interface, using commands from a Non-Volatile Memory Express (NVMe) standard**

104. It is my opinion that Olbrich teaches this limitation.

105. Olbrich teaches that the claimed systems and methods are:

[D]esigned to operate with various enterprise-level mass storage protocols, including SAS ("Serial Attached SCSI"), FC ("Fibre Chamiel") and FC-AL ("Fibre Chamiel Arbitrated Loop), all of which are based on the Small 40 Computer Systems Interface ("SCSI"), and Serial ATA ("SATA") protocols. These protocols are highly familiar to those of ordinary skill in the art, and will not be further described herein. Except where particular protocols are called out, **the systems and methods disclosed herein do not depend on the particular protocol being used and are designed to operate correctly with all of them.** Moreover, **these systems and methods may be adapted for use with other similar protocols, either currently in use or not yet developed,** including protocols designed for enterprise-level applications as well as protocols designed for other applications, such as end-user.

EX1004, 4:35-55. A POSA would therefore be motivated to incorporate an NVMe standard, such as NVMe1.0b (EX1006)—***wherein the SSD and the host communicate over a host interface, using commands from a Non-Volatile Memory Express (NVMe) standard.***

2. Claim 2: A solid state storage drive (SSD) comprising:

106. It is my opinion that Olbrich teaches this limitation, as I described in Claim 1[pre].

(a) 2[a]: a host interface configured to communicate with a host using commands from a Non-Volatile Memory Express (NVMe) standard;

107. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this limitation obvious, as I described in Claim 1[f].

(b) 2[b]: non-volatile memory comprising NAND flash memory, the NAND flash memory comprising a plurality of erase units and each erase unit comprising a plurality of pages;;

108. It is my opinion that Olbrich teaches this limitation, as I described in Claim 1[pre].

(c) 2[c]: logic coupled to the host interface and to the flash memory, wherein the logic comprises at least one of hardware logic and software logic, and communicates with the host via the host interface using NVMe commands;

109. It is my opinion that Olbrich teaches this limitation, as I described in Claim 1[pre],
1[f].

110. Olbrich further teaches hardware logic and software logic, “the control logic consisting of hardware, software and/or firmware.” EX1004, 33:47-61.

- (d) **2[d]: wherein the logic operates to: divide the flash memory into zones; wherein the zones are non-overlapping subdivisions of the flash memory, each zone comprises two or more erase units, the erase units in at least one zone are on two or more different planes,**

111. It is my opinion that Olbrich teaches this limitation, as I described in Claim 1[a].

- (e) **2[e]: and each zone is associated with a zone start logical block address and a range of logical block addresses;**

112. It is my opinion that Olbrich teaches this limitation, as I described in Claim 1[b].

- (f) **2[f]: in response to a write command from the host to write data within the range of logical block addresses of a specified zone, perform a multi-plane write of said data to erase units on different planes in the specified zone using a common page address for the respective erase units on the different planes;**

113. As I described in Claim 1[c], Olbrich teaches this limitation.

- (g) **2[g]: generate metadata based on accessing and maintenance of erase units in the specified zone; and;**

114. As I described in Claim 1[d], Olbrich teaches this limitation.

- (h) **2[h]: based on the metadata, make a notification available to the host that recommends performing maintenance on the specified zone, wherein the recommendation is associated with the zone start logical block address of the specified zone.**

115. As I described in Claim 1[e], the combined system of Olbrich and Yamada renders this limitation obvious.

- 3. **Claim 3: The SSD of claim 2 further comprising: additional non-volatile memory, wherein the logic further operates to store the generated metadata in the additional non-volatile memory.**

116. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

117. As set out in Claim 2, it is my opinion the combination of Olbrich, Yamada and NVMe1.0b teaches the SSD of Claim 2.

118. Olbrich teaches that the controller has logic “store metadata associated with the SuperPage” through the use of “extra bytes” and that “SuperPage 0 of Block 0 of each SuperBlock contains a data structure known as the SuperBlock Metadata Table”—***additional non-volatile memory, wherein the logic further operates to store the generated metadata in the additional non-volatile memory.*** EX1004, 17:55-18:2, 16:8-36, 19:16-19.

4. Claim 4: The SSD of claim 2 wherein the logic further operates to compare the metadata with at least one threshold, and wherein making the notification available to the host is dependent on the comparison.

119. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

120. As set out in Claim 1[d]-[e] and Claim 2, the combination of Olbrich, Yamada and NVMe1.0b teaches the SSD of Claim 2 and teaches the use of SuperBlock metadata to trigger garbage collection.

121. Olbrich further teaches the user of a Counter indicating SuperBlock capacity which is compared against a “Garbage Collection Threshold”:

This Counter contains the number of SuperBlocks currently on SuperBlock Free List 2912. Following the decrementing of Counter 2913, the value currently held in the ***Counter is compared to two values maintained in Garbage Collection Thresholds Section 2915 of the Flash HEMi’s mRAM: the Critical Threshold and the Non-Critical Threshold.*** ... If the Counter value (and thus the number of SuperBlocks on the Super Block Freelist) is at or below the Critical Threshold, a Critical Flag is set and the Flash HEMi ***shifts into critical garbage collection mode.***

EX1004, 80:1-24. In sum, Olbrich teaches the use of “***Garbage Collection Thresholds,***” used to trigger “***garbage collection operations.***”

122. Olbrich also teaches the use of an erase count threshold to identify particular SuperBlocks for garbage collection—“This contains a count of the number of times the SuperBlock has been erased, and can be used, for example, to prioritize SuperBlocks with higher erase counts for garbage collection”—and to exclude particular SuperBlocks from garbage collection to prevent further wear—“SuperBlocks with an Erase Count value that exceeds a threshold could be identified so that they are preferentially not chosen for normal garbage collection, thereby reducing additional wear on such SuperBlocks.” *See, id.*, 92:18-22, 41:13-18, 86:24-35; 47:50-54, 91:64-92:22.

123. Olbrich thus teaches ***the logic further operates to compare the metadata with at least one threshold.***

124. Yamada also teaches this limitation, the controller identifies blocks for maintenance “based on the number of errors,” where that metadata is compared against a “criterion value” or “threshold,”—***logic further operates to compare the metadata with at least one threshold.*** EX1005, [0081], [0189]. Upon satisfying the threshold, the controller “transmits a maintenance request to the access module” (host), *i.e.*, ***wherein making a notification available to the host is dependent on the comparison.*** *Id.* A POSA would understand that in the combined system this analysis would be conducted by Olbrich’s controller and be at the Superblock-level.

5. Claim 5: The SSD of claim 2

125. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

126. As set out in Claim 2, it is my opinion that the combination of Olbrich, Yamada and NVMe1.0b teaches the SSD of Claim 2.

- (a) wherein the logic further operates to determine, based on the metadata, that maintenance is recommended for the specified zone and to identify the zone start logical block address**

associated with the specified zone for which maintenance is recommended;

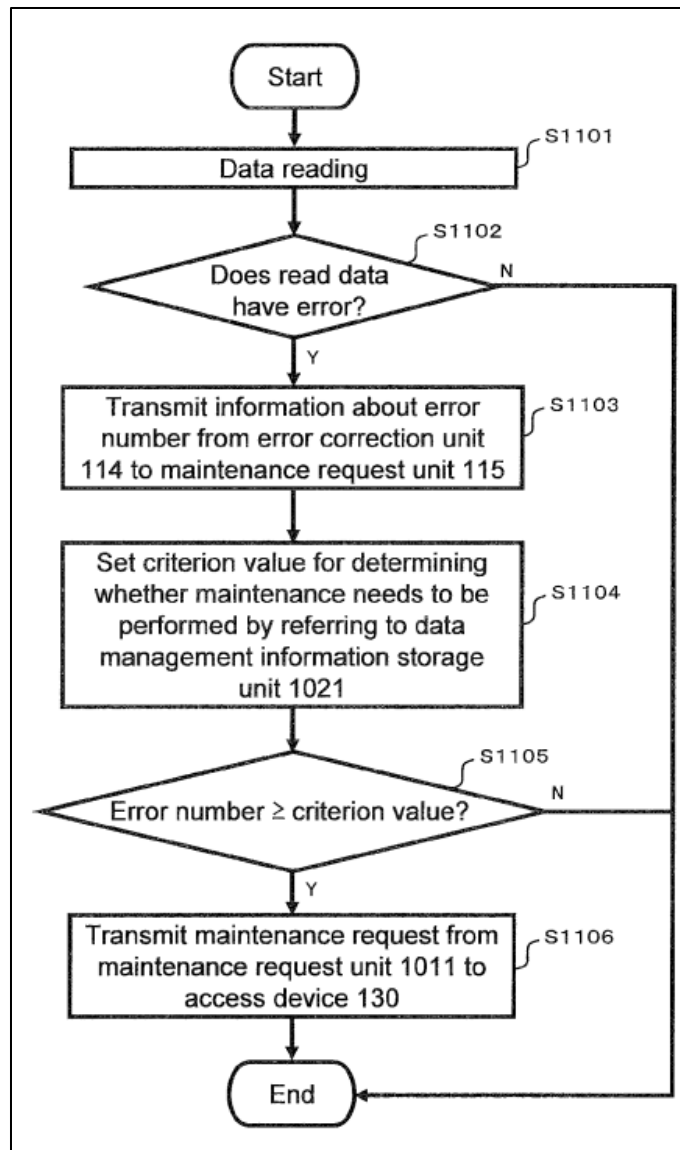
127. As I described for Claim 4, it is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b, discloses an SSD with logic ***to determine, based on the metadata, that maintenance is recommended for the specified zone.***

128. As I described for Claim 1[e], it is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b, discloses an SSD with logic ***to identify the zone start logical block address associated with the specified zone for which maintenance is recommended.***

(b) wherein determining that maintenance is recommended for the specified zone, identifying the zone start logical block address associated with the specified zone, and making the notification available to the host are all unsolicited by the host.

129. As set out in Claim 1[e], it is my opinion that Yamada teaches that the maintenance request unit notifies the host that maintenance is recommended.

130. Further, Yamada teaches that “[t]he information for identifying the maintenance-needed block [in the notification to the host] may be, for example, ... a logical address of the start of the maintenance-needed block and the size of the block.” EX1005, [0101]. Yamada teaches that this maintenance request process is unsolicited by the host as shown in Figure 11 which illustrates “a flowchart showing maintenance need determination and maintenance request generation performed by a nonvolatile memory module [(the controller)].” *Id.*, [0048].



Id., Fig. 11, [0185]-[0185], [0178] (“The nonvolatile memory module 100A includes a memory controller 1010 and a nonvolatile memory 1020.”). A POSA would understand that the process of identifying the data for maintenance and recommending maintenance may be ***performed entirely without the host.***

131. It would be obvious to incorporate this teaching into the system of Olbrich, organized into units of SuperBlocks; a POSA would understand that the maintenance request must include an identifying SuperBlock address such as the SuperBlock starting logical address—

wherein determining that maintenance is recommended for the specified zone, identifying the zone start logical block address associated with the specified zone.

132. That a POSA would be so motivated is corroborated by the NVMe1.0b—which the combination system complies pursuant to Claim 1. The NVMe1.0b describes “an asynchronous event ... issued to the host,” *e.g.*, an unsolicited host notification based on comparing block capacity to a threshold: “Available Spare Threshold: When the Available Spare falls below the threshold indicated in this field, an asynchronous event may be issued to the host. The value is indicated as a normalized percentage (0 to 100%);” “Asynchronous events are used to notify host software of error and health information as these events occur.” EX1006, § 5.2, Fig. 59, Fig. 31.

6. **Claim 6: The SSD of claim 2 wherein the logic further operates to receive a command from the host; and, in response to said command, transmit to the host (a) information representing list of host accessible zones including a zone start logical block address for each listed zone, and (b) a host accessible capacity for each listed zone.**

133. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

134. As I set out in Claim 2, the combination of Olbrich, Yamada and NVMe1.0b teaches the *SSD of Claim 2*.

135. Olbrich further teaches that the controller tracks *information representing list of host accessible zones*—for example, the controller maintains a “SuperBlock Freelist ... contains identifiers for the SuperBlocks... that are free, and therefore available to be written;” a POSA would understand the Freelist to collectively describe the list of accessible SuperBlocks. EX1004, 41:3-6; 79:46-59 (“Each Flash Group includes a number of SuperBlocks, which are in one of three states: (1) open (a single SuperBlock that is currently open for writing; (2) closed (SuperBlocks into which data has been written, but which are no longer open for writing) and (3) free

(SuperBlocks on SuperBlock Freelist 2912, which have been erased and are available for future writes).”).

136. As another disclosure of **metadata representing host accessible zones**, as further set out *infra*, Olbrich discloses that in the “SuperBlock Metadata Table,” each SuperBlock is marked with an open, closed, free and/or defect flag—which a POSA would understand to collectively represent host accessibility. EX1004, 48:30-46. The metadata showing host accessibility is associated with each SuperBlock through the identifying LBA, including the LBA for SuperPage 0 in Bank 0, *i.e.*, **a zone start logical block address for each listed zone**. *Id.*, Fig. 35, 88:44-49.

137. To the extent this limitation is construed such that host accessible capacity represents the amount of unwritten space available to the host, the identification of each LBA as written or defective, or unwritten and not defective, provides this information to the POSA.

138. Olbrich therefore teaches the maintenance of **information representing list of host accessible zones including a zone start logical block address for each listed zone**.

139. To the extent Olbrich does not teach this limitation, a POSA would be motivated to incorporate the teaching of the NVMe1.0b regarding the utility and value of maintaining metadata in the form of a list describing zones (SuperBlocks) and sharing that information with the host. As I described for Claim 1[f], the combined system complies with NVMe1.0b. Accordingly, a POSA would heed its teaching that the host may request list metadata. The NVMe1.0b organizes memory based on namespaces—defined as “a collection of logical blocks that range from 0 to the capacity of the namespace”—and provides metadata to the host including the “Number of Namespaces” (**a list of host accessible zones**) “Namespace Utilization” and “Namespace Capacity” (representing

“the number of logical blocks currently allocated” and “the maximum number of logical blocks that may be allocated” respectively – **host accessible capacity**). EX1006, Fig. 67.

140. A POSA would understand that in the combined system, this list metadata would be adapted to SuperBlocks and when transmitted to the host would describe Superblock metadata. Indeed, based on the teaching of Olbrich and Yamada, the combined system already teaches the maintenance of a SuperBlock Freelist (**a list of host accessible zones**) and SuperBlock Metadata Table (**host accessible capacity**), and functionality to transmit this information **to the host**; the NVMe1.0b provides additional motivation to a POSA to implement the claimed functionality.

7. Claim 8: The SSD of claim 2 wherein:

141. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

142. As set out in Claim 2, the combination of Olbrich, Yamada and NVMe1.0b teaches the SSD of Claim 2.

- (a) the logic further operates to receive read commands and write commands from the host, and to execute the read commands and the write commands by transferring associated data between the host and the NAND flash memory;**

143. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

144. Olbrich teaches that the controller’s logic enables it to receive and accept “a Host-initiated read or write command”—**the logic further operates to receive read commands and write commands from the host**. EX1004, 6:1-8. In a host-initiated write command, the host “provides the data to the SSD controller” to write in flash memory, and in response to host-initiated read commands, the controller “convey[s] the data [in flash memory] to the Host,”—**to execute the**

read commands and the write commands by transferring associated data between the host and the NAND flash memory. EX1004, 8:64-65, 79:20-23, Figs. 1, 4, 5.

- (b) **and each read command directed to the SSD is accompanied by a logical read address, each logical read address has a first address portion which identifies one of the zones according to a value of the first portion, and a second address portion which identifies a physical memory location in the SSD in a manner that depends on the first address portion.**

145. Olbrich teaches that host commands, including read and write commands, include a “Command Descriptor Block (‘CDB’) or other corresponding command-related information from the Host” which “specifies the address range to be read from or written to, in Logical Block Addresses (‘LBAs’).” EX1004, 6:1-8. For example, Olbrich teaches that a read command includes “the LBA of the data.” *Id.*, 8:36-7, 46:30-37. Using the LBA, the controller conducts a division operation where the “LBA received from a Host ... is divided by 15, with the whole-number quotient used.” *Id.*, 54:53-60. The resulting quotient represents the address’s first portion and is used to identify the block and SuperBlock which is subject to the read command—***read address has a first address portion which identifies one of the zones according to a value of the first portion.*** *Id.*, 46:30-37; 54:53-60, 89:30-32 (“The Forward Table entry for the LBA now points to this SuperPage [and SuperBlock] as storing the current, valid version of the LBA data.”). The division operation includes a remainder (as the quotient is a whole-number), and this remainder (which is dependent on the first portion) represents the address’s second portion and is used to identify the exact physical location of the memory based on page and sector—***a second address portion which identifies a physical memory location in the SSD in a manner that depends on the first address portion.*** *Id.*, 46:51-60.

8. Claim 11: The SSD of claim 2 wherein the flash memory comprises multiple dies of flash memory and the erase units on different planes are also in different dies.,

146. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

147. As set out in Claim 2, it is my opinion that the combination of Olbrich, Yamada and NVMe1.0b teaches the SSD of Claim 2.

148. As set out in Claim 1[a]-[c], Olbrich teaches that SuperBlocks span “four different Flash Memory Dies”—*wherein the flash memory comprises multiple dies of flash memory and the erase units on different planes are also in different dies*. EX1004, 16:64-67 (“Pages 1202, 1203, 1204 and 1205 are stored at the same address location on four different Flash Memory Dies in the same Flash Bank. Thus, SuperPage 30 is spread across four separate Dies.”), 16:12-16 (“The currently preferred embodiment also makes use of an additional memory structure that is not shown in FIG. 12: SuperBlocks, which are made up of the same Block on each Bank in the Flash Group (e.g., Block 0 1201 is part of SuperBlock 0 of the Flash Group).”).

9. Claim 12: The SSD of claim 2 wherein the logic further operates to, upon the performance of the multi-plane write, update the metadata to indicate an amount of unerased host accessible space in the specified zone.

149. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

150. As I set out in Claim 2, it is my opinion that the combination of Olbrich, Yamada and NVMe1.0b teaches the SSD of Claim 2.

151. As I set out in Claim 1[c], Olbrich teaches a controller with logic to *perform a multi-plane write*. During the multi-plane write, the controller updates the “Metadata Table” to memorialize where the data is located, “the appropriate LBA,” along with the “SuperPage Time

Stamp entry in Column 3505 is written with the time the SuperPage was written.” EX1004, 48:47-54 (“When a SuperBlock is open for writing, its Metadata Table is copied from flash into DRAM 107. As each SuperPage is written, the LBA field corresponding to that SuperPage is written with the appropriate LBA, and the SuperPage Time Stamp entry in Column 3505 is written with the time the SuperPage was written. Similarly, if a SuperPage is discovered to be defective, the Defect field corresponding to that SuperPage is filled in.”). In addition, the controller updates the metadata showing the available “room for a write”—the number of “free SuperPages”—***logic further operates to ... update the metadata to indicate an amount of unerased host accessible space in the specified zone.*** *Id.*, 79:60-67 (“...if the open SuperBlock contains insufficient room for a write (i.e., no free SuperPages shown on SuperPage Pointer List 2919)...”).

10. Claim 13: The SSD of claim 12 wherein the logic further operates to: upon the performance of the multi-plane write, detect that the specified zone contains no erased host accessible space; and, in response to said detection, transition the specified zone to a full state.

152. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

153. As I set out in Claim 12, the combined system of Olbrich, Yamada and NVMe1.0b teaches ***the SSD of claim 12, wherein the controller has logic to perform a multi-plane write.***

154. Olbrich further teaches that upon the completion of writing, *e.g.*, where the SuperBlock’s capacity is reached, the controller has logic to classify the SuperBlock in a state of “closed.” EX1004, 77:59-67 (“[i]f all Banks are checked and no SuperPages are available (“yes” outcome to Step 5406), then the currently open SuperBlock is full and must be closed”).

155. Specifically, the controller determines whether the SuperBlock is full or if there is “insufficient room for a write (i.e., no free SuperPages shown on SuperPage Pointer List 2919),” whereby a “closed flag is set”—***detect that the specified zone contains no erased host accessible***

space; and, in response to said detection, transition the specified zone to a full state. EX1004, 79:60-67, 48:44-46, 79:46-50 (“Each Flash Group includes a number of SuperBlocks, which are in one of three states: (... (2) closed (SuperBlocks into which data has been written, but which are no longer open for writing) ...”).

11. **Claim 15: The SSD of claim 2 wherein the logic further operates to receive a maintenance command directed to the specified zone and, in response to the maintenance command, erase host accessible space only within the specified zone and transition the specified zone to an empty state.**

156. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

157. As I described in Claims 1 and 2, the combined system of Olbrich, Yamada, and NVMe1.0b teaches the *SSD of claim 2 wherein the logic further operates to receive a maintenance command directed to the specified zone.* Yamada’s maintenance command inherently requires erasing the blocks identified in the maintenance command.

158. Moreover, Olbrich already teaches in completing maintenance on a source SuperBlock (zone), “valid data [is] ... erased in connection with a garbage collection,”—*erases host accessible space only within the specified zone.* EX1004, 81:6-7 (“Garbage collection requires the selection of a SuperBlock for erasure”), 91:64-92:22 (“as is described above in connection with FIG. 56, in one embodiment valid data from SuperBlocks that are being erased in connection with a garbage collection operation are stored in a garbage collection SuperBlock, resulting in relatively fewer data invalidations on the garbage collection SuperBlock, such that erasures of that SuperBlock will tend to occur relatively less frequently”), Claim 1 (“performing garbage collection operations for the respective group of flash memory devices, comprising copying data from a first flash memory area to a second flash memory area and erasing the first flash memory area”).

159. Following the erasure of the SuperBlock as part of maintenance, the SuperBlock metadata is updated whereby the SuperBlock is flagged as “free” and is identified on the “SuperBlock FreeList”—***and transition the specified zone to an empty state. E.g., EX1004, Fig. 56; 83:6-9*** (“In Step 5610, the SuperBlock is placed back onto SuperBlock Free List 2912 and SuperBlock Freelist Counter 2913 is incremented.”). Following completion of that metadata update, “[t]he garbage collection operation for that SuperBlock then completes.” *Id.*

12. Claim 16: The SSD of claim 2 wherein the metadata is dependent on an erase count for the specified zone; and the logic further operates to compare the metadata with at least one threshold for erase count, wherein making the notification available to the host is dependent on the comparison.

160. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

161. As I described in Claims 2, the combination of Olbrich, Yamada and NVMe1.0b teaches the ***SSD of Claim 2***.

162. As I described in Claims 1 and 4, the combination of Olbrich, Yamada and NVMe1.0b teaches a controller ***with logic to compare the metadata with at least one threshold ... wherein making the notification available to the host is dependent on the comparison.***

163. Olbrich teaches the SuperBlocks may be identified for, or excluded from, maintenance based on erase count metadata and comparison of that metadata to a threshold—“a count of the number of times the SuperBlock has been erased, [which] can be used, for example, to prioritize SuperBlocks with higher erase counts for garbage collection;” “SuperBlocks with an Erase Count value that exceeds a threshold could be identified so that they are preferentially not chosen for normal garbage collection, thereby reducing additional wear on such SuperBlocks.” EX1004, 47:50-54, 92:18-22. *See also id.*, Claim 1, Claim 24 (“...the instructions that select the first memory area do so at least in part based on the number of times the first memory area has

been erased in relation to the number of times at least one other memory area in the first subgroup has been erased; ...”). Olbrich thus teaches ***wherein the metadata is dependent on an erase count for the specified zone and the logic further operates to compare the metadata with at least one threshold for erase count.***

164. In the combined system, a POSA would be motivated to rely on this erase count threshold as one of the triggering conditions for the maintenance request sent to the host by the controller—***wherein making the notification available to the host is dependent on the comparison.*** Indeed, Yamada already teaches a controller with logic to “transmit[] a maintenance request” based comparison against a “criterion value” or “threshold.” EX1005, [0081], [0189]. A POSA would view it as obvious to incorporate additional metadata-threshold comparisons available: (i) in order to increase the utility of the maintenance request generation cycle through reliance on multiple different means to assess when data is in need of maintenance (especially where Olbrich already teaches use of erase count metadata threshold comparison) and (ii) Yamada’s teaching that the controller’s threshold comparison “dynamically changes the criterion value used to determine whether maintenance needs to be performed,” *id.*, [0183], [0189].

13. Claim 17: The SSD of claim 2 wherein the logic further operates to: in response to a maintenance command from the host directed to the zone start logical block address of the specified zone, perform an erase of erase units in the specified zone, and transition a state of the specified zone to an empty state whereby all host accessible pages of the specified zone are available for writing by the host.

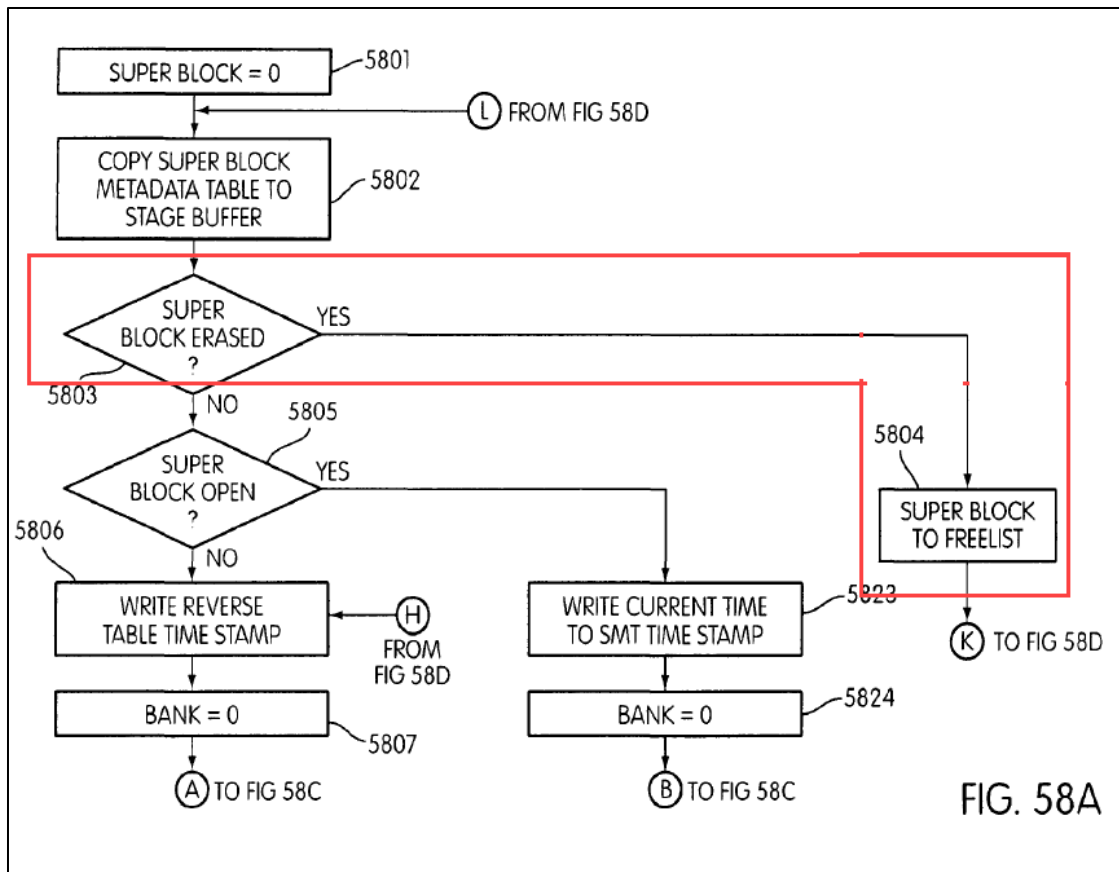
165. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

166. As I described for Claim 2, the combined system of Olbrich, Yamada and NVMe1.0b teaches the ***SSD of claim 2.***

167. As I described in Claim 15, the combined system of Olbrich, Yamada and NVMe1.0b, teaches ***the SSD of claim 2 wherein the logic further operates to: in response to a maintenance command from the host ... perform an erase of the erase units in the specified zone, and transition a state of the specified zone to an empty state.***

168. As I described in Claim 1[b]-[c], Olbrich teaches that a command directed to a SuperBlock may include a “***starting address***”—“a zone start logical block address” (“ZSLBA”). EX1004, 72:33-38. Thus, as I described in Claim 1[e], to incorporate Yamada’s teaching the teaching of Yamada into the system of Olbrich, a POSA would understand that the maintenance request as applied to a SuperBlock would include a start logical block address corresponding to the SuperBlock—***in response to a maintenance command from the host directed to the zone start logical block address of the specified zone.***

169. Additionally, Olbrich teaches that, after completing an erasure as part of a maintenance request, the controller designates the SuperBlock as “free”—representing that the SuperBlock’s constituent pages are “erased and are available for future writes”—***whereby all host accessible pages of the specified zone are available for writing by the host.*** *Id.*, 41:3-7; 79:46-59. A clear illustration of the transition to a free state is shown in, *e.g.*, Figure 58A annotated below:



Id., Fig. 58A.

14. **Claim 18: The SSD of claim 17 wherein the maintenance command from the host is in response to the notification that recommends performing maintenance on the specified zone.**

170. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

171. As I described for Claim 17, the combined system of Olbrich, Yamada and NVMe1.0b teaches the *SSD of claim 17*.

172. Further, as I described in Claim 1[e], the combined system of Olbrich, Yamada and NVMe1.0b teaches a maintenance request from the controller to the host for performing maintenance on SuperBlock. A POSA would understand that the purpose of the maintenance request—the *notification that recommends performing maintenance on the specified zone*—

results in the host, via the maintenance determination unit, sending ***a maintenance command in response***—indeed, Yamada teaches that upon “receiving a maintenance request from the nonvolatile memory module,” the host “generat[es] a maintenance command.” EX1005, [0100], Claim 1.

15. Claim 19: The SSD of claim 17 wherein the metadata is dependent on erase counts for erase units in the specified zone; and the logic further operates to update the metadata responsive to performance of the erase.

173. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

174. As I described for Claim 17, it is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b teaches ***the SSD of claim 17***.

175. As I described for Claim 16, it is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b teaches a maintenance operation ***wherein the metadata is dependent on erase counts for erase units in the specified zone***.

176. Olbrich further includes an explicit teaching where the controller’s logic operates to update the metadata responsive to performance of the erase—the controller “***increments Erase Count Field 3507 in the SuperBlock Metadata Table, reflecting the fact that the SuperBlock has now been erased***.” See, e.g., EX1004, 82:57-60.

16. Claim 20: The SSD of claim 17 wherein performing the erase of erase units in the specified zone comprises performing the erase on erase units of different planes in the specified zone

177. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

178. As I described for Claim 17, it is my opinion that Olbrich teaches the SSD of claim 17 with logic for ***performing the erase of erase units in the specified zone***.

179. As I described for Claim 1[a], it is my opinion that Olbrich teaches that SuperBlocks comprises erase units on different planes, *i.e.*, ***performing the erase on erase units of different planes in the specified zone.***

17. Claim 22: The SSD of claim 2

180. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

181. As I described for Claim 2, it is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b teaches the SSD of claim 2.

(a) wherein the logic further operates to, in response to a write command from the host to the zone start logical block address of the specified zone:

182. As I described for Claim 1[b]-[c], it is my opinion that Olbrich teaches a controller may receive a ***write command from the host*** where the SuperBlock includes a ***start logical block address of the specified SuperBlock [zone]***. See also EX1004, 15:27-49 (“When a Host writes data to mass storage, the Host typically transmits a CDB that contains a write command and an LBA, though the details of Host communication vary depending on the specific protocol used by the Host. When the Host subsequently wishes to read that data, it issues a read command using the same LBA.”), Fig. 5 (“Host issues write command with LBA and data”).

(b) assign a logical block address within the range of logical block addresses of the specified zone, transfer data from the host to the specified zone at the assigned logical block address,

183. Olbrich teaches that a “write command” from the host specifies an LBA. EX1004, Fig. 5. In response to the write command, the controller starts the “Input-Output Process” (“IOP”) (comprised of “Transfer Requests”) which ***specifies the address range*** to be ... written to, in Logical Block Addresses (‘LBAs’).” *Id.*, 6:1-24. During the IOP, Olbrich teaches that the controller may “automatically” assign LBAs for the data subject to the write request within the

“entire LBA range” to effectuate the transfer operations completed as part of the IOP—***assign a logical block address within the range of logical block addresses of the specified zone***—“Each IOP specifies an LBA range, and the IOPs are issued periodically until the entire LBA range governed by SSD Controller 106 is covered. In this embodiment, it is not necessary to separately calculate which LBAs are assigned to which Flash Groups, since ***this is automatically handled as SSD Controller*** 106 deals with the IOP.” *Id.*, 86:48-55.

184. Based on the “Transfer Request[s]” the controller “write[s] a segment of Flash Memory Module 108 corresponding to a portion of the LBA range specified by the Transfer Request”—***transfer data from the host to the specified zone at the assigned logical block address***. *Id.*, 6:9-24; *see also id.* 52:63-53:4 (“LBA Field 3903. This contains the starting LBA for the transfer. For example, if the calling IOP requires a read from 65 LBAs 0-100, LBA Field 3903 for a first Transfer Request would read ‘0’, LBA Field 3903 for a second Transfer Request would read ‘45’ and LBA Field 3903 for a third Transfer Request would read ‘90’ (as is explained above, each Transfer Request can handle a maximum of 45 LBAs). Transfer Length Field 3904. This contains the number of LBAs to be transferred.”).

185. As an additional disclosure demonstrating that Olbrich renders this limitation obvious, Olbrich teaches that as data is written into a range of LBAs in a SuperBlock, the controller determines the physical page to which the stored LBAs are written to. EX1004, 48:47-54 (“When a SuperBlock is open for writing, its Metadata Table is copied from flash into DRAM 107. As each SuperPage is written, the LBA field corresponding to that SuperPage is written with the appropriate LBA, and the SuperPage Time Stamp entry in Column 3505 is written with the time the SuperPage was written.”).

(c) and return the assigned logical block address to the host.

186. Yamada teaches that upon completion of a write operation as part of a maintenance operation, the controller transmits to the host a “maintenance completion notification [which] contains a logical address of a block included in the nonvolatile memory 120 for which maintenance has been completed”—***return the assigned logical block address to the host.*** EX1005, [0108]-[0111]. A POSA would be motivated to incorporate this teaching in the combined system as part of the broader maintenance request already part of the combined system as I described *supra* with respect to, *e.g.*, Claim 1.

18. Claim 25: The SSD of claim 2

187. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

188. As I described for Claim 2, it is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b teaches ***the SSD of claim 2.***

(a) wherein the metadata is dependent on a time period since data was written to the specified zone;

189. Olbrich further teaches collecting and storing “Time Stamp” metadata which represents information showing “the time when each SuperBlock was originally closed for writing” as part of the “SuperBlock Metadata Table” and “Reverse Table;” moreover, on top of that metadata showing the final write to a SuperBlock prior to reaching capacity, Olbrich teaches that the controller maintains metadata for the individual writes to SuperPages within a SuperBlock: “Time Stamp 1403 contains a time stamp representing the time when the SuperPage was written.” EX1004, 48; 30-38, 47:45-49, 18:8-9. Thus Olbrich teaches—***the metadata is dependent on a time period since data was written to the specified zone.***

190. To the extent Olbrich does not teach this limitation, a POSA would be motivated to incorporate the additional variant of metadata in Yamada which includes the aggregate tracking of “**access history of data.**” EX1005, [0211]. This metadata is used to trigger maintenance, e.g., where it identifies data “that **has not been accessed for a long period of time and that needs maintenance**”—wherein the metadata is dependent on a time period since data was written to the specified zone. *Id.*

(b) **and the logic further operates to compare the metadata with at least one threshold, wherein making the notification available to the host is dependent on the comparison**

191. As I described for Claim 1[e], the combined system of Olbrich, Yamada and NVMe1.0b teaches that the controller, the maintenance request unit, “determines whether an area (in units of blocks) in the nonvolatile memory 120 from which data has been read needs maintenance.” EX1005, [0081]. Yamada also teaches that the controller identifies blocks to be copied based on comparison to a threshold, a “criterion value,” **the logic further operates to compare the metadata with at least one threshold.** *Id.*, [0032]-[0033], Fig. 10 (showing the metadata stored in the controller (“1021”)); *see also id.*, [0187] (in “the data management information storage unit 1021”). And, Yamada teaches the data in need of maintenance (copy request) may be identified because it “has not been accessed for a long period of time” or based on the “**whether the elapsed time is greater than or equal to a criterion value.**” *Id.*, [0211], [0164], [0204]. In incorporating these teachings into Olbrich, a POSA would understand that the controller must complete the metadata threshold comparison and that the maintenance request notification to the host requesting copying would be directed to a SuperBlock.

19. Claim 26: The SSD of claim 25 wherein comparing the metadata with at least one threshold and making the notification available to the host are unsolicited by the host

192. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

193. As I described for Claim 25, it is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b teaches *the SSD of claim 25 wherein comparing the metadata with at least one threshold.*

194. As I described for Claim 5, it is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b teaches *making the notification available to the host are unsolicited by the host.*

20. Claim 27: The SSD of claim 2 wherein logic further operates to, in response to a command from the host, transmit to the host information dependent on a time period since data was written to the specified zone.

195. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

196. As I described for Claim 2, it is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b teaches *the SSD of claim 2.*

197. Yamada teaches that the host, “the maintenance determination unit,” may request and “obtain[] the write history stored in the data management information storage unit 721 (information about the time at which data has been previously written to the address).” EX1005, [0163]-[0164]. To incorporate this teaching into the combined system, a POSA would understand that, because Olbrich teaches that Superblock metadata is stored in the nonvolatile memory and maintained by the controller, in order to provide the time period since last written metadata to the host for use operations such as maintenance, the controller may transmit this metadata to the host

in response to a query or command—*the controller must, in response to a command from the host, transmit to the host information dependent on a time period since data was written to the specified zone.* See also Claims 1[e], 25.

21. **Claim 28: The SSD of claim 2 wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone and transmit to the host the zone start logical block address and a state of the specified zone.**

198. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

199. As I described for Claim 2, it is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b teaches *the SSD of claim 2.*

200. Olbrich teaches that “SSD Controller is uniquely suited to deal with flash memory errors, since the large number of processors, and the multiple independent pipelines, allow Controller to detect and correct errors more or less continuously without degrading host-perspective system performance.” EX1004, 84:45-49. Such errors within a SuperBlock may represent a “physical defect [which] is permanent” where the affected SuperPage is “permanently mapped out and no longer used.” *Id.*, 84:22-37.

201. Olbrich teaches that the controller detects these and other defects and marks the failure condition in memory within the SuperBlock with “Defect Flag”—“Defect Flag Column 3510 contains a single bit indicating whether the SuperPage has been marked as “defective.” *Id.*, 48:13-20. A SuperPage is identified as defective if any page “is indicated as being defective during a flash read or a write.” *Id.*, 48:13-20. This teaching also applies at the Superblock-level. *Id.*, 44:64-45:7 (“In an alternate embodiment, a **SuperBlock containing errors may be identified** for preferential garbage collection, since the presence of errors in certain sectors may mean that

adjoining sectors also contain errors”). Olbrich thus teaches ***wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone.***

202. Olbrich further teaches that in circumstances where “some type of error” occurs, this results in the controller’s “***issuance of an appropriate ... code to the Host***, indicating a transaction failure (or other similar communication, depending on the protocol being used).” *Id.*, 16:64-17:27. In the combined system, a POSA would understand that such a code transmitted to the host may be used to ***transmit to the host the zone start logical block address and a state of the specified zone*** for which there is a failure condition.

203. Olbrich teaches that its flexible architecture, “[t]he design of SSD Controller 106[,] allows significant choice in terms of error correction algorithms, several of which are described herein,” *id.* 84:50-52, thus a POSA would be motivated to incorporate the teaching of Yamada or the NVMe1.0b regarding reporting of the failure state to the host as alternate means of addressing errors. Yamada, for example, describes that the controller contains logic when a block has accumulated sufficient errors to be in a failure condition—***logic further operates to detect a failure condition of one of the erase units of the specified zone.*** See EX1005, [0032] (“The maintenance request unit generates a maintenance request indicating that the nonvolatile memory needs maintenance to the access device when the number of errors determined by the error correction unit is greater than or equal to a criterion value.”), [0006] (“data can accumulate so many errors that cannot be corrected using an ECC”). Where such errors surpassing a threshold occur, the controller notifies the host via a maintenance request identifying the relevant blocks, *e.g.*, “logical address of the start of the maintenance-needed block and a logical address of the end of the maintenance-needed block.” *Id.*, [0101]. The maintenance request notification to the host represents information “indicating that the nonvolatile memory needs maintenance”—***a state of***

the specified zone. *Id.*, Claim 10, [0032]. To apply this teaching to the system in Olbrich, organized by SuperBlocks, a POSA would understand that the notification would include a SuperBlock starting LBA—***transmit to the host the zone start logical block address.***

204. Moreover, the NVMe1.0b provides an analogous teaching regarding transmitting to the host the “state of the specified zone.” As set out above at Claim 1[f], a POSA would be motivated to incorporate the teaching of the NVMe Standard because the combined system already complies with it. The NVMe1.0b teaches that the controller may provide information to the host via SMART / Health functionality, where the underlying health [*state*] information is sent on a per namespace basis. *Id.*, § 5.10.1.2. A POSA would understand that to apply the teaching to the combined system, the information transmitted to the host, including the error state of a namespace and the health status of a namespace, would correspond to a SuperBlock—***state of the zone.***

22. Claim 29: The SSD of claim 2 wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone and, in response to a write command from the host within the range of logical block addresses of the specified zone, transmit to the host a notification of a write failure.

205. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

206. As set out for Claim 28, it is my opinion that the combined system of Olbrich, Yamada, and NVMe1.0b teaches the SSD of claim 2, wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone and transmit to the host the state of the zone—a failure state.

207. Olbrich teaches a controller that flags a failure condition in a SuperBlock through a defect flag, where that SuperBlock may be detected “as being defective during a flash read or a write”—***wherein the logic further operates to detect a failure condition of one of the erase units***

of the specified zone and, in response to a write command from the host within the range of logical block addresses of the specified zone. EX1004, 48:13-20.

208. To the extent Olbrich does not teach this limitation, it is my opinion that a POSA would be motivated to incorporate the teaching of the NVMe1.0b and/Yamada.

209. The NVMe1.0b teaches an explicit ***notification of a write failure in response to a write command from the host***: “[w]hen the command is completed with success or failure, the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.” EX1006, § 6.9.1 Command Completion.

210. Yamada provides a complimentary teaching where errors are detected in “data that is written to nonvolatile memory,” and the host is notified about the failure condition via a maintenance request sent to the host. EX1005, [0081]. A POSA would understand that to apply the teaching of Yamada to the combination system in Olbrich, the write failure would correspond to a SuperBlock—***wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone and, in response to a write command from the host within the range of logical block addresses of the specified zone.***

23. Claim 30: The SSD of claim 2 wherein the metadata is dependent on read errors detected for the specified zone.

211. It is my opinion that the combined system of Olbrich, Yamada and NVMe1.0b renders this claim obvious.

212. As set out for Claim 2, it is my opinion that the combined system of Olbrich, Yamada, Cornwell, and NVMe1.0b teaches ***the SSD of Claim 2.***

213. Yamada teaches that the failure condition in data may be detected during a read command; specifically, the failure condition is determined by “the number of errors included in a data read”—“The maintenance request unit 115 determines whether an area (in units of blocks) in

the nonvolatile memory 120 *from which data has been read needs maintenance based on the number of errors* determined by the error correction unit 114”—*wherein the metadata is dependent on read errors detected for the specified zone.* EX1005, [0081]-[0082] (“For data that is read from the nonvolatile memory 120, the error correction unit 114 performs error detection and error correction of the data and also determines the number of errors in the data using an ECC. The functions of the error correction unit 114 may be implemented by the CPU 112 performing software processing. The maintenance request unit 115 determines whether an area (in units of blocks) in the nonvolatile memory 120 from which data has been read needs maintenance based on the number of errors determined by the error correction unit 114. When determining that the area needs maintenance, the maintenance request unit 115 transmits a maintenance request to the access module 130.”), [0032] (“The error correction unit determines the number of errors included in data read from the nonvolatile memory [t]he maintenance request unit generates a maintenance request indicating that the nonvolatile memory needs maintenance to the access device when the number of errors determined by the error correction unit is greater than or equal to a criterion value.”).

214. A POSA would be motivated to incorporate this disclosure from Yamada into the combined system because the NVMe1.0b provides an analogous teaching of notifying the host of a failure condition in response to a read command: “When the command is completed with success or failure, the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.” EX1006, § 6.8.1 Command Completion

VIII. GROUND II: Claims 1-6 ARE RENDERED OBVIOUS BY GONZALEZ in light of GOSS and NVMe1.0b

A. Motivation to Combine

215. A POSA would have been motivated to combine the teachings of Goss and NVMe1.0b with Gonzalez. Specifically, a POSA would have been motivated to (i) incorporate Goss’s teaching regarding notification to the host regarding the status of data and the need for maintenance, and (ii) implement various aspects of the NVMe1.0b standard.

216. First, Gonzalez and Goss involve similar techniques to solve the same problem, *i.e.*, collecting, storing and using metadata (such as metadata describing “data retention characteristics”) to “improve performance,” “operational reliability,” and efficiency of NAND flash memory systems, based on, *e.g.*, informed host commands and operations to minimize and address wear. *E.g.*, EX1007, [0003]-[0008], [0016], [0051], [0066], [0075], [0088]; EX1008, [0005], [0021], [0024], [0048], [0085], [0091].

217. Second, A POSA would have been motivated to combine the teachings of Goss, direct commands by the host with respect maintenance such as data hardening based on metadata provided, because Gonzalez already teaches that memory management responsibilities may be distributed between the host and controller, including “direct control by the host system.” EX1007, [0037]. Goss teaches the benefits of the controller’s “real-time notifications to the host,” including the providing of “status” information relating to the “wear history of each block” and the use of the same status information as the triggering condition for such a notification. *See, e.g.*, EX1008, [0024], [0048], [0085], [0091] (“The wear history of each block 130 is tracked in relation to a wear indication (WI) value which indicates the relative status of the block in terms of its remaining operational life.”). A POSA would have had a reasonable expectation of success in incorporating this shared host functionality—adding controller notification to the host and translocating the maintenance command responsibility from SSD controller firmware to host-side software—and it

would have been straightforward and routine. In light of Gonzalez’s teaching of host control and memory management, it would have been obvious to a POSA to incorporate this teaching from Goss as it represents a discrete means to provide the host with the necessary information (at the appropriate time) regarding the state of memory units and the need for maintenance activities to prevent undue data degradation.

218. Third, a POSA would be motivated to ensure the combination system complies with the NVMe1.0b, as it represents a then-contemporaneous widely adopted industry standard for NAND flash memory devices.

219. The reasons for combining various references in other grounds, as relevant to broadly applicable motivations to incorporate additional forms of metadata and discrete implementations of shared host functionality set out in the other grounds are incorporated by reference here. Further, my reasoning for motivations to combine as set out within the claim-by-claim analysis is also incorporated by reference.

B. Claim-by-Claim Analysis

1. **Claim 1[pre]: A method for providing a host with access to an SSD that contains logic and non-volatile memory comprising NAND flash memory, the logic comprising circuitry and/or a processor executing instructions, and the NAND flash memory comprising a plurality of erase units and each erase unit comprising a plurality of pages, the method comprising, at the SSD:**

220. It is my opinion that, to the extent the preamble is limiting, it is disclosed by Gonzalez.

221. Gonzalez discloses “non-volatile memory products” *including flash memory products comprising a “NAND array” where “the block is the erase unit*, a minimum number of cells that are simultaneously erasable[,] [and] [e]*ach block typically stores one or more pages* of data. EX1007, [0003]-[0008].

222. Gonzalez discloses methods *for providing a host with access to an SSD that contains ... logic comprising circuitry and/or a processor executing instructions*—a “controller” that interfaces with the “host interface” and is “able, by its structure and controlling firmware, to cause data to be programmed and read under a variety of conditions imposed upon it by the host.” EX1007, [0011], Figs. 1, 2.

- (a) **Claim 1[a]: dividing the flash memory into zones; wherein the zones are non-overlapping subdivisions of the flash memory, each zone comprises two or more erase units, the erase units in each zone are on two or more different planes,**

223. It is my opinion that Gonzalez teaches this limitation.

Gonzalez teaches:

To further efficiently manage the memory, *blocks may be linked together as metablocks*; that is, each logically addressable metablock is defined to *include one physical block from each plane*. Use of the metablock is described in international patent application publication no. WO 02/058074. The metablock is identified by a common logical address as a destination for programming and reading data. Similarly, *all blocks of a metablock are erased together*.

EX1007, [0010], [0018] (“each metablock is formed of one block from each of several or all of the planes of the memory array”). A POSA would understand that metablocks represent non-overlapping subdivisions. Therefore, Gonzalez teaches metablocks which *divide the flash memory into zones; wherein the zones are non-overlapping subdivisions of the flash memory, each zone comprises two or more erase units, the erase units in each zone are on two or more different planes*.

- (b) **Claim 1[b]: and each zone is associated with a zone start logical block address and a range of logical block addresses;**

224. It is my opinion that Gonzalez teaches this limitation.

225. Gonzalez teaches “[t]he metablock is identified by a common logical address as a destination for programming and reading data.” EX1007, [0010]. A POSA would understand the common logical address to be a ***zone start logical block address***.

226. Additionally, Gonzales further teaches “[t]here are various methods of maintaining mapping information, and a table of starting LBAs is one example. A table that is kept in memory is used to locate the starting LBAs for each physical memory block.” *Id.*, [0076]. A POSA would understand that the ***zone start logical block address*** would be the “starting LBA” of the first block in the metablock. Gonzales also teaches that the metablock includes a “***range of consecutive host addresses***”—a range of logical block addresses. *Id.*, [0044], [0018].

(c) **Claim 1[c]: in response to a write command from the host to write data to the range of logical block addresses of a specified zone, performing a multi-plane write of said data to erase units on different planes in the specified zone using a common page address for the respective erase units on the different planes;**

227. It is my opinion that Gonzalez teaches this limitation.

228. It is my opinion that, as I set out in Claim 1[b], Gonzalez teaches metablocks comprised of blocks on different planes, and that a “metablock is identified by a ***common logical address*** as a destination for programming and reading data.” EX1007, [0010]. A POSA would understand that the common logical address is equivalent to a ***common page address***.

229. Gonzalez teaches ***in response to a write command from the host to write data to the range of logical block addresses of a specified zone, performing a multi-plane write of said data to erase units on different planes in the specified zone***, as set out in the following disclosure:

Parallel interleaving refers to the memory being formatted such that sequentially addressed host sectors are interleaved between planes; that is, ***multiple sequentially addressed data sectors are simultaneously programmed in parallel in pages of different planes. Data are simultaneously programmed to one block of each plane making up a metablock*** before data are programmed into the next series of blocks that form another metablock.

EX1007, [0057]. *See also id.*, [0018] (“a number of consecutively addressed data sectors are programmed in parallel across multiple blocks of the metablock”).

230. Gonzalez further teaches that “[t]he metablock is identified by ***a common logical address as a destination for programming and reading data***”—using a common page address for the respective erase units on the different planes. *Id.*, [0010].

(d) Claim 1[d]: generating metadata based on accessing and maintenance of erase units in the specified zone; and;

231. It is my opinion that Gonzalez teaches this limitation.

232. Gonzalez teaches that the controller “monitor[s] and store[s]” “programming characteristic[s]” are “monitored and stored by the controller” “on a metablock basis,” such as “***host accesses***” and “the number of times less than all the pages of a block or metablock have been accessed for data writing”—***generating metadata based on accessing ... of erase units in the specified zone***. EX1007, [0066], [0075], [0088] (“the accumulated statistics may [] be maintained ... on a metablock basis”).

233. Gonzalez also teaches that the “number of times one or more pages or blocks have been written due to host commanded writes versus those due to data consolidation (garbage collection)” is monitored—***generating metadata based on ... maintenance of erase units in the specified zone***. *Id.*

234. Goss also discloses—the controller may monitor “total number of access operations”—and as well as additional forms of useful metadata that a POSA would be motivated to incorporate, such as the “number of erase cycles for the erasure block”—***generating metadata based on accessing and maintenance***. EX1008, [0021]. A POSA would be motivated to apply this teaching from Goss to the system in Gonzales and would understand that the metadata variants necessarily must correspond to the metablock-level.

- (e) **Claim 1[e]: based on the metadata, making a notification available to the host that recommends performing maintenance on the specified zone, wherein the recommendation is associated with the zone start logical block address of the specified zone;**

235. It is my opinion that the combined system of Gonzalez and Goss renders this limitation obvious.

236. Gonzalez teaches “various scenarios under which a metablock, sub-metablock, or portion of either are relocated and/or data are consolidated” as part of system maintenance. EX1007, [0052].

237. In light of Goss’s teaching regarding means “for enhancing the reliability and integrity of data stored in a non-volatile memory” in SSDs, a POSA would be motivated to incorporate Goss’s teaching regarding additional scenarios under which maintenance should be performed. EX1008, [0018]-[0024]. For example, Goss teaches:

In still further embodiments, the wear indication values can be used to alter the voltage profiles used to write and/or read data to/from various locations. ***Self-monitoring routines can provide real-time notifications to a host with regard to the status of the array and/or the data, allowing the user to take affirmative steps to alter the operation of the device to extend its useful life.***

EX1008, [0024]. Such affirmative steps may include, *e.g.*, data hardening—which a POSA would understand to be part of maintenance activities—and “alter[ing] the voltage profiles used to write and/or read data to/from various locations.” *Id.*, Claim 2, [0023]-[0024], [0085] (“The routine may also be implemented by the device performing monitoring and providing a notification to the host that, without data hardening, the data may be below retention specification.”).

238. As part of the “real-time notifications to the host” the controller provides “status” information which may include erase count metadata or “wear history of each block”; and this status information may be the trigger for such a notification, *e.g.*, “the data may be below retention specification.” *Id.*, [0023]-[0024], [0085], [0048] (“The wear history of each block 130 is tracked

in relation to a wear indication (WI) value which indicates the relative status of the block in terms of its remaining operational life.”). Goss thus teaches ***based on the metadata, making a notification available to the host that recommends performing maintenance on the specified zone.*** *Id.*.

239. A POSA would understand that the status information provided to the host, including wear count metadata, includes “overhead control information such ***as the logical addresses of the data stored in the associated erasure block,***” EX1008, [0035], [0076], and, further, a POSA would understand that in order to incorporate this teaching into the system of Gonzalez, the notification would correspond to metablock-level metadata by a common or starting LBA—***wherein the recommendation is associated with the zone start logical block address of the specified zone.*** Indeed, Goss provides just the teaching to the POSA:

Metadata may be generated for use with each block 130. The metadata may be stored in a metadata block 132 appended to the end of the erasure block 130 as shown, or may be stored elsewhere in the device 100 such as in one or more erasure blocks dedicated to this purpose. The metadata provides certain types of overhead control information such as the logical addresses of the data stored in the associated erasure block. Other data relating to the erasure block may also be stored in the metadata block 132, such as a timestamp or other sequence information relating to when the block 130 was most recently allocated for use.

Id., [0035].

(f) Claim 1[f]: wherein the SSD and the host communicate over a host interface, using commands from a Non-Volatile Memory Express (NVMe) standard

240. It is my opinion that the combined system of Gonzalez, Goss and NVMe1.0b renders this limitation obvious.

241. As I described in Claim 1[pre], the combined system of Gonzalez and Goss teaches an SSD.

242. Goss teaches “a controller 102 which *provides top-level control and communication functions as the device interacts with a host device* (not shown) to store and retrieve host user data.” EX1008, [0026]. A POSA would be motivated to incorporate the teaching of the NVMe1.0b standard to achieve these control and communication functions in the combined system of Gonzalez and Goss—*wherein the SSD and the host communicate over a host interface, using commands from a Non-Volatile Memory Express (NVMe) standard*. Indeed, the NVMe1.0b represents a then-contemporaneous widely adopted industry standard for NAND flash memory devices.

2. Claim 2: A solid state storage drive (SSD) comprising:

243. It is my opinion, as I described in Claim 1[pre], that Gonzalez teaches this limitation.

(b) 2[a]: a host interface configured to communicate with a host using commands from a Non-Volatile Memory Express (NVMe) standard;

244. It is my opinion, as I described in Claim 1[f], that the combined system of Gonzalez, Goss and NVMe1.0b renders this limitation obvious.

(c) 2[b]: non-volatile memory comprising NAND flash memory, the NAND flash memory comprising a plurality of erase units and each erase unit comprising a plurality of pages;;

245. It is my opinion, as I described in Claim 1[pre], that Gonzalez teaches this limitation.

(d) 2[c]: logic coupled to the host interface and to the flash memory, wherein the logic comprises at least one of hardware logic and software logic, and communicates with the host via the host interface using NVMe commands;

246. It is my opinion, as I described in Claim 1[f], that the combined system of Gonzalez, Goss and NVMe1.0b renders this limitation obvious.

247. Goss further teaches hardware logic and software logic, “[t]he techniques described herein may be implemented in hardware, software, firmware, or any combination thereof.”

EX1008, 20:63-21:4.

- (e) **2[d]: wherein the logic operates to: divide the flash memory into zones; wherein the zones are non-overlapping subdivisions of the flash memory, each zone comprises two or more erase units, the erase units in at least one zone are on two or more different planes,**

248. It is my opinion, as I described in Claim 1[a], that Gonzalez teaches this limitation.

- (f) **2[e]: and each zone is associated with a zone start logical block address and a range of logical block addresses;**

249. It is my opinion, as I described in Claim 1[b], that Gonzalez teaches this limitation.

- (g) **2[f]: in response to a write command from the host to write data within the range of logical block addresses of a specified zone, perform a multi-plane write of said data to erase units on different planes in the specified zone using a common page address for the respective erase units on the different planes;**

250. It is my opinion, as I described in Claim 1[c], that Gonzalez teaches this limitation obvious.

- (h) **2[g]: generate metadata based on accessing and maintenance of erase units in the specified zone; and;**

251. It is my opinion, as I described in Claim 1[d], the combined system of Gonzalez and Goss renders this limitation obvious.

- (i) **2[h]: based on the metadata, make a notification available to the host that recommends performing maintenance on the specified zone, wherein the recommendation is associated with the zone start logical block address of the specified zone.**

252. It is my opinion, as I described in Claim 1[e], the combined system of Gonzalez and Goss renders this limitation obvious.

3. Claim 3: The SSD of claim 2 further comprising: additional non-volatile memory, wherein the logic further operates to store the generated metadata in the additional non-volatile memory.

253. It is my opinion that the combined system of Gonzalez, Goss and NVMe1.0b renders this claim obvious.

254. It is my opinion that, as set out in Claim 2, the combined system of Gonzalez, Goss and NVMe1.0b teaches *the SSD of Claim 2*.

255. Gonzalez teaches that the metadata “may be stored as overhead data within the data sectors stored in those blocks, or in separate physical blocks,” EX1007, [0065], and Goss teaches metadata may be stored in a “metadata block 132 appended to the end of the erasure block 130 as shown, or may be stored elsewhere in the device 100 such as in one or more erasure blocks dedicated to this purpose,” EX1008, [0035]. Each disclosure teaches this limitation—*wherein the logic further operates to store the generated metadata in the additional non-volatile memory*.

4. Claim 4: The SSD of claim 2 wherein the logic further operates to compare the metadata with at least one threshold, and wherein making the notification available to the host is dependent on the comparison.

256. It is my opinion that the combined system of Gonzalez, Goss and NVMe1.0b renders this claim obvious.

257. It is my opinion that, as set out in Claim 2, the combined system of Gonzalez, Goss and NVMe1.0b teaches *the SSD of Claim 2*.

258. Gonzalez teaches that the controller collects metadata and that “[t]his monitoring can include maintaining a count of the number of times a particular block or blocks have been accessed by the host” where the controller conducts an “*analysis of the stored number or numbers may include comparing with a threshold number*, or comparing the difference of the frequency of two or more access usage patterns with a threshold number.” EX1007, [0065].

259. In the combined system, a POSA would understand that the outcome of the analysis taught in Gonzales, where the comparison results in a metablock satisfies a threshold may result in a host notification flagging the need for maintenance (*i.e.*, data hardening)—**and wherein making the notification available to the host is dependent on the comparison.**

260. Goss provides an additional teaching disclosing this limitation:

The wear indication (WI) value for the target location is obtained at step 208, such as by retrieval from a wear indication value table (142, FIG. 5). Alternatively, the WI value may be calculated at this time based on accumulated metrics. **A determination is made whether the target location has an advanced state of wear. This can be carried out by comparing the WI value to a predetermined threshold T**, as shown by decision step 210.

EX1008, [0074]-[0075]. Thus Goss teaches **wherein making the notification available to the host is dependent on the comparison.**

261. As I set out in Claim 1[e], Goss teaches that during controller initiated “monitoring,” the controller may “provid[e] a notification to the host.” *Supra*, 1[e], EX1008, [0085]. For example, where data is “below retention specifications”—it has been determined to be in an advanced state of wear based on the threshold comparison—a self-monitoring notification may be sent to the host. *See id.*, [0074]-[0075], [0085] (“The routine may also be implemented by the device performing monitoring and providing a notification to the host that, without data hardening, the data may be below retention specifications.”), Claim 2 (“... controller detects said **loss of retention characteristics** of the first location **in relation to a wear indication value** associated with the first location...”). Thus Goss teaches that **wherein making the notification available to the host is dependent on the comparison.**

262. As yet further evidence the combined system teaches this claim, the NVMe1.0b – to which the combination system complies pursuant to Claim 1 – already describes the utility of

unsolicited host notification, and describes “an asynchronous event ... issued to the host,” based on comparing block capacity to a threshold

- “***Asynchronous events are used to notify host software of error and health information as these events occur***”;
- “Available Spare Threshold: ***When the Available Spare falls below the threshold indicated in this field, an asynchronous event may be issued to the host.*** The value is indicated as a normalized percentage (0 to 100%)”

EX1006, § 5.2, Figs 31, 59.

5. Claim 5: The SSD of claim 2

263. It is my opinion that the combined system of Gonzalez, Goss and NVMe1.0b renders this claim obvious.

264. It is my opinion that, as set out in Claim 2, the combined system of Gonzalez, Goss and NVMe1.0b teaches ***the SSD of Claim 2.***

- (a) **wherein the logic further operates to determine, based on the metadata, that maintenance is recommended for the specified zone and to identify the zone start logical block address associated with the specified zone for which maintenance is recommended;**

265. It is my opinion that, as I described for Claim 4, the combined system of Gonzalez, Goss and NVMe1.0b, discloses an SSD with logic ***to determine, based on the metadata, that maintenance is recommended for the specified zone.***

266. It is my opinion that, as I described for Claim 1[e], the combined system of Gonzalez, Goss and NVMe1.0b, discloses an SSD with logic ***“to identify the zone start logical block address associated with the specified zone for which maintenance is recommended.”***

- (b) **wherein determining that maintenance is recommended for the specified zone, identifying the zone start logical block address**

associated with the specified zone, and making the notification available to the host are all unsolicited by the host.

267. It is my opinion that, as I described for Claims 1[e] and 5[a], the combined system of Gonzalez, Goss and NVMe1.0b, discloses an SSD with logic “***wherein determining that maintenance is recommended for the specified zone, identifying the zone start logical block address associated with the specified zone, and making the notification available to the host.***”

268. In the combined system, a POSA would understand that the controller’s determination that maintenance is recommended, and the resulting notification to the host with identifying address information, ***are unsolicited by the host***—Goss teaches that the controller identifies the need for maintenance and sends real-time notifications to the host as part of the controller’s “***self-monitoring*** routines” without any participation of the host, EX1008, [0024], [0085].

269. As I referenced in Claim 4, such asynchronous notification to the host by the controller is also taught by the NVMe1.0b standard, to which the combination system complies pursuant to Claim 1. This is further evidence that the combination of Goss and Gonzalez is obvious to try and reasonably likely to succeed.

6. Claim 6: The SSD of claim 2 wherein the logic further operates to receive a command from the host; and, in response to said command, transmit to the host (a) information representing list of host accessible zones including a zone start logical block address for each listed zone, and (b) a host accessible capacity for each listed zone.

270. It is my opinion that the combined system of Gonzalez, Goss and NVMe1.0b renders this claim obvious.

271. It is my opinion that, as set out in Claim 2, the combined system of Gonzalez, Goss and NVMe1.0b teaches ***the SSD of Claim 2.***

272. As I set out in Ground I, Claim 6, a POSA would be motivated to incorporate the teaching of the NVMe1.0b regarding the maintenance of list metadata and transmitting the same to the host. A POSA would understand that to incorporate this teaching from the NVMe1.0b into the system in Gonzalez, that the list metadata would be applied at the metablock-level—***(a) information representing list of host accessible zones including a zone start logical block address for each listed zone, and (b) a host accessible capacity for each listed zone***—and when transmitted to the host the metadata for each metablock would be identified via a ***zone start logical block address, see supra Claim 1[b], 1[e]***—***and, in response to said command, transmit to the host ... including a zone start logical block address for each listed zone***

IX. GROUND III – CLAIM 8 IS RENDERED OBVIOUS BY GONZALEZ IN VIEW OF GOSS, NVME1.0B AND BAN

A. Motivation to Combine

273. A POSA would have been motivated to combine the teachings of Ban with the combined system of Gonzalez, Goss and NVMe1.0b, namely, Ban’s teaching regarding deriving data location from address information.

274. **First**, Ban, like Gonzalez and NVMe1.0b, teaches the use of groupings of blocks as the unit of a flash memory, namely “zones.” EX1010, 2:11-19. A POSA would understand that the techniques disclosed in Ban to solve known inefficiencies and shortcomings via the use of streamlined mapping information to enable derivation of useful information from host provided addresses, *e.g., id.*, 4:32-44, may naturally be applied to the metablocks in Gonzalez. EX1010, Claim 1.

275. **Second**, A POSA would have been motivated to combine the teaching of Ban regarding address structuring. Ban’s teaches fundamental principles in the field regarding addressing, whereby the address structure enables the controller to more efficiently and effectively

locate the relevant memory structures subject to host commands. This teaching is narrow, relating to the host's construction of addresses provided to the controller, and a POSA would be able to, using well-known methods, incorporate this teaching into the system of Olbrich with a reasonable expectation of success. The taught benefits of such addressing would make this incorporation obvious to a POSA.

276. The reasons for combining various references in other grounds, as relevant to broadly applicable motivations to incorporate additional forms of metadata and discrete implementations of shared host functionality set out in the other grounds, are incorporated by reference here. Further, my reasoning for motivations to combine as set out within the claim-by-claim analysis is also incorporated by reference.

B. Claim-by-claim Analysis

1. Claim 8: The SSD of claim 2 wherein:

277. It is my opinion that the combined system of Gonzalez, Goss and NVMe1.0b, and the combined system of Gonzalez, Goss, NVMe1.0b and Ban renders this claim obvious.

278. It is my opinion that, as set out in Claim 2, the combined system of Gonzalez, Goss and NVMe1.0b teaches *the SSD of Claim 2*.

- (a) the logic further operates to receive read commands and write commands from the host, and to execute the read commands and the write commands by transferring associated data between the host and the NAND flash memory;**

279. Gonzalez teaches that the “controller integrated circuit chip performs host interface and memory control function” including *receiving read commands* and *write commands* including the “logical block addresses (LBAs)” *from the host*. EX1007, [0010], [0037]. In executing read and write commands, the controller transfers data subject to write commands from the host to the nonvolatile memory and transfers data subject to read commands to the host from

the NAND nonvolatile memory—*and to execute the read commands and the write commands by transferring associated data between the host and the NAND flash memory.* See also Claim

1.

- (b) **and each read command directed to the SSD is accompanied by a logical read address, each logical read address has a first address portion which identifies one of the zones according to a value of the first portion, and a second address portion which identifies a physical memory location in the SSD in a manner that depends on the first address portion.**

280. Gonzalez teaches that:

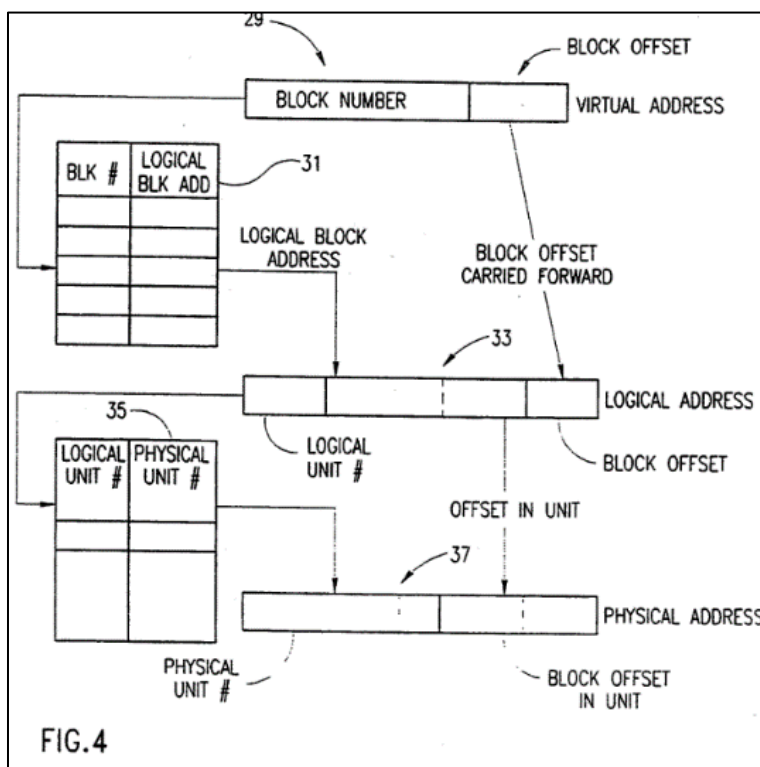
The controller in a memory system operated with such large blocks and/or metablocks performs a number of functions including the *translation between logical block addresses (LBAs) received from a host, and physical block numbers (PBNs) and page addresses* within the memory cell array. This translation often *involves use of intermediate terms of a logical block number (LBN) and logical page.*

EX1007, [0010]. A POSA would understand the PBN derived by the controller from the host LBA *identifies a metablock/one of the zones*—“Data with a given logical block address (LBA) is mapped by the controller into a specific physical memory block within that plane that is identified by a physical block number (PBN).” *Id.*, [0047]. The derived Page address *identifies a physical memory location in the SSD.* Because LBA translation needs to be unique, in particular, an LBA must map into exactly one PBN and one page address, the LBA, which is comprised of a set of address bits, must be partitioned into two disjoint subsets (portions) of address bits—a POSA would thus understand that these portions, used for deriving the PBD and page addresses from the LBA *may rely on a first portion and a second portion of an LBA depending on the first portion.*

281. To the extent Gonzalez does not teach this limitation, a POSA would be motivated to incorporate the teaching of Ban. Ban describes fundamental methods of optimizing mapping and storage in flash memory, including directed to memory devices organized into units of memory comprising a grouping of blocks: “One or more physically contiguous flash memory areas (called

zones) make up a unit which can be physically erased using suitable prior art flash memory technology. Each unit contains an integral number of blocks.” EX1010, 2:15-19.

282. Ban teaches that host commands include an address which comprised two segments or portions directed to “block number and a block offset;” the controller uses these portions to identify the physical block for the data and the particular location of the data within the block respectively—as shown in Figure 4 below.



Id., Fig. 4. See also *id.*, 4:32-44 (“Referring now to FIG. 3, each unit contains an integral number of contiguous data blocks 21 that are in turn comprised of contiguous byte addresses, that can be addressed as a block number and offset within the block. Each block in a unit can be addressed by block number and offset with the unit.”). Incorporating this teaching into the combined system, the first portion of the address would identify the metablock, and the second portion would identify the physical location within the metablock—***each logical read address has a first address portion which identifies one of the zones according to a value of the first portion, and a second address***

portion which identifies a physical memory location in the SSD in a manner that depends on the first address portion.

X. GROUND IV - CLAIM 11 IS RENDERED OBVIOUS BY GONZALEZ IN VIEW OF GOSS, NVME1.0B, AND SPROUSE

A. Motivation to Combine

283. A POSA would have been motivated to combine the teachings of Sprouse with the combined system of Gonzalez, Goss and NVMe1.0b, namely, Sprouse’s teaching regarding metablocks composed of blocks from different dies.

284. The reasons for combining various references to accomplish shared host functionality set out in the other grounds, and those set out in the claims within this ground, are incorporated by reference here.

285. **First**, both Sprouse and Gonzales are part of the SanDisk Technologies LLC patent portfolio, which a POSA would view as evidence of likely interoperability.

286. **Second**, both Sprouse and Gonzalez describe memory systems where the base unit of memory is a groupings of NAND blocks into a “metablock,” where that design is implemented in order to increase parallel performance. EX1007, [0010], [0018]; EX1011, [0027], [0032].

287. **Third**, in light of the matching objectives between Sprouse and Gonzalez, whereby metablocks are used to increase parallelism, it would have been obvious to a POSA to incorporate the additional, additive means to improve parallelism where block interleaving is employed to increase the performance and efficiency of commands directed to a single metablock. EX1007, [0013]-[0018] (discussing “advantages of a high degree of parallelism offered by writing data to a metablock”); EX1011, [0004]-[0007]. Moreover, in light of the shared design structure—organization of memory by metablocks—a POSA would be able to implement the teaching of

Gonzales with a reasonable expectation of success and using well-known methods (as interleaving is a technique long-used in memory devices).

288. The reasons for combining various references in other grounds, as relevant to broadly applicable motivations to incorporate additional forms of metadata and discrete implementations of shared host functionality set out in the other grounds, are incorporated by reference here. Further, my reasoning for motivations to combine as set out within the claim-by-claim analysis is also incorporated by reference.

B. Claim-by-claim Analysis

1. Claim 11: The SSD of claim 2 wherein the flash memory comprises multiple dies of flash memory and the erase units on different planes are also in different dies.

289. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Sprouse renders this claim obvious.

290. It is my opinion that, as set out in Claim 2, the combined system of Gonzalez, Goss and NVMe1.0b teaches *the SSD of Claim 2*.

291. A POSA would be motivated to incorporate the teaching of Sprouse as a related patent that like Gonzalez is assigned to SanDisk and due to the benefits of metablock interleaving discussed therein. Sprouse teaches improving performance through increased parallelism via interleaving blocks across dies to form a metablock: “For example, in a 4-die memory 1200 (Die 0-Die 3) data may be interleaved across 4 MLC planes in two die to create a 6 MB metablock 1202, and then the two pairs of die (Die 0, Die 1 and Die 2, Die 3) may be block interleaved.” EX1011, [0047]. Thus, the combined system teaches *wherein the flash memory comprises multiple dies of flash memory and the erase units on different planes are also in different dies*.

XI. GROUND V – CLAIMS 12, 13, 15-20, 22, 25-27 ARE RENDERED OBVIOUS BY GONZALEZ IN VIEW OF GOSS, NVME1.0B, AND SON

A. Motivation to Combine

292. Son provides a robust disclosure of various forms of metadata and their utility in managing groups of block (termed SuperBlocks), where that metadata is provided to the host in order to take advantage of increased compute power by the host and to enable more efficient memory management. This teaching represents a strong signal to the a POSA that would motivate the incorporation of Son's particular forms of metadata (as well as the transmission to the host of the same) into the system in Gonzalez.

293. **First**, Son, like Gonzalez and Goss, involves similar techniques to solve the same problem, *i.e.*, organizing nonvolatile memory into groupings of NAND blocks, where metadata is tracked for those groupings of blocks, and that metadata is relied upon in order to optimize and streamline the memory device useful life and overall performance. *E.g.*, EX1009, [0002], [0014], Fig. 3C, [0080], [0029], [0038], [0078]-[0080].

294. **Second**, A POSA would be motivated to include, in the system of Gonzalez, the additional metadata variants described in Son, due to the usefulness of such metadata described by Son with respect to host-write, -erase, and -maintenance commands. *E.g.*, EX1009, [0078]-[0080]. Adding these additional types of metadata would have been a straightforward exercise and incorporation into Gonzales could be done with a reasonable expectation of success, as Gonzales already provides the framework and infrastructure of tracking metadata from groupings of blocks.

295. **Third**, a POSA would be motivated to combine Son's host functionality with the combined system. As I described *supra*, Gonzalez teaches that in certain embodiments memory management responsibility may be shared between the controller and the host. A POSA would therefore consider Son's teaching regarding the benefits of "moving solid-state management functions above the firmware level" and providing the host access "both file system information

and information related to the underlying physical solid-state storage media,” whereby several advantages are realized including “enhanced I/O performance, simplified SSD firmware, and extended SSD lifespan.” EX1009, [0014].

296. In light of the commonality in system design and Son’s clear teaching regarding the benefits of its particular variant of shared host controller memory management functionality, a POSA would be motivated to incorporate this teaching from Son. The implementation of this functionality could have been accomplished using well-known methods because such translocation of certain responsibility from the controller’s firmware to host-side software is a common practice that can be completed with a high likelihood of success.

297. The reasons for combining various references in other grounds, as relevant to broadly applicable motivations to incorporate additional forms of metadata and discrete implementations of shared host functionality set out in the other grounds, are incorporated by reference here. Further, my reasoning for motivations to combine as set out within the claim-by-claim analysis is also incorporated by reference.

B. Claim-by-claim Analysis

1. Claim 12: The SSD of claim 2 wherein the logic further operates to, upon the performance of the multi-plane write, update the metadata to indicate an amount of unerased host accessible space in the specified zone.

298. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Son renders this claim obvious.

299. It is my opinion that, as set out in Claim 2, the combined system of Gonzalez, Goss and NVMe1.0b teaches *the SSD of Claim 2*.

300. As set out in Claim 1, the combination of Gonzalez, Goss and NVMe1.0b teaches *logic for performance of the multi-plane write*.

301. A POSA would be motivated to implement the additional forms of metadata and categorization set out in Son, due to the utility described therein.

302. Specifically, a POSA would be motivated to combine Son's teaching of additional useful types of metadata that "associates each SuperBlock [] with free page count, valid page count, invalid page count," and the "number of remaining free pages." EX1009, [0065], [0039], [0067]. Further, Son teaches that the SuperBlock state, including states showing host accessible space, is tracked:

The state column indicates the current state of each SuperBlock. For example, as shown at row 340a, the state column indicates that *SuperBlock 0 is "CLEAN. meaning the SuperBlock has been erased and is available to take new writes.* As another example, as shown at row 340b, the state column indicates that SuperBlock 1 is *"IN USE because there are pages within the SuperBlock that are available for new writes. Other non-exhaustive examples of possible SuperBlock states include "SEALED. indicating that there are no free pages within the SuperBlock because all pages have been written,* "GC. indicating that garbage collection is currently in progress within the SuperBlock, and "WL indicating that wear leveling is currently in progress within the SuperBlock. Of course other SuperBlock states are possible.

Id., [0039]. Thus, Son teaches, *metadata to indicate an amount of unerased host accessible space in the specified zone.*

303. Son also teaches that this metadata is updated after writing—the SuperBlock "state, valid page count, [and] invalid page count" are "update[d]"—*logic further operates to, upon the performance of the multi-plane write, update the metadata.* *Id.*, [0050]-[0052].

304. A POSA would understand that to incorporate the SuperBlock-level metadata described in Son, to the system in Gonzalez, the metadata would be to the metablock-level.

2. **Claim 13: The SSD of claim 12 wherein the logic further operates to: upon the performance of the multi-plane write, detect that the specified zone contains no erased host accessible space; and, in response to said detection, transition the specified zone to a full state.**

305. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Son renders this claim obvious.

306. It is my opinion that, as set out in Claim 12, the combined system of Gonzalez, Goss, Son and NVMe1.0b teaches ***the SSD of claim 12, wherein the controller has logic for performance of the multi-plane write, and to detect that the specified zone contains no erased host accessible space*** such as detection of the “number of remaining free pages.”

307. A POSA would be further motivated to combine Son’s teaching of the classification of SuperBlocks, discussed above, including “‘***SEALED,***’ ***indicating that there are no free pages within the SuperBlock because all pages have been written***”—a POSA would interpret the SEALED state to be tantamount to the claimed FULL state. EX1009, [0040].

3. **Claim 15: The SSD of claim 2 wherein the logic further operates to receive a maintenance command directed to the specified zone and, in response to the maintenance command, erase host accessible space only within the specified zone and transition the specified zone to an empty state.**

308. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Son renders this claim obvious.

309. It is my opinion that, as set out in Claim 2, the combined system of Gonzalez, Goss and NVMe1.0b teaches ***the SSD of Claim 2 wherein the logic further operates to receive a maintenance command directed to the specified zone.***

310. A POSA would understand that a maintenance command, *e.g.*, such as the garbage collection (or consolidation) in Olbrich, includes the copying of data from a source metablock to a target metablock, where the source metablock is erased—***in response to the maintenance command, erase host accessible space only within the specified zone.*** *Supra* Claim 1. *See also* EX1007, [0051] (“erasures associated with consolidation”); EX1008, [0044] (“the garbage collection process will erase the selected block and place the erased block back into an allocation pool of available block”); EX1009, [0070] (“the SSD module instructs the SSD controller to erase the source super block.”).

311. A POSA would be further motivated to combine Son’s teaching of the classification of the state of SuperBlocks, discussed above, “*CLEAN, meaning the SuperBlock has been erased and is available to take new writes*”—a POSA would interpret the CLEAN state to be tantamount to the claimed EMPTY state. EX1009, [0040].

4. Claim 16 The SSD of claim 2 wherein the metadata is dependent on an erase count for the specified zone; and the logic further operates to compare the metadata with at least one threshold for erase count, wherein making the notification available to the host is dependent on the comparison.

312. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Son renders this claim obvious.

313. It is my opinion that, as I described in Claims 2 and 4, the combination of Gonzalez, Goss, and NVMe1.0b teaches the SSD of Claim 2 *with logic to compare the metadata with at least one threshold ... wherein making the notification available to the host is dependent on the comparison.*

314. As I set out in Claims 1[e] and 4, Goss teaches the use of a threshold and wear indication metadata—“[t]he wear indication values provide an accurate indication of the wear history of the associated blocks, and may be determined in relation to the total number of access operations (e.g., erasures, writes) that have taken place upon the associated block. Other metrics associated with the block may also be used to formulate the wear indication value”—for maintenance and host notification. *See generally supra* 1[e]. *See also* EX1008, [0021], [0059], [0075]-[0076]. A POSA would understand that another metric used to calculate wear indication would be “erase count.”

315. To the extent the combination of Gonzales, Goss and NVMe1.0b does not teach this claim, a POSA would be motivated to implement the additional forms of metadata and state classifications set out in Son, into the controller of the combined system due to their described

utility and the existing infrastructure to accommodate the same in the combined system. Specifically, Son teaches the maintenance of “wear count,” “indicat[ing] how many times a particular SuperBlock has been erased,” EX1009, [0080], [0039], [**wherein the metadata is dependent on an erase count for the specified zone**]. Son further teaches that the wear count metadata is useful in a comparison against a threshold to trigger maintenance, “the SSD module determines **whether one or more SuperBlocks have a wear count above a predetermined threshold value**” where upon exceeding that threshold, maintenance, (wear leveling) is implemented. EX1009, [0079]. A POSA would be motivated to incorporate this teaching into the controller of the combined system of Gonzalez and Goss, whereby the erase count metadata would be compared against a threshold for triggering a notification to the host.

5. **Claim 17: The SSD of claim 2 wherein the logic further operates to: in response to a maintenance command from the host directed to the zone start logical block address of the specified zone, perform an erase of erase units in the specified zone, and transition a state of the specified zone to an empty state whereby all host accessible pages of the specified zone are available for writing by the host.**

316. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Son renders this claim obvious.

317. It is my opinion that, as set out in Claim 2, the combined system of Gonzalez, Goss and NVMe1.0b teaches **the SSD of Claim 2**.

318. It is my opinion that, as set out in Claim 15, the combined system of Gonzalez, Goss and NVMe1.0b teaches **the logic further operates to: in response to a maintenance command from the host ... perform an erase of the erase units in the specified zone, and transition a state of the specified zone to an empty state whereby all host accessible pages of the specified zone are available for writing by the host. See, e.g., EX1009, [0040].** (discussing

classifying a SuperBlock as “‘CLEAN,’ meaning the SuperBlock has been erased and is available to take new writes.”)

319. As I described in Claim 1[b]-[c], the combined system teaches that a command to a SuperBlock may be directed to a “**starting address**,” “a zone start logical block address,” representing the first SuperPage in a SuperBlock. EX1004, 72:33-38.

320. As I described in Claim 1[e], when incorporating the teaching of Goss into the system of Gonzalez, a POSA would understand that the maintenance notification would be applied at the metablock level such that the notification would identify a metablock via a start logical block address—***in response to a maintenance command from the host directed to the zone start logical block address of the specified zone.***

6. Claim 18: The SSD of claim 17 wherein the maintenance command from the host is in response to the notification that recommends performing maintenance on the specified zone.

321. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Son renders this claim obvious.

322. It is my opinion that, as set out in Claim 17, the combined system of Gonzalez, Goss and NVMe1.0b teaches ***the SSD of Claim 17.***

323. As I described in Claim 1[e], the combined system of Gonzalez, Goss, and NVMe1.0b teaches ***the notification that recommends performing maintenance on the specified zone.*** The purpose of Goss’s “real-time notifications to a host,” is to provide relevant metadata to the host such that the host may take affirmative actions, such as maintenance operations based on said metadata—***wherein the maintenance command from the host is in response to the notification.*** EX1008, [0024]. For example, Goss teaches that: “The routine may also be implemented by the device performing monitoring and providing a notification to the host that, without data hardening, the data may be below retention specification.” *Id.*, EX1008, [0085].

324. To the extent of the combined system of Gonzalez, Goss and NVMe1.0b, does not teach this limitation, a POSA would be motivated to incorporate the teaching of Son, regarding describes the benefits of host-initiated maintenance commands. A POSA would be motivated to incorporate this teaching in light of the metadata shared with the host in the combined system and the notification of the need for maintenance sent to the host. Son teaches the SSD is managed by a host which is comprised of “an SSD management module (“the SSD module”),” EX1009, [0016] as shown in annotated Figure 1 (yellow box is host, red box is SSD).

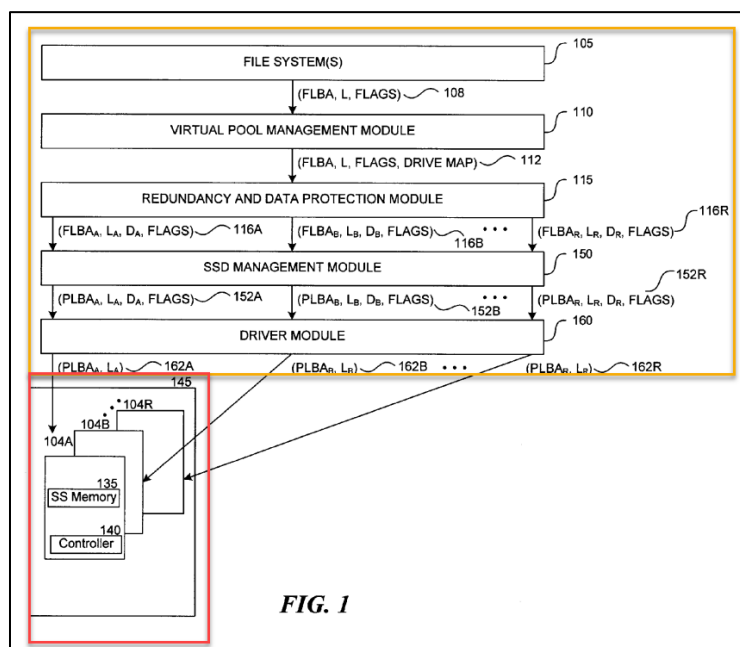


FIG. 1

Id., Fig. 1. Son describes “the failure of conventional SSDs to incorporate file system information into their operational logic [which] often results in inefficiencies.” *Id.*, [0013]. Son describes the shortcomings of “wear-leveling and garbage collection” implemented by the “embedded controller” due to the failure to implement host metadata “during these data management operations;” “[f]or example, embedded controllers typically do not customize data management operations to the type of data implicated by the operation ... [and] do not physically group together data that is logically contiguous within the overlying file system.” *Id.*.

325. Based on this teaching, a POSA would be motivated to “rely upon the SSD module [host] to manage such [a maintenance] operation.” *Id.*, [0017], [0021] (“the SSD module performs various housekeeping operations to help improve the performance of the SSD array, such as wear-leveling, garbage collection, and table recovery operations”). Specifically, Son teaches that the host “combines both a garbage collection process in conjunction with a wear leveling process (both described herein) into a single thread that may be triggered periodically, e.g., every 30 seconds, or upon the occurrence of another condition.” *Id.*, [0063]. Here, the occurrence of a condition triggering maintenance, would be the “real-time notification[]” sent to the host in the combined system—***wherein the maintenance command from the host is in response to the notification.***

7. **Claim 19: The SSD of claim 17 wherein the metadata is dependent on erase counts for erase units in the specified zone; and the logic further operates to update the metadata responsive to performance of the erase.**

326. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Son renders this claim obvious.

327. It is my opinion that, as I described for Claims 15 and 17, the combined system of Gonzalez, Goss, Son and NVMe1.0b teaches ***the SSD of claim 17*** and the updating of metadata responsive to erase and write commands, ***transitioning the SuperBlock to a state of FREE—update the metadata responsive to the performance of the erase.*** Further, it is my opinion that, as I described for Claim 16, the combined system of Gonzalez, Goss, Son and NVMe1.0b teaches a maintenance operation ***wherein the metadata is dependent on erase counts for erase units in the specified zone.***

328. As a separate disclosure of this claim, Son teaches the need to, following host-initiated garbage collection, “update[] management data structures [(e.g., erase count metadata)]

to reflect the erasure of the source SuperBlock.” EX1009, [0074], [0062]-[0082] (“the management data structures are updated to reflect the copying and erasure”).

8. Claim 20: The SSD of claim 17 wherein performing the erase of erase units in the specified zone comprises performing the erase on erase units of different planes in the specified zone.

329. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Son renders this claim obvious.

330. It is my opinion that, as I described for Claims 1 and 17, the combined system of Gonzalez, Goss, Son and NVMe1.0b teaches the SSD of claim 17 with logic for *performing the erase of erase units in the specified metablock*; where the metablock is comprised of metablocks including blocks from different planes and the controller has logic for *performing the erase of erase units of different planes in the specified zone*.

9. Claim 22: The SSD of claim 2

331. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Son renders this claim obvious.

332. It is my opinion that, as set out in Claim 2, the combined system of Gonzalez, Goss and NVMe1.0b teaches *the SSD of Claim 2*.

(a) wherein the logic further operates to, in response to a write command from the host to the zone start logical block address of the specified zone:

333. It is my opinion that, as I described for Claim 1[b]-[c], the combined system of Gonzalez, Goss, and NVMe1.0b teaches a controller may receive a *write command from the host* where the SuperBlock includes a *start logical block address within the range of logical block addresses of the specified SuperBlock*.

- (b) **assign a logical block address within the range of logical block addresses of the specified zone, transfer data from the host to the specified zone at the assigned logical block address,**

334. It is my opinion that, as I described for Claim 1[b]-[c], the combined system of Gonzalez, Goss, and NVMe1.0b teaches a controller may receive a ***write command from the host*** where the controller ***transfers data from the host to the specified zone at the assigned logical block address.***

335. Gonzalez teaches that the controller may receive a host write command with “Data with a given logical block address (LBA),” EX1007, [0047], and the controller “map[s]” that data to “a specific physical memory block within that plane that is identified by a physical block number (PBN),” through the determination of a “***a logical block number (LBN) and logical page,***” ***assign a logical block address within the range of logical block addresses of the specified zone, transfer data from the host to the specified zone at the assigned logical block address.*** *Id.* See also *id.*, [0010].

- (c) **and return the assigned logical block address to the host.**

336. A POSA would be motivated to incorporate the teaching from Son into the combination system, *i.e.*, the benefits of sharing mapping (physical and logical address) information with the host. EX1009, [0014] (“In response to these discoveries and other shortcomings of conventional systems, the inventors have developed a file system-aware SSD management system (‘the system’) including an SSD management module (‘the SSD module’) that ***incorporates both file system information and information related to the underlying physical solid-state storage media into its operations.*** By incorporating both file system and physical storage information, the system may achieve various advantages over conventional systems, such as enhanced I/O performance, simplified SSD firmware, and extended SSD lifespan.”). In light of this clear teaching, describing the need for and benefits of providing

mapping information to the host, a POSA would therefore understand that upon completion of the write command, in the combined system, the controller would transmit the underlying mapping information to the host including “a logical block number (LBN) and logical page”—***return the assigned logical block address to the host.*** Indeed, the system in Goss already sets the framework for providing mapping and related metadata to the host as part of its self-notification functionality as described *supra*.

10. Claim 25: The SSD of claim 2

337. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Son renders this claim obvious.

338. It is my opinion that, as I described for Claim 2, the combined system of Gonzalez, Goss, and NVMe1.0b teaches the ***SSD of claim 2.***

(a) wherein the metadata is dependent on a time period since data was written to the specified zone;

339. To the extent the combined system does not teach this limitation, a POSA would be motivated to incorporate the additional types of metadata described in Son, including “last modification time,” which is defined as “the most recent time that any page within the SuperBlock was ... programmed,” EX1009, [0042], [0080]. A POSA would be so motivated due to the utility of this type of metadata, *e.g.*, in selecting source and target SuperBlocks in garbage collection. EX1009, [0042], Claim 36. *See also infra* Claim 25[b].

(b) and the logic further operates to compare the metadata with at least one threshold, wherein making the notification available to the host is dependent on the comparison

340. To the extent the combined system does not teach this limitation, a POSA would be motivated to incorporate Son’s teaching of the utility of comparing the “last modification time” ***tracked metadata against a threshold*** defined by “a predetermined time period” in light of its

utility in selecting SuperBlocks for garbage collection and the known relevance of data age in memory longevity. EX1009, [0042], Claim 36 (“selecting a source SuperBlock ... [based on] last modification times”).

341. Indeed, incorporating this teaching would be straightforward and obvious to try in light of the existing metadata threshold comparison taught in the combined system, whereby the controller would conduct this metadata comparison. EX1008, [0075] (“A determination is made whether the target location has an advanced state of wear. This can be carried out by comparing the WI value to a predetermined threshold T, as shown by decision step 210”), *see also, e.g.*, Claim 16 (describing Son’s threshold comparison). A POSA would understand that, in the combined system, such a threshold comparison may result in ***making the notification available to the host is dependent on the comparison.***

11. Claim 26: The SSD of claim 25 wherein comparing the metadata with at least one threshold and making the notification available to the host are unsolicited by the host

342. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Son renders this claim obvious.

343. It is my opinion that, as I described for Claim 25, the combined system of Gonzalez, Goss, Son, and NVMe1.0b teaches ***the SSD of claim 25 wherein comparing the metadata with at least one threshold.***

344. It is my opinion that, as I described for Claim 5, the combined system of Gonzalez, Goss, and NVMe1.0b teaches ***making the notification available to the host are unsolicited by the host.***

12. Claim 27: The SSD of claim 2 wherein logic further operates to, in response to a command from the host, transmit to the host

information dependent on a time period since data was written to the specified zone.

345. It is my opinion that the combined system of Gonzalez, Goss, NVMe1.0b and Son renders this claim obvious.

346. It is my opinion that, as I described for Claim 2, the combined system of Gonzalez, Goss, and NVMe1.0b teaches *the SSD of claim 2*.

347. There are two additional disclosures teaching this claim. First, Son teaches that the host may “query” SuperBlock metadata, including *time period since data was written to the specified zone*, and in response to the query command, the controller *transmits to the host the metadata*. EX1009, [0042], [0047]. Second, Goss teaches that the host may issue “a read request” directed to the metadata stored in the nonvolatile memory where that metadata is “returned to the host responsive to” the read command [*in response to a command from the host*]. EX1008, [0043]. For both teachings, a POSA would understand that in the combined system in response to the request, the controller would *transmit to the host information*, the metadata *dependent on a time period since data was written to the specified metablock*. See *supra* Claims 1, 25.

XII. GROUND VI - GONZALEZ IN VIEW OF GOSS, SON, NVME1.0B, AND CORNWELL

A. Motivation to Combine

348. A POSA would have been motivated to combine the teachings of Cornwell with the combined system of Gonzalez, Goss, Son and NVMe1.0b, specifically, to incorporate Cornwell’s teaching reaching sharing certain metadata with the host to inform further commands and operations.

349. **First**, As with all other references in this Ground, Cornwell teaches a similar technique to solve the same problem, where metadata is tracked for NAND flash memory in order to enable informed maintenance functionality and commands, and where the NAND flash memory

is architected to include groupings of physical blocks, here, termed “virtual blocks.” *E.g.*, EX1012, 1:34-3:8, 3:29-65, 4:35-39, 9:17-40.

350. **Second**, in light of the existing functionality in the combined system—incorporating teachings from Goss, Son and the NVMe1.0b regarding host memory management responsibilities and the need to share metadata with the host to enable the same—there is an existing framework and infrastructure to accommodate the teaching from Cornwell regarding the transmission of additional types of metadata to the host. For example, the transmission of bad-block metadata to the host could be accomplished in the combination system’s real-time notifications to the host regarding data status information conducted as part of the controller’s self-monitoring functionality. Adding in the additional metadata from Cornwell would be straightforward and would not require the addition of new or different functionality but rather merely augmenting the classes of data applicable to existing functions. This incorporation would have a reasonable likelihood of success.

351. The reasons for combining various references in other grounds, as relevant to broadly applicable motivations to incorporate additional forms of metadata and discrete implementations of shared host functionality set out in the other grounds, are incorporated by reference here. Further, my reasoning for motivations to combine as set out within the claim-by-claim analysis is also incorporated by reference.

B. Claim-by-claim Analysis

1. **Claim 28: The SSD of claim 2 wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone and transmit to the host the zone start logical block address and a state of the specified zone.**

352. It is my opinion that, as I described for Claim 2, the combined system of Gonzalez, Goss, and NVMe1.0b teaches *the SSD of claim 2*.

353. It is my opinion that, as I described for Claims 13 and 15, the combined system of Gonzalez, Goss, Son and NVMe1.0b teaches metadata describing **a state of the specified zone**. A POSA would understand that the notification to the host in the combined system would naturally include such information about the state of the zone subject to the maintenance request. Indeed, Goss teaches “real-time notifications to a host with regard to the “status” or the state of “data.”

354. To the extent the combined system of Gonzalez, Goss, and NVMe1.0b does not teach this limitation, a POSA would be motivated to incorporate the teaching of Cornwell into the combined system.

355. Cornwell describes the utility of the controller sending metadata to the host, consistent with the teaching in Son. Specifically, Cornwell teaches that “[m]etadata may be transmitted from a memory of a host [(a controller)] to an external host [(a host)],” where such “metadata may include bad block data, where bad block data includes a listing of blocks in the memory that do not satisfy performance criteria.” EX1012, [0004], [0006]. In order to identify, generate and report this bad block metadata, the controller in Cornwell and in the combined system must contain **logic that operates to detect a failure condition of one of the erase units**. A POSA would further understand that, in implementing this teaching from Cornwell into the combined system organized in metablocks, the failure or bad block information would correspond to an individual metablock—**detect a failure condition of one of the erase units of the specified zone**.

356. As part of the information sent to the host, Cornwell teaches that that the controller transmits to the host “mapping metadata” which corresponds to groupings of blocks, called “virtual blocks.” *Id.*, [0045] (“Metadata may include data that describes other data, such as listings of bad blocks in memory (e.g., factory specified bad blocks or other bad blocks), type of error correcting code executed on memory, write and/or erase counts, time stamps for when data is written to

memory, and/or data describing mapping between PBAs and LBAs.”). Based on this teaching and the existing teaching of the combined system, *supra* Son’s teaching in Claim 20[c], the metadata ***transmitted to the host*** would include the ***zone start logical block address***.

357. To the extent the combination of Gonzalez, Goss, Cornwell, and NVMe1.0b does not teach transmitting to the host the “state of the specified zone,” a POSA would be motivated to incorporate the teaching of the NVMe Standard which discloses this limitation, as I described with respect to Claim 1 and Claim 27.

2. **Claim 29: The SSD of claim 2 wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone and, in response to a write command from the host within the range of logical block addresses of the specified zone, transmit to the host a notification of a write failure.**

358. It is my opinion that, as I described for Claim 2, the combined system of Gonzalez, Goss, and NVMe1.0b teaches ***the SSD of claim 2***.

359. It is my opinion that, as I described in Claim 28, the combined system of Gonzalez, Goss, Son and NVMe1.0b teaches ***wherein the logic further operates to detect a failure condition of one of the erase units of the specified zone***.

Cornwell teaches:

A memory of the host may be tested to determine if bad blocks are appropriately identified (operation 525). For example, ***an external host may transmit a write and/or an erase command to the memory of the host to determine if a block is bad***.

EX1012, [0048]. Therefore, Cornwell teaches, in connection with a host-initiated write command, the controller may detect and identify a failure condition in connection with a write command. A POSA would be motivated to incorporate this teaching in light of the utility described in Cornwell and implement the suite of functionality taught by Cornwell. Moreover, in light of Cornwell’s teaching of virtual blocks, a POSA would be able to easily transpose the failure or bad block

information to the metablock (discrete ranges of LBAs) in the combination system subject to the write request—*in response to a write command from the host within the range of logical block addresses of the specified zone, transmit to the host a notification of a write failure.*

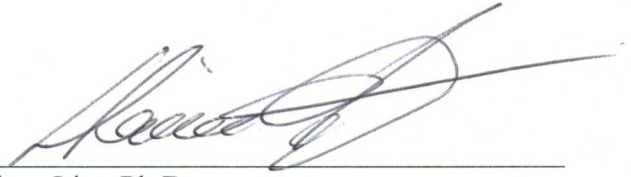
3. Claim 30: The SSD of claim 2 wherein the metadata is dependent on read errors detected for the specified zone.

360. It is my opinion that, as I described for Claim 2, the combined system of Gonzalez, Goss, and NVMe1.0b teaches *the SSD of claim 2*.

361. It is my opinion that, as I described for Claims 28 and 29, the combined system of Gonzalez, Goss, Son, Cornwell, and NVMe1.0b transmits failure condition metadata corresponding to metablocks based on certain “blocks in the memory that do not satisfy performance criteria.” EX1012, [0006]. A POSA would understand that such a performance criteria must include read errors (as a fundamental category of failure conditions typically addressed in nonvolatile memory systems)—*wherein the metadata is dependent on read errors detected for the specified zone*—expectedly, this understanding is consistent with the teaching of Cornwell. *Id.*, [0049].

I, Heiner Litz, declare under penalty of perjury under the laws of the United States that the foregoing is true and correct.

Dated: July 16, 2025

A handwritten signature in dark ink, appearing to read 'Heiner Litz', is written over a horizontal line.

Heiner Litz, Ph.D.

EXHIBIT 1

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 Ph.D. 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. Slogsnat, **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