

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 IPR2021-01350

U.S. Patent No. 11,740,801

**DECLARATION OF DR. HEINER LITZ IN SUPPORT OF PETITION
FOR INTER PARTES REVIEW OF U.S. PATENT NO. 11,740,801**

TABLE OF CONTENTS

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
V.	Overview of the '801 Patent.....	11
A.	Priority	11
A.	Prosecution History.....	11
B.	Specification	12
B.	The Challenged Claims	14
VI.	Level of Ordinary Skill in the Art	17
VII.	Overview of Prior Art References.....	18
1.	Sinclair.....	18
2.	Yamada.....	25
3.	Terasaki	26
4.	Olbrich.....	29
5.	Sinclair-II.....	30
6.	Yu	32
VIII.	Invalidity of Claims.....	33
A.	Ground I.....	33
1.	Motivation to combine Sinclair and Yamada	33
2.	Claim 1	36
3.	Claim 2	58
4.	Claim 3	61
5.	Claim 4	63
6.	Claim 6	66
7.	Claim 9	66
8.	Claim 10	71
9.	Claim 11	72

10. Claim 12	73
11. Claim 13	74
12. Claim 14	75
13. Claim 15	76
14. Claim 16	78
15. Claim 21	79
16. Claim 24	85
B. Ground II.....	86
1. Motivation to combine Terasaki and Yamada.....	86
2. Claim 1	88
3. Claim 9	106
4. Claim 13	109
5. Claim 14	110
6. Claim 15	111
C. Ground III	113
1. Claim 22	113
D. Ground IV	115
1. Claim 23	115

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. 11,740,801 (the “’801 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 ’801 patent. Specifically, I have been asked to provide my opinion regarding whether claims 1-4, 6, 9-16, and 21-24 of the ’801 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 analyzing the prior art without the benefit of hindsight. A POSA is presumed to be one who thinks along the lines of conventional wisdom in the art.

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

16. I understand that factors considered in determining the level of ordinary skill in the art at the time of an invention include (1) the type of problems encountered in the art; (2) the prior art solutions to those problems; (3) the educational level of active workers in the field; (4) the

rapidity with which innovations are made; and (5) the sophistication of the technology. I further understand that the inventors' specific levels of skill are not relevant to this inquiry.

B. Claim Construction

17. I have been informed by counsel that the Patent Trial and Appeal Board ("PTAB") applies the same claim construction standard used in district courts, where the claims are given their ordinary meaning as understood by one skilled in the art at the time of the invention, informed by the claim language itself, the specification, and the prosecution history. I also understand that "extrinsic evidence"—*i.e.*, evidence other than the patent and prosecution history—can be relevant in determining how a POSA would understand terms of art used in the claims. I have been informed, however, that extrinsic evidence may not be used to contradict the meaning of the claims as 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 *reclaim unit*, *subdivision*, or *zone*, all of 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. If no such erase unit is entirely free of valid blocks, the SSD performs *garbage collection*. During this process, the SSD itself rewrites valid blocks to new locations, updates the L2P accordingly, and invalidates the old PBAs. Once all valid blocks have been relocated, the erase unit can be safely erased. Garbage collection involves additional reads and writes, and therefore, should be minimized to improve performance and endurance, as NAND flash has a limited number of program/erase (P/E) cycles.

34. To optimize garbage collection, the SSD maintains metadata to track the number of valid blocks per erase unit. Selecting an erase unit 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 erase units. 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; *see also* EX1007, ¶¶11-12.

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.

41. To increase the efficiency of flash memory, SSDs may employ a larger unit of memory. For example, SSDs may be organized into groupings of blocks operated as a single unit, termed, *e.g.*, “super blocks,” EX1012, 6:43-65, “metablocks,” EX1004, ¶5, EX1017, 6:1-7, or “zones,” EX1013, ¶¶41-12. Moreover, to increase parallelism, blocks of data in a grouping may be written or erased together, including blocks that are on different planes. EX1011, §2.2 (“Multi-plane command activates multiple read, program or erase operations in all planes of the same die”), Fig. 4.

V. OVERVIEW OF THE ‘801 PATENT

A. Priority

The ’801 patent issued on August 29, 2023, and claims priority to Provisional Application 61/757,464, filed on January 28, 2013. The prior art references relied upon in this Petition predate January 28, 2013, the earliest possible priority date.

A. Prosecution History

The nonprovisional application that became the ’801 patent was filed on January 5, 2022.

On December 14, 2022, the examiner rejected claims 2, 5-7, 11-12, 14, and 17-15 as obvious over U.S. Patent Publication No. 2004/0083335 (“Gonzalez”) in view of U.S. Patent Publication No. 2011/0161560 (“Hutchinson”) and claims 3-4, 10, 13, and 15-16 as obvious over Gonzalez in view of Hutchinson and in further view of U.S. Patent Publication No. 2005/0073884 (“Gonzalez’2005”). EX1002. The remaining claims, Claims 8 and 9, were also rejected as obvious in light of Gonzalez in view of Hutchinson and in further view of U.S. Patent Publication No.2012/0260150 (“Cideciyan”). *Id.*

On March 10, 2023, applicant filed an Amendment and Request for Reconsideration After Non-Final Rejection. *Id.* Contrasting the identified art, applicant asserted that the claims concern “a memory controller [that] identifies maintenance conditions (e.g. old data, reclaimable memory space, etc.) and instead permits a host to initiate a physical erasure operation,” where the host secures the underlying “information representing the maintenance condition, for example, by reading this information from the storage drive or, in other embodiments, this information can be synchronously transmitted to the host once it has been identified by the memory controller.” *Id.* Accordingly, applicant asserted that Gonzalez and Hutchinson did not render “obvious, the information, metadata, query, and host identification features of the independent claims.” *Id.*

Based on the above reasoning, applicant identified four limitations which it claimed were not taught or rendered obvious by the cited prior art:

in association with the programming of the data accompanying the one of the write requests, update information, for the specific one of the respective subdivisions, representing an extent to which the erase blocks of the specific one of the respective subdivisions are full,

store a value representing a size of the NANO flash memory that is to be physically erased in fulfilling a maintenance request issued by the host,

provide to the host, responsive to a query received from the host, the information, as updated, and the stored value,

...

store information, responsive to detection of the condition, wherein the information is accessible to the host and indicates that the condition was detected;

Id.

There were no further rejections. On April 12, 2023, the examiner issued a Notice of Allowance, stating the “applicant’s reply makes evident the reason for allowance.” *Id.* As discussed below, the prior art in this Petition discloses the limitations that applicant challenged.

B. Specification

The ’801 patent describes a memory system, including a memory controller (in SSD) and a host, where responsibility for memory management is shared between the controller and host, where: “(1) the host maintains a reasonably-accurate understanding of physical location of data in flash memory and is thus capable of directly addressing data to the specific physical locations in memory at which the data will be stored, and (2) many of the operations which might traditionally compete with host read/write requests will instead be managed by the host, i.e., such that tasks are schedule[d] so that there is no competition,” including e.g., maintenance operations. EX1001, 39:42-67; *see also id.*, Abstract.

The specification describes a memory architecture organized into subdivisions, where “each subdivision is an EU or page of NAND flash memory.” EX1001, 26:45-47.

Host memory management responsibility is facilitated by the sharing of certain data by the controller with the host. The specification describes that, “[i]n embodiments where the memory 507 is a NAND flash memory, the storage retains information [metadata] for each EU or physical page of the flash memory.” EX1001, 27:29-33. For example, the controller may provide raw metadata or filtered metadata based on various threshold comparison, *i.e.*, (i) “provide a map to the host indicating relative information (e.g., page utilization) for all subdivisions” or “derived information based... on some threshold,” *id.*, 26:47-54; (ii) “wear metric (e.g., EU erase count) exceeds a programmably-defined value,” *id.*, 34:44-35:21, and (iii) an “age field” metric which tracks “how long it has been since data was written to the erase unit,” *id.*, 39:2-35.

The ’801 patent also describes a means for the controller to share metadata with the host, whereby the controller may provide the metadata in response to “a query” or via “asynchronous command,” based on satisfaction of a condition or threshold. *Id.*, 28:10-59.

The claims contrast the specification in two primary manners. First, despite the specification defining “subdivision” as a single erase block or page (*id.*, 26:45-47 (“each subdivision is an EU or page of NAND flash memory”)), the claims recite a different definition, namely “subdivisions” are “non-overlapping ranges of logical addresses to respective subdivisions of physical storage in the NAND flash memory, [where] each of the respective subdivisions comprising erase blocks.” *Id.*, Claim 1. Second, despite the specification describing metadata for individual erase blocks or page, the claims recite metadata at the subdivision-level, *i.e.*, the “metadata represents a characteristic of the data in the specific one of the respective subdivisions,” where a subdivision is a grouping of nonoverlapping erase blocks. *Id.*, Claim 1.

In sum, as described by the applicant during prosecution, the claimed novelty of the '801 patent concerns “a memory controller identifies maintenance conditions (e.g., old data, reclaimable memory space, etc.) and [] permits a host to initiate a physical erasure operation at a time when those operations will not collide with upcoming data access requests of the host” based on “reading [] information [(representing the ‘maintenance condition’)] from the storage drive or, in other embodiments, this information can be synchronously transmitted to the host.” EX1002.

B. The Challenged Claims

42. Below is a table described the Challenged Claims, including claims 1-4, 6, 9-16, and 21-24.

Claim	'801 Patent
1[pre]	A storage device comprising:
1[a]	NAND flash memory; and
1[b]	a memory controller having logic to cause the memory controller to
1[c]	map non-overlapping ranges of logical addresses to respective subdivisions of physical storage in the NAND flash memory, each of the respective subdivisions comprising erase blocks that must each be physically erased as a unit,
1[d]	receive write requests, accompanying addresses and accompanying data from a host and, with respect to each one of the write requests,
1[d.i]	identify a specific one of the respective subdivisions dependent on which one of the non-overlapping ranges encompasses the address accompanying the one of the write requests,
1[d.ii]	program the data accompanying with the one of the write requests into a next available storage location within the specific one of the respective subdivisions, and
1[d.iii]	in association with the programming of the data accompanying the one of the write requests, update information, for the specific one of the respective subdivisions, representing an extent to which the erase blocks of the specific one of the respective subdivisions are full,
1[e]	store a value representing a size of the NAND flash memory that is to be physically erased in fulfilling a maintenance request issued by the host,
1[f]	provide to the host, responsive to a query received from the host, the information, as updated, and the stored value,

1[g]	update metadata for the specific one of the respective subdivisions, wherein the metadata represents a characteristic of the data in the specific one of the respective subdivisions,
1[h]	with respect to the metadata, as updated, perform a comparison based on the metadata to detect a condition, and
1[i]	store information, responsive to detection of the condition, wherein the information is accessible to the host and indicates that the condition was detected;
1[j]	wherein each said logic comprises at least one of circuitry or instructions stored on physical storage media that, when executed, are to control operation of circuitry.
2[a]	The storage device of claim 1 wherein: the metadata comprises information representing an age-since-write of data stored in the specific one of the respective subdivisions; and
2[b]	the memory controller also comprises logic to, responsive to the identification of the condition, control copying of data associated with an age-since-write greater than a threshold to a new physical storage location in the NAND flash memory.
3[a]	The storage device of claim 2 wherein the maintenance request is an erase request;
3[b]	the logic is to further cause the memory controller to receive a copy request from the host, the copy request directed to the specific one of the respective subdivisions;
3[c]	the copy request is to cause the memory controller to control the copying in a manner responsive to the copy request;
3[d]	the erase request is to cause the memory controller to erase a source location of the data being copied; and
3[e]	the copy request and the erase request are respective requests which are received by the memory controller at different times.
4[a]	The storage device of claim 1 wherein: the memory controller further comprises logic to receive a query from the host;
4[b]	the comparison is between the metadata and a threshold; and
4[c]	the information that indicates that the condition was detected is sent to the host responsive to the query and is to provide to the host with at least one address associated with the specific one of the respective subdivisions.
6	The storage device of claim 4 wherein the at least one address comprises a logical address corresponding to the one of the non-overlapping ranges which has been mapped to the specific one of the subdivisions.
9[a]	9. The storage device of claim 1 wherein: the metadata is a function of data error in the data being characterized; and
9[b]	the condition corresponds to a determination by logic of the memory controller that data should be moved to a new physical location in the NAND flash memory.

10[a]	The storage device of claim 1 wherein the metadata represents a data read frequency associated with the data being characterized; and
10[b]	the condition corresponds to a determination by logic of the memory controller that data should be moved to a new physical location in the NAND flash memory.
11[a]	11. The storage device of claim 1 wherein: the metadata represents a data rewrite frequency of the data being characterized; and
11[b]	the condition corresponds to a determination by the memory controller that data should be moved to a new physical location in the NAND flash memory.
12[a]	The storage device of claim 1 wherein: the metadata comprises information representing an age-since-write of data stored in the specific one of the respective subdivisions; and
12[b]	the memory controller also comprises logic to, responsive to the identification of the condition, copy data associated with the specific one of the respective subdivisions to a new physical storage location in the NAND flash memory.
13	The storage device of claim 1 wherein the logic comprises at least one processor and instructions stored on at least one physical storage medium, wherein said instructions when executed, are to control operations of the at least one processor.
14	The storage device of claim 1 wherein the storage device further comprises circuitry to detect a failure condition of one of the erase blocks in the specific one of the respective subdivisions and to transmit information to the host that identifies that a storage location in the specific one of the respective subdivisions has experienced the failure condition.
15[a]	The storage device of claim 1 wherein: the storage device comprises circuitry to detect a failure condition of one of the erase blocks of the flash memory; and
15[b]	the logic is to cause the memory controller to, for one of the write requests which is addressed to a given one of the respective subdivisions corresponding to the one of the erase blocks for which a failure condition has been detected, transmit to the host a notification of a write failure and logical address information identifying the given one of the respective subdivisions corresponding to the one of the erase blocks for which the failure condition has been detected.
16[a]	The storage device of claim 1 wherein: for each of the write requests, the logic is to cause the memory controller to generate an address offset that designates a storage location within an addressed one of the respective subdivisions; and
16[b]	the logic is to further cause the memory controller to transmit the address offset to the host.
21[a]	The storage device of claim 1 wherein: the NAND flash memory comprises physical planes, wherein the specific one of the

	respective subdivisions comprises erase blocks that are in respective ones of the physical planes; and
21[b]	the logic is further to cause the memory controller to identify a need for maintenance for the specific one of the respective subdivisions dependent on the respective metadata, and an associated logical address, and
21[c]	transmit the associated logical address to the host, wherein an erase request from the host is dependent on the associated logical address which was transmitted to the host, and
21[d]	responsive to the erase request, control erasure of two or more of the erase blocks corresponding to the specific one of the respective subdivisions, including erase blocks in each of at least two different ones of the physical planes.
22[a]	The storage device of claim 1 wherein: the NAND flash memory comprises physical planes, wherein the specific one of the respective subdivisions comprises erase blocks that are in respective ones of the physical planes; and
22[b]	the logic is to cause the memory controller to, in response to the one of the write requests, control writing of the data into the erase blocks that are in the respective ones of the physical planes using a common address offset for each of the respective ones of the physical planes.
23[a]	The storage device of claim 1 wherein the logic is further to cause the memory controller to: in connection with physical erasure of the one or more erase blocks, detect a defect in a given erase block of the one or more erase blocks being erased and, responsive to detecting the defect, to remap each logical address which corresponds to the given erase block to a different erase block in the nonvolatile memory; and
23[b]	direct ones of the write requests from the host, for which the accompanying address designates a logical address which has been remapped responsive to detection of the defect, to a storage location associated with the different erase block
24	24. The storage device of claim 1 wherein the logic is further to cause the memory controller to receive an erase request from the host which is directed to the specific one of the respective subdivisions, responsively control the physical erasure of the one or more of the erase blocks in the specific one of the respective subdivisions and, in association with the physical erasure, modify the information representing the extent to which the one or more of the erase blocks of the specific one of the respective subdivisions are full.

VI. 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.

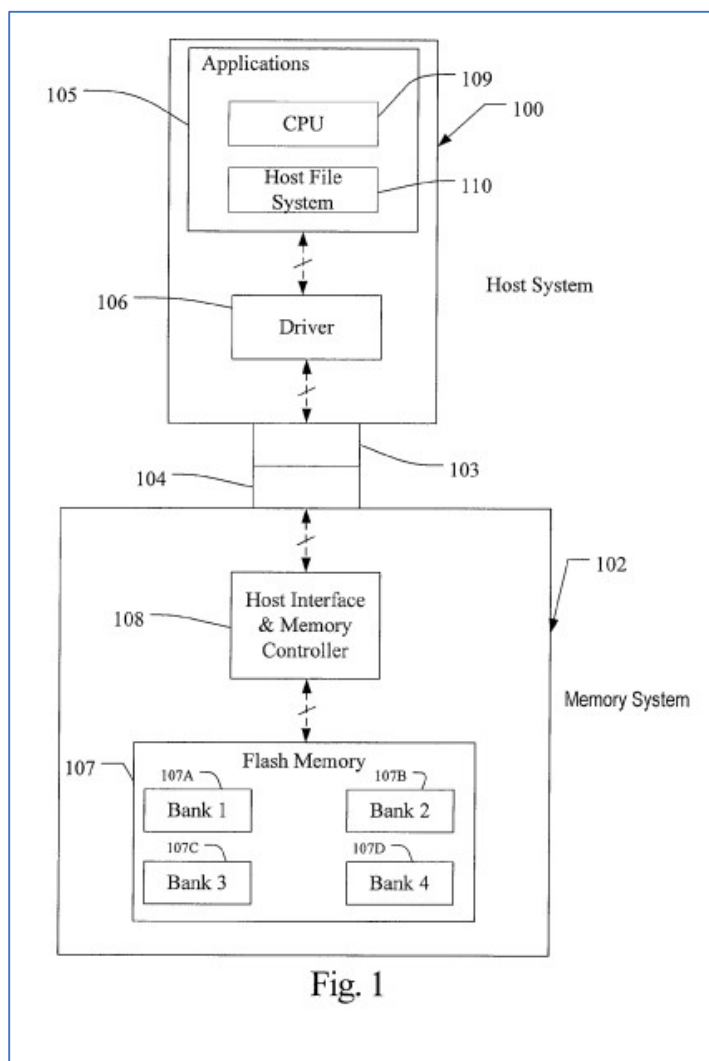
44. Under this formulation, my opinions expressed in this declaration apply.

VII. OVERVIEW OF PRIOR ART REFERENCES

1. Sinclair

45. U.S. Patent Publication No. 2009/0271562 (“Sinclair”) was filed on April 25, 2008 and was published on October 29, 2009. It is prior art under at least AIA 35 U.S.C. § 102(a).

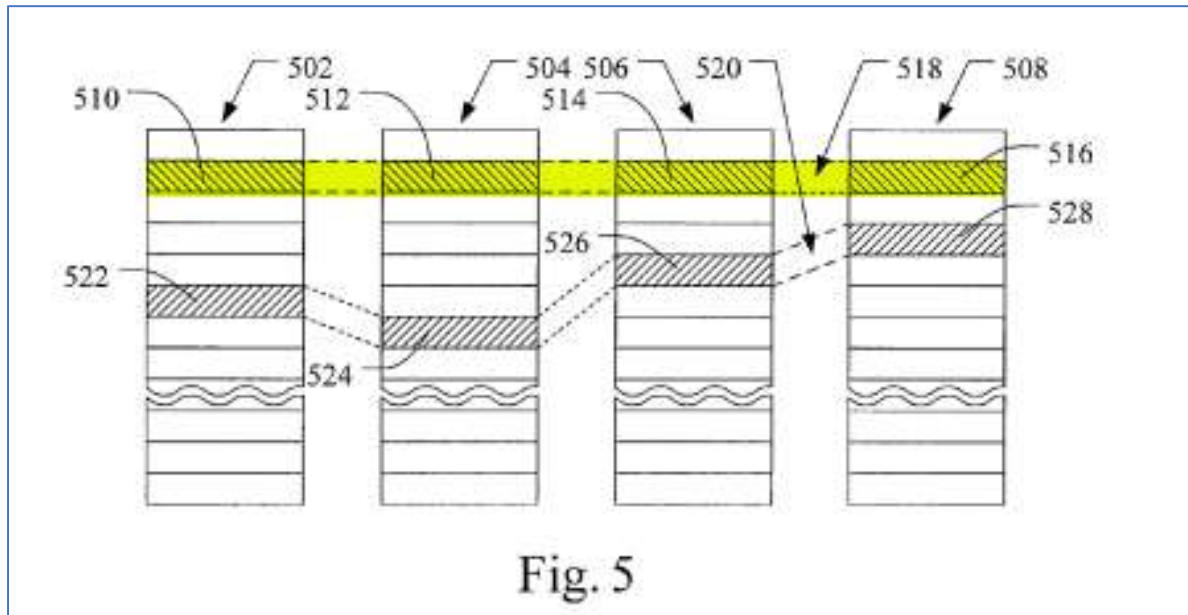
46. Sinclair discloses a host system that communicates with a memory system:



EX1004, Figs. 1-2.

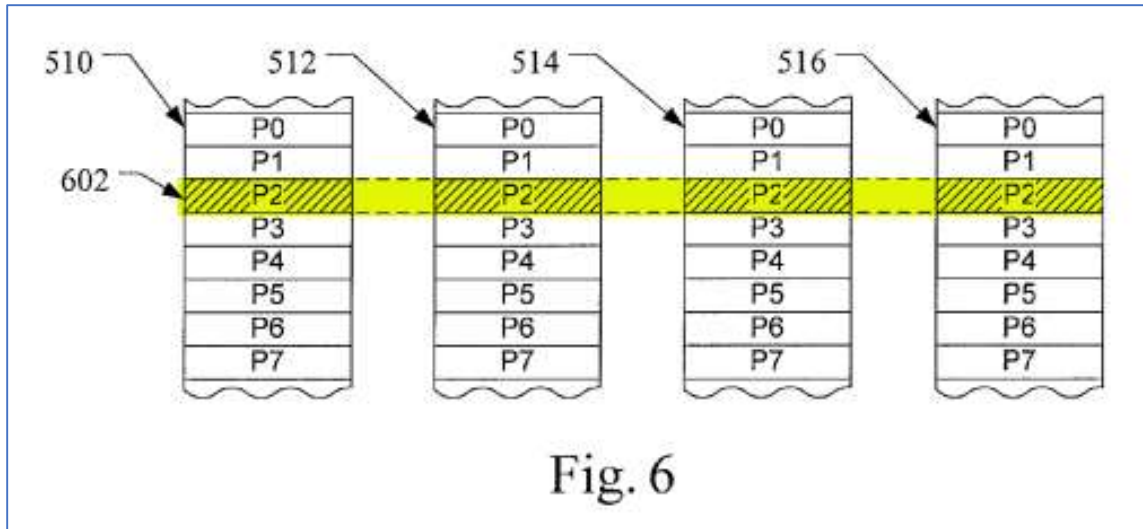
47. The memory system includes a flash memory that comprises a number of memory

banks (107A-107D). EX1004, Fig. 1. Figure 5 illustrates one memory bank (107A). EX1004, Fig. 5, ¶____. To increase parallelism, the flash memory may be organized into “metablocks,” which comprises a group of blocks across multiple planes that may be operated together (e.g., written and erased together). EX1004, Fig. 5, ¶¶53-55. For instance, a metablock (yellow) comprising four individual blocks, 510-516, that are located in respective planes 502-508:



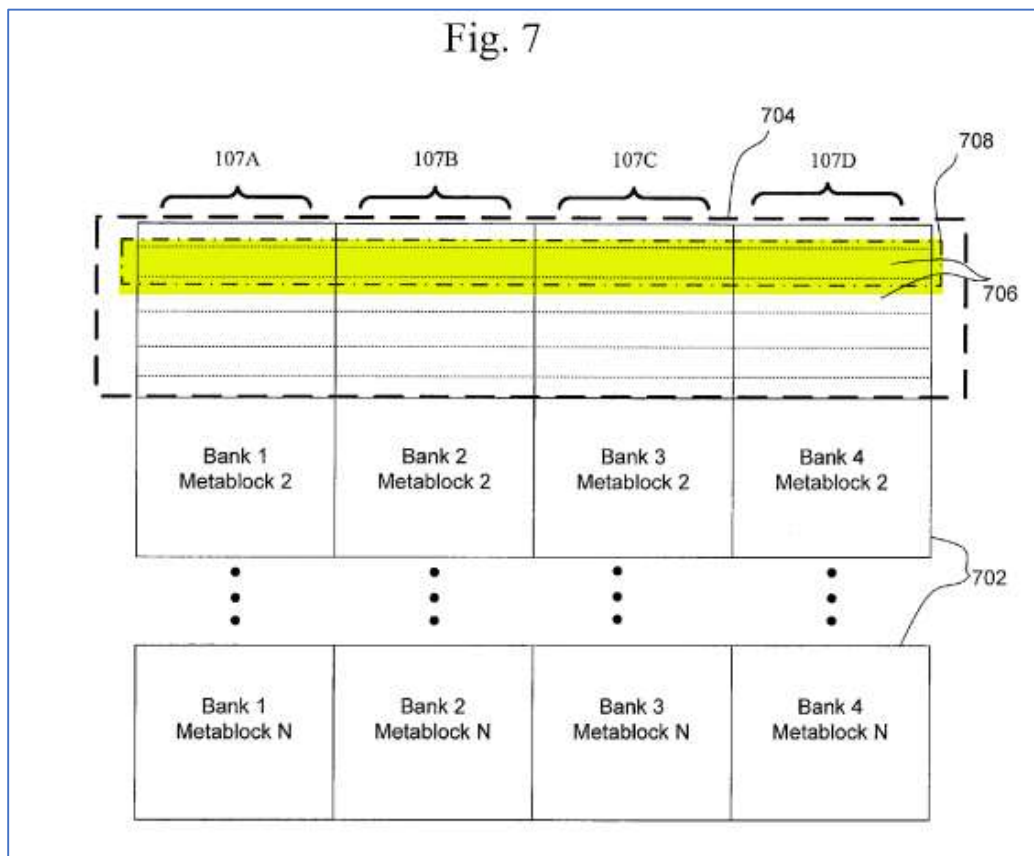
EX1004, Fig. 5, ¶¶53-55.

48. The individual blocks are in turn divided into pages of memory cells. EX1004, ¶63. To further increase parallelism the pages within two block may be logically linked to form a “metapage” (yellow):



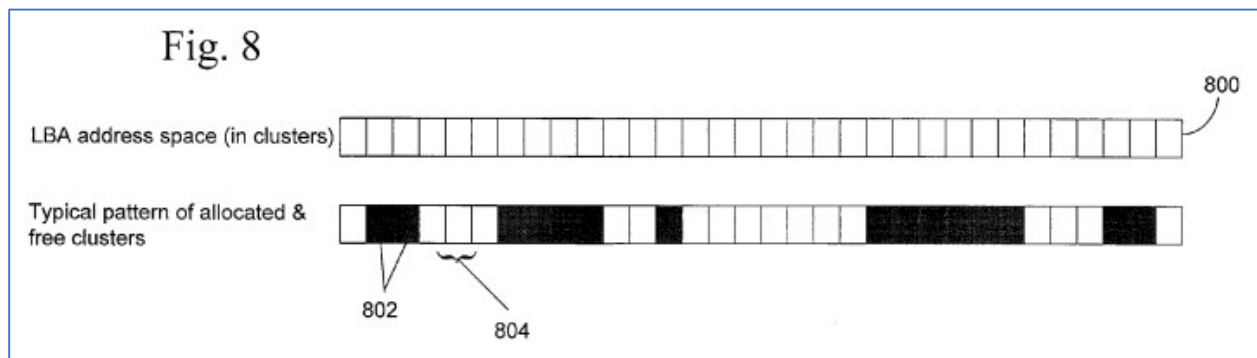
EX1004, Fig. 6, ¶¶55-56, 63.

49. Each bank of the flash memory may contain a number of metablocks, and metablocks from different banks can be grouped into a “megablock” (yellow):



EX1004, Fig. 7, ¶56.

50. Sinclair discloses that the host and memory system may utilize a “continuous logical address space 800.” EX1004, Fig. 8, ¶57. “Referring to FIG. 8, a typical pattern of allocated clusters (shaded) 802 and free clusters (unshaded) 804 in logical address space 800” is shown:



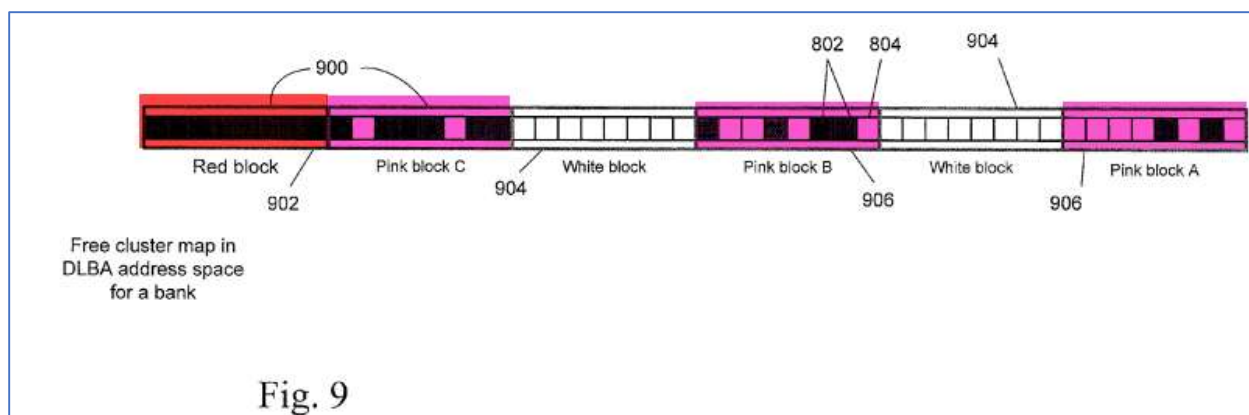
EX1004, Fig. 8, ¶57. Data that is destined to be written to memory is typically received with a “logical block address (LBA),” which in Sinclair are allocated in terms of “metablocks.” *Id.*, ¶¶57-58.

51. Sinclair categorizes metablocks into the following block states:

- Red block (filled with valid data): “blocks 900 completely filled with valid data are referred to as red blocks 902.”
- White block (no valid data): “blocks with no valid data, and thus containing only unwritten capacity, are referred to as white blocks 904.”
- Pink block (valid and invalid data): “Blocks that have been fully programmed and have both valid 802 and invalid (also referred to as obsolete) 804 clusters of data are referred to as pink blocks 906.”

EX1004, Fig. 8, ¶58.

52. Figure 9 of Sinclair illustrates red, pink and white blocks:



EX1004, Fig. 9 (annotated), ¶58. “A megablock 704, which is made up of at least one white block 904” is allocated to receive data from the host and is referred to as a “write megablock.” *Id.*

53. Sinclair discloses a write algorithm and flushing technique for addressing fragmentation of the logical address space. EX1004, ¶58, 63, Fig. 13. In association with the write algorithm, a block information table (BIT) containing lists of block addresses for white, pink and other types of DLBA address blocks is separately maintained by the storage address re-mapping function. *Id.*, ¶82. A BIT page 3002 comprises a white block list (WBL) field 3008, a pink block list (PBL) field 3010, a SAT block list (SBL) field 3012 and an index buffer field 3014, plus two control pointers 3016. *Id.*, ¶127. The BIT page is updated to add or remove entries from the WBL, PBL, and SBL. A BIT block is shown in Figure 30:

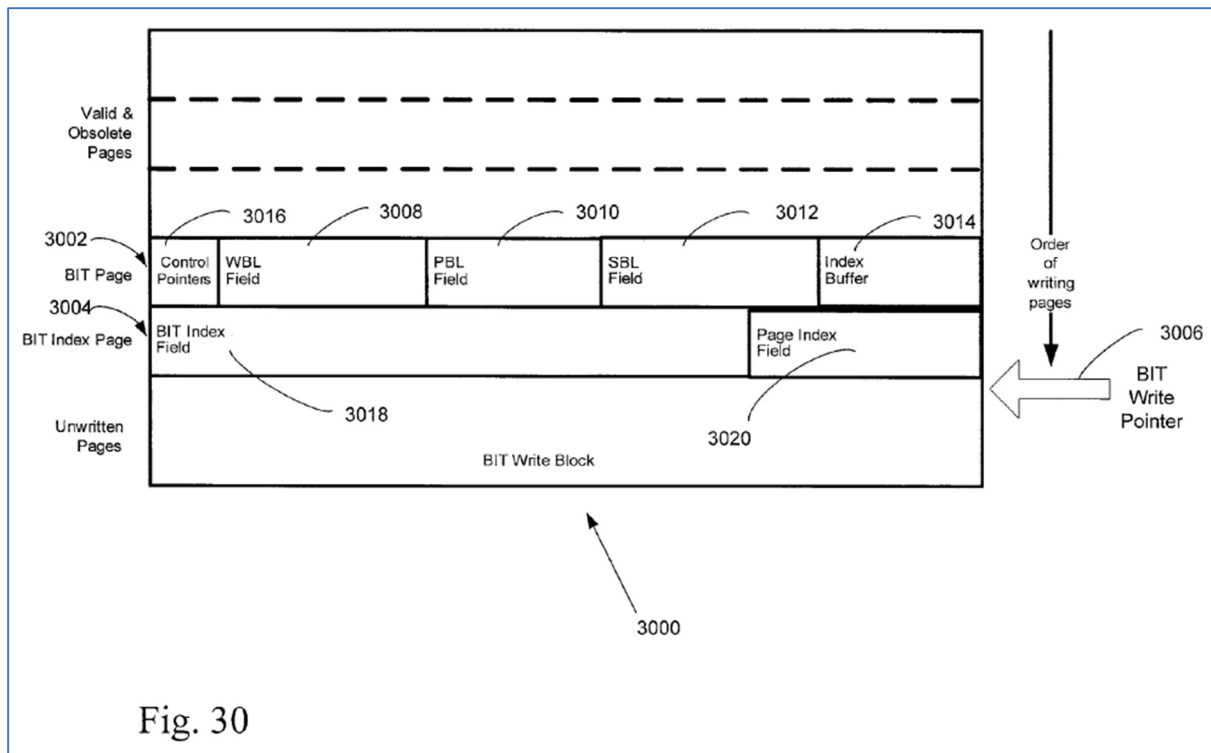
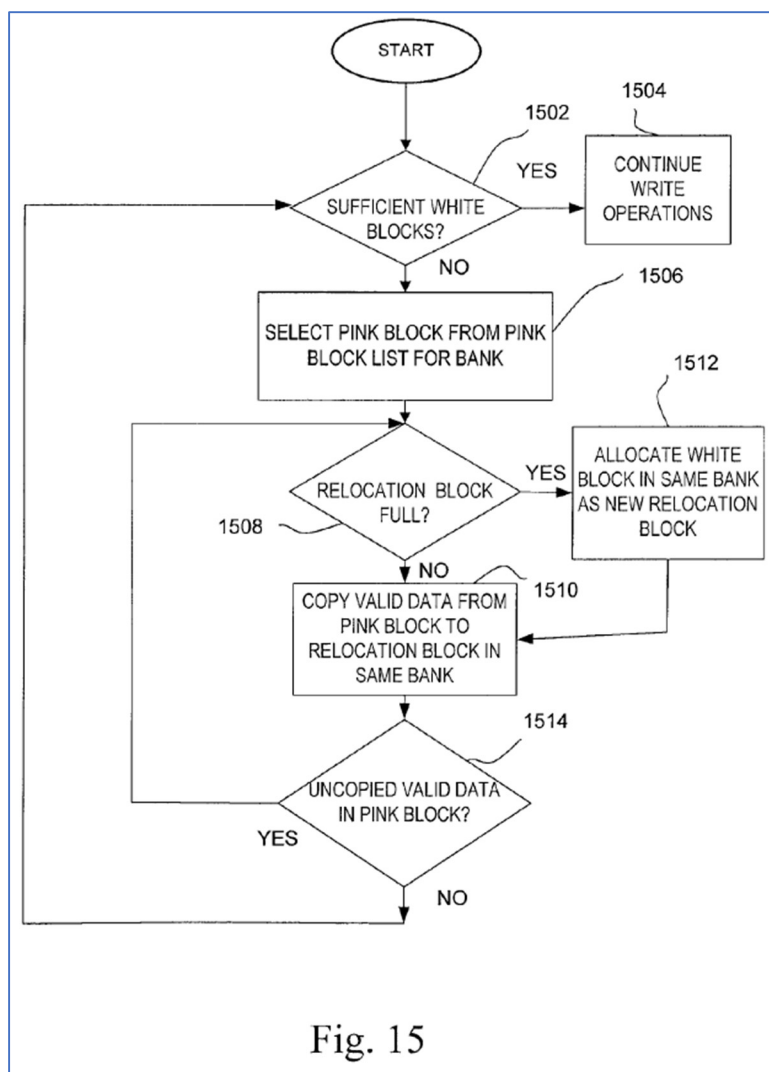


Fig. 30

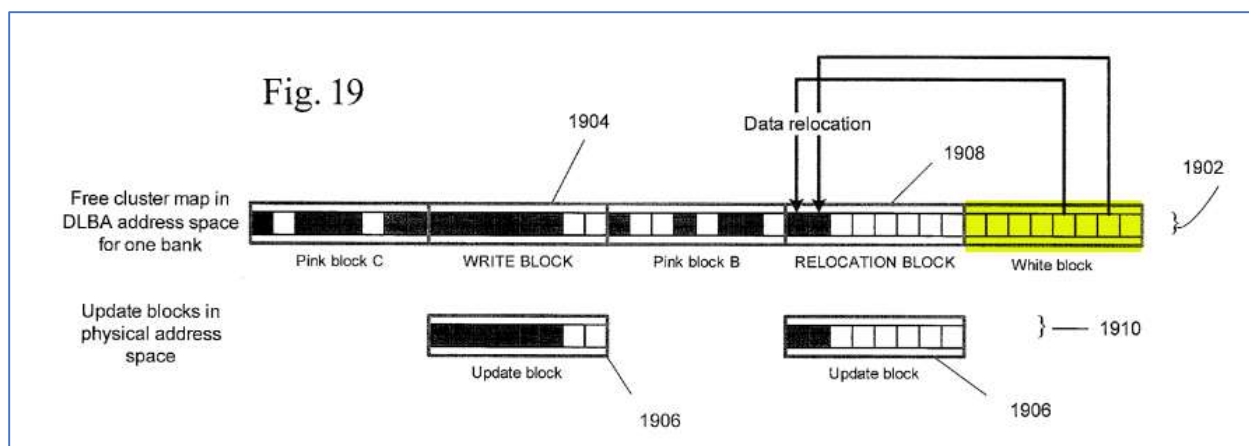
EX1004, Fig. 30.

54. If it is determined that there is an insufficient number of white blocks to accommodate the write request, then an appropriate pink block in the bank is identified and a flush operation is performed. EX1004, ¶67, Fig. 15:



EX1004, Fig. 15, ¶67.

55. Sinclair explains that the flush operation transforms pink blocks into white blocks. EX1004, ¶67, Fig. 15. In order to flush a pink block, the valid data of a pink block is sequentially relocated in an order of occurrence to a white block that has been designated as a relocation block. *Id.* As illustrated in Figure 19, at step 1908 the flush operation creates a white block (highlighted in yellow) by performing data relocation and subsequently physically erasing data from pink blocks.



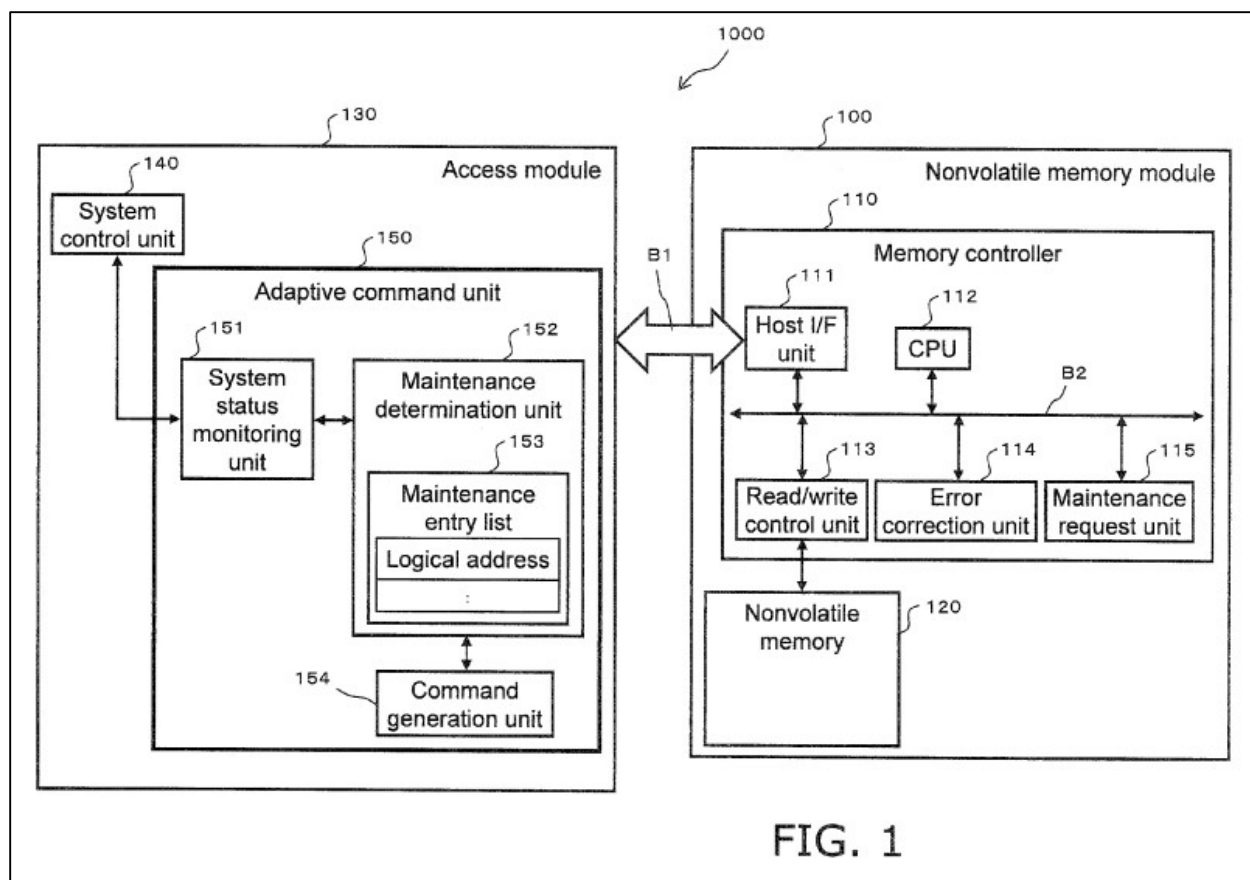
EX1004, Fig. 19 (annotated), ¶76.

56. Sinclair discloses that this flush operation performed in fulfilling a maintenance request may be initiated by a host activity, when a host “flush cache” command occurs, or in response to host commands. *Id.*, ¶112.

2. Yamada

57. U.S. Patent Publication No. 2011/0271032A1 (“Yamada”) was filed on July 27, 2010 and was published November 3, 2011. It is prior art under at least AIA 35 U.S.C. § 102(a).

58. Yamada teaches an SSD system where memory management is shared between the host and controller, where requests for copying data originate from the controller and are commanded by the host. Yamada teaches that the nonvolatile memory device includes a controller with a “maintenance request unit,” and the access device (host) includes a “maintenance command unit.” EX1005, ¶12.



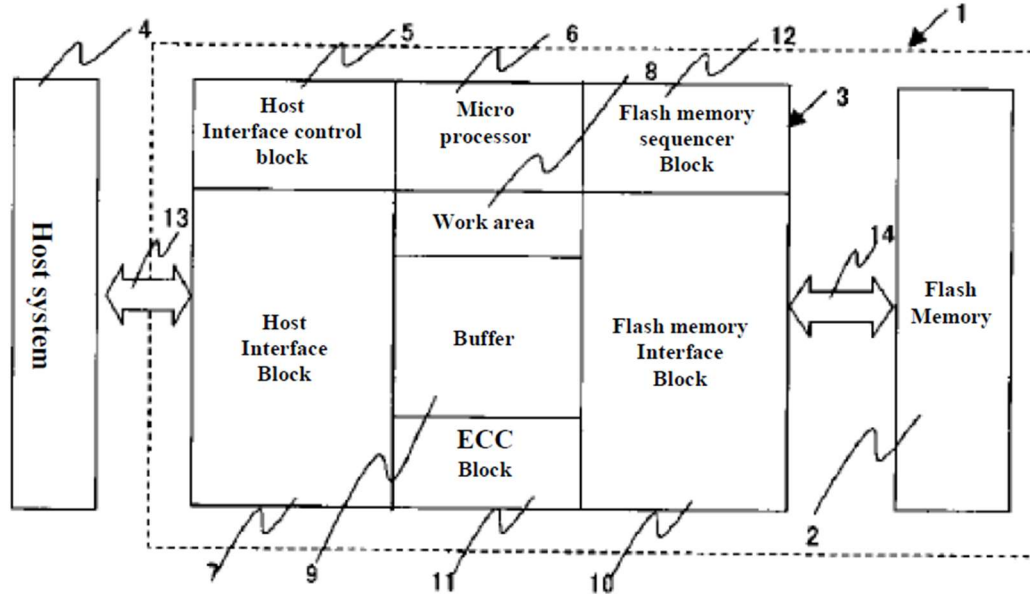
59. The controller identifies an LBA (or group of LBAs corresponding to a set of blocks for maintenance) based on comparison of metadata against thresholds. *Id.*, ¶¶13, 32, 80-81. The controller then transmits the LBA (or groups of LBAs) to the host via a maintenance request, and the host may then issue a maintenance command, e.g., where copying of data from a source location to a destination location “is to be performed.” *Id.*, ¶¶14, 109-111.

3. Terasaki

60. Japan Patent No. JP2005018490A (“Terasaki”) was filed on June 26, 2003 and was published on January 20, 2005. It is prior art under at least 35 U.S.C. § 102(a).

61. Terasaki discloses a flash memory system equipped with a memory controller used to optimize the frequent rewriting of data requested by a host system. EX1018, ¶9, Abstract. Terasaki’s flash memory system is illustrated in Figure 1:

(FIGURE 1)

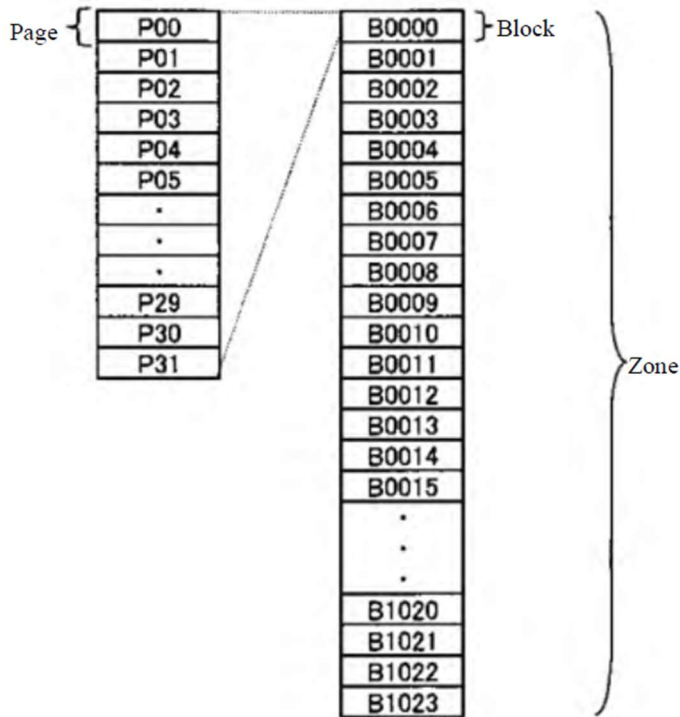


EX1018, Fig. 1.

62. The flash memory system includes a flash memory 2 and controller 3 that controls the flash memory and is attached to a host system 4, such as a personal computer or digital camera, that it interfaces with. EX1018, ¶¶18-19, 21. The controller and host exchange data, address information, status information, and command information via host interface block 7 which includes a task file register storing addressess and commands, and an error register that may be set in the event of an error. EX1018, ¶¶24-25.

63. The flash memory system is organized in zones consisting of a plurality of blocks having consecutive physical block addresses within the memory. EX1018, Abstract, ¶41, Fig. 5. In addition, a zone management table manages the logical block addressess (LBAs) of the data written onto each zone and the data storage status of each zone. EX1018, ¶¶45. Figure 5 illustrates the relationship between the zones, blocks, and pages of the flash memory whereas the block is the unit of erasing; and Figures 6-8 illustrate the zone management table:

(FIGURE 5)



(FIGURE 8)

Zone	Logical block address
Zone 1	LBA10000
Zone 2	Erase request
Zone 3	LBA12000
Zone 4	LBA13000
Zone 5	LBA14000
Zone 6	LBA15000
Zone 7	LBA11000
Zone 8	Erased
Zone 9	Erased
Zone 10	Erased
Zone 11	Erased
Zone 12	Erased
Zone 13	Erased
Zone 14	Erased
Zone 15	Erased
Zone 16	Erased
Zone 17	Erased
Zone 18	Erased
Zone 19	Erased
Zone 20	Erased

EX1018, Figs. 5-8.

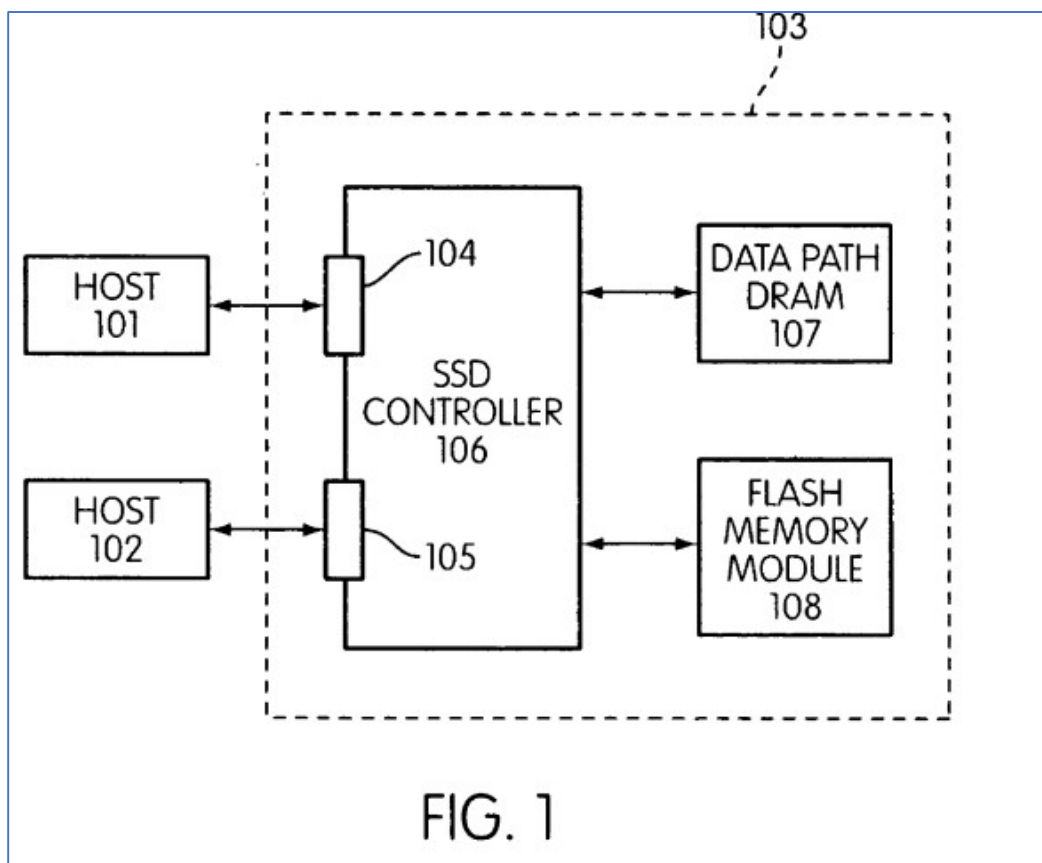
64. For instance, a Zone is illustrated comprising blocks B000 – B1023 with each block comprising multiple pages, such as block B000 comprising pages P00-P31. *Id.* The zone management table stores the state of each zone (write state, erase state, erase request state, etc.). EX1018, ¶49. Information on the data in each zone is also stored, including an erase count. EX1018, ¶¶47-49. The erase count for that zone indicates the number of erases performed for a zone, and is used to manage the order of priority of the erased zones to be used. EX1018, ¶49.

65. The stored erase request flag is a flag indicating that erase processing is necessary to convert the zone to an erased zone. EX1018, ¶47. However, there is no need to perform erase processing each time an update is made to the data. EX1018, ¶59. Instead, zones marked with the erase request flag are erased only during a subsequent zone erase processing, triggered by the write processing of data corresponding to a separate set of LBAs. EX1018, ¶60.

4. Olbrich

66. U.S. Patent 8,621,137 (“Olbrich”) was filed on April 8, 2008, was published on July 2, 2009, and issued on December 31, 2013. It is prior art under at least AIA 35 U.S.C. § 102(a).

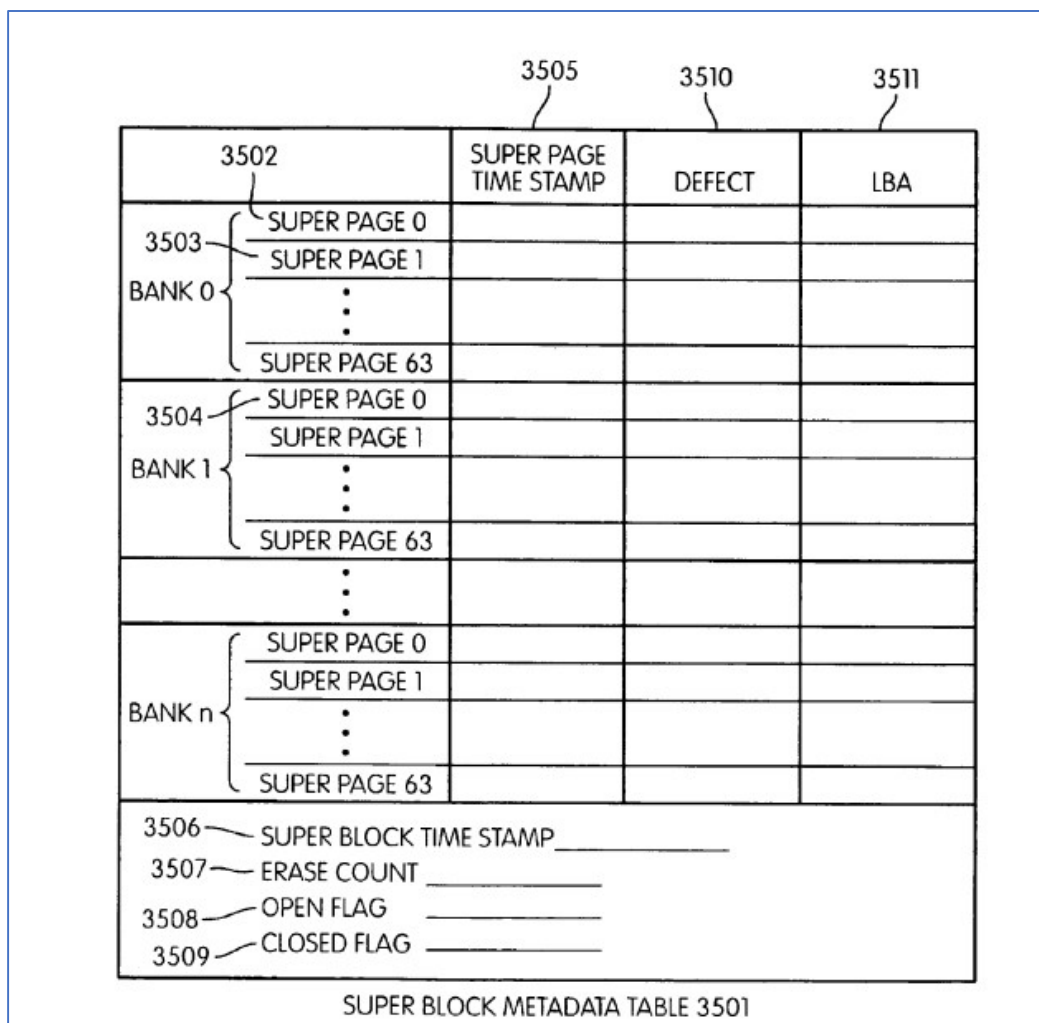
67. Olbrich discloses a storage system comprising an SSD Controller in communication with one or more hosts and a flash memory module:



EX1010, Fig. 1.

68. The storage device is subdivided into superblocks that are comprised of super pages, and the system maintains information about the superblocks in a “superblock metadata table.” An exemplary superblock metadata table is shown below and includes metadata, such as the time when the super page was last written (3505), whether a superpage is defective (3510), the

logical block address of a superpage (3511), an open flag (3508) indicating the superblock is open for writing, a closed flag (3509) and time stamp (3506) that indicates when the superblock was closed for writing, and an erase count for the superblock (3507).



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

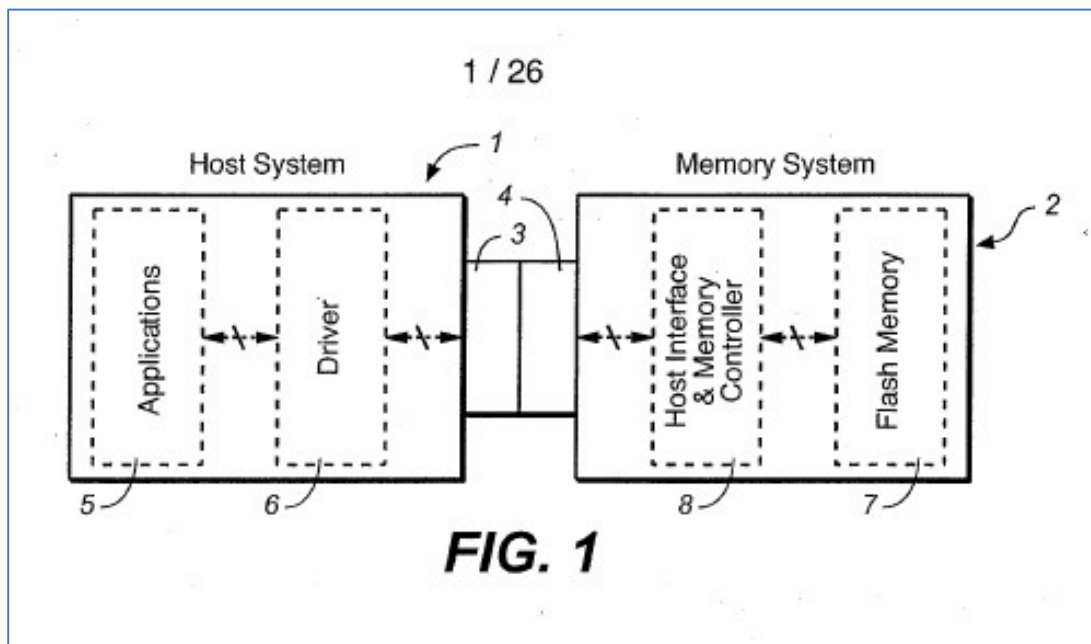
69. Olbrich also discloses that a superpage of a superblock consists of pages across multiple dies (or planes). EX1010, 15:58-67. The host can issue a multi-plane write to these pages using the “same address location.” *Id.*

5. Sinclair-II

70. International Publication No. WO 2008/082996 (“Sinclair-II”) was filed on

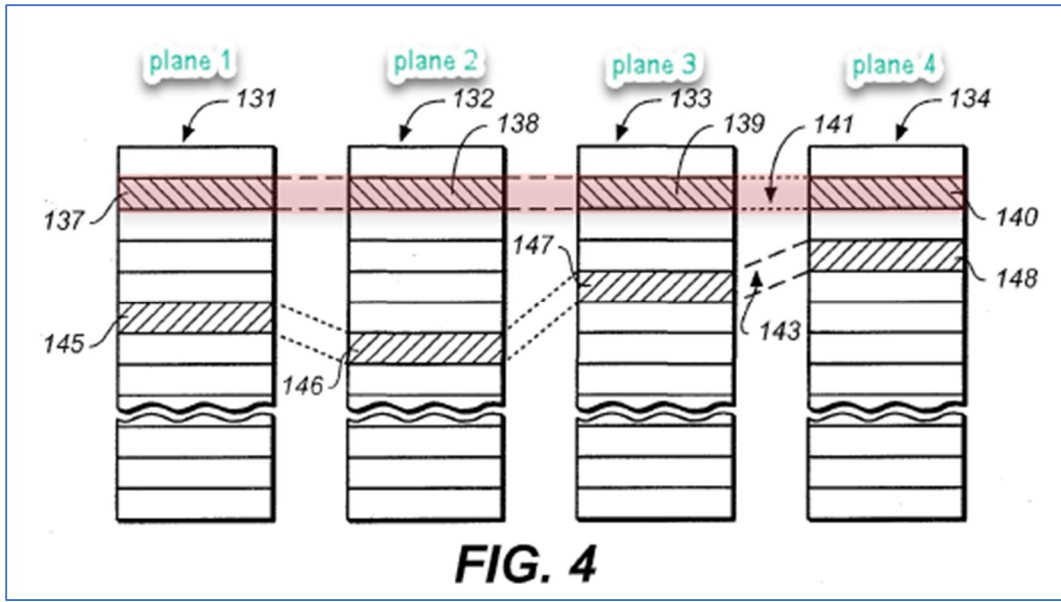
December 19, 2007 and was published on July 2008. It is prior art under at least AIA 35 U.S.C. § 102(a).

71. The Sinclair-II reference is similar to Sinclair. Sinclair-II discloses a host system that communicates with a memory system:



EX1015, Fig. 1.

72. Sinclair-II organizes its flash memory in a similar fashion as Sinclair, for example a multi-plane architecture in which a metablock is comprised of blocks that span multiple planes. EX1015, Fig. 4, ¶62. For instance, Figure 4 illustrates a metablock comprising four individual blocks, 137-140:



EX1015, Fig. 4, ¶62.

73. The individual blocks are in turn divided into pages of memory cells. EX1015, Fig. 4, ¶63.

6. Yu

74. U.S. Pat. Pub. 2012/0284587 (Yu) was filed July 2, 2012 and published November 8, 2012, and is prior art under at least 35 U.S.C. § 102(a).

75. Yu describes an SSD system comprising a host and a controller, where the host and controller work cooperatively to optimize memory storage and to prolong longevity. The system “uses a barrage of advanced management techniques that together reduce the number of writes to flash, hence reducing program-erase cycles on the underlying flash memory,” EX1016, ¶12, “to solve the endurance issues of NAND flash memory,” *id.*, ¶291. Yu describes the use of an “Endurance Translation Layer” to “manage the interaction of DRAM and Flash Memory.” *Id.*, ¶70.

76. Yu teaches the maintenance 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.*, ¶83.

77. Yu describes functionality where the host is able to query the controller for metadata through “Self-Monitoring, Analysis and Reporting Technology” (“S.M.A.R.T”) commands. *Id.*, ¶¶95, 213. The S.M.A.R.T. functionality is a discrete example of shared responsibility between the host and controller to “monitor and control error correction, wear, bad block, and other flash memory management.” *Id.*

VIII. INVALIDITY OF CLAIMS

A. Ground I

1. Motivation to combine Sinclair and Yamada

78. A POSA would be motivated to combine the teachings of Yamada with the system disclosed in Sinclair.

79. As with Sinclair, 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.*, EX1005, [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.”).

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

81. Yamada describes a discrete implementation of the shared host and controller functionality disclosed in Sinclair. 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. EX1005, ¶¶13-15. 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]. Further, Yamada discloses that “maintenance” requests “require[] a certain amount of power consumption” and that “conventional techniques fail to consider the device conditions” when maintenance is performed. EX1005, ¶10. Yamada teaches that to “solve this problem” it is desirable to have the host issue management request, such that the host may consider device conditions when scheduling those commands. EX1005, ¶¶11.

82. Yamada also describes an error correction unit 114 that determines whether the read data has an error and determines the number of errors in the data. EX1005, ¶¶32, 96. Yamada explains that “data that has not been read and has been stored in a memory cell for a long period of time can accumulate errors.” EX1005, ¶6. Thus, Yamada discloses that it is important to track the number of errors and issue maintenance requests to “enhance the data integrity of non-volatile memories.” *Id.* While Sinclair discloses that its memory controller, like Yamada, includes an error correct unit, Sinclair is silent regarding its error correction techniques. A POSA would thus look to other references, like Yamada, that describe these management operations.

83. Thus, in a combined system, Sinclair’s error correction unit would maintain and

update the number of errors in the data, as disclosed by Yamada, and then would notify the host that a maintenance command should issue when the number of errors exceeds a threshold. The combination reflects the application of a known error correction and bad block management technique to improve Sinclair's flash memory device by with error correction and data integrity, so that the combined system tracks the number of errors in the data. Such cooperation of memory maintenance between a controller and a host was known in the art at the time, and other references also teach that a storage system may be improved by having the host issue maintenance requests. For instance, Jung discloses a system for "cooperative memory management" and teaches that moving some or all management functions to the host can increase "execution speed" and avoid the controller becoming "overloaded." EX1007, ¶¶3-5, 7, 41. Similarly, Son explains that a storage device that performs "management operations," such as "address remapping" or "garbage collection," can be improved by having the host issue the management command so that it may "incorporate file system information" and "relieve the SSD controller and firmware of these responsibilities." EX1006, ¶¶13, 63. Thus, Son discloses that a device like Sinclair can be improved by configuring a host to issue maintenance requests, thereby "improv[ing] the performance of the SSD array." EX1006, ¶¶17, 21. Thus, *Sinclair, Yamada, Jung, and Son* all provide an express teaching, suggestion, and motivation for modifying Sinclair so that its flush operation is issued by the host.

84. In light of teaching from Yamada regarding the benefits of shared memory management for maintenance activities and error correction, and Sinclair's teaching that flush operations or STAR address remapping, e.g., maintenance, may be initiated by the host or distributed to a combination of the host and controller, EX1004, ¶112, 65, 68, a POSA would be 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.

85. A POSA would be able to incorporate Yamada's teaching into the combined system of Sinclair 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 host based on information or metadata detected (as in Sinclair) and the information necessary to perform that maintenance is also accessible by the host.

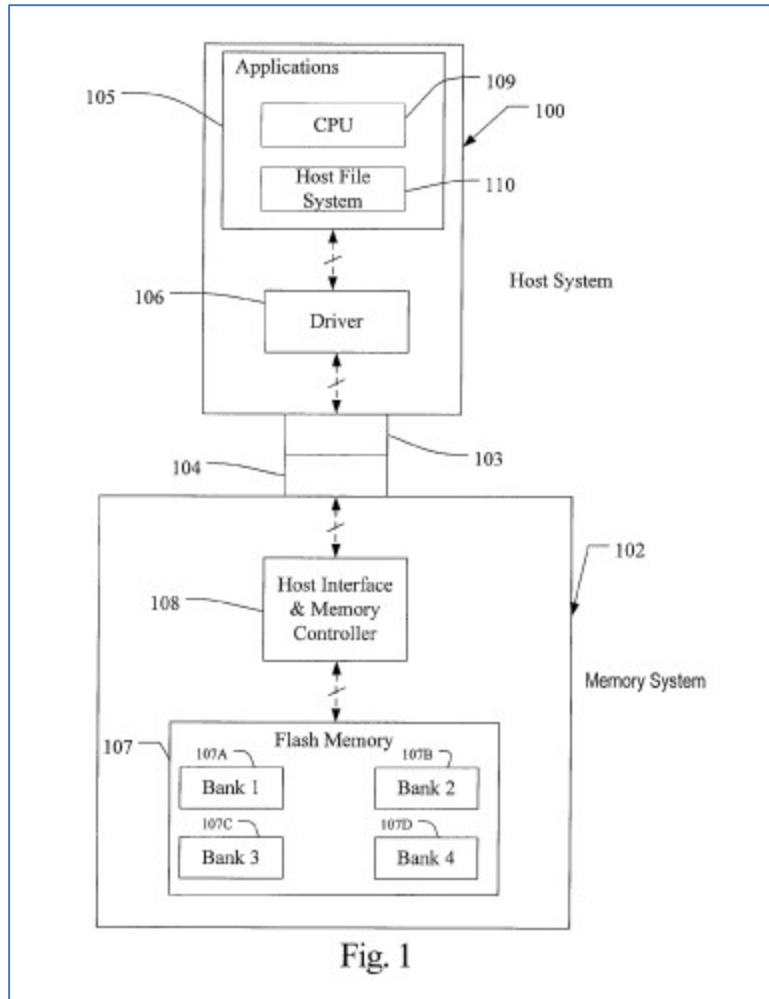
86. A POSA would also have had a reasonable expectation of success in combining Sinclair and Yamada because it represents a straightforward implementation of maintenance distribution and error correction techniques. Indeed, Sinclair's system already includes maintenance activities that Yamada discloses may be initiated by the host, and its flash memory already includes the error correction unit that Yamada discloses can count the number of errors in the data.

2. Claim 1

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

87. It is my opinion that Sinclair renders this limitation obvious.

88. Sinclair discloses a storage device. Figure 1 discloses a memory system comprises a host interface and memory controller 108 and a flash memory 107, such as "in the form of a solid state disk (SSD) drive" installed in a personal computer:



EX1004, Fig. 1, ¶¶40, 43-44.

(b) Claim 1[a]: NAND flash memory; and

89. It is my opinion that Sinclair renders this limitation obvious.

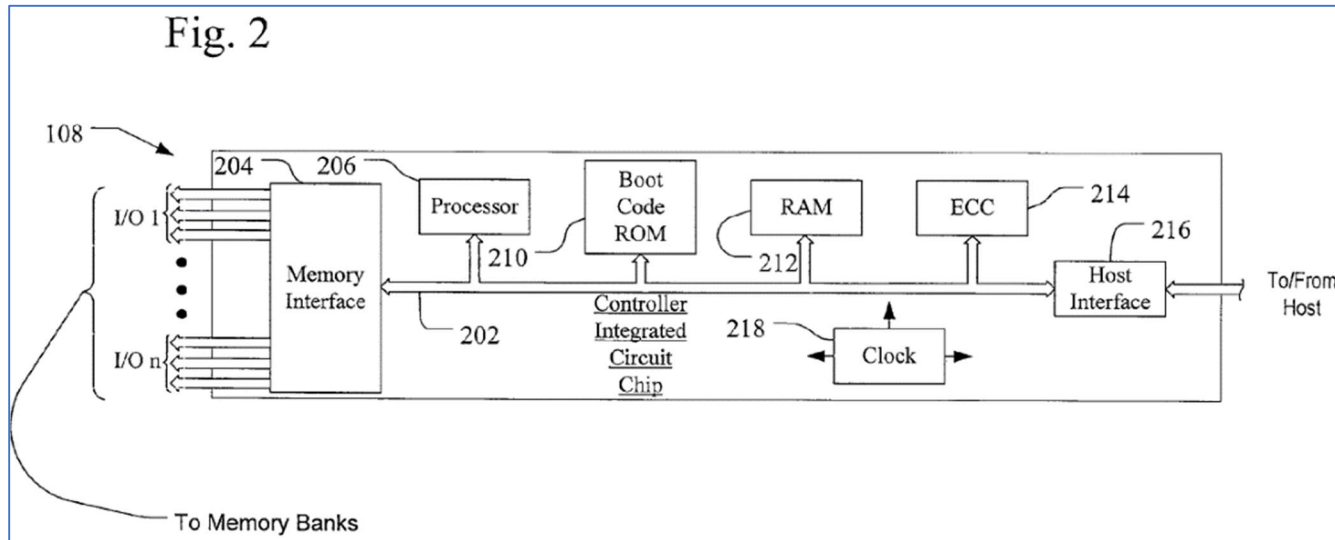
90. Sinclair discloses that its memory system employs a “NAND architecture” and includes “NAND flash memories” or a “NAND array” as part of its memory system. EX1004, ¶49, Figs. 1, 3-4.

(c) Claim 1[b]: a memory controller having logic to cause the memory controller to

91. It is my opinion that Sinclair renders this limitation obvious.

92. Sinclair discloses a “memory controller 108 in the memory system 102” as shown

in Figures 1 and 2:



EX1004, ¶44, Figs. 1-2.

93. The memory controller has logic to cause the controller to perform the claimed limitations, as discussed below, by interfacing with the host to which the memory system is connected for passing data back and forth and by controlling the memory 107. EX1004, ¶44.

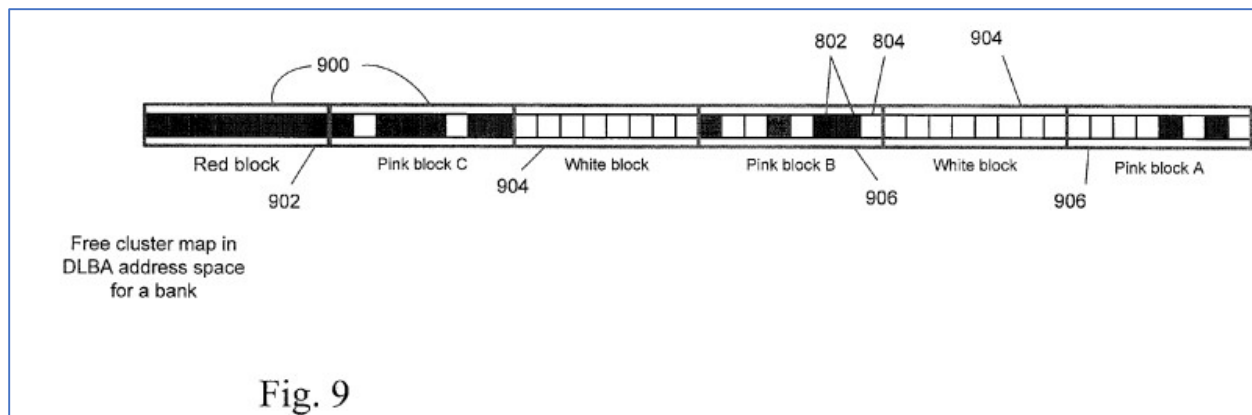
(d) Claim 1[c]: map non-overlapping ranges of logical addresses to respective subdivisions of physical storage in the NAND flash memory, each of the respective subdivisions comprising erase blocks that must each be physically erased as a unit,

94. It is my opinion that Sinclair renders this limitation obvious.

95. Sinclair discloses that its NAND flash memory is organized into “megablocks.” EX1004, ¶¶53-56, 63; Figs. 5-6. The megablocks are comprised of metablocks, which are a grouping of blocks that are “logically linked together” and which can be “erased together.” EX1004, ¶54 (“One block from each plane is logically linked together to form a metablock.... All of the cells within a metablock are typically erased together.”). A POSA would understand that megablocks are subdivisions of physical storage in the NAND flash memory, and because they are logically linked and erased together, that each of the respective subdivisions comprising erase

blocks that must each be physically erased as a unit.

96. Sinclair also discloses that the memory system maps a non-overlapping range of “continuous logical address space 800” to megablocks and metablocks. EX1004, ¶¶5 (“allocating a megablock of contiguous storage LBA addresses for addressing the data associated with the host LBA addresses, the megablock of contiguous storage LBA addresses comprising at least one block of memory cells in each of a plurality of banks of memory cells in the mass storage system.”), ¶¶57-58 (storage system maps “LBA addresses in terms of metablocks of clusters 900, referred to generally as ‘blocks’ in the discussion below.”). For instance, Figure 9 illustrates a contiguous non-overlapping range logical address space mapped to a megablock comprising metablocks (*i.e.*, red blocks, pink blocks, and white blocks):



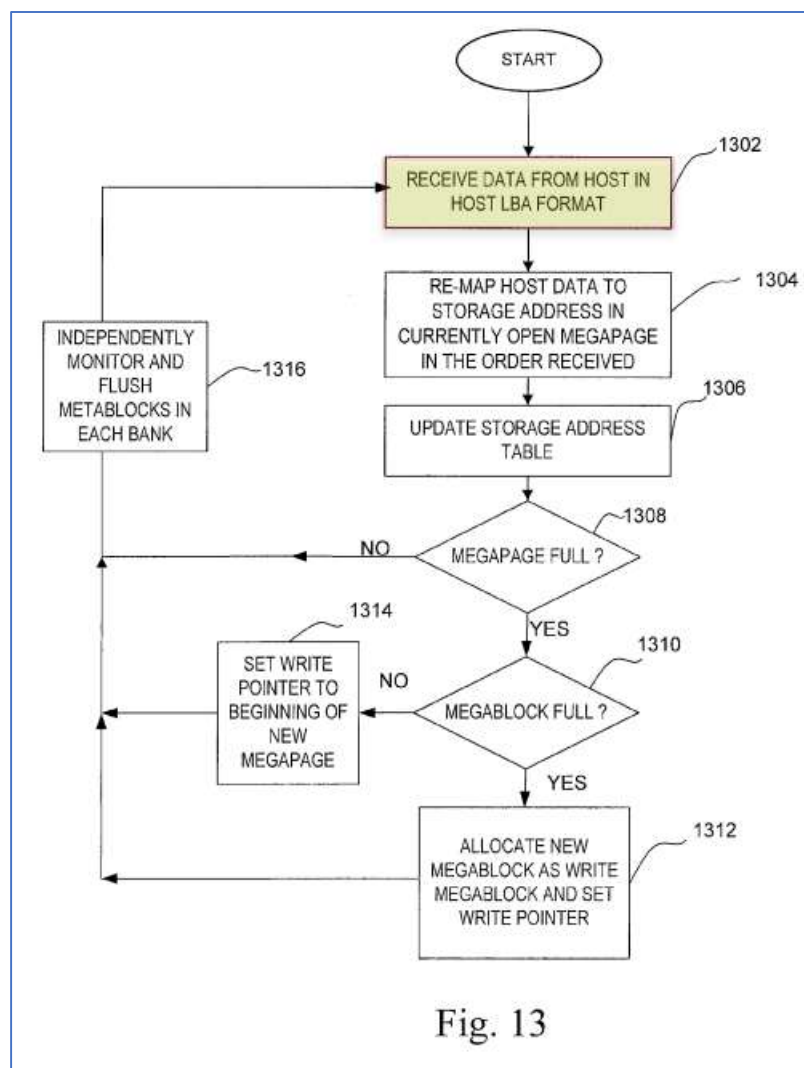
EX1004, Fig. 8, ¶¶57-58, Figs. 8-9.

- (e) Claim 1[d]: receive write requests, accompanying addresses and accompanying data from a host and, with respect to each one of the write requests,

97. It is my opinion that Sinclair renders this limitation obvious.

98. Sinclair discloses its controller receives write requests. For instance, Sinclair discloses a “megablock write algorithm” in which its storage device receives “a host data write operation” or write requests. Ex. 1004, ¶63. Sinclair explains that, as part of the write request, the

storage device “receiv[es] host data from a host file system 10” and a “host LBA [logical block address]” as an accompanying addresses. Ex. 1004, ¶63 (“original host LBA addresses”), 5 (“host LBA addresses”). The host data is then remapped and written to a megapage, which refers to a metapage from each of the metablocks in a megablock. EX1004, ¶¶63, 56. More specifically, “the host data is re-mapped to a storage address by writing the host data to the currently open megapage in the currently open write megablock in the order it is received regardless of host LBA order.” Ex. 1004, ¶63, ¶5 (“receiving data associated with host logical block address (LBA) addresses assigned by the host system....”). This is illustrated in steps 1302-1304 below:

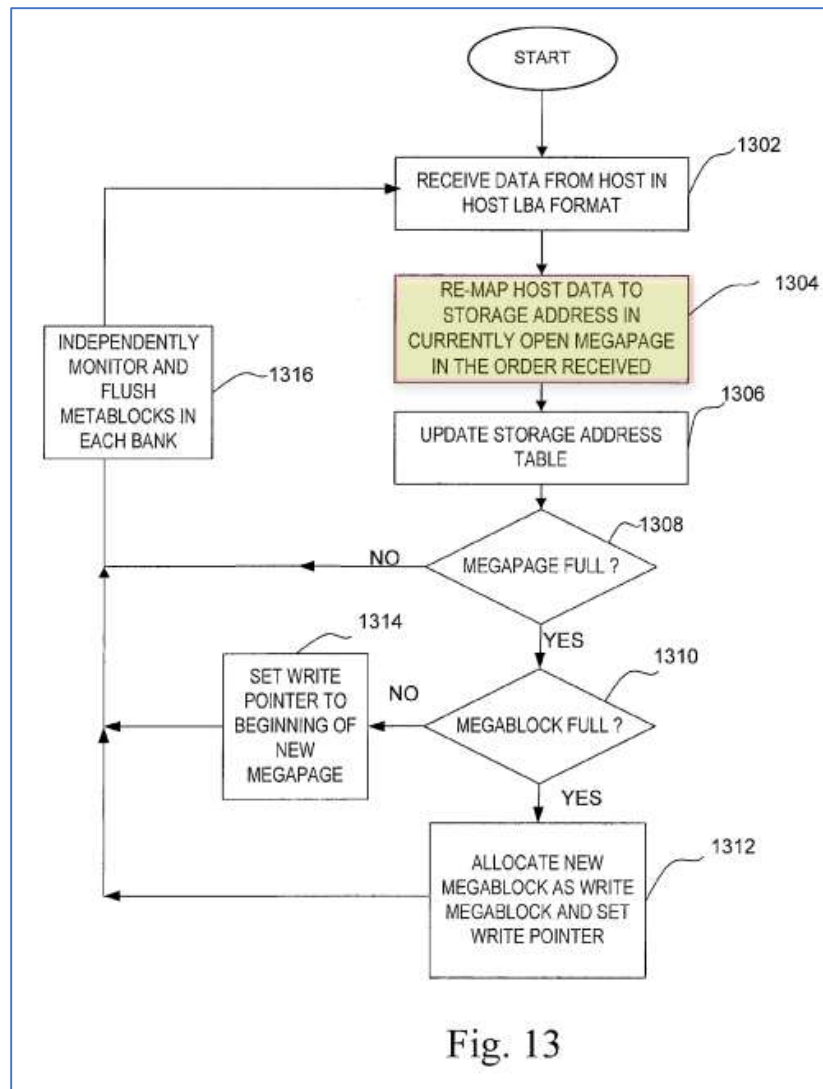


Ex. 1004, Fig. 13.

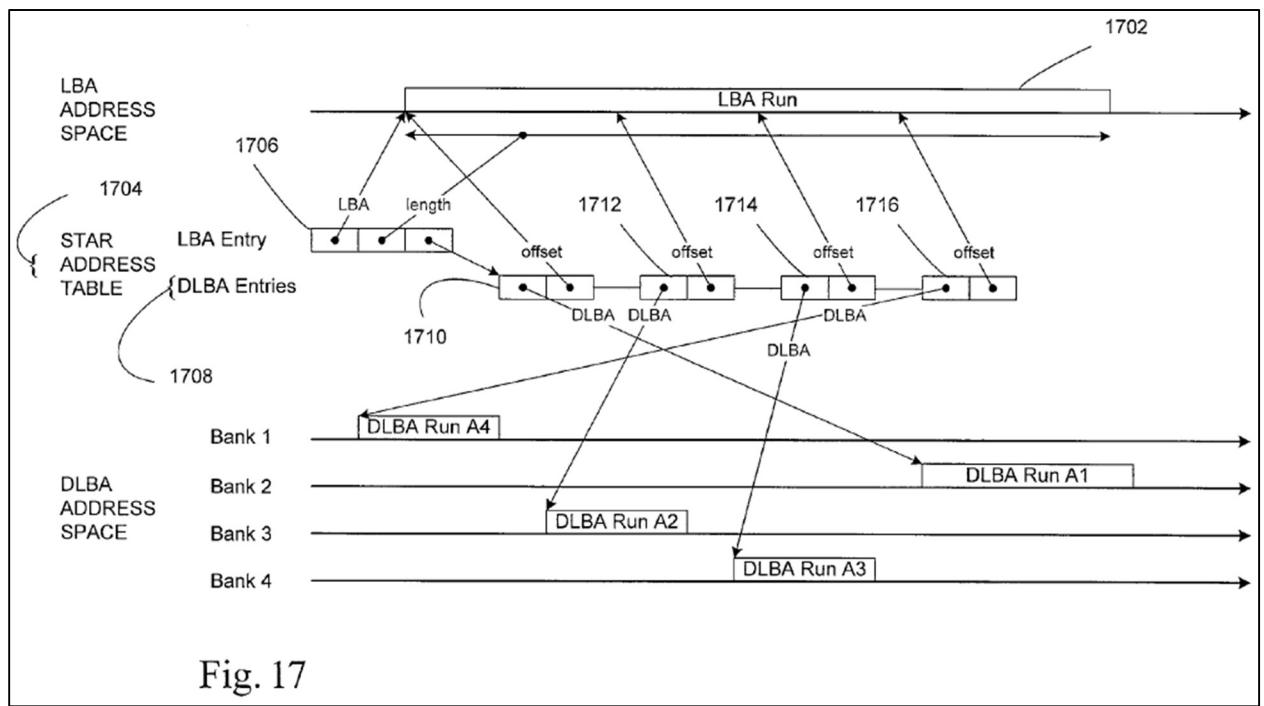
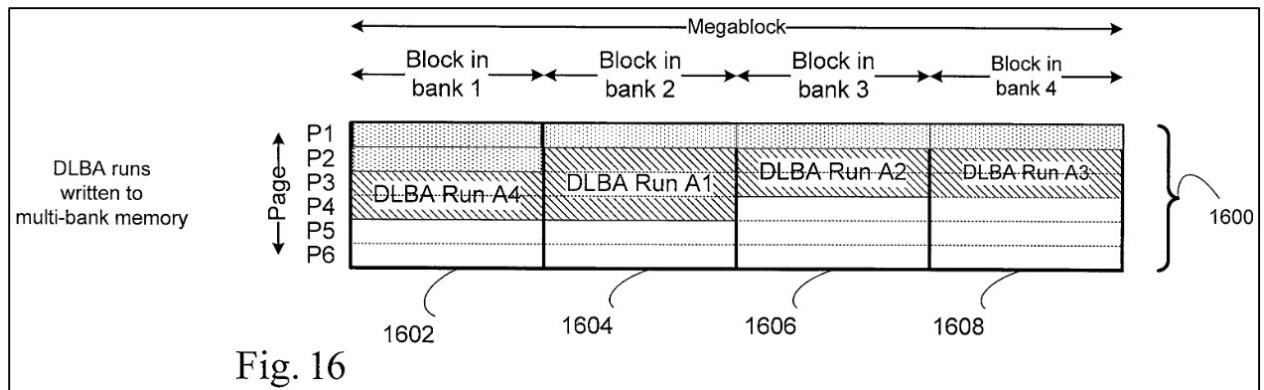
- (f) Claim 1[d.i]: identify a specific one of the respective subdivisions dependent on which one of the non-overlapping ranges encompasses the address accompanying the one of the write requests.

99. It is my opinion that Sinclair renders this limitation obvious.

100. Sinclair discloses identifying a specific one of the respective subdivisions during a write request from the host by identifying a currently open write megablock or subdivision. As illustrated in step 1304 of Figure 13, Sinclair discloses “receiving data associated with a host logical block address” and “[a]s the host data is received, the data is re-mapped to a storage address by writing the host data to the currently open megapage in the currently open write megablock.” EX1004, ¶¶5, 63, Fig. 13.



101. Sinclair also discloses that its controller identifies the open write megablock or respective subdivision dependent on which one of the intermediate storage LBA addresses encompasses the host LBA addresses accompanying the one of the write requests. For instance, with reference to Figure 17, Sinclair discloses that “the host LBA addresses are translated to an intermediate storage LBA address, also referred to as a DLBA address, in an application run by the controller 108 in the memory system 102.... As shown in FIG. 16, the open write megablock 1600 in a four bank memory with metablocks 1602 each having six metapages (P1-P6) is associated with the LBA addresses for the LBA run 1702 shown in FIG. 17.” EX1004, ¶70.



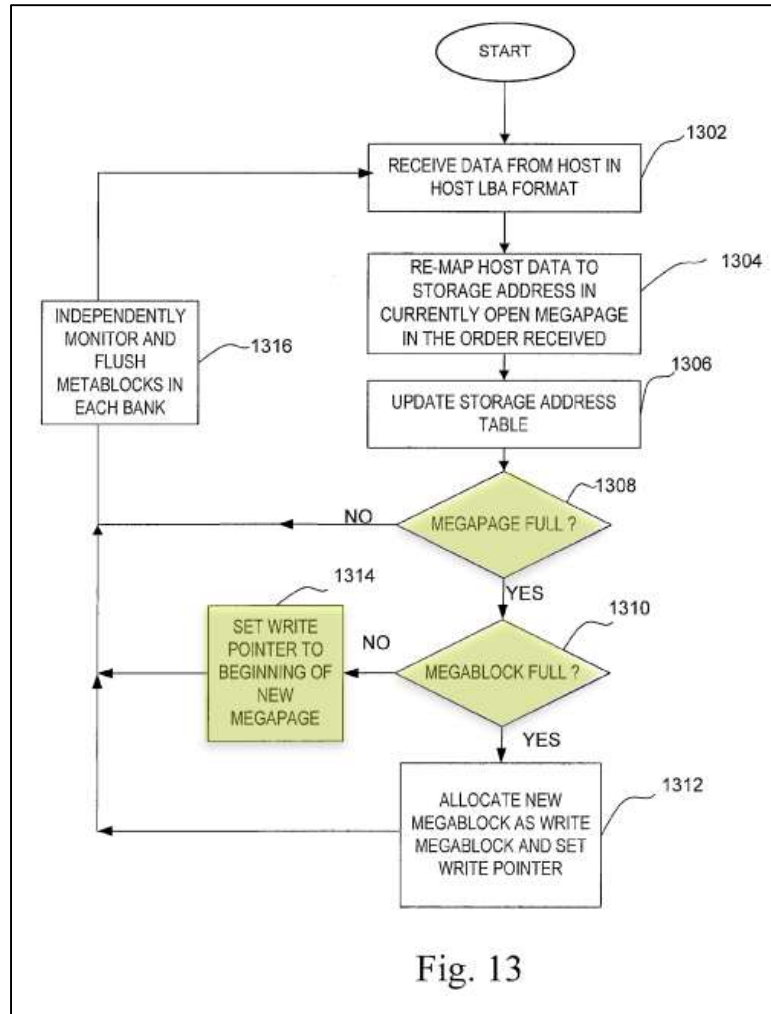
EX1004, Fig. 17; ¶¶23, 70-71. The SAT 1704 maps each LBA run containing valid data to the associated DLBA runs where the LBA entry 1706 in the SAT 1704 includes the first LBA address in the run, the length of the run and the DLBA address and bank identifier of the first DLBA run mapped to the LBA run 1702. EX1004, ¶71. The corresponding DLBA entries 1708 include a first DLBA entry 1710 that has the first DLBA address and bank number of the DLBA run and the LBA address offset in the LBA run 1702 the first DLBA address is mapped to which, in the case of the first DLBA entry 1710 will be zero and in all subsequent DLBA entries for a given LBA run 1702 will be non-zero values. *Id.*

102. The mapping from host LBA addresses is tracked in a “storage address table” (SAT) 1704 that maps each LBA run 1702 containing valid data to the associated DLBA runs and the host data is written to “address locations in the megablock 1600” until the “megablock is fully programmed.” EX1004, ¶72; *see also id.*, ¶5 (“allocating a megablock of contiguous storage LBA addresses for addressing the data associated with the host LBA addresses.”).

- (g) Claim 1[d.ii]: program the data accompanying with the one of the write requests into a next available storage location within the specific one of the respective subdivisions, and

103. It is my opinion that Sinclair renders this limitation obvious.

104. Sinclair discloses that “as the host data is received,” it is remapped (or programmed) “by writing the host data to the currently open megapage” which is the next available storage location in the currently open write megablock or subdivision. Ex. 1004, ¶63. The “host data continues to be re-mapped to contiguous storage addresses in each metapage of the megapage, bank-by-bank, in the order received” until the “current write megablock is fully written.” *Id.* For instance, Figure 13 illustrates the process by which the data is written into the current write megablock with the write pointer being updated to the next available metapage (step 1314) until the megablock is fully programmed (at 1308, 1310, 1312):



Ex. 1004, Fig. 13; *see also id.*, ¶¶63-64, Fig. 14 (“When the last page in the current write block is filled with valid data, the current write block becomes a red block (at step 1404) **and a new write block is allocated from a white block list** (at step 1404) to be part of the next megablock 704.”) (emphasis added). Meaning, the data is written to the next available storage location.

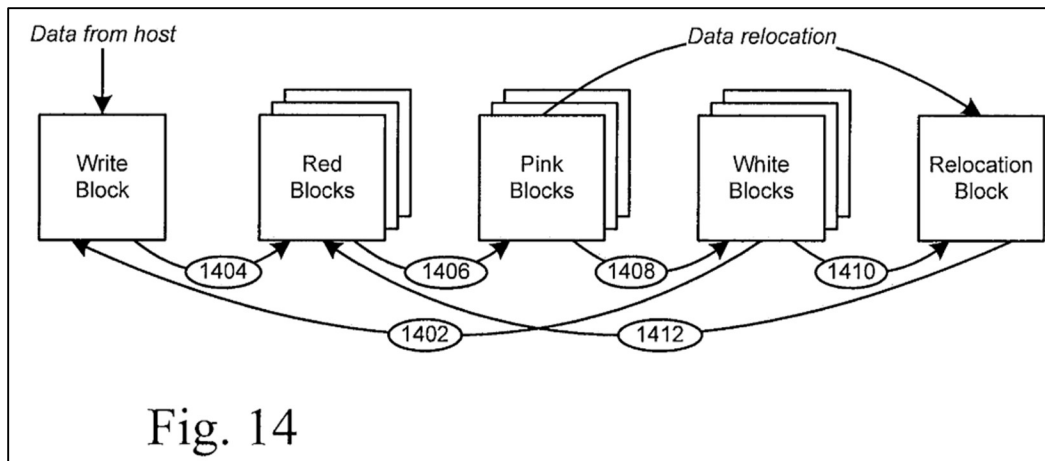


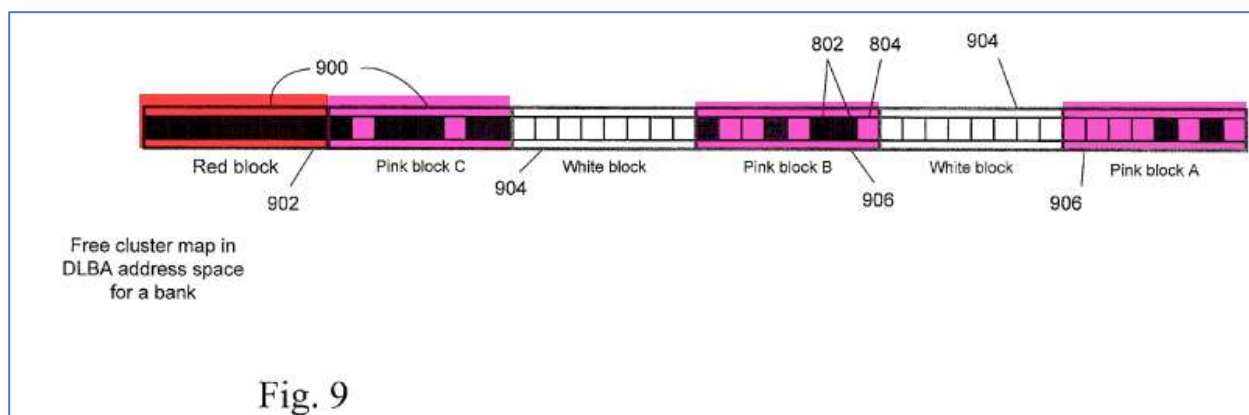
Fig. 14

EX1004, Fig. 14.

- (h) Claim 1[d.iii]: in association with the programming of the data accompanying the one of the write requests, update information, for the specific one of the respective subdivisions, representing an extent to which the erase blocks of the specific one of the respective subdivisions are full,

105. It is my opinion that Sinclair renders this limitation obvious.

106. Sinclair tracks the extent to which metablocks or erase blocks of a specific megablock or subdivision are full, by tracking whether metablocks are “white,” “red,” or “pink.” For instance, a “red block” is a metablock that is “completely filled with valid data,” while a white block is a metablock “with no valid data, and thus containing only unwritten capacity.” EX1004, Fig. 8, ¶58. A pink block is one that has been “fully programmed and have both valid 802 and invalid (also referred to as obsolete) 804 clusters of data.” *Id.* Figure 9 of Sinclair illustrates exemplary red blocks, pink blocks (with valid and invalid clusters of data), and white blocks:



EX1004, Fig. 9, ¶58. As shown, a red block indicates that the metablock is completely full, while the white block indicates that the metablock is empty. *Id.*

107. In addition, a “valid page count value” (or cluster count value) is built to represent the count of valid pages (or clusters) of each block, and is “incremented accordingly” for each identified valid data page located in a pink block set. EX1004, ¶812. A POSA would understand that this list of red, pink and white blocks and their “valid page count” is information representing an extent to which the erase blocks of the specific one of the respective subdivisions are full. This information is maintained by Sinclair, for instance in a block information table (BIT). EX1004, ¶82 (“A block information table (BIT) containing lists of block addresses for white, pink and other types of DLBA address blocks is separately maintained by the storage address re-mapping function.”); *see also id.*, ¶¶67 (“If it is determined that there is an insufficient number of white blocks, then a pink block in the bank is selected...”), 124.

108. Sinclair also discloses that in association with the write algorithm, or the programming of the data accompanying the one of the write request, the memory controller updates the red, white and pink status of the metablocks. EX1004, ¶¶127 (“A BIT page 3002 is the minimum updatable unit of block list information in the BIT), 136 (“A BIT page 3002 is updated to add or remove entries from the WBL 3008, PBL 3010 and SBL 3012”), Fig. 30.

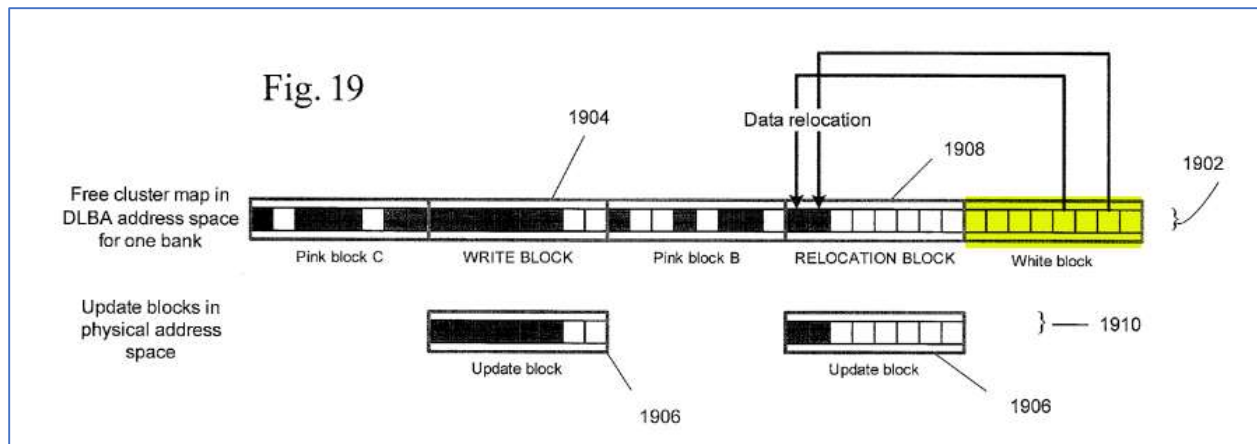
109. Sinclair also discloses that during a write operation a flush operation may be performed if it is determined that there are not sufficient white blocks to accommodate the write request. EX1004, ¶67, Fig. 15. Sinclair explains that the flush operation transforms pink blocks into white blocks. EX1004, ¶67, Fig. 15. A POSA would thus understand Sinclair to disclose that in association with a write command the valid page count value is updated and if the page count value is equal to the total number of blocks in the metablock, the metablock is set to the full (red) state. A POSA would understand this to be information representing an extent to which the erase blocks of the specific one of the respective subdivisions are full.

- (i) Claim 1[e]: store a value representing a size of the NAND flash memory that is to be physically erased in fulfilling a maintenance request issued by the host.

110. It is my opinion that Sinclair and Yamada render this limitation obvious.

111. Sinclair discloses that when the number of white blocks in the system is running low, a flush operation is initiated to transform pink blocks into white blocks. EX1004, ¶67 (“When the storage address re-mapping algorithm detects a need for more white blocks in the bank, the algorithm initiates a flush operation...”), Fig. 15. A POSA would understand this flush operation to be a maintenance request, as described by Sinclair.

112. To fulfill this maintenance request, the memory controller determines and stores the number of pink blocks that need to be relocated, which results in the data being physically erased. EX1004, ¶67 (“In order to flush a pink block, the valid data of a pink block is sequentially relocated in an order of occurrence to a white block that has been designated as a relocation block.”), ¶111 (“in this example, pink blocks selected for flush operations will have 52.4% white capacity”), Fig. 15 (step 1506: “select pink block from pink block list for bank”). As illustrated in Figure 19, at step 1908 the flush operation creates a white block (highlighted in yellow) by performing data relocation and subsequently physically erasing data from pink blocks:



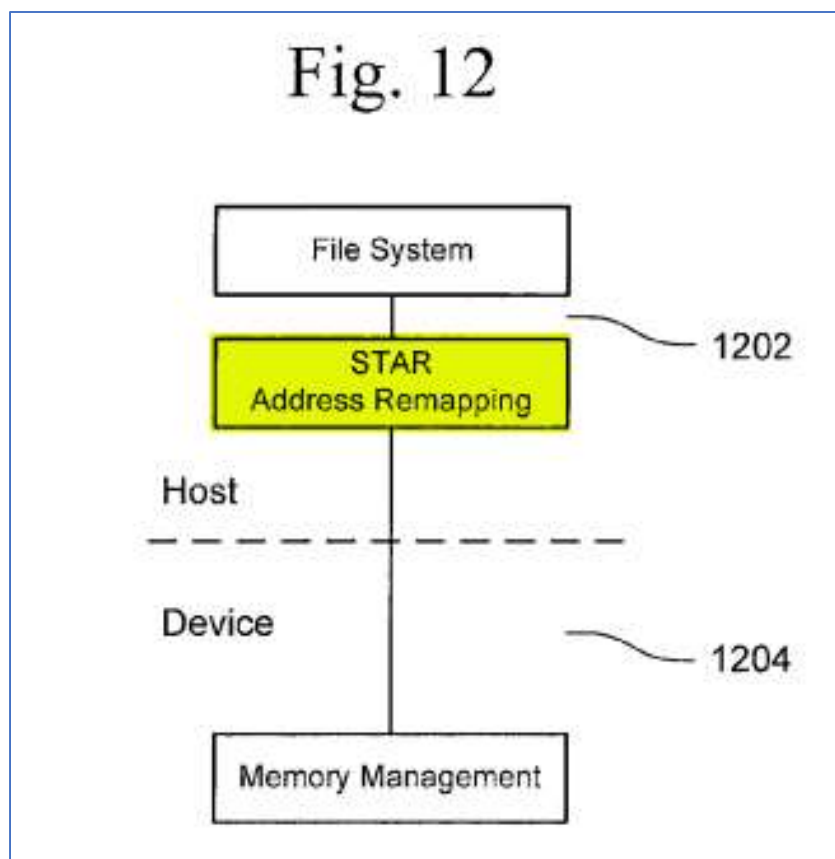
EX1004, ¶76 (“The storage address re-mapping algorithm manages the creation of white blocks by flushing data from pink blocks to a special write block known as the relocation block 1908”).

113. For instance, Sinclair discloses that the memory controller may maintain a list of pink blocks where the “amount of valid data” is below a “threshold,” and those pink blocks would then be erased to fulfill the flush operation. EX1004, ¶67 (“a dynamic list of a defined number (e.g. sixteen) or percentage (e.g. 30 percent) of pink blocks currently satisfying the threshold requirement may be maintained and any pink block may be selected from that list for flushing”). A POSA would understand that each pink block is associated with a memory size such that the storing a listing of pink blocks that are to be flushed discloses storing a value representing a size of the NAND flash memory that is to be physically erased. Indeed, Sinclair discloses that the memory system may provide the host with “information on the block size of physical blocks in the flash memory.” EX1004, ¶61.

114. Sinclair also discloses that the maintenance flush operation may be initiated by the host. For instance, Sinclair discloses that its flush operation performed in fulfilling a maintenance request may be initiated by a host activity, when a host “flush cache” command occurs, or in response to host commands. EX1004, ¶112. In some embodiments, “the flush operation may be utilized in a foreground operation that is active when the host is writing data.” *Id.* A POSA would

understand these disclosures to teach that during foreground operation, the maintenance flush operation is requested or initiated by a “host activity” such as an activity which requires more white blocks to be created, or at minimum suggest that the maintenance request can be issued by the host.

115. In addition, Sinclair explains that “when the storage address re-mapping algorithm detects a need for more white blocks in the bank, the algorithm initiates a flush operation.” EX1004, ¶65. In some embodiments, “re-mapping (STAR) functionality” may be contained “within the host” (EX1004, ¶68), as shown below:

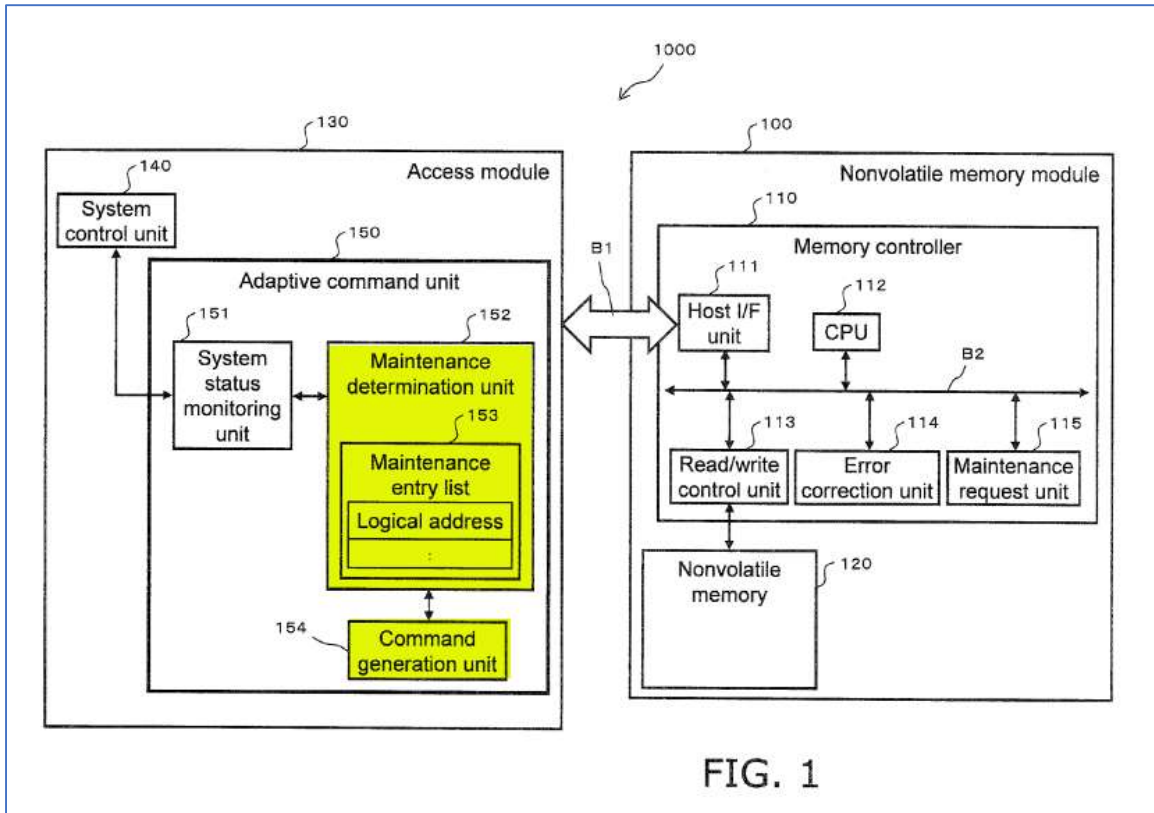


EX1004, Fig. 12.

116. Sinclair discloses that in the host-issued flush operation embodiment, the host and memory system may “exchange information on the block size of physical blocks” and their status

(EX1004, ¶61), such that the “host [] may select pink blocks” for the flush operation (EX1004, ¶68). A POSA would understand these disclosures to teach that the host may issue the maintenance flush operation, or at minimum suggest that the maintenance request can be issued by the host to the memory controller in various embodiments.

117. To the extent Sinclair does not teach or suggest the flush operation may be initiated by the host, it would have been obvious in further view of Yamada. Yamada discloses moving maintenance operations from the storage controller to the host. EX1005, ¶10 (desirable to perform “maintenance” on the memory controller so that “device conditions” may be considered). For instance, Yamada discloses that a maintenance determination unit in the access module may include a “command generation unit” that “generated a maintenance command for performing maintenance for the address of the nonvolatile memory device selected by the maintenance determination unit.” *Id.*, ¶22. For instance, Figure 1 illustrates the access module [*i.e.*, host] and its “maintenance determination unit” and “command generation unit” that “generates a maintenance command for causing maintenance for data retention, such as refresh, to be performed for an address stored in the maintenance entry list 153”:



EX1005, Fig. 1, ¶¶89-90; *see also id.*, ¶90 (“The maintenance command includes a logical address of a block that needs maintenance (hereafter referred to as a “maintenance-needed block”) stored in the maintenance entry list 153.”).

118. In the combined Sinclair-Yamada system, the host would issue the flush operation for the pink blocks that need to be relocated or copied and then erased.

- (j) Claim 1[f]: provide to the host, responsive to a query received from the host, the information, as updated, and the stored value,

119. It is my opinion that Sinclair renders this limitation obvious.

120. Sinclair discloses that in the host-issued flush operation embodiment, the host and memory system may “exchange information on the block size of physical blocks” and their status (EX1004, ¶61), such that the “host [] may select pink blocks” for the flush operation (EX1004, ¶68); *see also* Ground 1, Limitation 1[e]. A POSA would understand these disclosures to teach,

or at least suggest, that the list of red, pink, and white blocks for a megablock and their valid page count (or information) and the list of pink blocks that are to be flushed (or stored value), is provided to the host issuing maintenance requests in order for the host to successfully perform this task.

121. To the extent Sinclair does not teach or suggest providing the host the information, as updated, and the stored value, in response to a host query, the limitation is at least obvious in view of Yamada.

122. Yamada discloses a maintenance determination unit for performing maintenance of the nonvolatile memory device, where it generates a maintenance command for causing maintenance for data retention, such as refresh, to be performed for an address stored in the maintenance entry list. EX1005, Fig. 1, ¶¶22, 89-90. In the combined Sinclair-Yamada system, the host would need access to the white, pink, and red block list and their valid page count value (or updated information) and list of pink blocks that are needed to be flushed (or stored value) in order to issue the flush operation. Yamada teaches that the controller maintains information about the valid block count and that this information can be retrieved via host queries. EX1005, ¶¶14, 109-111. Yamada also teaches that the host can issue maintenance operations. Thus, it would be obvious for a POSA to combine these techniques, e.g. allowing the host to send a maintenance command to which the controller responds with the number of valid blocks moved by the maintenance command.

- (k) Claim 1[g]: update metadata for the specific one of the respective subdivisions, wherein the metadata represents a characteristic of the data in the specific one of the respective subdivisions,

123. It is my opinion that Sinclair renders this limitation obvious.

124. Sinclair discloses updating metadata for the megablocks or subdivisions. Sinclair discloses that “the selection [of] a pink block for flushing” may be made by selecting a group of pink blocks that are “associated with less than some threshold amount of valid data.” EX1004,

¶69. A POSA would understand that the “amount of valid data” associated with a pink block is metadata and also that pink blocks are associated with a megablock, meaning one of the respective subdivisions. The amount of valid data also represents a characteristic of the data, for example that it is valid.

125. In addition, Sinclair discloses that the “selection of pink blocks” for the flushing operation “could be based on an assumption that data that has survived the longest in the memory is least likely to be deleted.” EX1004, ¶69. A POSA would understand this disclosure to teach, or at least suggest, that Sinclair maintains and updates metadata on how long the data in a megablock has survived. The time the data has survived represents the age of the data since it was last written, which a POSA understand is a characteristic of that data.

126. As another example, Sinclair discloses that the controller determines and maintains metadata on whether the data in the megablock is “cold” or “hot” data. EX1004, ¶69. In this manner, “[t]he flush operation relocates relatively ‘cold’ data from a block from which ‘hot’ data has been made obsolete to a relocation block containing similar relatively cold data.” *Id.*, ¶80. A POSA would understand that the term “hot” and “cold” data refers to how recently or frequently the data has been accessed and represents a characteristic of the data, with hot data being more recently or frequently accessed than cold data. Sinclair expressly teaches that “elements within the SAT data structures are updated [by] the storage-address-remapping write and flush algorithms” which is an update to the metadata. EX1004, ¶113.

- (l) Claim 1[h]: with respect to the metadata, as updated, perform a comparison based on the metadata to detect a condition, and

127. It is my opinion that Sinclair renders this limitation obvious.

128. Sinclair discloses performing a comparison based on the metadata to detect a condition. For instance, Sinclair determines a list of pink blocks that should be flushed based on

the amount of valid data in the pink blocks. EX1004, ¶68. For instance, in one embodiment the amount of valid data is compared to the amount of data in other pink blocks to determine which pink block has the “least amount of valid data.” *Id.* In other implementations, Sinclair determines a list by identifying “pink blocks that fall within the threshold requirement” (*e.g.*, “less than some threshold amount of valid data”). *Id.* Thus, Sinclair discloses performing a comparison to determine whether a pink block is less than some threshold. Based on that comparison, Sinclair detects the condition that the pink block should be marked for a flush operation.

129. As another example, Sinclair discloses that the storage controller may create “separate populations of relatively hot and relatively cold blocks” and then selects “hot block[s] containing the least amount of data” or “‘cold’ data from a block from which ‘hot’ data has been made obsolete” to be flushed. EX1004, ¶80. A POSA would understand that determining whether data is hot or cold involves comparing how frequently the data is accessed to some threshold. EX1008 at 5:13-23 (“If the data access frequency of the migration unit is greater than the threshold, it indicates that the migration unit is frequently accessed and has become ‘hot data’; if the data access frequency of the migration unit is less than the threshold, it indicates that the migration unit is infrequently accessed and is ‘cold data’.”). Thus, Sinclair also discloses performing a comparison based on access frequency metadata to detect the condition that data should be flushed.

- (m) Claim 1[i]: store information, responsive to detection of the condition, wherein the information is accessible to the host and indicates that the condition was detected;

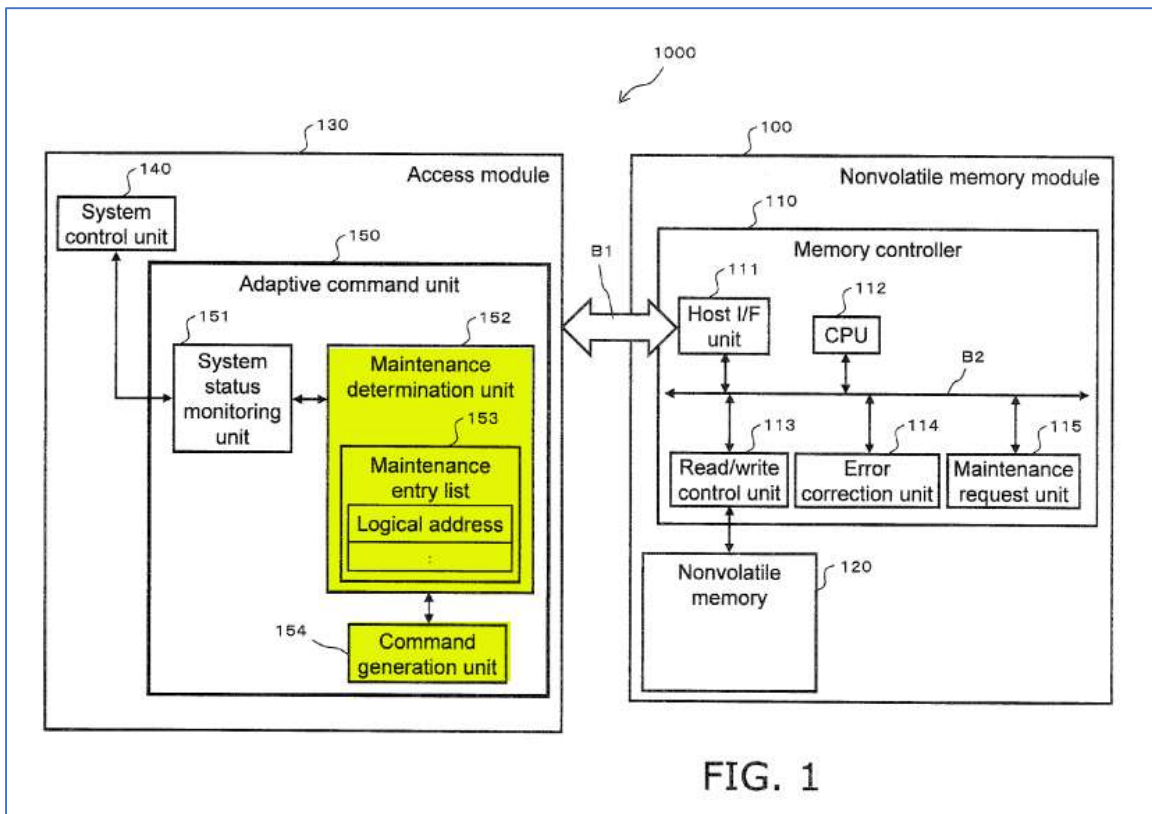
130. It is my opinion that Sinclair renders this limitation obvious.

131. Sinclair discloses storing information, responsive to detection of the condition. For instance, Sinclair detects that a pink block should be added to a list of pink blocks that should be flushed based on the amount of valid data in the pink block. *See* Ground 1, Limitation 1[h]. As

another example, Sinclair discloses detecting “relatively hot and relatively cold blocks,” wherein relatively hot blocks may be added to the list of blocks to be flushed. *See* Ground 1, Limitation 1[h]. The list of blocks is information that is responsive to the detection of the condition because blocks are added to the list based on the detection of the aforementioned conditions.

132. Sinclair also discloses that the information responsive to the detection of pink blocks that need to be flushed or that blocks are cold or hot is accessible to the host. For instance, Sinclair discloses that the host may select pink blocks from the list of pink blocks that meet a threshold requirement and need to be flushed. EX1004, ¶68 (“A subset of the pink blocks at or below the threshold amount of valid data may be maintained in a list from which the host or memory system may select pink blocks.”). A POSA would understand that this detection of pink blocks that need to be flushed is accessible to the host and indicates that the condition (*e.g.*, a pink block falling below a threshold amount of valid data) was detected. At minimum, a POSA would understand that this disclosure suggests that the host may access the list of pink blocks that are below the threshold so that it may then select a pink block to flush.

133. To the extent Sinclair does not teach or suggest that information, responsive to detection of the condition, is accessible to the host and indicates that the condition was detected, the limitation is at least obvious in view of Yamada. Sinclair already discloses that in the host-issued flush operation embodiment, the host and memory system may “exchange information on the block size of physical blocks” and their status (EX1004, ¶61), such that the “host [] may select pink blocks” for the flush operation (EX1004, ¶68). *See also* 1[e]. Yamada discloses moving maintenance operations to a maintenance determination unit on the host:



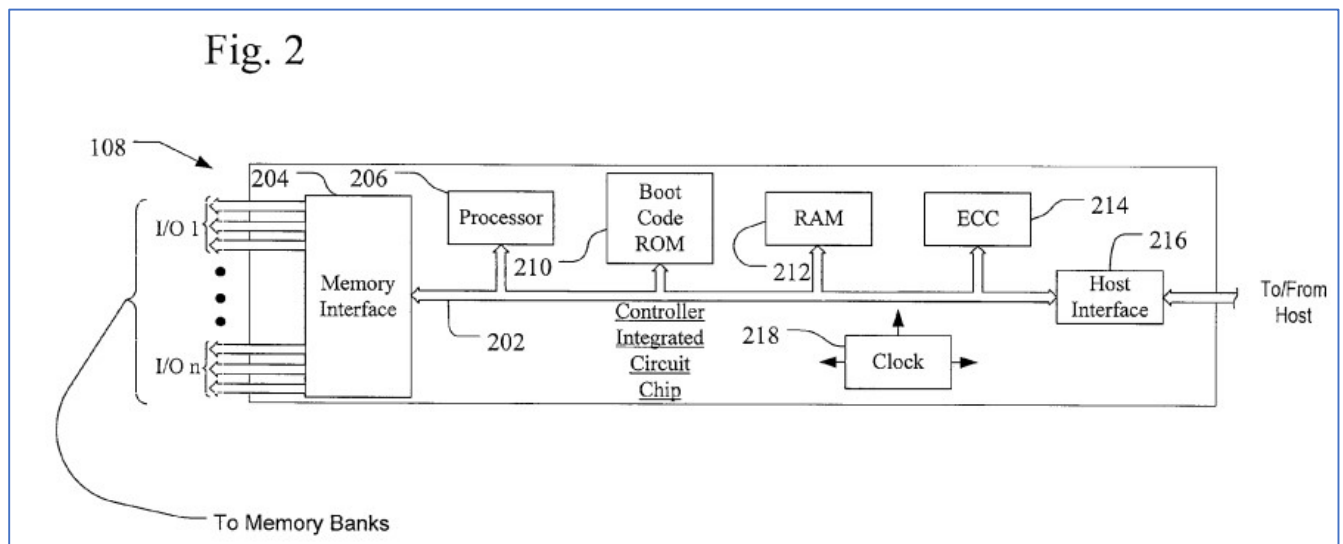
EX1005, Fig. 1, ¶¶10, 89-90.

134. Yamada’s “maintenance determination unit determines that the maintenance is to be performed when determining that the data stored in the nonvolatile memory needs maintenance.” EX1005, ¶18. Once the maintenance condition is detected, the “the maintenance determination unit **stores** at least one address of the nonvolatile memory that needs maintenance” in a maintenance entry list. EX1005, ¶¶23, 89-91 (emphasis added). A POSA would understand these disclosures to teach that in the combined Sinclair-Yamada system, the host issues the flush operation for the pink blocks that need maintenance, and would thus need to access the list of pink blocks that need to be flushed (the stored information) in order for the host to successfully perform this task.

- (n) Claim 1[j]: wherein each said logic comprises at least one of circuitry or instructions stored on physical storage media that, when executed, are to control operation of circuitry.

135. It is my opinion that Sinclair renders this limitation obvious.

136. Sinclair discloses a “memory controller 108” having logic. EX1004, ¶44, Figs. 1-2. Figure 2 illustrates the memory controller 108, which a POSA would understand includes processing circuitry (e.g. processor) and software instructions that are executed by the processor to perform the functionalities described in this Ground:



EX1004, Fig. 2, ¶45.

3. Claim 2

- (a) 2[a]: The storage device of claim 1 wherein: the metadata comprises information representing an age-since-write of data stored in the specific one of the respective subdivisions; and

137. It is my opinion that Sinclair and Yamada render this limitation obvious.

138. Sinclair discloses that the metadata comprises information representing an age-since-write of data stored in the specific one of the respective subdivisions. For instance, Sinclair discloses that the “selection of pink blocks” for the flushing operation “could be based on an assumption that data that has survived the longest in the memory is least likely to be deleted.”

EX1004, ¶69. A POSA would understand this disclosure to teach, or at least suggest, that Sinclair maintains and updates how long the data in a megablock has survived which is metadata comprising information representing an age-since-write of data stored in the specific one of the respective subdivisions.

139. As another example, Sinclair discloses that the controller determines and maintains metadata on whether the data in the megablock is “cold” or “hot” data. EX1004, ¶69. In this manner, “[t]he flush operation relocates relatively ‘cold’ data from a block from which ‘hot’ data has been made obsolete to a relocation block containing similar relatively cold data.” *Id.*, ¶80. A POSA would understand that the term “hot” and “cold” data refers to how recently or frequently the data has been accessed, or information representing an age-since-write, with hot data being more recently written to than cold data which has not been written in a long time.

140. To the extent Sinclair does not teach or suggest storing information representing an age-since-write, the limitation is at least obvious in view of Yamada. Sinclair already discloses storing metadata representing a characteristic of the data such as “hot” and “cold” data contained in megablocks and metablocks, to be used by its maintenance algorithm. EX1004, ¶69. A POSA would understand these disclosures to teach, or at least suggest, that the “hot” and “cold” data information may be used during maintenance to perform the “wear leveling of the nonvolatile memory” disclosed in Yamada, which a POSA understands utilizes the stored age-since-write information. EX1005, ¶30, 33. This is also true in light of the prior art at the time. *See, e.g.*, EX1006 (Son):

Superblock Table ³⁵⁰

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

FIG. 3C

EX1006, Fig. 3C (showing last modification time 335j).

- (b) 2[b] the memory controller also comprises logic to, responsive to the identification of the condition, control copying of data associated with an age-since-write greater than a threshold to a new physical storage location in the NAND flash memory.

141. It is my opinion that Sinclair renders this limitation obvious.

142. As discussed above, Sinclair discloses the identification of a condition where “relatively ‘cold’ data” needs to be flushed. EX1004, ¶80; *see* 1[h]. This limitation further requires that in response to this condition, the controller controls copying the data associated with an age-since-write greater than a threshold, to a new physical storage location in the NAND flash memory.

143. Sinclair discloses that “[t]he flush operation relocates relatively ‘cold’ data from a block from which ‘hot’ data has been made obsolete to a relocation block containing similar relatively cold data.” EX1004, ¶80. Thus, Sinclair relocates data by copying the data from a selected pink block that has been marked to be flushed due to being “cold” to a relocation block. *Id.* A POSA understands that cold data is infrequently accessed data and thus has an age-since-write value greater than some threshold. *See, e.g.*, EX1008 at 5:13-23 (cold data is infrequently accessed data). To perform this flush operation, Sinclair’s re-mapping algorithm relocates data by copying the data from a selected pink block that has been marked to be flushed due to being “cold”

to the relocation block which is a new physical storage location in the NAND flash memory. EX1004, ¶67 (“valid data is copied from the selected pink block in an order of occurrence in the pink block to contiguous locations in the relocation block.”).

4. Claim 3

- (a) 3[a]: The storage device of claim 2 wherein the maintenance request is an erase request;

144. It is my opinion that Sinclair renders this limitation obvious.

145. Sinclair discloses to fulfill this maintenance request, the memory controller relocates (or copies and erases) valid data from a pink block to a relocation white block. EX1004, ¶67 (“In order to flush a pink block, the valid data of a pink block is sequentially relocated in an order of occurrence to a white block that has been designated as a relocation block.”), ¶111 (“in this example, pink blocks selected for flush operations will have 52.4% white capacity”), Fig. 15 (step 1506: “select pink block from pink block list for bank”). Furthermore, Sinclair discloses that to become a white block a pink block needs to be erased. EX1004, ¶58 (“Blocks ... with no valid data, and thus containing only unwritten capacity, are referred to as white blocks 904. The unwritten capacity in a white block 904 may be in the erased state if the memory system 102 employs an ‘erase-after-use’ type of procedure. Alternatively, the unwritten capacity in the white block 904 may consist of obsolete data that will need to be erased upon allocation if the memory system 102 employs an ‘erase-before-use’ type of procedure”). A POSA would therefore understand that Sinclair’s maintenance request that transforms pink into white blocks is an erase request.

- (b) 3[b] the logic is to further cause the memory controller to receive a copy request from the host, the copy request directed to the specific one of the respective subdivisions

146. It is my opinion that Sinclair renders this limitation obvious.

147. Sinclair discloses this limitation. For instance, Sinclair discloses that “[i]n order to flush a pink block, the valid data of a pink block is sequentially relocated in an order of occurrence to a white block that has been designated as a relocation block.” EX1004, ¶67. To perform this flush request, Sinclair’s re-mapping algorithm relocates data by copying the data from a selected pink block that has been marked to be flushed to the relocation white block. EX1004, ¶67 (“valid data is copied from the selected pink block in an order of occurrence in the pink block to contiguous locations in the relocation block.”). As discussed in element 1[f], Sinclair or Sinclair combined with Yamada issues the maintenance flush request that causes data to be copied during a flush.

(c) 3[c] the copy request is to cause the memory controller to control the copying in a manner responsive to the copy request;

148. It is my opinion that Sinclair renders this limitation obvious.

149. Sinclair discloses this limitation. For instance, Sinclair discloses that the flush operation which causes relocation and copying of data is performed by the controller, that is, “the write algorithm [is] managed by the controller 108” (EX1004, ¶71), the “storage address re-mapping... is executed by an application run by the controller 108,” (*id.*, ¶ 73) and the “block flush process... [is] executed by the controller 108.” (*id.*, ¶¶77, 84).

(d) 3[d] the erase request is to cause the memory controller to erase a source location of the data being copied; and

150. It is my opinion that Sinclair renders this limitation obvious.

151. Sinclair discloses this limitation. For instance, Sinclair discloses to fulfill this maintenance request, the memory controller relocates (or copies and erases) valid data from a pink block source location to a relocation white block. EX1004, ¶67 (“In order to flush a pink block, the valid data of a pink block is sequentially relocated in an order of occurrence to a white block that has been designated as a relocation block.”), ¶111 (“in this example, pink blocks selected for flush operations will have 52.4% white capacity”), Fig. 15 (step 1506: “select pink block from

pink block list for bank”). A POSA would therefore understand that Sinclair’s maintenance request is to erase a source location. *See, e.g.*, EX1004, ¶58 (“Blocks ... with no valid data, and thus containing only unwritten capacity, are referred to as white blocks 904. The unwritten capacity in a white block 904 may be in the erased state if the memory system 102 employs an ‘erase-after-use’ type of procedure. Alternatively, the unwritten capacity in the white block 904 may consist of obsolete data that will need to be erased upon allocation if the memory system 102 employs an ‘erase-before-use’ type of procedure”).”

- (e) 3[e] the copy request and the erase request are respective requests which are received by the memory controller at different times.

152. It is my opinion that Sinclair renders this limitation obvious.

153. Sinclair discloses that in some embodiments, its valid data “may consists of obsolete data that will need to be erased upon allocation if the memory system 102 employs an “erase before use” type of procedure.” EX1004, ¶58. That is, in such situations, valid data can be “later made obsolete.” *Id.*, ¶65. Thus, at a different later time the “accumulation of obsolete data... may be independently flushed to unrelated relocation blocks.” *Id.*, ¶72.

5. Claim 4

- (a) 4[a]: The storage device of claim 1 wherein: the memory controller further comprises logic to receive a query from the host;

154. It is my opinion that Sinclair and Yamada render this limitation obvious.

155. Sinclair discloses that the memory controller comprises logic to receive a query from the host. For instance, Sinclair discloses that the host may select pink blocks from the list of pink blocks that meet the threshold requirements. EX1004, ¶68 (“A subset of the pink blocks at or below the threshold amount of valid data may be maintained in a list from which the host or memory system may select pink blocks.”). A POSA would understand that Sinclair’s disclosure that the host may select pink blocks from the list for flushing teaches that the host may query the

memory controller for the list of pink blocks. At minimum, a POSA would understand that this disclosure suggests that the host may query the list of pink blocks that are below the threshold so that the host may then select a pink block to flush.

156. To the extent Sinclair does not teach or suggest the query is received from the host, it would have been obvious in further view of Yamada. Yamada discloses moving maintenance operations from the storage controller to the host. EX1005, ¶10 (desirable to perform “maintenance” on the memory controller so that “device conditions” may be considered). For instance, Yamada discloses that a maintenance determination unit in the access module may include a “command generation unit” that “generated a maintenance command for performing maintenance for the address of the nonvolatile memory device selected by the maintenance determination unit.” *Id.*, ¶22, Fig. 1. In the combined Sinclair-Yamada system, the host would issue the query to the controller for the metadata of the list of pink blocks that meet the threshold requirements.

(b) 4[b] the comparison is between the metadata and a threshold; and

157. It is my opinion that Sinclair renders this limitation obvious.

158. Sinclair discloses that the comparison is between the metadata and a threshold. For instance, Sinclair discloses that the host may select pink blocks from the list of pink blocks that meet the threshold requirements. EX1004, ¶68 (“A subset of the pink blocks at or below the threshold amount of valid data may be maintained in a list from which the host or memory system may select pink blocks.”).

(c) 4[c] the information that indicates that the condition was detected is sent to the host responsive to the query and is to provide to the host with at least one address associated with the specific one of the respective subdivisions.

159. It is my opinion that Sinclair and Yamada render this limitation obvious.

160. A POSA would understand that Sinclair teaches or suggests that in order for the host to select a pink block to flush, the query returns the addresses for the pink blocks in the list. Indeed, Sinclair discloses a “block information table (BIT)” that contains “lists of block addresses for,” among others, “pink” blocks and that the BIT is “maintained by the storage address re-mapping function.” EX1004, ¶82. Sinclair also discloses that in some embodiments the “storage address re-mapping functionality is contained totally within the host.” EX1004, ¶59, Fig. 12. Thus, a POSA would understand that where the storage address re-mapping functionality is contained within the host, the host would query the memory controller for the list of block addresses for the pink blocks.

161. To the extent Sinclair does not teach or suggest the query returns the addresses for the pink blocks in the list, it would have been obvious in further view of Yamada. Yamada discloses moving maintenance operations from the storage controller to the host. EX1005, ¶10 (desirable to perform “maintenance” on the memory controller so that “device conditions” may be considered). For instance, Yamada discloses that a maintenance determination unit in the access module may include a “command generation unit” that “generated a maintenance command for performing maintenance for the address of the nonvolatile memory device selected by the maintenance determination unit.” *Id.*, ¶22. When a maintenance request is received, the “logical address of the maintenance-needed block, or a logical address of the start of the maintenance-needed block and the size of the block, or a logical address of the start of the maintenance-needed block and a logical address of the end of the maintenance-needed block” is included information for “identifying the maintenance-needed block and information about the processing time for the maintenance that will be required by the nonvolatile memory module.” *Id.*, ¶101.

6. Claim 6

- (a) The storage device of claim 4 wherein the at least one address comprises a logical address corresponding to the one of the non-overlapping ranges which has been mapped to the specific one of the subdivisions.

162. It is my opinion that Sinclair renders this limitation obvious.

163. Sinclair teaches that the addresses for the pink blocks in the list is a logical address corresponding to one of its metablocks. Indeed, Sinclair discloses a “block information table (BIT)” that contains “lists of block addresses for,” among others, “pink” blocks and that the BIT is “maintained by the storage address re-mapping function.” EX1004, ¶82. Each pink block address is a logical address mapped to a megablock comprising metablocks. *Id.* ¶¶57-58, Figs. 8-9.

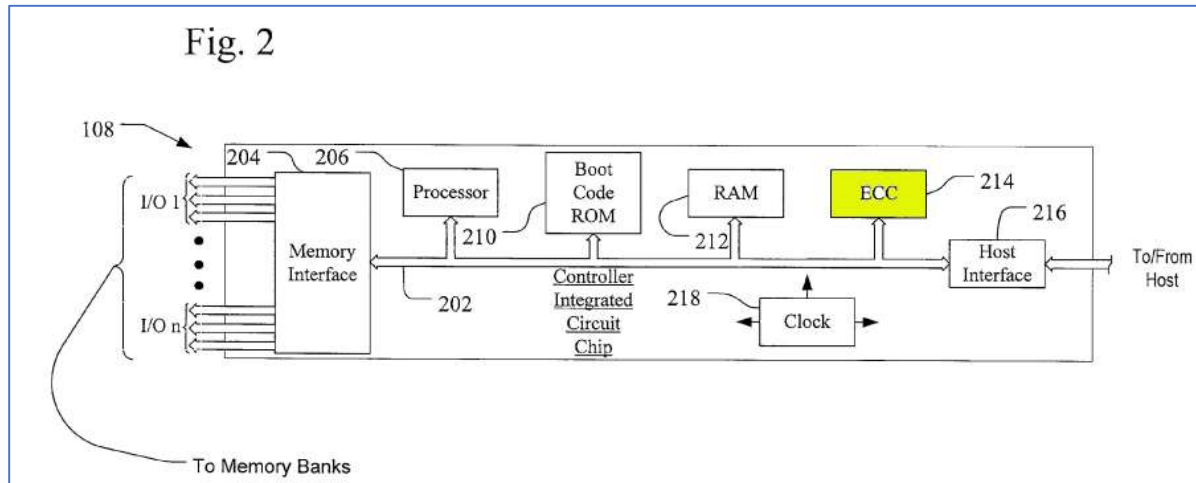
7. Claim 9

- (a) 9[a]: The storage device of claim 1 wherein: the metadata is a function of data error in the data being characterized; and

164. It is my opinion that Sinclair and Yamada render this limitation obvious.

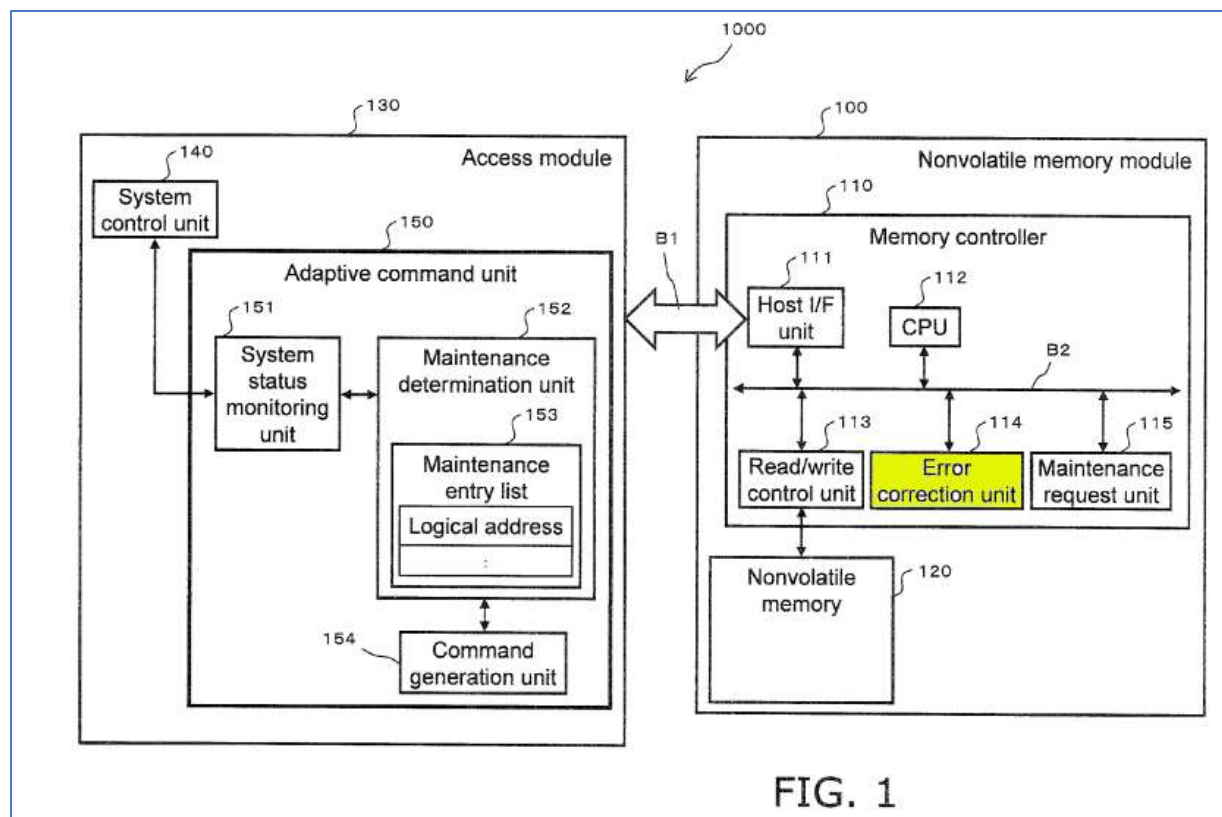
165. A POSA would understand that Sinclair discloses, or at least suggests, that requiring maintenance such that data needs to be flushed (needs maintenance) is a function of a metadata condition. For example, a POSA would understand that data identified as needing to be flushed because it is “obsolete” requiring maintenance *See, e.g.*, EX1004, ¶72.(“...as [blocks of megablocks] each become pink blocks due to the accumulation of obsolete data, [they] may be independently flushed to unrelated relocation blocks.”). Thus, a POSA would be motivated to enable other metadata conditions such as read errors, detected by an ECC unit, to trigger a maintenance operation, because read errors indicate that a block may go bad in the near future, thus indicating the need that the data needs to be moved to a different segment to ensure data consistency. In fact, Sinclair discloses that the memory controller includes an “error correction

unit” (ECC) that is shown below in yellow:



EX1004, Fig. 2, ¶45. But Sinclair is otherwise silent regarding error correction.

166. However, a POSA would have found it obvious for the metadata to be a function of data error in the data being characterized in view of Yamada. Like Sinclair, Yamada discloses that the memory controller includes an “error correction unit” (yellow):

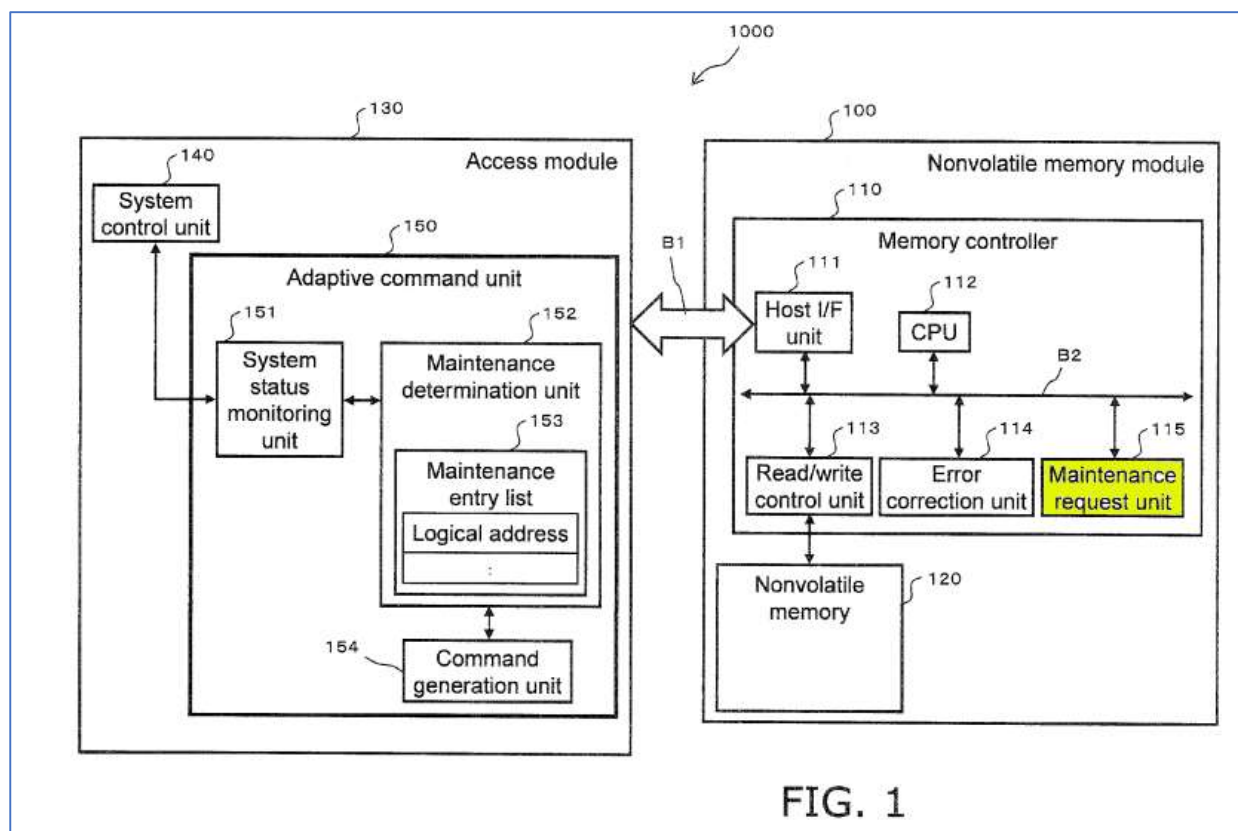


EX1005, Fig. 1, ¶76. Yamada discloses that its “maintenance request unit 1011 determines whether maintenance needs to be performed based on whether the number of errors is greater than or equal to the criterion value....” EX1005, ¶188. Yamada discloses that the “error correction unit determines the number of errors included in data read from the nonvolatile memory.” *Id.*, ¶¶32, 96 (“The error correction unit 114 determines whether the read data has an error, and determines the number of errors in the data.”). A POSA would understand that the number of errors in the data is metadata that is a function of data error in the data being characterized.

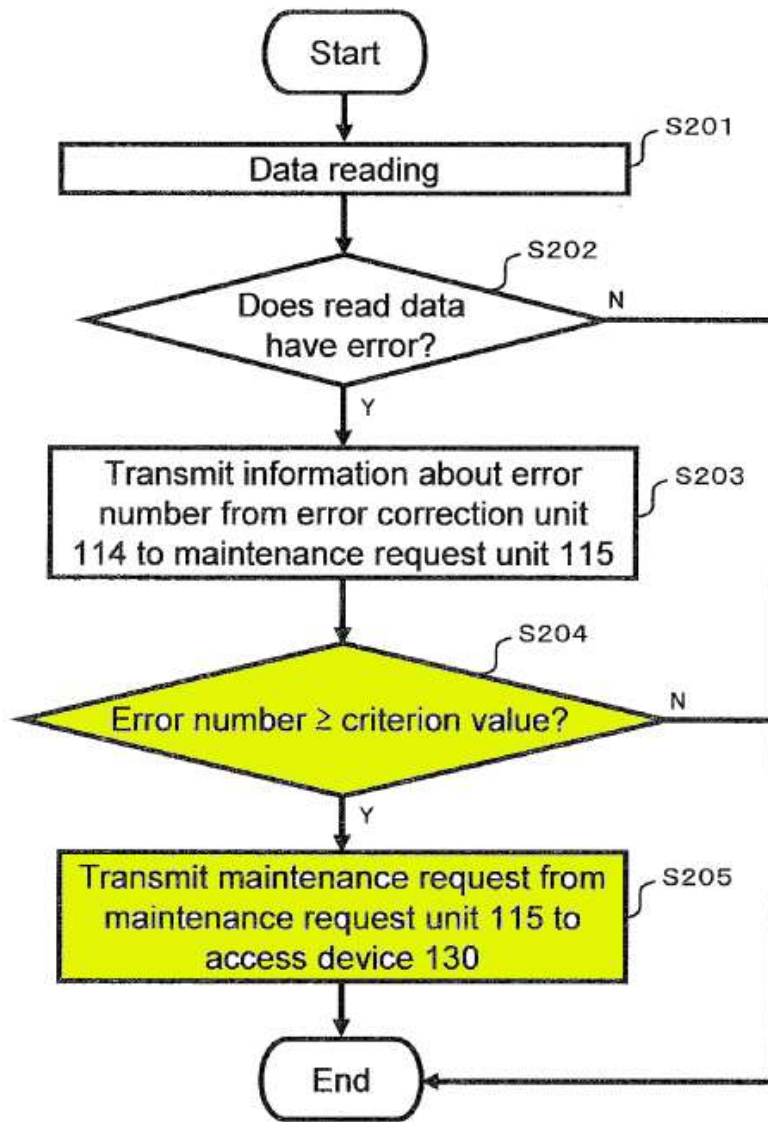
(b) 9[b]: the condition corresponds to a determination by logic of the memory controller that data should be moved to a new physical location in the NAND flash memory.

167. It is my opinion that Sinclair and Yamada render this limitation obvious.

168. Yamada discloses detecting a condition where the number of errors is greater than a threshold value. For instance, the controller includes a “maintenance request unit” that “determines whether the read area needs maintenance based on the received information about the number of errors.”



EX1005, Fig. 1, ¶99. “When the number of errors is greater than or equal to the criterion value, the maintenance request unit 115 transmits a maintenance request to the access module,” as shown in steps s204-s205 of Figure 2:



EX1005, Fig. 2, ¶99.

169. The maintenance request sent by the controller to the access module in response to detecting the condition involves moving data in the block to a new physical location in the NAND flash memory. For instance, Yamada explains that the maintenance request includes an identification of “the maintenance-needed block” and that the access module will then issue a maintenance command for refreshing or wear leveling the specified block to the controller (EX1005, ¶¶30 101-106, Fig. 3) and that the controller then “performs maintenance, such as

refresh of data stored at a designated address (logical address)” (EX1005, ¶¶109-110).

170. A POSA would understand that a refresh or wear leveling command performs data rewriting by moving data to in the block to a new physical location in the NAND flash memory. EX1005, ¶7 (“data refreshing (data rewriting)”). For instance, a POSA would understand that the process of wear leveling typically involves moving cold data from a lower wear count block into a worn block, so that the lower wear count block can be used by hot data. EX1006, Fig. 7 (step 710-730).

8. Claim 10

- (a) 10[a]: The storage device of claim 1 wherein: the metadata represents a data read frequency associated with the data being characterized; and

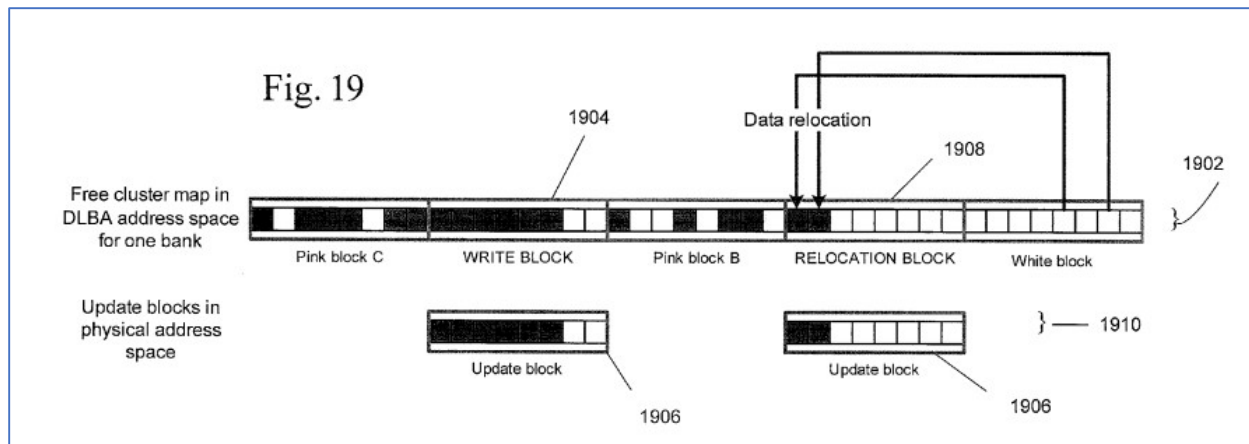
171. It is my opinion that Sinclair renders this limitation obvious.

172. As discussed above, Sinclair discloses that the metadata may be whether the block data is cold or hot. EX1004, ¶69. A POSA would understand that the term “hot” and “cold” data refers to how likely it is that the data will be changed after it is written and that determining whether data is hot or cold typically involves comparing how frequently the data is accessed to some threshold. EX1008 at 5:13-23 (“If the data access frequency of the migration unit is greater than the threshold, it indicates that the migration unit is frequently accessed and has become ‘hot data’; if the data access frequency of the migration unit is less than the threshold, it indicates that the migration unit is infrequently accessed and is ‘cold data’.”). A POSA would thus understand that metadata indicating whether the data is hot or cold represents a data read frequency associated with the data being characterized.

- (b) 10[b]: the condition corresponds to a determination by logic of the memory controller that data should be moved to a new physical location in the NAND flash memory.

173. It is my opinion that Sinclair renders this limitation obvious.

174. Sinclair discloses a controller performing a comparison based on the metadata to detect a condition, namely whether a block should be flushed. The condition corresponds to a determination that data should be flushed from its existing block to a relocation block. For instance, as illustrated in step 1908 of Figure 19, the flush operation performs data relocation by moving data in an existing block to a relocation block:



EX1004, ¶76 (“The storage address re-mapping algorithm manages the creation of white blocks by flushing data from pink blocks to a special write block known as the relocation block 1908”).

9. Claim 11

- (a) 11[a]: The storage device of claim 1 wherein: the metadata represents a data rewrite frequency of the data being characterized; and

175. It is my opinion that Sinclair renders this limitation obvious.

176. Sinclair discloses that the metadata may be whether the block data is cold or hot. EX1004, ¶69. A POSA would understand that the term “hot” and “cold” data refers to how likely it is that the data will be changed after it is written and that determining whether data is hot or cold typically involves comparing how frequently the data is written (or re-written) to some threshold. EX1008 at 5:13-23 (“If the data access frequency of the migration unit is greater than the threshold, it indicates that the migration unit is frequently accessed and has become ‘hot data’; if

the data access frequency of the migration unit is less than the threshold, it indicates that the migration unit is infrequently accessed and is ‘cold data’.”). A POSA would thus understand that metadata indicating whether the data is hot or cold represents a data rewrite frequency associated with the data being characterized.

- (b) 11[b]: the condition corresponds to a determination by the memory controller that data should be moved to a new physical location in the NAND flash memory.

177. It is my opinion that Sinclair renders this limitation obvious.

178. Sinclair discloses performing a comparison based on the metadata to detect a condition, namely whether a block should be flushed. The condition corresponds to a determination that data should be flushed from its existing block to a relocation block.

10. Claim 12

- (a) 12[a]: The storage device of claim 1 wherein: the metadata comprises information representing an age-since-write of data stored in the specific one of the respective subdivisions; and

179. It is my opinion that Sinclair renders this limitation obvious.

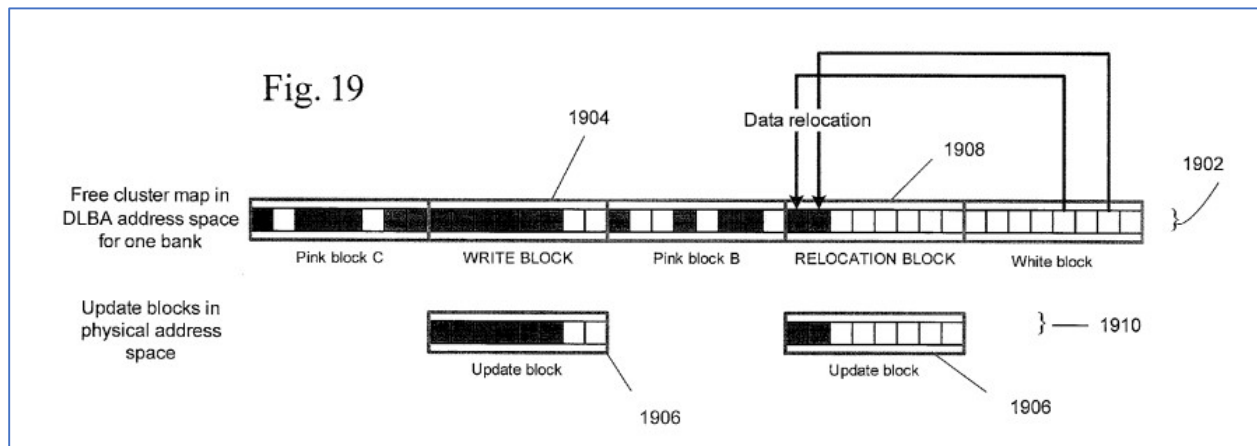
180. Sinclair discloses that the metadata comprises information representing an age-since-write of data stored in the specific one of the respective subdivisions. For instance, Sinclair discloses that the “selection of pink blocks” for the flushing operation “could be based on an assumption that data that has survived the longest in the memory is least likely to be deleted.” EX1004, ¶69. A POSA would understand this disclosure to teach, or at least suggest, that Sinclair maintains and updates how long the data in a megablock has survived, meaning metadata comprising information representing an age-since-write of data stored in the specific one of the respective subdivisions.

- (b) 12[b]: the memory controller also comprises logic to, responsive to the identification of the condition, copy data associated with the

specific one of the respective subdivisions to a new physical storage location in the NAND flash memory.

181. It is my opinion that Sinclair renders this limitation obvious.

182. The condition corresponds to a determination that data should be flushed from its existing block to a relocation block, meaning, that data should be moved to a new physical location in the NAND flash memory. As illustrated in step 1908 of Figure 19, in response to detecting that data should be flushed, the flush operation performs data relocation by copying data in an existing block to a relocation block in the NAND flash memory:



EX1004, ¶76 (“The storage address re-mapping algorithm manages the creation of white blocks by flushing data from pink blocks to a special write block known as the relocation block 1908”).

11. Claim 13

- (a) The storage device of claim 1 wherein the logic comprises at least one processor and instructions stored on at least one physical storage medium, wherein said instructions when executed, are to control operations of the at least one processor.

183. It is my opinion that Sinclair renders this limitation obvious.

184. Sinclair discloses a “memory controller 108” having logic. *See* EX1004, ¶44, Figs. 1-2. Figure 2 illustrates the memory, which a POSA would understand includes processors and software instructions that are executed by the processor to perform the functionalities described in

Fig. 2

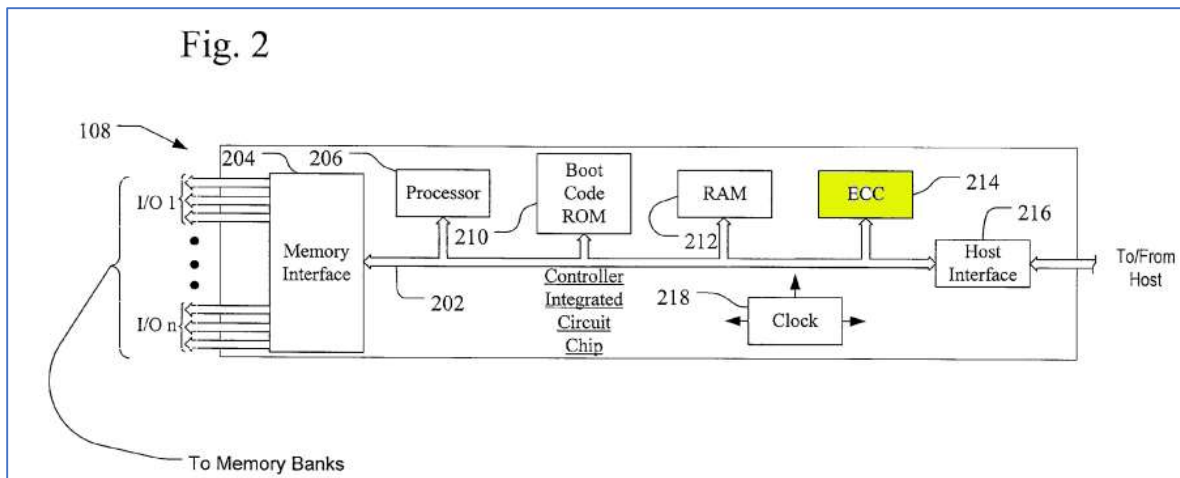
The diagram illustrates a memory controller chip (108) with the following components and connections:

- External Interfaces:**
 - I/O 1 to I/O n:** Multiple input/output ports on the left side of the chip.
 - To Memory Banks:** A connection line extending from the left side of the chip.
 - To/From Host:** A connection line on the right side of the chip.
- Internal Components:**
 - Memory Interface (204):** Receives data from the I/O ports and manages data flow to memory banks.
 - Processor (206):** Manages memory operations and coordinates with the host interface.
 - Boot Code ROM:** Provides initial boot instructions.
 - RAM (212):** Stores data and code within the chip.
 - ECC (214):** Error Correction Code block for data integrity.
 - Host Interface (216):** Manages communication with the host system.
 - Controller Integrated Circuit Chip (218):** The central processing unit of the chip.
 - Clock:** Provides a timing signal to the internal components.
- Internal Connections:**
 - 202:** A bus connecting the Memory Interface, Processor, and Host Interface.
 - 210:** A connection from the Processor to the Controller Integrated Circuit Chip.
 - 212:** A connection from the Controller Integrated Circuit Chip to the RAM.

12. Claim 14

185. It is my opinion that Sinclair renders this limitation obvious.

75



EX1004, Fig. 2, ¶45. As discussed in limitation 4[c] and 15[b], the Sinclair-Yamada system discloses transmitting information to the host when a maintenance operation is needed, e.g., when a failure condition is experienced.

13. Claim 15

- (a) 15[a] The storage device of claim 1 wherein the storage device comprises circuitry to detect a failure condition of one of the erase blocks of the flash memory; and

187. It is my opinion that Sinclair renders this limitation obvious.

188. As discussed, Sinclair discloses circuitry to detect a failure condition of one of the erase blocks in the specific one of the respective subdivisions. *See* Ground 1, Limitation 14[a].

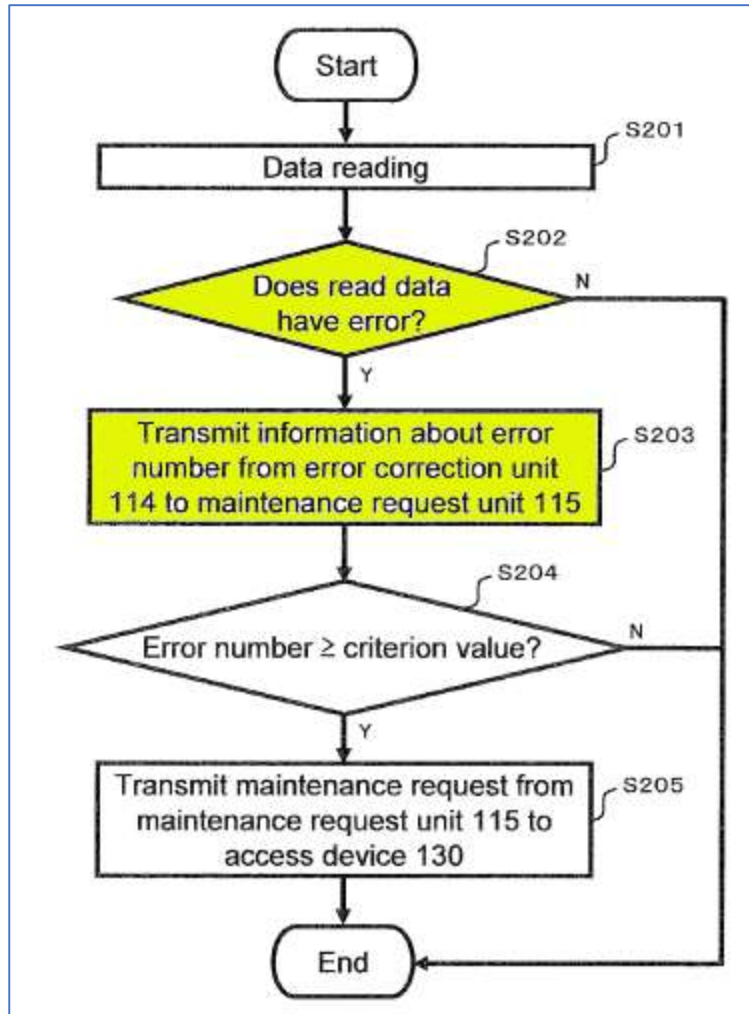
- (b) 15[b] the logic is to cause the memory controller to, for one of the write requests which is addressed to a given one of the respective subdivisions corresponding to the one of the erase blocks for which a failure condition has been detected, transmit to the host a notification of a write failure and logical address information identifying the given one of the respective subdivisions corresponding to the one of the erase blocks for which the failure condition has been detected.

189. It is my opinion that Sinclair and Yamada render this limitation obvious.

190. A POSA would understand that the Sinclair-Yamada system teaches, or at least suggests, that pink blocks that require flushing are in a maintenance condition and cannot be

written to until flushed, and that error blocks that need maintenance are in a failure condition. *See* Ground 1, Limitation 9[a]. The only additional element in this limitation is to transmit to the host a notification of a write failure and logical address information identifying the given one of the respective subdivisions corresponding to the one of the erase blocks for which the failure condition has been detected. But Sinclair is otherwise silent regarding the notification of an error condition, such as a write failure.

191. However, also as discussed above, a POSA would have found it obvious for combine the error correction unit of Yamada to implement the disclosed error correction ECC 214 of Sinclair. *See* Ground 1, Limitation 9[a]. And Yamada discloses transmitting a “notification of the number of data errors (S1103)” to the host to notify it that the data stored in the nonvolatile memory is to be refreshed, as shown in Figure 11:



EX1005 Fig. 11 (annotated).

14. Claim 16

- (a) 16[a] The storage device of claim 1 wherein: for each of the write requests, the logic is to cause the memory controller to generate an address offset that designates a storage location within an addressed one of the respective subdivisions; and

192. It is my opinion that Sinclair renders this limitation obvious.

193. Sinclair discloses that its controller generates and stores an LBA offset in its storage address table (SAT) corresponding to a DLBA run mapped to an LBA run, meaning, an address offset. Sinclair, ¶89 (“A DLBA run contains a backward link to its offset address within the LBA run to which it is mapped The SAT records the LBA offset that corresponds to the beginning

of each DLBA run that is mapped to the LBA run.”). Sinclair also discloses that it maintains a “SAT page field pointer” that defines the offset address from the start of the LBA field as an address offset. Sinclair, ¶98 (“The SAT page field pointer 2310 defines the offset from the start of the LBA field to the start of the DLBA field. It contains the offset value as a number of LBA entries.”). This offset determines the target DLBA address to be written next. *Id.*, ¶108 (“The offset... is used together with one of more DLBA entries... to determine the target DLBA address”). A POSA would understand that this SAT page field pointer offset is generated (or updated) during write requests.

- (b) 16[b]: the logic is to further cause the memory controller to transmit the address offset to the host.

194. It is my opinion that Sinclair and Yamada render this limitation obvious.

195. A POSA would understand that Sinclair teaches or suggests that in order for the host to perform the LBA to DLBA lookup, the host needs to be aware of the updated SAT page field pointer (or address offset). Sinclair also discloses that in some embodiments the “storage address re-mapping functionality is contained totally within the host.” EX1004, ¶59, Fig. 12. Thus, a POSA would understand that where the storage address re-mapping functionality is contained within the host, the host would maintain the SAT page field pointer, or address offset.

196. To the extent Sinclair does not teach or suggest the query returns the SAT page field pointer to the host, it would have been obvious in further view of Yamada. Yamada discloses moving maintenance operations from the storage controller to the maintenance determination unit in a host. EX1005, ¶10, 22. In this Sinclair-Yamada embodiment, the host maintenance determination unit would maintain the SAT page field pointer.

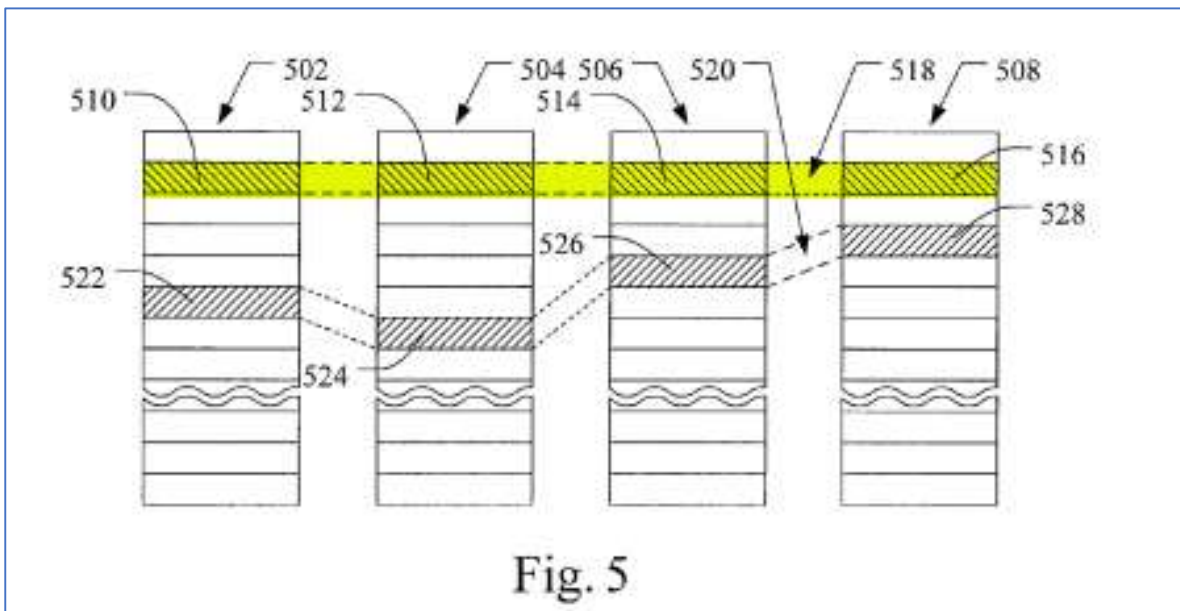
15. Claim 21

- (a) 21[a] The storage device of claim 1 wherein: the NAND flash memory comprises physical planes, wherein the specific one of the

respective subdivisions comprises erase blocks that are in respective ones of the physical planes; and

197. It is my opinion that Sinclair renders this limitation obvious.

198. Sinclair discloses that the NAND flash memory comprises physical planes. EX1004, ¶46 (“Referring to the single bank 7A illustration in FIG. 3, each bank in the multi-bank flash memory 107 may consist of one or more integrated circuit chips, where each chip may contain an array of memory cells organized into multiple sub-arrays or planes. Two such planes 310 and 312 are illustrated for simplicity but more, such as four or eight such planes, may instead be used.”). For instance, Sinclair discloses that a megablock is comprised of “at least one metablock 702 in each bank 107A-107D” and may form a “multiple plane configuration.” EX1004, Fig. 7, ¶56. For instance, Figure 5 illustrates a metablock (yellow) of a megablock comprising four individual blocks, 510-516 (or erase blocks), that are located in respective planes 502-508 [e.g., in respective ones of the physical planes]:



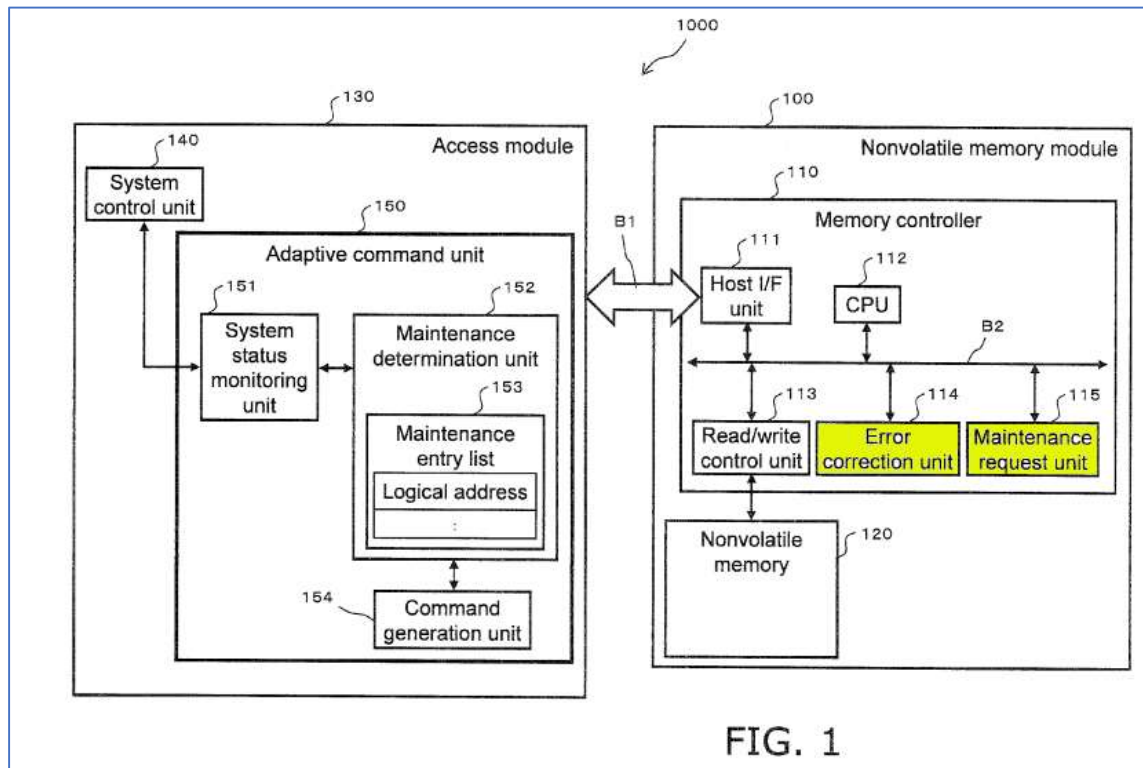
EX1004, Fig. 5, ¶¶53-55.

- (b) 21[b]: the logic is further to cause the memory controller to identify a need for maintenance for the specific one of the

respective subdivisions dependent on the respective metadata, and an associated logical address, and

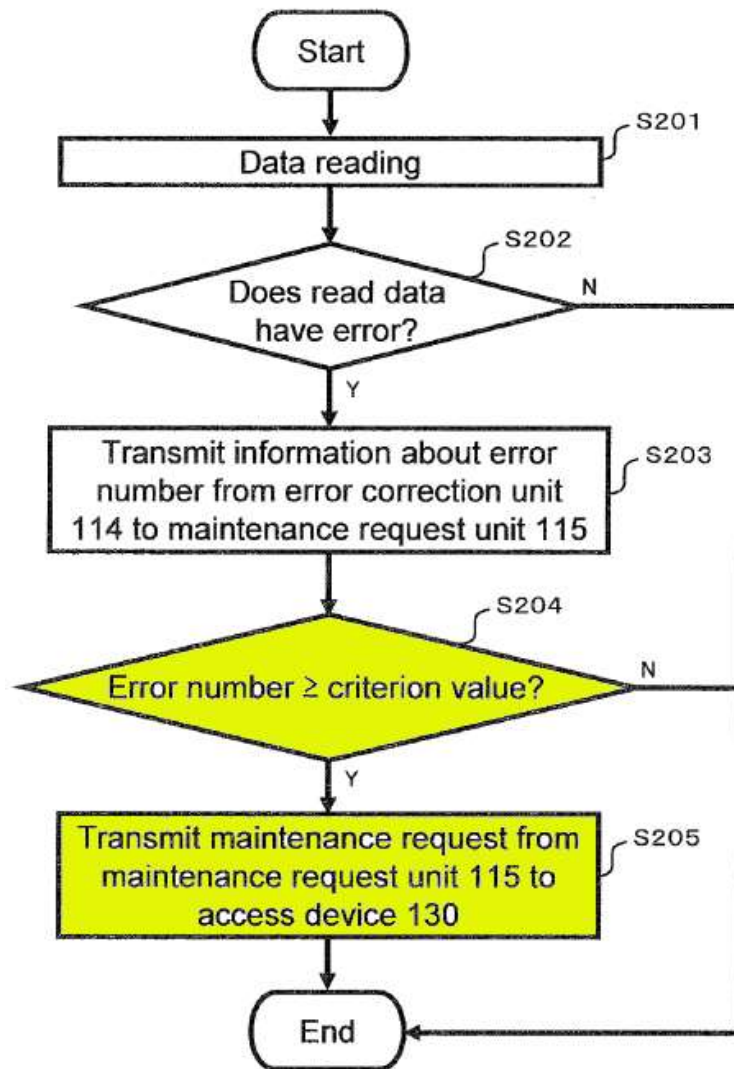
199. It is my opinion that Sinclair and Yamada render this limitation obvious.

200. Yamada discloses that the memory controller includes logic for an “error correction unit” (yellow) and a “maintenance request unit”:



EX1005, Fig. 1, ¶76.

201. These two blocks work together to identify a need for maintenance for the specific one of the respective subdivisions dependent on the respective metadata, and an associated logical address. For instance, the error correction unit “determines the number of errors in the data” and “[w]hen the number of errors is greater than or equal to the criterion value, the maintenance request unit 115 transmits a maintenance request to the access module” as shown in steps s204-s205 of Figure 2:



EX1005, Fig. 2, ¶¶96, 99. “The maintenance request includes a logical address of a block that needs maintenance.” EX1005, ¶¶82, 89.

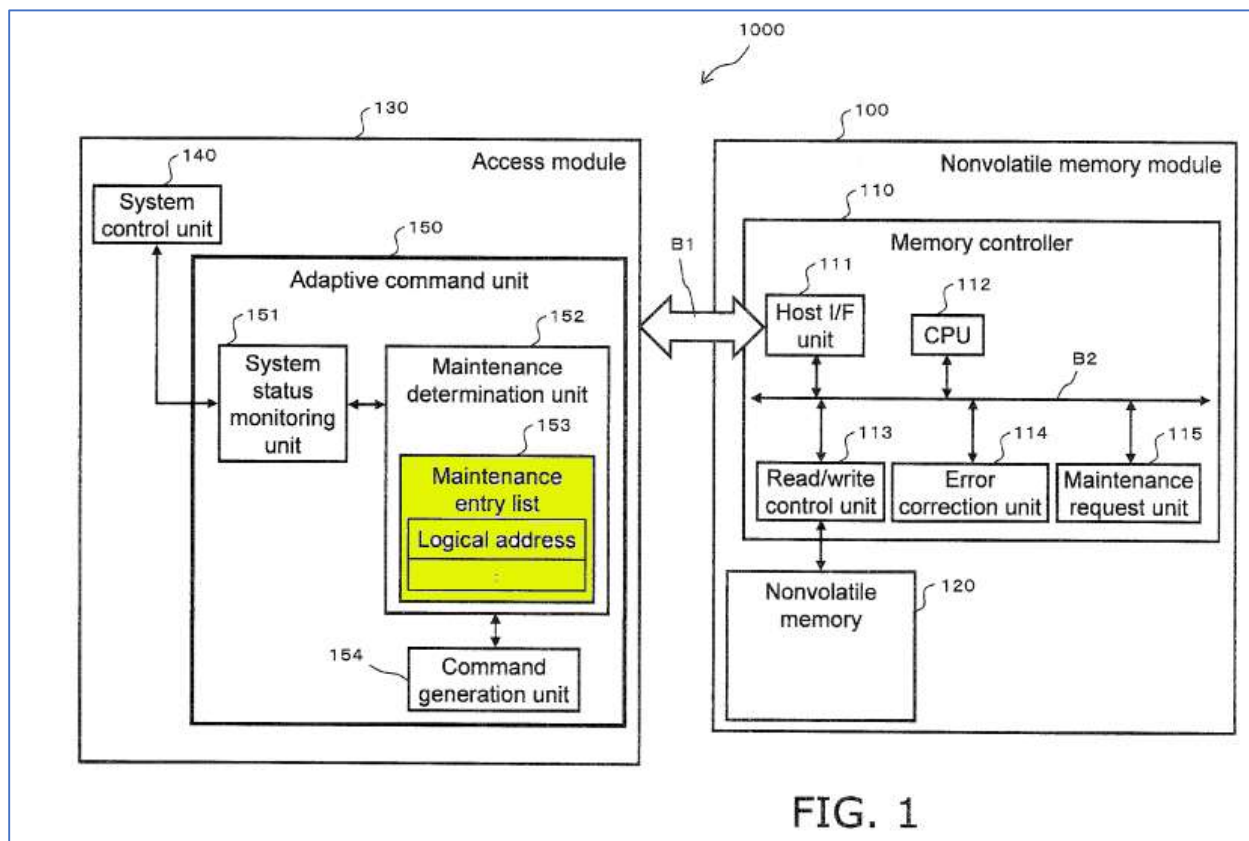
202. Thus, in the combined system the memory controller would identify a need for maintenance of a megablock when, for instance, the error count of a metablock exceeds a threshold that is dependent on the respective metadata, and would also identify the logical block address associated with the metablock to transmit with the maintenance request sent to the host, meaning, an associated logical address.

- (c) 21[c]: transmit the associated logical address to the host, wherein an erase request from the host is dependent on the associated logical address which was transmitted to the host, and

203. It is my opinion that Sinclair and Yamada render this limitation obvious.

204. Yamada discloses that the maintenance request transmitted from the memory controller to the host includes an associated logical address. EX1005, Fig. 2, ¶¶96, 99.

205. Yamada further discloses that the host sends a maintenance command that comprises an erase request and is dependent on the associated logical address. For instance, as discussed above, “[t]he maintenance request [transmitted to the host] includes a logical address of a block that needs maintenance,” which is stored in the maintenance entry list (yellow) of the access module (host):



EX1005, ¶¶82, 89-91.

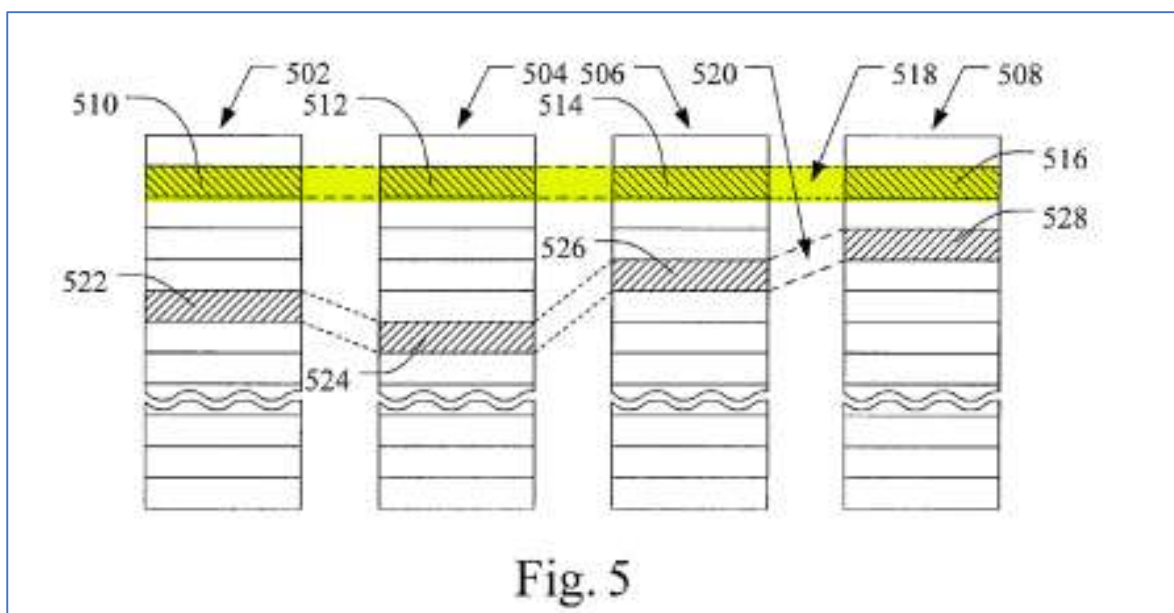
206. The host then issues a “maintenance command... for an addressed stored in the

maintenance entry list,” such as a “refresh and/or wear leveling” command for the block associated with the logical address. EX1005, ¶¶90-91, 30. A POSA would understand that a refresh and/or wear leveling command includes an erase request. For instance, a POSA would understand that wear leveling involves selecting a worn block, copying its contents to a block with a lower wear count, and then erasing the worn superblock. EX1006, ¶82 (wear leveling process involves host instructing the controller to “cop[y] the contents of the worn superblock to the target superblock and erase[] the worn superblock”), Fig. 7. Thus, in the combined Sinclair-Yamada system, the controller would transmit a maintenance request to the host with the associated logical address and the host would then send a refresh and/or wear-leveling command for the address.

- (d) 21[d]: responsive to the erase request, control erasure of two or more of the erase blocks corresponding to the specific one of the respective subdivisions, including erase blocks in each of at least two different ones of the physical planes.

207. It is my opinion that Sinclair and Yamada render this limitation obvious.

208. In the combined Sinclair-Yamada system, the controller transmits a maintenance command (*e.g.*, a refresh and/or wear-leveling command) comprising an erase request for the associated logical address. *See* Ground 1, Limitation 12[c]. The associated logical address may be associated with a metablock having erase blocks that span two or more physical plans, as disclosed in Sinclair. EX1004, Fig. 5, ¶¶53-56. For instance, it may be for the metablock (yellow) that is shown in Figure 5, which has four individual blocks, 510-516 [*i.e.*, **erase blocks**] that are located in respective planes 502-508:



EX1004, Fig. 5, ¶¶53-55. Sinclair discloses that the controller erases the blocks of the metablock, which correspond to a specific megablock together. EX1004, ¶¶50, 54. A POSA would understand this disclosure to be a control erasure of two or more of the erase blocks... including erase blocks in each of at least two different ones of the physical planes.

16. Claim 24

- (a) The storage device of claim 1 wherein the logic is further to cause the memory controller to receive an erase request from the host which is directed to the specific one of the respective subdivisions, responsively control the physical erasure of the one or more of the erase blocks in the specific one of the respective subdivisions and, in association with the physical erasure, modify the information representing the extent to which the one or more of the erase blocks of the specific one of the respective subdivisions are full.

209. It is my opinion that Sinclair and Yamada render this limitation obvious.

210. In the combined Sinclair-Yamada system the memory controller receives an erase request from the host which is directed to the specific one of the respective subdivisions, and responsively controls the physical erasure of the one or more of the erase blocks in the specific one of the respective subdivisions. The combined system would also, in association with the

physical erasure, modify the information representing the extent to which the one or more of the erase blocks of the specific one of the respective subdivisions are full. For instance, Sinclair discloses that its system tracks the number of red blocks, pink blocks, and white blocks. EX1004, Fig. 8, ¶¶58, 67. Sinclair discloses that white blocks are “blocks with no valid data, and thus containing only unwritten capacity,” such that in association with the physical erasure (which will transform a red or pink block into a white block) the system will update the number of a red, pink, and white blocks, which represents the extent to which erase blocks of each megablock are full. *See* EX1004, Fig. 8, ¶¶58, 67 (“This determination may be made based on a total number of white blocks that currently exist...”).

B. Ground II

1. Motivation to combine Terasaki and Yamada

211. A POSA would be motivated to combine the teachings of Yamada with the system disclosed in Terasaki.

212. As with Sinclair, the prior art includes express teachings, suggestions and motivations to combine the teachings of Terasaki and Yamada such that the erase processing request is issued by the host. For instance, Terasaki at least suggests that the host may exchange status information and command information via its host interface. EX1018, ¶¶24-25, Fig. 1. Specifically, data, status information, and command information is “supplied from the host system” and “taken into the interior of controller 3 using host interface block 7 as an inlet.” EX1018, ¶24. Yamada discloses that “maintenance” requests “require[] a certain amount of power consumption” and that “conventional techniques fail to consider the device conditions” when maintenance is performed. EX1005, ¶10. Yamada teaches that to “solve this problem” it is desirable to have the host issue management request, such that the host may consider device conditions when scheduling those commands. EX1005, ¶¶11.

213. Similarly to Sinclair, as discussed in Ground 1, such cooperation of memory maintenance between a controller and a host was known in the art at the time, other references teach that a storage system may be improved by having the host issue management requests, and *Sinclair, Yamada, Jung, and Son* all provide an express teaching, suggestion, and motivation for modifying a device like Terasaki so that a management request, such as erase processing, is issued by the host. EX1004, ¶¶61, 68, Fig. 12; EX1007, ¶¶3-5, 7, 41; EX1006, ¶¶13, 17, 21, 63. For instance, Terasaki describes performance improvements by using zone structures and host-managed information to prevent excessive rewrites and improve the speed of rewrite processing. EX1018, ¶¶12, 44, 82. Yamada similarly shows a need for efficient management of its flash system such that maintenance is controlled in a reliable manner. EX1005, ¶¶15, 20, 36. Thus, a POSA would be motivated to combine Terasaki and Yamada for their teachings of performance improvements by systems that cooperate maintenance between a controller and a host. In addition, both Terasaki and Yamada have ECC-related circuitry to address error correction in cooperation with the host. For instance, Terasaki discloses that to keep track and perform error correction on memory, its error collection code (ECC) block 11 (Fig. 1) generates an error collection code when an error is detected during writing of data to the flash memory in each zone. EX1018, ¶30 (“ECC block 11 is a functional block that generates an error collection code to be appended to the data to be written to flash memory 2, and it also detects and corrects errors contained in the retrieved data based on the error collection code appended to the data that was read out.”). Yamada also discloses that its memory controller includes an “Error correction unit 114” (Fig. 1), and that its “maintenance request unit 1011 determines whether maintenance needs to be performed based on whether the number of errors is greater than or equal to the criterion....” EX1005, ¶188. Thus, a POSA would have also combined Terasaki and Yamada to leverage each reference’s maintenance

cooperation and error correction features in their respective ECC-related circuitry.

214. A POSA would also have had a reasonable expectation of success in combining Terasaki and Yamada because it represents a straightforward implementation of issuing management requests, such as Terasaki's erase processing request, from a host device. Indeed, Terasaki already teaches and suggests that its disclosures encompass performing erase commands on the host or the memory controller and exchanging status information. EX1018, ¶¶24-25.

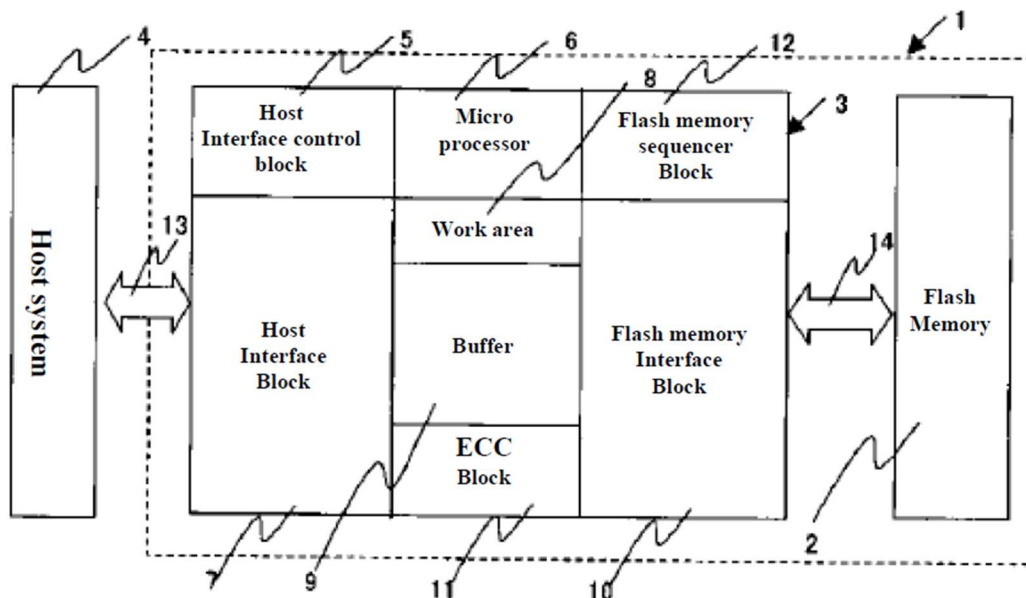
2. Claim 1

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

215. It is my opinion that Terasaki renders this limitation obvious.

216. Terasaki discloses a storage device. For instance, Figure 1 discloses a flash memory system equipped with a flash memory and memory controller interfaced with a host system. EX1018, ¶9, Abstract:

(FIGURE 1)



EX1018, Fig. 1. The flash memory system includes a flash memory 2 and controller 3 that controls the flash memory and is attached to a host system 4, such as a personal computer or digital camera, that it interfaces with. EX1018, ¶¶18-19, 21.

(b) Claim 1[a]: NAND flash memory; and

217. It is my opinion that Terasaki renders this limitation obvious.

218. Terasaki discloses a flash memory system including flash memory that is non-volatile memory. EX1018, ¶¶18-19, 2-3.

(c) Claim 1[b]: a memory controller having logic to cause the memory controller to

219. It is my opinion that Terasaki renders this limitation obvious.

220. Terasaki's flash memory system includes flash memory 2 and a controller 3 with logic to perform the claimed limitations, as discussed below, to control the flash memory and interface with host system 4, such as a personal computer or digital camera. EX1018, ¶¶18-19, 21. For instance, the controller and host exchange data, address information, status information, and command information via host interface block 7 which includes a task file register storing addresses and commands, and an error register that may be set in the event of an error. EX1018, ¶¶24-25.

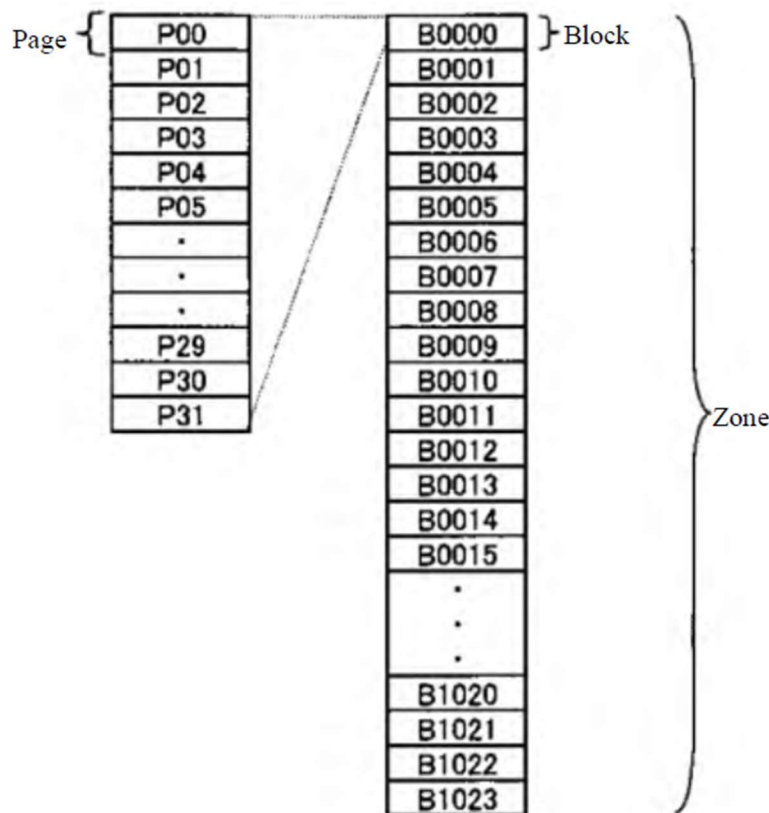
(d) Claim 1[c]: map non-overlapping ranges of logical addresses to respective subdivisions of physical storage in the NAND flash memory, each of the respective subdivisions comprising erase blocks that must each be physically erased as a unit,

221. It is my opinion that Terasaki renders this limitation obvious.

222. Terasaki discloses its flash memory system is organized in zones consisting of a plurality of blocks having consecutive physical block addresses within the memory, meaning, subdivisions of physical storage in the NAND flash memory. EX1018, Abstract, ¶41, Fig. 5. A block is a "unit of erase processing" such that the data on all of the pages in a block must each be

physically erased a unit. EX1018, ¶3. Figure 5 illustrates the relationship between the zones, blocks, and pages of the flash memory whereas the block is the unit of erasing:

(FIGURE 5)



EX1018, Figs. 5-8. For instance, a Zone is illustrated comprising blocks B000 – B1023 with each block comprising multiple pages, such as block B000 comprising pages P00-P31. *Id.*

223. Terasaki discloses its logical block addresses (LBAs) are generally mapped to physical logical block addresses in memory by the use of an address translation table that is updated each time data is written. EX1018, ¶¶5-6. In Terasaki’s system, the LBAs of the data stored within adjacent blocks will be “continuous” and thus non-overlapping “when aligning the blocks constituting said zone in sequence of the physical block address.” EX1018, ¶10. Figures 6-8 illustrate Terasaki’s zone management table at different states:

(FIGURE 6)

Zone	Logical block address
Zone 1	LBA10000
Zone 2	LBA11000
Zone 3	LBA12000
Zone 4	LBA13000
Zone 5	LBA14000
Zone 6	Erased
Zone 7	Erased
Zone 8	Erased
Zone 9	Erased
Zone 10	Erased
Zone 11	Erased
Zone 12	Erased
Zone 13	Erased
Zone 14	Erased
Zone 15	Erased
Zone 16	Erased
Zone 17	Erased
Zone 18	Erased
Zone 19	Erased
Zone 20	Erased

(FIGURE 8)

Zone	Logical block address
Zone 1	LBA10000
Zone 2	Erase request
Zone 3	LBA12000
Zone 4	LBA13000
Zone 5	LBA14000
Zone 6	LBA15000
Zone 7	LBA11000
Zone 8	Erased
Zone 9	Erased
Zone 10	Erased
Zone 11	Erased
Zone 12	Erased
Zone 13	Erased
Zone 14	Erased
Zone 15	Erased
Zone 16	Erased
Zone 17	Erased
Zone 18	Erased
Zone 19	Erased
Zone 20	Erased

EX1018, Figs. 6-8. The zone management table stores the state of each zone (write state, erase state, erase request state, etc.). EX1018, ¶49.

- (e) Claim 1[d]: receive write requests, accompanying addresses and accompanying data from a host and, with respect to each one of the write requests,

224. It is my opinion that Terasaki renders this limitation obvious.

225. Terasaki teaches receiving write requests, accompanying addresses and accompanying data from a host and, with respect to each one of the write requests. For instance, Terasaki discloses receiving write commands from the host system, where each write command contains data and a logical address supplied from the host system. EX1018, ¶7 (“when a write command for data is received from the host system, said address translation table is used as the

basis to determine whether or not there is a block where the same data as that data has been stored in a logical address (the address that may be supplied from the host system).”). Terasaki discloses that in its flash memory system, data and logical block addresses (or a “host address”) may be supplied from host system 4. EX1018, ¶¶38, 43.

- (f) Claim 1[d.i]: identify a specific one of the respective subdivisions dependent on which one of the non-overlapping ranges encompasses the address accompanying the one of the write requests,

226. It is my opinion that Terasaki renders this limitation obvious.

227. Terasaki discloses that a corresponding logical block address (LBA) indicates which LBA the block corresponds to when data is stored in the block. EX1018, ¶40.

228. Specifically, Terasaki discloses a zone management table that manages the storage status of each zone and manages the logical block addresses of the data written onto each zone by zone. EX1018, ¶45. An exemplary initial zone management table is illustrated in Figure 6:

(FIGURE 6)

Zone	Logical block address
Zone 1	LBA10000
Zone 2	LBA11000
Zone 3	LBA12000
Zone 4	LBA13000
Zone 5	LBA14000
Zone 6	Erased
Zone 7	Erased
Zone 8	Erased
Zone 9	Erased
Zone 10	Erased
Zone 11	Erased
Zone 12	Erased
Zone 13	Erased
Zone 14	Erased
Zone 15	Erased
Zone 16	Erased
Zone 17	Erased
Zone 18	Erased
Zone 19	Erased
Zone 20	Erased

EX1018, Fig. 6.

229. This initial stored data is written onto the zone management table above for the following LBAs:

Zone 1: LBA10000 to LBA10999
Zone 2: LBA11000 to LBA11999
Zone 3: LBA12000 to LBA12999
Zone 4: LBA13000 to LBA13999
Zone 5: LBA14000 to LBA14999

EX1018, ¶45. For all remaining zones (Zone 6 – 20), all of the blocks excluding bad blocks will be in an erase state. *Id.*

- (g) Claim 1[d.ii]: program the data accompanying with the one of the write requests into a next available storage location within the specific one of the respective subdivisions, and

230. It is my opinion that Terasaki renders this limitation obvious.

231. Terasaki discloses that in its flash memory system, write processing (or programming of data) in zone units are performed “in order to enable high-speed data processing of data in a single group with consecutive host addresses in continuous write processing” referred to as group data write processing. EX1018, ¶44.

232. During write processing, Terasaki’s zone management table is written with data corresponding to the logical block address into the next available zone. EX1018, ¶45. For instance, if an external host command is received to write data (or program data) corresponding to LBA15000 to LBA15999, an empty erased zone is found within the zone management table, and the data is written onto that erased zone, meaning the data is programmed into a next available storage location within the zone. EX1018, ¶46. This operation is described with reference to Figures 6 and 7, where Zone 6 is the next erased zone available to be written into, and therefore the management table is updated so that Zone 6 “is revised from the erased flag to LBA15000 (the initial logical block address of the stored data).” EX1018, ¶46.

(FIGURE 6)

Zone	Logical block address
Zone 1	LBA10000
Zone 2	LBA11000
Zone 3	LBA12000
Zone 4	LBA13000
Zone 5	LBA14000
Zone 6	Erased
Zone 7	Erased
Zone 8	Erased
Zone 9	Erased
Zone 10	Erased
Zone 11	Erased
Zone 12	Erased
Zone 13	Erased
Zone 14	Erased
Zone 15	Erased
Zone 16	Erased
Zone 17	Erased
Zone 18	Erased
Zone 19	Erased
Zone 20	Erased

(FIGURE 7)

Zone	Logical block address
Zone 1	LBA10000
Zone 2	LBA11000
Zone 3	LBA12000
Zone 4	LBA13000
Zone 5	LBA14000
Zone 6	LBA15000
Zone 7	Erased
Zone 8	Erased
Zone 9	Erased
Zone 10	Erased
Zone 11	Erased
Zone 12	Erased
Zone 13	Erased
Zone 14	Erased
Zone 15	Erased
Zone 16	Erased
Zone 17	Erased
Zone 18	Erased
Zone 19	Erased
Zone 20	Erased

→

EX1018, Figs. 6 (annotated), 7 (annotated).

233. Terasaki further discloses an embodiment that includes a redundant region of the zones as well as the zone management information table, where the manner which data is written to the physical data region is further described with reference to Figures 9-13. EX1018, ¶53. For instance, when an external command is received from the host system to write data corresponding to a logical block address for LBA10000 through LBA10999 in this erased zone, the data will be written in order from B0001 set to the write start pointer. EX1018, ¶53. This is shown by Figure 9:

(FIGURE 9)

		Data region	Redundant region
Erased Start B0001 End B0001 Bad B0004 B0009 B1021		B0000	
		B0001	
		B0002	
		B0003	
		B0004	Bad
		B0005	
		B0006	
		B0007	
		B0008	
		B0009	Bad
		B0010	
		B0011	
		B0012	
		B0013	
		B0014	
		B0015	
		.	.
		.	.
		.	.
		.	.
		.	.
		B1020	
		B1021	Bad
		B1022	
		B1023	

EX1018, Fig. 9.

234. In this write processing, the data corresponding to a logical block address for LBA10000 to LBA10999 will be sequentially written (or programmed) using write registers starting at B0001 of the physical block address. EX1018, ¶53, Fig. 9.

- (h) Claim 1[d.iii]: in association with the programming of the data accompanying the one of the write requests, update information, for the specific one of the respective subdivisions, representing an

extent to which the erase blocks of the specific one of the respective subdivisions are full,

235. It is my opinion that Terasaki renders this limitation obvious.

236. Terasaki discloses that in association with writing or programming of data, its zone management table is updated to change the storage status of each zone after a host write request, managing the logical block addresses of the data written onto each zone by zone. EX1018, ¶¶45-47. As such, the zone management table maintains updated “information indicating the state of each zone (write state, erase state, erase request state, etc.)” representing an extent to which the erase blocks of the specific one of the respective subdivisions are full and “information on the data stored in each zone (information on the physical block address of the block where the data is stored, information on the logical block address of the stored data, etc.).” EX1018, ¶49.

237. A zone in a write state contains the initial logical block address of the stored data. EX1018, ¶¶45-46. A zone for which all blocks are in an erase state are referred to as erased zones having “erased” flags indicating this status. EX1018, ¶45. Finally, a zone with a stored erase request flag indicates that erase processing is necessary to convert the zone to an erased zone. EX1018, ¶47. Thus, a zone having information on data stored indicates a zone that is completely full, whereas a zone flagged as an erased zone indicates that the zone is completely empty. EX1018, ¶¶45-47.

- (i) Claim 1[e]: store a value representing a size of the NAND flash memory that is to be physically erased in fulfilling a maintenance request issued by the host,

238. It is my opinion that Terasaki and Yamada render this limitation obvious.

239. Terasaki discloses zones with erase request flags indicating a need for maintenance. Terasaki discloses that a zone management table that contains the state of each zone (write state, erase state, erase request state, etc.) and information on the data stored in each zone such as its

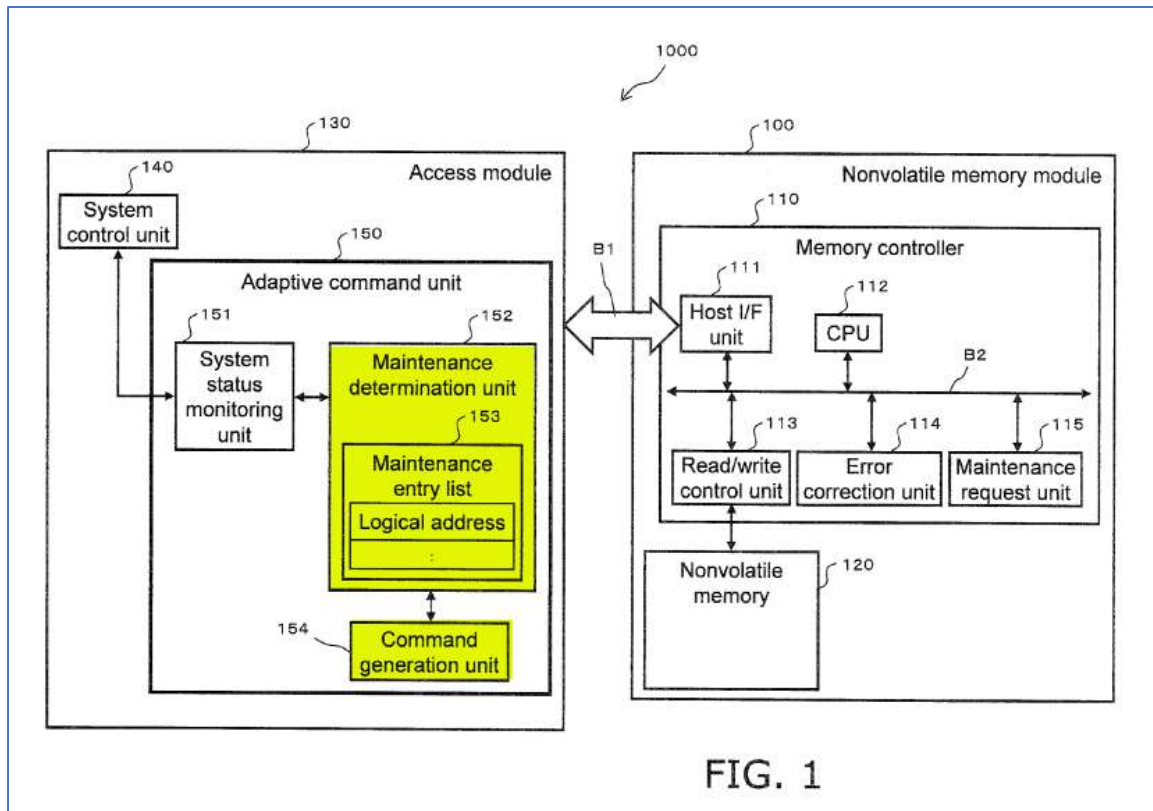
size. EX1018, ¶49 (“Each zone may store up to 1000 blocks of data, and 24 blocks are spare blocks that take into account the occurrence of bad blocks or the like.”). Terasaki further discloses that a zone with a stored “erase request” flag indicates that erase processing is necessary to convert the entirety of that zone to an erased zone. EX1018, ¶47. When data is rewritten, the written flag on the first page of the initial good block in the zone in which the original data was written is changed to an erase request flag for the up to 1000 blocks of that one zone, which represents a size of the NAND flash memory that is to be physically erased. EX1018, ¶68. This erase request flag indicates that erase processing for that zone is necessary. *Id.*

240. Thereafter, during a subsequent write request from the host, an erase process is performed where the block is changed to an erased block. EX1018, ¶¶60, 68, 65 (“the zone management information that has been written on the first page of the initial block... and then the data of the initial good block is erased.... In this change, the erase request flag... is changed to the erased flag”). A POSA would understand this erase process on zones marked with an erase request flag, to be a maintenance request issued by the host.

241. Terasaki also discloses, or at a minimum suggests, that requests that change the erase request flag to the erased flag may be initiated or requested by the host. For instance, Terasaki discloses that its controller and host exchange data, address information, status information, and command information via host interface block 7 which includes a task file register storing addressess and commands. EX1018, ¶¶24-25. A POSA would understand these disclosures to teach that the necessary erase processing that is performed on zones flags as “erase request” is initiated from command information sent by the host to the flash memory controller.

242. To the extent Terasaki does not teach or suggest the erase processing request may be initiated by the host, it would have been obvious in further view of Yamada. Yamada discloses

moving maintenance operations from the storage controller to its “maintenance determination unit” and “command generation unit” at the host for causing maintenance for data retention, such as refresh, to be performed for an address stored in the maintenance entry list. EX1005, ¶10, 22.



Ex 1005, Fig. 1, ¶¶89-90.

243. In the combined Terasaki-Yamada system, the host would issue the erase processing request to change zones flagged as “erase requested” to an “erased status.”

- (j) Claim 1[f]: provide to the host, responsive to a query received from the host, the information, as updated, and the stored value,

244. It is my opinion that Terasaki and Yamada render this limitation obvious.

245. As discussed, Terasaki maintains a zone management table containing updated information indicating the write and erase state of each zone, and stores an erase request flag in its zone management table indicating that erase processing of that zone is necessary. The only

additional requirement of this limitation is that this zone state information and erase request flag value are provided to the host.

246. Terasaki discloses that the controller and host exchange data, address information, status information (the stored value), and command information via host interface block 7 which includes a task file register storing addresses and commands, and an error register that may be set in the event of an error. EX1018, ¶¶24-25. Specifically, data, status information, and command information that is supplied from the flash memory system to the host is “fed into host system 4 using host interface block 7 as an outlet.” EX1018, ¶24. For instance, Terasaki discloses that to pass error information to the host, the host interface block 7 that is used to pass information from the controller to the host may include an error register that may be set in the event of an error. EX1018, ¶25.

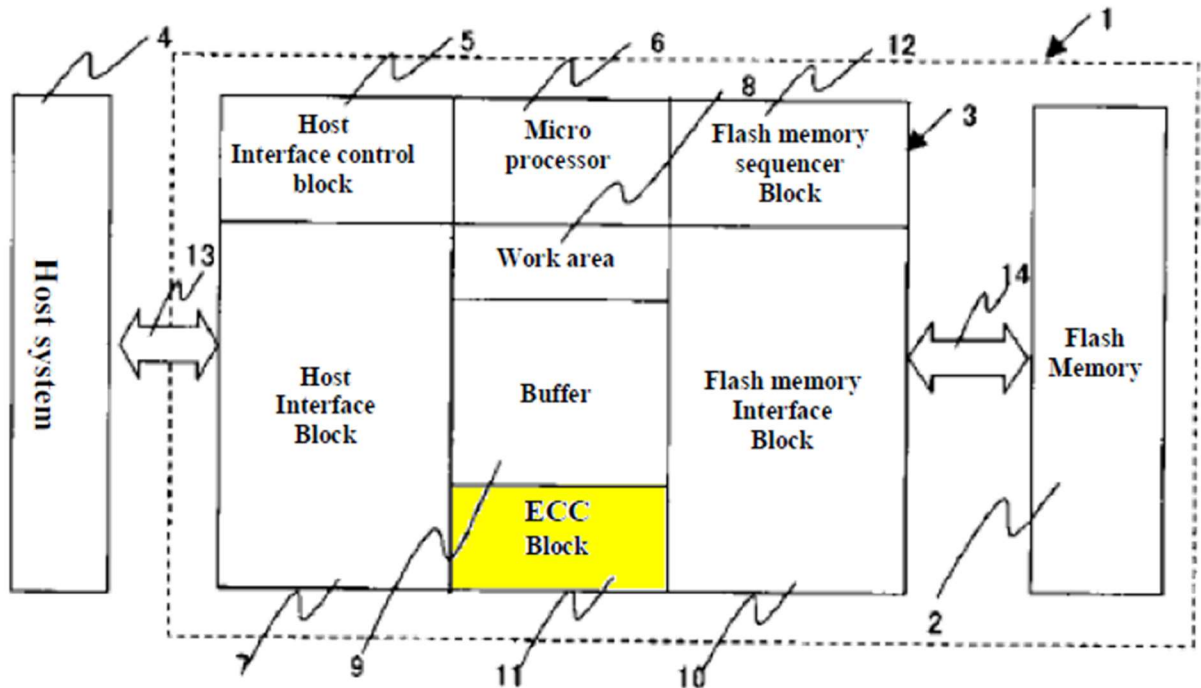
247. To the extent Terasaki does not teach or suggest providing the host the information and the stored value in response to a query, the limitation is at least obvious in view of Yamada and in the combined Terasaki-Yamada system, the host would issue the erase processing request. EX1005, ¶¶14, 109-111. A POSA would understand these disclosures to teach, or at least suggest, that in such an embodiment the host would maintain Terasaki’s zone management table including the updated status of each zone, in order for the host to successfully perform this task.

(k) Claim 1[g]: update metadata for the specific one of the respective subdivisions, wherein the metadata represents a characteristic of the data in the specific one of the respective subdivisions,

248. It is my opinion that Terasaki and Yamada render this limitation obvious.

249. Terasaki further discloses that to keep track and perform error correction on memory, its flash memory system includes an error collection code (ECC) block 11, illustrated in Figure 1:

(FIGURE 1)



EX1018, Fig. 1 (annotated). The ECC block 11 generates an error collection code which is considered metadata for each zone, identifying when an error is detected during writing of data to the flash memory. EX1018, ¶30 (“ECC block 11 is a functional block that generates an error collection code to be appended to the data to be written to flash memory 2, and it also detects and corrects errors contained in the retrieved data based on the error collection code appended to the data that was read out.”). A POSA would understand error to be a characteristic of the data in the zone which includes metadata describing the data, i.e., the data is in a zone with an error collection code.

250. In addition, Terasaki also discloses that its zone management information table stores “information related to the number of erase operations that have been performed in said zone” as an erase count metadata for each zone. EX1018, ¶¶16, 47-49. Terasaki discloses that its zone management table contains the state of each zone (write state, erase state, erase request state, etc.) and information on the data stored in each zone. EX1018, ¶49. This additional erase count

information for each zone indicates the number of erases performed for a zone, and is used to manage the order of priority of the erased zones to be used. EX1018, ¶49 (“when managing the erase count, etc., for each zone to ensure that each zone is used equally, it is preferable to write the erase count, etc., for that zone in the management information storage block.”). A POSA would understand that this order of priority is a characteristic of the data in each zone.

251. Terasaki maintains the zone information metadata of the zone management table up to date so that priority of the erased zones is possible. EX1018, ¶49 (“As long as information showing the state of each zone, information related to the data that has been stored in each zone, or the zone management information such as the erase count performed for a zone, etc., is written in the management information storage block set within each zone in this manner, it will be possible to prepare the zone management table based on that information and to manage the order of priority of the erased zones to be used.”). As a result, this erase count may be used to perform wear-leveling so that writing processing for the data into a zone is balanced. EX1018, ¶49 (“manage the order of priority of the erased zones to be used.”).

- (l) Claim 1[h]: with respect to the metadata, as updated, perform a comparison based on the metadata to detect a condition, and

252. It is my opinion that Terasaki renders this limitation obvious.

253. Terasaki further discloses that an error collection code metadata is used to detect an error condition where the errors will need to be corrected. As discussed, Terasaki discloses an error correction code (ECC) block that generates an error collection code for data written to the flash memory. *See* Ground 2, Limitation 1[g]. In Terasaki’s flash memory system, the “error collection code is additional information for correcting errors contained in the data... and it is generated by ECC block 11.” EX1018, ¶39. Terasaki’s flash memory system stores this error information in pages constructed of byte regions, where “additional information, such as error

collection codes, corresponding logical block addresses, and the block status, may be stored.” EX1018, ¶38, Fig. 4.

254. Terasaki discloses that based on the stored error collection code, if the number of errors contained in the data in a user area is less than or equal to a predetermined number based on a comparison of the error code metadata, then errors will be corrected. EX1018, ¶39. In the event of an error, an error register is set. EX1018, ¶38 (“host interface block 7 includes... an error register (not shown in the figure) that may be set in the event of an error.”).

255. In addition, Terasaki maintains and updates a zone erase count metadata that is used to prioritize zones to be used for writing of host data. EX1018, ¶¶16, 47-49. Terasaki explains that the stored erase count performed for a zone is written in the management information storage block set within each zone so that “it will be possible to prepare the zone management table based on that information and to manage the order of priority of the erased zones to be used.” EX1018, ¶49.

256. A POSA would understand that in order to prioritize erase zones to be used based on their zone erase count value, a comparison of the zone erase counts from one zone to another is necessary in order to prioritize the zone with the least amount of erase counts to use. EX1018, ¶49 (“As long as... zone management information such as the erase count performed for a zone, etc., is written in the management information storage block set within each zone in this manner, it will be possible to prepare the zone management table based on that information and to manage the order of priority of the erased zones to be used.”).

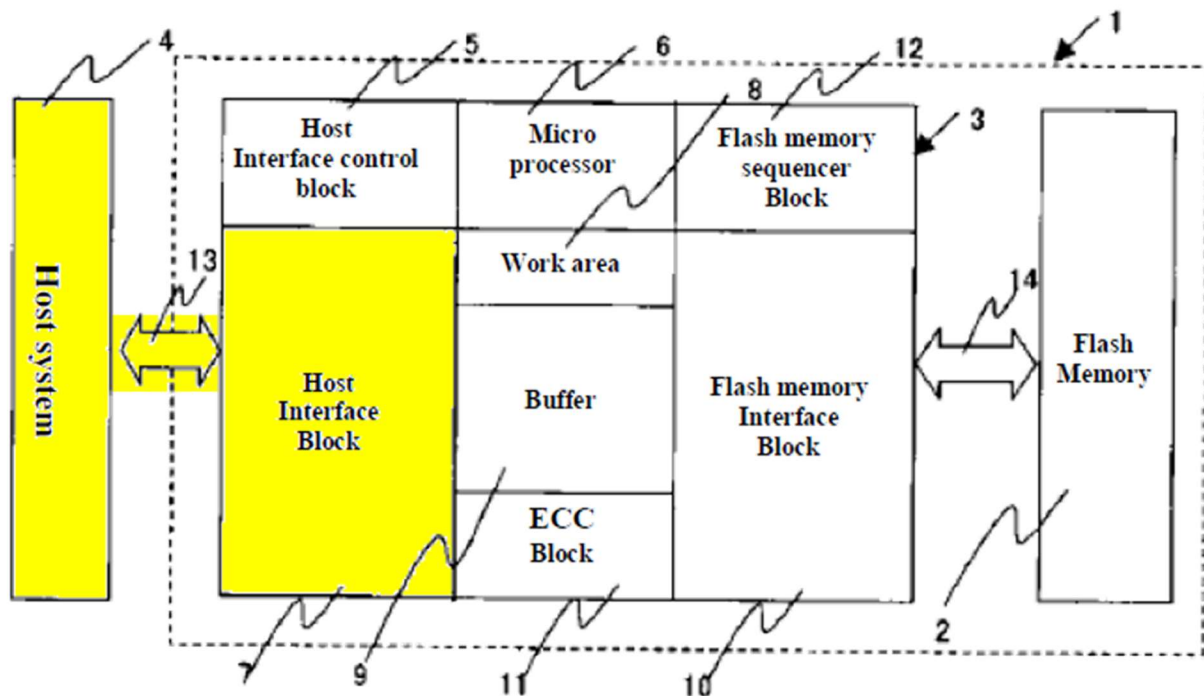
(m) Claim 1[i]: store information, responsive to detection of the condition, wherein the information is accessible to the host and indicates that the condition was detected;

257. It is my opinion that Terasaki and Yamada render this limitation obvious.

258. Terasaki discloses that based on the stored error collection code metadata used to

detect that errors need to be corrected, an “error register” included in host interface block “may be set in the event of an error” and this stored information indicates that the condition was detected. EX1018, ¶25. This error register is included in the “host interface block 7” that is used to “exchange data, address information, status information, and external command information with host system 4” so that it is accessible to the host. EX1018, ¶24 (“Host interface block 7 is a functional block that exchanges data, address information, status information, and external command information with host system 4”), ¶25 (“host interface block 7 includes... an error register (not shown in the figure) that may be set in the event of an error”). Host interface block 7 is illustrated in Figure 1:

(FIGURE 1)



EX1018, Fig. 1 (annotated). As explained, status information including the “error register” containing information set in the event of an error is “fed into host system 4 using host interface block 7 as an outlet.” EX1018, ¶24.

259. In addition, Terasaki discloses storing erase count information responsive to detection of the condition. For instance, Terasaki maintains an erase count used to prioritize the zone with the least amount of erase counts. The erase counts is responsive to the detection of the condition because erase counts are increased when a zone is identified for erasing based on the detection of the aforementioned conditions.

260. Terasaki discloses that the erase count information is accessible to the host and indicates that the condition was detected. For instance, as discussed above, Terasaki teaches that its controller and host exchange status information via host interface 7. EX1018, ¶¶24-25. A POSA would understand that Terasaki's disclosure that the controller exchanges status information with the host teaches that the erase count information is accessible to the host and indicates the condition (*e.g.*, a zone is prioritized for erasing) was detected. EX1018, ¶49. At a minimum, a POSA would understand that this disclosure suggests that the host may access the zone erase count so that it may prioritize a zone with a low erase count to prioritize for erasing. EX1018, ¶49.

261. To the extent Terasaki does not teach or suggest that erase count information is accessible to the host and indicates that the condition was detected, the limitation is at least obvious in view of Yamada. As explained in Ground 2, Limitation 1[e], a POSA would understand Terasaki's disclosures of exchanging status information with the host to teach, or at least suggest, that the zone erase count or zone to be prioritized is provided to the host issuing erase requests in order for the host to successfully perform this task. A POSA understand this is information responsive to the detection of a condition.

(n) Claim 1[j]: wherein each said logic comprises at least one of circuitry or instructions stored on physical storage media that, when executed, are to control operation of circuitry.

262. It is my opinion that Terasaki renders this limitation obvious.

263. Terasaki discloses a memory controller 3 having logic. EX1018, ¶¶18-19, 21.

Terasaki further discloses a microprocessor 6 that controls the overall operations of the controller (EX1018, ¶22) and that its flash memory system may be hosted on a personal computer or digital camera. EX1018, ¶¶18-19.

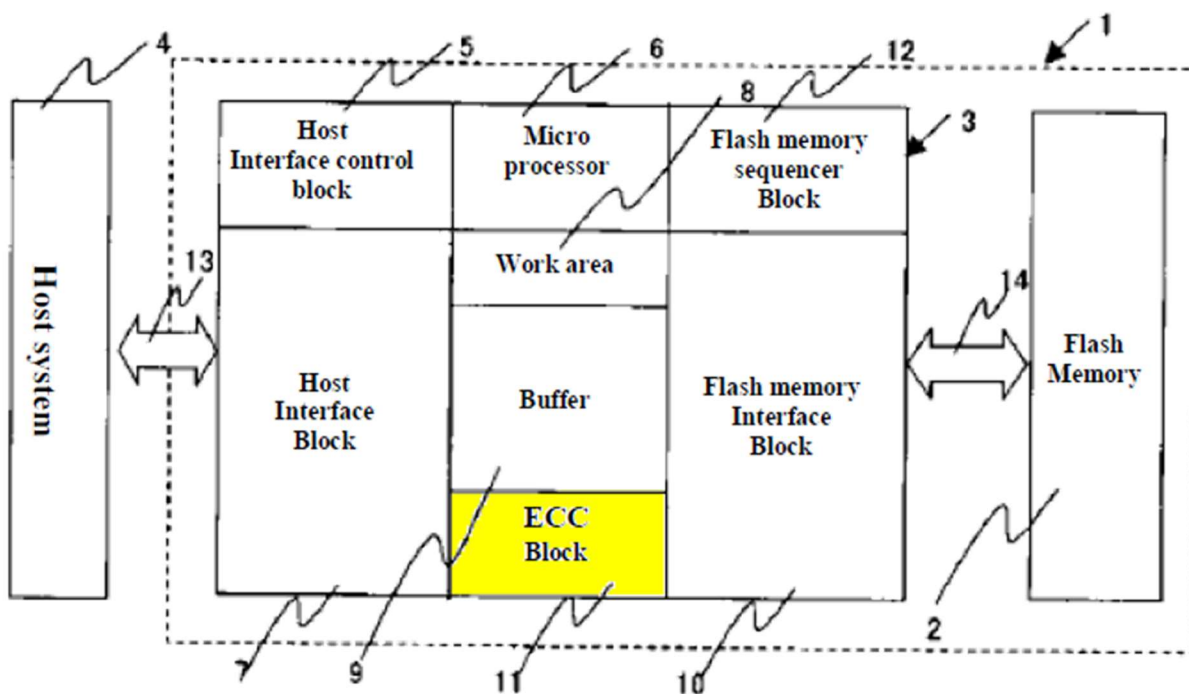
3. Claim 9

- (a) 9[a]: The storage device of claim 1 wherein the metadata is a function of data error in the data being characterized; and

264. It is my opinion that Terasaki renders this limitation obvious.

265. Terasaki discloses that to keep track and perform error correction on memory, its flash memory system includes an error collection code (ECC) block 11 as illustrated in Figure 1:

(FIGURE 1)



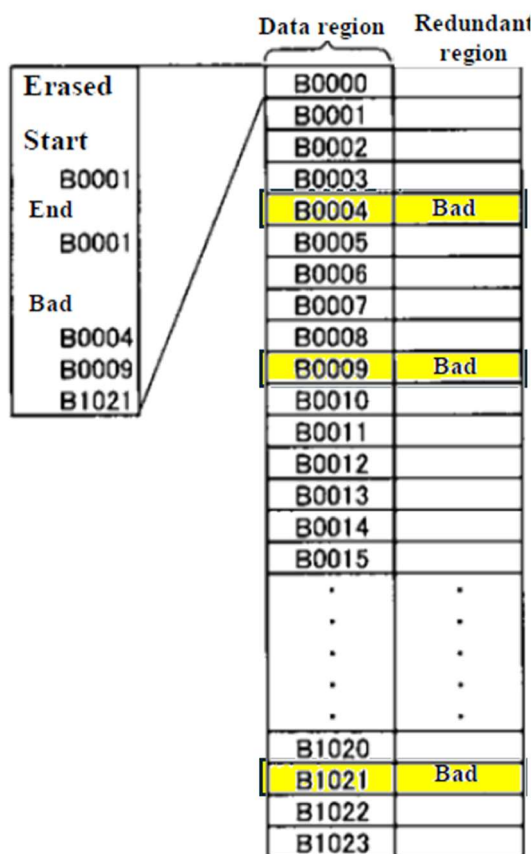
EX1018, Fig. 1 (annotated).

266. The ECC block 11 generates an error collection code for data written to the flash memory and detects and corrects errors contained in the data based on that collection code. EX1018, ¶30. A POSA would understand this error collection code to be metadata that is a

function of data error in the data being characterized. For instance, Terasaki's flash memory is comprised of pages constructed of byte regions, where "additional information, such as error collection codes, corresponding logical block addresses, and the block status, may be stored." EX1018, ¶38, Fig. 4. The "error collection code is additional information for correcting errors contained in the data... and it is generated by ECC block 11." EX1018, ¶39.

267. Terasaki further discloses a "bad block flag" that "indicates whether the block is a bad block." EX1018, ¶41. A POSA would understand this "bad block flag" to be metadata that is a function of data error in the data being characterized. For instance, if it is determined that a memory block is a "bad block" then the system will set the bad block flag. *Id.* Figure 9 illustrates this information on bad blocks present within a zone:

(FIGURE 9)



EX1018, Fig. 9 (annotated), ¶52.

268. Finally, Terasaki discloses an “error register [] that may be set in the event of an error” EX1018, ¶25. A POSA would understand this set error register to be metadata that is a function of data error in the data being characterized. This error register is included in the “host interface block 7” that is used to “exchange data, address information, status information, and external command information with host system 4.” EX1018, ¶¶24, 25.

- (b) 9[b]: the condition corresponds to a determination by logic of the memory controller that data should be moved to a new physical location in the NAND flash memory.

269. It is my opinion that Terasaki renders this limitation obvious.

270. As discussed above, Terasaki discloses that ECC block 11 generates an error collection code for data written to the flash memory, and also marks blocks as bad blocks using a “bad block flag.” EX1018, ¶¶30, 41.

271. Terasaki further discloses that, “[b]ased on this error collection code, if the number of errors contained in the data stored... is less than or equal to a predetermined number” condition, then “the errors will be corrected.” EX1018, ¶39.

272. A POSA would understand that when a block that holds valid data is determined to be bad the data contained in that block needs to be moved to a new physical location, i.e., a good block to prevent data loss. For instance, a POSA would understand that if the number of read errors in one block (correctable errors) surpasses some threshold, that data needs to be moved before the block is no longer readable. EX1006, Fig. 7 (step 710-730); EX1005, ¶7 (“data refreshing (data rewriting)”).

4. Claim 13

- (a) The storage device of claim 1 wherein the logic comprises at least one processor and instructions stored on at least one physical storage medium, wherein said instructions when executed, are to control operations of the at least one processor.

273. It is my opinion that Terasaki renders this limitation obvious.

274. Terasaki discloses a memory controller 3 having logic. EX1018, ¶18-19, 21. Terasaki further discloses a microprocessor 6, which is a processor that controls the overall operations of the controller (EX1018, ¶22) and that its flash memory comprising non-volatile flash memory system may be hosted on a personal computer or digital camera, which a POSA would understand includes processors and software instructions that are executed by the processor to perform the functionalities described in this Ground. EX1018, ¶¶18-19.

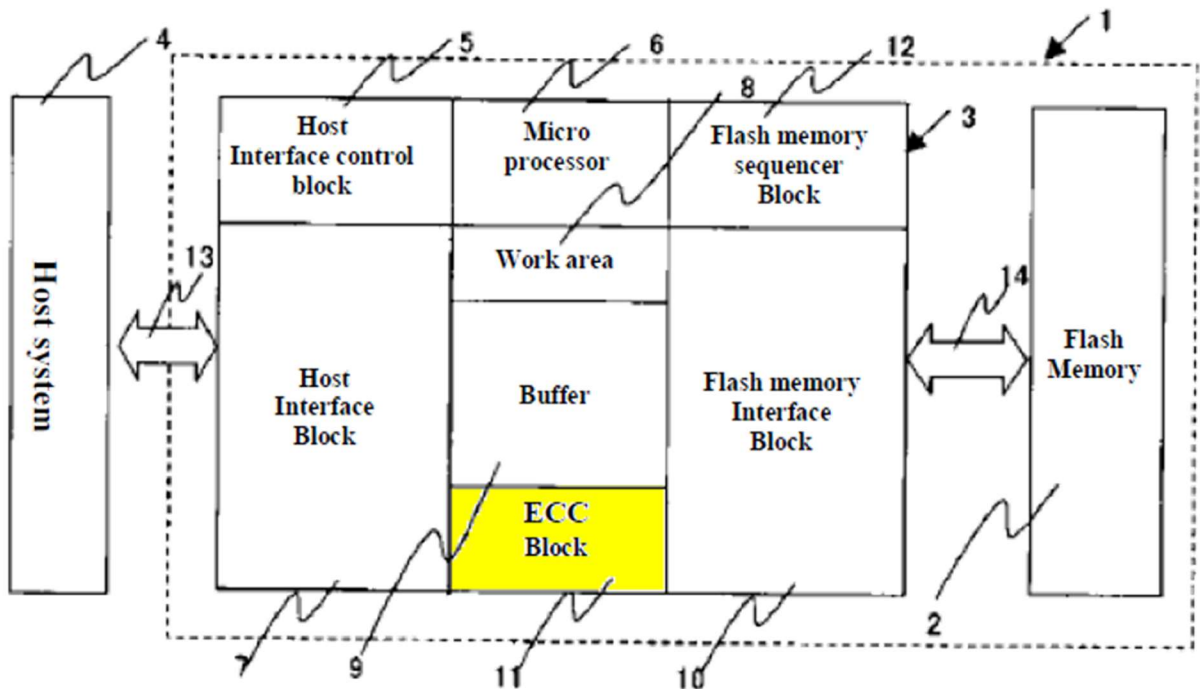
5. Claim 14

- (a) The storage device of claim 1 wherein the storage device further comprises circuitry to detect a failure condition of one of the erase blocks in the specific one of the respective subdivisions and to transmit information to the host that identifies that a storage location in the specific one of the respective subdivisions has experienced the failure condition.

275. It is my opinion that Terasaki renders this limitation obvious.

276. Terasaki discloses that its flash memory system includes an error collection code (ECC) block 11 which is circuitry that detects failure conditions, as illustrated in Figure 1:

(FIGURE 1)



EX1018, Fig. 1 (annotated).

277. The ECC block 11 generates an error collection code for data written to the flash memory and detects and corrects errors contained in the data based on that collection code. EX1018, ¶30. The “error collection code is additional information for correcting errors contained in the data... and it is generated by ECC block 11.” EX1018, ¶39. Terasaki’s flash memory

system stores this information in pages constructed of byte regions, where “additional information, such as error collection codes, corresponding logical block addresses, and the block status, may be stored.” EX1018, ¶38, Fig. 4.

278. Terasaki also discloses an “error register [] that may be set in the event of an error” that was detected [*i.e.*, information to the host that identifies that a storage location in the specific one of the respective subdivisions has experienced the failure condition]. EX1018, ¶25. This error register is included in the “host interface block 7” that is used to “exchange data, address information, status information, and external command information with host system 4” [*i.e.*, transmit information to the host]. EX1018, ¶¶24, 25.

6. Claim 15

- (a) 15[a] The storage device of claim 1 wherein the storage device comprises circuitry to detect a failure condition of one of the erase blocks of the flash memory; and

279. It is my opinion that Terasaki renders this limitation obvious.

280. As discussed above, Terasaki discloses circuitry to detect a failure condition of one of the blocks in the specific one of the respective subdivisions. *See* Ground 2, Limitation 14[a].

- (b) 15[b] the logic is to cause the memory controller to, for one of the write requests which is addressed to a given one of the respective subdivisions corresponding to the one of the erase blocks for which a failure condition has been detected, transmit to the host a notification of a write failure and logical address information identifying the given one of the respective subdivisions corresponding to the one of the erase blocks for which the failure condition has been detected.

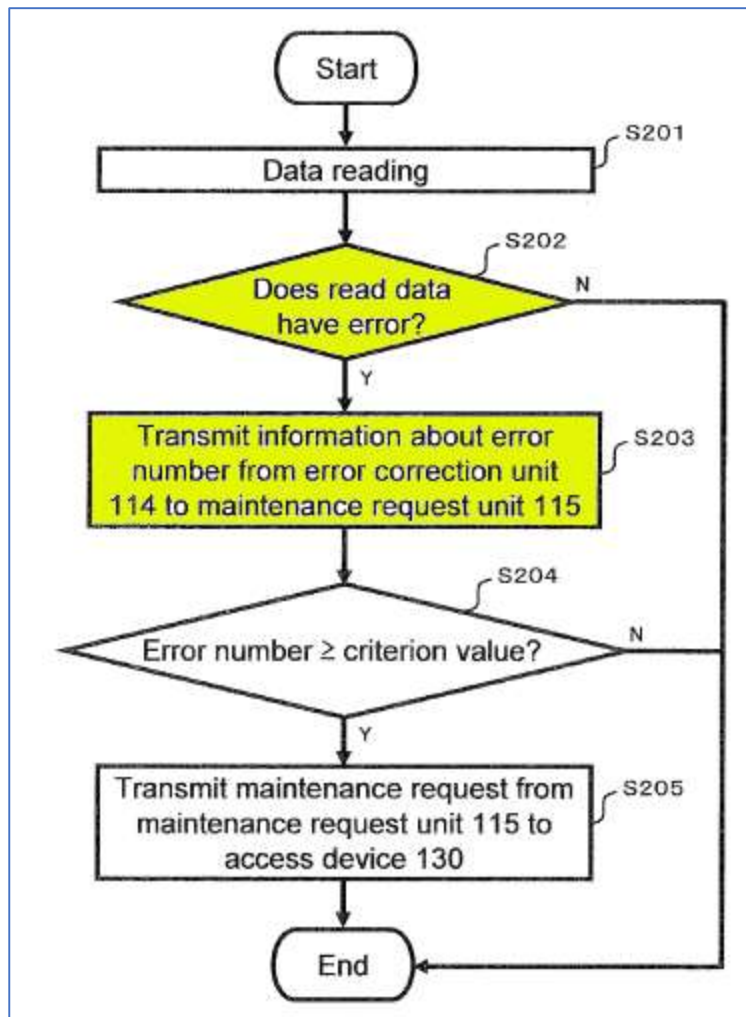
281. It is my opinion that Terasaki and Yamada render this limitation obvious.

282. As discussed above, Terasaki discloses an ECC block that detects errors during a write and generates an error collection code and set an error register in the event of an error. *See* Ground 2, Limitation 9[a]. The only additional element in this limitation is to transmit to the host

a notification of the write failure and logical address information identifying the given one of the respective subdivisions corresponding to the one of the erase blocks for which the failure condition has been detected.

283. Terasaki discloses that the “error register” information with the detected error collection code and “address information” are both “exchanged” with the host via host interface block 7. EX1018, ¶¶24, 25.

284. In addition, a POSA would have found it obvious for combine the additional features of the error correction unit of Yamada to implement the disclosed error correction ECC Block 11 of Terasaki. *See* Ground 2, Limitation 1[e]. Yamada discloses transmitting a “notification of the number of data errors (S1103)” to the host to notify it that the data stored in the nonvolatile memory is to be refreshed, as shown in Figure 11:



EX1005, Fig. 11.

C. Ground III

1. Claim 22

- (a) 22[a]: The storage device of claim 1 wherein: the NAND flash memory comprises physical planes, wherein the specific one of the respective subdivisions comprises erase blocks that are in respective ones of the physical planes; and

285. It is my opinion that Sinclair renders this limitation obvious.

286. Sinclair discloses that the NAND flash memory comprises physical planes, wherein the specific one of the respective subdivisions comprises erase blocks that are in respective ones of the physical planes. *See* Ground 1, Limitation 21[a].

- (b) 22[b]: the logic is to cause the memory controller to, in response to the one of the write requests, control writing of the data into the erase blocks that are in the respective ones of the physical planes using a common address offset for each of the respective ones of the physical planes.

287. It is my opinion that Sinclair and Olbrich render this limitation obvious.

288. Sinclair discloses a multi-plane architecture where the memory controller may write data into erase blocks that span multiple planes. For instance, Sinclair discloses that for “increased parallelism” a set of erase blocks may be grouped together and “operated in larger metablock units” and that the erase blocks may span “all of the planes.” EX1004, ¶¶53-54, Fig. 5.

289. To the extent Sinclair does not disclose using a “common address offset for each of the respective ones of the physical planes,” it would have been obvious in view of Olbrich.

290. Multi-plane command was well-known in the art. EX1011, §1.3 (“SSD manufacturers have provided advanced commands, such as... multi-plane...”), §2.2 (“Multi-plane command activates multiple read, program or erase operations in all planes of the same die”), Fig. 4. A POSA further understood that a multi-plane command typically has the same “page addresses” across planes. EX1011, §2.3 (“The pages executing a multi-plane read/write operation must have the same chip, die, block and page addresses,” but may have different plane addresses), Fig. 3. For instance, Olbrich discloses an example of a multi-plane command for a flash memory that is, like Sinclair, subdivided into groups of blocks that comprised of pages and logically linked together. EX1010, 15:60-67, Fig. 12. Olbrich further teaches that its multi-plane command uses the “same address location” when writing to pages that are “spread across” multiple dies (and planes). EX1010, 15:57-67. For instance, Figure 12 shows “SuperPage 30 is spread across four separate dies” and consists of “four flash Pages” that have the “same address location” across planes. *Id.*, Fig. 12. A POSA would understand that pages are typically specified by an offset and that pages that “spread across four separate dies” are on different planes. Thus, Sinclair’s

disclosure that pages on different planes have the “same address location” discloses a common address offset for each of the respective ones of the physical planes.

D. Ground IV

1. Claim 23

- (a) 23[a]: The storage device of claim 1 wherein the logic is further to cause the memory controller to: in connection with physical erasure of the one or more erase blocks, detect a defect in a given erase block of the one or more erase blocks being erased and, responsive to detecting the defect, to remap each logical address which corresponds to the given erase block to a different erase block in the nonvolatile memory; and

291. It is my opinion that Sinclair-II and Yu render this limitation obvious.

292. This limitation is obvious in view of Sinclair-II. Sinclair-II teaches that if a defective block is detected in a metablock then a different block is responsively substituted [*i.e.*, responsive to detecting the defect, to remap each logical address which corresponds to the given erase block to a different erase block in the nonvolatile memory]:

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

EX1015, ¶77 (emphasis added).

293. A POSA would also understand that detection of a defective block can be performed during read, write, and erase operations “during the lifetime of the memory” and thus discloses, or at least suggests, that the detection may be in connection with physical erasure of the one or more erase blocks.

294. To the extent that Sinclair-II does not teach detecting defective blocks in connection with an erasure, it is obvious in view of Yu’s disclosure of “a process for detection and handling of bad pages and bad blocks” where “[a]fter the block is erased, the status of each page in that

block is checked” to determine if it is bad. EX1016, ¶¶241-243.

295. A POSA would have been motivated to apply Sinclair-II and Yu’s teachings to Sinclair and would have had a reasonable expectation of success. Sinclair and Sinclair-II are similar references from the same inventor and assignee. The two reference describe a flash memory device with a similar architecture and organization. While Sinclair does not discuss its bad block management, a POSTIA would recognize the importance of detecting bad blocks and that blocks may go bad over time. Indeed, detecting bad block merely reflects the application of a known technique to a known device (Sinclair’s flash memory) that is ready for improvement (the ability to detect bad blocks improves Sinclair’s device by replacing blocks that are unusable) and yields predictable results (bad block detection and management). Thus, a POSA would look to references like Sinclair-II and Yu which describe bad block detection operations.

296. A POSA would also have recognized that erasure performance can help identify bad blocks, such that integrating bad block detection with erasure improves efficiency and accuracy. *See, e.g.*, EX1017, 6:21-7:7. For instance, a POSA would understand that where a “block needs more than predetermined number of erase cycles applied to fully erase the block,” it is an indication that the block has failed and should be marked bad. *Id.*

- (b) 23[b]: direct ones of the write requests from the host, for which the accompanying address designates a logical address which has been remapped responsive to detection of the defect, to a storage location associated with the different erase block.

297. It is my opinion that Sinclair-II renders this limitation obvious.

298. Sinclair-II discloses “the substitution of a redundant block for a defective block originally assigned to the metablock” [*i.e.*, a logical address which has been remapped responsive to detection of the defect, to a storage location associated with the different erase block]. EX1015, ¶77. “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” for the substituted block [i.e., direct ones of the write requests from the host, for which the accompanying address designates a logical address which has been remapped]. *Id.* A POSA would have been motivated to combined Sinclair and Sinclair-II for the reason discussed above. *See* Ground III, Limitation 23[a].

Dated: July 28, 2025



Heiner Litz, Ph.D.