

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 IPR2025-01266

U.S. Patent No. 11,544,183

**DECLARATION OF DR. HEINER LITZ IN SUPPORT OF PETITION
FOR INTER PARTES REVIEW OF U.S. PATENT NO. 11,544,183**

TABLE OF CONTENTS

	<u>Page</u>
I. SUMMARY OF OPINIONS	1
II. QUALIFICATIONS	1
III. MATERIALS CONSIDERED	
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 Patents	10
1. The '183 Patent	10
2. The Challenged Claims	10
V. LEVEL OF ORDINARY SKILL IN THE ART	16
VI. OVERVIEW OF PRIOR ART REFERENCES	16
1. Ellis	16
2. Son	17
3. Yamada	18
4. Sinclair	19
5. Yu	20
6. Jung	21
VII. INVALIDITY OF CLAIMS	22
A. Ground I	22
1. Motivation to combine Ellis and Son	22

2.	Claim 1[pre]: A memory controller to control flash memory, the flash memory having subdivisions, each of the subdivisions corresponding to a respective group of erase blocks, each of the erase blocks such that it must be erased before storage locations i that erase block can be reprogrammed, the memory controller comprising:	25
	(a) Claim 1[a]: a host interface;	25
	(b) Claim 1[b]: a memory interface;	25
	(c) Claim 1[c]: logic to track information indicating extent of page utilization of each of the subdivisions;	25
	(d) Claim 1[d]: logic to receive a write request from a host via the host interface, the write request accompanied by an address that designates a specific one of the subdivisions, wherein the logic to track is to, in association with execution of the write request, update the information indicating extent of page utilization of the specific one of the subdivisions;	26
	(e) Claim 1[e]: logic to send the host the information indicating extent of page utilization of the specific one of the subdivisions and an address corresponding to the specific one of the subdivisions;	28
	(f) Claim 1[f]: logic to receive an erasure request from the host, via the host interface, the erasure request accompanied by the address corresponding to the specific one of the subdivisions, to responsively control erasure of each unerased erase block in the group respective to the specific one of the subdivisions, and to update the information indicating extent of page utilization of the specific one of the subdivisions; and	29
	(g) Claim 1[g]: logic to, in connection with the erasure, detect a defect in one of the erase blocks in the group respective to the specific one of the subdivisions, to responsively substitute one or more different erase blocks of the flash memory for the one of the erase blocks for which the defect was detected	30
	(h) Claim 1[h]: wherein each said logic comprises at least one of hardware circuitry or instructions stored on non-transitory, machine-readable media that when executed are to control the operation of hardware circuitry.	30
3.	Claim 2: The memory controller of claim 1, ...	31
	(a) [2a] wherein the host interface is also to receive a data copy request from the host, ...	31

(b)	[2b] wherein the host interface is also to receive a data copy request from the host, wherein the memory controller is to responsively retrieve data from a source location in the flash memory, the source location associated with the specific one of the subdivisions, and write the retrieved data to a destination location in the flash memory, without sending the retrieved data to the host, ...	31
(c)	[2c] and wherein the logic to track is to responsively update the tracked information for one of the subdivisions which corresponds to the destination location, in association with the write of the retrieved data into the destination location.	32
4.	Claim 8: The memory controller of claim 1, wherein the memory controller further comprises logic to receive a query from the host, and wherein the logic to send the information to the host is to do so as a response to the query.	33
5.	Claim 10: The memory controller of claim 1, ...	34
(a)	wherein the memory controller further comprises logic to store erase count information for each of the subdivisions, ...	34
(b)	and to send to the host, via the host interface, the erase count information for the specific one of the subdivisions, together with the address corresponding to the specific one of the subdivisions, ...	35
(c)	and wherein the logic to store the erase count information is to update the erase count information for the specific one of the subdivisions in association with the erasure.	35
6.	Claim 11: The memory controller of claim 1, wherein the extent of the page utilization of the specific one of the subdivisions represents the amount of valid data written to the specific one of the subdivisions since the specific one of the subdivisions was last erased.	36
7.	Claim 12: The memory controller of claim 1, wherein the extent of the page utilization of the specific one of the subdivisions represents a remaining space of the specific one of the subdivisions that is available for writing without first erasing the specific one of the subdivisions.	36
8.	Claims 13, 14, 18 and 20	37
B.	Ground II	37
1.	Motivation to combine Yamada with the combined system of Ellis and Son.	37
2.	Claim 6: The memory controller of claim 2, ...	39

(a)	the memory controller further comprises logic to track metadata for each of the subdivisions, the metadata indicating a time since data was programmed, ...	39
(b)	to compare the tracked metadata to a threshold, and,.....	39
(c)	responsive to a determination that the tracked metadata for the specific one of the subdivisions satisfies the threshold, to transmit to the host the address corresponding to the specific one of the subdivisions; and the data copy request is accompanied y the address corresponding to the specific one of the subdivisions.	40
3.	Claim 9: The memory controller of claim 1, wherein:	42
(a)	the memory controller further comprises logic to assign a write destination in the flash memory, in which data accompanying the write request is to be stored, the write destination corresponding to the specific one of the subdivisions,.....	42
(b)	and to transmit to the host a read address that will be used to retrieve the data specifically from the write destination;	42
(c)	and the memory controller further comprises logic to receive, from the host, and via the host interface, a read request for the data, accompanied by the read address, and to identify both of the specific one of the subdivisions and the write destination rom the read address.	43
4.	Claims 17 and 19.....	44
C.	Ground III.....	44
1.	Motivation to Combine	44
2.	Claim 1[pre]: A memory controller to control flash memory, the flash memory having subdivisions, each of the subdivisions corresponding to a respective group of erase blocks, each of the erase blocks such that it must be erased before storage locations i that erase block can be reprogrammed, the memory controller comprising:	47
(a)	Claim 1[a]: a host interface;	47
(b)	Claim 1[b]: a memory interface;	47
(c)	Claim 1[c]: logic to track information indicating extent of page utilization of each of the subdivisions;	47
(d)	Claim 1[d]: logic to receive a write request from a host via the host interface, the write request accompanied by an address that designates a specific one of the subdivisions, wherein the logic to track is to, in association with execution	

	of the write equest, update the information indicating extent of page utilization of the specific one of the subdivisions;	48
(e)	Claim 1[e]: logic to send the host the information indicating extent of page utilization of the specific one of the subdivisions and an address corresponding to the specific one of the subdivisions;	49
(f)	Claim 1[f]: logic to receive an erasure request from the host, via the host interface, the erasure request accompanied by the address corresponding to the specific one of the subdivisions, to responsively control erasure of each unerased erase block in te group respective to the specific one of the subdivisions, and to update the information indicating extent of page utilization of the specific one of the subdivisions; and.....	50
(g)	Claim 1[g]: logic to, in connection with the erasure, detect a defect in one of the erase blocks in the group respective to the specific one of the subdivisions, to responsively substitute one or more different erase blocks of the flash memory for the one of the erase blocks for which the defect was detected	51
(h)	Claim 1[h]: wherein each said logic comprises at least one of hardware circuitry or instructions stored on non-transitory, machine-readable media that when executed are to control the operation of hardware circuitry.	51
3.	Claim 2: The memory controller of claim 1, ...	52
(a)	[2a]: wherein the host interface is also to receive a data copy request from the host, ...	52
(b)	[2b] wherein the memory controller is to responsively retrieve data from a source location in the flash memory, the source location associated with the specific one of the subdivisions, and write the retrieved data to a destination location in the flash memory, without sending the retrieved data to the host, ...	52
(c)	[2c] and wherein the logic to track is to responsively update the tracked information for one of the subdivisions which corresponds to the destination location, in association with the write of the retrieved data into the destination location.....	53
4.	Claim 8: The memory controller of claim 1, wherein the memory controller further comprises logic to receive a query from the host, and wherein the logic to send the information to the host is to do so as a response to the query.	54
5.	Claim 10: The memory controller of claim 1, ...	54

(a)	wherein the memory controller further comprises logic to store erase count information for each of the subdivisions,.....	54
(b)	and to send to the host, via the host interface, the erase count information for the specific one of the subdivisions, together with the address corresponding to the specific one of the subdivisions,.....	55
(c)	and wherein the logic to store the erase count information is to update the erase count information for the specific one of the subdivisions in association with the erasure.	56
6.	Claim 11: The memory controller of claim 1, wherein the extent of the page utilization of the specific one of the subdivisions represents the amount of valid data written to the specific one of the subdivisions since the specific one of the subdivisions as last erased.	56
7.	Claim 12: The memory controller of claim 1, wherein the extent of the page utilization of the specific one of the subdivisions represents a remaining space of the specific one of the subdivisions that is available for writing without first erasing the specific one of the subdivisions.	57
8.	Claims 13, 14, 17-20	57
D.	Ground IV	57
1.	Motivation to Combine	57
2.	Claim 6: The memory controller of claim 2, ...	60
(a)	[6a] the memory controller further comprises logic to track metadata for each of the subdivisions, the metadata indicating a time since data was programmed, ...	60
(b)	[6b] to compare the tracked metadata to a threshold,.....	60
(c)	[6c] and, responsive to a determination that the tracked metadata for the specific one of the subdivisions satisfies the threshold, to transmit to the host the address corresponding to the specific one of the subdivisions; and the data copy request is accompanied by the address corresponding to the specific one of the subdivisions.	61
3.	Claim 9: The memory controller of claim 1, wherein:	64
(a)	[9a] the memory controller further comprises logic to assign a write destination in the flash memory, in which data accompanying the write request is to be stored, the write destination corresponding to the specific one of the subdivisions,.....	64

(b)	[9b] and to transmit to the host a read address that will be used to retrieve the data specifically from the write destination;	64
(c)	[9c] and the memory controller further comprises logic to receive, from the host, and via the host interface, a read request for the data, accompanied by the read address, and to identify both of the specific one of the subdivisions and the write destination from the read address.	65
4.	Claims 17 and 19.....	66

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. 11,544,183 (the “’183 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 ’183 patent. Specifically, I have been asked to provide my opinion regarding whether claims 1, 2, 6, 8-14, 17-20 of the ’183 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-IV 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 analyze 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 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. To safely erase a unit, 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. For example, if no subdivision or 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, erasure can be done safely. 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.

34. To optimize garbage collection, the SSD maintains metadata to track the number of valid blocks per zone. Selecting a zone 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 zones. This increases the likelihood of finding hot erase units with few valid blocks. One implementation approach is to maintain multiple open units to which writes can be applied: one for cold LBAs and one for hot LBAs. More

than two units may be maintained, with write frequency tracked using finer-grained *temperature* metrics.

35. 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 units, enabling those blocks to be erased and reused, thereby balancing wear across all units in the SSD.

36. 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 erase units by migrating cold data.
- Erase Management: Erasing only those units that no longer contain valid data.

37. 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).

38. 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.

39. 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.” EX1010, Abstract.

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

B. Summary of Patents

1. The ’183 Patent

41. The ’183 Patent issued January 3, 2023, and is titled “Nonvolatile Memory Controller Host-Issued Address Delimited Erasure and Memory Controller Remapping of Host-Address Space for Bad Blocks.”

2. The Challenged Claims

42. Below is a table described the Challenged Claims, including claims 1, 2, 6, 8-14, and 17-20.

Claim 1	
1[pre]	A memory controller to control flash memory, the flash memory having subdivisions, each of the subdivisions corresponding to a respective group of erase blocks, each of the erase blocks such that it must be erased before storage locations in that erase block

	can be reprogrammed, the memory controller comprising:
1[a]	a host interface;
1[b]	a memory interface;
1[c]	logic to track information indicating extent of page utilization of each of the subdivisions;
1[d]	logic to receive a write request from a host via the host interface, the write request accompanied by an address that designates a specific one of the subdivisions, wherein the logic to track is to, in association with execution of the write request, update the information indicating extent of page utilization of the specific one of the subdivisions;
1[e]	logic to send the host the information indicating extent of page utilization of the specific one of the subdivisions and an address corresponding to the specific one of the subdivisions
1[f]	logic to receive an erasure request from the host, via the host interface, the erasure request accompanied by the address corresponding to the specific one of the subdivisions, to responsively control erasure of each unerased erase block in the group respective to the specific one of the subdivisions, and to update the information indicating extent of page utilization of the specific one of the subdivisions; and
1[g]	logic to, in connection with the erasure, detect a defect in one of the erase blocks in the group respective to the specific one of the subdivisions, to responsively substitute one or more different erase blocks of the flash memory for the one of the erase blocks for which the defect was detected
1[h]	wherein each said logic comprises at least one of hardware circuitry or instructions stored on non-transitory, machine-readable media that when executed are to control

	the operation of hardware circuitry.
Dependent Claims	
2	The memory controller of claim 1, wherein the host interface is also to receive a data copy request from the host, wherein the memory controller is to responsively retrieve data from a source location in the flash memory, the source location associated with the specific one of the subdivisions, and write the retrieved data to a destination location in the flash memory, without sending the retrieved data to the host, and wherein the logic to track is to responsively update the tracked information for one of the subdivisions which corresponds to the destination location, in association with the write of the retrieved data into the destination location.
6	The memory controller of claim 2, wherein: the memory controller further comprises logic to track metadata for each of the subdivisions, the metadata indicating a time since data was programmed, to compare the tracked metadata to a threshold, and, responsive to a determination that the tracked metadata for the specific one of the subdivisions satisfies the threshold, to transmit to the host the address corresponding to the specific one of the subdivisions; and the data copy request is accompanied by the address corresponding to the specific one of the subdivisions
8	The memory controller of claim 1, wherein the memory controller further comprises logic to receive a query from the host, and wherein the logic to send the information to the host is to do so as a response to the query.
9	The memory controller of claim 1, wherein: the memory controller further comprises logic to assign a write destination in the flash memory, in which data accompanying the write request is to be stored, the write destination corresponding to the specific one

	of the subdivisions, and to transmit to the host a read address that will be used to retrieve the data specifically from the write destination; and the memory controller further comprises logic to receive, from the host, and via the host interface, a read request for the data, accompanied by the read address, and to identify both of the specific one of the subdivisions and the write destination from the read address.
10	The memory controller of claim 1, wherein the memory controller further comprises logic to store erase count information for each of the subdivisions, and to send to the host, via the host interface, the erase count information for the specific one of the subdivisions, together with the address corresponding to the specific one of the subdivisions, and wherein the logic to store the erase count information is to update the erase count information for the specific one of the subdivisions in association with the erasure.
11	The memory controller of claim 1, wherein the extent of the page utilization of the specific one of the subdivisions represents the amount of valid data written to the specific one of the subdivisions since the specific one of the subdivisions was last erased.
12	The memory controller of claim 1, wherein the extent of the page utilization of the specific one of the subdivisions represents a remaining space of the specific one of the subdivisions that is available for writing without first erasing the specific one of the subdivisions.
Claim 13	
13[pre]	A method of operation in a memory controller to control flash memory, where the flash memory has subdivisions, each of the subdivisions corresponding to a respective

	group of erase blocks, each of the erase blocks such that it must be erased before storage locations in that erase block can be reprogrammed, the memory controller having a host interface, a memory interface and at least one of hardware circuitry or instructions stored on non-transitory, machine readable media that, when executed, are to control the operation of hardware circuitry, the method comprising:
13[a]	tracking information indicating extent of page utilization of each of the subdivisions;
13[b]	receiving a write request from a host via the host interface, the write request accompanied by an address that designates a specific one of the subdivisions, and, in association with execution of the write request, updating the information indicating extent of page utilization of the specific one of the subdivisions;
13[c]	sending the host the information indicating extent of page utilization of the specific one of the subdivisions and an address corresponding to the specific one of the subdivisions;
13[d]	receiving an erasure requests from the host, via the host interface, the erasure request accompanied by the address corresponding to the specific one of the subdivisions, responsively controlling erasure of each unerased erase block in the group respective to the specific one of the subdivisions, and updating the information indicating extent of page utilization of the specific one of the subdivisions; and
13[e]	in connection with the erasure, detecting a defect in one of the erase blocks in the group respective to the specific one of the subdivisions, responsively substituting one or more different erase blocks of the flash memory for the one of the erase blocks for which the defect was detected.

Dependent Claims	
14	The method of claim 13, further comprising also receiving via the host interface a data copy request from the host and responsively retrieving data from a source location in the flash memory, the source location associated with the specific one of the subdivisions, and writing the retrieved data which to a destination location in the flash memory, without sending the data which has been read to the host, and responsively updating the tracked information for the one of the subdivisions which corresponds to the destination location, in association with the write of the retrieved data into the destination location.
17	The method for claim 14, wherein: the method further comprises tracking metadata for each of the subdivisions, the metadata indicating a time since data was programmed, comparing the tracked metadata to a threshold, and, responsive to a determination that the tracked metadata for the specific one of the subdivision satisfies the threshold, transmitting to the host the address corresponding to the specific one of the subdivisions; and the data copy request is accompanied by the address corresponding to the specific one of the subdivisions.
18	The method of claim 13, further comprising receiving a query from the host, and sending the information to the host as a response to the query.
19	The method of claim 13, wherein: the method further comprises assigning a write destination in the flash memory, in which data accompanying the write request is to be stored, the write destination corresponding to the specific one of the subdivisions, and transmitting to the host a read address that will be used to retrieve the data specifically from the write destination; and the method further comprises receiving,

	from the host, and via the host interface, a read request for the data, accompanied by the read address, and identifying both of the specific one of the subdivisions and the write destination from the read address.
20	The method of claim 13, wherein the method further comprises: storing erase count information for each of the subdivisions; sending to the host, via the host interface, the erase count information for the specific one of the subdivisions, together with the address corresponding to the specific one of the subdivisions; and updating the erase count information for the specific one of the subdivisions in association with the erasure.

V. LEVEL OF ORDINARY SKILL IN THE ART

43. The level of ordinary skill in the art would be a person with had at least a Bachelor of Science degree in electrical engineering, or an equivalent field, and at least three years of experience with SSDs or other flash memory devices.

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

VI. OVERVIEW OF PRIOR ART REFERENCES

1. Ellis

45. U.S. Patent 9,239,781 (“Ellis”) was filed October 10, 2012, and I understand that is prior art under at least pre-AIA 35 U.S.C. § 102(e).

46. Ellis teaches a NAND flash memory system comprised of superblocks—each superblock is a group of erase blocks. EX1004, 6:43-65, Fig. 5. Memory is managed at the superblock level where “host writes, recycling operations, or erasing of blocks can be performed on one of the super blocks as a single unit.” *Id.*, 8:2-7. To facilitate this management, Ellis teaches that the controller tracks metadata for each superblock, including, *e.g.*, “valid-page count,”

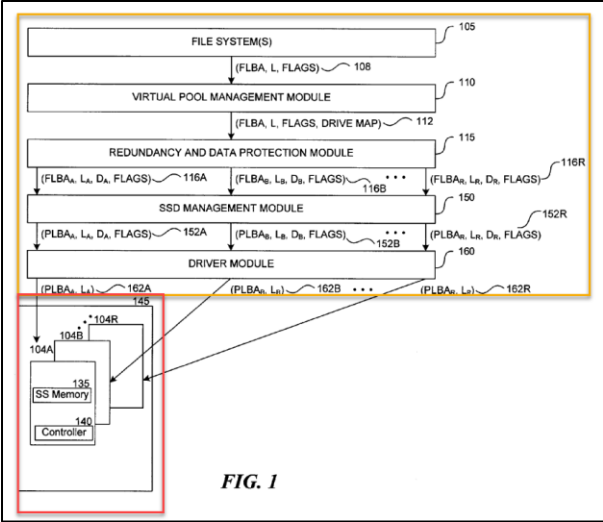
defined as the “number of pages that are valid in each of the super blocks 502.” *Id.*, 11:58-12:2; 12:37-41. This metadata and other “control and status information generated by the memory controller 104” may be sent “to the host system 108.” *Id.*, 4:51-53. This information is sent to the host because the host shares memory management responsibilities with the controller. *Id.*, 4:58-65 (“the memory controller [or] the host system ... can operate any of the modules and functions of the memory controller”).

2. Son

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

48. 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.” EX1005, [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.*

49. 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.



50. Son’s system, like Ellis, is comprised of superblocks which are “a grouping of ... blocks.” *Id.*, [0029]. Like Ellis, Son teaches that memory management is done at the superblock level where commands are directed at superblocks using a “superblock address.” *Id.*, [0029]; [0038]. Like Ellis, Son describing 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
	...	...	...	...	...	...	...	...	...	...

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

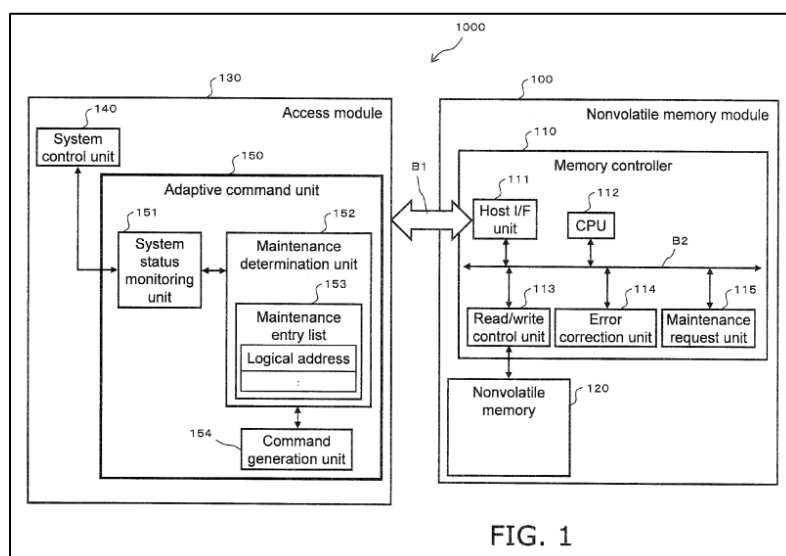
3. Yamada

51. 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 pre-AIA 35

U.S.C. § 102(a) and (b).

52. Like Ellis and Son, Yamada teaches that memory management responsibilities are shared between the host and controller. Yamada describes particular implementations of such shared responsibility, *e.g.*, where copy operations for maintenance are recommended by the controller and commanded by the host.

53. Specifically, Yamada describes a controller including a “maintenance request unit,” and a host – termed access module – including a “maintenance determination unit.” EX1006, [0012], [0017], [0031].



Yamada describes that the host analyzes the metadata of blocks or groups of blocks to determine if certain thresholds have been met. *Id.*, [0013], [0032], [0080]-[0081]. If so, the controller transmits the metadata and address information to the host in “maintenance request” from the “maintenance request unit.” *Id.*, [0014], [0109]-[0116]. In response, the host may then issue a “maintenance command,” commanding the controller to copy data from the block or blocks that need maintenance to new block(s). *Id.*

4. Sinclair

54. U.S. Pat. No. 7,739,444 (Sinclair) was filed December 26, 2006, published June 26,

2008, and granted June 15, 2010. I understand that Sinclair is prior art under at least pre-AIA 35 U.S.C. §§ 102(a) and (b).

55. Similar to the superblocks in Son and Ellis, Sinclair teaches a system, and describes the benefits of, organizing memory into “metablocks,” which include “four blocks” “from each plane [are] logically linked together to form a metablock.” EX1007, 9:52-57, 7:2-4. Also similar to Son and Ellis, Sinclair teaches means to manage metablocks by storing and collecting metadata for each of the metablocks, *id.*, 31:65-32:4, such as “[t]he number of pages” or “the number of pages of unprogrammed capacity” for each metablock, *id.*, 26:5-17.

56. Similar to Ellis, Son, and Yamada, Sinclair teaches that the metablocks should be managed with shared host and controller responsibility where information about the memory is provided to the host. Sinclair describes the need for the controller to share with the host information such as the physical characteristics of the metablocks, *e.g.*, “[t]he number of pages per metablock; and [] [t]he lowest logical address that is mapped to a first page of a physical metablock.” *Id.*, 18:16-28.

5. Yu

57. U.S. Pat. Pub. 2012/0284587 (Yu) was filed July 2, 2012, and published November 8, 2012. I understand that Yu is prior art under at least pre-AIA 35 U.S.C. §§ 102(a) and 102(e).

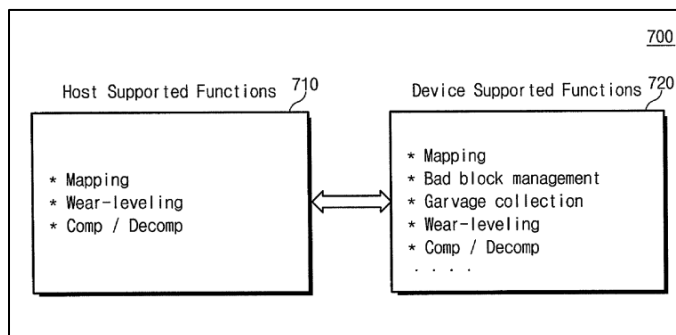
58. Like Ellis, Son, Yamada, and Sinclair, Yu describes additional implementations of memory systems where the host and controller share responsibility for memory management and work cooperatively to optimize memory storage and to prolong longevity. As a discrete example of this shared responsibility, Yu teaches the use of “Self-Monitoring, Analysis and Reporting Technology” (“S.M.A.R.T”) functions, where the controller sends the host information about the memory systems, and the host “monitor[s] and control[s] error correction, wear, bad block, and other flash memory management.” EX1008, [0064], [0228].

59. Like Ellis, Son, and Sinclair, Yu describes the tracking and storage of metadata, including page utilization through maintenance of a “page status table,” and other information describing block-level usage, *e.g.*, (“erase counters and block status”). *Id.*, [0083].

6. Jung

60. U.S. Pat. Pub. No. 2008/0189485 (Jung) was filed August 3, 2007, and published August 7, 2008. I understand that Jung is prior art under at least pre-AIA 35 U.S.C. §§ 102(a) and (b).

61. Tightly matching the disclosure in the '183 Patent, Jung teaches a system of “Cooperative memory management,” where memory management responsibilities are overlapping or shared between the host and controller. EX1010, [0044] (“the host and the device may have one or more overlapping functions, so it remains to be determined which will actually implement each of the overlapping functions.”). An example of the shared host and controller responsibilities is reproduced in the below figure.



Id., Fig. 7.

62. Jung describes the need for the controller to share broad categories of information with the host to enable proper memory management. For example, Jung describes the need for the controller to “load[] meta information, such as an erase count, garbage collection data and a mapping table” to the host. *Id.*, [0038].

VII. INVALIDITY OF CLAIMS

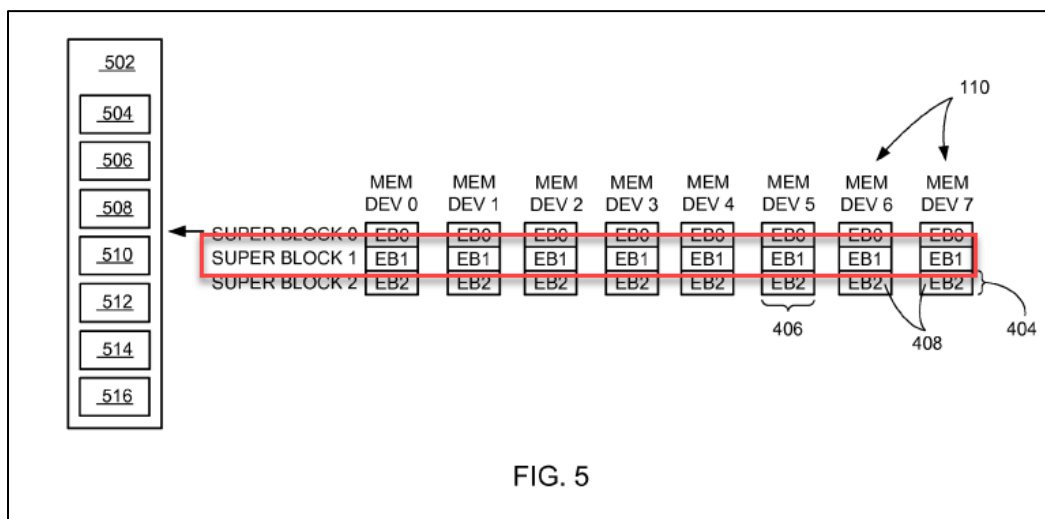
A. Ground I

1. Motivation to combine Ellis and Son

63. A POSA would be motivated to combine the teachings of Son with the system disclosed in Ellis.

64. First, both systems describe means to improve and optimize SSD performance in NAND systems through the use of “superblocks” as the unit of memory. EX1004, 7:32-45; EX1005, [0002], [0014].

65. Ellis teaches that the memory device may include “a number of the super blocks 502 of FIG. 5, each of which can include a number of erase blocks 408 (EB). The super blocks 502 are defined as logical groupings of the erase blocks 408[.]” EX1004, 6:43-49. The Superblocks are illustrated in Fig. 5, I have included an annotation emphasizing a single superblock.



66. Son also teaches the use of superblocks as the unit of memory, EX1005, [0029] (“Typically, the controller permits erasures (or “flashes”) to occur only at the superblock-level, and does not permit erasures at the level of a superblock’s constituent individual physical flash

blocks.”), and further describes means to designate and identify superblocks through use of superblock addresses, *id.* (“Each superblock is a grouping of one or more flash blocks that the controller associates with a unique superblock address; the constituent flash blocks in a superblock are typically physically contiguous.”).

67. Second, both systems describe the use of superblock metadata to manage the superblocks, including page utilization, to optimize memory management.

68. Ellis describes the utility of superblock metadata, including, *e.g.*, the consolidation of metadata for memory management purposes, where that metadata includes “statistics [such as] program-erase count 510 and a valid-page count 512.” EX1004, 11:58-12:23. Ellis states that “valid-page count 512 is defined as a numerical value indicating a number of pages that are valid in each of the super blocks 502.” *Id.*

69. Son describes a diverse set of metadata maintained for each superblock and included in the “Superblock table,” as illustrated in Fig. 3C, below. The superblock metadata includes “valid page count (column 335 c), an invalid page count (column 335 d), ... [and] a next page indicator (column 335 g).” EX1005, [0039].

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

Son describes the use of the above superblock metadata for wear-leveling operations, write operations and other maintenance activities. *E.g.*, *id.*, [0078]-[0080], Claims 32 and 40, [0017], [0046].

70. Third, both systems teach that memory management responsibility may be distributed between the controller and the host.

71. Ellis teaches that “the host system ... can operate any of the modules and functions of the memory controller.” EX1004, 4:58-65.

72. Son describes the need for “moving solid-state management functions above the firmware level” to the host. EX1005, [0012]-[0014]. By moving these functions to the host, and incorporating controller information (*e.g.*, regarding physical storage) into the host system, there are “various advantages over conventional systems, such as enhanced I/O performance, simplified SSD firmware, and extended SSD lifespan.” *Id.*, [0014]. This is a clear teaching to a POSA regarding the benefits of shared information and memory management responsibilities between the host and controller.

73. In light of the core similarities in the systems in Ellis and Son, a POSA would be motivated to combine the teaching of Son with the system in Ellis, including the forms of superblock metadata described in Son, as well as the sharing of information with the host and the distributed host level responsibility taught by Son.

74. A POSA would be able to apply these teachings in Son, such as additional variants of metadata, to the system in Ellis—which already includes means to track and store certain superblock metadata—in a straightforward manner using typical techniques. Indeed, in light of the reliance on certain superblock metadata in Ellis, it would have been obvious to a POSA to use the other forms of superblock metadata described in Son.

75. Additionally, given Ellis’s teaching that memory management functions done by the controller may be moved to the host, a POSA would make use of known techniques in the art to relocate the memory management functions to the host consistent with how those functionalities are described in Son—Son provides the motivation to move these functions to the host. A POSA

would merely need to transfer the functionality embodied in the firmware to the host's software. Such relocation of functionality is commonplace, for instance, in software defined networking (SDN) the approach of moving firmware functionality to host software has been applied to network switches. *See generally*, EX1011 (Nick McKeown et al, OpenFlow: Enabling Innovation in Campus Networks), page 1.

2. Claim 1[pre]: A memory controller to control flash memory, the flash memory having subdivisions, each of the subdivisions corresponding to a respective group of erase blocks, each of the erase blocks such that it must be erased before storage locations in that erase block can be reprogrammed, the memory controller comprising:

76. Ellis teaches a “memory controller” that “control[s] the memory array” which is comprised of “flash memory.” EX1004, 3:17-24. The flash memory has subdivisions, called “superblocks,” where each superblock is a group of “erase blocks.” EX1004, 6:43-49. A POSA would understand that the NAND erase blocks cannot be over-written or re-programmed without being erased first, as this is a fundamental property of NAND.

77. It is my opinion that Ellis teaches the preamble.

(a) Claim 1[a]: a host interface;

78. The system in Ellis includes “a host interface unit.” EX1004, 3:38-45.

79. It is my opinion that Ellis renders this limitation obvious.

(b) Claim 1[b]: a memory interface;

80. The system in Ellis includes “a memory interface unit.” EX1004, 3:38-45.

81. It is my opinion that Ellis renders this limitation obvious.

(c) Claim 1[c]: logic to track information indicating extent of page utilization of each of the subdivisions;

82. Ellis teaches that the controller may contain “logic” and “software.” EX1004, 3:38-50.

83. Ellis teaches metadata representing the “number of pages that are valid in each of

the super blocks” is tracked by logic of the controller. EX1004, 3:38-45, 11:58-12:2. This metadata is “tracked” because it is constantly updated after “performing an operation on the superblock.” EX1004, Claims 1 and 5, 20:11-20 (“updating metadata” such as “valid-page count in ... the super blocks.”).

84. A POSA would understand that tracking the number of valid pages in the superblock is “indicating the extent of page utilization of each of the subdivisions”, as utilization may be represented as, *i.e.*, “the extent to which pages of a given EU are valid (in use) or are stale or not in use.” EX1001, 16:54-57. This metadata is tracked throughout operation, because Ellis teaches that the controller “updat[es] metadata” values after “performing an operation on the superblock.” EX1004, Claim 1, 20:11-20 (“updating metadata” such as “valid-page count in ... the super blocks.”).

85. To the extent Ellis does not render this limitation obvious, Son also teaches superblock metadata including “free page count, valid page count, invalid page count” and “a next page indicator,” where this information is updated following operations such as writes. EX1005, [0039], [0050]-[0052], [0065]. A POSA would understand this tracking, *i.e.*, valid page count, in the superblock is “indicating the extent of page utilization of each of the subdivisions.”

86. It is my opinion the combined system of Ellis and Son renders this limitation obvious.

(d) Claim 1[d]: logic to receive a write request from a host via the host interface, the write request accompanied by an address that designates a specific one of the subdivisions, wherein the logic to track is to, in association with execution of the write request, update the information indicating extent of page utilization of the specific one of the subdivisions;

87. Ellis teaches that the host may issue a write request to “be performed on a set of the erase blocks included within one of the super blocks 502 as a single unit.” EX1004, 7:64-8:7. A

POSA would understand that for the host to issue a write request to a superblock, the controller must have logic to receive the same.

88. Ellis teaches the use of “super block identification (ID) values.” EX1004, 10:27-32. A POSA would understand that the superblock ID value needs to be included in or associated with the address in a write request to designate a superblock.

89. After the controller executes the write request—or any other operation—from the host to a superblock, the metadata for page utilization is updated. *Id.*, 20:11-20; Claim 1 (“...performing an operation on the super block; and updating metadata corresponding to the super block...”).

90. To the extent Ellis does not render this limitation obvious, a POSA would be motivated to combine its teaching with Son which provides a more detailed disclosure describing a host request to write to a superblock where the write request includes an assigned file system logical block address (“FLBA”) which designates a particular a “superblock address.” EX1005, [0023], [0029], [0038]. Note, a POSA would understand that an FLBA is a type of, or higher order formulation of, an LBA. A POSA would be able to apply Son’s teaching regarding FLBAs to the LBAs disclosed in Ellis without difficulty.

91. Following the write to the “superblock address,” the page utilization metadata is updated. EX1005, [0050], [0051]-[0052] (“updates other entries in the mapping table, superblock table, and the page status table to reflect the changes to all of the pages that are implicated by the write operation”). In the combined system the additional superblock metadata described in Son would also be updated by the controller—in Ellis, the superblock metadata is updated by the controller. EX1004, Claim 1, Claim 12.

92. It is my opinion that Ellis and the combined system of Ellis and Son renders this limitation obvious.

- (e) **Claim 1[e]: logic to send the host the information indicating extent of page utilization of the specific one of the subdivisions and an address corresponding to the specific one of the subdivisions;**

93. Ellis teaches that the controller may send “control and status information generated by the memory controller 104 to the host system 108.” EX1004, 4:51-53. The metadata pages including the “status” information may include “health and use” information for each super block. EX1004, 12:37-41. By way of example, Ellis provides illustrative examples where the controller sends “metadata pages” to the host including metadata describing the “state of data stored within the one of the superblocks” *id.*, 12:34-37. A POSA would understand that “status information,” *i.e.*, the state of the units of memory in the SSD may include information such as the “valid-page count” for metablocks. *Id.*, 11:58-62.

94. In order to send the host information indicating the valid state of one of the superblocks, the information shared with the host must include super block identification (ID) values. Showing just this, Ellis describes examples where the controller sends this mapping information “to restore a host logical to physical mapping of data and a valid state of data stored within the one of the superblocks.” *Id.*, 12:34-41.

95. In sum, Ellis teaches that the controller has logic to send to the host health and use status information, identifying an address of a superblock and providing its page utilization.

96. To the extent Ellis does not render this limitation obvious, Son provides further teaching regarding the need to share such information with the host and its utility. Son teaches that metadata, such as page utilization, is shared with the host (SSD module) to perform operations such as “garbage collection or wear leveling,” where that information is used for the identification and implementation of copying data from certain blocks which need maintenance. EX1005, [0065], [0078]-[0080], [0017], [0039], [0047], Fig. 3(c), Claims 32 and 40. A POSA therefore

would be motivated to incorporate this teaching into the system in Ellis.

97. It is my opinion that the combined system of Ellis and Son renders this limitation obvious.

(f) **Claim 1[f]: logic to receive an erasure request from the host, via the host interface, the erasure request accompanied by the address corresponding to the specific one of the subdivisions, to responsively control erasure of each unerased erase block in the group respective to the specific one of the subdivisions, and to update the information indicating extent of page utilization of the specific one of the subdivisions; and**

98. Ellis teaches that the controller may erase superblocks as a single unit, where the superblock to be erased is identified based on a superblock identifier. EX1004, 7:64-8:7, 10:27-32. During erasure, “all the erase blocks 408 in each of the super blocks 502 can be erased at the same time.” *Id.*, 17:31-36. A POSA would understand that, in order to direct an erasure to a superblock as a single unit, the superblock ID value needs to be incorporated into the address for the erasure request.

99. Following erasure of the superblock, the page utilization is updated. *Id.*, Claim 1, 20:11-20.

100. To the extent Ellis does not render this limitation obvious, a POSA would be motivated to incorporate Son’s detailed description of the means to execute superblock erasures commanded by the host. Son teaches that “erasures” are carried out by the controller “at the superblock-level.” EX1005, [0029]. Son describes the “process for a delete operation,” where the host issues “an explicit command” for erasure of a superblock using a “superblock address,” where following that host-initiated erasure page utilization is updated. *Id.*, [0029], [0038], [0060].

101. Son provides additional disclosures regarding host-initiated erasure requests and the updating of page utilization and related metadata:

In some embodiments, to reflect the erasure, the SSD module modifies the entry in the superblock table corresponding to the source superblock as follows: (a) setting state to CLEAN, (b) zeroing out the valid page count, invalid page count, next superpage, and next page entries, (c) incrementing the wear count to reflect the erasure, (d) deleting any page detail array, (e) and updating the last modified time. In the page status table the SSD module may also mark all pages in the source superblock as having a free status.

Id., [0074]-[0075]; *see also id.*, [0064].

102. It is my opinion that the combined system of Ellis and Son renders this limitation obvious.

(g) Claim 1[g]: logic to, in connection with the erasure, detect a defect in one of the erase blocks in the group respective to the specific one of the subdivisions, to responsively substitute one or more different erase blocks of the flash memory for the one of the erase blocks for which the defect was detected

103. Ellis describes the identification of defects and the substitution of new blocks during erasure. Specifically, Ellis describes the “substitution of one physical erase block for another,” EX1004, 11:44-45, where the block is identified as “bad” or “unstable” “during normal operation of the storage device,” such as erasure. *Id.*, 11:16-44, Claim 5. *See also id.*, 3:34-36 (describing “operation[s] including reading, writing, or erasing”), 19:66-20:4.

104. It is my opinion that Ellis renders this limitation obvious.

(h) Claim 1[h]: wherein each said logic comprises at least one of hardware circuitry or instructions stored on non-transitory, machine-readable media that when executed are to control the operation of hardware circuitry.

105. Ellis teaches a system including a memory controller with hardware circuitry and instructions stored on non-transitory media used to control the hardware circuitry. *See, e.g.*, EX1004, 2:24-25, 3:3-9, 3:62-4:3.

106. It is my opinion that Ellis renders this limitation obvious.

3. **Claim 2: The memory controller of claim 1,**

(a) [2a] wherein the host interface is also to receive a data copy request from the host,

107. Ellis teaches a “recycle operation” where data from a source superblock is read and then copied to a target superblock. EX1004, 19:4-15, 8:67-9:3 (discussing “host read and host write (or recycle) operations”). A POSA would understand that a command for a recycle operation is a type of “data copy request.” A POSA would therefore understand that a host recycle command is a data copy request from the host.

108. Because Ellis teaches that the recycle functionality carried out by the controller can be implemented by the host or a “combination” of the host and controller, *id.*, 20:40-44, 4:58-65, a POSA would be motivated to incorporate Son’s teaching regarding a host-initiated copy command directed to a superblock. Son teaches that “[the host] instructs the controller to copy the contents of all pages in the target superblock over to the fresh superblock, erases the target superblock and updates the data structures [including page utilization in the Superblock table] to reflect the movement and erasure.” EX1005, [0085]. A POSA would therefore understand that Son also teaches a data copy request from the host.

109. It is my opinion that the combination of Ellis and Son renders this limitation obvious.

(b) [2b] wherein the host interface is also to receive a data copy request from the host, wherein the memory controller is to responsively retrieve data from a source location in the flash memory, the source location associated with the specific one of the subdivisions, and write the retrieved data to a destination location in the flash memory, without sending the retrieved data to the host,

110. In Ellis’s host initiated recycle operation, “valid pages can be read from a donor super block and written to a new open super block.” EX1004, 19:4-15. A POSA would understand

that “reading” data from a source superblock is the same as “responsively retriev[ing] data.”

111. The reading and writing of data is carried out by the controller—specifically, the controller’s “recycle-write module 908” and “recycle-read module” in the “memory access module”—and as a result the data read from target is not sent to the host, but rather directly written by the controller to the target location. *Id.*, 18:15-34, 4:66-5:3 (“memory controller 104 can include a memory access module”), 11:51-53 (“recycling operation performed by the memory access module”), 3:39-42.

112. To the extent Ellis does not render this limitation obvious, a POSA would be motivated to incorporate Son’s teaching that the host copy command includes a source and a destination location. EX1005 [0065] (“the SSD module identifies a source superblock and target superblocks”). In order to complete the copying of data from a source superblock to a “fresh superblock,” the controller responsively reads data from the source superblock “using firmware calls [in the controller] to move the data within the drive.” EX1005, [0068]. Son’s disclosure that the copying is done by the firmware, which a POSA would understand to be done by the controller’s firmware, is a clear teaching that the copy operation is completed entirely by the controller without sending data to the host.

113. It is my opinion that the combination of Ellis and Son renders this limitation obvious.

(c) [2c] and wherein the logic to track is to responsively update the tracked information for one of the subdivisions which corresponds to the destination location, in association with the write of the retrieved data into the destination location.

114. As I described for Claim 1[d], Ellis and Son teach updating “tracked information” for write commands. A copy operation includes writing data to a destination location. Therefore, Ellis teaches the updating of metadata in connection with writing the data from the source location

to the destination location.

115. To the extent Ellis does not render this limitation obvious, Son provides an explicit teaching that the destination location metadata is updated following writing in connection with a copy command. EX1005, [0069] (“As another example, the SSD module may update the information in the superblock table associated with the target superblock and the source superblock to indicate the target location was written with valid data and the source page is now invalid.”).

116. It is my opinion that the combined system of Ellis and Son renders this limitation obvious.

4. Claim 8: The memory controller of claim 1, wherein the memory controller further comprises logic to receive a query from the host, and wherein the logic to send the information to the host is to do so as a response to the query.

117. As described with respect to claim 1, Ellis teaches a system where the controller may send “status information generated by the memory system to the host system.” EX1004, 4:51-57.

118. A POSA would understand that the host may send a command or query for status information to the controller. A POSA would have been aware of industry standards, including for example, the base NVMe specification which describes a Host Command for “Get Log Page,” to which the controller provides status information such as, “SMART / Health information and Error information.”

119. As another example of teaching a host query for information from the controller, the host may issue a read command directed to metadata pages containing information about the superblock. EX1004, 12:31-33 (“The metadata pages 508 that are held within one of the super blocks 502 can represent metadata specific to the one of the super blocks 502”), 12:14-25, 3:22-28 (“The host system 108 can request the memory controller 104 for reading... data from or to a

logical address space of a storage device.”), 12:42-50.

120. To the extent Ellis does not render this limitation obvious, Son renders this limitation obvious—the host may “query[] the SSD controller” for information where that information is stored in flash memory. EX1005, [0047], [0092], [0034]. In light of the distributed functionality of the combined Ellis and Son system, incorporating this functionality from Son would be obvious and straightforward.

121. It is my opinion that the combined system of Ellis and Son renders this claim obvious.

5. Claim 10: The memory controller of claim 1,

(a) wherein the memory controller further comprises logic to store erase count information for each of the subdivisions,

122. As described for Claim 1, Ellis teaches metadata that may be tracked, shared, and updated, like page utilization.

123. Ellis also teaches tracked, shared and updated metadata, where that metadata is “erase count.” Ellis teaches a controller that tracks metadata including “program-erase count ... defined as a numerical value indicating a number of times or cycles the memory devices 110 associated with the corresponding instances of the super blocks 502 are programmed or erased.” EX1004, 11:58-67. *See also id.*, 13:14-24, claim 1.

124. Additionally, Son teaches “wear count”—erase count—metadata for each superblock that may be tracked, shared, and updated, like page utilization in claim 1. Son defines wear count metadata as “how many times a particular superblock has been erased,” and explicitly teaches that this metadata is updated on erasure. EX1005, [0039], [0080].

125. It is my opinion that the combined system of Ellis and Son renders this limitation obvious.

- (b) **and to send to the host, via the host interface, the erase count information for the specific one of the subdivisions, together with the address corresponding to the specific one of the subdivisions,**

126. As described with respect to claim 1, the combination of Ellis and Son describes a controller which tracks superblock metadata and transmits that metadata—*e.g.*, control and status information—to the host with an address identifying the relevant superblock.

127. Ellis discloses that the controller has logic to send “control and status information generated by the memory controller 104 to the host system 108,” EX1004, 4:51-53. A POSA would understand that control and status information include “NAND statistics” and metadata such as “program-erase count” for each superblock. *Id.*, 10:27-32, 11:58-67. Therefore, Ellis teaches that the controller may send the host “program-erase count.”

128. In order for the erase count information sent to the host to be useful, *e.g.*, with respect to maintenance activities, the erase count information must be sent with a “superblock ID value” or other address designating the corresponding superblock. EX1005, [0065], [0039].

129. To the extent Ellis does not teach this limitation, Son teaches that the host may retrieve information by “*querying the SSD controller*,” such as “*wear count*” and “*superblock address*.” EX1005, [0047], [0092].

130. It is my opinion that the combined system of Ellis and Son renders this limitation obvious.

- (c) **and wherein the logic to store the erase count information is to update the erase count information for the specific one of the subdivisions in association with the erasure.**

131. Ellis teaches that the controller has logic to update erase count “each time [the superblocks] are accessed for a program operation or an erase operation by the memory access module 308.” EX1004, 13:14-25.

132. Son teaches the same; to “reflect [an] erasure of the [a] superblock, the SSD module modifies the entry in the superblock table ... incrementing the wear count to reflect the erasure.” EX1005, [0083]. Thus, the combined system of Ellis and Son thus render this limitation obvious.

133. It is my opinion that the combined system of Ellis and Son renders this limitation obvious.

6. Claim 11: The memory controller of claim 1, wherein the extent of the page utilization of the specific one of the subdivisions represents the amount of valid data written to the specific one of the subdivisions since the specific one of the subdivisions was last erased.

134. As described for claim 1, Ellis and Son both teach tracking page utilization where page utilization, including page utilization represented by the number of “valid pages,” and therefore valid data, written.

135. As described in Claim 10, both Ellis and Son teach tracking of erase count.

136. A POSA would understand based on those two classes of metadata, the tracked page utilization metadata for any superblock with an erase count of 1 or greater, represents the amount of valid data written since last erased.

137. It is my opinion that the combined system of Ellis and Son renders this claim obvious.

7. Claim 12: The memory controller of claim 1, wherein the extent of the page utilization of the specific one of the subdivisions represents a remaining space of the specific one of the subdivisions that is available for writing without first erasing the specific one of the subdivisions.

138. As described for claim 1, Ellis teaches tracking page utilization including “valid-page count.” Ellis teaches that superblocks contain a set number of blocks and pages. EX1004, 7:7-31. A POSA would understand that the “remaining space of the specific one of the subdivisions that is available for writing without first erasing the specific one of the subdivisions” is the total number of pages in a superblock minus the number of valid pages. Therefore, Ellis

teaches a system where the extent of space remaining for writing is tracked because the total number of superblock pages is known, and the valid-page count is known, and difference between those two numbers equals the remaining space in the superblock. *Id.*, 11:58-12:2.

139. To the extent Ellis does not render this limitation obvious, Son provides an explicit teaching of “remaining space” metadata, namely, the “number of remaining free pages.” EX1005, [0067], [0065]. A POSA would be motivated to include this information in the combination system of Ellis and Son due to its utility in maintenance, *e.g.*, selecting a target superblock to copy data from a source superblock. *Id.*

140. It is my opinion that the combined system of Ellis and Son renders this claim obvious.

8. Claims 13, 14, 18 and 20

141. Given the similarity in claim language, my opinion for Claim 1 applies to Claim 13.

142. Given the similarity in claim language, my opinion for Claim 3 applies to Claim 14.

143. Given the similarity in claim language, my opinion for Claim 8 applies to Claim 18.

144. Given the similarity in claim language, my opinion for Claim 10 applies to Claim 20.

B. Ground II

1. Motivation to combine Yamada with the combined system of Ellis and Son

145. As with Ellis and Son, Yamada teaches techniques to optimize memory management by the host and controller and prolong SSD life through the storage and analysis of maintained metadata in NAND flash memory. *E.g.*, EX1006, [0005]-[0007], [0011] (“it is an object of the present invention to provide an access device, a nonvolatile memory device, a memory controller used in a nonvolatile memory device, and a nonvolatile memory system that

improve their data retention properties and expand the range of their applications to various devices including portable devices.”), [0208], [0211], [0029] (“For example, the access device can solely determine an address for which maintenance is to be performed among predetermined addresses of the nonvolatile memory by managing the write history of the predetermined addresses of the nonvolatile memory. This improves the data retention properties of the nonvolatile memory.”).

146. A POSA would have been motivated to combine the teachings of Yamada with the combination of Ellis and Son. Specifically, a POSA would have been motivated to incorporate Yamada’s teaching regarding host-initiated maintenance/copy commands.

147. Yamada describes a discrete implementation of the shared host and controller functionality disclosed in Ellis and Son. Specifically, Yamada teaches a split maintenance functionality, where the controller tracks metadata and identifies data for maintenance and transmits that data to the host, where the host then issues a maintenance command for that data. EX1006, [0013]-[0015]. Yamada teaches a POSA the benefits of such a system:

The present invention enables an access device to receive the status of a nonvolatile memory from a nonvolatile memory device and 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. The invention thus enables the maintenance to be performed with a sufficiently long processing time allocated to the maintenance and improves the data retention properties. The invention also enables the system management status, such as the battery level, to be controlled in a manner that the maintenance will be completed in a reliable manner.

Id., [0036].

148. In light of teaching from Yamada regarding the benefits of shared memory management for maintenance activities, Ellis’s teaching that functions, *e.g.*, maintenance, may be distributed to a combination of the host and controller, EX1004, 4:58-65; 20:40-44, and Son’s disclosure of the benefits of such shared functionality, a POSA would motivated to, and it would be obvious to, incorporate Yamada’s teaching regarding a controller which tracks metadata to

identify data for maintenance and a host which initiates maintenance based on controller recommendation.

149. A POSA would be able to incorporate Yamada's teaching into the combined system of Ellis and Son with known methods and with a reasonable expectation of success. The combined system already includes shared host and controller functionality and incorporating Yamada's teaching would merely require a discrete implementation of such shared functionality where the request for maintenance originates from the controller based on collected metadata (as in Ellis) and the command for maintenance originates from the host (as in Son).

2. Claim 6: The memory controller of claim 2,

(a) the memory controller further comprises logic to track metadata for each of the subdivisions, the metadata indicating a time since data was programmed,

150. As described in claim 2, Ellis teaches copying from a source to a target superblock. Ellis also teaches that copying, a command used to implement maintenance and garbage collection, may be commanded based on "super blocks 502 that are older or written a longer time in the past." EX1004, 19:4-15. A POSA would understand that this data represents "time since data was programmed."

151. Further, a POSA would be motivated to include the additional types of metadata used for copying described in Son. For example, Son teaches the tracking of superblock metadata including "last modification time," defined as "the most recent time that any page within the superblock was ... programmed." EX1005, [0042], [0080].

152. It is my opinion that the combined system of Ellis and Son renders this limitation obvious.

(b) to compare the tracked metadata to a threshold, and,

153. In order to implement Ellis's teaching that data may be copied based on "super

blocks 502 that are older or written a longer time in the past,” EX1004, 19:4-15, a POSA would understand that the terms “older” and “longer” inherently represent that a comparison is conducted thus teaching this limitation.

154. Further, a POSA would be motivated to incorporate Son’s teaching that a threshold may be used to identify superblocks (source and destination) for copying. Son teaches that the time since last programmed metadata is used to select target and source superblocks, through the use of a threshold, *i.e.*, “last modification time falling within a [threshold, a] predetermined time period (e.g., is at least 10 days old).” EX1005, [0042], [0081], Claim 36 (“The method of claim 32 wherein selecting a source superblock and target superblock comprises evaluating two or more of the following factors related to the superblocks: ... last modification times”). A POSA would be able to implement the time since last accessed threshold for determination of maintenance/copying in the combined system of Ellis and Son in a straightforward manner.

155. It is my opinion that the combined system of Ellis and Son renders this limitation obvious.

(c) responsive to a determination that the tracked metadata for the specific one of the subdivisions satisfies the threshold, to transmit to the host the address corresponding to the specific one of the subdivisions; and the data copy request is accompanied by the address corresponding to the specific one of the subdivisions.

156. As described for Claim 1[a]-[b], the combination system of Ellis and Son teach a copy operation based on time since last programmed metadata where the identification of superblocks for that copy operations depend on satisfaction of a threshold.

157. In the combined system of Ellis and Son, where memory management responsibility is shared between the host and controller, a POSA would be motivated to incorporate Yamada’s teachings describing the controller-initiated copy request in light of the benefits of such

distributed management control (as described above with respect to Motivation to Combine). Yamada describes means to make additional use of time since last programmed to optimize memory management and longevity in a system with shared host and controller functionality.

158. Yamada teaches that the controller identifies data to be copied based on metadata based on a threshold or criterion value. EX1006, [0032]-[0033] (“The maintenance request unit dynamically changes the criterion value used to determine whether to generate a maintenance request in accordance with a status of data stored in the nonvolatile memory”), [0099]. Where that threshold or criteria is met, the controller sends a “maintenance request” (a copy request) to the host identifying the “logical address of a block that needs maintenance,” *id.*, [0082], [0032], [0204]. If the data identified for copying spans multiple blocks, “[t]he maintenance request may include a plurality of logical addresses of a plurality of blocks instead of including only a single address of a single block.” *Id.*, [0116], [0095], [0213]. A POSA, incorporating this teaching from Yamada into the combined system of Ellis and Son, using superblocks, would understand that the maintenance request (the data copy request) would include a superblock address.

159. Further, Yamada also teaches that the data to be copied in the maintenance operation may be identified based on “the elapsed time from when the data has been written previously based on the write history, and determines whether the elapsed time is greater than or equal to a criterion value.” EX1006, [0164], [0211]. The similarity in metadata illustrates the commonality between the systems disclosed in Ellis, Son and Yamada and would further bolster a POSA’s motivation to combine.

160. It is my opinion that the combined system of Ellis, Son and Yamada renders this limitation obvious.

3. **Claim 9: The memory controller of claim 1, wherein:**

- (a) **the memory controller further comprises logic to assign a write destination in the flash memory, in which data accompanying the write request is to be stored, the write destination corresponding to the specific one of the subdivisions,**

161. As I described with respect to Ground I, Claim 1[d], Ellis and Son teach that the host issues requests, such as a copy command, to a superblock as a single unit.

162. Upon receipt of that command, Ellis teaches that the controller's logic "assign[s] a write destination" via a "a mapping operation of logical pages [] mapped to physical pages," where the controller "map[s]" the data "to physical pages" "to be stored." EX1004, 15:45-53, Claim 15. In other words, Ellis teaches that upon receiving a write request with a logical address, the controller assigns a physical address to store the data.

163. It is my opinion that the combined system of Ellis and Son renders this limitation obvious.

- (b) **and to transmit to the host a read address that will be used to retrieve the data specifically from the write destination;**

164. A POSA would be motivated to incorporate Yamada's teaching that, in conjunction with the shared controller host responsibility for initiating copy requests, the controller should provide an address for the data subject to the copy request.

165. Yamada teaches that when writing data or when copying data to perform a maintenance operation, the controller notifies the host of the write location: "The maintenance completion notification [sent by the controller to the host] contains a logical address of a block included in the nonvolatile memory 120 for which maintenance has been completed." EX1006, [0111]. The host may then use that address to identify that data for subsequent read commands. *Id.*, [0095], [0211]-[0213]. Additionally, Yamada teaches that data copied in maintenance, may include "a plurality of logical addresses of a plurality of blocks," such that the copy command is

to a group of blocks and the host notification following copying identifies the group of blocks for subsequent read addresses. *See, e.g., id.*, [0116]. Therefore, Yamada teaches that the controller transmits the address of the data subject to the copy request, and the host may use that address for subsequent read operations.

166. A POSA applying the teaching of Yamada, directed to multiple logical addresses and multiple blocks, to the combined system of Ellis and Son would rely on superblock addressing for both the initial write request and the read request.

167. It is my opinion that the combined system of Ellis, Son and Yamada renders this limitation obvious.

- (c) **and the memory controller further comprises logic to receive, from the host, and via the host interface, a read request for the data, accompanied by the read address, and to identify both of the specific one of the subdivisions and the write destination from the read address.**

168. Ellis teaches that “[t]he host system [may] request the memory controller 104 for reading.... data from ... a logical address space of a storage.” EX1004, 3:25-28. Therefore, Ellis teaches a memory controller with logic to receive a read request coupled with a read address.

169. Ellis also teaches that the controller stores tracked metadata including mapping information for each superblock. *See, e.g.*, Claim 15 (the “controller [is] configured to ... update the metadata corresponding to the super block, wherein updating the metadata includes updating a mapping of the super block.”). A POSA would understand that upon receiving the read command with a read address, that the controller would use the mapping information to identify the corresponding read addresses (at the write destination) in the superblock that the data was written to.

170. Further, Yamada provides a similar teaching. Yamada teaches that a host “command for reading” includes the “address (logical address) of data to be read” and the

controller, via the read/write unit, identifies the data by identifying “a designated address,” namely, “a physical address corresponding to the logical address” in the read request. EX1006, [0095]. In the combination system, a POSA would apply Yamada’s teaching to the superblocks in Ellis.

171. Alternatively, in the combined system, a POSA would be motivated to incorporate Son’s teaching that the host-initiated read request includes the physical address of the data written by the controller. Son teaches that, for “a read operation, the SSD module may use a received or determined FLBA as the lookup into the mapping table in order to identify the corresponding PLBA for the read request. The SSD module may then send a read request, including the identified PLBA, to the SSD.” EX1005, [0083]. In light of the organization into superblocks, a POSA would understand that this process, identifying a write location and read location, would include superblock addresses for the write and read command.

172. It is my opinion that the combined system of Ellis, Son and Yamada renders this limitation obvious.

4. Claims 17 and 19

173. Given the similarity in claim language, my opinion for Claim 6 applies to Claim 17.

174. Given the similarity in claim language, my opinion for Claim 9 applies to Claim 19.

C. Ground III

1. Motivation to Combine

175. A POSA would be motivated to combine the teachings of Sinclair, Yu and Son because they are in the same field of endeavor, “advanced management techniques” to optimize storage and retention of memory data in NAND flash storage devices. EX1008, [0012], [0291], EX1005, [0002], [0014], EX1007, 8:46-53, 3:52-5:13.

176. A POSA would be motivated to combine the teachings of Sinclair, Yu and Son due to the core similarities in the disclosed systems and the methods.

177. First, Sinclair and Son both rely on memory systems organized into superblocks or metablocks and teach the use superblock or metablock level metadata.

178. Sinclair teaches combining “four of more blocks” into metablocks “to increase the degree of parallelism in data programming and reading.” EX1007, 3:39-48, 9:52-55. Sinclair teaches the benefits of metablock level metadata for memory management. *Id.*, 31:40-32:10, 26:5-17.

179. I incorporate my discussion with respect to Ground I, regarding Son’s parallel teaching regarding superblocks and superblock metadata.

180. Second, Sinclair, Yu and Son, each describe the benefits of shared memory and host management responsibilities and disclose systems where metadata such as page utilization is used for those management responsibilities.

181. Sinclair teaches that “improved performance and endurance may also be realized” through shared memory management responsibility between the host and controller, *e.g.*, *id.*, 4:23-47, and discloses systems with variable host control, as shown in, *e.g.* Fig 8A, 8B. Sinclair teaches the benefits of “possible divisions of [] functions as between the host and the memory system.” EX1007, 16:26-67, Figs. 8A-8C. Indeed, the “primary embodiment” “a Flash Optimized File System” incorporates this shared functionality. *Id.*

182. Sinclair also teaches the tracking of page utilization metadata for metablocks due to its utility in memory management such as maintenance activities. EX1007, 31:40-32:10, 26:5-17.

183. Yu teaches that responsibility for memory management is done cooperatively between the host and controller. For example, Yu teaches host functionality including a “Super Enhanced Endurance Device” and “S.M.A.R.T.” functionality to “ease the loads of [the] SSD” and to “monitor some important data from the SSD device” including “wear,” “wear-leveling

count” “and other flash memory management.” EX1008, [0055], [0064].

184. Yu also teaches the usefulness of metadata showing “the status of each individual page of flash,” *i.e.*, “an empty page” or “a used page.” *Id.*, [0083], [0092].

185. I incorporate my discussion with respect to Ground I, regarding Son’s teaching regarding superblock page utilization and shared memory management responsibility between the host and controller.

186. Due to the similar disclosures regarding split host and controller functionality, where responsibility for managing memory is accomplished based on groups of blocks or block-level metadata including page utilization, a POSA would be motivated to incorporate the teachings from Son and Yu into the system of Sinclair.

187. Specifically, a POSA would be motivated to combine the teachings from Son—*e.g.*, additional variants of metadata and increased host responsibility—to the system in Sinclair, because Sinclair already teaches the utility of various forms of metablock metadata as well as the need to share information with the host for the purposes of memory management. Son simply provides additional means to carry out the core teachings disclosed in Sinclair. Due to the identity techniques to solve the same problems, the use of groupings of block termed superblocks or metablocks, where metadata from those groups is used to manage the memory, the combination of Sinclair and Son would be straightforward and would be obvious for a POSA to try.

188. Additionally, a POSA would be motivated to include Yu’s teaching regarding for S.M.A.R.T. functionality into the combination of Sinclair and Son. Yu’s S.M.A.R.T. functionality simply represents a discrete implementation of the shared host and controller responsibility taught in Sinclair and Son. Adding this functionality would be a straightforward endeavor that could be done without fear of failure.

189. A POSA would be motivated to combine Sinclair, Son and Yu.

2. **Claim 1[pre]: A memory controller to control flash memory, the flash memory having subdivisions, each of the subdivisions corresponding to a respective group of erase blocks, each of the erase blocks such that it must be erased before storage locations in that erase block can be reprogrammed, the memory controller comprising:**

190. Sinclair teaches a “controller” that “control[s] operation of the flash memory.” EX1007, 7:2-4. The flash memory has subdivisions, called “metablock units,” where each metablock includes “four blocks” “from each plane [] logically linked together to form a metablock.” EX1007, 9:52-57, 7:2-4. Sinclair teaches that each of the blocks “must be erased before any data stored in the block can be overwritten.” *Id.*, 3:15-18.

191. It is my opinion Sinclair renders this limitation obvious.

(a) Claim 1[a]: a host interface;

192. Sinclair teaches a “host interface” as shown in Figure 1, where the controller comprises “circuits that ... interface with the host.” EX1007, 7:2-31, Fig. 1.

193. It is my opinion Sinclair renders this limitation obvious.

(b) Claim 1[b]: a memory interface;

194. Sinclair teaches a “memory interface” as shown in Figure 2, where the controllers “control operation of the flash memory.” EX1007, 7:2-31, Fig. 2.

195. It is my opinion Sinclair renders this limitation obvious.

(c) Claim 1[c]: logic to track information indicating extent of page utilization of each of the subdivisions;

196. Sinclair teaches a controller with logic to track metadata for metablocks in the form of lists, where the “list(s) can include other information of the blocks ... such as the amount of valid data in each block and the amount of erased space in each block.” EX1007, 31:46-49. The metablock metadata includes information describing the “number of ... metapages ... of data” and “the number of [meta]pages of unprogrammed capacity.” *Id.*, 26:5-17. A host would understand that the number of valid metapages describes the “extent of page utilization” as each metapage

corresponds to four pages in a set ratio: “A metapage 151 is illustrated in FIG. 5, being formed of one physical page from each of the four blocks.” *Id.*, 10:12-14.

197. Further as described in Ground I, Son discloses additional variants of superblock metadata disclosed in Son including “free page count.”

(d) Claim 1[d]: logic to receive a write request from a host via the host interface, the write request accompanied by an address that designates a specific one of the subdivisions, wherein the logic to track is to, in association with execution of the write request, update the information indicating extent of page utilization of the specific one of the subdivisions;

198. Sinclair teaches that the host assigns logical address locations for files subject to write requests. EX1007, 13:9-12. The host’s assignment of an address to a metablock is possible because Sinclair teaches that the “host ... needs to know the physical characteristics of the memory with which it is operating” including, *i.e.*, “[t]he number of pages per metablock; and [] [t]he lowest logical address that is mapped to a first page of a physical metablock.” *Id.*, 18:16-43. Accordingly, the host has knowledge of the relationship between logical address and a specific metablock.

199. Sinclair teaches that the host issues write requests with an address that designates a metablock: “When the host writes data to the memory system, the controller converts the logical addresses assigned by the host to physical addresses within a metablock.” *Id.* 13:9-15. Therefore, a POSA would understand that Sinclair discloses a controller which may receive a write request from the host, where that host includes a logical address; and a POSA would further understand that the logical address—upon conversion to a physical address—designates a metablock.

200. Upon completing the write to the designated metablock, the metadata comprising page utilization is updated:

The list(s) of such blocks changes constantly as data are written, updated, moved, deleted, and so forth. Changes that result in blocks changing their types to and from partial, obsolete and invalid cause the list(s) maintained by the step 701 of FIG. 29

to change. Changes in the amount of valid data individually stored in such blocks, and in the amount of erased capacity are also noted in the block lists.

Id., 31:65-32:4—throughout the disclosure in Sinclair, “block” and “metablock” are used interchangeably, *see id.*, 16:22-26 (“it is intended that the term ‘block’ as used herein refer to either the block unit of erase or a multiple block ‘metablock’”).

201. Further, Son provides additional implementation details for superblock-level (or metablock-level) write request, as described for Ground I, through the use of a superblock address.

202. Therefore, it is my opinion that the combination system of Sinclair and Son teach this limitation.

(e) Claim 1[e]: logic to send the host the information indicating extent of page utilization of the specific one of the subdivisions and an address corresponding to the specific one of the subdivisions;

203. Sinclair teaches the benefit of sharing memory management responsibility, EX1007, 16:26-67; Figs. 8A-8C, with the host which would motivate a POSA to incorporate the teaching of Yu, which includes S.M.A.R.T. host-initiated monitoring of memory management based on queries to the controller.

204. Yu describes a “S.M.A.R.T. [(“stands for Self-Monitoring, Analysis and Reporting Technology”)] function [that] processes S.M.A.R.T. commands or vendor commands from host 100, such as to monitor and control error correction, wear, bad block, and other flash memory management.” EX1008, [0064]. The “S.M.A.R.T.” functionality enables the host to “monitor some important data from the SSD device” including “wear,” “wear-leveling count” “and other flash memory management.” *Id.* The data monitored via S.M.A.R.T. is flexible and may “be expanded [to] collect[] other important data from the SSD” which the host may use “to analyze the data for debugging, statistics, etc.” *Id.*, [0225].

205. Yu teaches that such S.M.A.R.T. data and information, “can be requested by the

host” via “command[]” and responsively sent by the controller to the host. *Id.*, [0085] (“S.M.A.R.T data collector 170 has data tables and other information used by SMART function 39 (FIG. 1) and can be requested by the host through SMART commands or vendor commands”), [0232]. The information is aggregated and stored in various tables. *Id.*

206. To implement the S.M.A.R.T. functionality into the combination system of Sinclair and Son, a POSA would rely on metablock-level metadata where that includes an address corresponding to the specific one of the subdivisions so that the information is useful to the host.

207. Additionally, and further supporting the motivation to combine, Yu also teaches the maintenance of page utilization data. This data is located in “tables” such as the “[p]age status table.” *Id.*, [0083]-[0085], [0092]. A POSA would understand that the page status table represents page utilization and “show[s] the status of each individual page of flash,” *i.e.*, “an empty page” or “a used page.” *Id.*, [0083], [0092].

208. Therefore, it is my opinion that the combination system of Sinclair, Son and Yu renders this limitation obvious.

(f) **Claim 1[f]: logic to receive an erasure request from the host, via the host interface, the erasure request accompanied by the address corresponding to the specific one of the subdivisions, to responsively control erasure of each unerased erase block in the group respective to the specific one of the subdivisions, and to update the information indicating extent of page utilization of the specific one of the subdivisions; and**

209. As discussed for Claim 1[d], Sinclair teaches that page utilization is updated upon write and erasure. Sinclair further teaches that an erasure request may be directed to a metablock as a single unit. *Id.*, 27:61-28:12, 9:39-67 (“All of the cells within a metablock are typically erased together”). In order to accomplish erasures at the superblock-level a POSA would understand that the erasure request must designate a particular metablock via metablock address.

210. As discussed for Ground I, Son teaches a particular means of accomplishing

metablock or superblock level erasures “at the superblock-level,” [0029], using a command designating a “superblock address,” [0038], where page utilization is updated afterwards, [0060]-[0061]. Given the teaching of variable host level responsibility, including in mapping, and the need for the host to know information about the physical memory, a POSA would be motivated to apply the teaching from Son leveraging host knowledge of mapping to the system in Sinclair.

211. Therefore, it is my opinion that the combination system of Sinclair, Son and Yu renders this limitation obvious.

(g) Claim 1[g]: logic to, in connection with the erasure, detect a defect in one of the erase blocks in the group respective to the specific one of the subdivisions, to responsively substitute one or more different erase blocks of the flash memory for the one of the erase blocks for which the defect was detected

212. Sinclair teaches “the substitution of a redundant block for a defective block originally assigned to the metablock.” EX1007, 13:1-9. Further, Yu provides specific examples for defect detection, including “[a]fter the block is erased, the status of each page in that block is checked” to determine if it is bad. EX1008, [0241]-[0243]. A POSA would understand that defect detection may occur in connection with operations such as erasures.

213. Therefore, it is my opinion that the combination system of Sinclair, Son and Yu renders this limitation obvious.

(h) Claim 1[h]: wherein each said logic comprises at least one of hardware circuitry or instructions stored on non-transitory, machine-readable media that when executed are to control the operation of hardware circuitry.

214. Sinclair teaches a controller “wherein each said logic comprises at least one of hardware circuitry or instructions stored on non-transitory, machine-readable media that when executed are to control the operation of hardware circuitry.” *See, e.g.*, EX1007, 7:32-8:23.

215. Therefore, it is my opinion that the combination system of Sinclair, Son and Yu

renders this limitation obvious.

3. Claim 2: The memory controller of claim 1,

(a) [2a]: wherein the host interface is also to receive a data copy request from the host,

216. Sinclair teaches that the host may issue reclaim commands, which are commands to the controller to initiate copy operations of data from a source metablock to a destination metablock: “Copying of the valid data of one file from the source reclaim [metablock] to the destination [metablock],” “[i]f the host has issued an idle command.” EX1007, 32:11-52. Therefore, the controller, via the host interface, is capable of receiving a copy request from the host.

217. A POSA would understand that the idle command is equivalent to “reclaim_on” command or an affirmative copy command. As corroborating evidence of this understanding, Sinclair “cross-reference[s]” another patent authored by Sinclair (U.S. Pat. No. 7,409,489) (Sinclair II) which states exactly this: “[t]he ‘Reclaim_on’ is equivalent to an ‘idle’ command.” EX1009, 23:17-46.

218. Additionally, as I described for Ground II, Claim 2[b], Son discloses a controller that may receive a superblock (or metablock) copy operation, in response to a host copy command.

219. Therefore, it is my opinion that the combination system of Sinclair, Son and Yu renders this limitation obvious.

(b) [2b] wherein the memory controller is to responsively retrieve data from a source location in the flash memory, the source location associated with the specific one of the subdivisions, and write the retrieved data to a destination location in the flash memory, without sending the retrieved data to the host,

220. The reclaim commands (described for 2[a]) from the host are a product of shared host and controller memory management, where the controller copy executions are initiated based

on knowledge of host activities, *e.g.*, periods of idle time. *See, e.g.*, EX1007, 32:11-26, (“If the host has issued an idle command, or something similar that indicates there will be some period of time when the host will not be expecting the memory system to perform a particular operation, then the system is free to carry out overhead operations in the foreground including a reclaim operation.”). The fact that the copy operations are completed during host idle periods shows that, following the command, the host is not involved in the copy operation and does not receive the retrieved data. Instead, the responsive copying of data from a source superblock to a destination superblock is completed by the controller, without sending the copied data to the host. *Id.*, 30:13-31:27, 14:19-63.

221. Additionally, as I described for Ground II, Claim 2[b], Son discloses a superblock (or metablock) copy operation, in response to a host copy command, where data is retrieved from a source location and written to a destination location, without sending data to the host.

222. Therefore, it is my opinion that the combination system of Sinclair, Son and Yu renders this limitation obvious.

(c) **[2c] and wherein the logic to track is to responsively update the tracked information for one of the subdivisions which corresponds to the destination location, in association with the write of the retrieved data into the destination location.**

223. As described for claim 1[d], upon the completion of a write operation (as in a copy operation), Sinclair and Son teach responsively updating the page utilization metadata for the metablock/superblock where data was written (the destination location). *See also* EX1007, 31:65-32:4.

224. Additionally, as I described for Ground II, Claim 2[c], Son also discloses updating the tracked metadata for the destination superblock (or metablock) in the copy operation.

225. Therefore, it is my opinion that the combination system of Sinclair, Son and Yu

renders this limitation obvious.

4. Claim 8: The memory controller of claim 1, wherein the memory controller further comprises logic to receive a query from the host, and wherein the logic to send the information to the host is to do so as a response to the query.

226. Sinclair teaches that the controller may send to the host mapping information including “the characteristics of the physical memory blocks.” EX1007, 4:54-57.

227. Sinclair also references known methods in the art for querying information from the controller, such as a “vendor specific command that is issued by the host” which causes the controller “to return the stored parameter values to the host.” *Id.*, 18:48-60.

228. As described for Ground I, Son teaches this limitation.

229. As described with respect to 1[e], Yu teaches a controller which includes logic to receive a query or request from the host via S.M.A.R.T functionality, where the logic enables the controller to send to the host the information requested in response to the query.

230. Therefore, it is my opinion that the combination system of Sinclair, Son and Yu renders this claim obvious.

5. Claim 10: The memory controller of claim 1,

(a) wherein the memory controller further comprises logic to store erase count information for each of the subdivisions,

231. As described for Claim 1, the combination of Sinclair, Yu and Son describes a controller which tracks metablocks. Sinclair teaches additional metadata including erase count for each metablock: The “experience count of the number of times the [metablock] has been erased and re-programmed.” EX1007, 10:65-11:5.

232. As described in Ground I, Son also teaches metadata including “wear count” for each superblock.

233. Additionally, Yu teaches tracking erase count via an “an erase count table for the

erased block.” EX1008, Claim 17; *see also id.* [0083].

234. Therefore, it is my opinion that the combination system of Sinclair, Son and Yu renders this limitation obvious.

(b) and to send to the host, via the host interface, the erase count information for the specific one of the subdivisions, together with the address corresponding to the specific one of the subdivisions,

235. As described for Claim 1, the combination of Sinclair, Yu and Son describes a controller which tracks metablock metadata which is sent to the host along with an address identifying the relevant metablock.

236. Yu provides an explicit teaching that erase count metadata is transmitted to the host along with address information.

237. Yu teaches that the controller may transmit to the host “wear-leveling count” as part of “S.M.A.R.T. functionality.” *Id.* [0064]. As wear-leveling requires knowledge of erasures, and both wear-leveling and erase cycles are tracked by the controller, a POSA would understand that it is straightforward to also transmit the erase count to the host. *See, e.g., id.*, [0060] (“Wear-leveler 24 provides two types of wear leveling methods. One is to dynamically choose the lowest wear leveling count block from the available empty blocks to write data to.”). A POSA would understand that, for the erase count information to be of use in wear-leveling, the erase count must include address information.

238. To implement this S.M.A.R.T. functionality—transmitting erase count metadata from the controller to the host, representing metadata tracked by Sinclair and Son—in the combination system with the metablocks, a POSA would be able to make the straightforward modification to track and transmit erase count based on individual metablocks.

239. Therefore, it is my opinion that the combination system of Sinclair, Son and Yu

renders this limitation obvious.

(c) and wherein the logic to store the erase count information is to update the erase count information for the specific one of the subdivisions in association with the erasure.

240. A POSA would understand that for Sinclair to track “the number of times” a metablock is “erased,” EX1007, 10:65-11:5, the system must update erase count in association with an erasure.

241. As described in Ground I, Son also teaches metadata including “wear count” for each superblock, where each superblock is associated with a superblock address, is updated upon erasure, and that information is provided to the host.

242. Additionally, Yu teaches tracking and updating erase count upon erasure. Yu teaches that “when an erased block in a flash memory has been erased, [to] increase[e] an erase count in an erase count table for the erased block.” EX1008, Claim 17; *see also id.* [0083].

243. Therefore, it is my opinion that the combination system of Sinclair, Son and Yu renders this limitation obvious.

6. Claim 11: The memory controller of claim 1, wherein the extent of the page utilization of the specific one of the subdivisions represents the amount of valid data written to the specific one of the subdivisions since the specific one of the subdivisions was last erased.

244. As discussed for Claim 1, Sinclair teaches the maintenance of page utilization metadata including the “number of pages” written in a metablock. EX1007, 26:5-17; *see also* 1[c].

245. As described in Claim 10, both Sinclair and Son teach tracking of erase count.

246. A POSA would understand based on those two classes of metadata, the tracked page utilization metadata for any superblock with an erase count of 1 or greater, represents the amount of valid data written since last erased.

247. It is my opinion that the combined system of Sinclair, Yu and Son renders this

limitation obvious.

7. **Claim 12: The memory controller of claim 1, wherein the extent of the page utilization of the specific one of the subdivisions represents a remaining space of the specific one of the subdivisions that is available for writing without first erasing the specific one of the subdivisions.**

248. As discussed for claim 1, Sinclair teaches the maintenance of page utilization metadata including the remaining space—“pages of unprogrammed capacity”—of a metablock. EX1007, 26:5-17; *see also* 1[c].

249. It is my opinion that the combined system of Sinclair, Yu and Son renders this limitation obvious.

8. Claims 13, 14, 18, and 20

250. Given the similarity in claim language, my opinion for Claim 1 applies to Claim 13.

251. Given the similarity in claim language, my opinion for Claim 3 applies to Claim 14.

252. Given the similarity in claim language, my opinion for Claim 8 applies to Claim 18.

253. Given the similarity in claim language, my opinion for Claim 10 applies to Claim 20.

D. Ground IV

1. Motivation to Combine

254. A POSA would be motivated to combine Jung with Sinclair, Yu and Son.

255. First, Jung describes methods and systems for optimizing, *e.g.*, “NAND flash” “SSD[s]” memory systems. EX1010, [0003]-[0004].

256. Second, Jung provides a motivation to delegate certain memory management functions to the host. Jung describes the downfalls and limitations of conventional SSDs which rely entirely on the controller for memory management. For example, it describes the limited resources of flash controllers which may result in overloading or slow speeds:

A conventional flash card may perform all FTL functions in the flash controller. Unfortunately, the resources of flash controllers are generally smaller than those of hosts. Thus, a resulting execution speed of such FTL functions performed in the controller is typically slow. In some cases, the FTL functions cannot be performed at all, such as where the controller becomes overloaded.

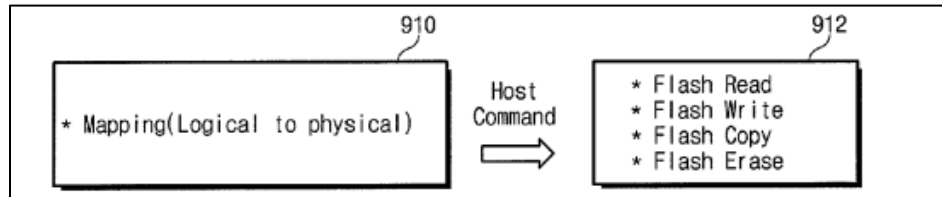
EX1010, [0005]. To address this known shortcoming, Jung discloses systems and methods for “Cooperative memory management,” where responsibilities for memory management are split or shared between the host and controller and “the host assumes at least one memory management function for the memory card device.” *E.g., id.*, [0007], [0044] (“the host and the device may have one or more overlapping functions, so it remains to be determined which will actually implement each of the overlapping functions.”).

257. In short, Jung teaches that memory management responsibilities and functions should be shared between the host and controller in order to “efficiently distribut[e] the resources of such systems” by moving functionality to the host to optimize memory. EX1010, [0002], [0005]. This teaching is similar to that of Sinclair, Yu and Son, *e.g.*, EX1007, Fig 8A, 8B; EX1008, [0055] (teaches that the memory management to “ease the loads of [the] SSD”); EX1005, [0063] (“SSD module performs various SSD maintenance operations including garbage collection and wear leveling, which can relieve the SSD controller and firmware of these responsibilities”), which each describe certain implementations of the same fundamental concept.

258. Third, Jung teaches the need to share certain metadata with the host to support “functions on the host.” EX1010, [0038].

259. To enable the host to take on this memory management responsibility, the controller “loads meta information, such as an erase count, garbage collection data and a mapping table” to the host. *Id.*, [0038]. For example, Jung teaches that mapping information is provided to the host to allow for, *i.e.*, more efficient write commands, EX1010, [0015] (“Another method is provided comprising producing a write request from the host reading mapping information using host

memory management functions”), as shown in excerpted Figure 9 below.



This teaching from Jung parallels the disclosure in Sinclair, which describes the need to share certain mapping information—such as “[t]he lowest logical address that is mapped to a first page of a physical metablock”—and other characteristics of physical memory with the host in order to optimize memory. EX1007, 18:16-28.

260. Yu teaches that the host has S.M.A.R.T. functionality, which is enabled by the controller sending “important data” including “health information,” “wear,” “wear-leveling count,” “bad block, and other flash memory management” information to the host. EX1008, [0064], Claim 6. Like Jung, the data sent to the host may be enlarged to support a broader scope of host functionality, *i.e.*, “expanded [to] collect[] other important data from the SSD” which the host may use to, *e.g.*, “analyze the data for debugging, statistics, etc.” *Id.*, [0225].

261. Incorporating the teaching of Jung into the combination system of Sinclair, Yu and Son—the transmission of mapping information and other metadata to the host—would be straightforward to implement. Each reference in the combination system already provides for functionality where the controller sends metadata and other information to the host. To incorporate Jung’s teaching, the class of information sent to the host would only be modestly increased. This incorporation could be completed with a reasonable expectation of success.

262. In light of the teachings of the downfalls of reliance entirely on the controller for memory management, as described in, *e.g.*, Sinclair and Son, the incorporation of additional forms of information sent to the host into the combination system would be obvious to try.

2. Claim 6: The memory controller of claim 2,

(a) [6a] the memory controller further comprises logic to track metadata for each of the subdivisions, the metadata indicating a time since data was programmed,

263. As described in Ground III, Claim 2, Sinclair teaches that metablock metadata, such as page utilization, is tracked in connection with copy commands, where copy commands make use of metablock addressing. Sinclair further teaches that metablock metadata and “other information” is used for “the selection” of a metablock to be copied. EX1007, 31:46-49 (“This list(s) can include other information of the blocks that enable the selection of one reclaim block at a time, such as the amount of valid data in each block and the amount of erased space in each block.”).

264. In light of these teachings, a POSA would be motivated to incorporate Son’s teaching—as I described for Ground II, Claim 6[a]—that the metablock/superblock to be copied is selected based on another type of metablock/superblock metadata—“last modification time.” EX1005, [0042], [0065], Claim 36.

265. It is my opinion that the combined system of Sinclair, Yu and Son renders this limitation obvious.

(b) [6b] to compare the tracked metadata to a threshold,

266. In light of Son’s teaching of the utility of a threshold for the time since last accessed metadata in maintenance and copy operations discussed above, a POSA would be motivated to incorporate the use of time since last accessed metadata thresholds in the combination system and could do so in a straightforward manner. Moreover, Sinclair already provides a teaching regarding the utility of thresholds, establishing the ease of incorporation and the basis to include additional thresholds for new metadata incorporated into the combination. EX1007, 31:56-61 (“only those blocks that have a small amount of valid data below some set threshold amount are included, since

one characteristic of a chosen reclaim block is that it has little or no valid data that needs to be copied”).

267. As further evidence that such incorporation of a threshold would be obvious to a POSA to try, Yu describes the use of various thresholds based on tracked metadata. EX1008, [0060], [0190].

268. Adding this metadata threshold taught by Son into the system described in Sinclair would be obvious to try, and a straightforward and easily accomplishable task for a POSA.

269. It is my opinion that the combined system of Sinclair, Yu and Son renders this limitation obvious.

(c) **[6c] and, responsive to a determination that the tracked metadata for the specific one of the subdivisions satisfies the threshold, to transmit to the host the address corresponding to the specific one of the subdivisions; and the data copy request is accompanied by the address corresponding to the specific one of the subdivisions.**

270. As described for 6[a]-6[b], the combined system of Sinclair, Son and Yu, teaches a controller which tracks page utilization and time since data was programmed for each subdivision, and compares the time since data was programmed to a threshold to determine the need for maintenance (reclaim operations) such as copying.

271. As described for Ground I, Claim 1[d], and Ground III, Claim 1[d], Son and Sinclair respectively disclose write and copy commands which are accompanied by addressing corresponding to a metablock or superblock, *i.e.*, a metablock address or superblock address.

272. In the copy operation of the combined system, a POSA would be motivated to rely on the “S.M.A.R.T.” functionality in Yu (which is part of the combined system as set out in Claim 1). Yu describes the use of “S.M.A.R.T.” functionality to “monitor some important data from the SSD device.” EX1008, [0064], [0085], [0232]. Because the combined system includes thresholds

that when satisfied trigger certain copy operations to improve SSD functionality, a POSA would understand that satisfying such a threshold would qualify as “important data” that S.M.A.R.T. monitors and sends “to the host.” *Id.*

273. Thus, in the combined system of Sinclair, Son, and Yu, the controller would “transmit to the host” information identifying the metablock which exceeds the time since last accessed threshold using an address “corresponding to the subdivision.”

274. To the extent the combination of Sinclair, Yu and Son does not render this limitation obvious, a POSA would be motivated to include the teaching from Jung regarding the scope of information sent to the host.

275. A POSA would be motivated to incorporate the teaching of Jung, “to transmit to the host the address corresponding to the specific one of the subdivisions.” Sinclair, Son, and Yu, describe various applications of host and controller shared memory management responsibility and metadata, and provide motivations for the same. For example, Sinclair already teaches the need for the controller to share mapping information with the host, such as “[t]he lowest logical address that is mapped to a first page of a physical metablock.” *Id.*, 18:16-28. Accordingly, a POSA would be motivated to incorporate the specific teaching that the controller may transmit additional metadata, including complete mapping information, to the host for the purpose of identifying data for the host to issue a copy command.

276. Jung teaches that the controller may provide metadata to the host including, *e.g.*, “an erase count, garbage collection data and a *mapping table*.” EX1010, [0038] (“The function block 418, in turn, determines which FTL functions will be performed in the host, on the basis of the host's remaining resources, and passes control to a function block 420. The function block 420 initializes the determined FTL functions on the host, and passes control to a function block 422. The function block 422 loads meta information, such as an erase count, garbage collection data

and a mapping table from the flash memory to the host RAM, and passes control to a function block 424. The function block 424 activates the determined host FTL functions, and passes control to a function block 426.”), Claim 59 (“A method as defined in claim 58 wherein the meta data comprises at least one of erase count data, garbage collection data or mapping table data loaded to the host random access memory from the memory card device”).

277. This and other metadata are provided to the host in order for the host to perform memory management as part of “cooperative memory management” such as “mapping [], garbage collection, power-off recovery, bad block management, [and] wear-leveling.” *Id.*, [0041]. Jung also teaches that the host uses the metadata provided by the controller to issue copy commands. *Id.*, [0048] (“Invocation of the mapping function 910 causes the host to issue one of a plurality of mapping commands 912, such as a flash read command, a flash write command, a flash copy command, or a flash erase command.”), [0015] (“Another method is provided comprising producing a write request from the host reading mapping information using host memory management functions, checking for write space available on the memory card device, and sending the write request to the device if there is write space available on the device.”).

278. Thus, Jung teaches that the controller may send mapping information to the host to facilitate operations such as copy commands.

279. A POSA would be motivated to incorporate the teaching from Jung—that the controller transmits mapping information and other metadata to identify data for copying to the host, and the host issues copy commands—to the combined system of Sinclair, Son and Yu. A POSA would apply this teaching to the metablocks in Sinclair, such that the time-since accessed metadata and mapping information would apply to a metablock.

280. It is my opinion that the combined system of Sinclair, Yu, Son and Jung renders this limitation obvious.

3. **Claim 9: The memory controller of claim 1, wherein:**

- (a) **[9a] the memory controller further comprises logic to assign a write destination in the flash memory, in which data accompanying the write request is to be stored, the write destination corresponding to the specific one of the subdivisions,**

281. Sinclair describes a system where host write commands include logical addresses and the controller converts those logical addresses “to physical addresses within a metablock.” EX1007, 13:9-12, 12:20-30. *See also id.*, 12:31-45, 2:66-3-6. The data is “stored” in the physical addresses. *Id.* Sinclair therefore teaches a controller with logic to assign the destination in physical memory of data accompanying a write request.

282. Additionally, Yu teaches that the controller engages in “block mapping” where “an entire logical block from the host is mapped to an entire physical block in flash memory.” EX1008, [0096]. In the combination system, a POSA would use this teaching to implement metablock-level mapping because metablocks are the unit of memory. Therefore, the combined system teaches the controller “*assigns a write destination in the flash memory ... in which data accompanying the write request is to be stored, the write destination corresponding to the specific one of the subdivisions.*”

283. It is my opinion that the combined system of Sinclair, Yu, and Son renders this limitation obvious.

- (b) **[9b] and to transmit to the host a read address that will be used to retrieve the data specifically from the write destination;**

284. A POSA would be motivated to incorporate the teaching from Jung that the controller “transmits a read address that will be used to retrieve the data specifically from the write destination” in the combination system for the reasons set out in Claim 6[c].

285. Jung teaches that the controller “load[s]” or transmits “mapping table data” to the host. *See, e.g.*, EX1010, Claim 59 (“A method as defined in claim 58 wherein the meta data

comprises at least one of erase count data, garbage collection data or mapping table data loaded to the host random access memory from the memory card device”). Jung teaches the host may then use that mapping information to identify a read address and to issue read requests to the written data. *Id.*, [0048] (“Invocation of the mapping function 910 causes the host to issue ... a flash read command.”).

- (c) **[9c] and the memory controller further comprises logic to receive, from the host, and via the host interface, a read request for the data, accompanied by the read address, and to identify both of the specific one of the subdivisions and the write destination from the read address.**

286. Sinclair teaches that the memory controller contains logic to receive read requests or “read operations” from the host. EX1007, 32:11-26.

287. As described for 9[a], Sinclair teaches that the controller maps the physical address (destination) address for data subject to a write request. Sinclair further teaches that the controller, *i.e.*, in response to a read command, uses “the logical to-physical address table ... to identify the [] physical block to which the data at those logical addresses are stored,” [“the write destination from the read address”]. EX1007, 13:16-33. A POSA would understand that the controller’s identification of the block in which the data subject to read address is stored would necessarily include the identification of the corresponding metablock (in which that block is located).

288. As described for Ground III, Claim 2[b], the combined system of Sinclair, Yu and Son a controller with logic to “responsively retrieve data from a source location in the flash memory,” based on an address provided by the host in connection with a copy command. The copy operation includes, a read and a write operation, where for the read operation, the controller contains logic to identify the metablock and the address for the data subject to the host command.

289. As I described for Ground II, Claim 9[c], Son renders this limitation obvious.

290. It is my opinion that the combined system of Sinclair, Yu, Son and Jung renders

this limitation obvious.

4. Claims 17 and 19

291. Given the similarity in claim language, my opinion for Claim 6 applies to Claim 17.

292. Given the similarity in claim language, my opinion for Claim 9 applies to Claim 19.

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

Dated: July 9, 2025

A handwritten signature in black ink, appearing to read 'Heiner Litz', positioned above 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 PhD. in Computer Engineering, Mannheim University
Advisor: Professor Ulrich Bruening
2010 Visiting Researcher, University of Valencia
Advisor: Professor Jose Duato
2006 Diplom Informatik (M.Sc), Mannheim University

Honors and Awards

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

Publications

Refereed Conference Papers

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

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

35. Fröning, H., **Litz, Heiner**, and U. Brüning (2009). Efficient Virtualization of High-Performance Network Interfaces. In: *8th International Conference on Networks (ICN'8)*.
36. **Litz, Heiner**, H. Fröning, M. Thürmer, and U. Brüning (2009). An FPGA based Verification Platform for HyperTransport 3.x. In: *19th International Conference on Field Programmable Logic and Applications (FPL'19)*.
37. **Litz, Heiner**, H. Fröning, M. Nüssle, and U. Brüning (2008). VELO: A Novel Communication Engine for Ultra-low Latency Message Transfers. In: *37th International Conference on Parallel Processing (ICPP'08)*.
38. Fröning, H., M. Nüssle, D. 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