

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

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

v.

RADIAN MEMORY SYSTEMS, LLC,
Patent Owner.

Case IPR2025-01378

U.S. Patent No. 11,347,657

**DECLARATION OF DR. HEINER LITZ IN SUPPORT OF PETITION
FOR INTER PARTES REVIEW OF U.S. PATENT NO. 11,347,657**

TABLE OF CONTENTS

	<u>Page</u>
I. SUMMARY OF OPINIONS	1
II. QUALIFICATIONS	1
III. MATERIALS CONSIDERED	3
A. Person Of Ordinary Skill In The Art.....	3
B. Claim Construction	3
C. Obviousness	4
IV. LEVEL OF ORDINARY SKILL IN THE ART	6
V. DETAILED OPINIONS	6
A. Technology Background And State Of The Art	6
B. Summary Of Patents	10
VI. OVERVIEW OF PRIOR ART REFERENCES	21
A. Reiter.....	21
B. Sinclair-233.....	23
C. Sinclair-367.....	24
D. Yamada	26
E. ZBC, SBC-4, and SPC-4.....	27
VII. INVALIDITY OF CLAIMS	29
A. GROUND I: CLAIMS 1-4, 6-9, 13, 15, AND 19-22 ARE RENDERED OBVIOUS BY REITER IN LIGHT OF ZBC, SBC-4, AND SPC-4	29
B. GROUND II: CLAIMS 5, 10-13, AND 18 ARE RENDERED OBVIOUS BY REITER IN LIGHT OF ZBC, SBC-4, SPC-4, AND SINCLAIR-367	65
C. GROUND III: CLAIMS 14 AND 16-17 ARE RENDERED OBVIOUS BY REITER IN LIGHT OF ZBC, SBC-4, SPC-4, SINCLAIR-367, AND YAMADA.....	78
D. GROUND IV: CLAIMS 1-2, 8-9, 18-22 ARE RENDERED OBVIOUS BY SINCLAIR-233.....	84

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. 11,347,657 (the “’657 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 ’657 patent. Specifically, I have been asked to provide my opinion regarding whether claims 1-23 of the ’657 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-V 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 $7\times$ reduction in read access latency through host-orchestrated SSD data placement.

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

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

9. Since 2023, I have served as Director of the Center for Research in Storage Systems (CRSS) at UCSC, a center comprised of 5 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 ("POSITA").

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

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

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

B. Claim Construction

17. I have been informed by counsel that the Patent Trial and Appeal Board ("PTAB") applies the same claim construction standard used in district courts, where the claims are given their

ordinary meaning as understood by one skilled in the art at the time of the invention, informed by the claim language itself, the specification, and the prosecution history. I also understand that “extrinsic evidence”—*i.e.*, evidence other than the patent and prosecution history—can be relevant in determining how a POSITA would understand terms of art used in the claims. I have been informed, however, that extrinsic evidence may not be used to contradict the meaning of the claims as described in the intrinsic evidence—*i.e.*, evidence in the claim language itself, the specification, and the prosecution history.

C. Obviousness

18. 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.

19. 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.

20. 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.

21. 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.

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

23. 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.

24. 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.

25. I understand that whether a prior art reference “teaches away” from the claimed invention is a fact to be considered in determining obviousness. I understand that it is improper to combine references where the references teach away from their combination. However, I understand that the nature of the teaching is highly relevant and must be weighed in substance – a known or obvious composition does not become patentable simply because it has been described as somewhat inferior to some other product for the same use. Furthermore, I understand that the prior art's mere disclosure of more than one alternative does not constitute a teaching away from any of these alternatives because such disclosure does not criticize, discredit, or otherwise discourage the solution claimed.

IV. LEVEL OF ORDINARY SKILL IN THE ART

26. In my opinion, a person of ordinary skill in the art at the time of the '656 Patent (below) 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.

27. I meet these criteria now and met them at the time of the alleged invention.

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

V. DETAILED OPINIONS

A. Technology Background And State Of The Art

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

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

- L2P Management: Mapping LBAs to the most recent PBAs for servicing reads.
- Data Placement: Determining whether hot or cold data should be written to a given 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.

39. 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).

40. 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.

41. 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.” EX1212, Abstract.

42. The operations implemented by the FTL are functionally equivalent regardless of whether they are executed on the SSD or on the host processor. In fact, in most commercial SSDs, the FTL is implemented in software (firmware) and executed on an embedded processor within the SSD. To

enable FTL functionality on the host side, the SSD must expose additional interfaces to the host—such as support for explicit erase operations and fine-grained control over data placement for instance by allowing host software to directly read and write PBAs and access meta data from the SSD controller.

B. Summary Of Patents

1. The '657 Patent

43. The '657 Patent issued May 31, 2022, and is titled “Addressing Techniques For Write And Erase Operations In A Non-volatile Storage Device.”

2. The Challenged Claims

44. Below is a table described the Challenged Claims, including claims 1-23.

Claim 1	
1[pre]	A storage device comprising:
1[a]	flash memory comprising erase units; and
1[b]	circuitry to receive from a host, and control execution of, incoming requests including data write requests and erase requests, wherein each of the incoming requests is accompanied by incoming address information;
1[c]	wherein said circuitry is to, for each of the incoming requests:
1[c][i]	derive, from the accompanying incoming address information, a first address portion and a second address portion
1[c][ii]	identify an addressed block device from the first address portion,
1[c][iii]	use a first division operation on the second address portion, to identify an addressed segment within the addressed block device, wherein the addressed segment corresponds to a subset of the erase units of the flash memory, and

1[c][iv]	for each of the data write requests, identify from the first division operation a third address portion, and use a second division operation on the third address portion to identify an address value to select one of the erase units from the subset which corresponds to the addressed segment and to identify a storage location within the selected one of the erase units;
1[d]	wherein said circuitry is to execute each one of the data write requests by programming accompanying data into the storage location that was identified for that one of the data write requests; and
1[e]	wherein said circuitry is also to execute each one of the incoming erase requests by controlling erasure in the flash memory of one or more of the erase units in the subset corresponding to the addressed segment that was identified from the incoming address information accompanying the one of the incoming erase requests.
Dependent Claims	
2[pre]	The storage device of claim 1 wherein:
2[a]	said circuitry comprises at least one processor; and
2[b]	the storage device further comprises instructions stored on at least one physical storage medium, said instructions when executed to control the at least one processor.
3[pre]	The storage device of claim 1 wherein:
3[a]	said circuitry can receive a request from a host to copy host-selected data from one storage location in flash memory to another storage location in the flash memory, and is to issue commands to the flash memory, in response to the request from the host to copy, to read the host-selected data from a first physical storage location in

	the flash memory and write the host-selected data to a second physical storage location in the flash memory without sending the data being copied to the host;
3[b]	the request from the host to copy is to designate destination address information corresponding to the second physical storage location; and
3[c]	the circuitry is to use the first division operation on the designated destination address information to identify a destination segment within a destination block device.
4[pre]	The storage device of claim 3 wherein
4[a]	the circuitry is to identify from the destination address information a destination logical erase unit and is to select the second physical storage location to be within a physical erase unit that is already mapped to the destination logical erase unit.
5[pre]	The storage device of claim 1 wherein
5[a]	the storage device is to store information identifying a number of structures in the flash memory and is to transmit the information identifying the number of structures to a host, and wherein the storage device is to store the values representing a time since data was programmed on a basis that is respective to each structure of the number of structures, wherein each structure comprises at least one of a page or a unit of memory comprising one or more erase units of the flash memory, and wherein each value represents time since data was programmed into the respective structure.
6[pre]	The storage device of claim 1 wherein
6[a]	the storage device is to store information identifying a number of structures in the flash memory and is to transmit the information identifying the number of structures to a host, and wherein the circuitry is to fulfill the at least one of the incoming

	requests by issuing commands to the flash memory for which a quantum of flash memory selected by the commands which are issued in response to the at least one request is dependent on the information transmitted to the host identifying the number of structures.
7[pre]	The storage device of claim 1 wherein
7[a]	the storage device is to store information identifying a number of structures in the flash memory and is to transmit to a host information that identifies an amount of flash memory that is to be erased as a unit.
8[pre]	The storage device of claim 1 wherein
8[a]	the third address portion specifies a location within a sequential range of locations that correspond to the erase units in the subset.
9[pre]	The storage device of claim 1 wherein
9[a]	the first division operation and the second division operation each subdivide a value obtained from the incoming address information accompanying each of the write requests into respective bitfields.
10[pre]	The storage device of claim 1 wherein:
10[a]	the storage device is to store, for each segment having an erase unit which stores valid data, a value representing a time since data was programmed;
10[b]	the storage device is to compare the values representing time since data was programmed to at least one threshold;
10[c]	upon detection that a specific one of the values has satisfied the one or more thresholds, the circuitry is to copy data stored at one or more physical storage

	locations associated with the specific one of the values from a first physical erase unit to a second physical erase unit; and
10[d]	the storage device is to remap a logical address for the data being copied from the first physical erase unit to the second physical erase unit.
11[pre]	The storage device of claim 10 wherein:
11[a]	the flash memory is to have block devices, one of the block devices being the addressed block device; and
11[b]	the storage device is to identify, for the data being copied, an address of a logical erase unit, the logical erase unit being associated with the addressed block device to the exclusion of each other one of the block devices; and
11[c]	the storage device is to remap a correspondence between the address of the logical erase unit and the first physical erase unit to instead be between the address of the logical erase unit and the second physical erase unit.
12[pre]	The storage device of claim 1 wherein:
12[a]	the flash memory is to have block devices, including the addressed block device; and
12[b]	the storage device is to store metadata on an independent basis for portions of flash memory associated with each one of the block devices, wherein the stored metadata is dependent on an erase count of associated physical memory space;
12[c]	the storage device is to compare the metadata to at least one threshold;
12[d]	upon detection that specific metadata has satisfied the one or more thresholds, the circuitry is to copy data stored at one or more physical storage locations associated with the specific metadata from a first physical erase unit to a second physical erase unit; and

12[e]	the storage device is to remap a logical erase unit value associated with the third address portion from the first physical erase unit to the second physical erase unit.
13[pre]	The storage device of claim 1 wherein:
13[a]	the erase units of the flash memory are physical erase units;
13[b]	the circuitry is to use the second division operation to identify a logical erase unit, wherein each logical erase unit is mapped to one of the physical erase units; and
13[c]	the storage device is to detect a failure condition of the physical erase units and is to remap each logical erase unit which corresponds to one of the physical erase units for which the failure condition is detected to instead correspond to one of the physical erase units for which the failure condition has not been detected and is to utilize the remapped correspondence to service one or more of the incoming data access requests.
14[pre]	The storage device of claim 1 wherein the storage device is to:
14[a]	store, for each segment having an erase unit which stores valid data, a value representing a time since data was programmed;
14[b]	compare the values representing time since data was programmed to at least one threshold; and
14[c]	responsive to detection of a condition that a specific one of the values representing time since data was programmed has satisfied the one or more thresholds, transmit information to a host wherein such transmitted information conveys address information for data stored at one or more physical storage locations associated with the specific one of the values.
15[pre]	The storage device of claim 1 wherein:

15[a]	the flash memory is to have block devices, including the addressed block device; and
15[b]	the storage device is further to:
15[b][i]	store metadata on an independent basis for portions of flash memory associated with each one of the block devices,
15[b][ii]	receive a query, issued by a host, directed to a specific one of the block devices, and
15[b][iii]	responsive to the query, compare the metadata to at least one threshold, to identify specific metadata, and transmit information to the host wherein such transmitted information conveys address information for data stored at one or more physical storage locations associated with the specific metadata.
16[pre]	The storage device of claim 1 wherein:
16[a]	the flash memory is to have block devices, including the addressed block device; and
16[b]	the storage device is further to:
16[c]	store metadata on an independent basis for portions of flash memory associated with each one of the block devices, wherein the stored metadata represents an erase count of associated physical memory space, and
16[d]	receive a query, issued by a host, wherein said query is specific to one of the block devices, and responsive to the query, identify specific metadata, associated with the specified one of the block devices, and transmit information to the host that conveys the specific metadata.
17[pre]	The storage device of claim 1 wherein the storage device is further to:
17[a]	store metadata on an independent basis for portions of flash memory associated with each one of the block devices, wherein the stored metadata represents an erase count of associated physical memory space;

17[b]	compare the metadata to at least one threshold; and
17[c]	responsive to detection that specific metadata has satisfied the one or more thresholds, transmit information to a host wherein such transmitted information conveys address information for data stored at one or more physical storage locations associated with the specific metadata.
18[pre]	The storage device of claim 1 wherein:
18[a]	the flash memory comprises multiple planes; and
18[b]	the circuitry is to, for each one of a set of the data write requests having sequential addressing, identify a commonly addressed page and concurrently write data accompanying the set of write requests into respective ones of the multiple planes in a manner corresponding to the commonly addressed page.
Claim 19	
19[pre]	A storage device comprising:
19[a]	flash memory comprising erase units; and
19[b]	circuitry to receive from a host, and control execution of, incoming requests including data write requests and erase requests, wherein each of the incoming requests is accompanied by incoming address information;
19[c]	wherein said circuitry is to, for each of the incoming requests:
19[c][i]	derive, from the accompanying incoming address information, a first address portion and a second address portion,
19[c][ii]	identify an addressed block device from the first address portion,
19[c][iii]	use a first operation on the second address portion to subdivide the incoming address information into first bit fields, and to identify an addressed segment within the

	addressed block device from a first one of the first bit fields, wherein the addressed segment corresponds to a subset of the erase units of the flash memory, and
19[c][iv]	for each of the data write requests, use a second one of the first bit fields as a third address portion and use a second operation to subdivide the third address portion into second bit fields, to identify an address value to select one of the erase units from a first one of the second bit fields and to identify a storage location within the selected one of the erase units from a second one of the second bit fields;
19[d]	wherein said circuitry is to execute each one of the data write requests by programming accompanying data into the storage location that was identified for that one of the data write requests;
19[e]	wherein said circuitry is also to execute each one of the incoming erase requests by controlling erasure in the flash memory of one or more of the erase units in the subset corresponding to the addressed segment that was identified from the incoming address information accompanying the one of the incoming erase requests.
Claim 20	
20[pre]	A storage device comprising:
20[a]	flash memory, the flash memory to have one or more block devices at a hierarchically-superior level, segments within each of the block devices at a hierarchically-intermediate level, and addressable storage locations within each of the segments at a hierarchically-inferior level, each segment corresponding to a respective, mutually exclusive set of one or more erase units, each segment being associated with only one of the block devices, each erase unit being associated with only one of the segments; and

20[b]	circuitry to receive from a host, and control execution of, incoming requests including data write requests and erase requests, wherein each of the incoming requests is accompanied by incoming address information;
20[c]	wherein said circuitry is to, for each of incoming requests:
20[c][i]	derive a first address portion and a second address portion,
20[c][ii]	identify an addressed one of the block devices from the first address portion,
20[c][iii]	use a first operation to subdivide the second address portion into first bit fields, and to identify an addressed segment within the addressed block device from a first one of the first bit fields, and to identify a third address portion from a second one of the first bit fields, wherein the addressed segment corresponds to a subset of the erase units of the flash memory, and
20[c][iv]	for each of the data write requests, use a second operation to subdivide the third address portion into second bit fields, to identify an address value to select one of the erase units from a first one of the second bit fields, and to identify a storage location within the selected one of the erase units from a second one of the second bit fields;
20[d]	wherein said circuitry is to execute each one of the data write requests by programming accompanying data into the storage location that was identified for that one of the data write requests;
20[e]	wherein said circuitry is also to
20[e][i]	execute each one of the incoming erase requests by controlling erasure in the flash memory of one or more of the erase units in the subset corresponding to the

	addressed segment that was identified from the incoming address information accompanying the one of the incoming erase requests.
Claim 21	
21[pre]	A storage device comprising:
21[a]	flash memory, the flash memory to have one or more block devices at a hierarchically-superior level, segments within each of the block devices at a hierarchically-intermediate level, and addressable storage locations within each of the segments at a hierarchically-inferior level, each segment corresponding to a respective, mutually exclusive set of one or more erase units, each segment being associated with only one of the block devices, each erase unit being associated with only one of the segments; and
21[b]	circuitry to receive from a host, and control execution of, incoming requests including data write requests and erase requests, wherein each of the incoming requests is accompanied by incoming address information;
21[c]	wherein said circuitry is to, for each of the incoming requests:
21[c][i]	derive, from the accompanying incoming address information, a first address portion and a second address portion,
21[c][ii]	identify an addressed one of the block devices from the first address portion,
21[c][iii]	use a first operation to subdivide the second address portion into first bit fields, and to identify an addressed segment within the addressed block device from a first one of the first bit fields and a third address portion from a second one of the first bit fields, wherein the addressed segment corresponds to a subset of the erase units of the flash memory, and

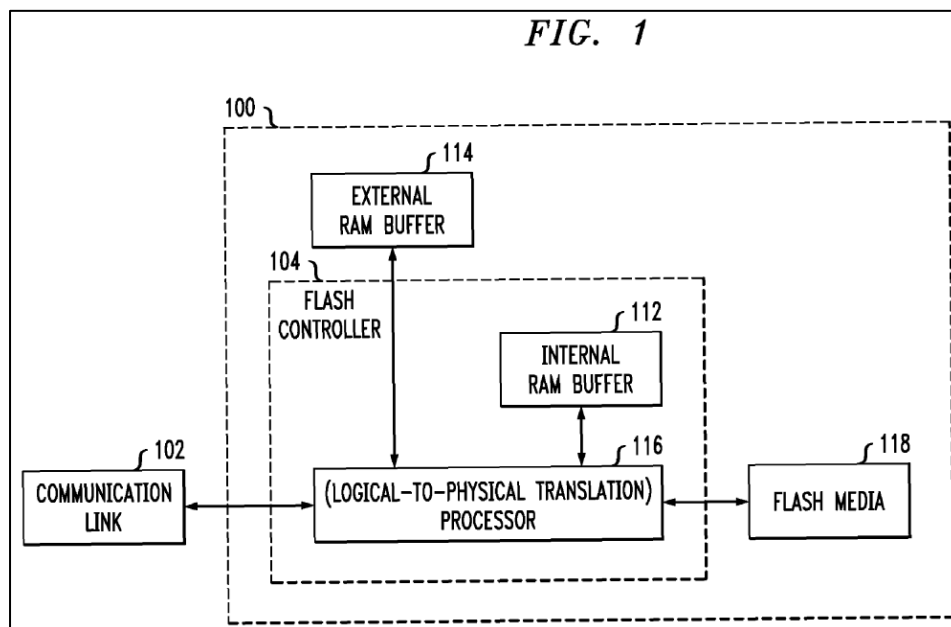
21[c][iv]	for each of the data write requests, use a second operation to subdivide the third address portion into second bit fields, to identify, from a first one of the second bit fields, an address value to select one of the erase units and, from a second one of the second bit fields a storage location within the selected one of the erase units;
21[d]	wherein said circuitry is to execute each one of the data write requests by programming accompanying data into the storage location that was identified for that one of the incoming data write requests; and
21[e]	wherein said circuitry is also to execute each one of incoming erase requests by controlling erasure in the flash memory of the one or more erase units in the subset corresponding to the addressed segment that was identified from the incoming address information accompanying the one of the incoming erase requests.
Dependent Claims	
22[pre]	The storage device of claim 1 wherein
22[a]	the second address portion comprises a first set of bits and a second set of bits, the bits of the first set of bits being mutually-exclusive to the bits of the second set of bits, wherein the first division operation is to separate the first set of bits from the second set of bits, and the circuitry is to identify the addressed segment from the first set of bits.

VI. OVERVIEW OF PRIOR ART REFERENCES

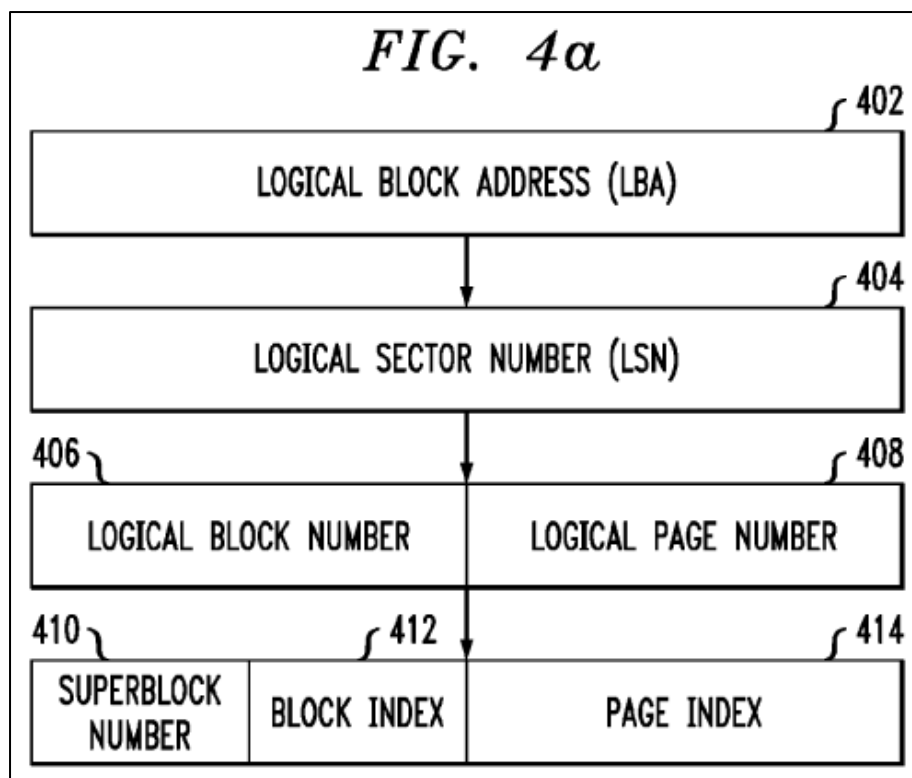
A. Reiter

45. U.S. Patent 8,301,861 (“Reiter”) was filed on April 29, 2010, was published on March 24, 2011, and issued on October 30, 2012. I understand that it is prior art under AIA 35 U.S.C. § 102(a)(1)-(2).

46. Reiter discloses a flash system 100, comprising flash controller 104 for receiving and executing read and write commands from communication link 102 sent by an external computer system (*i.e.*, host). EX1004, 6:63-66, 8:1-6, FIG. 1:



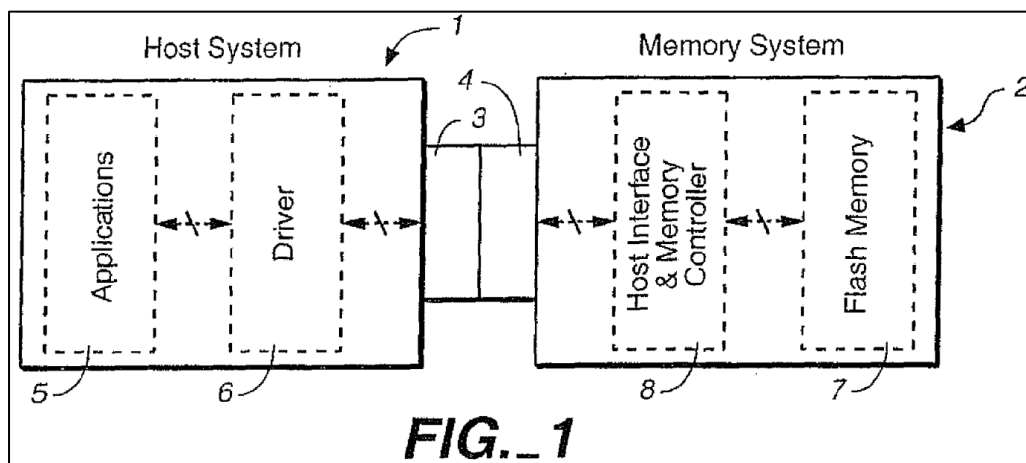
47. The read and write commands are accompanied by a logical address, which corresponds to a physical address that details the location in flash media 118. And to execute the command at that location, flash controller 104 performs a logical-to-physical translation method to translate the logical address to a physical address. EX1004, 7:31-35, 8:1-10. Reiter discloses a four-level logical-to-physical translation method including (1) the incoming logical block address 402, (2) logical sector number 404, (3) a bitstring comprising logical block number 406 and logical page number 408, and (4) a bitstring comprising superblock number 410, block index 412, and page index 414. EX1004, 8:54-56, 8:66-9:2, 9:5-20, FIG. 4a:



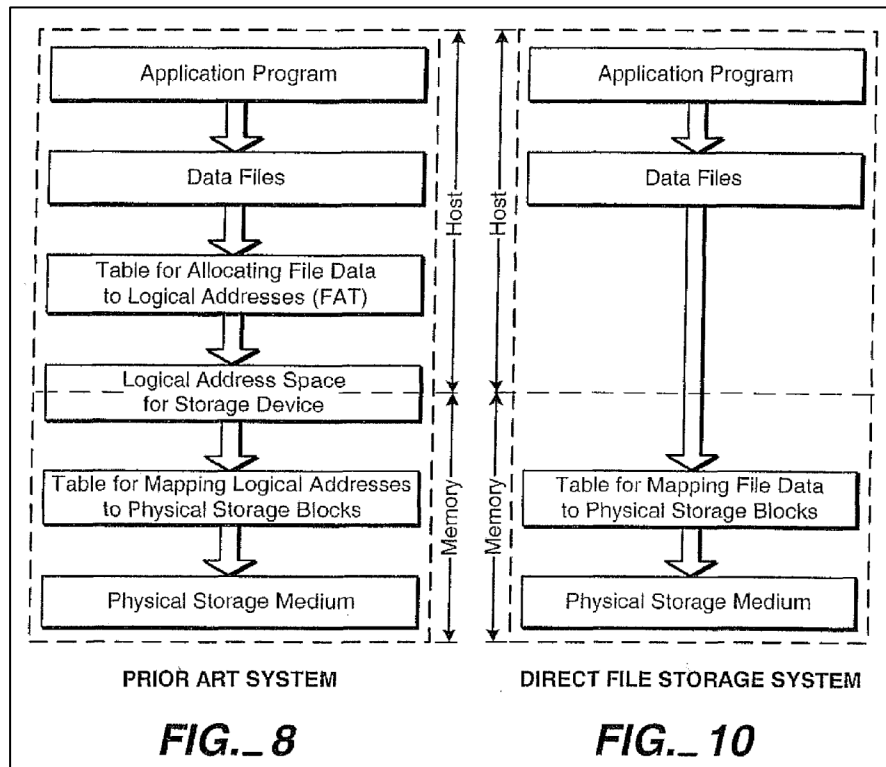
B. Sinclair-233

48. U.S. Patent 7,984,233 (“Sinclair-233”) was filed on May 5, 2010, was published on August 26, 2010, and issued on July 19, 2011. It is prior art under AIA 35 U.S.C. § 102.

49. Sinclair-233 discloses flash memory system 2, where flash memory system 2 has a direct file interface with the host. EX1005, Abstract, 13:24-34, FIG. 1:



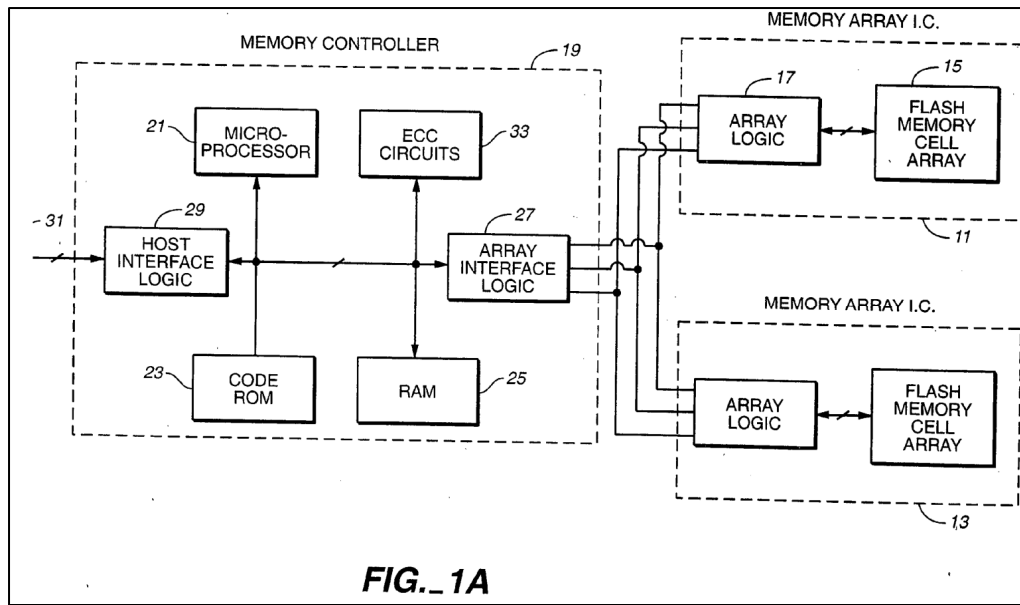
50. A direct file interface means that commands operate on a filename or an offset address within a linear address space unique to a file. EX1005, 13:24-34. This is as opposed to conventional logical-to-physical address system, where the host commands target a logical block address translated to a physical block address by the storage device. See EX1005, FIGS. 8, 10:



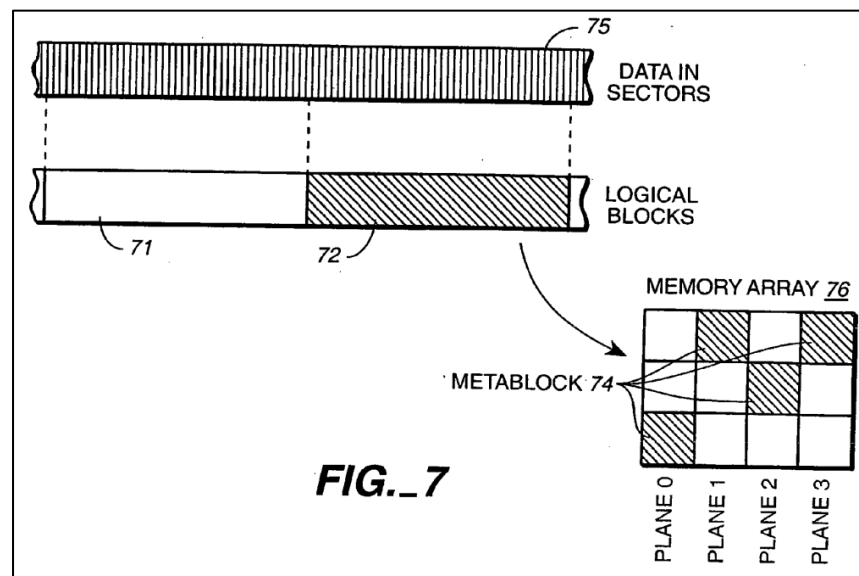
C. Sinclair-367

51. U.S. Patent Publication No. 2005/0144367 (“Sinclair-367”) was filed on December 16, 2004, published on June 30, 2005, and is prior art under AIA 35 U.S.C. § 102.

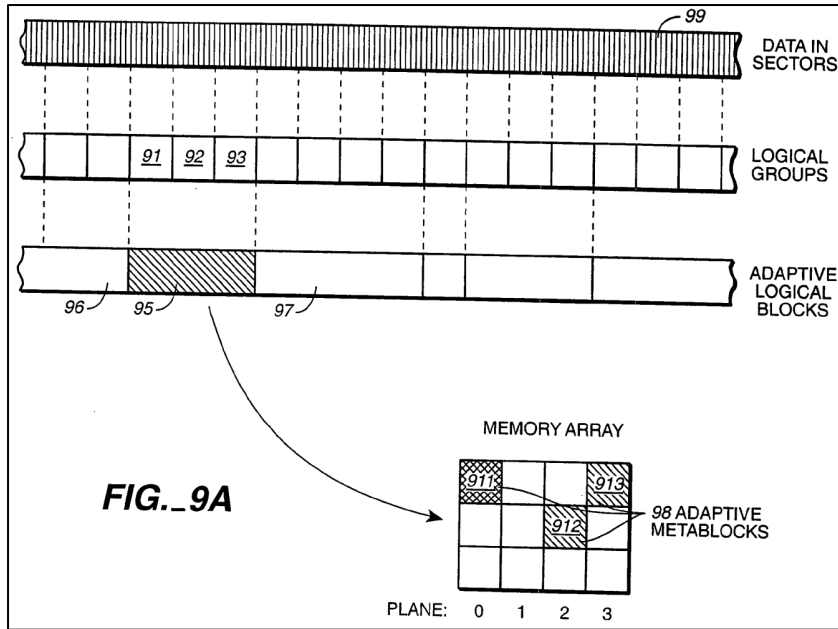
52. Sinclair-367 discloses a non-volatile memory array (*i.e.*, flash memory) via adaptive metablocks. EX1009, Abstract, FIG. 1A:



53. Metablocks are a logical grouping of erase units with each erase unit in a different plane so that reading, writing, and erasing across the metablock occurs simultaneously. EX1009, ¶[0115], FIG. 7:



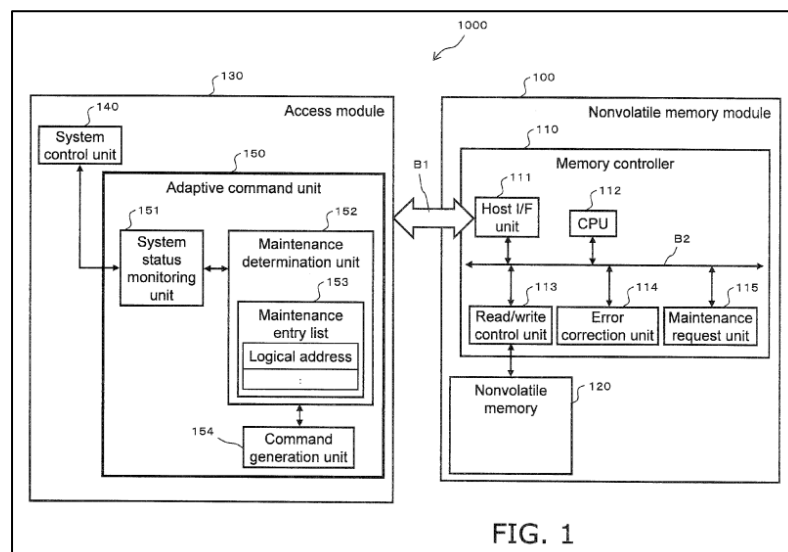
54. Adaptive metablocks are also a logical grouping of erase units, but do not have a fixed size. EX1009, ¶[0117]. Adaptive metablocks might not have one erase unit in each plane (e.g., two erase units in one plane), limiting the extent of simultaneous operation. EX1009, ¶¶[0117], [0119]-[0120], FIG. 9A:



D. Yamada

55. U.S. Publication No. 2011/0271032 (“Yamada”) was filed on July 27, 2010, and was published on November 3, 2011. I understand that it is prior art under AIA 35 U.S.C. § 102(a)(1)-(2).

56. Yamada discloses a maintenance scheme for managing nonvolatile memory module 100 (*i.e.*, flash memory). Specifically, Yamada describes memory controller 110 including a “maintenance request unit,” and access module 130 (*i.e.*, host) including a maintenance determination unit 152 with command generation unit 154. EX1010, ¶¶ [0012], [0017], [0031].



57. Yamada describes that the host analyzes the metadata of blocks or groups of blocks to determine if certain thresholds have been met. EX1010, ¶¶[0013], [0032], [0080]-[0081]. If so, the controller transmits the metadata and address information to the host in “maintenance request” from the “maintenance request unit.” EX1010, ¶¶[0014], [0109]-[0111]. In response, the host may then issue a “maintenance command” from the “maintenance command unit,” commanding the controller to copy data from the block or blocks that need maintenance to new block(s). EX1010, ¶¶[0014], [0109]-[0111].

58. Moreover, Yamada discloses performing maintenance operations during the lulls between sets of normal storage operations. EX1010, Abstract. The maintenance determination unit, by being on the access module, monitors the access module’s status. EX1010, ¶¶[0018]-[0019], [0022]. When the maintenance determination unit finds downtime in normal processing, the maintenance determination unit performs requested maintenance operations. EX1010, ¶¶[0018]-[0019], [0022].

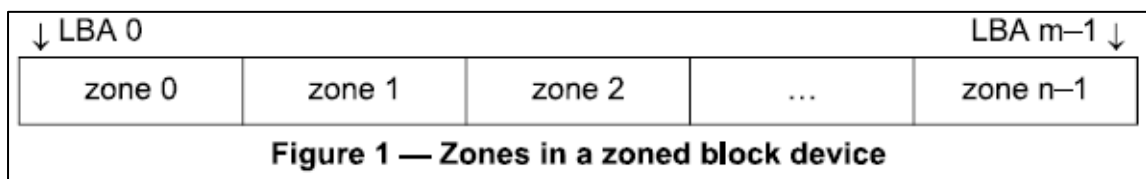
E. ZBC, SBC-4, and SPC-4

59. ZBC, SBC-4, and SPC-4 published on May 15, 2014, May 20, 2014, and May 17, 2014, respectively. EX1100, ¶¶2-10; EX1103; *see also* EX1109 (depicting T10’s website). I understand that all three were published by T10 InterNational Committee for Information Technology Standards (“INCITS”), and publicly available through the T10 website. EX1100, ¶¶2-11; EX1101; EX1103; *see also* EX1109. I also understand that all three are prior art under at least 35 U.S.C. § 102(a)(1).

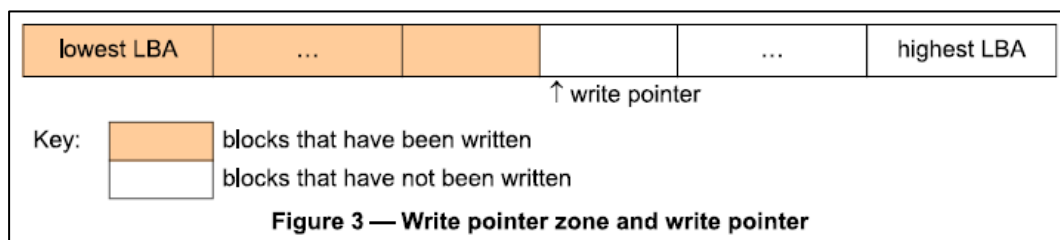
60. I am aware of INCITS, and T10 and T10’s website in particular. T10 publishes SCSI standards and makes publications readily available on T10’s website. *See, e.g.*, EX1101; EX1103. And SCSI protocols and command sets are popular interfaces for storage devices. *See, e.g.*, EX1200; EX1201, 190; EX1202, 1-3; EX1203, 10-12.

61. ZBC is another SCSI command set detailing an interface for a zoned block device model. EX1006, 1, Figure 0. Moreover, ZBC is an extension command set, thereby also accepting commands from core SCSI specifications such as SBC-4 and SPC-4. EX1006, x, 12-13, 22, Figure 0.

62. According to SPC-4, which is “shared across all device types” (EX1006, Figure 0), SCSI logical block devices (*e.g.*, logical units, volumes, and/or partitions) are created for physical storage devices. EX1008, 14, 61. Each logical block device has an independent address range that “shall begin with block zero and be contiguous up to the last block of that logical unit or within that volume or partition.” EX1008, 14. According to ZBC, a “logical unit,” volume, or partition can “be a zoned block device,” (EX1006, 1) and with zoned block devices, the underlying address range is “divided into ranges called zones ... The entire capacity of a zoned block device is organized into a set of contiguous, non-overlapping zones.” EX1006, 13, Figure 1:



63. ZBC includes at least two different kinds of zoned block devices—(1) “host aware zoned block device[s]” and (2) “host managed zoned block device”—and at least two different kinds of zones—(1) “sequential write preferred zone” and (2) “sequential write required zone.” EX1006, 4, 13-14, Table 14. Sequential write preferred zones and sequential write required zones include metadata such as the state of the zone and write pointers. The state of a zone is the zone condition, and includes “EMPTY,” “OPEN,” “READ ONLY,” “FULL,” and “OFFLINE.” EX1006, Table 15. Write pointers generally track the next available logical block address zones have write pointers. EX1006, 14, 16-17, Figure 3:



64. With each write command, the write pointer increments its logical block address until

the end of the zone is reached or a reset write pointer command is applied. EX1006, Table 15. A reset write pointer command resets the write pointer back to the lowest logical block address in the zone and also resets the data in the zone to the manufacturer’s default initialization pattern. EX1006, 3, 15-18.

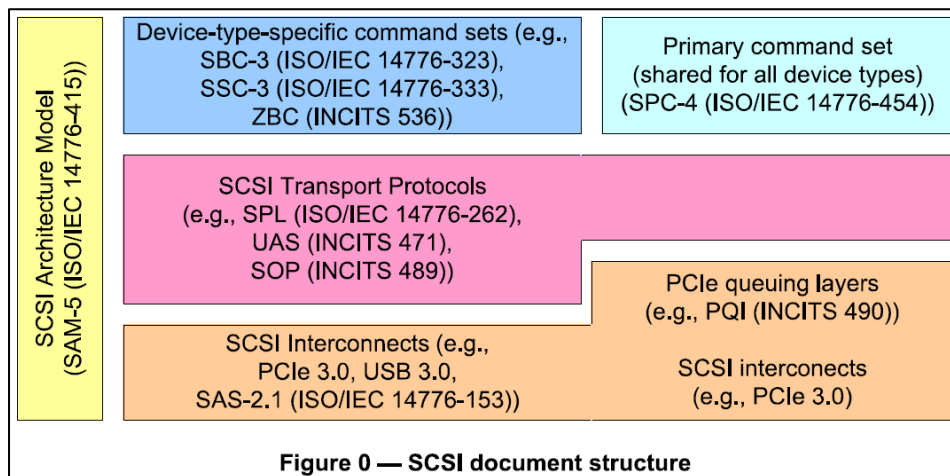
VII. INVALIDITY OF CLAIMS

A. **GROUND I: CLAIMS 1-4, 6-9, 13, 15, AND 19-22 ARE RENDERED OBVIOUS BY REITER IN LIGHT OF ZBC, SBC-4, AND SPC-4**

1. **Motivation To Combine Reiter, ZBC, SBC-4, And SPC-4**

65. A POSITA would be motivated to combine the teachings of ZBC, SBC-4, and SPC-4 (collectively, “ZBC”) with flash system 100 disclosed in Reiter. Specifically, a POSITA would be motivated to use ZBC, SBC-4, and SPC-4 as the standard governing requests sent to Reiter’s flash system 100.

66. **First**, to use ZBC, SBC-4, and SPC-4 as the governing standard, a POSITA would be motivated to combine ZBC, SBC-4, and SPC-4. This combination makes sense. ZBC, SBC-4, and SPC-4 are SCSI specifications intended to be used with one another. *See* EX1006, x, 1, 12-13, 22, Figure 0:

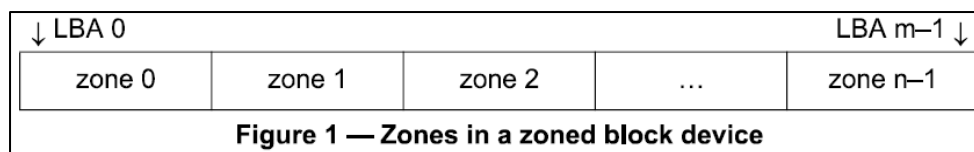


The “purpose of [ZBC] is to define the model and command set extensions to be used in conjunction with ... SPC-4” EX1006, x, 1. ZBC expressly discloses that it is “implemented in conjunction with ... SPC-4.” EX1006, 1. And SPC-4 is listed as “shared for all device types.” EX1006, Figure 0. ZBC

expressly relies not only on SPC-4, but also SBC-4, throughout the specification. EX1006, x, 1, 12-13, 22, Figure 0.

67. **Second**, with a combined ZBC, SBC-4, and SPC-4, a POSITA would be motivated to apply that standard to govern Reiter's flash system 100. Reiter does not disclose using, or otherwise requiring, a particular command set, but generally discloses support for the widely used SCSI protocols. EX1004, 7:9-18. Reiter states that "[c]ommunication link 102 might be a custom-designed communication link, or might conform to a standard communication protocol such as, for example ... SCSI ... protocol bus, ... Serial Attached SCSI ... protocol bus, ... or any other similar interface link for connecting a peripheral device to a computer." EX1004, 7:9-18. Reiter also discloses "receiv[ing] ... requests ... such as read or write operations from ... external devices," where those "requests ... generally include a[] ... logical block address where data should be read or written." EX1004, 8:1-6. Thus, given the ZBC being an SCSI extension and being able to fill Reiter's lack of a detailed interface, a POSITA would be motivated to support commands such as those disclosed in ZBC to enable specific read and write operations based on Reiter's memory structure.

68. Moreover, a POSITA would be particularly motivated to implement Reiter to run on ZBC commands. SCSI protocols and commands such as those disclosed in ZBC are widely used, and would thus improve the compatibility and interoperability of Reiter's flash system. And ZBC's zoned block device model (EX1006, 1) is well suited for Reiter's flash system because it would enable effective utilization of Reiter's superblock memory organization. *See* EX1004, FIG. 4c. ZBC manages an interface for a "zoned block device." EX1006, 1. Zoned block devices have an address space comprising logical block addresses ("LBAs") "divided into ranges called zones." EX1006, 13, 29, Table 16, Figure 1:



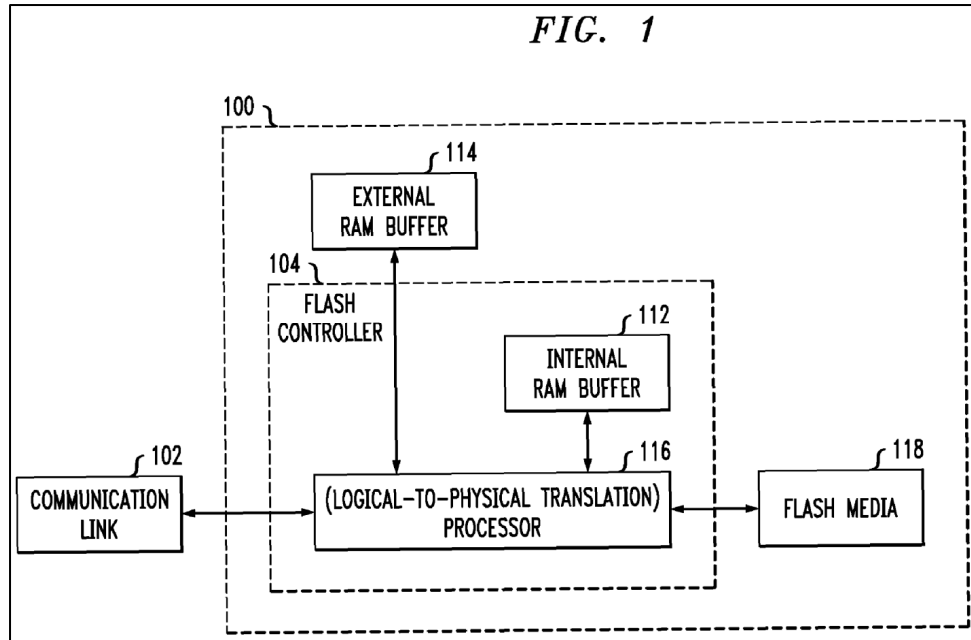
69. Zoned block devices' zones, as governed by ZBC, are analogous to Reiter's superblocks. Like zones, superblocks comprise "a logical collection of blocks representing a fixed range of LBAs." EX1004, 9:11-13. ZBC's zones and Reiter's superblocks include metadata indicating the next available memory location within the zone and superblock, respectively. EX1006, 14, 16-17, Figure 3 (illustrating a zone with a write pointer indicating next available LBA); EX1004, 9:52-10:1, FIG. 4b (illustrating superblock summary page with "free page pointer" indicating next available memory location). And like ZBC, which use LBAs to access different zones, Reiter's superblocks are addressed through the received LBA. EX1004, 8:54-9:25, FIG. 4a (deriving superblock number 410 from logical block address 402); EX1006, 13.

70. A POSITA would have a reasonable expectation of success in adding support for ZBC in Reiter's flash system 100. SCSI commands and protocols such as ZBC were widely known and would have been familiar to a POSITA. SCSI specifications were intended to be used across a wide range of storage devices, including flash memory. A POSITA would have also understood that Reiter's superblock-based memory organization is suitable for zone-based commands like ZBC. Further, a POSITA would have understood that ZBC's zoned block device model is a logical construct that can be layered on top of any storage device.

71. Moreover, Reiter and ZBC disclose analogous art. Both relate to storage devices, including accessing blocks of memory.

2. Claim 1, Limitation 1[pre]: *A storage device comprising:*

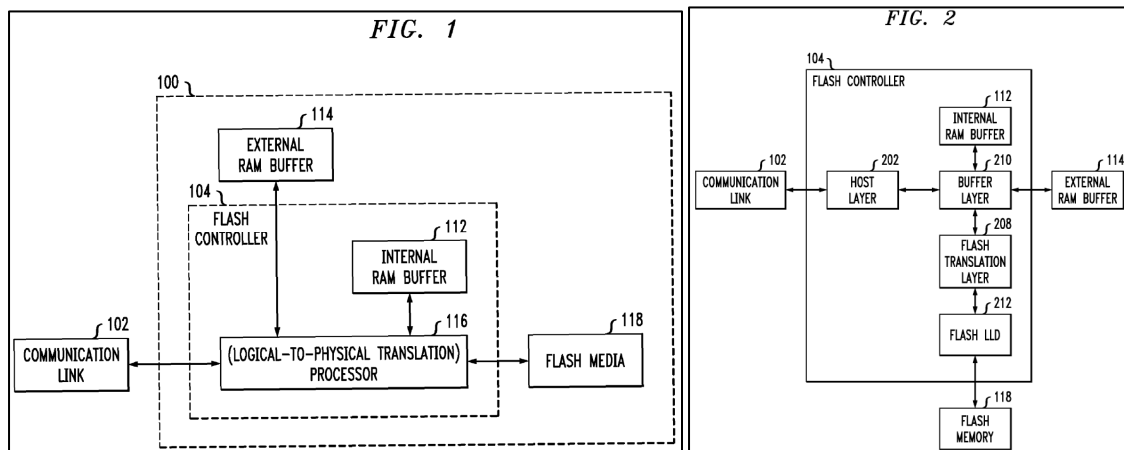
72. Reiter discloses a "flash ... system 100" (*i.e.*, ***storage device***) "implementing a logical-to-physical translation" scheme. EX1004, 6:63-66, FIG. 1:



73. It is my opinion that Reiter renders this limitation obvious.

(a) Limitation 1[a]: *flash memory comprising erase units; and*

74. Reiter’s flash system 100 includes “flash media 118” or “flash memory 118” (*i.e.*, *flash memory*). EX1004, 6:63-7:5, 7:56-61, FIGS. 1, 2:



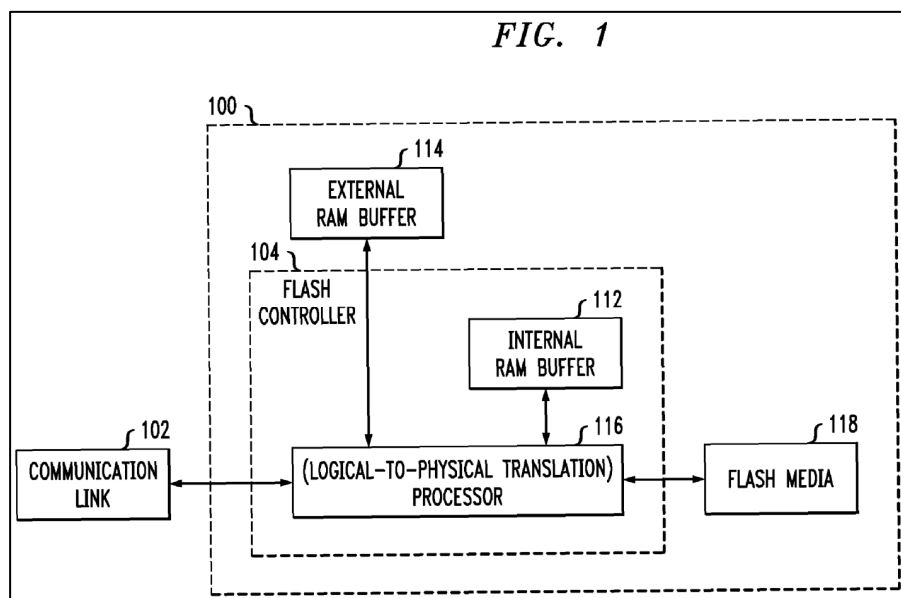
75. Reiter discloses that a “commonly employed type of flash memory technology is NAND flash memory.” EX1004, 1:40-45. “There are three basic operations for NAND devices: read, write, and erase.” EX1004, 1:51-52. Read and write operations are performed on “[p]ages,” which “are typically arranged in blocks.” EX1004, 1:52-57. “[A]n erase operation is performed on a block by block

basis.” EX1004, 1:56-57. Reiter also expressly discloses that its flash media comprises “blocks” that may be erased. EX1004, 6:60-62, 17:61, cls. 2, 9. Reiter thus discloses that its flash memory is made up of independently erasable blocks (*i.e.*, **flash memory having erase units**).

76. It is my opinion that Reiter renders this limitation obvious.

(b) **Limitation 1[b]:** *circuitry to receive from a host, and control execution of, incoming requests including data write requests and erase requests, wherein each of the incoming requests is accompanied by incoming address information;*

77. Reiter’s flash system 100 includes a “flash controller 104” (*i.e.*, **circuitry**). EX1004, 6:63-7:2, FIG. 1:



78. Reiter’s “[F]lash controller 104 **receives** one or more requests for flash media access, such as read or write operations [*i.e.*, **incoming requests**)] from one or more external devices via communication link 102” (*i.e.*, **from a host**). EX1004, 8:1-4. The “requests for access to flash media 118 generally include at least one logical block address (LBA) where data should be read or written” (*i.e.*, **incoming requests is accompanied by incoming address information**). EX1004, 8:4-6.

79. For example, Reiter discloses a “flash media write operation 630” that is “**executed** by FTL 208” in flash controller 104. EX1004, 13:38-46, FIG. 2. The “flash media write operation” is

executed “in response to a flash media write request received from an external device,” which “includes a logical block address (LBA).” EX1004, 13:38-46.

80. A POSITA would be motivated to implement Reiter’s flash controller 14 to accept and execute ZBC commands. Section VIII.B. The ZBC command set includes a write command—a request disclosed in Reiter (EX1004, 8:1-6)—and a reset write pointer command (*i.e., incoming requests including data write requests and erase requests*). EX1007, Table 112; *see also* EX1006, 22 (“The commands for host aware zoned block devices are defined in SBC-4.”). Both ZBC commands include a logical block address (*i.e., wherein each of the incoming requests is accompanied by incoming address information*)—also as disclosed with Reiter’s requests (Reiter, 8:1-6, 11:55-63, 13:3-46). EX1007, Table 112; ZBC, 29, Table 16. A ZBC write command includes a logical block address. EX1007, Table 112:

Table 112 — WRITE (32) command

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (7Fh)							
1		CONTROL							
2		Reserved							
...									
5									
6		Reserved			GROUP NUMBER				
7		ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (000Bh)							
9									
10		WRPROTECT			DPO	FUA	Reserved	Obsolete	Reserved
11		Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS							
...									
19		(LSB)							
20	(MSB)	EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG							
...									
23		(LSB)							
24	(MSB)	EXPECTED LOGICAL BLOCK APPLICATION TAG							
25									
26	(MSB)	LOGICAL BLOCK APPLICATION TAG MASK							
27									
28	(MSB)	TRANSFER LENGTH							
...									
31		(LSB)							

81. A ZBC reset write pointer command “sets a write pointer to the lowest LBA of a write pointer zone.” EX1006, 3. A reset write pointer command includes a “ZONE ID,” which is the first logical block address of the targeted zone. EX1006, 29, Table 16:

Table 16 — RESET WRITE POINTER command								
Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Fh)							
1	Reserved			SERVICE ACTION (14h)				
3	(MSB)							
...	ZONE ID							
9	(LSB)							
10								
...	Reserved							
13								
14	Reserved							RESET ALL
15	CONTROL							

82. A POSITA would be motivated to implement the reset write pointer command to erase data in a zone. After applying a reset write pointer command, the zone’s data is no longer accessible. EX1006, 16-17. For instance, “[a] read operation on an LBA that has not been written since the last reset write pointer operation ... shall return user data set to the last initialization pattern ... set by ... the manufacturer[.]” EX1006, 16-17.

83. If the data is no longer accessible, it should be erased immediately or at least immediately queued for erasure. Immediate erasure is simpler and with less risk of error. The effects of the reset write pointer command is fulfilled immediately, and cannot get tangled up with later commands. Alternatively, immediate queueing for erasure provides flexibility for erasing when normal read and write operations are not being processed. Reiter expressly complains about delays of normal operations for erasure. EX1004, 4:19-26. “[G]arbage collection routines,” such as for “erasing out-of-date blocks,” “impact performance of the flash memory system by utilizing processor resources and potentially delaying write operations” EX1004, 4:19-26.

84. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

(c) **Limitation 1[c]:** *wherein said circuitry is to, for each of the incoming requests:*

85. As addressed in Sections VII.A.2.d-g, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

(d) **Limitation 1[c][i]:** *derive, from the accompanying incoming address information, a first address portion and a second address portion*

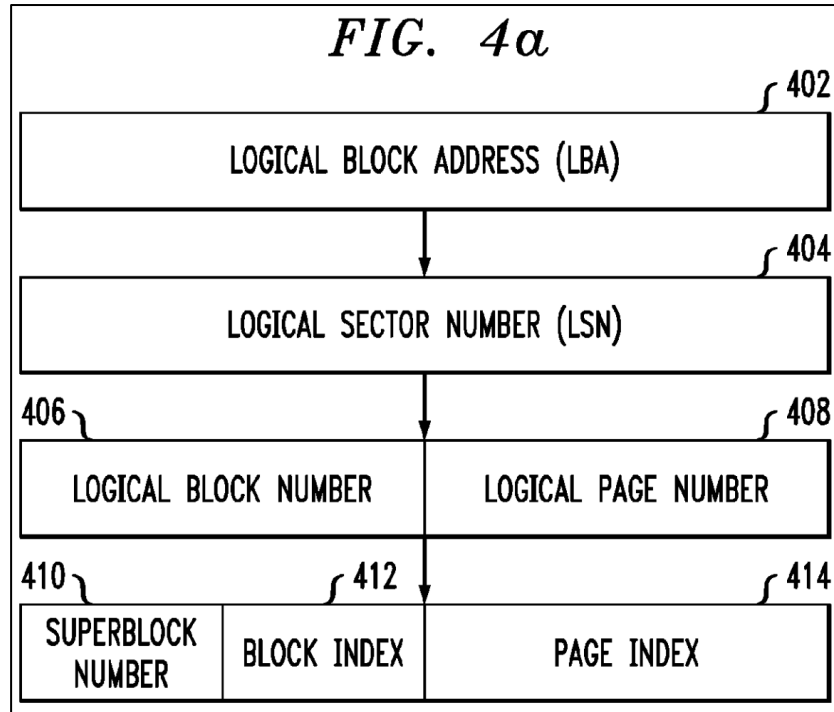
86. From ZBC write and reset pointer commands (*i.e., accompanying incoming address information*), flash controller 104 *derives* (1) a number for a logical unit, volume, or partition (*i.e., first address portion*) and (2) a bitstring made up of superblock number 410, block index 412, and page index 414, or alternatively logical sector number 404 (*i.e., second address portion*).

87. **First Address Portion:** A POSITA would be motivated to incorporate a number associated with a logical unit, volume, or partition in ZBC commands, and motivated to modify flash controller 104 to decode (*i.e., derive*) that number (*i.e., first address portion*) from the command (*i.e., from the accompanying address information*). ZBC read commands target a “logical unit,” “volume,” or “partition” (hereinafter “logical unit”). *See* EX1008, 14, 61; *see also* EX1006, x, 1, 4. An “objective of ... [ZBC]” is “to permit an application client to communicate with a logical unit that declares itself to be a zoned block device” EX1006, 1. But logical units are logical structures, where multiple can exist on the same physical storage device. *See* EX1008, 14 (describing a logical unit as an “externally addressable entity **within a SCSI target device** that implements a SCSI device model”), 462 (“request[ing] ... the peripheral device[’s] logical unit inventory”—a list of logical unit numbers). And each logical unit “shall begin with block zero and be contiguous up to the last logical block of that logical unit” EX1008, 61. With multiple logical units, and overlapping LBA spaces, an LBA by itself is not

enough to identify a location on a physical storage device. A POSITA would look to incorporate commands to distinguish the same LBA to two different logical units.

88. A POSITA would be motivated to incorporate a separate address found in SPC-4 into ZBC commands. Section VII.A.1. SPC-4 recites that a “logical unit” is “[a]n **externally addressable** entity within a SCSI target device” EX1008, 7, 14. Addressing a logical unit is done with a “logical unit number”—“[a]n encoded 64 bit identifier for a logical unit.” EX1008, 14; *see also* EX1006, 4. ZBC already mentions logical unit number. EX1006, 4. A POSITA would be motivated to more fully incorporate that address in its commands via the teachings in SPC-4.

89. **Second Address Portion:** A POSITA would be motivated to use Reiter’s flash controller 104 to *derive* the bitstring made up of superblock number 410, block index 412, and page index 414, or alternatively logical sector number 404 (*i.e., second address portion*), from the LBA in the ZBC write and reset write pointer commands (*i.e., from the accompanying address information*). ZBC write and reset write pointer commands include LBAs. EX1006, Table 16; EX1007, Table 112. Reiter discloses that read and write “request[s] ... include[] a[n] ... LBA” EX1004, 8:4-6, 8:62-67. “FIG. 4a shows an exemplary relation of a[n] ... LBA ... to a physical address ... as managed by FTL” EX1004, 8:56-59, FIG. 4a:



Translating an LBA results in logical sector number 404, followed by a bitstring of superblock number 410, block index 412, and page index 414. EX1004, 8:62-67, 9:5-20, FIG. 4a.

90. A POSITA would understand the (1) LBA 402, (2) LSN 404, (3) logical block number 406 and logical page number 408, and (4) superblock number 410, block index 412, and page index 414 in FIG. 4a to be bitstrings. Each is depicted as one unit. EX1004, FIG. 4a (above). This matches the single unit depiction of logical block address 402 and logical sector number 404 bitstrings in the prior translation steps. EX1004, FIG. 4a (above). Other prior art references depicting bits as one block with separate portions of the block associated with different values (EX1004, FIG. 4a (above for superblock number 410, block index 412, and page index 414)) describe that block as a bitstring, and the values as bitfields within the bitstring. *See* EX1211, 5:7-10 (“The logical unit number is the high order binary digits of the logical address and may be derived from the logical address by a bit shift operation.”), FIG. 4; EX1210, ¶[0157], FIG. 33.

91. A POSITA would also be motivated to store or keep the bitfields of each translation level together in bitstrings (as depicted in FIG. 4a). This creates a simple system where one translation

step is only started after completing the prior completion step (*see* EX1004, FIG. 4a), because the resulting bitfields need to be produced and then combined into one bitstring prior to the next translation step. Ensuring lock-step translation of addresses simplifies management where (1) bitfields are stored all over the place and gathered from different parts of memory and (2) some bitfields are further along translation than other bitfields associated with the same address. This simplicity becomes increasingly important with concurrent and/or out-of-order execution of read/write/erase commands.

92. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

(e) **Limitation 1[c][ii]:** *identify an addressed block device from the first address portion,*

93. The number associated with a logical unit (*i.e., from the first address portion*) *identif[ies]* the targeted logical unit (*i.e., addressed block device*). A logical unit has an independent range of logical block addresses starting at zero. EX1008, 61.

94. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

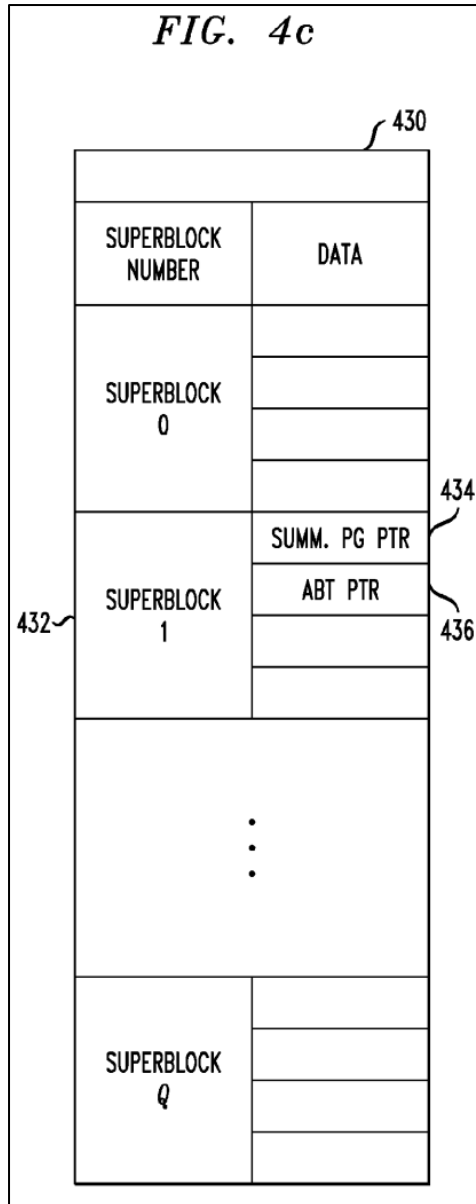
(f) **Limitation 1[c][iii]:** *use a first division operation on the second address portion, to identify an addressed segment within the addressed block device, wherein the addressed segment corresponds to a subset of the erase units of the flash memory, and*

95. A POSITA would understand *using*, or be motivated to *use*, integer *division operation on the* superblock-block-page bitstring (*second address portion*) to extract superblock number 410. Alternatively, Reiter discloses *using* integer *division operation[s] on* logical sector number 404 (*second address portion*) to obtain superblock number 410. EX1004, 8:64-67, 9:5-20, FIG. 4a. Superblock number 410 *identifies* a superblock, which a POSITA would be motivated to correspond the superblock to a ZBC zone (*i.e., an addressed segment*). Section VII.A.1; *see also* EX1011 at 24 (“Samsung’s Accused Instrumentalities ... identify an addressed zone (segment).”).

96. **Integer Division On Superblock-Block-Page Bitstring**: Superblock number 410 is a bitfield in the bitstring of superblock number 410, block index 412, and page index 414. EX1004, FIG. 4a. Integer division by a certain power-of-two extracts bitfields out of bitstrings. Specifically, the quotient of integer division by certain power-of-two will extract the most significant bitfields (*i.e.*, bits on the lefthand portion of a bitstring). Integer division is preferable because it is fast, as division by a power-of-two on a bitstring is implemented with a simple bit shift operation. Superblock number 410 bitfield is the higher order or most significant bitfield (*i.e.*, leftmost bits in the bitstring) of the superblock-block-page bitstring. EX1004, FIG. 4a. Applying integer division on the bitstring causes superblock number 410 to be extracted through the quotient. Because superblock number 410 is a bitfield, a POSITA would be motivated to implement integer division via a bit shift operation to perform the extraction quickly.

97. **Alternative, Integer Division On Logical Sector Number 404**: Reiter discloses “deriv[ing] [logical block number] 406 by dividing LSN 404 by a number of sectors per block of flash media 118,” and “deriv[ing] [logical page number] 408 by dividing LSN 404 by a number of sectors per page of flash media.” EX1004, 9:5-9, FIG. 4a. From logical block number 406, superblock number 410 is obtained. EX1004, 9:9-11, 9:13-17, FIG. 4a.

98. **Identify An Addressed Segment**: Superblock number 410 identifies a superblock by selecting, from page global director (PGD) 430, the “summary page associated with [the] Superblock.” EX1004, 10:13-19, FIG. 4c:



99. A POSITA would be motivated to use the superblocks disclosed in Reiter as zones for ZBC commands. Section VII.A.1. “[A] Superblock generally is a logical collection of blocks” (*i.e., addressed segment corresponds to a subset of the erase units of the flash memory*). EX1004, 9:11-14, FIG. 4c.

100. Further, superblock number 410 is derived from LBA 402 (EX1004, FIG. 4a), which is from a ZBC command’s LBA (Section VIII.D.1.e). Because a ZBC LBA is within the LBA range specified by the targeted logical unit (*see* Sections VIII.D.1.e-f; EX1006, x, 1, 4; EX1008, 61),

superblock number 410 identifies a superblock *within* the targeted logical unit (*addressed block device*).

101. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

- (g) **Limitation 1[c][iv]:** *for each of the data write requests, identify from the first division operation a third address portion, and use a second division operation on the third address portion to identify an address value to select one of the erase units from the subset which corresponds to the addressed segment and to identify a storage location within the selected one of the erase units;*

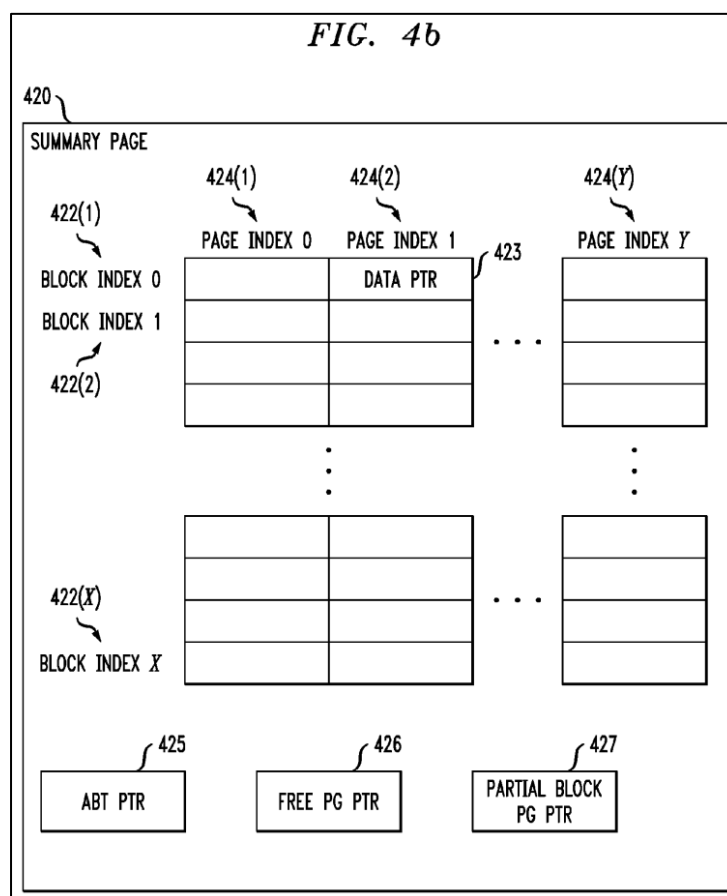
102. A POSITA would understand using, or be motivated to use, integer division on the superblock-block-page bitstring (*i.e., first division operation*) bitstring, or alternatively integer divisions on logical sector number 404 (*i.e., first division operation*), to obtain superblock number 410 through the quotient. Section VII.A.2.f. Such division operations leave as a remainder a block-page bitstring of block index 412 and page index 414 (EX1004, FIG. 4a) or alternatively, for the latter, results a block-page bitstring of logical block number 406 and logical page number 408 (EX1004, 9:5-9, FIG. 4a) (*identify from the first division operation a third address portion*).

103. Reiter discloses further translation for write requests (*i.e., ZBC write commands* (Section VII.A.2.b)) (*i.e., for each of the data write requests*). EX1004, 8:56-58, 9:60-63 (indexing summary page 420 with block index 412 and page index 414 to retrieve data pointer 423), 13:38-46, FIGs. 4a-b, 6b.

104. **Second Division Operation On Logical Block Number 406 And Logical Page Number 408:** Reiter discloses performing a second set of division operations on logical block number 406 and logical page number 408 (*use a second division operation on the third address portion*) to *identify* block index 412 (*address value*) and page index 414 (*storage location*). Specifically, Reiter discloses “deriv[ing] ... block index 412 from LBN 406 by dividing LBN 406 by a number of blocks per Superblock, where ... block index 412 corresponds to the remainder,” and “[p]age index 414 is derived

from LPN 408 by dividing LPN 408 by a number of pages per block.” EX1004, 9:13-20.

105. Block index 412 (*address value*) *selects* one of the physical blocks in the superblock (*erase units from the subset which corresponds to the addressed segment*). “[A] Superblock generally is a logical collection of blocks” EX1004, 9:11-14. And the block index 412 selects a block within the superblock’s collection. *See* EX1004, 9:13-17, 9:52-58, FIGS. 4b-c. “Each Superblock has a summary page ... 420,” which “contains mapping data for the ... Superblock.” EX1004, 9:52-54, FIG. 4b. “For example, summary page 420 contains the block index and page index ... for all X blocks and Y pages in [the] Superblock.” EX1004, 9:54-58, FIG. 4b:



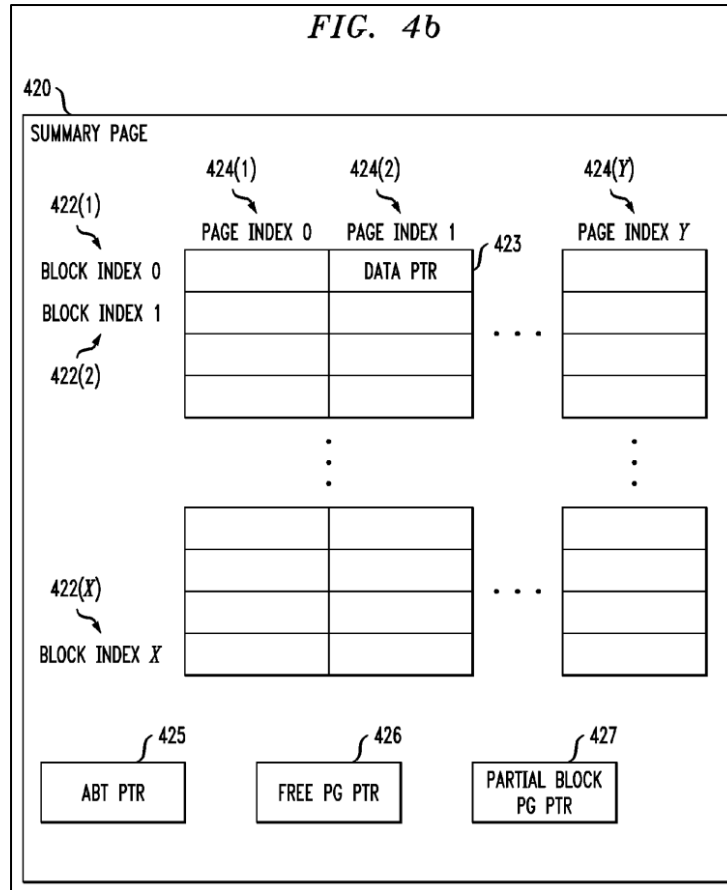
106. Block index 414 indexes the “summary page block indices [that] include all physical blocks ... within the Superblock.” EX1004, 9:58-60, FIG. 4b.

107. Similarly, page index 414 (*storage location*) selects one of the pages within one of

the selected physical blocks (*within the selected one of the erase units*). As previously stated, “summary page 420 contains the block index and page index ... for all X blocks and Y pages in [the] Superblock.” EX1004, 9:54-58, FIG. 4b (above).

108. **Alternative, Second Division Operation On Bitstring Of Block Index 412 And Page Index 414:** A POSITA would understand using, or be motivated to use, the same division operation (*i.e., use a second division operation on the third address portion*), that subdivided off superblock number 410 (Section VII.A.2.f), to separate the bitstring of block index 412 and page index 414 into respective bitfields—block index 412 (*i.e., to identify an address value*) and page index 414 (*i.e., to identify a storage location*). Block index 412 and page index 414 would be the quotient and remainder, respectively, from integer division.

109. Block index 412 (*i.e., address value*) *selects* one of the physical blocks in the superblock (*i.e., erase units from the subset which corresponds to the addressed segment*), and page index 414 (*storage location*) selects one of the pages within one of the selected physical blocks (*within the selected one of the erase units*). EX1004, FIG. 4b:



110. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

(h) **Limitation 1[d]:** *wherein said circuitry is to execute each one of the data write requests by programming accompanying data into the storage location that was identified for that one of the data write requests; and*

111. A POSITA would be motivated to implement Reiter’s flash controller 104 (*i.e., said circuitry*) to *execute* ZBC commands, including write commands (*i.e., each one of the data write requests*). Section VII.A.2.b. Reiter discloses flash controller 104’s “determin[ing] the Superblock number, Block index, and Page index of the first page to be written” (*i.e., storage location that was identified for that one of the data write requests*), and the “requested data is written at the active page address at step 644” (*i.e., programming accompanying data*). EX1004, 13:38-60, FIG. 6b.

112. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4

renders this limitation obvious.

- (i) **Limitation 1[e]:** *wherein said circuitry is also to execute each one of the incoming erase requests by controlling erasure in the flash memory of one or more of the erase units in the subset corresponding to the addressed segment that was identified from the incoming address information accompanying the one of the incoming erase requests.*

113. A POSITA would be motivated to implement Reiter's flash controller 104 (*i.e., said circuitry*) to accept and *execute* ZBC commands, including reset write pointer commands (*i.e., each one of the incoming erase requests*). Section VII.A.2.b. A POSITA would be motivated to implement the reset write pointer command to erase data in the zone targeted by the command (*i.e., addressed segment that was identified from the incoming address information accompanying the one of the incoming erase requests*). Sections VII.A.2.b, VII.A.2.d-f. A POSITA would be motivated to use the superblocks disclosed in Reiter as zones for ZBC commands. Section VII.A.1. "[A] Superblock generally is a logical collection of blocks." EX1004, 9:11-14. A POSITA would understand that erasing a superblock erases the blocks of that superblock (*i.e., controlling erasure in the flash memory of one or more of the erase units in the subset corresponding to the addressed segment*). *See also* EX1004, 6:60-62, 10:4-7, 17:39-46, 17:61-63.

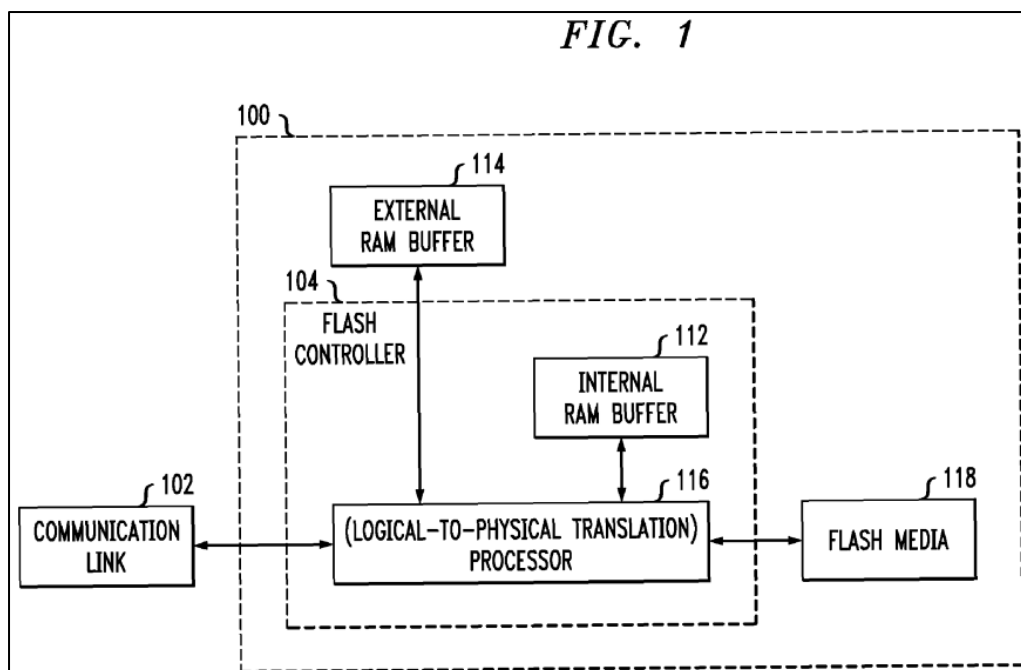
114. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

3. Claim 2, Limitation 2[pre]: *The storage device of claim 1 wherein:*

115. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 disclose this limitation.

- (a) **Limitation 2[a]:** *said circuitry comprises at least one processor; and*

116. Reiter's flash controller 104 (*i.e., said circuitry*) *comprises* "processor 116" (*i.e., at least one processor*). EX1004, 7:21-22, 7:31-39, FIG. 1:



117. It is my opinion that Reiter renders this limitation obvious.

- (b) **Limitation 2[b]:** *the storage device further comprises instructions stored on at least one physical storage medium, said instructions when executed to control the at least one processor.*

118. Reiter's flash system (*i.e.*, **storage device**) includes flash controller 104. EX1004, FIG. 1. And flash controller 104 includes processor 116. EX1004, 7:19-39, FIG.1. "Processor 116 includes software and/or firmware as needed for operation" (*i.e.*, **said instructions when executed control the at least one processor**), which a POSITA would understand as including **instructions stored on at least one physical storage medium**. EX1004, 7:31-34.

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

4. Claim 3, Limitation 3[pre]: *The storage device of claim 1 wherein:*

120. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 disclose this limitation.

- (a) **Limitation 3[a]:** *said circuitry can receive a request from a host to copy host-selected data from one storage location in flash memory to another storage location in the flash memory, and is to issue commands to the flash memory, in response to the request*

from the host to copy, to read the host-selected data from a first physical storage location in the flash memory and write the host-selected data to a second physical storage location in the flash memory without sending the data being copied to the host;

121. A POSITA would be motivated to implement Reiter's flash controller 104 (*i.e.*, *said circuitry*) to *receive* and execute ZBC commands. Section VII.A.2.b. The ZBC command set includes an extended copy command (*i.e.*, *request from a host to copy*). EX1006, 1, 22; EX1007, Table 33; EX1008, 287-89, Table 128. The extended copy command "cop[ies] data from one set of copy sources (e.g.,... logical units) to a set of copy destinations (e.g.,... logical units)" on flash system 100 (*i.e.*, *copy host-selected data from one storage location in flash memory to another storage location in flash memory*). EX1008, 287-89. Specifically, the extended copy command includes a parameter list, which has a set of segment descriptors. EX1008, Table 128:

Table 128 — EXTENDED COPY(LID4) parameter list (part 1 of 2)								
Bit Byte	7	6	5	4	3	2	1	0
0	PARAMETER LIST FORMAT (01h)							
1	Reserved		STR	LIST ID USAGE		PRIORITY		
2	(MSB) _____							
3	HEADER CSCD DESCRIPTOR LIST LENGTH (0020h)							(LSB)
4	_____							
...	Reserved							_____
14	_____							
15	Reserved					G_SENSE	IMMED	
16	HEADER CSCD DESCRIPTOR TYPE CODE (FFh)							
17	_____							
18	Reserved							_____
19	_____							
20	(MSB) _____							
...	LIST IDENTIFIER							_____
23	_____							(LSB)
24	_____							
...	Reserved							_____
41	_____							
42	(MSB) _____							
43	CSCD DESCRIPTOR LIST LENGTH (n-47)							(LSB)
44	(MSB) _____							
45	SEGMENT DESCRIPTOR LIST LENGTH (m-n)							(LSB)
46	(MSB) _____							
47	INLINE DATA LENGTH (k-m)							(LSB)
CSCD descriptor list								
48	_____							
...	CSCD descriptor [ID 1] (see 6.4.5)							_____
79	_____							
⋮								
n-31	_____							
...	CSCD descriptor [ID x] (see 6.4.5)							_____
n	_____							

Table 128 — EXTENDED COPY(LID4) parameter list (part 2 of 2)								
Bit Byte	7	6	5	4	3	2	1	0
	Segment descriptor list							
n+1	Segment descriptor (see 6.4.6) [first]							
...								
	⋮							
...	Segment descriptor (see 6.4.6) [last]							
m								
m+1	Inline data							
...								
k								

122. The segment descriptors detail the source logical block address (*i.e.*, **one storage location**) and destination logical block address (*i.e.*, **another storage location**). EX1008, Table 153:

Table 153 — Block device to block device segment descriptor													
Bit Byte	7	6	5	4	3	2	1	0					
0	DESCRIPTOR TYPE CODE (02h or 0Dh)												
1	Reserved						DC	CAT					
2	(MSB)	DESCRIPTOR LENGTH (0018h)						(LSB)					
3													
4	(MSB)	SOURCE CSCD DESCRIPTOR ID						(LSB)					
5													
6	(MSB)	DESTINATION CSCD DESCRIPTOR ID						(LSB)					
7													
8	Reserved												
9	Reserved												
10	(MSB)	BLOCK DEVICE NUMBER OF BLOCKS						(LSB)					
11													
12	(MSB)	SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS						(LSB)					
...													
19		DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS						(LSB)					
20	(MSB)												
...								(LSB)					
27													

123. A POSITA would understand, or find it obvious, that executing the extended copy command (*i.e., in response to the request from the host to copy*) includes flash controller 104 (*i.e., circuitry*) to issue commands reading data from the source logical block address (*i.e., host-selected data from a first physical storage location in the flash memory*), and writing data to the destination logical block address (*i.e., write the host-selected data to a second physical storage location in the flash memory*). EX1008, 245, 270.

124. SPC-4 discloses executing the extended copy command only via a “copy manager” which “may be contained in... a standalone SCSI target device”—Reiter’s flash system 100—(*i.e., without sending the data being copied to the host*). EX1008, 245-47, 287-89.

125. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

(b) **Limitation 3[b]:** *the request from the host to copy is to designate destination address information corresponding to the second physical storage location; and*

Limitation 3[c]: *the circuitry is to use the first division operation on the designated destination address information to identify a destination segment within a destination block device.*

126. SPC-4 discloses the extended copy command from the host (*i.e., the request from the host to copy*) includes a destination logical block address (*i.e., designate destination address information*). Section VII.A.4.a. A POSITA would be motivated to use flash controller 104 (*i.e., circuitry*) and perform the same translation as provided to ZBC write commands on the destination logical block address to a corresponding physical storage location (*i.e., designate destination address information corresponding to the second physical storage location*), which would include *using the first division operation on the designated destination address information to identify a destination segment within a destination block device*. See Sections VII.A.2.c-g.

127. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

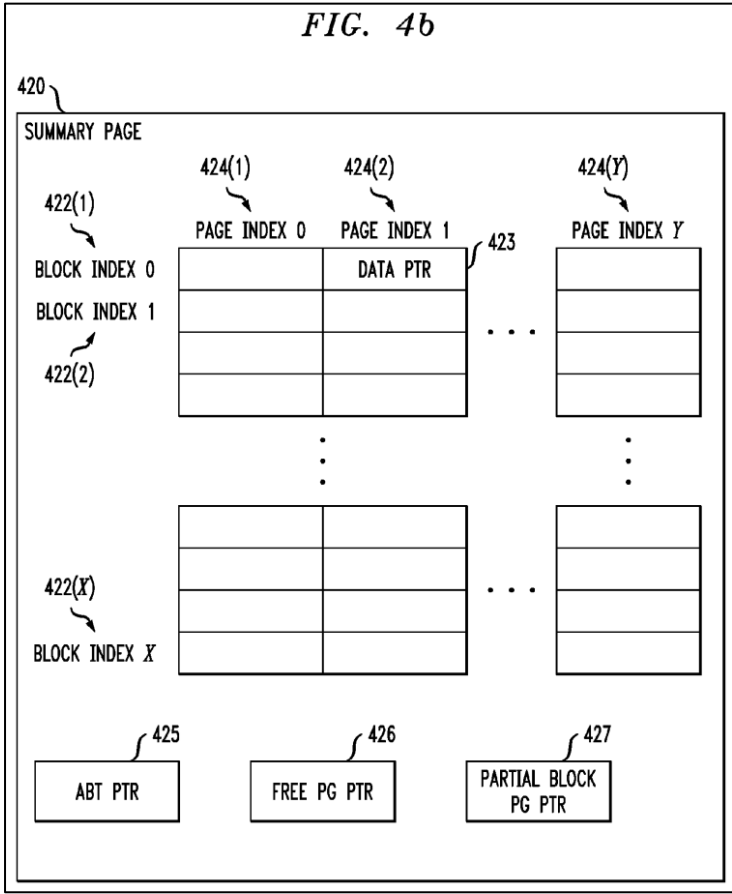
5. Claim 4, Limitation 4[pre]: *The storage device of claim 3 wherein*

128. As addressed in Section VII.A.4, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 disclose this limitation.

(a) **Limitation 4[a]:** *the circuitry is to identify from the destination address information a destination logical erase unit and is to select the second physical storage location to be within a physical erase unit that is already mapped to the destination logical erase unit.*

129. A POSITA would be motivated to implement Reiter's flash controller 104 (*i.e., circuitry*) to accept and execute ZBC commands, including extended copy commands. Section VII.A.4.a. A POSITA would understand, or find it obvious, that the destination logical block address decoded from a ZBC extended copy command (*i.e., identify from the destination address information*) is translated by Reiter's flash controller 104 into block index 412 (*i.e., logical erase unit*). Section VII.A.2.d-g.

130. Block index 412 is a **logical erase unit**. Block index is a logical unit (*i.e.*, “**logical erase unit**”), indexing a superblock’s “logical collection of blocks”—not the physical block number itself. EX1004, 9:11-13, FIG. 4b-c. Block index 412 refers to an erase unit (*i.e.*, “**logical erase unit**”), as block index 412 indexes the blocks in a superblock, and “an erase operation is performed on a block by block basis.” EX1004, 1:56-57. “For example, summary page 420 contains the block index and page index ... for all X blocks and Y pages in [the] Superblock” EX1004, 9:54-58, FIG. 4b:



131. And block index 414 indexes the “summary page block indices [that] include all physical blocks ... within the Superblock.” EX1004, 9:58-60, FIG. 4b. And as disclosed in Reiter, “[p]ages are typically arranged in blocks, and an erase operation is performed on a block by block basis.” EX1004, 1:56-57.

132. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4

renders this limitation obvious.

6. Claim 6, Limitation 6[pre]: *The storage device of claim 1 wherein*

133. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 disclose this limitation.

- (a) **Limitation 6[a]:** *the storage device is to store information identifying a number of structures in the flash memory and is to transmit the information identifying the number of structures to a host, and wherein the circuitry is to fulfill the at least one of the incoming requests by issuing commands to the flash memory for which a quantum of flash memory selected by the commands which are issued in response to the at least one request is dependent on the information transmitted to the host identifying the number of structures.*

134. A POSITA would be motivated to implement Reiter's flash memory controller 104 (*i.e., storage device*) to accept and execute ZBC commands. Section VII.A.2.b. The ZBC command set includes a report zones command. EX1006, 24. The report zones command optionally "return[s]" (*i.e., transmit*) a "[l]ist [of] zones in the zoned block device" back *to [the] host* via "parameter data" that includes the length of the zone list and zone descriptors indicating the starting logical block address and length of each zone (*i.e., the information identifying the number of structures to a host*). EX1006 25, Table 11, Tables 12, 13:

Table 12 — REPORT ZONES parameter data								
Bit Byte	7	6	5	4	3	2	1	0
1	(MSB)							
...	ZONE LIST LENGTH (n-63)							
3	(LSB)							
4	Reserved							SAME
5	Reserved							
...								
63								
Zone descriptors list								
64	Zone descriptor [first]							
...								
127								
	⋮							
n-63	Zone descriptor [last]							
...								
n								

Table 13 — Zone descriptor format								
Bit Byte	7	6	5	4	3	2	1	0
0	Reserved				ZONE TYPE			
1	ZONE CONDITION				Reserved			RESET
2	Reserved							
...								
7								
8	(MSB)							
...	ZONE LENGTH							
15	(LSB)							
16	(MSB)							
...	ZONE START LBA							
23	(LSB)							
24	(MSB)							
...	WRITE POINTER LBA							
31	(LSB)							
32	Reserved							
...								
63								

135. Because flash system 100 transmits the zone list information back to the host, a POSITA would understand that flash system 100 also stores that information (*i.e.*, *store information*

identifying a number of structures in the flash memory). Or a POSITA would be motivated to store such information. As opposed to wasting time re-calculating such information with every command requiring zone metadata, flash system could store zone metadata and update it as zone settings are changed.

136. Later read and reset write pointer commands by the host (*i.e., at least one of the incoming requests*) depend on the information reported by the zone report command to ensure valid read and reset write pointer commands (*i.e., at least one request is dependent on the information transmitted to the host identifying the number of structures*). Valid read commands require reading an “LBA that has been written since the last reset write pointer operation.” EX1006, 16-17. Valid reset write pointer commands require the “ZONE ID field [to] specif[y] the zone start LBA” EX1006, 29. In order to keep read and reset write pointer commands valid, the list of zones with the zone metadata is needed. See EX1006, 25, Table 11, Tables 12, 13.

137. A POSITA would be motivated to implement Reiter’s flash system’s 100 flash controller 104 (*storage device comprises circuitry*) to accept and execute ZBC read and reset write pointer commands (*i.e., the storage device comprises circuitry to fulfill the at least one request by issuing commands to the flash memory for which a quantum of flash memory selected by the commands*). Sections VII.A.2.b-i, VII.A.4.b.

138. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

7. Claim 7, Limitation 7[pre]: *The storage device of claim 1 wherein*

139. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 discloses this limitation.

(a) Limitation 7[a]: *the storage device is to store information identifying a number of structures in the flash memory and is to*

transmit to a host information that identifies an amount of flash memory that is to be erased as a unit.

140. A POSITA would be motivated to implement Reiter’s flash memory controller 104 (*i.e., storage device*) to accept and execute ZBC commands. Section VII.A.2.b. The ZBC command set includes a report zones command. EX1006, 24. The list of zones reported by the report zone command (*i.e., the information transmitted to the host identifying the number of structures*) also provides the length of each zone (*i.e., identifies an amount of flash memory that is to be erased as a unit*). The report zones command optionally “return[s] ... parameter data” that includes a zone descriptor for each zone indicating the length of each zone. EX1006, 25, Tables 11-12, Table 13:

Table 13 — Zone descriptor format								
Bit Byte	7	6	5	4	3	2	1	0
0	Reserved				ZONE TYPE			
1	ZONE CONDITION				Reserved			RESET
2	Reserved							
...								
7								
8	(MSB)	ZONE LENGTH						
...								
15		(LSB)						
16	(MSB)	ZONE START LBA						
...								
23		(LSB)						
24	(MSB)	WRITE POINTER LBA						
...								
31		(LSB)						
32	Reserved							
...								
63								

141. As discussed with Section VII.A.2.b, a POSITA would be motivated to implement the reset write pointer command to cause erasure of the targeted zone. Because the reset write pointer command erases an entire zone, the zone length *identifies an amount of flash memory that is to be erased as a unit*.

142. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

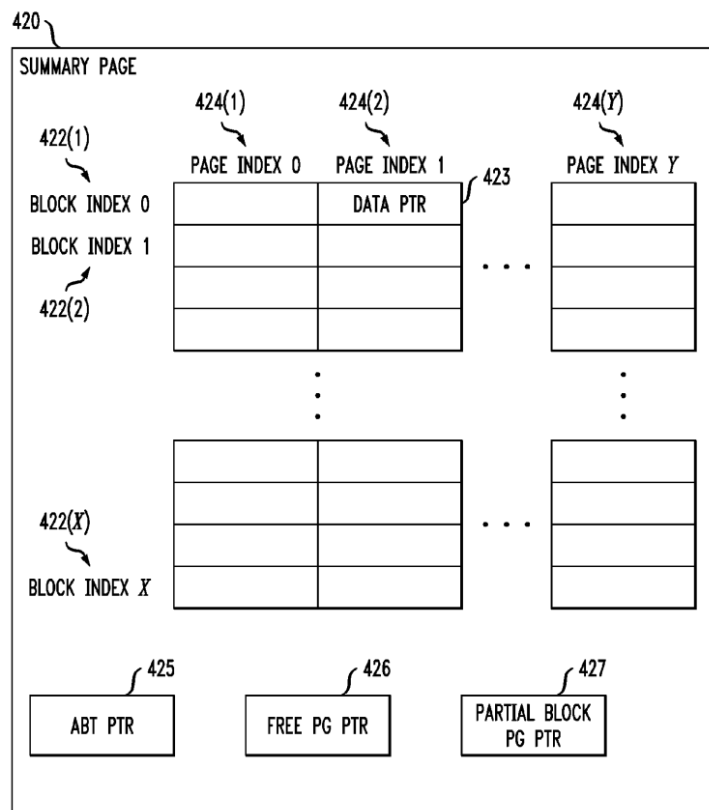
8. Claim 8, Limitation 8[pre]: *The storage device of claim 1 wherein*

143. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 discloses this limitation.

(a) Limitation 8[a]: *the third address portion specifies a location within a sequential range of locations that correspond to the erase units in the subset.*

144. The block-page bitstring—bit string of block index 412 and page index 414—constitutes the *third address portion*. Section VII.A.2.f. Both block index 412 and page index 414 are from a set of sequential indexes for selecting within a superblock’s list of blocks and pages within those blocks (*i.e., specifies a location within a sequential range of locations that correspond to the erase units in the subset*). Section VII.A.2.f. Specifically, “summary page 420 contains the block index and page index ... for all X blocks and Y pages in [the] Superblock” EX1004, 9:54-58, FIG. 4b:

FIG. 4b



145. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

9. **Claim 9, Limitation 9[pre]:** *The storage device of claim 1 wherein*

146. As addressed in Section VII.A.8, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 discloses this limitation.

(a) **Limitation 9[a]:** *the first division operation and the second division operation each subdivide a value obtained from the incoming address information accompanying each of the write requests into respective bitfields.*

147. My opinion for Sections VII.A.2.f-g renders this limitation obvious.

**10. Claim 13, Limitation 13

```
: The storage device of claim 1 wherein:
```**

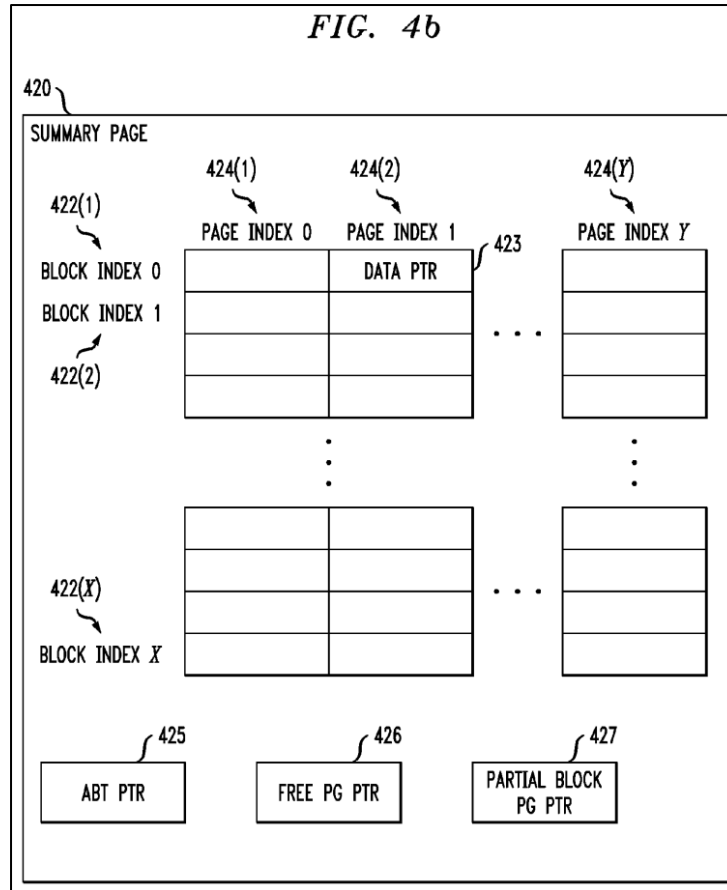
148. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 discloses this limitation.

(a) **Limitation 13[a]:** *the erase units of the flash memory are physical erase units;*

149. Given the similarity in claim language, my opinion for Section VII.A.2.a applies to this limitation.

(b) **Limitation 13[b]:** *the circuitry is to use the second division operation to identify a logical erase unit, wherein each logical erase unit is mapped to one of the physical erase units; and*

150. A POSITA would understand, or find it obvious, to use integer division (*i.e.*, ***second division operation***) to extract block index 412 (*i.e.*, ***logical erase unit***). Section VII.A.2.f. Block index 412 is a ***logical erase unit*** that *maps to one of the physical erase units*. Block index is a logical unit (*i.e.*, “***logical erase unit***”), referring to an index number in a logical list—a block within a superblock, which is a “logical collection of blocks”—to obtain “data pointer 423, which points to the physical address” (*i.e.*, *each logical erase unit is mapped to one of the physical erase units*). EX1004, 9:11-13, 9:54-58, 9:60-63, FIG. 4b:



151. And a block is an *erase unit* (i.e., “**logical erase unit**”), because “an erase operation is performed on a block by block basis.” EX1004, 1:56-57.

152. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

- (c) **Limitation 13[c]:** *the storage device is to detect a failure condition of the physical erase units and is to remap each logical erase unit which corresponds to one of the physical erase units for which the failure condition is detected to instead correspond to one of the physical erase units for which the failure condition has not been detected and is to utilize the remapped correspondence to service one or more of the incoming data access requests.*

153. Reiter’s flash storage system 100 (*storage device*) *detects* bad blocks (*failure condition of the physical erase units*). Upon detecting a bad block, a POSITA would be motivated to *remap* logical block number 406, and block index 412 (*logical erase unit*), from the bad block (*one of*

the physical erase units for which the failure condition is detected) to instead a working block (one of the physical erase units for which the failure condition has not been detected).

154. **Detect A Failure Condition Of The Physical Erase Unit:** Reiter discloses “maintain[ing] a record of bad blocks” by scanning for bad blocks after powering on flash system 100. EX1004, 4:27-37, 17:13-15, 17:47-63. For example, “[o]n startup of flash ... system 100, mapping data ... requires reconstruction.” EX1004, 17:13-15. This includes “scan[ning] and group[ing] ... blocks” into various “categories,” including “bad blocks.” EX1004, 17:47-53. Bad blocks are detected if (1) FTL 208 is “unsuccessful” in “read[ing] metadata that is stored in the ... first page of the block” and (2) the following “erase operation of the [same] block was unsuccessful, the block cannot be erased and has failed.” EX1004, 17:59-67. “[T]he block address is [then] added to the bad block list.” EX1004, 18:4-8.

155. **Remap Each Logical Erase Unit Which Corresponds To One Of The Physical Erase Units For Which The Failure Condition Is Detected:** When a block is added to the bad block list, a POSITA would be motivated to replace the bad block with a working block—remapping logical block number 406, and block index 412, to a new block. Data from a bad block cannot be trusted, especially given the unsuccessful read during the startup scan and subsequent failed erasure of the block. 4:27-37, 17:13-15, 17:47-63. Keeping a bad block around risks reading erroneous data. As a first remapping step, Reiter discloses, upon detecting a bad block, adding the “block address ... to the bad block list.” EX1004, 18:4-8.

156. A POSITA would be motivated to reconstruct the lost data through Reiter’s RAID functionalities, and change data pointer 423 from the old bad block to a new working block with the reconstructed data. Reiter discloses configuring flash system 100 “analogously to hard drives in a redundant array of independent disks (RAID).” EX1004, 8:46-50. RAID allows data lost on one disk to be reconstructed with data from the other redundant disks. *See, e.g.*, EX1204, ¶¶[0007]-[0009]; EX1205,

¶¶[0003]-[0006]; EX1206, 97-99; EX1207, 1:22-50; EX1208, 110-15; EX1209, 164-65, 168, 170.

157. Because the previous block is an unreliable bad block, a POSITA would understand that a new block is needed for the reconstructed data. And a POSITA would be motivated to change data pointers 423 in summary page 420 pointing to the old bad block to the new block with the reconstructed data. This allows the LBA addressing, including logical block number 406 and block index 412, to be unchanged despite the data loss. With the new working block, data at logical block number 406 and block index 412 is trustworthy and usable for later reads (*utilize the remapped correspondence to service one or more of the incoming data access requests*). See EX1004, 17:13-21.

158. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

11. Claim 15, Limitation 15[pre]: *The storage device of claim 1 wherein:*

159. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 discloses this limitation.

(a) Limitation 15[a]: *the flash memory is to have block devices, including the addressed block device; and*

160. A POSITA would understand, or find it obvious, that Reiter's flash system 100's flash media 118 (*i.e., flash memory*) can *have* multiple SCSI logical units, volumes, and/or partitions (*i.e., block devices*). Section VII.A.2.d. ZBC write and erase commands target one of those logical units, volumes, or partitions (*i.e., including the addressed block device*). Section VII.A.2.e.

161. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

(b) Limitation 15[b]: *the storage device is further to:*

162. As addressed in Sections VII.A.11.c-d, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

- (c) **Limitation 15[b][i]:** *store metadata on an independent basis for portions of flash memory associated with each one of the block devices,*

163. A POSITA would understand to implement Reiter's flash controller 104 to *store* a RESET bit (*i.e., metadata*) for each zone of a block device (*i.e., on an independent basis for flash memory associated with each one of the block devices*). A RESET bit is stored for each zone because a ZBC report zones command is optionally set to return the "[l]ist [of] zones for which the RESET bit is set to one." EX1006, 25. A POSITA would understand that flash controller 104 returning the list of zones with RESET bits set to one means storing a RESET bit for each zone.

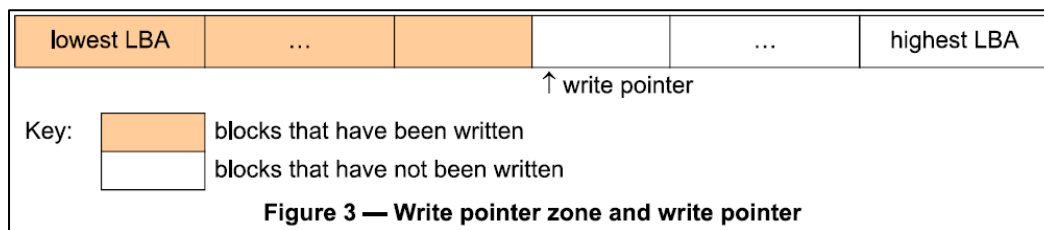
164. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

- (d) **Limitation 15[b][ii]:** *receive a query, issued by a host, directed to a specific one of the block devices, and*

Limitation 15[b][iii]: *responsive to the query, compare the metadata to at least one threshold, to identify specific metadata, and transmit information to the host wherein such transmitted information conveys address information for data stored at one or more physical storage locations associated with the specific metadata.*

165. Reiter's flash controller 104 *receives* a ZBC report zones command (*i.e., query*), that is sent by the host (*i.e., issued by a host device*). ZBC discloses that a ZBC report zones command is optionally set to return the "[l]ist [of] zones for which the RESET bit is set to one." EX1006, 25. Following this report zones command (*i.e., responsive to the query*), each zone's RESET bit (*i.e., identify specific metadata*) is checked if equal to one (*i.e., compare metadata to at least one threshold*), and a list of zones with the RESET bit equal to one sent to the host (*i.e., transmit information to the host device*). See EX1006, 25, Tables 12, 13. The list of zones is sent back via parameter data, which includes a list of zone descriptors. EX1006, 25, Tables 12, 13. The zone descriptors include the zone's write pointer logical block address, which details how much of the zone has user data (*i.e., such transmitted*

information conveys address information for data stored at one of the one or more physical storage locations associated with the specific metadata). See EX1006, Figure 3:



166. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

12. Claim 19-21

167. Claims 19-21 are substantively identical to claim 1 incorporated with claims 9 and 22. Claims 20 and 21 also add that “block devices,” “segments,” and “storage locations” are at “hierarchically-superior,” “-intermediate,” and “-inferior level[s],” respectively, and that block devices and segments contain mutually exclusive sets of segments and logical erase units, respectively. This maps to limitations 1[c][iii]-[iv] which require “*block device[s],*” “*segment[s] within the ... block device[s],*” “*segment[s] correspond[] to a subset of erase units,*” and “*storage location within ... one of the erase units.*” And Ground I discloses logical units, each with its own LBA range, thereby its own zones/superblocks, where each zone/superblock contains its own logical blocks and pages. EX1004, FIGs. 4a-b (depicting a superblock’s summary page 420 containing a list of block indices 422(1)-(X), where block indices 422(1)-(X) each contain a set of page index indices 424(1)-(Y)); EX1006, 13-14, Figures 1-2; EX1008, 14, 61 (“[L]ogical block addresses on a logical unit ... shall begin with block zero and be contiguous up to the last logical block of that logical unit”); Sections VII.A.2.d-g.

| Claim 19 | Claim 20 | Claim 21 | Corresponding Claim 1 Limitation(s) |
|----------|----------|----------|--|
| 19[pre] | 20[pre] | 21[pre] | 1[pre] |
| 19[a] | 20[a] | 21[a] | 1[a] |
| 19[b] | 20[b] | 21[b] | 1[b] (1[c][iii]-[iv] for claims 20-21) |
| 19[c] | 20[c] | 21[c] | 1[c] |
| 19[c][i] | 20[c][i] | 21[c][i] | 1[c][i] |

| Claim 19 | Claim 20 | Claim 21 | Corresponding Claim 1 Limitation(s) |
|------------|------------|------------|-------------------------------------|
| 19[c][ii] | 20[c][ii] | 21[c][ii] | 1[c][ii] |
| 19[c][iii] | 20[c][iii] | 21[c][iii] | 1[c][iii], 9[a], 22[a] |
| 19[c][iv] | 20[c][iv] | 21[c][iv] | 1[c][iv], 9[a], 22[a] |
| 19[d] | 20[d] | 21[d] | 1[d] |
| 19[e] | 20[e] | 21[e] | 1[e] |

168. It is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders these claims obvious.

13. Claim 22, Limitation 22[pre]: *The storage device of claim 1 wherein*

169. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 disclose this limitation.

- (a) **Limitation 22[a]:** *the second address portion comprises a first set of bits and a second set of bits, the bits of the first set of bits being mutually-exclusive to the bits of the second set of bits, wherein the first division operation is to separate the first set of bits from the second set of bits, and the circuitry is to identify the addressed segment from the first set of bits.*

170. My opinion for Sections VII.A.2.f-g renders this limitation obvious.

B. GROUND II: CLAIMS 5, 10-13, AND 18 ARE RENDERED OBVIOUS BY REITER IN LIGHT OF ZBC, SBC-4, SPC-4, AND SINCLAIR-367

1. Motivation To Combine Sinclair-367 With The Combined System Of Reiter And ZBC, SBC-4, And SPC-4

171. A POSITA would have been motivated to combine Reiter’s flash system executing ZBC commands (Section VII.A.1) with Sinclair-367’s bad block management, metadata storage, and planes and metablocks.

172. Sinclair-367 manages bad blocks—identifying “defective cells within the erase block,” and remapping the “logical address” with “the address of any substitute” block, where the “primary erase block is defective.” EX1009, ¶¶[0107]-[0109]. This improves system resiliency. Because Reiter contemplates bad block management, (see Section VII.A.2.h.) a POSITA would have a reasonable expectation of success implementing bad block management in Reiter’s flash device.

173. Further, Sinclair-367 discloses storing in each erase unit a sector of metadata (*e.g.*, error correcting codes (“ECC”) and number of program/erase cycles (*i.e.*, erase count)). EX1009, ¶¶[0106]-[0107], FIG. 3. ECCs recognize and repair damaged data. EX1009, ¶¶[0007], [0102]. Program/erase cycles determine the wear and tear on a physical block. EX1009, ¶¶[0107]-[0108]. Cycle count “is used in a wear-leveling algorithm ... even[ing] out the usage (wear) of all the erase blocks,” increasing the resiliency and lifespan of the flash system. EX1009, ¶¶[0107]-[0108]. Because a single sector is a small overhead for a block, yet reduces data loss, a POSITA would have a reasonable expectation of success storing a metadata sector in physical blocks.

174. Further, Sinclair-367 discloses dividing flash memory into planes, and forming metablocks by linking erase units in each plane. EX1009, ¶¶[0009]-[0012], [0111]-[0112]. This improve speed and throughput. Each plane operates independently—concurrently executing reading/writing. EX1009, ¶[0111]. And “to increase ... parallelism[,] erase blocks ... are logically linked to form metablocks.” EX1009, ¶[0114]. Because Sinclair-367 and Reiter disclose analogous flash systems, and planes and metablocks are common in the flash memory art, a POSITA would have a reasonable likelihood of success implementing them in Ban’s flash device. Moreover, Reiter, ZBC, SBC-4, SPC-4, and Sinclair-367 disclose analogous art. All relate to storage devices, including accessing blocks of memory.

2. Claim 5, Limitation 5[pre]: *The storage device of claim 1 wherein*

175. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 discloses this limitation.

- (a) **Limitation 5[a]: *the storage device is to store information identifying a number of structures in the flash memory and is to transmit the information identifying the number of structures to a host, and wherein the storage device is to store the values representing a time since data was programmed on a basis that is respective to each structure of the number of structures, wherein each structure comprises at least one of a page or a unit of memory***

comprising one or more erase units of the flash memory, and wherein each value represents time since data was programmed into the respective structure.

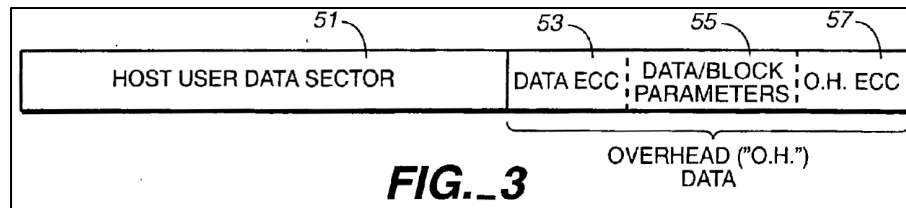
176. A POSITA would be motivated to implement Reiter’s flash system 100 (*i.e.*, *storage device*) to accept and execute ZBC commands. Section VII.A.2.b. The ZBC command set includes a report zones command. The report zones command optionally “return[s]” (*i.e.*, *transmit*) a “[l]ist [of] zones in the zoned block device” back *to [the] host* via “parameter data” that includes the length of the zone list and zone descriptors indicating the starting logical block address and length of each zone (*i.e.*, *the information identifying the number of structures to a host*). EX1006, 25, Table 11, Tables 12, 13:

| Table 12 — REPORT ZONES parameter data | | | | | | | | |
|--|-------------------------|---|---|---|---|---|---|------|
| Bit
Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | (MSB) | | | | | | | |
| ... | ZONE LIST LENGTH (n–63) | | | | | | | |
| 3 | (LSB) | | | | | | | |
| 4 | Reserved | | | | | | | SAME |
| 5 | Reserved | | | | | | | |
| ... | | | | | | | | |
| 63 | | | | | | | | |
| Zone descriptors list | | | | | | | | |
| 64 | Zone descriptor [first] | | | | | | | |
| ... | | | | | | | | |
| 127 | | | | | | | | |
| | ⋮ | | | | | | | |
| n–63 | Zone descriptor [last] | | | | | | | |
| ... | | | | | | | | |
| n | | | | | | | | |

| Table 13 — Zone descriptor format | | | | | | | | |
|-----------------------------------|-------------------|---|---|---|-----------|---|---|-------|
| Bit
Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | Reserved | | | | ZONE TYPE | | | |
| 1 | ZONE CONDITION | | | | Reserved | | | RESET |
| 2 | Reserved | | | | | | | |
| ... | | | | | | | | |
| 7 | | | | | | | | |
| 8 | (MSB) | | | | | | | |
| ... | ZONE LENGTH | | | | | | | |
| 15 | | | | | | | | |
| 16 | (MSB) | | | | | | | |
| ... | ZONE START LBA | | | | | | | |
| 23 | | | | | | | | |
| 24 | (MSB) | | | | | | | |
| ... | WRITE POINTER LBA | | | | | | | |
| 31 | | | | | | | | |
| 32 | Reserved | | | | | | | |
| ... | | | | | | | | |
| 63 | | | | | | | | |

177. Because flash system 100 transmits the zone list information back to the host, a POSITA would understand that flash system 100 also stores that information (*i.e., store information identifying a number of structures in the flash memory*). Or a POSITA would be motivated to store such small amount of information, faster to simply retrieve from storage rather than waste time recalculating it with each request. As opposed to wasting time re-calculating such information with every command requiring zone metadata, flash system could store zone metadata and update it as zone settings are changed.

178. Sinclair-367's erase blocks use a "single sector of data" to *store* overhead data, including error correction codes (ECC) 53 for data and data/block parameters 55 (*i.e., metadata*). EX1009, FIG. 3:



179. Therefore, every block in a logical unit, volume, and partition includes this single sector of overhead data (*i.e., independent basis for flash memory associated with each one of the block devices*). Sinclair-367 also includes the “time of programming ... as a field in [the] sector or page overhead data” (*i.e., metadata representing a time since one or more physical storage locations associated with the one or more block devices has been a subject of a write operation*). EX1009, ¶[0111].

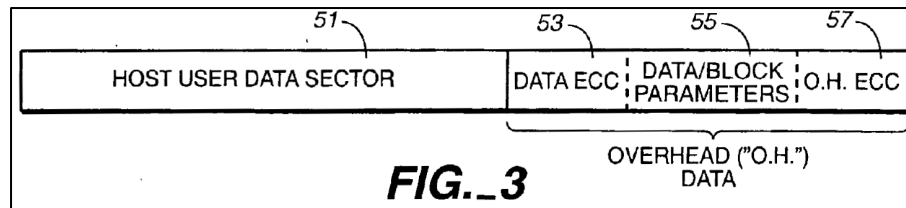
180. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, and Sinclair-367 renders this limitation obvious.

3. Claim 10, Limitation 10[pre]: *The storage device of claim 1 wherein:*

181. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 disclose this limitation.

(a) Limitation 10[a]: *the storage device is to store, for each segment having an erase unit which stores valid data, a value representing a time since data was programmed;*

182. A POSITA would be motivated to incorporate Reiter’s flash system 100 (*i.e., storage device*) to *store* physical blocks with Sinclair-367’s metadata sector (*i.e., for each segment having an erase unit which stores valid data*). Sinclair-367’s erase blocks use a “single sector of data” to *store* overhead data, including error correction codes (ECC) 53 for data and data/block parameters 55 (*i.e., metadata*). EX1009, FIG. 3:



183. Therefore, every block in a logical unit, volume, and partition includes this single sector of overhead data (*i.e., independent basis for flash memory associated with each one of the block devices*). Sinclair-367 also includes the “time of programming ... as a field in [the] sector or page overhead data” (*i.e., metadata representing a time since one or more physical storage locations associated with the one or more block devices has been a subject of a write operation*). EX1009, ¶[0107].

184. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, and Sinclair-367 renders this limitation obvious.

(b) **Limitation 10[b]:** *the storage device is to compare the values representing time since data was programmed to at least one threshold;*

185. A POSITA would be motivated to implement flash controller 104 of Reiter’s flash memory storage 104 (*i.e., storage device*) to *compare* the time of programming data in a physical block (*i.e., values representing time since data was programmed*) against a retention period (*i.e., at least one threshold*). A retention period is the period of time where flash memory reliably stores a value, and flash memory does not reliably store the value after that period of time. This is a problem well known in the art. Checking a time-since-programming alerts flash controller 104 to physical blocks becoming unreliable, allowing flash controller 14 to remedy the situation and reduce data loss.

186. Sinclair-367 recognizes that flash memory has a retention period problem. For instance, Sinclair-367 discloses how a memory cell relies on “threshold voltage window ... divided into a number of ranges” to create storage states. EX1009, ¶[0007]. But maintaining the voltage level within

one of the slim ranges associated with certain storage states is difficult. “[S]torage levels do shift” shift over usage of the memory cell and surrounding memory cells. EX1009, ¶¶[0007]. One way to predict and stop an intolerable shift in voltage levels is to check if the time since programming is nearing the average retention period of a memory cell. This can alert flash controller 104 of potential data loss, and provide an opportunity to save the data.

187. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, and Sinclair-367 renders this limitation obvious.

- (c) **Limitation 10[c]:** *upon detection that a specific one of the values has satisfied the one or more thresholds, the circuitry is to copy data stored at one or more physical storage locations associated with the specific one of the values from a first physical erase unit to a second physical erase unit; and*

188. Upon nearing or reaching the average retention period (*i.e., upon detection that a specific one of the values has satisfied one or more thresholds*), a POSITA would be motivated to copy data out of the unreliable block past its retention period (*i.e., copy data stored at one of the one or more physical storage locations associated with the specific one of the values*) and into a working physical block (*i.e., from a first physical erase unit to a second physical erase unit*). The data is saved by copying data away to reset the voltage levels. And copying data away from a bad block to a working block is a common, simple solution. Copying data into a new block, such as through garbage collection is faster than reading, erasing, and rewriting the same block. *See also* EX1009, ¶¶[0108]-[0109], [0114], [0132], [0177].

189. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, and Sinclair-367 renders this limitation obvious.

- (d) **Limitation 10[d]:** *the storage device is to remap a logical address for the data being copied from the first physical erase unit to the second physical erase unit.*

190. Sinclair-367 discloses that copying data away from a failing block to a new block

also includes remapping the logical address from the old physical address to the new physical address (*i.e., remap a logical address for the data being copied from the first physical erase unit to the second physical erase unit*). For example, the “logical address of the data ... mapped into th[e] ... address of any substitute erase block” if “the primary erase block is defective.” EX1009, ¶[0109].

191. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, and Sinclair-367 renders this limitation obvious.

4. Claim 11, Limitation 11[pre]: *The storage device of claim 10 wherein:*

192. As addressed in Section VII.B.3, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 disclose this limitation.

(a) Limitation 11[a]: *the flash memory is to have block devices, one of the block devices being the addressed block device; and*

193. Given the similarity in claim language, my opinion for Section VII.A.11.a applies to this limitation.

(b) Limitation 11[b]: *the storage device is to identify, for the data being copied, an address of a logical erase unit, the logical erase unit being associated with the addressed block device to the exclusion of each other one of the block devices; and*

Limitation 11[c]: *the storage device is to remap a correspondence between the address of the logical erase unit and the first physical erase unit to instead be between the address of the logical erase unit and the second physical erase unit.*

194. Sinclair-367 discloses that copying data away from a failing block to a new block also includes remapping the logical address from the old physical address to the new physical address (*i.e., remap a logical address for the data being copied from the first physical erase unit to the second physical erase unit*). For example, the “logical address of the data ... mapped into th[e] ... address of any substitute erase block” if “the primary erase block is defective.” EX1009, ¶[0109].

195. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, and Sinclair-367 renders this limitation obvious.

5. **Claim 12, Limitation 12[pre]:** *The storage device of claim 1 wherein:*

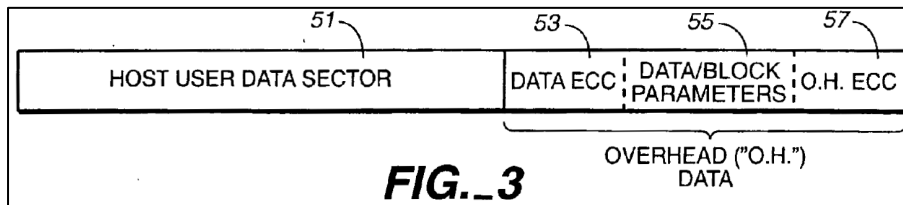
196. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 discloses this limitation.

(a) **Limitation 12[a]:** *the flash memory is to have block devices, including the addressed block device; and*

197. Given the similarity in claim language, my opinion for Section VII.A.11.a applies to this limitation.

(b) **Limitation 12[b]:** *the storage device is to store metadata on an independent basis for portions of flash memory associated with each one of the block devices, wherein the stored metadata is dependent on an erase count of associated physical memory space;*

198. Sinclair-367's erase blocks use a "single sector of data" to *store* overhead data, including error correction codes (ECC) 53 for data and data/block parameters 55 (*i.e., metadata*). EX1009, ¶[0107], FIG. 3:



199. Therefore, every block in a logical unit, volume, and partition includes this single sector of overhead data (*i.e., independent basis for flash memory associated with each one of the block devices*). Sinclair-367 also discloses that "parameter 55 may include a quantity related to the number of program/erase cycles experiences by the erase block" (*i.e., wherein the stored metadata is dependent on an erase count of associated physical memory space*). EX1009, ¶[0108].

200. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, and Sinclair-367 renders this limitation obvious.

- (c) **Limitation 12[c]:** *the storage device is to compare the metadata to at least one threshold;*

Limitation 12[d]: *upon detection that specific metadata has satisfied the one or more thresholds, the circuitry is to copy data stored at one or more physical storage locations associated with the specific metadata from a first physical erase unit to a second physical erase unit; and*

Limitation 12[e]: *the storage device is to remap a logical erase unit value associated with the third address portion from the first physical erase unit to the second physical erase unit.*

201. Sinclair-367 discloses that the “number of program/erase cycles” is “used in a wear leveling algorithm,” where “logical block addresses are regularly re-mapped to a different physical block addresses in order to even out the usage.” EX1009, ¶[0108].

202. A POSITA would understand that the remapping hinges on the number of program/erase cycles (*i.e.*, **metadata**) being high enough to indicate a heavily used erase block (*i.e.*, **at least one threshold**).

203. Moreover, detection that an erase block is heavily used (*i.e.*, **upon detection that specific metadata has satisfied the one or more thresholds**) causes logical block addresses to be remapped. A POSITA would understand that remapping a logical address to a different physical block means the data of the old block is copied to the new block stop data loss (*i.e.*, **copy data stored at one of the one or more physical storage locations associated with the specific metadata from a first one of the physical erase units for which the failure condition has not been detected by said storage device to a second one of the physical erase units for which the failure condition has not been detected by said storage device**). The certain threshold indicates a heavily used erase block.

204. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, and Sinclair-367 renders this limitation obvious.

6. Claim 13, Limitation 13[pre]: *The storage device of claim 1 wherein:*

205. As addressed in Section VII.A.2, it is my opinion that the combined system of

Reiter and ZBC, SBC-4, and SPC-4 discloses this limitation.

- (a) **Limitation 13[a]:** *the erase units of the flash memory are physical erase units;*

Limitation 13[b]: *the circuitry is to use the second division operation to identify a logical erase unit, wherein each logical erase unit is mapped to one of the physical erase units; and*

206. As addressed in Section VII.A.10.a-b, it is my opinion that the combined version of Reiter and ZBC, SBC-4, and SPC-4 discloses this limitation.

- (b) **Limitation 13[c]:** *the storage device is to detect a failure condition of the physical erase units and is to remap each logical erase unit which corresponds to one of the physical erase units for which the failure condition is detected to instead correspond to one of the physical erase units for which the failure condition has not been detected and is to utilize the remapped correspondence to service one or more of the incoming data access requests.*

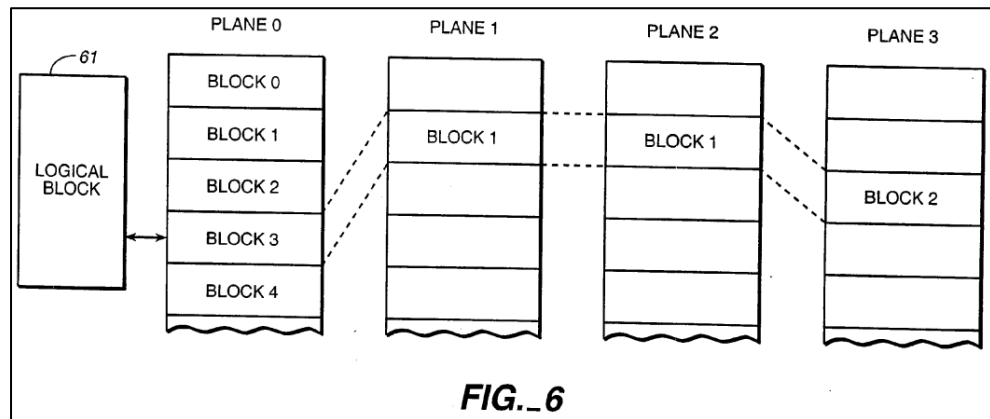
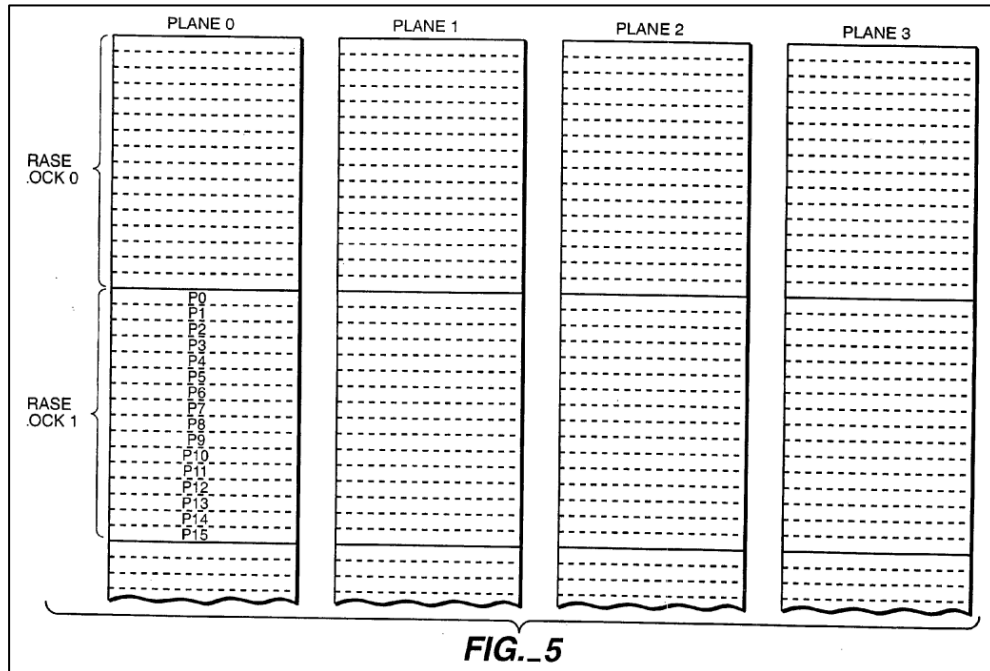
207. A POSITA would be motivated to modify Reiter's flash system 100 (*storage device*) to implement Sinclair-367's bad block management. Section VII.B.1. Sinclair-367 discloses "identification of any defective cells within the erase block," (*detect a failure condition of the physical erase units*) and *remapping* the "logical address" (*logical erase unit which corresponds to one of the physical erase units for which the failure condition is detected*) with the "address of any substitute erase block in case the primary erase block is defective" (*instead correspond to one of the physical erase units for which the failure condition has not been detected*) for later use (*utilize the remapped correspondence to service one or more of the incoming data access requests*). EX1009, ¶¶[0107]-[0109]. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, and Sinclair-367 renders this limitation obvious.

7. **Claim 18, Limitation 18[pre]:** *The storage device of claim 1 wherein:*

208. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 disclose this limitation.

(a) **Limitation 18[a]:** *the flash memory comprises multiple planes; and*

209. A POSITA would be motivated to implement Reiter's *flash* media 118 as "divided into two or more planes" (*i.e.*, **multiple planes**) per Sinclair-367's teachings. EX1009, ¶[0112], FIGs. 5-6 (below); Section VII.B.1.



210. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, and Sinclair-367 renders this limitation obvious.

(b) **Limitation 18[b]:** *the circuitry is to, for each one of a set of the data write requests having sequential addressing, identify a*

commonly addressed page and concurrently write data accompanying the set of write requests into respective ones of the multiple planes in a manner corresponding to the commonly addressed page.

211. A ZBC write command includes a “logical block address” and “transfer length.”

EX1007, Table 112.

| Table 112 — WRITE (32) command | | | | | | | | | |
|--------------------------------|-------|--|---|---|--------------|-----|----------|----------|----------|
| Byte | Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | | OPERATION CODE (7Fh) | | | | | | | |
| 1 | | CONTROL | | | | | | | |
| 2 | | Reserved | | | | | | | |
| ... | | | | | | | | | |
| 5 | | | | | | | | | |
| 6 | | Reserved | | | GROUP NUMBER | | | | |
| 7 | | ADDITIONAL CDB LENGTH (18h) | | | | | | | |
| 8 | (MSB) | SERVICE ACTION (000Bh) | | | | | | | |
| 9 | | | | | | | | | (LSB) |
| 10 | | WRPROTECT | | | DPO | FUA | Reserved | Obsolete | Reserved |
| 11 | | Reserved | | | | | | | |
| 12 | (MSB) | LOGICAL BLOCK ADDRESS | | | | | | | |
| ... | | | | | | | | | |
| 19 | | | | | | | | | (LSB) |
| 20 | (MSB) | EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG | | | | | | | |
| ... | | | | | | | | | |
| 23 | | | | | | | | | (LSB) |
| 24 | (MSB) | EXPECTED LOGICAL BLOCK APPLICATION TAG | | | | | | | |
| 25 | | | | | | | | | (LSB) |
| 26 | (MSB) | LOGICAL BLOCK APPLICATION TAG MASK | | | | | | | |
| 27 | | | | | | | | | (LSB) |
| 28 | (MSB) | TRANSFER LENGTH | | | | | | | |
| ... | | | | | | | | | |
| 31 | | | | | | | | | (LSB) |

212. “TRANSFER LENGTH ... specifies the ... logical blocks ... that shall be transferred ... starting [from] ... the LOGICAL BLOCK ADDRESS” EX1007, 198. With a large transfer length, a ZBC write command causes several write operations across *sequential addresses*.

213. To increase throughput for these large write commands, a POSITA would be motivated to implement Reiter’s *flash* media 118 with multiple *planes*, and metablocks spanning the multiple planes, as disclosed in Sinclair-367. EX1009, ¶¶[0111]-[0114]; Sections VII.B.4.a, VII.B.1.

“All blocks of the metablock are erased together, and pages of each block are generally programmed and read simultaneously” (*i.e., identify a commonly addressed page and concurrently write data accompanying the set of incoming data access requests into respective ones of the multiple planes in a manner corresponding to the commonly addressed page*). EX1009, ¶[0114].

214. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, and Sinclair-367 renders this limitation obvious.

C. GROUND III: CLAIMS 14 AND 16-17 ARE RENDERED OBVIOUS BY REITER IN LIGHT OF ZBC, SBC-4, SPC-4, SINCLAIR-367, AND YAMADA

1. Motivation To Combine Yamada With The Combined System Of Reiter, ZBC, SBC-4, SPC-4, And Sinclair-367

215. A POSITA would have been motivated to implement Reiter’s flash system to execute ZBC commands (Sections VII.A.1, VII.B.1) and with Yamada’s maintenance request and determination units in Reiter’s flash system and host processor, respectively. Reiter complains that “[g]arbage collection routines impact performance of the flash memory system by utilizing processor resources and potentially delaying write operations ...” EX1004, 4:23-26. In Yamada, “maintenance is performed without being ... performed together with normal system processing,” but “with sufficiently long processing time.” EX1010, Abstract. The maintenance request unit on the flash device determines and requests maintenance operations from the maintenance determination unit. EX1010, ¶[0032], [0081]-[0082]. The maintenance determination unit monitors status of the host, “determin[es] that the access device is not engaged in normal processing,” and performs maintenance. EX1010, Abstract, ¶¶[0018]-[0019], [0022]. Because maintenance scheme is implemented on a nonvolatile system analogous like that in Reiter, a POSITA would have a reasonable expectation of success implementing Reiter’s flash system with this scheme.

216. Moreover, Reiter, ZBC, SBC-4, SPC-4, Sinclair-367, and Yamada disclose

analogous art. All relate to storage devices, including the management and control of storage devices.

2. Claim 14, Limitation 14[pre]: *The storage device of claim 1 wherein the storage device is to:*

217. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 discloses this limitation.

(a) Limitation 14[a]: *store, for each segment having an erase unit which stores valid data, a value representing a time since data was programmed;*

218. Given the similarity in claim language, my opinion for Section VII.B.3.a applies to this limitation.

(b) Limitation 14[b]: *compare the values representing time since data was programmed to at least one threshold; and*

219. Given the similarity in claim language, my opinion for Section VII.B.3.b applies to this limitation.

(c) Limitation 14[c]: *responsive to detection of a condition that a specific one of the values representing time since data was programmed has satisfied the one or more thresholds, transmit information to a host wherein such transmitted information conveys address information for data stored at one or more physical storage locations associated with the specific one of the values.*

220. Yamada discloses maintenance determination unit 152 in the host that receives a “logical address included in [a] maintenance request transmitted from ... maintenance request unit 115” on the flash storage device (*i.e., transmit information to a host device wherein such transmitted information conveys address information*). EX1010, ¶[0101]. A POSITA would be motivated to incorporate Yamada’s maintenance determination unit and maintenance request unit scheme because Yamada’s implementation reduces the delay caused by garbage collection procedures. Use of Yamada’s maintenance modules for garbage collection is appropriate as both maintenance and garbage collection are automatic, background tasks for rewriting or otherwise managing data. See EX1010, Abstract,

¶¶[0007], [0036]. Reiter makes this complaint expressly, stating that “[g]arbage collection routines impact performance of the flash memory system by utilizing processor resources and potentially delaying write operations to the flash media.” EX1004, 4:23-26. The maintenance determination unit, by being on the host can determine the “system status” of the host, and is able to issue “data maintenance ... without being required to be performed together with normal system processing.” EX1010, Abstract, ¶[0036].

221. Given the use of Yamada’s maintenance determination and request units, a POSITA would be motivated to trigger a maintenance request upon detecting a physical block time since programming nears or reaches the average retention period (*i.e.*, ***responsive to detection that specific metadata has satisfied the one or more thresholds***). As indicated by the name of Yamada’s units, maintenance requests should be triggered for garbage collections, including operations reduce the risk of lost data. As stated with Section IX.C.1.d, after the average retention period, a memory cell can be unreliable, risking loss of the data stored in the cell. Saving that data from loss is done through garbage collection or maintenance operations, such as those issued from a maintenance request.

222. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, Sinclair-367, and Yamada renders this limitation obvious.

3. Claim 16, Limitation 16[pre]: *The storage device of claim 1 wherein:*

223. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 discloses this limitation.

(a) Limitation 16[a]: *the flash memory is to have block devices, including the addressed block device; and*

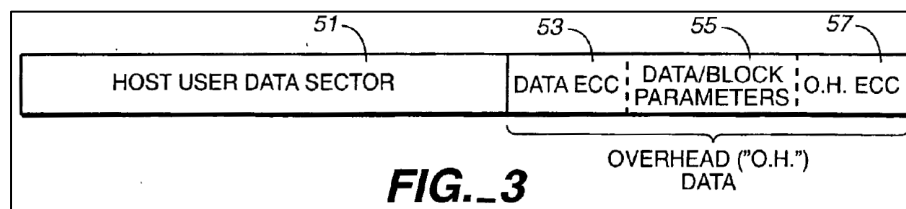
224. Given the similarity in claim language, my opinion for Section VII.A.11.a applies to this limitation.

(b) **Limitation 16[b]: *the storage device is further to:***

225. As addressed in Sections VII.C.3.c-d, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 renders this limitation obvious.

(c) **Limitation 16[b][i]: *store metadata on an independent basis for portions of flash memory associated with each one of the block devices, wherein the stored metadata represents an erase count of associated physical memory space, and***

226. With the writing capabilities of Reiter's flash controller 104 (*i.e.*, **circuitry**), Sinclair-367's erase blocks use a "single sector of data" to **store** overhead data, including error correction codes (ECC) 53 for data and data/block parameters 55 (*i.e.*, **metadata**). EX1009, ¶¶[0107]-[0110], FIG. 3:



227. Every block in a logical unit, volume, and partition includes this single sector of overhead data (*i.e.*, **independent basis for flash memory associated with each one of the block devices**). EX1009, ¶¶[0107]-[0110]. Sinclair-367 also discloses that "parameter 55 may include a quantity related to the number of program/erase cycles experiences by the erase block" (*i.e.*, **wherein the stored metadata is dependent on an erase count of associated physical memory space**). EX1009, ¶¶[0107]-[0110].

228. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, Sinclair-367, and Yamada renders this limitation obvious.

(d) **Limitation 16[b][ii]: *receive a query, issued by a host, wherein said query is specific to one of the block devices, and responsive to the query, identify specific metadata, associated with the specified one of the block devices, and transmit information to the host that conveys the specific metadata.***

229. Yamada discloses maintenance determination unit 152 in the host that receives a

“logical address included in [a] maintenance request transmitted from ... maintenance request unit 115” on the flash storage device. EX1010, ¶[0101]. A POSITA would be motivated to incorporate Yamada’s maintenance determination unit and maintenance request unit scheme because Yamada’s implementation reduces the delay caused by garbage collection procedures. Use of Yamada’s maintenance modules for garbage collection is appropriate as both maintenance and garbage collection are automatic, background tasks for rewriting or otherwise managing data. See EX1010, Abstract, ¶¶[0007], [0036]. Reiter makes this complaint expressly, stating that “[g]arbage collection routines impact performance of the flash memory system by utilizing processor resources and potentially delaying write operations to the flash media.” EX1004, 4:23-26. The maintenance determination unit, by being on the host, can determine the “system status” of the host, and is able to issue “data maintenance ... without being required to be performed together with normal system processing.” EX1010, Abstract.

230. Given the use of Yamada’s maintenance determination and request units, a POSITA would find a query for specific metadata obvious in Yamada. Yamada’s “non-volatile memory 1020” (e.g., Reiter’s flash media on flash system 100) “stores management information including the information about the importance level of data to enable the management information to be shared between different access modules” (i.e., hosts). EX1010, ¶[0190]. A POSITA would understand that sharing such metadata requires a *query, issued by a host device, wherein said query is specific to one of the block devices*, where the query causes (i.e., *responsive to the query*) the storage device to *identify specific metadata, associated with the specified one of the block devices, and transmit information to the host device that conveys the specific metadata*.

231. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, Sinclair-367, and Yamada renders this limitation obvious.

4. **Claim 17, Limitation 17[pre]:** *The storage device of claim 1 wherein the storage device is further to:*

232. As addressed in Section VII.A.2, it is my opinion that the combined system of Reiter and ZBC, SBC-4, and SPC-4 discloses this limitation.

(a) **Limitation 17[a]:** *store metadata on an independent basis for portions of flash memory associated with each one of the block devices, wherein the stored metadata represents an erase count of associated physical memory space;*

233. Given the similarity in claim language, my opinion for Section VII.C.3.c applies to this limitation.

(b) **Limitation 17[b]:** *compare the metadata to at least one threshold; and*

Limitation 17[c]: *responsive to detection that specific metadata has satisfied the one or more thresholds, transmit information to a host wherein such transmitted information conveys address information for data stored at one or more physical storage locations associated with the specific metadata.*

234. Yamada discloses maintenance determination unit 152 in the host that receives a “logical address included in [a] maintenance request transmitted from ... maintenance request unit 115” on the flash storage device (*i.e.*, *transmit information to a host wherein such transmitted information*). EX1010, ¶[0101]. A POSITA would be motivated to incorporate Yamada’s maintenance determination unit and maintenance request unit scheme because Yamada’s implementation reduces the delay caused by garbage collection procedures. Use of Yamada’s maintenance modules for garbage collection is appropriate as both maintenance and garbage collection are automatic, background tasks for rewriting or otherwise managing data. *See* EX1010, Abstract, ¶¶[0007], [0036]. Reiter makes this complaint expressly, stating that “[g]arbage collection routines impact performance of the flash memory system by utilizing processor resources and potentially delaying write operations to the flash media.” EX1004, 4:23-26. The maintenance determination unit, by being on the host can determine the “system status” of the host, and is able to issue “data maintenance ... without being required to be performed together with

normal system processing.” EX1010, Abstract, ¶[0036].

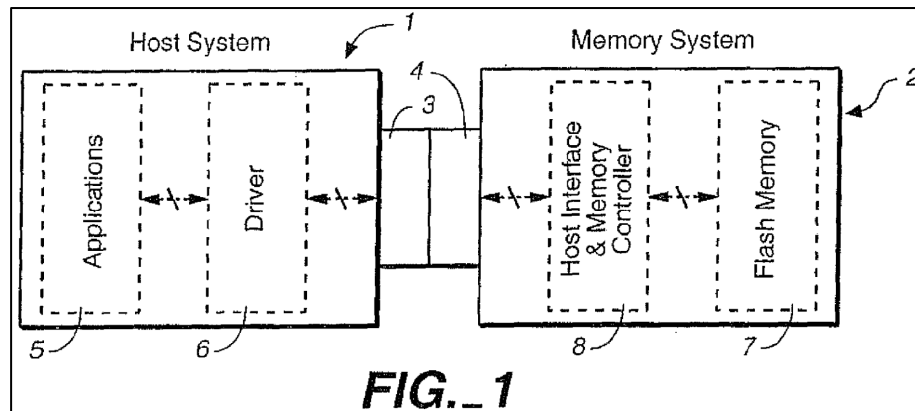
235. Given the use of Yamada’s maintenance determination and request units, a POSITA would be motivated to trigger a maintenance request upon detecting a high number of program/erase cycles (*i.e.*, ***responsive to detection that specific metadata has satisfied the one or more thresholds***). As indicated by the name of Yamada’s units, maintenance requests should be triggered for garbage collections, including wear-level management for improving data retention. As stated with Section IX.C.2.d, program/erase counts for physical blocks are “used in a wear leveling algorithm ... in order to even out the usage (wear) of all erase blocks.” EX1009, ¶[0108]. Excessive wear causes erase blocks to become unreliable, risking loss of data stored in the erase block. Spreading out wear to reduce data loss is done through garbage collection or maintenance operations, such as those issued from a maintenance request.

236. It is my opinion that the combined system of Reiter, ZBC, SBC-4, SPC-4, Sinclair-367, and Yamada renders this limitation obvious.

D. GROUND IV: CLAIMS 1-2, 8-9, 18-22 ARE RENDERED OBVIOUS BY SINCLAIR-233

1. Claim 1, Limitation 1[pre]: *A storage device comprising:*

237. Sinclair-233 discloses a “memory system 2” (*i.e.*, ***storage device***) that operates on a “file based interface between the host and memory system[]” EX1005, Abstract, 5:7-9, 5:26-28, 12:63-67, FIG. 1:



238. It is my opinion that Sinclair-233 renders this limitation obvious.

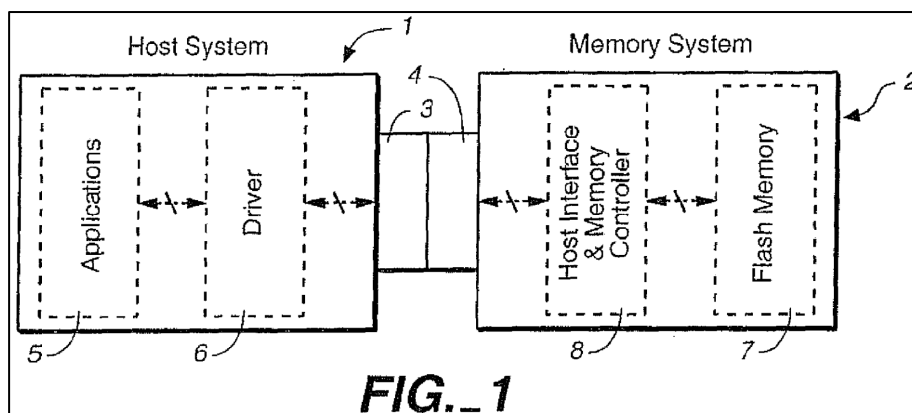
(a) **Limitation 1[a]: *flash memory comprising erase units; and***

239. Sinclair-233 discloses flash memory 7 (*i.e., flash memory*) made of memory cell groups called “*erase units*” where “such group[s] are the minimum number of memory cells that are erasable together.” EX1005, 1:30-34, 6:58-62.

240. It is my opinion that Sinclair-233 renders this limitation obvious.

(b) **Limitation 1[b]: *circuitry to receive from a host, and control execution of, incoming requests including data write requests and erase requests, wherein each of the incoming requests is accompanied by incoming address information;***

241. Sinclair-233 discloses memory system 2 including a “Host Interface & Memory Controller 8” (*i.e., circuitry to receive from a host*) for “convert[ing] between logical addresses of data used ... [for] data programming and reading” (*i.e., and control execution of, incoming requests*). EX1005, 5:19-25, FIG. 1.



242. “A set of direct file interface commands from the host system supports the operation of the memory system.” EX1005, 13:36-45. The direct file interface commands (*i.e., incoming requests*) **include** an insert command and erase command (*i.e., data write requests and erase requests*). EX1005, FIG. 12A-B:

| DATA COMMANDS | | |
|---------------|---|--|
| Command | Parameters | Description |
| Write | < fileID > | Appends data to end of the specified file. |
| Insert | < fileID > < offset > | Inserts data within specified file at < offset > from start. |
| Update | < fileID > < offset > | Overwrites data in specified file at < offset > from start. |
| Read | < fileID > < offset > | Reads data from specified file at < offset > from start. |
| Remove | < fileID > < offset 1 >
< offset 2 > | Removes data within specified file between < offset 1 > and < offset 2 > from start. |

FIG._12A

| FILE COMMANDS | | |
|---------------|-----------------------|--|
| Command | Parameters | Description |
| Open | < fileID > | Requests resources to be made available for writing to the specified file. |
| Close | < fileID > | Resources may be removed immediately from the specified file, and the file may be scheduled for garbage collection. |
| Close_after | < fileID > < length > | Resources may be removed from the specified file after further transfer of < length> of file data from host, at which time the file may be scheduled for garbage collection. |
| Delete | < fileID > | Indicates that table entries and data for the specified file should be deleted. File data may be erased. |
| Erase | < fileID > | Indicates that table entries and data for the specified file should be deleted and that file data should be erased immediately. |

FIG._12B

243. An “[i]nset ... command is followed by transmission of data from the host to the

memory system, and the memory system responds by writing the data in its memory array.” EX1005, 13:45-48. And “[a]n [e]rase command specifies that data of the specified file be immediately erased from the memory.” EX1005, 14:19-21.

244. The insert and erase commands include a “fileID” and/or “offset” parameter (*i.e., incoming requests is accompanied by incoming address information*), “designat[ing] [a] file (<fileID>) at a particular offset (<offset>) within the file [to be] written ... or read.” EX1005, 13:36-45.

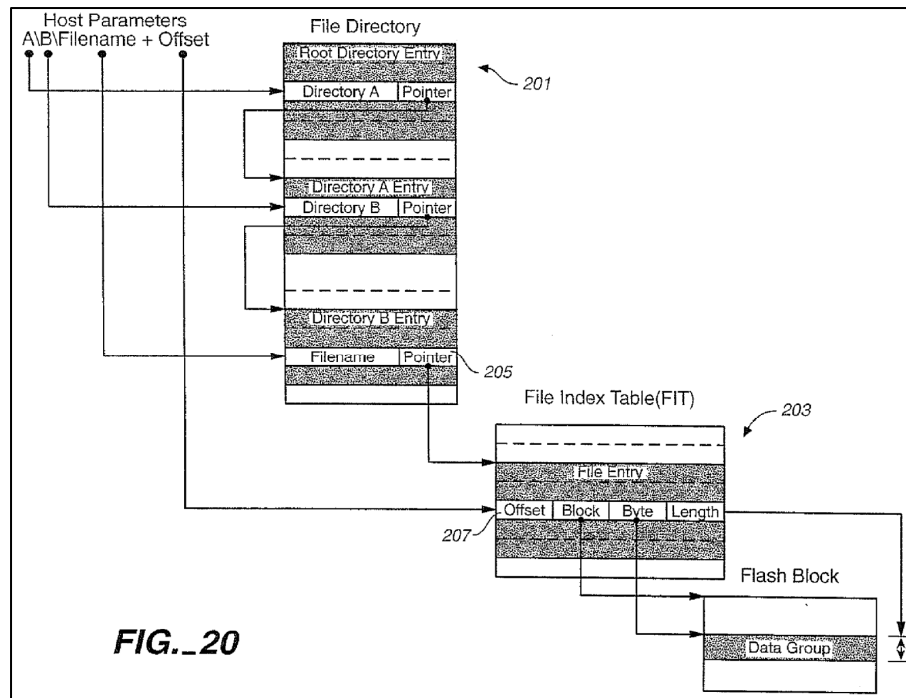
245. It is my opinion that Sinclair-233 renders this limitation obvious.

(c) **Limitation 1[c]:** *wherein said circuitry is to, for each of the incoming requests:*

246. As addressed in Sections VII.D.1.d-g, it is my opinion that Sinclair-233 renders this limitation obvious.

(d) **Limitation 1[c][i]:** *derive, from the accompanying incoming address information, a first address portion and a second address portion*

247. Sinclair-233 discloses insert and erase commands at least including a fileID. EX1005, FIGs. 12A-B. “The <fileID> parameter can be ... a full pathname for the file parameter.” EX1005, 14:47-61. Sinclair-233 discloses the pathname including (1) a storage ID and (2) a directory pathname for the file (*i.e., derive, from the accompanying incoming address information, a first address portion and a second address portion*). EX1005, 14:47-61, 34:50-58. “The pathname that is used for files at the Direct-File Interface may identify the ID of the storage device itself” EX1005, 34:50-58. “The pathname [also] describes a hierarchy of directories and a file within the lowest level of directory.” EX1005 at 14:47-61. An example of the pathname detailing the directories and filename is provided in FIG. 20:



248. It is my opinion that Sinclair-233 renders this limitation obvious.

(e) **Limitation 1[c][ii]:** *identify an addressed block device from the first address portion,*

249. The storage ID constitutes the *first address portion*. Section VII.D.1.d. The storage ID *identifies* “the storage device itself” (*i.e., addressed block device from the first address portion*). EX1005, 34:50-58.

250. It is my opinion that Sinclair-233 renders this limitation obvious.

(f) **Limitation 1[c][iii]:** *use a first division operation on the second address portion, to identify an addressed segment within the addressed block device, wherein the addressed segment corresponds to a subset of the erase units of the flash memory, and*

251. The directory-file pathname constitutes the *second address portion*. Section VII.D.1.d. “The pathname describes a hierarchy of directories and a file within the lowest level of directory.” EX1005, 14:47-61. Sinclair-233 discloses that the “[p]ath segments may be delimited by ‘\’” (*i.e., use a first division operation on the second address portion*). EX1003, ¶¶251-52; EX1005, 14:56-59; *see also* EX1213, 56:25-7. This separates and extracts the directory names and the filename in order

for traversing a file directory 201 as described in FIG. 20:

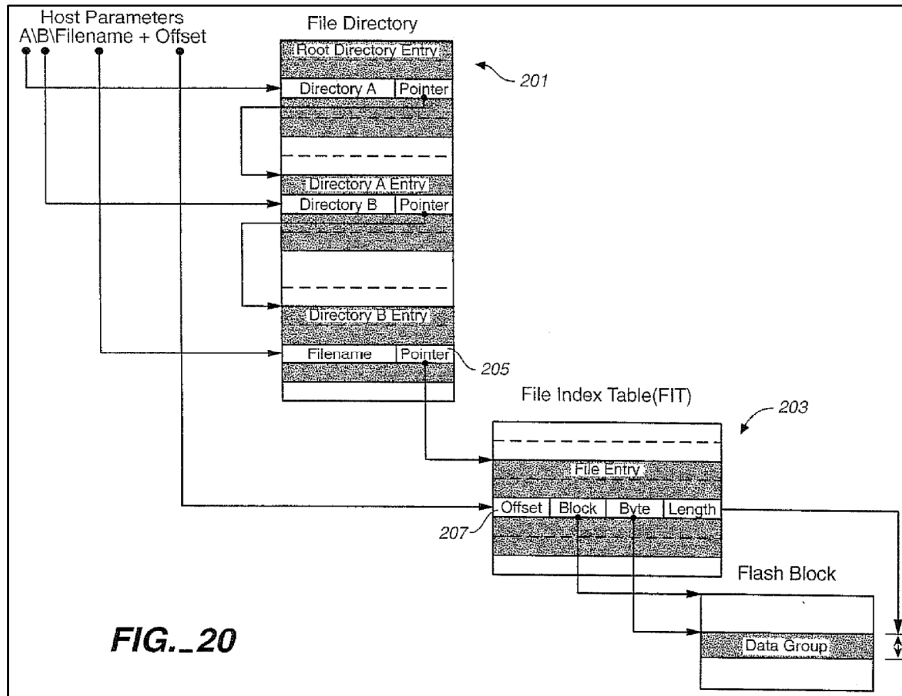


FIG. 20

252. Separating out the filename allows use of the filename for searching the corresponding directory entry to identify the file to be accessed (*i.e., an addressed segment within the addressed block device*). EX1005, 30:40-47.

253. A file *corresponds to a subset of the erase units of the flash memory*. “Host system data files are written directly to large erase block flash memory system.” EX1005, Abstract. The “file index table (FIT)” lists the set of blocks associated with a file. *See* EX1005, FIG. 19. For instance, “FIG. 19 illustrates a sequence of index entries in a file index table (FIT) for one file.” EX1005, 29:35-38, FIG. 19:

FIG. 19

| File Index Table (FIT) | | | | |
|------------------------|--------|-------|------|--------|
| | OFFSET | BLOCK | BYTE | LENGTH |
| Time 0
(Fig. 14A) | F0 | 001 | D0 | — |
| | F1 | 002 | D1 | — |
| Time 2
(Fig. 14C) | F0 | 001 | D0 | — |
| | F5 | 003 | D6 | — |
| | F6 | 004 | D7 | — |
| | F7 | 001 | D5 | — |
| | F8 | 002 | D3 | — |
| | F9 | 003 | D4 | — |
| | F10 | 002 | D2 | — |
| Time 4
(Fig. 14E) | F0 | 001 | D0 | — |
| | F5 | 003 | D6 | — |
| | F6 | 004 | D7 | — |
| | F7 | 001 | D5 | — |
| | F8 | 002 | D3 | — |
| | F9 | 003 | D4 | — |
| | F12 | 004 | D10 | — |
| | F13 | 005 | D11 | — |
| Time 5
(Fig. 15) | F14 | 004 | D9 | — |
| | F0 | 001 | D0 | — |
| | F1 | 002 | D1 | — |
| | F2 | 003 | D2 | — |
| | F3 | 004 | D3 | — |

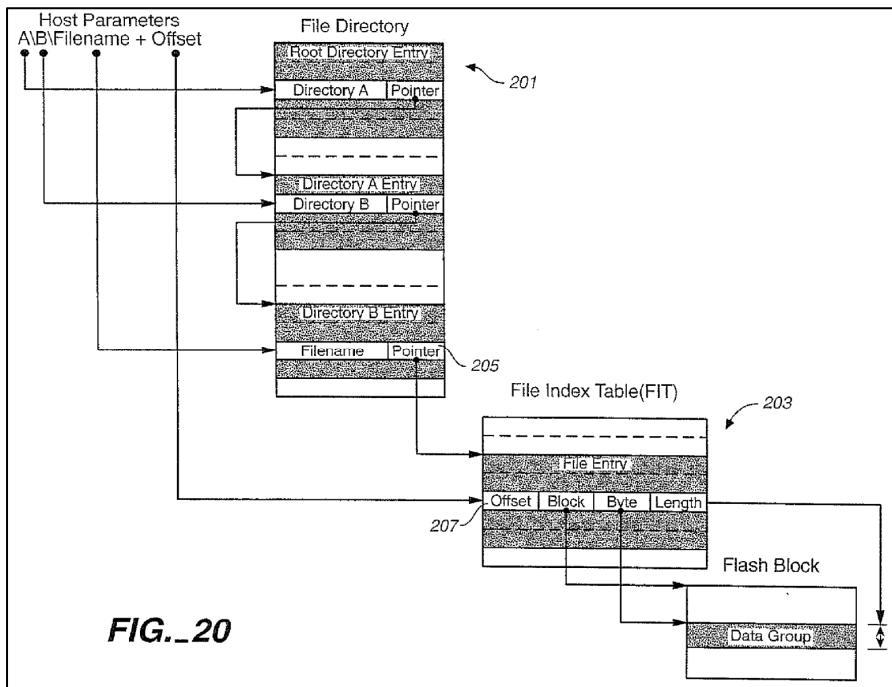
254. It is my opinion that Sinclair-233 renders this limitation obvious.

- (g) **Limitation 1[c][iv]:** *for each of the data write requests, identify from the first division operation a third address portion, and use a second division operation on the third address portion to identify an address value to select one of the erase units from the subset which corresponds to the addressed segment and to identify a storage location within the selected one of the erase units;*

255. Sinclair-233 discloses insert commands (*i.e.*, **data write requests**). Section VII.D.1.b. Each insert command includes a file pathname and an offset. EX1005, FIG. 12B. A delimiter operation on the file pathname (*i.e.*, **first division operation**) with the offset **identifies** the file entry 207 (*i.e.*, **third address portion**). Via the delimiter operation “[t]he pathname ... is used to traverse the hierarchy of the file directory” 201, where the “directory [is] then searched to find an entry 205 for the filename supplied by the host.” EX1005, 30:40-47. “When found, the entry 205 provides a pointer to a group of contiguous entries in the FIT 203 for that file.” EX1005, 30:48-49, FIG. 20. “The offsets of these entries are then compared with the offset address supplied by the host in order to identify a correct entry 207.” EX1005, 30:50-53.

256. A POSITA would understand, or find it obvious, to incorporate apply integer division to the bitstring in entry 207 (*i.e., use a second division operation on the third address portion*) in order to *identify* a block number (*i.e., address value to select one of the erase units from the subset which corresponds to the addressed segment*) and a byte number (*i.e., storage location within the selected one of the erase units*). “The block address field specifies the physical address of the block or metablock within the memory system that contains the data group.” EX1005, 32:26-44. And “[t]he byte address field specifies the address of the page within the block or metablock, and the byte within the page, at which the data group starts.” EX1005, 32:26-44.

257. A POSITA would understand entry 207 to be a bitstring with bitfields for the starting block number, starting byte number, and length of the data group. “The selected entry 207 of the FIT 203, in this example, contains the physical block and byte of the memory location containing data to which access is being sought by the host.” *See* EX1005, 30: 57-60, FIG. 20:



258. A POSITA would be motivated to use integer division operation on the obtained entry 207 bitstring to extract the block number and byte number. Integer division by a certain power-of-

two extracts bitfields out of bitstrings. Specifically, the quotient of integer division by certain power-of-two will extract the bitfields of the most significant bits. Integer division is preferable because it is fast, as division by a power-of-two on a bitstring is implemented with a simple bit shift operation. The block number is the higher order or most significant bitfield (*i.e.*, leftmost bits in the bitstring) of entry 207 bitstring. EX1005, FIG. 20 Applying integer division on the bitstring causes the block number and byte number to be extracted through the quotient and remainder. *See* EX1005, 32:37-39, FIG. 20. A POSITA would be motivated to implement integer division via a bit shift operation to perform the extraction of these two bitfields quickly.

259. It is my opinion that Sinclair-233 renders this limitation obvious.

(h) **Limitation 1[d]:** *wherein said circuitry is to execute each one of the data write requests by programming accompanying data into the storage location that was identified for that one of the data write requests; and*

260. Sinclair-233 discloses “Host Interface & Memory Controller 8” (*i.e.*, **circuitry**) for “convert[ing] between logical addresses of data used ... [for] data programming and reading” (*i.e.*, *execute each one of the data write requests*). EX1005, 5:19-25. Sinclair-233 describes the insert command (*i.e.*, *each one of the data write requests*) as “[i]nserting data within specified file at <offset> from start” (*i.e.*, *programming accompanying data into the storage location that was identified for that one of the data write requests*). EX1005, FIG. 12A:

| DATA COMMANDS | | |
|---------------|---|--|
| Command | Parameters | Description |
| Write | < fileID > | Appends data to end of the specified file. |
| Insert | < fileID > < offset > | Inserts data within specified file at < offset > from start. |
| Update | < fileID > < offset > | Overwrites data in specified file at < offset > from start. |
| Read | < fileID > < offset > | Reads data from specified file at < offset > from start. |
| Remove | < fileID > < offset 1 >
< offset 2 > | Removes data within specified file between < offset 1 > and < offset 2 > from start. |

FIG. 12A

261. It is my opinion that Sinclair-233 renders this limitation obvious.

- (i) **Limitation 1[e]:** *wherein said circuitry is also to execute each one of the incoming erase requests by controlling erasure in the flash memory of one or more of the erase units in the subset corresponding to the addressed segment that was identified from the incoming address information accompanying the one of the incoming erase requests.*

262. Sinclair-233 discloses “Host Interface & Memory Controller 8” (*i.e.*, **circuitry**) *is also to execute each one of the incoming* erase commands (*i.e.*, **erase requests**). EX1005, 5:19-25. Sinclair-233 describes the erase command as “[i]ndicat[ing] ... data for the specified file should be deleted and that file data should be erased immediately” (*i.e.*, **controlling erasure in the flash memory of one or more of the erase units in the subset corresponding to the addressed segment that was identified from the incoming address information accompanying the one of the incoming erase requests**). EX1005, FIG. 12B:

| FILE COMMANDS | | |
|---------------|-----------------------|--|
| Command | Parameters | Description |
| Open | < fileID > | Requests resources to be made available for writing to the specified file. |
| Close | < fileID > | Resources may be removed immediately from the specified file, and the file may be scheduled for garbage collection. |
| Close_after | < fileID > < length > | Resources may be removed from the specified file after further transfer of < length> of file data from host, at which time the file may be scheduled for garbage collection. |
| Delete | < fileID > | Indicates that table entries and data for the specified file should be deleted. File data may be erased. |
| Erase | < fileID > | Indicates that table entries and data for the specified file should be deleted and that file data should be erased immediately. |

FIG. 12B

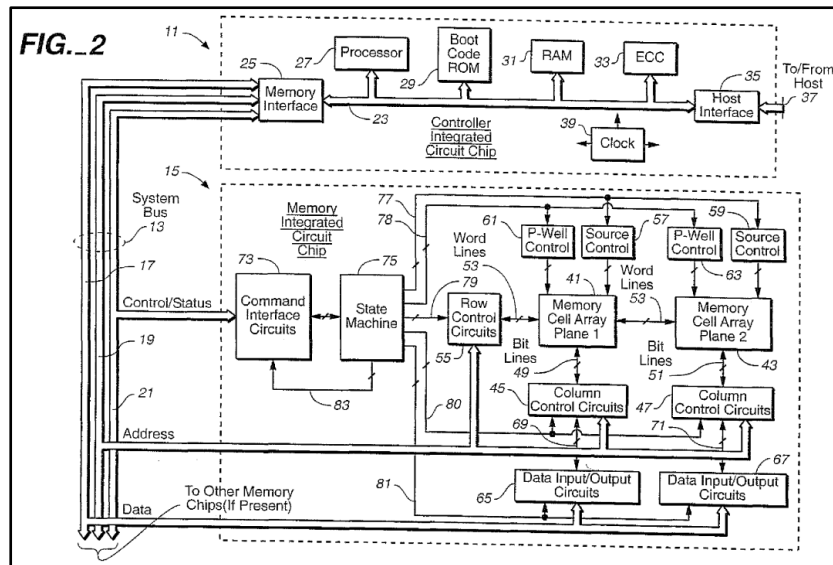
263. It is my opinion that Sinclair-233 renders this limitation obvious.

2. Claim 2, Limitation 2[pre]: *The storage device of claim 1 wherein:*

264. As addressed in Section VII.A.2, it is my opinion that Sinclair-233 discloses this limitation.

(a) Limitation 2[a]: *said circuitry comprises at least one processor; and*

265. Sinclair-233 discloses Host Interface & Memory Controller 8” (*i.e., circuitry*) “is usually implemented on a single integrated circuit chip 11,” which includes processor 27 (*i.e., comprises at least one processor*). EX1005, 5:26-40, FIG. 2:



266. It is my opinion that Sinclair-233 renders this limitation obvious.

- (b) **Limitation 2[b]:** *the storage device further comprises instructions stored on at least one physical storage medium, said instructions when executed to control the at least one processor.*

267. Sinclair-233 discloses the controller on memory system's 2 (*i.e.*, **storage device**) "interfaces with system bus 13," where the "functions normally connected to the bus are a processor 27[] [and] ... read-only-memory (ROM) 29 containing code to initialize ('boot') the system" (*i.e.*, **comprises instructions stored on at least one physical storage medium, said instructions when executed to control the at least one processor**). EX1005, 5:41-56.

268. It is my opinion that Sinclair-233 renders this limitation obvious.

3. Claim 8, Limitation 8[pre]: *The storage device of claim 1 wherein*

269. As addressed in Section VII.A.2, it is my opinion that Sinclair-233 discloses this limitation.

- (a) **Limitation 8[a]:** *the third address portion specifies a location within a sequential range of locations that correspond to the erase units in the subset.*

270. The bitstring of entry 207 constitutes the **third address portion**. Section VII.D.1.g. Sinclair-233 discloses a file containing block numbers within a range of sequentially numbered blocks

(i.e., *sequential range of locations that correspond to the erase units in the subset*). EX1005, 29:35-52, FIG. 19. For example, “FIG. 19 illustrates a sequence of index entries in a file index table (FIT) for one file.” EX1005, 29:35-38, FIG. 19:

FIG. 19

| File Index Table (FIT) | | | | |
|------------------------|--------|-------|------|--------|
| | OFFSET | BLOCK | BYTE | LENGTH |
| Time 0
(Fig. 14A) | F0 | 001 | D0 | — |
| | F1 | 002 | D1 | — |
| Time 2
(Fig. 14C) | F0 | 001 | D0 | — |
| | F5 | 003 | D6 | — |
| | F6 | 004 | D7 | — |
| | F7 | 001 | D5 | — |
| | F8 | 002 | D3 | — |
| | F9 | 003 | D4 | — |
| | F10 | 002 | D2 | — |
| Time 4
(Fig. 14E) | F0 | 001 | D0 | — |
| | F5 | 003 | D6 | — |
| | F6 | 004 | D7 | — |
| | F7 | 001 | D5 | — |
| | F8 | 002 | D3 | — |
| | F9 | 003 | D4 | — |
| | F12 | 004 | D10 | — |
| | F13 | 005 | D11 | — |
| Time 5
(Fig. 15) | F0 | 001 | D0 | — |
| | F1 | 002 | D1 | — |
| | F2 | 003 | D2 | — |
| | F3 | 004 | D3 | — |

The FIT entries show the file spanning a range of five blocks—blocks “001” to “005.” EX1005, FIG. 19. Entry 207 refers to a location within that sequential range of blocks.

271. It is my opinion that Sinclair-233 renders this limitation obvious.

4. Claim 9, Limitation 9[pre]: *The storage device of claim 1 wherein*

272. As addressed in Section VII.D.3, it is my opinion that Sinclair-233 discloses this limitation.

- (a) **Limitation 9[a]: *the first division operation and the second division operation each subdivide a value obtained from the incoming address information accompanying each of the write requests into respective bitfields.***

273. A POSITA would be motivated to implement the delimiter operation (i.e., *first division operation*) as subdividing the file pathname (i.e., *value obtained from the incoming address*

information accompanying each of the write requests) into directory names and the filename. Integer division by a certain power-of-two extracts bitfields out of bitstrings. Integer division is preferable because it is fast, as division by a power-of-two on a bitstring is implemented with a simple bit shift operation. A file pathname is a long bitstring where each directory name, file name, and ‘\’ character is a bitfield. This requires multiple operations to identify and extract each bitfield. And for time sensitive operations like data retrieval and storage, speed provided by integer division is put at a premium. Additionally, simple operations like bit shifting are more ideal on external devices, such as storage devices, where processing resources are limited. A POSITA would be motivated to implement delimiting with integer division by a power-of-two to subdivide the file pathname into respective bitfields.

274. A POSITA would understand, or find it obvious, to apply integer division by a power-of-two (*i.e., second division operation*) to the entry 207 bitstring (*i.e., value obtained from the incoming address information accompanying each of the write requests*), subdividing it into block-number and byte-number (*i.e., into respective bitfields*). Section VII.D.1.g; *see* EX1005, 32:37-39.

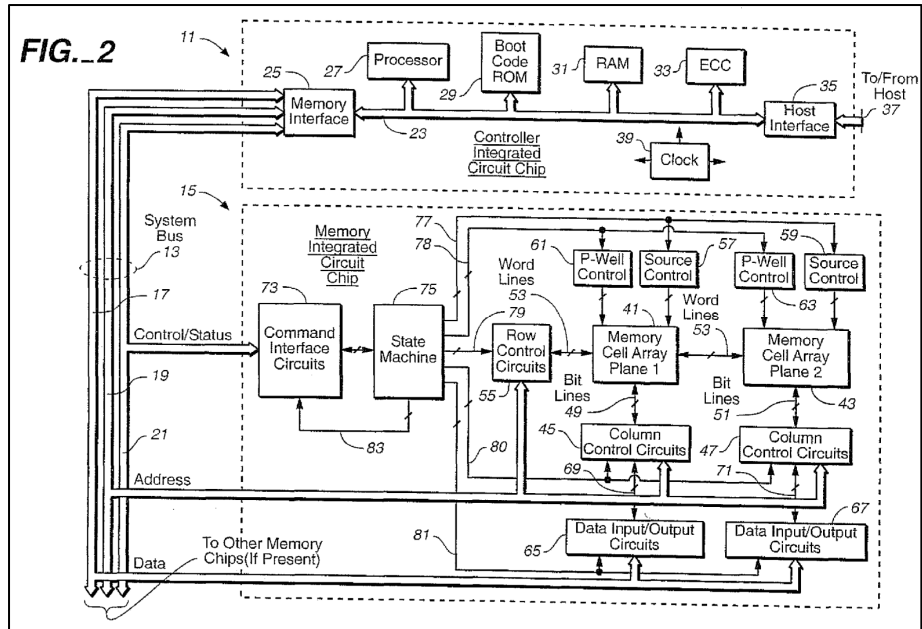
275. It is my opinion that Sinclair-233 renders this limitation obvious.

5. Claim 18, Limitation 18[pre]: *The storage device of claim 1 wherein:*

276. As addressed in Section VII.A.2, it is my opinion that Sinclair-233 discloses this limitation.

(a) Limitation 18[a]: *the flash memory comprises multiple planes; and*

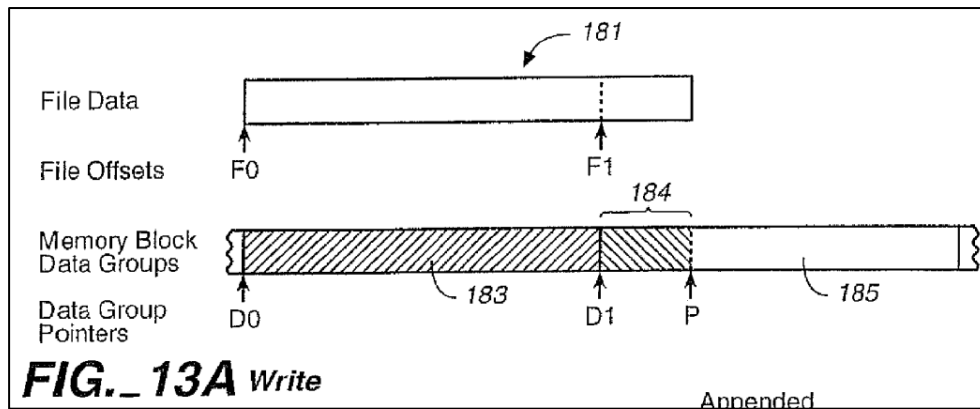
277. Sinclair-233 discloses “memory chip 15 ... typically contains an array of memory cells organized into multiple sub-arrays or planes, two such planes 41 and 43 being illustrated for simplicity but more, such as four or eight such planes, may instead be used.” EX1005, 5:57-61, FIG. 2:



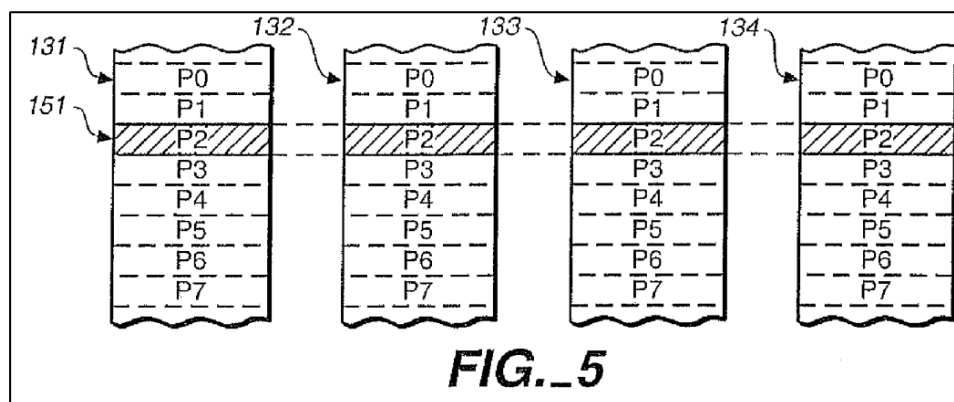
278. It is my opinion that Sinclair-233 renders this limitation obvious.

- (b) **Limitation 18[b]:** *the circuitry is to, for each one of a set of the data write requests having sequential addressing, identify a commonly addressed page and concurrently write data accompanying the set of write requests into respective ones of the multiple planes in a manner corresponding to the commonly addressed page.*

279. A POSITA would understand that Sinclair-233 discloses data write commands, such as the insert command, where the data to be written spans across several sequential addresses (*i.e.*, *for each one of the set of data write requests having sequential addressing*). For instance, “[r]eferring to FIG. 13A,” “data file 181 ... is larger than the storage capacity of one block” EX1005, 16:46-50. “A portion 184 of ... data file 181 is therefore also written into ... second block 185.” EX1005, 16:50-51, FIG. 13A:



280. To increase throughput for these large write commands, a POSITA would be motivated to use metapages as disclosed in Sinclair-233. Per Sinclair-233, “in order to increase the memory system operational parallelism, ... pages within two or more blocks may be logically linked into metapages.” EX1005, 7:62-64, FIG. 5:



“The metapage 151, for example, includes page P2 in of each of the four blocks” (*i.e.*, **identify a commonly addressed page**). EX1005, 7:66-8:3. “[M]etapages” include “pages in separate blocks in different planes.” EX1005, 8:3-7. This allows “for high system performance” due to “program[ming] and read[ing] the maximum amount of data in parallel across all four planes” (*i.e.*, **write data accompanying the set of incoming data access requests concurrently into respective ones of the multiple planes**). EX1005, 8:3-12. A POSITA would take advantage of this parallelism large write commands or multiple write commands spanning several sequential addresses.

281. It is my opinion that Sinclair-233 renders this limitation obvious.

6. **Claim 19-21**

282. Claims 19-21 is substantively identical to claim 1 incorporated with claims 9 and 22. Section VII.A.12.

7. **Claim 22, Limitation 22[pre]: *The storage device of claim 1 wherein***

283. As addressed in Section VII.A.2, it is my opinion that Sinclair-233 discloses this limitation.

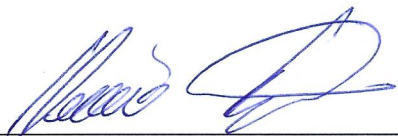
- (a) **Limitation 22[a]: *the second address portion comprises a first set of bits and a second set of bits, the bits of the first set of bits being mutually-exclusive to the bits of the second set of bits, wherein the first division operation is to separate the first set of bits from the second set of bits, and the circuitry is to identify the addressed segment from the first set of bits.***

284. The file pathname constitutes the *second address portion*. Section VII.D.1.d. The file pathname is a series of bitfields corresponding to directory names, the filename, and the “\” character. EX1005, 14:47-61. When dividing the pathname into these bitfields with a delimiter operation (*i.e., first division operation*), (Section VII.D.1.f) the filename bitfield (*i.e., first set of bits*) is *separated* from the next “\” character bitfield (*i.e., mutually exclusive bits of the second set of bits*). EX1005, 14:47-61, 30:40-49, FIG. 20; Section VII.D.1.f-g. The separate filename *identifies* the file (*i.e., addressed segment*). Section VII.D.1.f.

285. It is my opinion that Sinclair-233 renders this limitation obvious.

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

Dated: August 5, 2025



Heiner Litz, Ph.D.

CLAIM LISTING

| Claim 1 | |
|-----------|---|
| 1[pre] | A storage device comprising: |
| 1[a] | flash memory comprising erase units; and |
| 1[b] | circuitry to receive from a host, and control execution of, incoming requests including data write requests and erase requests, wherein each of the incoming requests is accompanied by incoming address information; |
| 1[c] | wherein said circuitry is to, for each of the incoming requests: |
| 1[c][i] | derive, from the accompanying incoming address information, a first address portion and a second address portion |
| 1[c][ii] | identify an addressed block device from the first address portion, |
| 1[c][iii] | use a first division operation on the second address portion, to identify an addressed segment within the addressed block device, wherein the addressed segment corresponds to a subset of the erase units of the flash memory, and |
| 1[c][iv] | for each of the data write requests, identify from the first division operation a third address portion, and use a second division operation on the third address portion to identify an address value to select one of the erase units from the subset which corresponds to the addressed segment and to identify a storage location within the selected one of the erase units; |
| 1[d] | wherein said circuitry is to execute each one of the data write requests by programming accompanying data into the storage location that was identified for that one of the data write requests; and |
| 1[e] | wherein said circuitry is also to execute each one of the incoming erase requests by controlling erasure in the flash memory of one or more of the erase units in the subset corresponding to the addressed segment that was identified from the incoming address information accompanying the one of the incoming erase requests. |
| Claim 2 | |
| 2[pre] | The storage device of claim 1 wherein: |
| 2[a] | said circuitry comprises at least one processor; and |

| | |
|----------------|---|
| 2[b] | the storage device further comprises instructions stored on at least one physical storage medium, said instructions when executed to control the at least one processor. |
| Claim 3 | |
| 3[pre] | The storage device of claim 1 wherein: |
| 3[a] | said circuitry can receive a request from a host to copy host-selected data from one storage location in flash memory to another storage location in the flash memory, and is to issue commands to the flash memory, in response to the request from the host to copy, to read the host-selected data from a first physical storage location in the flash memory and write the host-selected data to a second physical storage location in the flash memory without sending the data being copied to the host; |
| 3[b] | the request from the host to copy is to designate destination address information corresponding to the second physical storage location; and |
| 3[c] | the circuitry is to use the first division operation on the designated destination address information to identify a destination segment within a destination block device. |
| Claim 4 | |
| 4[pre] | The storage device of claim 3 wherein |
| 4[a] | the circuitry is to identify from the destination address information a destination logical erase unit and is to select the second physical storage location to be within a physical erase unit that is already mapped to the destination logical erase unit. |
| Claim 5 | |
| 5[pre] | The storage device of claim 1 wherein |
| 5[a] | the storage device is to store information identifying a number of structures in the flash memory and is to transmit the information identifying the number of structures to a host, and wherein the storage device is to store the values representing a time since data was programmed on a basis that is respective to each structure of the number of structures, wherein each structure comprises at least one of a page or a unit of memory comprising one or more erase units of the flash memory, and wherein each value represents time since data was programmed into the respective structure. |
| Claim 6 | |
| 6[pre] | The storage device of claim 1 wherein |

| | |
|-----------------|--|
| 6[a] | the storage device is to store information identifying a number of structures in the flash memory and is to transmit the information identifying the number of structures to a host, and wherein the circuitry is to fulfill the at least one of the incoming requests by issuing commands to the flash memory for which a quantum of flash memory selected by the commands which are issued in response to the at least one request is dependent on the information transmitted to the host identifying the number of structures. |
| Claim 7 | |
| 7[pre] | The storage device of claim 1 wherein |
| 7[a] | the storage device is to store information identifying a number of structures in the flash memory and is to transmit to a host information that identifies an amount of flash memory that is to be erased as a unit. |
| Claim 8 | |
| 8[pre] | The storage device of claim 1 wherein |
| 8[a] | the third address portion specifies a location within a sequential range of locations that correspond to the erase units in the subset. |
| Claim 9 | |
| 9[pre] | The storage device of claim 8 wherein |
| 9[a] | the first division operation and the second division operation each subdivide a value obtained from the incoming address information accompanying each of the write requests into respective bitfields. |
| Claim 10 | |
| 10[pre] | The storage device of claim 1 wherein |
| 10[a] | the storage device is to store, for each segment having an erase unit which stores valid data, a value representing a time since data was programmed; |
| 10[b] | the storage device is to compare the values representing time since data was programmed to at least one threshold; |
| 10[c] | upon detection that a specific one of the values has satisfied the one or more thresholds, the circuitry is to copy data stored at one or more physical storage locations associated with the specific one of the values from a first physical erase unit to a second physical erase unit; and |

| | |
|-----------------|--|
| 10[d] | the storage device is to remap a logical address for the data being copied from the first physical erase unit to the second physical erase unit. |
| Claim 11 | |
| 11[pre] | The storage device of claim 10 wherein: |
| 11[a] | the flash memory is to have block devices, one of the block devices being the addressed block device; and |
| 11[b] | the storage device is to identify, for the data being copied, an address of a logical erase unit, the logical erase unit being associated with the addressed block device to the exclusion of each other one of the block devices; and |
| 11[c] | the storage device is to remap a correspondence between the address of the logical erase unit and the first physical erase unit to instead be between the address of the logical erase unit and the second physical erase unit. |
| Claim 12 | |
| 12[pre] | The storage device of claim 1 wherein: |
| 12[a] | the flash memory is to have block devices, including the addressed block device; and |
| 12[b] | the storage device is to store metadata on an independent basis for portions of flash memory associated with each one of the block devices, wherein the stored metadata is dependent on an erase count of associated physical memory space; |
| 12[c] | the storage device is to compare the metadata to at least one threshold; |
| 12[d] | upon detection that specific metadata has satisfied the one or more thresholds, the circuitry is to copy data stored at one or more physical storage locations associated with the specific metadata from a first physical erase unit to a second physical erase unit; and |
| 12[e] | the storage device is to remap a logical erase unit value associated with the third address portion from the first physical erase unit to the second physical erase unit. |
| Claim 13 | |
| 13[pre] | The storage device of claim 1 wherein: |
| 13[a] | the erase units of the flash memory are physical erase units; |
| 13[b] | the circuitry is to use the second division operation to identify a logical erase unit, wherein each logical erase unit is mapped to one of the physical erase units; and |

| | |
|-----------------|--|
| 13[c] | the storage device is to detect a failure condition of the physical erase units and is to remap each logical erase unit which corresponds to one of the physical erase units for which the failure condition is detected to instead correspond to one of the physical erase units for which the failure condition has not been detected and is to utilize the remapped correspondence to service one or more of the incoming data access requests. |
| Claim 14 | |
| 14[pre] | The storage device of claim 1 wherein the storage device is to: |
| 14[a] | store, for each segment having an erase unit which stores valid data, a value representing a time since data was programmed; |
| 14[b] | compare the values representing time since data was programmed to at least one threshold; and |
| 14[c] | responsive to detection of a condition that a specific one of the values representing time since data was programmed has satisfied the one or more thresholds, transmit information to a host wherein such transmitted information conveys address information for data stored at one or more physical storage locations associated with the specific one of the values. |
| Claim 15 | |
| 15[pre] | The storage device of claim 1 wherein: |
| 15[a] | the flash memory is to have block devices, including the addressed block device; and |
| 15[b] | the storage device is further to: |
| 15[b][i] | store metadata on an independent basis for portions of flash memory associated with each one of the block devices, |
| 15[b][ii] | receive a query, issued by a host, directed to a specific one of the block devices, and |
| 15[b][iii] | responsive to the query, compare the metadata to at least one threshold, to identify specific metadata, and transmit information to the host wherein such transmitted information conveys address information for data stored at one or more physical storage locations associated with the specific metadata. |
| Claim 16 | |
| 16[pre] | The storage device of claim 1 wherein: |

| | |
|-----------------|---|
| 16[a] | the flash memory is to have block devices, including the addressed block device; and |
| 16[b] | the storage device is further to: |
| 16[b][i] | store metadata on an independent basis for portions of flash memory associated with each one of the block devices, wherein the stored metadata represents an erase count of associated physical memory space, and |
| 16[b][ii] | receive a query, issued by a host, wherein said query is specific to one of the block devices, and responsive to the query, identify specific metadata, associated with the specified one of the block devices, and transmit information to the host that conveys the specific metadata. |
| Claim 17 | |
| 17[pre] | The storage device of claim 1 wherein the storage device is further to: |
| 17[a] | store metadata on an independent basis for portions of flash memory associated with each one of the block devices, wherein the stored metadata represents an erase count of associated physical memory space; |
| 17[b] | compare the metadata to at least one threshold; and |
| 17[c] | responsive to detection that specific metadata has satisfied the one or more thresholds, transmit information to a host wherein such transmitted information conveys address information for data stored at one or more physical storage locations associated with the specific metadata. |
| Claim 18 | |
| 18[pre] | The storage device of claim 1 wherein: |
| 18[a] | the flash memory comprises multiple planes; and |
| 18[b] | the circuitry is to, for each one of a set of the data write requests having sequential addressing, identify a commonly addressed page and concurrently write data accompanying the set of write requests into respective ones of the multiple planes in a manner corresponding to the commonly addressed page. |
| Claim 19 | |
| 19[pre] | A storage device comprising: |
| 19[a] | flash memory comprising erase units; and |

| | |
|-----------------|---|
| 19[b] | circuitry to receive from a host, and control execution of, incoming requests including data write requests and erase requests, wherein each of the incoming requests is accompanied by incoming address information; |
| 19[c] | wherein said circuitry is to, for each of the incoming requests: |
| 19[c][i] | derive, from the accompanying incoming address information, a first address portion and a second address portion, |
| 19[c][ii] | identify an addressed block device from the first address portion, |
| 19[c][iii] | use a first operation on the second address portion to subdivide the incoming address information into first bit fields, and to identify an addressed segment within the addressed block device from a first one of the first bit fields, wherein the addressed segment corresponds to a subset of the erase units of the flash memory, and |
| 19[c][iv] | for each of the data write requests, use a second one of the first bit fields as a third address portion and use a second operation to subdivide the third address portion into second bit fields, to identify an address value to select one of the erase units from a first one of the second bit fields and to identify a storage location within the selected one of the erase units from a second one of the second bit fields; |
| 19[d] | wherein said circuitry is to execute each one of the data write requests by programming accompanying data into the storage location that was identified for that one of the data write requests; |
| 19[e] | wherein said circuitry is also to execute each one of the incoming erase requests by controlling erasure in the flash memory of one or more of the erase units in the subset corresponding to the addressed segment that was identified from the incoming address information accompanying the one of the incoming erase requests. |
| Claim 20 | |
| 20[pre] | A storage device comprising: |
| 20[a] | flash memory, the flash memory to have one or more block devices at a hierarchically-superior level, segments within each of the block devices at a hierarchically-intermediate level, and addressable storage locations within each of the segments at a hierarchically-inferior level, each segment corresponding to a respective, mutually exclusive set of one or more erase units, each segment being associated with only one of the block devices, each erase unit being associated with only one of the segments; and |

| | |
|-----------------|---|
| 20[b] | circuitry to receive from a host, and control execution of, incoming requests including data write requests and erase requests, wherein each of the incoming requests is accompanied by incoming address information; |
| 20[c] | wherein said circuitry is to, for each of incoming requests: |
| 20[c][i] | derive a first address portion and a second address portion, |
| 20[c][ii] | identify an addressed one of the block devices from the first address portion, |
| 20[c][iii] | use a first operation to subdivide the second address portion into first bit fields, and to identify an addressed segment within the addressed block device from a first one of the first bit fields, and to identify a third address portion from a second one of the first bit fields, wherein the addressed segment corresponds to a subset of the erase units of the flash memory, and |
| 20[c][iv] | for each of the data write requests, use a second operation to subdivide the third address portion into second bit fields, to identify an address value to select one of the erase units from a first one of the second bit fields, and to identify a storage location within the selected one of the erase units from a second one of the second bit fields; |
| 20[d] | wherein said circuitry is to execute each one of the data write requests by programming accompanying data into the storage location that was identified for that one of the data write requests; |
| 20[e] | wherein said circuitry is also to execute each one of the incoming erase requests by controlling erasure in the flash memory of one or more of the erase units in the subset corresponding to the addressed segment that was identified from the incoming address information accompanying the one of the incoming erase requests. |
| Claim 21 | |
| 21[pre] | A storage device comprising: |
| 21[a] | flash memory, the flash memory to have one or more block devices at a hierarchically-superior level, segments within each of the block devices at a hierarchically-intermediate level, and addressable storage locations within each of the segments at a hierarchically-inferior level, each segment corresponding to a respective, mutually exclusive set of one or more erase units, each segment being associated with only one of the block devices, each erase unit being associated with only one of the segments; and |
| 21[b] | circuitry to receive from a host, and control execution of, incoming requests including data write requests and erase requests, wherein each of the incoming requests is accompanied by incoming address information; |

| | |
|-----------------|--|
| 21[c] | wherein said circuitry is to, for each of the incoming requests: |
| 21[c][i] | derive, from the accompanying incoming address information, a first address portion and a second address portion, |
| 21[c][ii] | identify an addressed one of the block devices from the first address portion, |
| 21[c][iii] | use a first operation to subdivide the second address portion into first bit fields, and to identify an addressed segment within the addressed block device from a first one of the first bit fields and a third address portion from a second one of the first bit fields, wherein the addressed segment corresponds to a subset of the erase units of the flash memory, and |
| 21[c][iv] | for each of the data write requests, use a second operation to subdivide the third address portion into second bit fields, to identify, from a first one of the second bit fields, an address value to select one of the erase units and, from a second one of the second bit fields a storage location within the selected one of the erase units; |
| 21[d] | wherein said circuitry is to execute each one of the data write requests by programming accompanying data into the storage location that was identified for that one of the incoming data write requests; and |
| 21[e] | wherein said circuitry is also to execute each one of incoming erase requests by controlling erasure in the flash memory of the one or more erase units in the subset corresponding to the addressed segment that was identified from the incoming address information accompanying the one of the incoming erase requests. |
| Claim 22 | |
| 22[pre] | The storage device of claim 1 wherein |
| 22[a] | the second address portion comprises a first set of bits and a second set of bits, the bits of the first set of bits being mutually-exclusive to the bits of the second set of bits, wherein the first division operation is to separate the first set of bits from the second set of bits, and the circuitry is to identify the addressed segment from the first set of bits. |