

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SANDISK TECHNOLOGIES, INC. AND WESTERN DIGITAL
TECHNOLOGIES, INC.,
Petitioner

U.S. Patent No. 9,183,085

**DECLARATION OF LARA DOLECEK,
UNDER 37 C.F.R. § 1.68 IN SUPPORT OF PETITION FOR
INTER PARTES REVIEW**

TABLE OF CONTENTS

I.	INTRODUCTION	3
II.	QUALIFICATIONS AND PROFESSIONAL EXPERIENCE	5
III.	LEVEL OF ORDINARY SKILL IN THE ART	7
IV.	RELEVANT LEGAL STANDARDS	8
	A. Anticipation	9
	B. Obviousness	9
V.	TECHNICAL BACKGROUND	10
	A. Flash memory	10
	B. Error Correction Code	12
	1. Reed-Solomon (“RS”) code	13
	2. Hamming code	13
	3. BCH (Bose, Ray-Chaudhuri, Hocquenghem) Code	13
	4. Product Code	13
	C. Flash Based Solid-State Drive (SSD) Storage Device	17
	D. Page Stripe	19
VI.	THE ’085 PATENT	19
VII.	PROSECUTION HISTORY OF THE ’085 PATENT	20
VIII.	CLAIM CONSTRUCTION	21
	A. “gear” (claim 1)	22
	B. Means-Plus-Function Terms	23
IX.	FOUNDATIONS	23
	A. Prior Art	23
	1. Li	24
	2. Li combined with Cideciyan	26
	3. Li further combined with Frost	28
	4. Li further combined with Leung	29
	5. Analogous Prior Art	29

B.	Ground 1: Claims 1-2, 4-5, 14-17, 22-25, and 29-33 are obvious in view of Li and Cideciyan.	30
1.	Claim 1	30
2.	Claim 2	70
3.	Claim 4	71
4.	Claim 5	75
5.	Claim 14	77
6.	Claim 15	80
7.	Claim 16	85
8.	Claim 17	87
9.	Claim 22	88
10.	Claim 23	88
11.	Claim 24	89
12.	Claim 25	93
13.	Claim 29	95
14.	Claim 30	95
15.	Claims 31, 32, and 33	96
C.	Ground 2: Claims 1-2, 4-5, 14-17, 22-25, and 29-33 are obvious in view of Li, Cideciyan, and Frost.	102
1.	Claim Analysis	102
D.	Grounds 3 and 4: Claims 6-7 and 18 are rendered obvious by each of Combination of Li, Cideciyan, and Leung (Ground 3) and Combination of Li, Cideciyan, Frost, and Leung (Ground 4).	106
1.	Reasons to Combine of Leung with Each of Grounds 1 and 2.	106
2.	Claim 6	108
3.	Claim 7	110
4.	Claim 18	111
X.	DECLARATION	112

I. INTRODUCTION

1. I, Lara Dolecek, am making this declaration at the request of Sandisk Technologies, Inc. and Western Digital Technologies, Inc. (“Petitioner”), in the matter of the *Inter Partes* Review of U.S. Patent No. 9,183,085 (“the ’085 patent”) to Northcott.

2. Compensation for my work in this matter is based on an hourly rate. In addition, reasonable and customary expenses associated with my work and testimony in this matter are reimbursed. This compensation is not contingent on the outcome of this matter, nor is it contingent on the specifics of my testimony. I have no personal or financial stake, nor any interest in the outcome of the present proceeding.

3. In the preparation of this declaration, I have studied the following documents:

Ex.1001	U.S. Patent No. 9,183,085 (“the ’085 patent”)
Ex.1002	Prosecution History of U.S. App. No. 13/477,600 (issued as ’085 patent) (“’600 App”)
Ex.1005	U.S. Pub. 2014/0136927 to Li, et al.
Ex.1006	U.S. 9,037,951 to Cideciyan, et al.
Ex.1007	U.S. Patent No. 8,176,360 to Frost et al.
Ex.1008	U.S. 2008/0222490 to Leung et al.
Ex.1009	Solid-State Drive (SSD) Requirements and Endurance Test Method, JESD218, JEDEC STANDARD, September 2010
Ex.1010	U.S. 7,823,043 to Lasser

Ex.1011	U.S. 8,042,029 to Alrod et al.
Ex.1012	Performance Impact and Interplay of SSD Parallelism through Advanced Commands, Allocation Strategy and Data Granularity, Hu et al., ICS '11: Proceedings of the international conference on Supercomputing, pp. 96 – 107 (https://doi.org/10.1145/1995896.1995912), May 31, 2011
Ex.1013	FTL Design Exploration in Reconfigurable High-Performance SSD for Server Applications, ICS '09: Proceedings of the 23rd international conference on Supercomputing, pp. 338 – 349 (https://doi.org/10.1145/1542275.1542324), June 8, 2009
Ex.1014	Flash Management – Why and How?, A Detailed Overview of Flash Management Techniques, SMART Modular Technologies, 2009
Ex.1018	Solid State Drives 101, Cactus Technologies, available at https://www.cactus-tech.com/wp-content/uploads/2019/04/Solid-State-Drives-101-EBook.pdf
Ex.1019	Product Code Schemes for Error Correction in MLC NAND Flash Memories, Yang et al., December 29, 2011, available at: https://ieeexplore.ieee.org/document/6118315 .
Ex.1020	Error Control Coding Fundamentals and Applications, Lin et. al, 1983

4. In forming the opinions expressed below, I have considered:

- (1) The documents listed above;
- (2) Any additional documents discussed below; and
- (3) My own knowledge and experience based upon my work in the fields of semiconductor technology as described below.

5. Unless otherwise noted, all emphasis in any quoted material has been added.

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE

6. My complete qualifications and professional experience are described in my *Curriculum Vitae*, a copy of which can be found in Exhibit 1004. The following is a brief summary of my relevant qualifications and professional experience.

7. I am a full professor of Electrical and Computer Engineering at the University of California, Los Angeles (UCLA). At UCLA, I lead a research group called Laboratory for Robust Information Systems.

8. I earned a B.S., M.S., and Ph.D. in Electrical Engineering and Computer Sciences (EECS) as well as an M.A. degree in Statistics from the University of California, Berkeley (UC Berkeley). For my doctoral research on error correcting codes, I was awarded the 2007 David J. Sakrison Memorial Prize from the EECS Department at UC Berkeley, given for the most outstanding doctoral dissertation.

9. I have written over 160 peer-reviewed articles in leading venues on the topics related to error correction, statistical algorithms, and data processing and storage. I have also written a book titled “Channel Coding Methods for Non-Volatile Memories,” published in 2016.

10. I was a 2021 – 2022 Distinguished Lecturer of the IEEE Information Theory Society and I am currently an elected member of the Board of Governors

of the IEEE Information Theory Society.

11. I have received several career awards for my work, including the CAREER Award from the National Science Foundation, Intel Early Career Award, and IBM Faculty Award. With my research group and collaborators, I have received over dozen best paper awards and recognitions for the work on error correction coding and related topics, including Best Paper Awards from the IEEE Data Storage Society, Memorable Paper Awards from the Non-Volatile Memories Workshop, and Best-of-SELSE Awards from the IEEE Workshop on Silicon Errors in Logic – System Effects (SELSE).

12. I currently serve as an Associate Editor for the Coding Theory for IEEE Transactions on Information Theory, and I was a Technical Program Committee Co-Chair for the 2022 IEEE Information Theory Workshop, a flagship conference of the IEEE Information Theory Society.

13. I have used my education and experience researching, publishing and teaching, and my understanding of the knowledge, creativity, and experience of a person having ordinary skill in flash memory systems, including error correction coding (“ECC”) arts as applied to flash memory systems, to which the ’085 Patent relates, in forming the opinions expressed in this declaration, as well as any other materials discussed herein.

III. LEVEL OF ORDINARY SKILL IN THE ART

14. I understand that the level of ordinary skill may be reflected by the prior art of record, and that a Person of Ordinary Skill in The Art (“POSITA”) to which the claimed subject matter pertains would have the capability of understanding the scientific and engineering principles applicable to the pertinent art. I understand that a POSITA has ordinary creativity, and is not an automaton.

15. I understand that there are multiple factors relevant to determining the level of ordinary skill in the pertinent art, including (1) the levels of education and experience of persons working in the field at the time of the invention; (2) the sophistication of the technology; (3) the types of problems encountered in the field; and (4) the prior art solutions to those problems.

16. I have been informed by counsel that the earliest claimed priority date for the ’085 patent is May 22, 2012, although any given claim of the ’085 patent may or may not be entitled to the earliest claimed priority date. I am familiar with flash memory systems, including error correction coding (“ECC”) technologies as applied to flash memory systems pertinent to the ’085 patent. I am also aware of the state of the art at the time of the earliest claimed priority date.

17. Based on the technologies disclosed in the ’085 patent, I believe that a POSITA would include someone who had, as of the claimed priority date of the ’085 patent, a bachelor’s or the equivalent degree in electrical engineering or a

related field and 2-3 years of professional experience relating to flash memory systems, including error correction coding (“ECC”) technologies as applied to flash memory systems. In addition, I recognize that someone with less formal education but more experience, or more formal education but less experience could have also met the relevant standard for a POSITA. I believe that I am at least a POSITA and, furthermore, I have supervised students and engineers who were also POSITAs. Accordingly, I believe that I am qualified to opine from the perspective of a POSITA regarding the ’085 patent.

18. For purposes of this Declaration, unless otherwise noted, my opinions and statements, such as those regarding the understanding of a POSITA (and specifically related to the references I consulted herein), reflect the knowledge that existed in the art before the earliest claimed priority date of the ’085 patent.

IV. RELEVANT LEGAL STANDARDS

19. I have been asked to provide my opinions regarding whether claims 1-2, 4-7, 14-18, 22-25, and 29-33 (the “Challenged Claims”) of the ’085 patent would have been obvious to a POSITA at the time of the alleged invention in light of the prior art.

20. I am not an attorney. In preparing and expressing my opinions and considering the subject matter of the ’085 patent, I am relying on certain legal principles explained to me by counsel.

21. I understand that a claim is unpatentable if it is anticipated under 35 U.S.C. § 102 or obvious under 35 U.S.C. § 103.

A. Anticipation

22. I have been informed by counsel that a patent claim is unpatentable as anticipated if each element of that claim is present either explicitly or inherently in a single prior art reference. I have also been informed that, to be an inherent disclosure, the prior art reference must necessarily disclose the limitation, and the fact that the reference might possibly practice or contain a claimed limitation is insufficient to establish that the reference inherently teaches the limitation.

B. Obviousness

23. I have been informed and I understand that a claimed invention is unpatentable under 35 U.S.C. § 103(a) if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious to a POSITA at the time the invention was made. I understand that the appropriate analysis for determining obviousness of a claimed invention takes into account factual inquiries, including the level of ordinary skill in the art, the scope and content of the prior art, and the differences between the prior art and the claimed subject matter as a whole.

24. I have been informed and I understand that the United States Supreme Court has recognized several rationales for combining references or modifying a

reference to show obviousness of claimed subject matter. Some of these rationales include the following: (a) combining prior art elements according to known methods to yield predictable results; (b) simple substitution of one known element for another to obtain predictable results; (c) use of a known technique to improve a similar device (method, or product) in the same way; (d) applying a known technique to a known device (method, or product) ready for improvement to yield predictable results; (e) choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and (f) some teaching, suggestion, or motivation in the prior art that would have led a POSITA to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention. I have also been informed and I understand that a demonstration of obviousness does not require a physical combination or bodily incorporation, but rather may be found based on consideration of what the combined teachings would have suggested to a POSITA at the time of the alleged invention.

V. TECHNICAL BACKGROUND

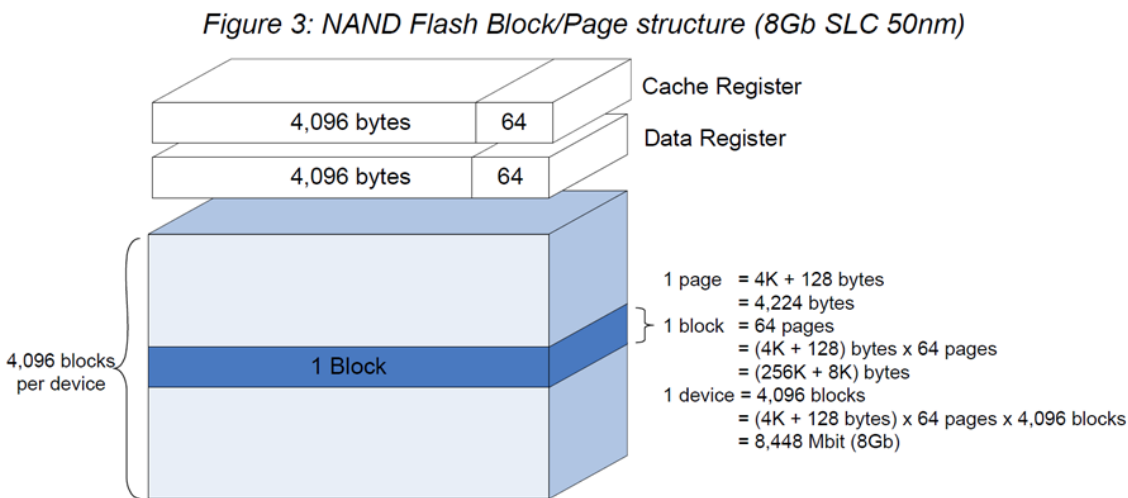
A. Flash memory

25. Flash is a nonvolatile solid state memory with the capability to retain stored data when unpowered. NAND and NOR are the two fundamental flash architectures used in electronic systems. NAND flash memory is more suitable for high-density data storage and is often employed in scenarios where cost-effective,

high-capacity storage is crucial, such as in USB drives, memory cards, and solid-state drives (SSDs). I will focus my discussion on the NAND flash memory technology below.

26. A flash memory chip may include multiple dies. A die is a physical piece of silicon where the NAND flash cells, pages, blocks, and planes are organized. A page is the smallest area of the flash memory that supports a write operation. An Erase Block is the smallest area of the flash memory that can be erased in a single operation. A plane is a collection of blocks within a NAND die.

27. An example NAND flash block and page structure is illustrated below.



Ex.1014, Spanjer, 6

28. There are challenges intrinsic to using NAND flash in a solid state drive (SSD). Ex.1014, Spanjer, 3. These challenges include:

- Need to erase before writing (Ex.1014, Spanjer, 6);
- Wear out mechanism that limits service life (Ex.1014, Spanjer, 9);
- Data errors caused by write and read disturb (Ex.1014, Spanjer, 6-8);
- Data retention errors (Ex.1014, Spanjer, 8); and
- Management of initial and runtime bad blocks (Ex.1014, Spanjer, 8).

29. Proper flash management techniques must incorporate mechanisms to overcome these challenges. Ex.1014, Spanjer, 9. There are many known techniques to address the challenges for achieving better reliability, higher performance, and a longer life span for NAND Flash. *Id.* Some of these techniques include using single-level cells (SLCs) and/or multi-level cells (MLCs), wear leveling, write amplification reduction, error detection and error correction coding (ECC), and bad block management. *Id.*

30. Single-Level Cell (SLC) and Multi-Level Cell (MLC) are well-known technologies in NAND flash. While SLC stores only one bit per cell, MLC stores two or even more bits per cell. MLC is inherently less reliable than SLC, and the MLC degrades faster than the SLC over time. Ex.1006, Cideciyan, 3:60-61, 3:66-4:1.

B. Error Correction Code

31. One of the key factors to increase flash reliability and lifetime is the implementation of an Error Detection and Correction mechanism. Below are some

well-known Error Correction algorithms (also called code types) used with NAND flash technology.

1. Reed-Solomon (“RS”) code

32. RS code was invented in 1960. RS codes are used in HDDs, CDs, NAND, and telecommunication and digital broadcast protocols. It is a linear code that is widely used.

2. Hamming code

33. Hamming code was invented in 1950, named after Richard Hamming. This algorithm detects and corrects single-bit errors, and detects (but does not correct) double-bit errors. It is a linear code that is widely used.

3. BCH (Bose, Ray-Chaudhuri, Hocquenghem) Code

34. BCH codes were invented in 1959 by Hocquenghem, and independently in 1960 by Bose and Ray Chaudhuri. It is a linear code that is widely used.

4. Product Code

35. Product code was first described by Elias in 1954. Ex.1020, Lin, 274-285. Textbooks from the 1980s include this well-known code type for introduction on error-correcting codes. Ex.1020, Lin, 274-278 (a textbook from 1983 describing “Product Codes” as introduction to error-correcting codes), xv (“The book can be used as a text for **an introductory course** on error-correcting codes”).

36. Product code has two component codes (also referred to as constituent

codes), C1 and C2. As explained in detail in the Lin textbook and as shown in the figure below, let C1 be an (n_1, k_1) linear code, and let C2 be an (n_2, k_2) linear code. Then an (n_1n_2, k_1k_2) linear code can be formed, such that each code word is a rectangular array of n_1 columns and n_2 rows, in which every row is a code vector in C1 and every column is a code vector in C2. “This **two-dimensional code** is called the direct product (or simply the product) of C1 and C2.” Ex.1020, Lin, 274-278. It has a code length of n_1n_2 , and a code rate of $\frac{k_1k_2}{n_1n_2}$. *Id.*

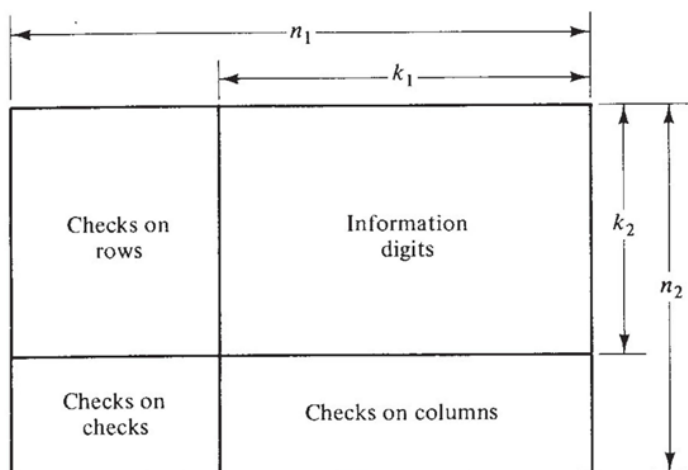


Figure 9.6 Code array for the product code $C_1 \times C_2$.

Ex.1020, Lin, 274.

37. A product code scheme used in MLC NAND Flash Memories is illustrated in Yang as shown below. Ex.1019, Yang, 4 (describing basics of product code scheme for error correction in MLC NAND flash memories).

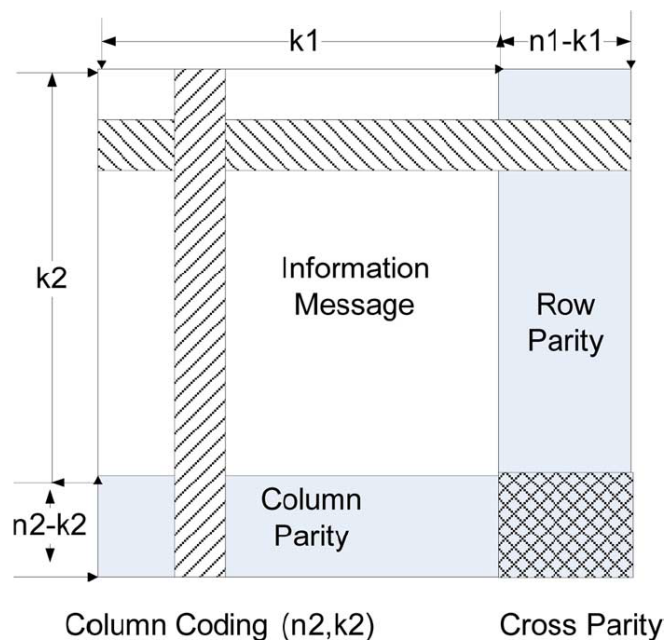


Fig. 5. Product code scheme.

Ex.1019, Yang, FIG. 5.

38. This is the same as the product code scheme used for error correction in flash memories in Cideciyan, as illustrated below. Ex.1006, Cideciyan, FIG. 4 (illustrating a produce code scheme with component codes C1 and C2 for error correction in flash memories).

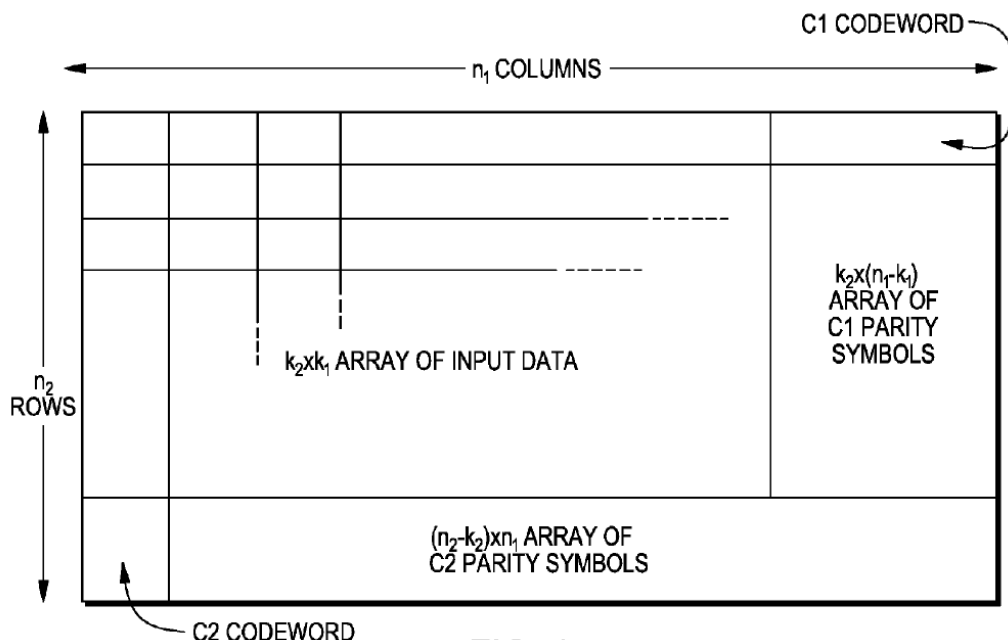


FIG. 4

Ex.1006, Cideciyan, FIG. 4.

39. Product code can form a long length code (of a code length $n_1 n_2$) with higher ECC capabilities using small length component code C1 (having an n_1 code length) and small length component code C2 (having an n_2 code length).

Ex.1019, Yang, 4.

40. As shown in the product code scheme, the C1 and C2 codewords are arranged orthogonally, which provides cross parity check and gives exceptional error performance. While “higher error correction capability can be achieved by using stronger BCH or RS codes,” they are “expensive both in terms of area and latency.” Ex.1019, Yang, 4. On the other hand, a product code using smaller component codes “achieves high error correction capability due to cross parity checking.” *Id.*, 1. “Compared to plain long length codes,” product code “has

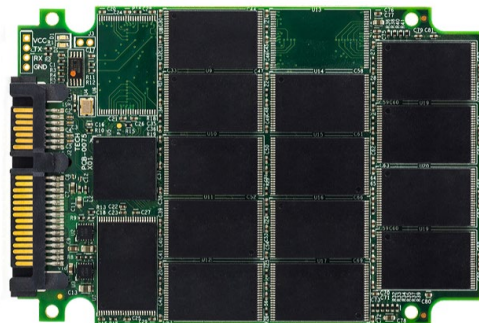
high performance from cross parity check, and low circuitry overhead.” *Id*;

Ex.1006, Cideciyan, FIG. 4, 11:29-31 (“the ‘orthogonal’ arrangement of C1 and C2 codewords ...**gives exceptional error performance.**”).

C. Flash Based Solid-State Drive (SSD) Storage Device

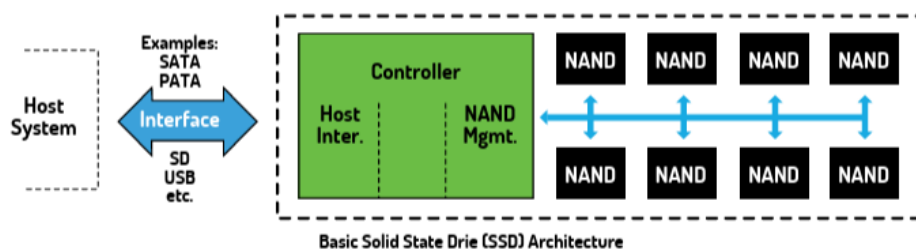
41. A solid-state drive (SSD) is a solid-state storage device that uses integrated circuit assemblies to store data persistently, typically using flash memory.

42. A real-life example of an SSD’s internal circuit board is illustrated below, where the rectangular components are NAND flash memory chips, and the square component to the left is the controller for this SSD.



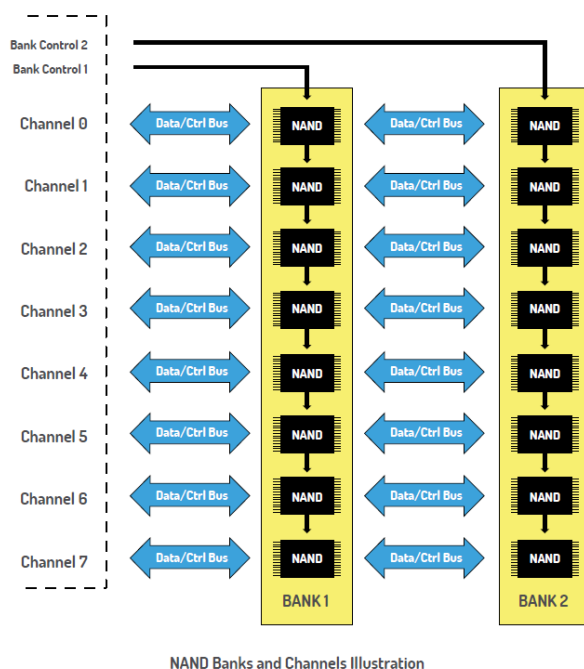
Ex.1018, SSD 101, 15.

43. A diagram of a basic SSD architecture is shown below. The basic SSD consists of a controller which manages one or more NAND components, each of which could be comprised of multiple NAND dies.



Ex.1018, SSD 101, 15.

44. The diagram below shows a common SSD NAND configuration. In this example, there are 8 channels connected between the SSD controller and the NAND chips. For each channel, there are 2 banks of NAND components.



Ex.1018, SSD 101, 17.

45. Channels in SSD refer to the number of flash chips the controller can talk to simultaneously. Low end SSDs usually have 2 or 4 channels; high end SSDs usually have 8 channels, some have 10 channels. Each flash chip at the same

location in a channel together constitutes a bank. Each channel can have multiple chips. Ex.1018, SSD 101, 18.

D. Page Stripe

46. Page stripe is one well-known technique for providing better performance, including quick read and write operations and better data protection. A page stripe includes pages distributed in multiple storage locations (e.g., multiple dies in separate dies), such that the pages can be accessed in parallel (e.g., through separate channels). *See* Ex.1007, Frost, Abstract, FIGS. 7A-7B, 11:20-64.

VI. THE '085 PATENT

47. The '085 patent is directed toward flash memory systems and to a method of adaptively selecting an ECC scheme from a plurality of such schemes based on data error rate. Ex.1001, '085 patent, Title.

48. In the Background section, the '085 patent acknowledges that “Flash memory devices require the use of a form of Error Correction Coding (ECC) to detect and correct the errors that inevitably occur.” Ex.1001, '085 patent, 1:52-54. The '085 patent further acknowledges that various different ECC schemes are well known in the art, and have different error correction strengths associated therewith. *See e.g.*, Ex.1001, '085 patent, 1:54-58. The '085 patent criticizes the conventional approaches of applying ECC to flash memory, saying that it “can be inefficient at achieving relatively low uncorrectable error rates over the total service life and

traffic loads.” Ex.1001, ’085 patent, 2:23-27.

49. The ’085 patent claims to provide a flash-based storage system with “relatively low uncorrectable bit error rates, low write amplification, long life, fast and efficient retrieval, and efficient storage density.” Ex.1001, ’085 patent, Abstract. One purported novelty of the ’085 patent is “***Adaptive ECC***” (Ex.1001, ’085 patent, 48:47), which selects from ECC schemes of different ECC strengths (referred to as different “gears”) based on an observed bit error rate, e.g., one ECC scheme with a higher ECC strength (a “gear” with a higher correction capability) is selected based on a higher observed bit error rate, and another ECC scheme with a lower ECC strength (a “gear” with a lower correction capability) is selected based on a lower observed bit error rate. *See also* Ex.1001, ’085 patent, Title (“Systems and Methods for ***Adaptively Selecting*** From Among a Plurality of Error Correction Coding Schemes in a Flash Drive For Robustness And Low Latency”).

VII. PROSECUTION HISTORY OF THE ’085 PATENT

50. The ’085 patent application was filed on May 22, 2012.

51. Multiple Office Actions rejected the claims as unpatentable over prior art including Diggs (US. App. Pub. 2009/0070651) in view of Manning (U.S. App. Pub. 2013/0246891), finding that Diggs teaches dynamic ECC scheme gear selection based on a determined bit error rate, and Manning teaches that data payload size is different for different gears. Ex.1002, 480-482, 559.

52. In response to the Final Office Action of July 16, 2014, Patentee filed an RCE, amending the claims to recite “determining a bit error rate associated with a region comprising a fixed number of two or more flash memory pages or integer fractions thereof, wherein the region stores at least data payload and primary and secondary ECC parity symbols corresponding to the data payload of the region.” (see limitations [1.1.1] and [1.1.2]). Ex.1002, 614-630. The amended claims were again rejected based on prior art in the Office Action of February 2, 2015. Ex.1002, 642-679.

53. In response, patentee amended the claims to recite “wherein the two or more flash memory pages of a region can be read simultaneously,” (see limitation [1.1.1]) which resulted in allowance. Ex.1002, 693-705, 715-719.

54. However, as will be explained in detail below, the claimed techniques, including the adaptive ECC scheme gear selection and gears with different data payload sizes as found by Examiner to be well-known), and the additional features recited in the amendments which resulted in allowance, were already well-known concepts at the time the '085 patent was filed.

VIII. CLAIM CONSTRUCTION

55. It is my understanding that in order to properly evaluate the '085 patent, the terms of the claims must first be interpreted. It is my understanding that for the purposes of this inter partes review, the claims are to be construed under the

so-called Phillips standard, under which claim terms are given their ordinary and customary meaning as would have been understood by one of ordinary skill in the art in light of the specification and prosecution history, unless the inventor has set forth a special meaning for a term. I have also been informed that claim terms only need to be construed to the extent necessary to resolve the obviousness inquiry.

56. I have reviewed the entirety of the '085 patent, as well as its prosecution history. It is my opinion that for purposes of applying the prior art presented herein to evaluate the patentability of the claims, no term requires express construction, other than the terms identified below.

A. “gear” (claim 1)

57. A POSITA would have understood that “gear” means “a set of parameter(s) specifying an error correction coding scheme, including one or more of the payload capacity, the type and parameters of each component code, the interleaving scheme, and parameters used for determining whether the gear is appropriate for a particular situation.” In the claims, a gear corresponds to an ECC scheme. *See* limitation [1.2.1].

58. This construction is a quotation from the specification.

Gear: In the context of this disclosure, a “gear” is a set of parameter(s) specifying an error correction coding scheme, including one or more of the payload capacity, the type and parameters of each component code,

the interleaving scheme, **and parameters used for determining whether the gear is appropriate for a particular situation.** For example, the “Gear 1” sections of Tables III, IV, and V, plus the Threshold G1, collectively define Gear 1.

Ex.1001, '085 patent, 20:18-25.

B. Means-Plus-Function Terms

59. Claims 24-25 include means-plus-function claim terms. I was informed construction of means-plus-function claim terms that invoke 35 U.S.C. §112, ¶6, requires identifying the claim terms and identifying corresponding structure disclosed in the specification that performs the identified functions and equivalents thereof. These means-plus-function claim terms are identified below at [24.1.1], [24.2.1], [24.3], [25.0], and [25.1], together with the corresponding structure disclosed in the specification.

IX. GROUNDS

A. Prior Art

60. Li is the primary prior art reference for each ground listed above. Li teaches or at least suggests every claim element in these grounds. Cideciyan, Frost, and Leung are secondary references provided to show various teachings or implementation details that would be obvious for Li to use.

1. Li

61. Li teaches adaptive ECC techniques for flash memory based data storage systems. Li teaches that gears corresponding to ECC schemes provide different error correcting capabilities, and are selected based on a monitored bit error rate.

62. As shown in FIGS. 1 and 2A below, Li, like the '085 patent, relates to “adaptive ECC techniques for flash memory based data storage,” where elements of SSD controller 200 of FIG. 2A may “implement various hardware blocks of FIG. 1 (or functions performed by the hardware blocks) in whole or in part.” Ex.1005, Li, Title, [0126]; *see also id.*, [0109] (“FIG. 2A illustrates selected details of an embodiment of an SSD including an SSD controller using **adaptive ECC techniques** for flash memory based data storage.”), [0126] (“For example, ECC 261 implements one or more functions performed by the **Error Statistics Collecting/Tracking, Universal Encoder, Universal Decoder, and/or Code Library hardware blocks** of FIG. 1.”).

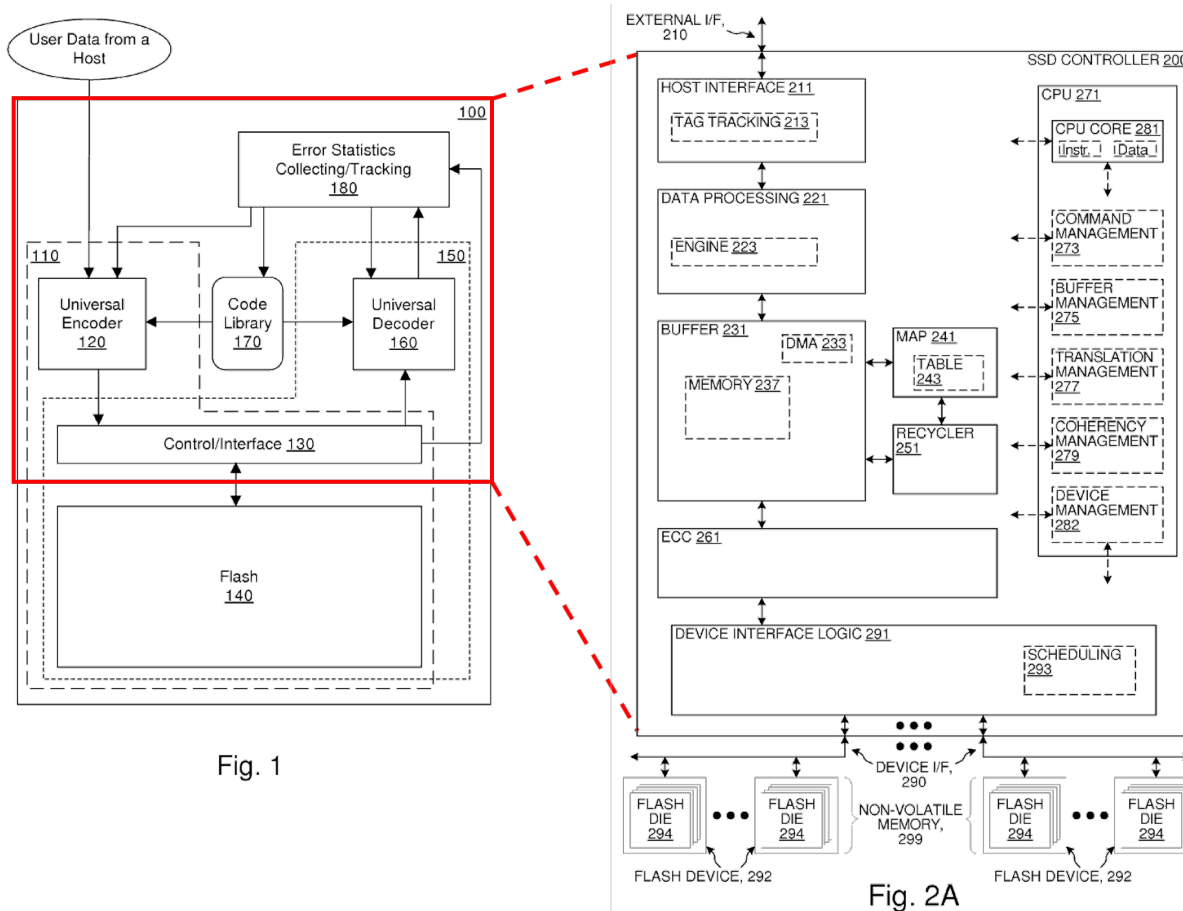


Fig. 1

Fig. 2A

Ex.1005, Li, FIGS. 1 and 2A, annotated.

63. Li states that “**adaptive ECC techniques** for use with flash memory enable improvements in flash memory lifetime, reliability, performance, and/or storage capacity.” Ex.1005, Li, [0019]. “The techniques include **a set of ECC schemes with various code types, code rates, and/or various code lengths** (providing **different error correcting capabilities**), and **error statistic collecting/tracking**.” *Id.* Li also describes conditions (e.g., thresholds) used to determining whether a particular ECC scheme is appropriate for a particular

situation. Ex.1005, Li, [0105].

64. Li also describes that a plurality of flash dies in the flash memory are **“accessible in parallel.”** Ex.1005, Li, [0111], [0112] (“device interfaces 290 are organized as: one or more busses with one or more flash devices 292 per bus; one or more groups of busses with one or more flash devices 292 per bus, where **busses in a group are generally accessed in parallel**; or any other organization of flash devices 292 onto device interfaces 290.”).

2. Li combined with Cideciyan

65. A POSITA would have found it obvious to implement Li’s error correction schemes using Cideciyan’s error correction technique, which uses a product code type ECC scheme (using primary and secondary codewords) to provide improved error correction.

66. Li describes a set of ECC schemes with **various code types** Ex.1005, Li, [0019] (“The techniques include **a set of ECC schemes** with **various code types**”). Cideciyan provides an example of a known ECC scheme using a well-known “product code” code type, dated back to 1950s and described in detail in textbooks. Ex.1020, Lin, 274-278. The product code which includes a primary component code C1 for generating primary codewords including ***primary ECC parity symbols***, and a secondary component code C2 for generating secondary codewords including ***secondary ECC parity symbols***. Ex.1006, FIG. 4 (illustrating

a “C1/C2 product code”), 10:44-49. In Cideciyan’s technique, the encoded symbols including data symbols and primary and secondary parity symbols are stored in a group of pages (*a region*).

67. Thus the combination of Li and Cideciyan teaches: *such that the region stores at least data payload and primary and secondary ECC parity symbols corresponding to the data payload of the region* – one of the two claim elements added during prosecution of the ’085 patent, after the Examiner found the claimed features of adaptive ECC scheme gear selection and gears with different data payload sizes were known.

68. Cideciyan’s ECC technique is applied on a group basis, where the group includes pages across different memory channels that are accessible in parallel. Thus, a POSITA implementing Li with Cideciyan’s product code type ECC scheme technique applied on a group basis would have similarly applied the ECC scheme across Li’s plurality of channels. Li explains that its plurality of flash dies in the flash memory are “accessible in parallel.” Ex.1005, Li, [0111], [0112]. Li also describes that its adaptive ECC scheme selection is performed on a “portion” basis, where the portion can be a group of multiple pages. Ex.1005, Li, [0022], [0097], [0099], [0038], [0039]. Cideciyan, which was not previously of record before the Examiner during the prosecution of the ’085 patent, also describes ECC techniques in flash based data storage systems. Ex.1006, Cideciyan

1:15-19. Cideciyan teaches that organizing a flash memory with groups, each group having a fixed number of pages, accessible in parallel. Ex.1006, Cideciyan, FIG. 2, 8:52-64 (describing Channels 1 through N in SSD, which were well-known techniques for parallel access in SSD); *see e.g.*, Ex.1012, Hu, 2 (Parallelism, including channel-level parallelism, is important for SSD performance and “has been regarded as a key to achieving the peak performance.”); Ex.1018, SSD 101, 18 (describing the well-known knowledge that channels in SSDs allow parallel access of flash memory dies); Ex.1007, Frost, Abstract, FIGS. 7A-7B, 11:20-64 (describing parallel access of memory dies using channels in SSDs); Ex.1006, Cideciyan, 11:44-46 (describing parallel access to pages distributed across channels).

69. Thus the combination of Li and Cideciyan teaches: *wherein the two or more flash memory pages of a region can be read simultaneously* – the other of the two claim elements added during prosecution of the '085 patent to overcome the cited prior art, after the Examiner found the claimed features of adaptive ECC scheme gear selection and gears with different data payload sizes were known.

70. The reasons to combine Li and Cideciyan are described in more detail in the analysis below.

3. Li further combined with Frost

71. As discussed above, both Li and Cideciyan teach accessing the flash

memory pages in parallel. Frost is also directed at a “flash-based storage system” (Ex.1007, Frost, Title), describes in greater detail how data in a particular region can be accessed, or read, “**at the same time.**” *Id.*, 6:46-67. In view of Frost, it would be obvious for Li’s parallel access capabilities to include *wherein the two or more flash memory pages of a region can be read simultaneously*. The reasons to combine Frost with Li and Cideciyan are described in more detail in the analysis below.

4. Li further combined with Leung

72. Leung is like Li in that it teaches selecting a gear in response to a change in a bit error rate. Leung is used to address some of the implementation details of the dependent claims directed to gear selection. The reasons to combine Frost with Li and Leung are described in more detail in the analysis below.

5. Analogous Prior Art

73. At the offset, it is noted that Li, Cideciyan, Frost, and Leung each are analogous art to the ’085 patent, because each is at least (1) in the same field of endeavor (flash memory based storage system); or (2) reasonably pertinent to a problem with which the inventor of the ’085 patent was involved (*e.g.*, to provide a flash-based storage system with “relatively low uncorrectable bit error rates, low write amplification, long life, fast and efficient retrieval, and efficient storage density” (Ex.1001, ’085 patent, Abstract). Compare Ex.1001, ’085 patent, Title (“Systems And Methods for Adaptively Selecting From Among a Plurality Of

Error Correction Coding Schemes in A Flash Drive for Robustness And Low

Latency”), 2:23-27 (related to “flash drive memory controllers, such as to solid-state drive memory controllers”); Ex.1005, Li, [0019] (“adaptive ECC techniques for use with flash memory enable improvements in flash memory lifetime, reliability, performance, and/or storage capacity”); Ex.1006, Cideciyan, 1:15-19 (“Methods and apparatus are provided for controlling data storage procedures, in particular the storage of error-correction (EC) encoded data, as well as other data management operations”); 2:16-18 (“The flash controller manages data in the flash memory generally”); Ex.1007, Frost, Title (“flash-based storage system”); Ex.1008, Leung, Abstract (“dynamic adjustment of an error control coding (ECC) code rate”), [0044] (“The system also includes a non-volatile memory device (640) capable of supporting ECC, such as, but not limited to, a NAND flash memory device.”).

B. Ground 1: Claims 1-2, 4-5, 14-17, 22-25, and 29-33 are obvious in view of Li and Cideciyan.

1. Claim 1

[1.0] *A method of selecting an error correction coding (ECC) scheme, the method comprising:*

74. To the extent the preamble is limiting, Li discloses or renders obvious [1.0].

75. Specifically, Li describes “encoding/decoding **in accordance with one or more of the ECC schemes, and dynamically switching**

encoding/decoding **amongst one or more of the ECC schemes** based at least in part on information from the error statistic collecting/tracking.” Ex.1005, Li, Abstract. Accordingly, Li discloses *a method of selecting an ECC scheme* from one or more of the ECC schemes, and dynamically switching to use that selected ECC scheme.

[1.1.1] *determining a bit error rate associated with a region comprising a fixed number of two or more flash memory pages or integer fractions thereof, wherein the two or more flash memory pages of a region can be read simultaneously,*

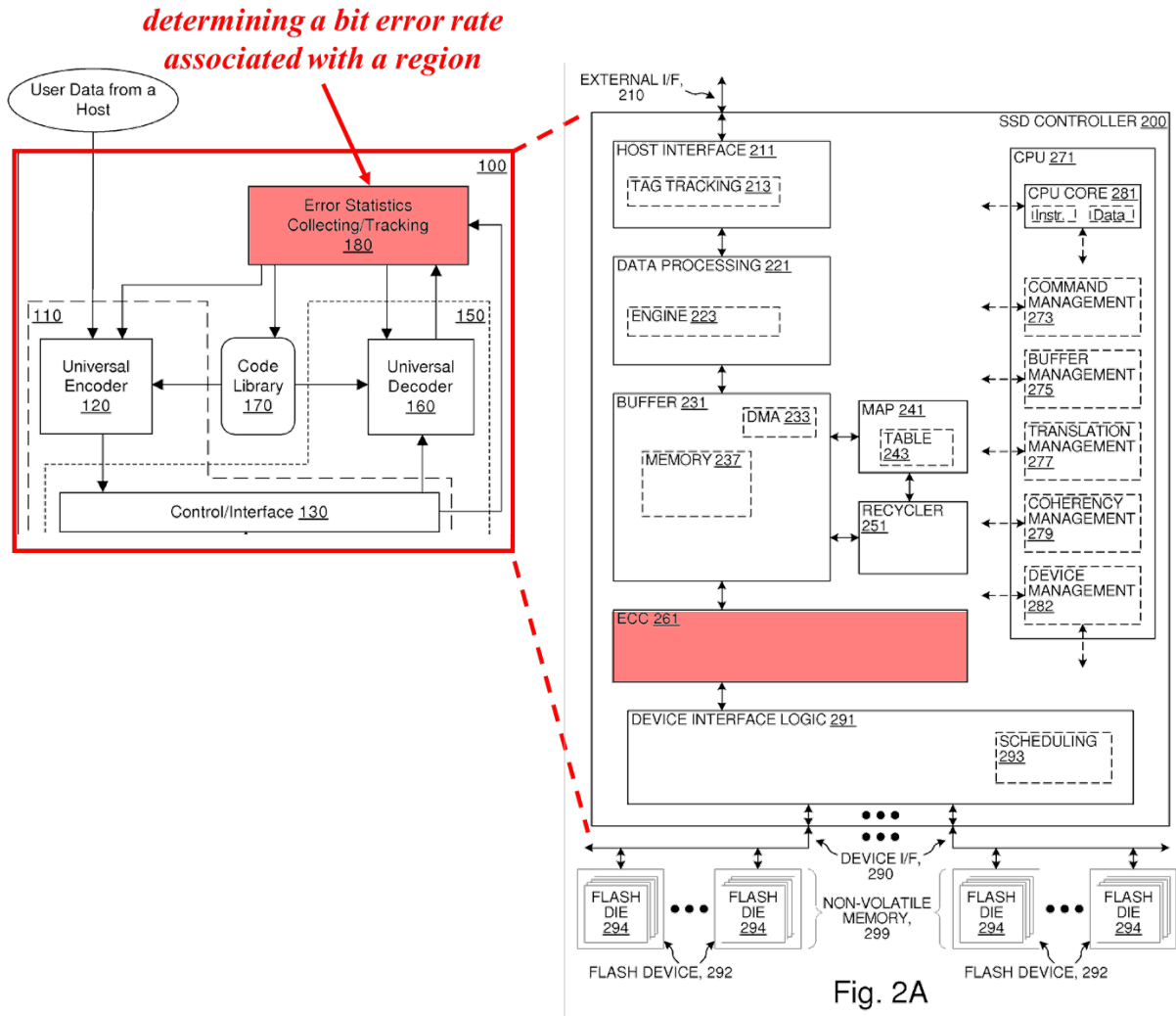
76. Li combined with Cideciyan renders obvious [1.1.1].

a. First, Li teaches “*determining a bit error rate associated with a region.*”

77. As shown in FIGS. 1 and 2A below, Li discloses *determining* (using “an error statistics collecting and tracking hardware logic block”) *a bit error rate* (“raw Bit Error Rate (BER)”) *associated with a region* (adaptive ECC scheme selection performed on a “portion” basis) *comprising a fixed number of two or more flash memory pages* (a portion is multiple pages, and “portions are all a same size”) *or integer fractions thereof*. Ex.1005, Li, [0022] (“an error statistics collecting and tracking hardware logic block enabled to determine **a raw Bit Error Rate (BER)** of accesses to **a portion** of a flash memory”), [0097] (“each of the portions is enabled to store a **pre-determined** amount of information”), [0099] (“a flash memory is operated as portions (such as **pages[]** or **multiples** thereof),

[0038] (“the portion is one or **more pages** of the flash memory”), [0039]

(“determine **respective raw BERs** of accesses to **respective portions** of the flash memory”); *see also* Section V above, explaining the well-known structure of flash based SSDs including pages, blocks, planes, dies, and chips.



Ex.1005, Li, FIGS. 1 (partial) and 2A, annotated.

78. Li describes adaptively selecting an ECC scheme for each “portion” of a plurality “portions” of the memory. In other words, Li’s adaptive ECC scheme

selection is performed on the basis of a particular “portion” (*a region*), which includes monitoring BER of the “portion,” selecting the ECC scheme for the “portion” base the “encoding/decoding with the selected ECC scheme is performed for that “portion” (*the region*). *See, e.g.*, Ex.1005, [0022] (“determine a raw Bit Error Rate (BER) of accesses to **a portion** of a flash memory”); [0047] (“raw BER is a measured raw BER”), [0048] (“the measured raw BER is determined periodically”), [0050] (“wherein the measured raw BER is determined at least in part by observing a BER associated with at least some **reads of the portion.**”), [0051] (“the measured raw BER is determined at least in part by comparing raw read data from the flash memory with an error-corrected version of the raw read data”); claim 15 (adaptive ECC scheme selection and encoding/decoding using the selected ECC scheme are performed for “each of a plurality of **portions** of a flash memory”); *see also id.* [0042]-[0043] (an estimated raw BER is determined based on parameters of the portion),

b. Second, Li’s “portion” associated with the bit error rate determination comprises a fixed number of multiple pages.

79. Li’s “portion” of the flash memory associated with the adaptive ECC scheme selection (including performing the bit error rate determination) “*comprising a fixed number of two or more pages,*” because Li’s number of pages in a portion is fixed by a predetermined amount of information of that portion and the fixed page size. Specifically, Li describes a “portion” of the flash memory that

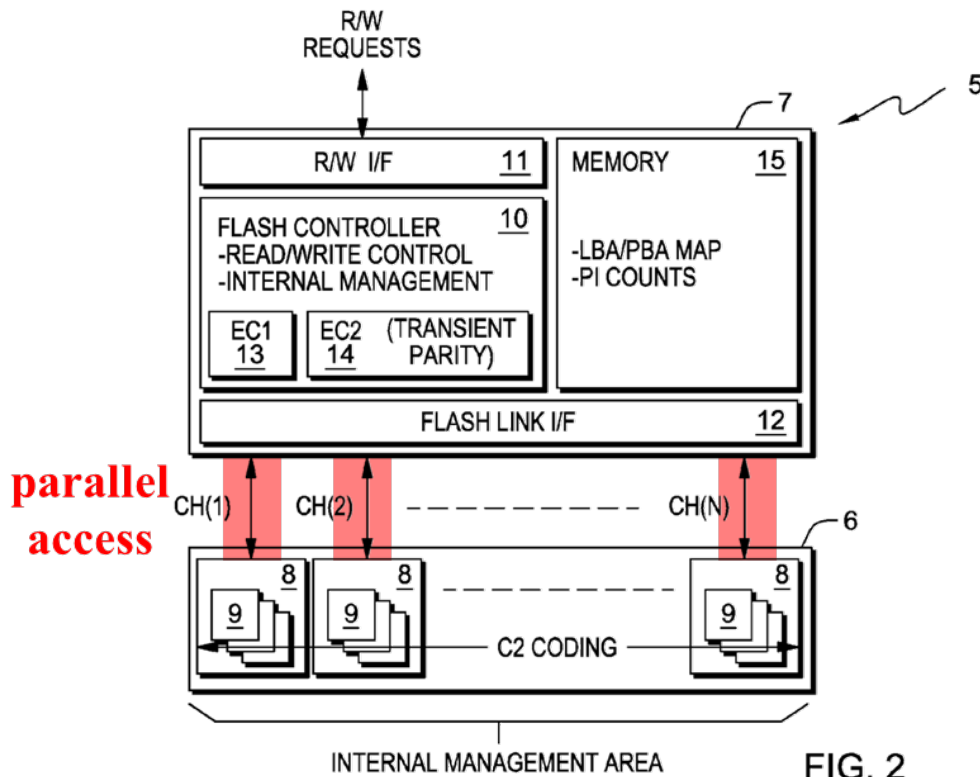
includes multiple pages, and “each of the portions is enabled to store a **pre-**
determined amount of information.” Ex.1005, Li, [0097], [0099]. Li’s “portion”
including a predetermined fixed number of pages, which is fixed by the
“predetermined amount of information” of the portion and the predetermined fixed
size for each page. For example, if a “portion” is implemented to have a
predetermined amount of information of 4K bytes, and each page is 1K bytes, then
the portion has a fixed number of 4 pages. Typically a flash page has a fixed size.

**c. Third, it would be obvious for Li’s portion of multiple pages
to span multiple channels as taught by Cideciyan.**

80. It would be obvious for Li’s portion of multiple pages to span multiple
channels as taught by Cideciyan.

81. As shown in FIG. 2 of Cideciyan below, Cideciyan describes that a
flash based SSD, which includes **N channels** (CH(1) through CH(N)) providing
parallel access to pages of the flash memory chips of different channels.
Specifically, Cideciyan describes “N logical subdivisions corresponding to
respective storage banks 8 on the set of N channels,” which are accessible in
parallel. *Id.*, 10:3-11; 11: 44-46 (pages allocated on different channels “can be
written in parallel to the N channels”). It was well-known that channels in SSDs
are assessable in parallel (i.e., simultaneously), which was well-known for
achieving optimal performance for both read and write operations in SSDs, as
shown in supporting references. *See, e.g.*, Ex.1012, Hu, FIG. 4, 98 (“in Figure 4, if

a request is served by channel 0 and channel 1 simultaneously, it exploits the channel-level parallelism.”), FIG. 11, 102 (describing for “a given read request whose size is a multiple of a logical page size,” channel-level parallelism is used for “decreasing the response time of this request.”); Ex.1018, SSD 101, 18 (“Channels refer to the number of flash chips the controller can talk to simultaneously.”).



Ex.1006, Cideciyan, FIG. 2, annotated.

82. Cideciyan describes that in its flash based SSD, applying a product code type ECC scheme on the basis of “groups” of pages, including the encoding

and decoding of data performed on the basis of the “groups” of pages. As will be discussed in detail at [1.1.2] below, Cideciyan’s encoding process is performed for input data for each group, and the encoded symbols (including data symbols and parity symbols) will be written into a single group of pages distributed across the channels. Similarly, Cideciyan’s decoding process is performed on data read from each group.

83. Cideciyan teaches that for its “group” of pages, “the **pages of each group** are **distributed across** all [N] logical subdivisions.” Ex.1006, Cideciyan, 10:3-11. Cideciyan uses the term “stride” to refer to a group of pages distributed across the channels. That is, a stride includes the pages accessible by channel CH(1), the pages accessible by channel CH(2), ... and the pages accessible by channel CH(N). *Id.*, 10:11-12. As shown in FIG. 5 below, a POSITA would understand that the group of pages (“page stripe”) as shown in FIG. 5 teaches *a region comprising a **fixed** number (N) of two or more flash memory pages* as claimed. In the example of FIG. 5, the stride includes one page for each of the N channels. Ex.1006, Cideciyan, 11:9-18 (“a stride contains **one page in each of the N channels.**”). Therefore, the *fixed number of pages* is N, which is the same number as the channels. Further still and as shown in FIG. 5, there are at least 4 channels, therefore the stride in this figure includes at least 4 pages, which teaches *a fixed number of **two or more** flash memory pages.*

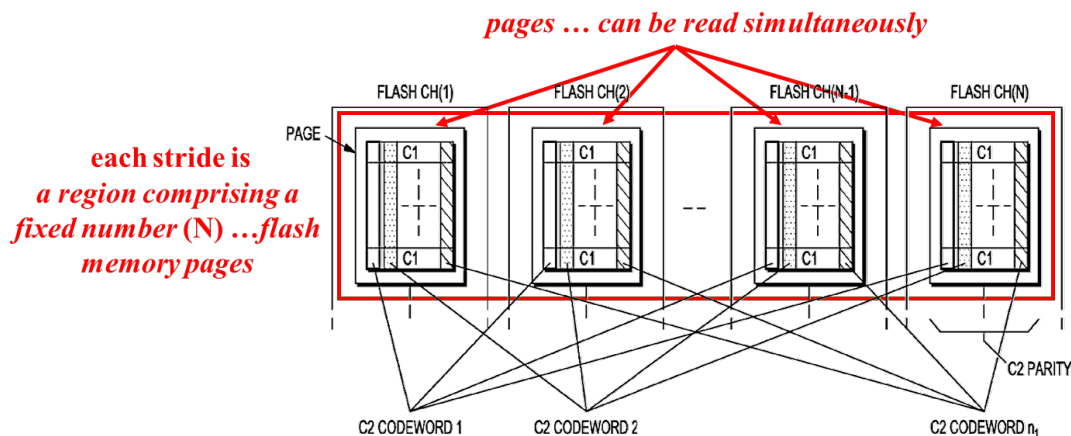


FIG. 5

Ex.1006, Cideciyan, FIG. 5, annotated.

84. Thus, it would be obvious for Li's portion of multiple pages to span multiple channels as taught by Cideciyan, for various benefits of parallel access, and additional benefits of implementing the product code ECC scheme on the group basis (e.g., "efficient, reliable and robust solid state storage systems with exceptional overall performance" with orthogonal primary and secondary codewords, with "improved error protection offered by the coding process **both within storage subdivisions** and **across the storage as a whole**." Ex.1006, Cideciyan, 5:2-13), which will be discussed in detail in [1.1.2].

d. Fourth, it would be obvious for multiple pages spanning multiple channels to be read simultaneously.

85. It would be obvious for multiple pages spanning multiple channels to be read simultaneously. In the group of pages as shown in FIG. 5, each of the N pages can be accessed in parallel through the N channels, and a POSITA would

understand that accessing the pages in parallel includes performing *read* operations *simultaneously* using the N channels. Ex.1012, Hu, FIG. 4, 98 (“in Figure 4, if a **request is served by channel 0 and channel 1 simultaneously**, it exploits the channel-level parallelism.”), FIG. 11, 102 (describing for “**a given read request** whose size is a multiple of a logical page size,” channel-level parallelism is used for “decreasing the response time of this request.”); Ex.1018, SSD 101, 18 (“Channels refer to the number of flash chips the controller can talk to **simultaneously**.”).

86. To the extent that it is argued that parallel access is not necessarily simultaneous, that argument is addressed at Ground 2 below.

Reasons to Combine Li and Cideciyan

87. It would have been obvious to a POSITA to combine Li and Cideciyan to provide the obvious, predictable, and beneficial result of Cideciyan memory arrangement and accessibility. As will be explained in more detail below with respect to [1.1.2], it would have been obvious to implement Li’s ECC technique using the product code type ECC scheme applied on groups of pages as taught by Cideciyan. Using that technique, it would have been obvious that the “portions” for ECC and bit error rate monitoring of Li would span multiple channels that are accessible simultaneously and in parallel.

88. **First**, the implementation with Cideciyan’s technique of groups of

pages (“strides”) distributed across channels has the advantage that the fixed number of multiple pages in a “stride” are distributed across the channels of an SSD, utilizing the channel-level parallelism in SSD to gain better parallel performance, which is important for overall SSD performance. Ex.1012, Hu, 105 (explaining the well-known channel-level parallelism in SSDs which is important for SSD performance); Ex.1013, Shin, 1 (describing various well-known technologies of using flash channels in an SSD for achieving desirable parallel read and write access, e.g., “employ independent channels per flash chips to gain maximum parallelism.”). Benefits of parallelism include lower latency, better data protection by spreading data across multiple dies if one die has higher failures. *See, e.g.*, Ex.1006, Cideciyan, 5:2-6 (“protection against failures of individual storage subdivisions in the system” by error correcting coding process across the channels); Ex.1007, Frost, 11:57-60 (parallel access through channels allowing “relatively quick read and write operations and allows data to be stored to and retrieved from the memory array 14 in an efficient and effective manner.”).

89. Second, Cideciyan provides implementation details of the “portions” and the parallel accessibility across a plurality of flash dies described in Li. Therefore, it would be obvious to implement Li’s teaching of organizing the flash memory (including a plurality of flash dies assessable in parallel) in portions, with the organization of the flash memory in groups of pages (“strides”) across channels

in an SSD as taught by Cideciyan.

90. As discussed at Ground 1:[1.1.1].a-b, Li describes organizing the flash memory (including a plurality of dies accessible in parallel), in a plurality of “**portions**” such that each “portion” includes a fixed number of multiple pages. Li describes applying ECC schemes and monitoring the bit error rate of “portions” generally. A POSITA would have found it obvious to look to known techniques for implementing those portions.

91. Cideciyan is such a reference. The specific implementation in Cideciyan’s FIG. 5 includes a stride including N pages across N channels, with the advantage that each page is accessible in parallel through the N channels, thereby achieving channel-level parallelism among these pages in the stride, providing a benefit for a POSITA to implement this teaching. It was well-known in the art that channel-level parallelism is important for achieving peak performance in SSD. Ex.1012, Hu, 2 (Parallelism, including channel-level parallelism, is important for SSD performance and “has been regarded as a key to achieving the peak performance.”).

92. Additionally, Cideciyan is a patent application by International Business Machines (IBM) Corporation, which was a top storage provider during the timeframe of the earliest priority date of the ’085 patent. A POSITA with Li in hand would have looked to patents and publications like Cideciyan from IBM to

improve Li's flash based SSD.

93. Indeed, I cited works from inventors of Cideciyan in several of my works on LDPC codes for data storage. Here are two examples:

94. T. Mittelholzer, "Efficient Encoding and Minimum Distance Bounds of Reed-Solomon-Type Array Codes," in Proc. *IEEE International Symposium on Information Theory*, Lausanne, Switzerland, Jul. 2002, p. 282. I cited this paper in my paper L. Dolecek et al. "Analysis of Absorbing Sets and Fully Absorbing Sets of Array-Based LDPC Codes." *IEEE Transactions on Information Theory* vol. 56, no. 1, Jan. 2010, pp. 181–201.

95. E. Eleftheriou and S. Olcer. "Low Density Parity Check Codes for Digital Subscriber Lines," In Proc. *IEEE Int. Conf. on Comm.*, New York City, NY, April – May 2002. I cited this work in my paper: L. Dolecek et al. "Predicting Error Floors of Structured LDPC Codes: Deterministic Bounds and Estimates," in *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 908-917, August 2009.

96. Third, as shown in FIG. 2A of Li and FIG. 2 of Cideciyan below, Li and Cideciyan are similar devices with similar objectives. Ex.1005, Li, FIG. 2A (illustrating a flash memory based storage system with SSD controller and flash devices accessible in parallel channels), [0019] (aiming for "improvements in flash memory lifetime, reliability, performance, and/or storage capacity"); Ex.1006,

Cideciyan, FIG. 2 (illustrating a flash memory based storage system with a flash controller and flash devices accessible in parallel channels), 5:11-13 (to “provide efficient, reliable and robust solid state storage systems with exceptional overall performance”).

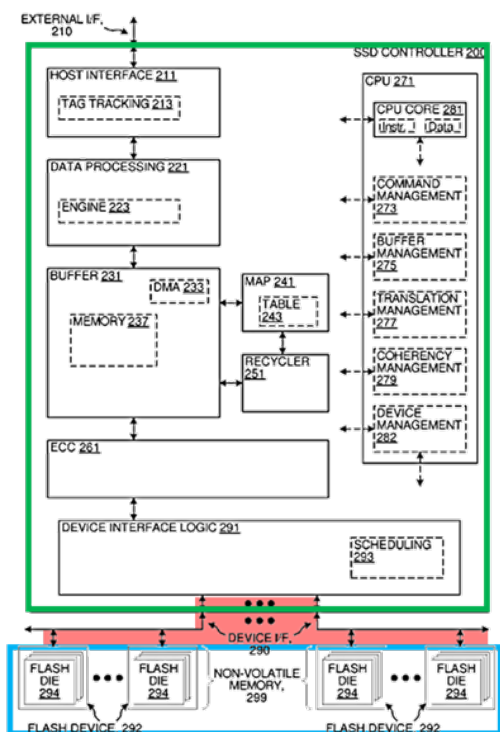


Fig. 2A

Ex.1005, Li, FIG. 2A, annotated

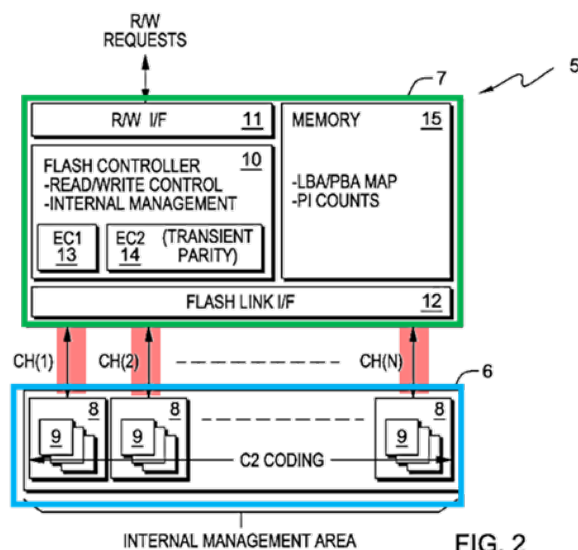


FIG. 2

Ex.1006, Cideciyan, FIG. 2,
annotated

97. A POSITA would recognize that a known technique (using strides, where each stride including N pages distributed across N channels) has been used to improve one device (the flash based storage system of Cideciyan), it would improve similar devices (Li's flash memory based storage system) in the same way (with the benefits as taught by Cideciyan), using the prior art elements (using

group of pages, where each group including N pages distributed across N channels) according to their established functions.

98. Additionally, the results of the proposed combination would have been predictable and a POSITA would have had a reasonable expectation of success, because the combination of Li and Cideciyan merely applies known technique to a known device (Li's flash memory based storage system implementing adaptive ECC schemes) ready for improvement to yield the predictable results of "efficient, reliable and robust solid state storage systems with exceptional overall performance." Ex.1006, Cideciyan, 5:11-13.

99. As shown in Example 1.1 of Li combined with Cideciyan (based on FIGS. 1, 2A of Li and FIG. 5 of Cideciyan) illustrated below, a POSITA would have reasonably expected to succeed in combining Li with Cideciyan, because Cideciyan's group-based ECC encoding/decoding is described as applicable to flash memory devices such as those taught by Li. See, e.g., Ex.1006, Cideciyan, FIG. 2. Additionally, the implementation of Cideciyan's teaching would have required only routine knowledge well within the skill of a POSITA prior to the earliest effective filing date of the '085 patent. See, e.g. Ex.1006, Cideciyan, 11:55-57 ("Corresponding decoder implementations will be readily apparent to those skilled in the art from the encoder description"); 17:13-17 (describing that considerations related to grouping pages "are similar to those encountered in RAID

applications as will be apparent to those skilled in the art.”).

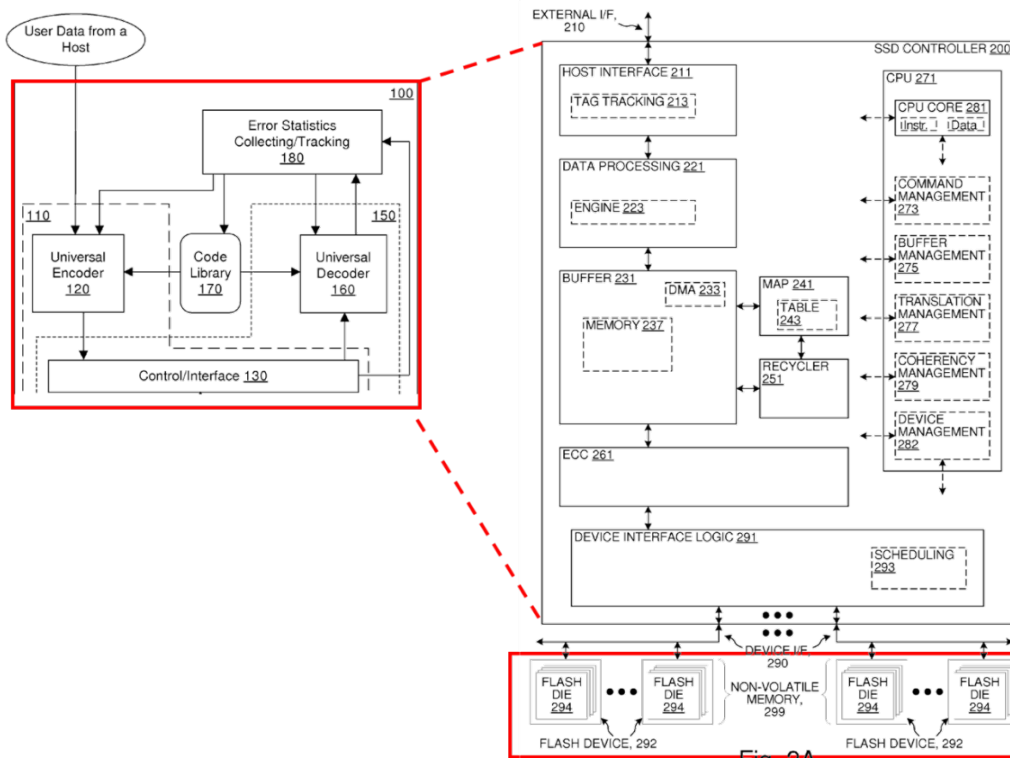
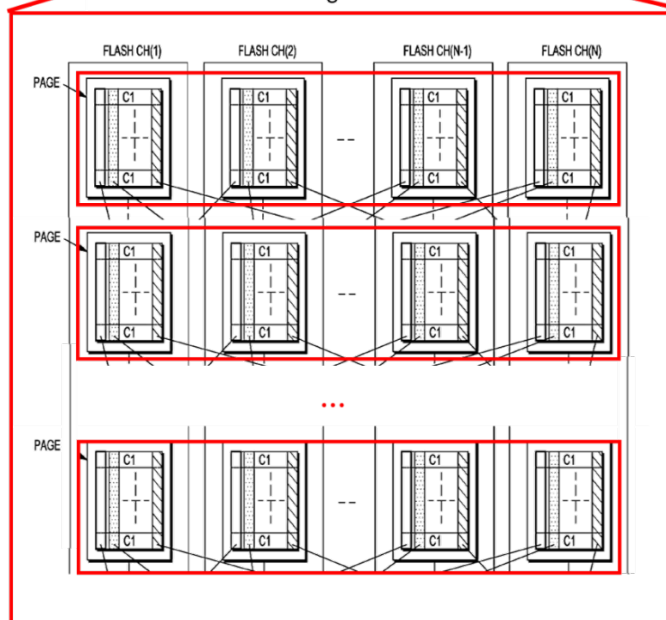


Fig. 2A

region

region

region



Example 1.1 of Combination of Li and Cideciyan.

[1.1.2] *wherein the region stores at least data payload and primary and secondary ECC parity symbols corresponding to the data payload of the region;*

100. The combination of Li and Cideciyan renders obvious [1.1.2].

101. Li teaches that the portion of a flash memory (*the region*) stores user data and a number of error correcting bits (*stores at least data payload and ECC parity symbols corresponding to the data payload of the region*). For example, claims 1-3 of Li recite:

1. A system, comprising:

means for error statistics collecting and tracking enabled to dynamically determine a raw Bit Error Rate (BER) of accesses to a **portion** of a flash memory; and

means for adaptive encoding enabled to encode according to a dynamically selected one of a plurality of error correcting codes, and further enabled to dynamically determine the dynamically selected error correcting code based at least in part on the raw BER.

2. The system of claim 1, wherein encoding according to a first one of the error correcting codes results in a number of **error correcting bits** to **store in the portion** that is less than when encoding according to a second one of the error correcting codes.

3. The system of claim 2, wherein when encoding according to the first error correcting code, a number of bits of the portion used as user data is increased by up to a difference between the number of error correcting bits used by the second error correcting code minus the number of error correcting bits used by the first error correcting code.

See also Ex.1005, Li, [0021]-[0025], claim 15; [0099] (“each portion is organized as a data sub-portion and a respective corresponding error correcting sub-portion”), [0060] (“the portion comprises a plurality of sub-portions, and the adaptive encoder is further enabled to encode such that error correcting information is storable to one or more of the sub-portions and data information is storable to only one of the sub-portions”).

102. It would have been obvious to implement Li’s ECC technique using one of the known, available ECC techniques such as Cideciyan’s product code type ECC scheme. As discussed below, Cideciyan describes implementing a product code ECC scheme on a group basis, where each group spans across the channels, which provides implementation detail of an ECC scheme and associated “portions” that would be obvious to use in Li.

Generation of the primary and secondary ECC parity symbols:

103. Cideciyan describes **generating** the primary codewords (including primary parity symbols) and secondary codewords (including secondary ECC

parity symbols) using a product code type ECC scheme as shown in FIG. 4. To be clear, FIG. 4 is merely a logical representation of the well-known product code scheme (see, e.g., Ex.1020, Lin, FIG. 9.6; Ex.1019, Yang, FIG. 5), which illustrates the encoded symbols (including data payload, primary parity symbols, and secondary parity symbols) using the product code (with component codes C1 and C2) **to be stored in a single physical region**. FIG. 4 does not display physical pages of a flash memory. However, FIG. 5 which is discussed in detail below, is a **physical** representation of a **single physical region** (a group of N pages from across N channels (“stride”)) storing the array of encoded symbols of FIG. 4.

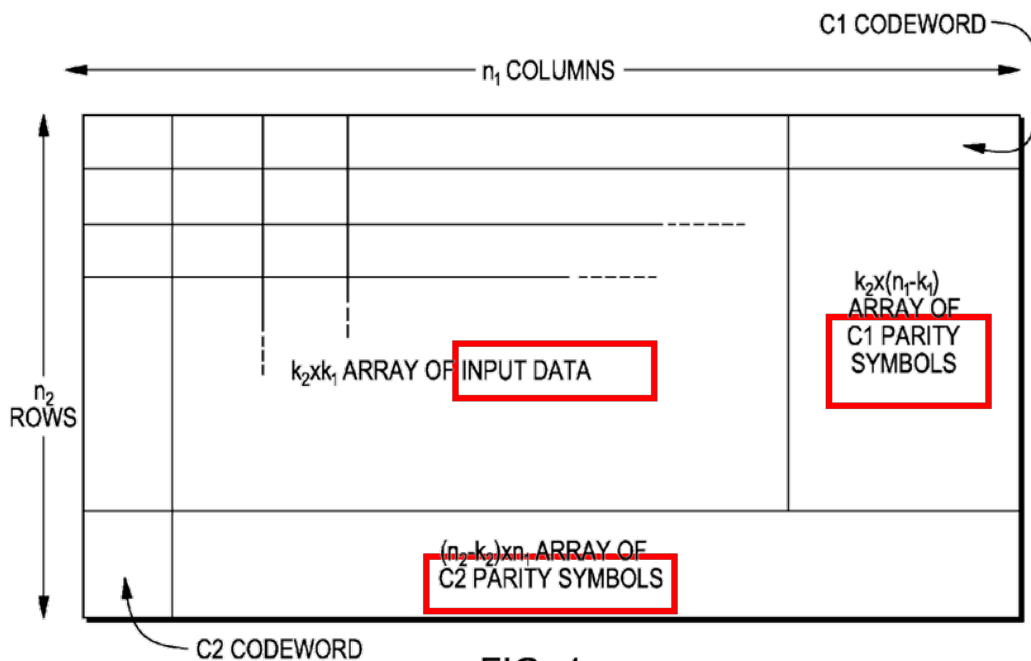


FIG. 4

Ex.1006, Cideciyan, FIG. 4, annotated.

104. Referring to the logical representation of the product code scheme

shown in FIG. 4, the encoded symbols from the product code include input data symbols (*data payload symbols*), C1 parity symbols (*primary ECC parity symbols*), and C1 parity symbols (*secondary ECC parity symbols* generated during the C1 and C2 encoding processes. Ex.1006, Cideciyan, 10:54-11:8. “The array of encoded symbols shown in the FIG. 4 is written **in a single stride**”. *Id.*, 11:9-10.

105. In further detail, Cideciyan describes “systematic encoding” is used, “whereby each codeword is produced by addition of parity symbols to the unencoded data and the data itself remains unchanged by the encoding process.” Ex.1006, Cideciyan, 10:53-56. Cideciyan describes in detail the C1 and C2 encoding process:

The input data to be encoded is partitioned by controller 10 into successive logical blocks each comprising a logical array of k_2 rows and k_1 columns of data symbols as shown in the figure. The EC1 encoder 13 systematically encodes each row in accordance with the C1 code to **generate $(n_1 - k_1)$ parity symbols** for that row. This **C1 parity code** is added to the **k_1 input data symbols** to produce an **n_1 -symbol C1 codeword**. When all k_2 rows have been encoded by the EC1 encoder, the resulting set of k_2 C1 codewords forms a logical array of k_2 by n_1 C1-encoded symbols. The C2 encoding performed by EC2 encoder 14 operates on columns of this array. The EC2 encoder 14 systematically encodes each column in accordance with the C2

code to generate **(n_2-k_2) parity symbols** for that column. This **C2 parity code** is added to the **k_2 column symbols** to produce an **n_2 -symbol C2 codeword**. When all n_1 columns have been encoded by the EC2 encoder, the resulting set of n_1 C2 codewords forms a logical (n_2 by n_1)-symbol array as shown in the figure.

Ex.1006, Cideciyan, 10:53-11:8.

Storage of the primary and secondary ECC parity symbols:

106. Referring to FIG. 5 below, Cideciyan further describes storing the array of encoded symbols of FIG. 4 in the single group of pages (“stride”).

Ex.1006, Cideciyan, 11:9-11 (“The array of encoded symbols shown in the FIG. 4 is written **in a single stride**”). Specifically, Cideciyan describes that *the region* (a group of pages (“stride”) including N pages across N channels) storing C1 codewords and C2 codewords. The C1 codewords include input data symbols (*data payload*) and C1 parity symbols (*primary ECC parity symbols*), and the C2 codewords include input data symbols (*data payload*) and C2 parity symbols (*secondary ECC parity symbols*). It is noted that in FIG. 5, the C2 parity symbols are stored in the N^{th} page (corresponding to $CH(N)$) in this group (“stride”) as an example only, and “**any channel** could be used for the parity.” Ex.1006, Cideciyan, 11:39-42).

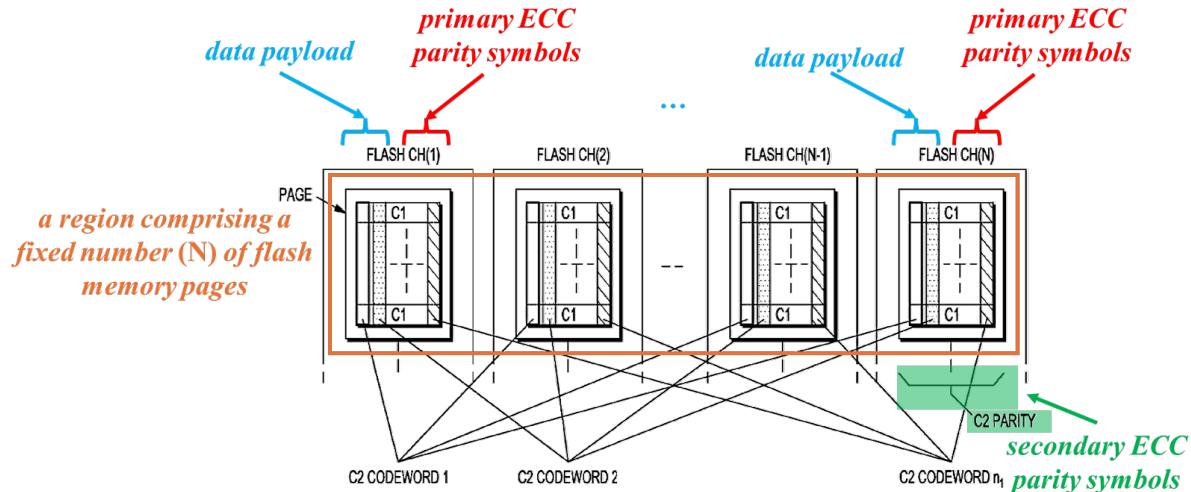


FIG. 5

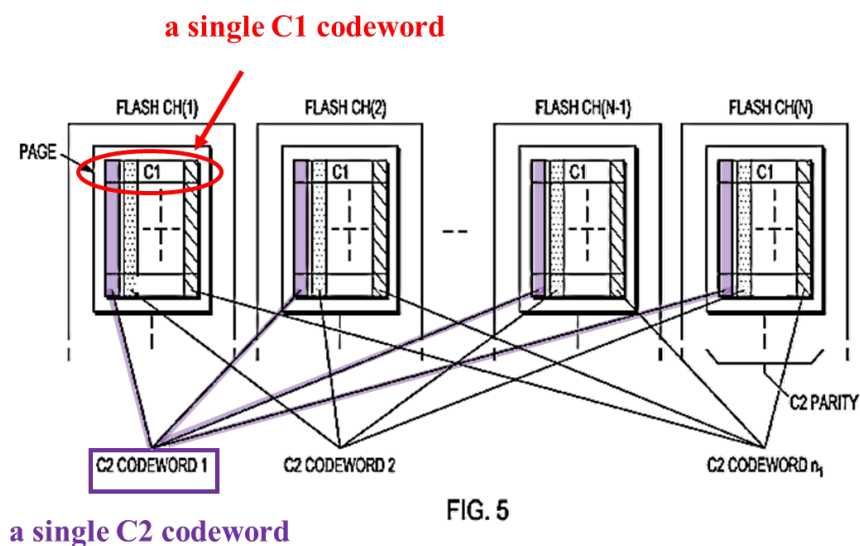
Ex.1006, Cideciyan, FIG. 5, annotated.

107. In further detail, referring to FIG. 5 above, Cideciyan describes:

The array of encoded symbols shown in the FIG. 4 is written in a **single stride** as described above with reference to FIG. 3....This configuration is illustrated by the schematic diagram of FIG. 5. In the example shown, it is assumed for simplicity that a **stride contains one page in each of the N channels and each page stores a plurality of C1 codewords**. The key features of the codeword arrangement here are that: (1) **each storage subdivision spanned by the stride contains a plurality of C1 codewords**; and (2) **each of these storage subdivisions contains a portion of each of the second codewords** in the stride. In the stride shown in FIG. 5, different C2 codewords are indicated by differently-shaded vertical bars. The first channel contains the first portion of each of the n_1 C2 codewords. The

second channel contains the second portion of each of the C2 codewords, and so on. The last channel, channel N, contains the C2 parity codes for each C2 codeword **in this example**.

Ex.1006, Cideciyan, 11:9-28. In annotated FIG. 5 below, a single C1 (primary) codeword is illustrated in red, and a single C2 (secondary) codeword is annotated in purple.



Ex.1006, Cideciyan, FIG. 5, annotated.

108. Like the '085 patent, Cideciyan's C1 (primary) codewords each are stored within a page and provides protection within a page. Further like the '085 patent, Cideciyan's C2 (secondary) codewords each is stored across the group ("stride") of N pages across the N channels, providing protection within a page stripe, because the example stride of FIG. 5 is a page stripe. Ex.1001, '085 Patent,

Table 1 below (describing the data protection of “Primary” and “Secondary”).

TABLE I

Type of Error Correction	Data Protection
Primary	within a page or integer fraction of a page
Secondary	within a page stripe
Tertiary	within a page grid

Ex.1001, '085 patent, Table 1.

109. Additionally, like the '085 patent, Cideciyan's C1 (primary) and C2 (secondary) codewords disclosure provides **orthogonal** primary and secondary ECC codewords. Ex.1001, '085 Patent, 3:20-25 (“the terms primary, secondary, and tertiary refer to orthogonal dimensions over which block data is taken for calculation of ECC parity bits.”). Ex.1006, Cideciyan, FIG. 4, 11:29-31 (“[t]he ‘**orthogonal**’ arrangement of C1 and C2 codewords...**gives exceptional error performance.**”).

110. Further like the '085 patent, Cideciyan also describes that various code types may be used for C1 code and C2 code. Ex.1006, Cideciyan, 18:31-34 (“the C1 code can be any linear code e.g. a binary BCH code or a q-ary BCH code,” and “while a RS C2 code is preferred, the C2 code can in general be any linear, and preferably also cyclic, code.”)

Reasons to Combine Li and Cideciyan

111. It would have been obvious to a POSITA to combine Li and Cideciyan to provide the obvious, predictable, and beneficial result from implementing ECC schemes using the product code type ECC scheme as taught by Cideciyan, which generates orthogonal primary and secondary codewords and provides improved error recovery.

112. First, a POSITA would have been motivated to combine Li with Cideciyan's teaching of using a product code type ECC scheme generating orthogonal primary and secondary codewords that provides improved error recovery in Li, because Li describes adaptive ECC selection using "a set of ECC schemes with **various code types.**" Ex.1005, Li, [0019]. Cideciyan teaches an obvious way of implementing Li's ECC scheme, by providing a product code type ECC scheme (using the well-known "product code" code type invented in the 1950s and described in detail in textbooks Ex.1020, Lin, 274-278), which includes component codes C1 and C2 and generates highly orthogonal C1 codewords and C2 codewords. **"Compared to plain long length codes,"** such product code **"has high performance from cross parity check, and low circuitry overhead."** Ex.1018, Yang, 4; Ex.1006, Cideciyan, FIG. 4, 11:29-31 ("the 'orthogonal' arrangement of C1 and C2 codewords ...**gives exceptional error performance.**"). A POSITA would have found that such implementation of applying a product code ECC scheme to the region provides the benefit of "high performance from cross

parity check” with orthogonal arrangement of C1 and C2 codewords, provides low circuit overhead by using two shorter length component codes, “improved error recovery,” (Ex.1006, Cideciyan, 3:54-4:56), and is a suitable option for Li’s implementation of ECC schemes with various code types.

113. Second, implementing Li’s portions to span different channels for parallel, simultaneous access provides the additional benefits of “efficient, reliable and robust solid state storage systems with exceptional overall performance,” and “improved error protection offered by the coding process **both within storage subdivisions** and **across the storage as a whole**.” Ex.1006, Cideciyan, 5:2-13; *see also* 7:30-41 (describing its invention provides “improved error protection... achieved **without the performance penalties**, in terms of increased write amplification and reduced endurance”).

114. In other words, in the example of Li combined with Cideciyan, improved protection can be achieved across multiple flash memory subdivisions, e.g., using information on other dies to reconstruct information on a failed die, as taught by Cideciyan, which Li alone does not describe.

115. Third, as discussed above at Ground 1:[1.1.1].e and as shown in FIG. 2A of Li and FIG. 2 of Cideciyan below, Li and Cideciyan are similar flash based SSD devices (each including a SSD controller and flash devices accessible in parallel) with similar objectives. Ex.1005, Li, FIG. 2A, [0019], Ex.1006,

Cideciyan, FIG. 2, 5:11-13.

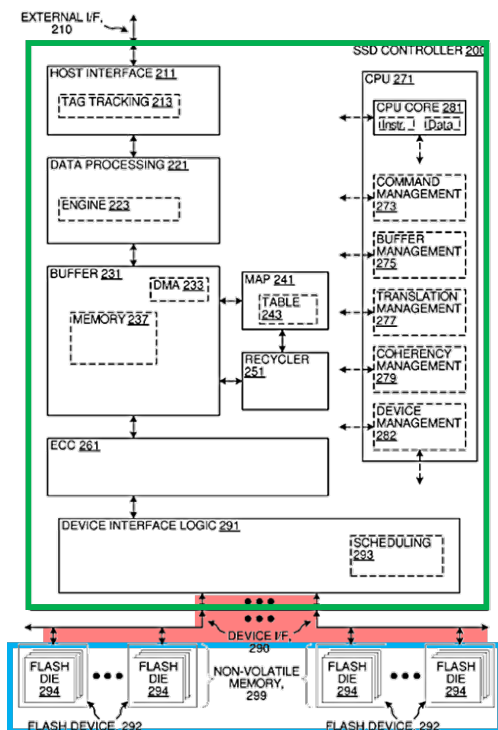


Fig. 2A

Ex.1005, Li, FIG. 2A, annotated

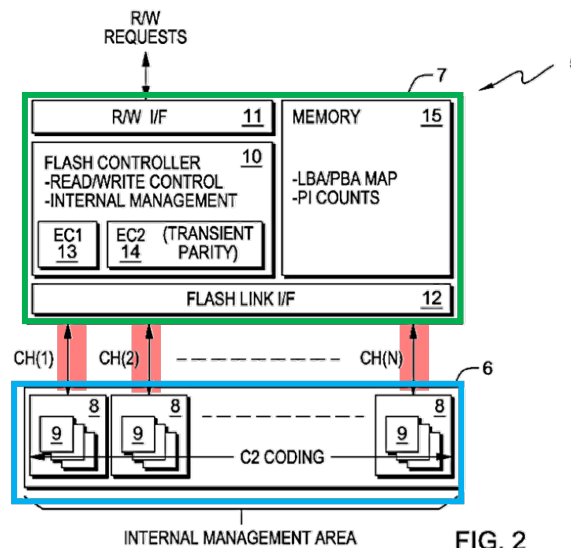


FIG. 2

Ex.1006, Cideciyan, FIG. 2,
annotated

116. A POSITA would have found it obvious for a known technique (applying a product code ECC scheme to a group of pages across the channels) improves one device (the flash based storage system of Cideciyan) to improve similar devices (Li's flash memory based storage system) in the same way (with the benefits as taught by Cideciyan), using the prior art elements according to their established functions.

117. **Fourth**, yet another reason to combine is that the combination of Li and Cideciyan allows the pages of the memory to be operated in multi-level cell

(MLC) mode for a longer period of time than Li alone, such that the switching from MLC to SLC is delayed or avoided. Li describes that “when a raw BER exceeds a threshold, a portion (such as a page) of flash memory previously operated as an MLC page is thereafter operated as an SLC [single level cell].” Ex.1005, Li, [0103]. Switching from MLC to SLC is at expense of a big reduction of data payload – for example, a reduction around 50% by switching from a 2-bit-level MLC mode to an SLC mode. This switching should be avoided or delayed as much as possible. By applying Cideciyan’s product code ECC scheme where the “the ‘**orthogonal**’ arrangement of C1 and C2 codewords ...**gives exceptional error performance,**” (Ex.1006, Cideciyan, FIG. 4, 11:29-31), stronger error recovery is achieved for deteriorated MLC cells, thereby the MLC’s switch to SLC mode is delayed. *See also* Ex.1006, Cideciyan, 3:54-4:4 (“MLC is inherently less reliable than single-level memory cells (SLC).... **Additional coding is therefore required to extend MLC NAND lifetime**”).

118. Additionally, the results of the proposed combination would have been predictable and a POSITA would have had a reasonable expectation of success, because the combination of Li and Cideciyan merely applies known technique to a known device (Li’s flash memory based storage system implementing adaptive ECC schemes) ready for improvement to yield the predictable results of “efficient, reliable and robust solid state storage systems with

exceptional overall performance.” Ex.1006, Cideciyan, 5:11-13.

[1.2.1] *comparing the determined bit error rate to one or more predetermined thresholds corresponding to a set of predefined gears comprising at least a first gear and a second gear, wherein the predefined gears correspond to different predefined ECC schemes,*

119. Li, as combined above with Cideciyan, renders obvious [1.2.1]. The term “gear” is discussed above in the Claim Construction section, which means “a set of parameter(s) specifying an error correction coding scheme, including one or more of the payload capacity, the type and parameters of each component code, the interleaving scheme, and parameters used for determining whether the gear is appropriate for a particular situation.” In the claims, a gear corresponds to an ECC scheme. *See* limitation [1.2.1].

120. I will first discuss the term “*a set of predefined gears*” below.

a. “*a set of predefined gears... the predefined gears correspond to different predefined ECC schemes*”

121. First, Li as combined above with Cideciyan renders obvious “*a set of predefined gears,*” where each predefined gear corresponds to a predefined “ECC scheme” as described in Li. Specifically, Li describes,

The techniques include **a set of ECC schemes** with **various code types,**
code rates, and/or various code lengths (providing different error correcting capabilities), and error statistic collecting/tracking (such as via a dedicated hardware logic block)...The techniques further include

encoding/decoding in accordance with **one or more of the ECC schemes**,
and **dynamically switching** encoding/decoding of all or any portions of the
flash memory **amongst a respective one or more of the ECC schemes**
based at least in part on information from the error statistic
collecting/tracking.”

Ex.1005, Li, [0019].

122. As such, Li as combined with Cideciyan describes a predefined set of ***different predefined ECC schemes*** having different error correcting capabilities, where each ECC scheme is specified by a set of parameters specifying its “code type,” “code rate,” and “code length.” Ex.1005, Li, [0091] (“The error correcting code used for encoding (and decoding) is selected from **a set of error correcting codes**....In various embodiments, the set includes more than one **type of code**, such as various combinations of RS, BCH, trellis, and/or LDPC code types, and each of the code types includes one or more specific codes of the respective type. In various embodiments, the set includes codes of **varying rates** and/or **lengths**.”).

123. This set of parameters specifying an ECC scheme, together with parameters (including the corresponding thresholds for comparing the raw BER) used for determining whether this ECC scheme is appropriate for a particular situation (as discussed below at [1.2.1.b]) is a “***gear***” corresponding to that ECC scheme.

124. In Li combined with Cideciyan, as discussed at [1.1.2], the predefined set of ECC schemes includes “**product code** type” ECC schemes as taught by Cideciyan, each is specified with a set of parameters specifying a code type of “product code,” component code C1 and its corresponding parameters, component code C2 and its corresponding parameters, a code rate determined by C1 and C2, and a code length determined by C1 and C2, as explained at V.B.4; *see also* Ex.1020, Lin, xv (“The book can be used as a text for an introductory course on error-correcting codes”), 274-278; Ex.1019, Yang, 4. The set of parameters specifying this product code ECC scheme, together with parameters (including the corresponding thresholds for comparing the raw BER) used for determining whether this product code ECC scheme is appropriate for a particular situation (as discussed below at [1.2.1.b]) is a “**gear**” corresponding to that ECC scheme.

125. Below are example “gears” in “*a set of predefined gears*” in Li combined with Cideciyan, each gear including a set of parameters specifying the corresponding ECC scheme, and the set of parameters for determining whether the gear is appropriate for a particular situation. *See e.g.*, Ex.1005, Li, Abstract, [0019], [0091], [0035], [0036] (Li describing multiple set of parameters with “various code types, code rates, and/or various code lengths” providing different error correcting capabilities); *see also* Ex.1006, Cideciyan, 18:31-34 (“the C1 code can be **any linear** code e.g. a binary BCH code or a q-ary BCH code as well as an

RS code. Similarly, while a RS C2 code is preferred, the C2 code can in general be **any linear**, and preferably also cyclic, code.”), FIG. 11 (illustrating different sets of parameters specifying ECC schemes having different ECC strengths for output BER requirements of “1e-11,” “1e-15,” and “1e-19”), 16:34-60 (further detail of FIG. 11, using product code with RS-based C1 and C2 codes), FIG. 12, 16:61-17:12 (further detail of FIG. 12, providing example product code with BCH-based C1 code and RS-based C2 code); Ex.1009, Yang, 5 (describing various component code types in product code); *see also* ¶¶35-40. Note these are merely examples, and various other choices are available. Ex.1006, Cideciyan, 18:31-34 (“the C1 code can be **any linear** code e.g. a binary BCH code or a q-ary BCH code as well as an RS code. Similarly, while a RS C2 code is preferred, the C2 code can in general be **any linear**, and preferably also cyclic, code.”). Each of BCH code, Hamming code, and RS code used in example component codes below is a linear code. *See also* ¶¶32-34.

<i>parameters specifying an ECC scheme</i>		<i>ECC scheme</i>	<i>parameters used for determining whether the gear is appropriate</i>
Set 1 (Example Gear 1)	code type: Product Code with c1:BCH, c2: Hamming code; code-rate-1; code-length-1; component-code-c1: c1-code-rate-1, c1-code-length-1;	ECC scheme 1	BER-threshold- x1 < Determined BER < BER- threshold-y1

	component-code-c2: c2- code-rate-1; c2-code-length-1; etc.		
Set 2 (Example Gear 2)	code type: Product Code with c1:BCH, c2:RS; code-rate-2; code-length-2; component-code-c1; c1-code-rate-2; c1-code-length-2; component-code-c2: c2-code-rate-2; c2-code-length-2; etc.	ECC scheme 2	BER-threshold- x2 < Determined BER < BER- threshold-y2

Dolecek, Table 1

126. In Li combined with Cideciyan, the set of predetermined gears includes “***a first gear***” (e.g., Example Gear 1) and “***a second gear***” (e.g., Example Gear 2), each corresponding to a set of parameters specifying an ECC scheme and the parameters for determining whether the gear is appropriate for a particular situation.

127. For further detail, Li provides, “2. The system of claim 1, wherein encoding according to **a first one** of the error correcting codes results in a number of error correcting bits to store in the portion that is less than when encoding according to **a second one** of the error correcting codes.” Ex.1005, Li, Claims 1-2.

128. According to Li’s description in claims 1-2, in Li combined with Cideciyan, a set of predefined ECC schemes include a first ECC scheme with “**a first**” error correcting code (of any suitable code type, code rate and code length), e.g., Example Gear 1, thereby corresponds to “***a first gear***.” *See, also e.g.,*

Ex.1005, [0035] (“wherein at least two of the error correcting codes are of different code rates”); [0036] (“wherein at least two of the error correcting codes are of different code lengths). A raw BER is compared to one or more BER thresholds associated with the first gear to determine whether this first gear is appropriate for a particular situation, as discussed in detail at [1.2.1].b below.

129. Similarly, the set of predefined ECC schemes include a second ECC scheme with “**a second**” error correcting code (of any suitable code type, code rate and code length), e.g., Example Gear 2, thereby corresponds to “*a second gear.*” A raw BER is compared to one or more BER thresholds associated with the second gear to determine whether this second gear is appropriate for a particular situation, as discussed in detail at [1.2.1].b below.

b. “comparing the determined bit error rate to one or more predetermined thresholds corresponding to a set of predefined gears”

130. Li as combined with Cideciyan teaches the “*comparing*” as claimed, because Li teaches comparing the determined raw BER to “*one or more predetermined thresholds*” (including “a first threshold,” “a second threshold,” and “a third threshold”), where the comparison results are used to select a gear having a corresponding ECC scheme with an appropriate ECC strength. Ex.1005, Li, [0105]. The one or more thresholds are parameters used for determining whether the gear is appropriate for a particular situation, and are included in the “*gear*” for

that corresponding ECC scheme, as construed at III.A.

131. Namely, as an example, Li teaches while a raw BER of a page of a flash memory is less than **a first threshold**, the page is operated as an MLC page and encoded with **a first short code length ECC**. If/when the raw BER **exceeds the first threshold (but remains less than a second threshold)**, then the page is encoded with **a first longer code length ECC** (while still operated as an MLC page). If/when the raw BER exceeds **the second threshold (but remains less than a third threshold)**, then the page is encoded with **an even longer code length ECC**.

Ex.1005, Li, [0105].

132. A POSITA would have understood that “a first short code length ECC,” “a first longer code length ECC,” “an even longer code length ECC” corresponding to different ECC schemes, each ECC scheme is specified by a set of parameters of a “*gear*,” including parameters of code type and code rate, in addition to the parameter of the code length.

133. Further, a POSITA would have understood that Li’s teaching of comparing and then selecting an ECC scheme based on the comparison as described in [0105] applies to various ECC schemes having various code types, code rates, and code lengths, and is not limited to only “a first short code length

ECC,” “a first longer code length ECC,” “an even longer code length ECC” as described in [0105] of Li.

c. Li’s thresholds relate to “gears”.

134. Li’s thresholds relate to the “gears” corresponding to ECC schemes. The thresholds used for determining whether the particular ECC scheme with corresponding correction capability is appropriate for a particular situation, together with the parameters (including e.g., code type, code rate, code length) for specifying the particular ECC scheme, collectively define a “gear”.

135. In other words, Li’s set of ECC schemes (“*different predefined ECC schemes*”) providing different error correcting capabilities correspond to “*a set of predefined gears*” (each gear includes a plurality of parameters specifying an ECC scheme and parameters including the one or more thresholds to determine whether the gear is appropriate for the situation). Ex.1005, Li, [0019] (“a **set** of ECC schemes with various code types, code rates and/or various code lengths,” where the set of ECC schemes provide “**different** error correcting capabilities.”).

136. According, as discussed at [1.2.1].a-c, Li as combined with Cideciyan renders obvious [1.2.1].

[1.2.2] wherein the first gear has a different data payload size and correction capability than the second gear, wherein the amount of memory space allocated for the storage of data payload within the region varies between the first gear and the second gear to accommodate a varying number of parity symbols between the first gear and the second gear, and

137. Li, as combined above with Cideciyan, renders obvious [1.2.2]. The term “gear” is discussed above in the Claim Construction section. VIII.A.

138. First, as discussed at [1.2.1].a and shown in Table 1 above at ¶125 (reproduced below), Li as combined with Cideciyan renders obvious “**first gear**” and “**second gear**,” which are selected based on different raw BERs, and as such, have different correction capabilities corresponding to the different raw BERs.

<i>parameters specifying an ECC scheme</i>		<i>ECC scheme</i>	<i>parameters used for determining whether the gear is appropriate</i>
Set 1 (Example Gear 1)	code type: Product Code with c1:BCH, c2: Hamming code; code-rate-1; code-length-1; component-code-c1: c1-code-rate-1, c1-code-length-1; component-code-c2: c2- code-rate-1; c2-code-length-1; etc.	ECC scheme 1	BER-threshold- x1 < Determined BER < BER- threshold-y1
Set 2 (Example Gear 2)	code type: Product Code with c1:BCH, c2:RS; code-rate-2; code-length-2; component-code-c1; c1-code-rate-2; c1-code-length-2; component-code-c2: c2-code-rate-2; c2-code-length-2; etc.	ECC scheme 2	BER-threshold- x2 < Determined BER < BER- threshold-y2

Dolecek, Table 1

139. Second, Li describes different gears having different data payload sizes and correction capabilities, and different number of parity symbols. As an

example, Li describes “*a varying number of parity symbols between the first gear and the second gear*”:

2. The system of claim 1, wherein encoding according to a **first** one of the error correcting codes results in a number of **error correcting bits** to **store in the portion** that is **less than** when encoding according to a **second** one of the error correcting codes.

Ex.1005, Li, claim 2. A POSITA would have understood that Li’s teaching of gears as described in claims 1-4 applies to various gears corresponding to various ECC schemes having various code types (e.g., code rates, and code lengths, including Example Gear 1 and Example Gear 2 in Table 1. *See e.g.*, Ex.1005, Li, [136].

140. As described in Claim 2, Li as combined with Cideciyan teaches “*a varying number of parity symbols between the first gear and the second gear,*” because Li a number of error correcting bits (“*number of parity symbols*”) according to a first gear corresponding to a first ECC scheme (e.g., specified by a set of parameters including a first ECC code of Example Gear 1 and other parameters) is less than (“*varies*”) when encoding according to a second gear corresponding to a second ECC scheme (e.g., specified by a set of parameters including a second ECC code of Example Gear 2 and other parameters).

141. Third, Li describes that “the first gear has a different data payload

size,” and “the amount of memory space allocated for the storage of data payload within the region varies between the first gear and the second gear to accommodate a varying number of parity symbols” as claimed. Specifically, Li provides,

3. The system of claim 2, wherein when encoding according to the first error correcting code, a number of bits of the portion used as user data is increased by up to a difference between the number of error correcting bits used by the second error correcting code minus the number of error correcting bits used by the first error correcting code.

4. The system of claim 2, wherein when encoding according to the second error correcting code, a number of bits of the portion used as user data is decreased by up to a difference between the number of error correcting bits used by the second error correcting code minus the number of error correcting bits used by the first error correcting code.

Ex.1005, Li, claims 3-4.

142. As described in claims 3 and 4 of Li, in the combination of Li and Cideciyan, “*the first gear has a different data payload size,*” because a number of bits of the portion used as user data (“*data payload size*”) of the first ECC scheme (“*first gear*”) is greater than number of bits of the portion used as user data of the second ECC scheme (“*second gear*”).

143. Further, as described in claims 3 and 4, in the combination of Li and

Cideciyan, in the “portion” (group of pages distrusted across N channels) of the flash memory, the number of bits of the portion used as user data between the first ECC scheme and the second ECC scheme changes (“increased by up to a difference” or “decreased by up to a difference”) based on the difference of error correcting bits in the region. As such, Li as combined with Cideciyan renders obvious “*the amount of memory space allocated for the storage of data payload within the region varies between the first gear and the second gear to accommodate a varying number of parity symbols.*”

144. In other words, Li as combined with Cideciyan applies adaptive ECC scheme selection to a portion (a group of pages across the channel) of the flash memory, and the portion has a fixed size. *See* discussion at Ground 1: [1.1.1]. In this portion with a fixed size, when dynamically switching between gears (different ECC schemes), the number of parity symbols (error correcting bits) varies (e.g., increases/decreases) between the gears, and the storage for the data payload changes (decreases/increases according to the difference of number of parity symbols) to accommodate the varying number parity symbols.

145. This is also consistent with the well-known knowledge in the art. *See e.g.,* Ex.1008, Leung, Table 1 (illustrating that gear 2 reduces data payload size to accommodate increased parity size).

[1.3] *selecting a gear from the set for the region based at least partly on the comparisons to the one or more predetermined thresholds;*

146. Li, as combined above with Cideciyan, renders obvious [1.3]. The term “gear” is discussed above in the Claim Construction section. VIII.A.

147. First, as discussed at [1.2.1].b, Li as combined with Cideciyan describes *comparing the determined bit error rate to one or more predetermined thresholds* for adaptive ECC scheme selection.

148. Second, as discussed at [1.2.1].b, Li as combined with Cideciyan describes that SSD controller 200 (e.g., implementing an adaptive encoder hardware block) is used for *selecting a gear from the set* (selecting from the set of predefined gears as discussed at [1.2.1]) *for the region* (the group including N pages distributed across N channels) *based at least partly on the comparisons to the one or more predetermined thresholds*. Ex.1005, Li, [0023] (“**an adaptive encoder hardware block ... enabled to dynamically determine the selected error correcting code based at least in part on the raw BER.**”); Ex.1005, Li, [0105]; *see also* Ex.1005, Li, Claim 1 (“**dynamically determine** the dynamically selected error correcting code based at least in part on the raw BER”).

149. It is noted that a POSITA would have understood that Li’s teaching of comparing and then selecting an ECC scheme based on the comparison as described in [0105] applies to various ECC schemes having various code types, code rates, and code lengths, and is not limited to only “a first short code length ECC,” “a first longer code length ECC,” “an even longer code length ECC” as

described in [0105] of Li.

[1.4] wherein determining, comparing, and selecting are performed by an integrated circuit,

150. Li, as combined above with Cideciyan, renders obvious [1.4].

151. As discussed at [1.1.1], [1.2.1], and [1.3], Li as combined with Cideciyan describes an SSD controller for performing *determining, comparing, and selecting*. Li further describes that the SSD controller is implemented on an integrated circuit. Ex.1005, Li, [0125], [0014] (“all [] of an SSD controller (or a computing-host flash memory controller) are implemented **on a single IC [Integrated Circuit]**, a single die of a multi-die IC.” *See also* Ex.1007, Frost, 5:4-7 (“The FLASH controller 10 may take many forms. In the example of FIG. 1, the FLASH controller 10 is a **field programmable gate array (FPGA)** that, during start-up of the system is programmed and configured by the CPU 15.”). It was well-known that an FPGA is an integrated circuit.

152. This is consistent with the description of the '085 patent. Ex.1001, '085 patent, 56:56-59 (“the SSD controller 1300 is implemented in an integrated circuit, such as by an application specific integrated circuit (ASIC), a field programmable gate array (FPGA).”).

2. Claim 2

[2.0] The method of claim 1, further comprising repeatedly determining the bit error rate and comparing the determined bit error rate to one or more predetermined thresholds such that the bit error rate is actively monitored.

153. The combination of Li and Cideciyan renders obvious [2.0].

154. Specifically, Li teaches repeatedly determining (periodically determining) the bit error rate and comparing it to the thresholds based on the number of program/erase cycles, *such that the bit error rate is actively monitored* (for adaptively selecting the suitable ECC scheme throughout the lifetime of the flash memory). Ex.1005, Li, [0092] (“the Error Statistics Collecting/Tracking hardware block is enabled to **periodically** determine (such as **once every 100 PE cycles**) a current raw (measured) BER of one or more portions of flash storage.”), [0048] (“wherein the measured raw BER is determined periodically”), [0019] (“**dynamically switching** encoding/decoding of all or any portions of the flash memory amongst a respective one or more of the ECC schemes based at least in part on information from the error statistic collecting/tracking”).

3. Claim 4

[4.0] *The method of claim 1, wherein determining, comparing, and selecting are performed in connection with at least one of flash initialization, read, or write operations for an end user.*

155. The combination of Li and Cideciyan renders obvious [4.0].

156. As illustrated in FIGS. 1 and 2A below, Li describes determining, comparing, and *selecting* (adaptive ECC scheme selection as discussed at [1.1.1], [1.2.1], and [1.3]) *are performed in connection with* (based on parameters related to) *at least one of flash initialization, read* (a number of errors corrected, a number

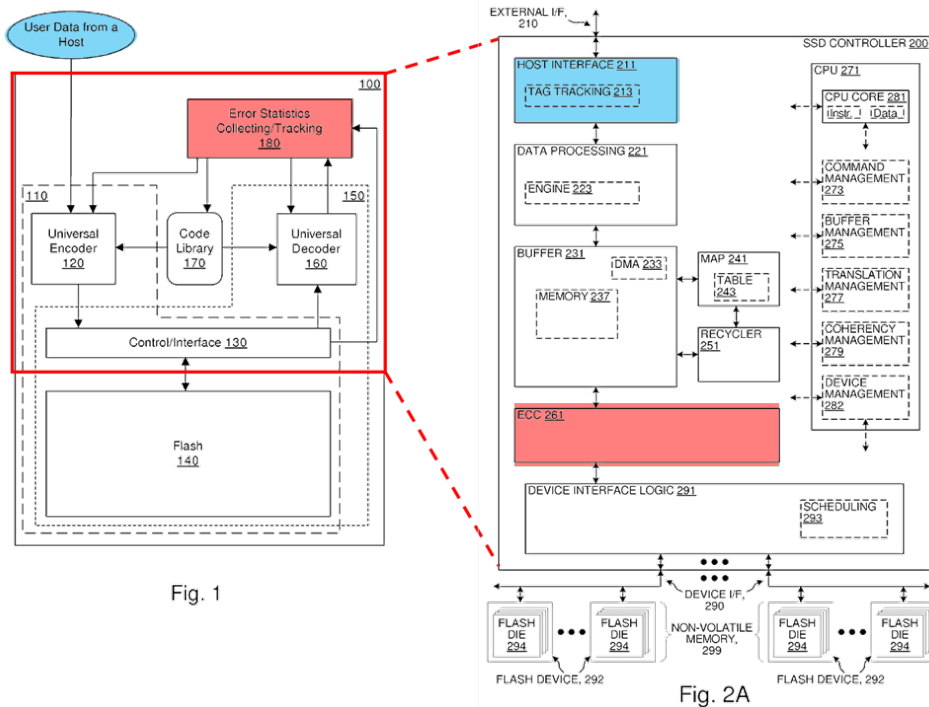
of errors detected, a number of read cycles, a read time), *or write operations* (a number of program/erase cycles, a program time, an erase time) *for an end user* (including read and write operations for an end user received through host interface 211). Ex.1005, Li, [0114] (“**Host interface 211 sends and receives commands and/or data via external interface 210**, and, in some embodiments, tracks progress of individual commands via tag tracking 213. For example, the commands include a **read command** specifying an address (such as an LBA) and an amount of data (such as a number of LBA quanta, e.g. sectors) to read; **in response the SSD provides read status and/or read data**. For another example, the commands include a write command specifying an address (such as an LBA) and an amount of data (such as a number of LBA quanta, e.g. sectors) to write; in response the SSD provides write status and/or requests write data and optionally subsequently provides write status.”).

157. As further detail, Li describes a write-storage-data path and a read-storage-data path. Ex.1005, Li, [0088] (“A **write-storage-data path 110** includes various hardware blocks: a Universal Encoder 120 coupled to a Control/Interface 130 that is in turn coupled to a Flash unit 140...A **read-storage-data path 150** includes various hardware blocks: the Flash unit and the Control/Interface coupled to a Universal Decoder 160... An **Error Statistics Collecting/Tracking 180** hardware block is coupled to the Universal Encoder, the Code Library, the

Universal Decoder, and the Control/Interface hardware blocks”).

158. A POSITA would have understood that in Li, the parameters “a number of errors corrected, a number of errors detected,...a number of read cycles..., a read time” are related to (*in connection with*) *read operations for an end user*, and parameters “a number of program/erase cycles,...a program time, an erase time” are related to (*in connection*) *with write operations for an end user*. Ex.1005, Li, Claim 16 (“**dynamic code rate selection** is enabled to **dynamically determine the respective code rate based at least in part on one or more parameters per one or more of the portions...**the parameters comprising **a number of errors corrected, a number of errors detected, a number of program/erase cycles, a number of read cycles, a program time, an erase time, a read time**, a temperature, and a threshold voltage.”).

determining, comparing, and selecting are performed in connection with at least one of flash initialization, read, or write operations for an end user



Ex.1005, Li, FIGS. 1 and 2A, annotated.

159. Li provides additional description on how the determining, comparing, and selecting are performed in connection with at least one of flash initialization, read, or write operations **for an end user**, because they are performed based on parameters related to read/write operations commanded by the host for an end user. Ex.1005, Li, FIG. 1 (illustrating “**user data** from a host,” which is from an end user), [0089] (describing write operation from an end user, where ““**User Data from a Host**” to write as storage data is received by the Universal Encoder and encoded according to an error correcting code”); [0090]

(describing read operations from an end user, where “[r]eading storage data begins by Control/Interface reading raw information,” and the decoded “data information is then **passed to the Host.**”); Ex.1005, Li, [0023] (“**an adaptive encoder hardware block** enabled to encode according to a selected one of a plurality of error correcting codes, and further enabled to dynamically **determine the selected error correcting code** based at least in part on the raw BER.”).

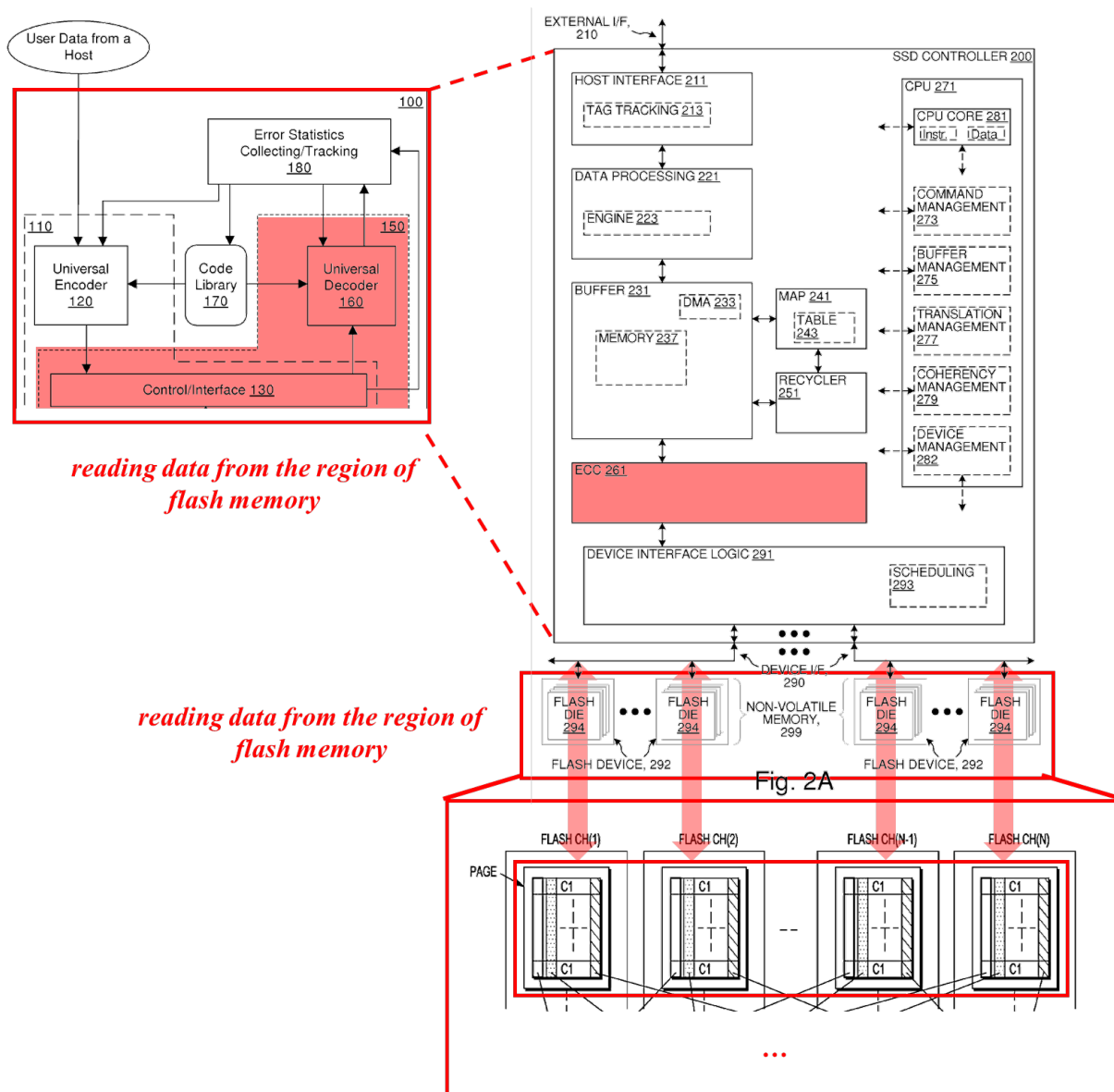
4. Claim 5

[5.0] *The method of claim 1, further comprising reading data from the region of flash memory,*

160. The combination of Li and Cideciyan renders obvious [5.0].

161. As shown in Example 1.1 of Li combined with Cideciyan below, Li as combined with Cideciyan describes “*reading data*” (e.g., using SSD controller 200 including elements implementing the read-storage-data path, including the Code Library, Error Statistics Collecting/Tracking 180, Universal Decoder 160) “*from the region of flash memory*” (from a portion/a group of pages across channels of the flash based SSD). Ex.1005, Li, [0050] (“the measured raw BER is determined at least in part by observing a BER associated with at least some **reads of the portion**”); [0088] (“A **read-storage-data path 150** includes various hardware blocks: the Flash unit and the Control/Interface coupled to a Universal Decoder 160... An **Error Statistics Collecting/Tracking 180** hardware block is coupled to the Universal Encoder, the Code Library, the Universal Decoder, and the

Control/Interface hardware blocks”).



Example 1.1 of Li Combined with Cideciyan.

[5.1] wherein determining the bit error rate further comprises estimating the bit error rate based at least partly on a number of errors corrected by ECC in connection with reading the data from the region.

162. The combination of Li and Cideciyan renders obvious [5.1].

163. Li as combined with Cideciyan describes *determining the bit error rate further comprises* includes *estimating* (using SSD controller 200 including elements implementing Error Statistics Collecting/Tracking 180) *the bit error rate based at least partly on a number of errors corrected by ECC* (by comparing raw read data from the flash memory with an error-corrected version of the raw read data) *in connection with reading the data from the region* (associated with the reads of the portion). Ex.1005, Li, [0050] (“the measured raw BER is determined at least in part by observing a BER associated with at least some **reads of the portion**”); [0051] (“measured raw BER is determined at least in part by comparing raw read data from the flash memory with an error-corrected version of the raw read data.”); *see also* Ex.1005, Li, [0079] (describing that an ECC scheme and/or its parameter base on “a number of errors corrected.”).

5. Claim 14

[14.0] *The method of claim 1, wherein the gear includes a lower-latency ECC decode option and one or more higher-latency ECC decode options, wherein the gear is selected such that performing ECC decoding using the lower-latency ECC decode option alone is sufficient to provide corrected data at least 90% of the time.*

164. The combination of Li and Cideciyan renders obvious [14.0].

165. Li describes “decoding in accordance with one or more of the ECC schemes.” Ex.1005, Li, Abstract; *see also* claim 15 (“means for decoding operable according to the respective determined code rates”). It would be obvious to

implement in Li, well-known examples of providing decode options with different latencies for a single ECC scheme, as taught by Cideciyan.

166. First, Cideciyan provides *a lower-latency ECC decode option* (C1 decode option) *and one or more higher-latency ECC decode option* (the decode option of using both C1 and C2 decoding), where the decode option of using both C1 and C2 decoding inherently has a higher-latency because the system first incurs the latency of C1 decoding. Ex.1006, Cideciyan 14:29-33 (describing that C2 decoding is performed after C1 decoding failed, where the failed “**C1 erasures can then be corrected** by the C2 decoder”); 14:41-50 (“Whenever a strip contains page or block errors that **cannot be corrected by the C1 error correction process**,” it is reconstructed “via the **C2 decoding** algorithm.”).

167. Second and alternatively, Cideciyan provides implementation detail, describing *a lower-latency ECC decode option and one or more higher-latency ECC decode options* (C1 decode option and C2 decode option having different latencies). Ex.1006, Cideciyan, 18:21-23 (“For example, early in the storage lifetime only C1 decoding might be routinely performed unless device failure requires C2 decoding”), 18:23-25 (“As the raw error rate increases with increased wear, however, C2 decoding could be routinely performed to improve data reliability and device endurance.”).

168. It was well-known to apply different coding options having different

latencies. Ex.1010, Lasser, 14:1- 5 (describing providing two decoding schemes with different processing times), 11:22-27 (describing two decoding schemes for correcting up to a first number of errors and a second number of errors in the particular number of data bits); Ex.1011, 11:5-8 (applying two decoding modes having different latency).

169. In further detail, Cideciyan describes performing C2 decoding only after C1 decoding fails, and a POSITA would have understood that C2 decoding has a higher latency because it is designed to correct more errors than the C1 decoding. Ex.1006, Cideciyan 14:29-33 (“C1 codeword is declared as ‘erased’ (erroneous) whenever the C1 decoder of EC1 module 13 is not able to decode the C1 codeword. **These C1 erasures can then be corrected** by the C2 decoder of EC2 module 14.”); 14:41-53 (“Whenever a strip contains page or block errors that **cannot be corrected by the C1 error correction process**, the following procedure is performed by flash controller 10....The content of the bad strip is read, and the content of the other strips in the stride is used to reconstruct erroneous data in the bad strip via the C2 decoding algorithm.”).

170. Third, the specification of the ’085 patent does not show any unexpected result or other unexpectedly improved properties for a percentage amount of “90%.” *See* Ex.1001, ’085 patent, 23: 40-46 (simply describing that “[o]ther values are applicable, such as, but not limited to, at least 95% of the time

or at least 90% of the time” without any showing of any unexpected result or other unexpectedly improved properties).

171. Regardless, a POSITA would have found it obvious to select a gear based on the determined BER, such that performing ECC decoding using the lower-latency ECC decode option alone is sufficient to provide corrected data at least 90% of the time. SSDs have requirements of low uncorrected BER (UBER) for error correction. Ex.1006, Cideciyan, 4:1-4 (“the low user bit error rates of $1e-11$ to $1e-19$ typically required by applications”); Ex.1009, JESD218, 11 (industry standard providing UBER requirement of less than 10^{-15} for SSDs). And as such, the requirements for corrected data for using the lower-latency ECC decode option alone, even for the low-latency ECC decode option alone, usually is much higher than 90%. For example, as shown in FIG. 11 of Cideciyan, the C1 decode only option alone achieves UBER of $1e-11$, $1e-15$, $1e-19$ with a raw NAND bit error rate of $1e-3$. *See* Ex.1009, Cideciyan, FIG. 11, 16:25-40.

6. Claim 15

[15.0] *An apparatus comprising:*

172. To the extent that the preamble is limiting, Li as combined with Cideciyan renders obvious [15.0].

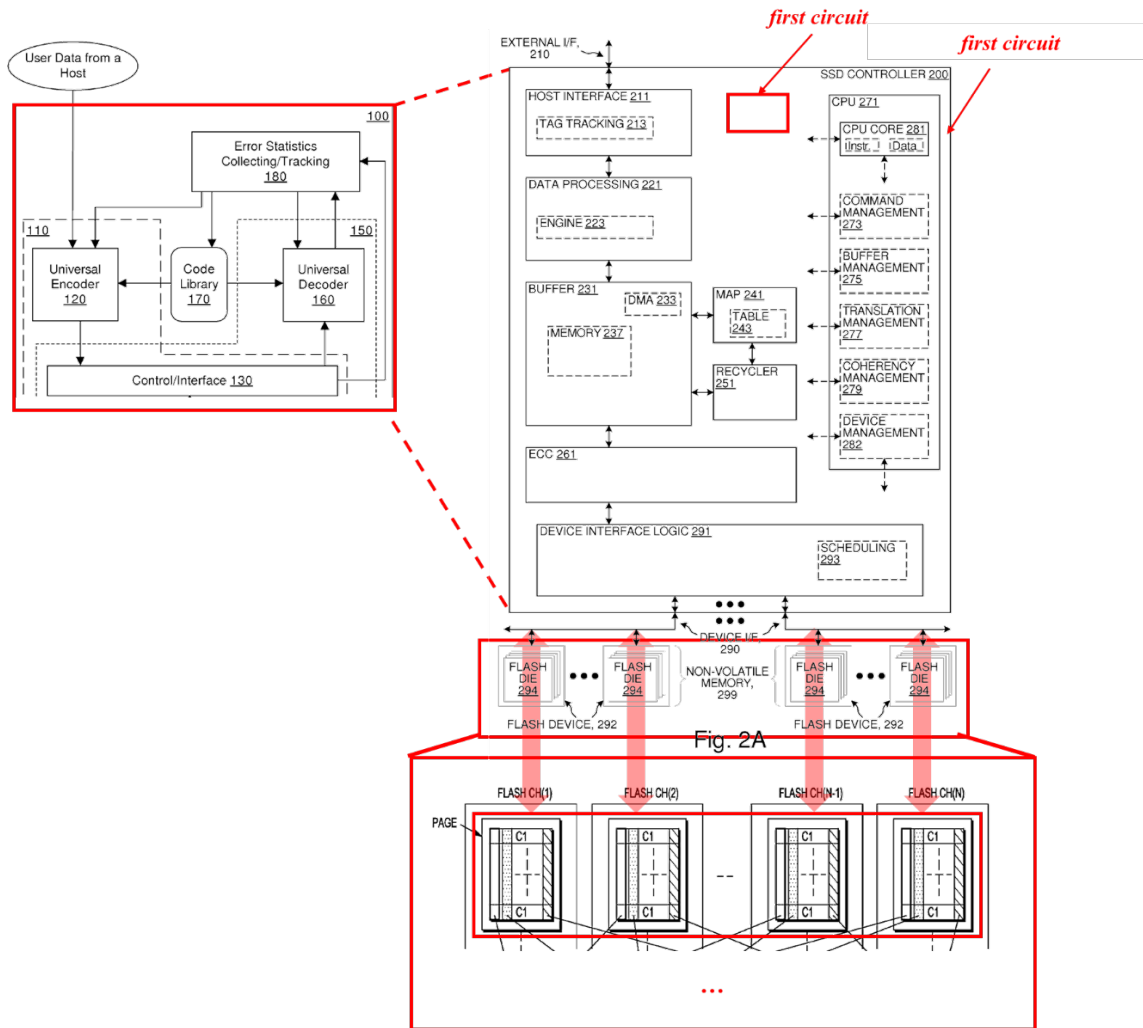
173. Specifically, Li describes that its invention “may be implemented in ...an **apparatus**.” Ex.1005, Li, [0007]. Further, as discussed below at [15.1.1] and

[15.3], Li describes that the apparatus includes the claimed first circuit and second circuit.

[15.1.1] *a first circuit configured to determine a bit error rate associated with a region comprising a fixed number of two or more flash memory pages or integer fractions thereof, wherein the two or more flash memory pages of a region can be read simultaneously,*

174. The combination of Li and Cideciyan renders obvious [15.1.1].

175. As discussed at [1.2.1] and shown in Example 1.1 of the combination of Li and Cideciyan below, the combination of Li and Cideciyan teaches *a first circuit* (e.g., using SSD controller 200 including elements implementing the read-storage-data path, including the Code Library, Error Statistics Collecting/Tracking 180, Universal Decoder 160) *configured to determine a bit error rate associated with a region comprising a fixed number of two or more flash memory pages or integer fractions thereof.* Ex.1005, Li, [0126] (“**elements of SSD controller 200 implement various hardware blocks** of FIG. 1 (or functions performed by the hardware blocks) in whole or in part. For example, ECC 261 implements one or more functions performed by the Error Statistics Collecting/Tracking, Universal Encoder, Universal Decoder, and/or Code Library hardware blocks of FIG. 1.”).



Example 1.1 of Combination of Li and Cideciyan

[15.1.2] wherein the region stores at least data payload and primary and secondary ECC parity symbols corresponding to the data payload of the region and

176. The combination of Li and Cideciyan renders obvious [15.1.2] for reasons discussed at [1.1.2].

[15.2.1] [the first circuit] configured to compare the determined bit error rate to one or more predetermined thresholds corresponding to a set of predefined gears comprising at least a first gear and a second gear, wherein the predefined gears correspond to different error correction coding (ECC) schemes,

177. The combination of Li and Cideciyan renders obvious [15.2.1], because *the first circuit* (e.g., using SSD controller 200 including elements implementing the read-storage-data path, including the Code Library, Error Statistics Collecting/Tracking 180, Universal Decoder 160) performs the comparing for reasons discussed at [1.2.1] and [1.2.2].

[15.2.2] wherein the first gear has a different data payload size and correction capability than the second gear, wherein the amount of memory space allocated for the storage of data payload within the region varies between the first gear and the second gear to accommodate a varying number of parity symbols between the first gear and the second gear; and

178. The combination of Li and Cideciyan renders obvious [15.2.2] for the reasons discussed at [1.2.2].

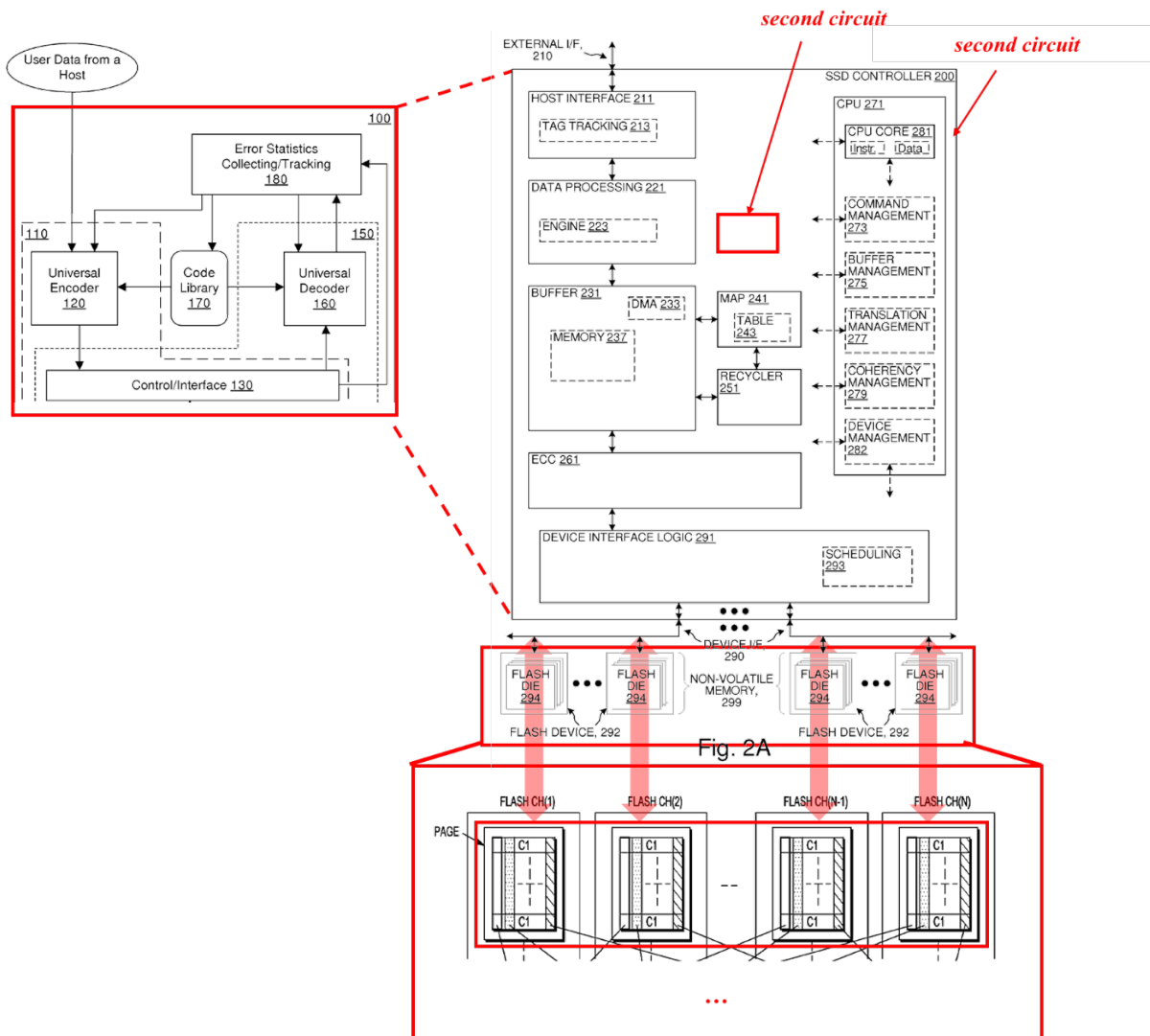
[15.3] a second circuit configured to select a gear from the set for the region based at least partly on the comparisons to the one or more predetermined thresholds.

179. The combination of Li and Cideciyan renders obvious [15.3].

180. As discussed at [1.3] and shown in Example 1.1 of the combination of Li and Cideciyan below, the combination of Li and Cideciyan teaches *a second circuit* (SSD controller 200 including elements implementing **adaptive encoder hardware block**, using any suitable elements of SSD controller 200 as well-known in the art) *configured to select a gear from the set for the region based at least partly on the comparisons to the one or more predetermined thresholds.*

Ex.1005, Li, [0023] (“**an adaptive encoder hardware block** enabled to encode

according to a selected one of a plurality of error correcting codes, and further enabled to dynamically **determine the selected error correcting code** based at least in part on the raw BER.”); [0126] (“**elements of SSD controller 200** **implement various hardware blocks** of FIG. 1 (or functions performed by the hardware blocks) in whole or in part. For example, ECC 261 implements one or more functions performed by the Error Statistics Collecting/Tracking, Universal Encoder, Universal Decoder, and/or Code Library hardware blocks of FIG. 1.”).



Example 1.1 of Combination of Li and Cideciyan.

181. As shown in the above annotated figure for Example 1.1 of Combination of Li and Cideciyan, the second circuit may be the same as the first circuit of [15.1.1] when both first circuit and second circuit are implemented using the same component of the SSD controller circuit 200. Alternatively, it would be obvious that the second circuit is different from the first circuit of [15.1.1], where different components of the SSD controller 200 are used to implement the first and second circuits.

7. Claim 16

[16.0] *The apparatus of claim 15, wherein the ECC encoder is configured to determine the bit error rate and*

182. The combination of Li and Cideciyan renders obvious [16.0].

183. Specifically, Li describes that the ECC encoder (e.g., ECC 261 implementing Error Statistics Collecting/Tracking, Universal Encoder, and/or Code Library) is configured to determine the bit error rate. Ex.1005, Li, [0126] (“ECC 261 implements one or more functions performed by the **Error Statistics Collecting/Tracking, Universal Encoder**, Universal Decoder, and/or Code Library hardware blocks of FIG. 1) (explaining that Li’s ECC 261 includes ECC encoder performing the functions of the Error Statistics Collecting/Tracking).

[16.1] *the processor is configured to compare and to select in connection with at least one of flash initialization, read, or write operations for an end user.*

184. The combination of Li and Cideciyan renders obvious [16.1].

185. As discussed at [4.0], the combination of Li and Cideciyan renders obvious comparing and selecting in connection with at least one of flash initialization, read, or write operations for an end user as claimed. Ex.1005, Li, [0114], Claim 16, FIGS. 1, 2A.

186. The combination of Li and Cideciyan renders that the “*processor*” is configured to perform selected functions including the comparing and selecting as a routine implementation. Specifically, Li describes using a processor to implement selected functions, which include the comparing and selecting functions. Ex.1005, Li, [0137] (“It is also understood that **many functional aspects** of embodiments **are realizable selectively** in either hardware (i.e., generally dedicated circuitry) or software (i.e., via some manner of programmed controller or **processor**)”); [0131] (“various combinations of **all or portions of operations** performed by a system implementing adaptive ECC techniques for flash memory based data storage, e.g. the hardware blocks of FIG. 1, a computing-host flash memory controller, and/or an SSD controller (such as SSD controller 200 of FIG. 2A), and **portions of a processor, microprocessor, system-on-a-chip**, application-specific-integrated-circuit, hardware accelerator, or other circuitry **providing all or portions of the aforementioned operations**, are specified by a specification **compatible with processing by a computer**

system.”); *see also* Ex.1006, Cideciyan, 9:29-31 (“control logic could be implemented in whole or in part by software which configures a processor of control apparatus 7 to perform the functions described.”).

8. Claim 17

[17.0] *The apparatus of claim 15, wherein the first circuit is further configured to read data from the region of flash memory,*

187. The combination of Li and Cideciyan renders obvious [17.0].

188. Specifically, in the combination of Li and Cideciyan, the first circuit (e.g., using SSD controller 200 including elements implementing the read-storage-data path, including the Code Library, Error Statistics Collecting/Tracking 180, Universal Decoder 160) *is further configured to read data from the region of flash memory* for the reasons discussed at [5.0].

[17.1] *wherein the ECC decoder is further configured to determine the bit error rate based at least partly on a number of errors corrected by ECC in connection with a read of data from the region.*

189. The combination of Li and Cideciyan renders obvious [17.1].

190. As discussed at [5.1], in the combination of Li and Cideciyan, *the ECC decoder* (e.g., using SSD controller 200 including elements implementing the read-storage-data path, including the Code Library, Error Statistics Collecting/Tracking 180, and Universal Decoder 160) is further configured to “*determine the bit error rate*” (raw BER) “*based at least partly on a number of errors corrected by ECC*” (by comparison between the raw BER and the error-corrected version of

the raw read data) *in connection with a read of data from the region* (associated with the reads of the portion). Ex.1005, Li, [0050] (“the measured raw BER is determined at least in part by observing a BER associated with at least some **reads of the portion**”); [0051] (“measured raw BER is determined at least in part by comparing raw read data from the flash memory with an error-corrected version of the raw read data.”); *see also* Ex.1005, Li, [0079] (describing that an ECC scheme and/or its parameter based on “a number of errors corrected.”).

9. Claim 22

[22.0] *The apparatus of claim 15, wherein the gear includes a lower-latency ECC decode option and one or more higher-latency ECC decode options, wherein the gear is selected such that performance of ECC decoding with the lower-latency ECC decode option alone is sufficient to provide corrected data at least 90% of the time.*

191. The combination of Li and Cideciyan renders obvious [22.0] for the reasons discussed at [14.0].

10.Claim 23

[23.0] *The apparatus of claim 15, further comprising a plurality of flash memory devices, wherein the apparatus comprises a flash drive.*

192. The combination of Li and Cideciyan renders obvious [23.0].

193. As shown in FIG. 2A, Li describes an apparatus comprising *a plurality of flash memory devices* (flash devices 292). Ex.1005, Li, [0110] (“SSD controller 200 is further communicatively coupled via one or more device interfaces 290 to non-volatile memory 299 including one or more storage devices,

such as **flash devices 292.**”). Li further describes the apparatus *comprises a flash drive* (Solid State Disk/Drive (SSD)). Ex.1005, Li, [0014], [0016] (“Some SSDs use flash memory to provide non-volatile storage.”).

11.Claim 24

[24.0] *An apparatus for selecting an error correction coding (ECC) scheme, the apparatus comprising:*

194. To the extent that the preamble is limiting, the combination of Li and Cideciyan renders obvious [24.0] for reasons discussed at [15.0].

[24.1.1] *a means for determining a bit error rate associated with a region comprising a fixed number of two or more flash memory pages or integer fractions thereof, wherein the two or more flash memory pages of a region can be read simultaneously,*

195. The combination of Li and Cideciyan renders obvious [24.1.1] for the reasons discussed at [1.1.1] and [15.1.1].

196. I was informed that construction of means-plus-function claim terms that invoke 35 U.S.C. §112, ¶6, requires identifying the claim terms and identifying corresponding structure disclosed in the specification that performs the identified functions and equivalents thereof.

197. A POSITA would have understood that [24.1.1] to have the structure and function presented below:

198. Function: determining a bit error rate associated with a region comprising a fixed number of two or more flash memory pages or integer fractions. Ex.1001, '085 patent, 4:38-40;

199. Structure: a circuit. Ex.1001, '085 patent, 4:38-40 (“**a first circuit** configured to determine a bit error rate associated with a region associated with a plurality of flash memory pages or integer fractions thereof”).

200. As discussed at [1.1.1] and [15.1.1], the combination of Li and Cideciyan renders obvious [24.1.1], because it renders obvious a circuit (ECC 261 implementing Error Statistics Collecting/Tracking 180) for performing the function of determining a bit error rate associated with a region comprising a fixed number of two or more flash memory pages.

[24.1.2] *wherein the region stores at least data payload and primary and secondary ECC parity symbols corresponding to the data payload of the region;*

201. The combination of Li and Cideciyan renders obvious [24.1.2] for the reasons discussed at [1.1.2].

[24.2.1] *a means for comparing the determined bit error rate to one or more predetermined thresholds corresponding to a set of predefined gears comprising at least a first gear and a second gear, wherein the predefined gears correspond to different predefined ECC schemes,*

202. The combination of Li and Cideciyan renders obvious [24.2.1] for reasons discussed at [1.2.1] and [15.2.1].

203. I was informed that construction of means-plus-function claim terms that invoke 35 U.S.C. §112, ¶6, requires identifying the claim terms and identifying corresponding structure disclosed in the specification that performs the identified functions and equivalents thereof.

204. A POSITA would have understood that “a means for comparing the determined bit error rate to one or more predetermined thresholds corresponding to a set of predefined gears comprising at least a first gear and a second gear” to have the structure and function presented below:

205. Function: comparing the determined bit error rate to one or more predetermined thresholds corresponding to a set of predefined gears comprising at least a first gear and a second gear. Ex.1001, '085 patent, 4:38-40.

206. Structure: a circuit. Ex.1001, '085 patent, 4:38-47 (“**a first circuit** configured...to compare the determined bit error rate to one or more predetermined thresholds corresponding to a set of predefined gears comprising at least a first gear and a second gear, wherein the predefined gears correspond to different error correction coding (ECC) schemes”).

207. As discussed at [1.2.1] and [15.2.1], the combination of Li and Cideciyan renders obvious [24.2.1], because it renders obvious a circuit (Error Statistics Collecting/Tracking 180) for performing the function of comparing the determined bit error rate to one or more predetermined thresholds corresponding to a set of predefined gears, wherein the predefined gears correspond to different predefined ECC schemes.

[24.2.2] wherein the first gear has a different data payload size and correction capability than the second gear, wherein the amount of memory space allocated for the storage of data payload within the region varies between the first gear and the second gear to accommodate a varying number of parity

symbols between the first gear and the second gear;

208. The combination of Li and Cideciyan renders obvious [24.2.2] for reasons discussed at [1.2.2].

[24.3] *a means for selecting a gear from the set for the region based at least partly on the comparisons to the one or more predetermined thresholds.*

209. The combination of Li and Cideciyan renders obvious [24.3] for reasons discussed at [1.3] and [15.3].

210. A POSITA would have understood that “a means for selecting a gear from the set for the region based at least partly on the comparisons to the one or more predetermined thresholds” to have the structure and function presented below:

211. I was informed that construction of means-plus-function claim terms that invoke 35 U.S.C. §112, ¶6, requires identifying the claim terms and identifying corresponding structure disclosed in the specification that performs the identified functions and equivalents thereof.

212. Function: selecting a gear from the set for the region based at least partly on the comparisons to the one or more predetermined thresholds.

213. Structure: a circuit. Ex.1001, '085 patent, 4:47-49 (“**a second circuit** configured to select a gear from the set for the region based at least partly on the comparisons to the one or more predetermined thresholds”).

214. As discussed at [1.3] and [15.3], the combination of Li and Cideciyan

renders obvious [24.3], because it renders obvious a circuit (SSD controller 200 including elements implementing adaptive encoder hardware block, using any suitable elements of SSD controller 200 as well-known in the art) for performing the function of selecting a gear from the set for the region based at least partly on the comparisons to the one or more predetermined thresholds. Ex.1005, Li, [0023]; [0126].

12.Claim 25

[25.0] *The apparatus of claim 24, further comprising means for reading data from the region of flash memory,*

215. The combination of Li and Cideciyan renders obvious [25.0] for the reasons discussed at [5.0] and [17.0].

216. First, a POSITA would have understood that “*means for reading data from the region of flash memory*” to have the structure and function presented below:

217. Function: reading data from the region of flash memory.

218. Structure: a circuit including an ECC encoder/decoder. Ex.1001, '085 Patent, 53:1-7 (“Whenever a read occurs, **the ECC encoder/decoder 1310** performs ECC decoding 1122, 1124, 1126 to perform error detection and correction as needed as previously described. As it does so, the ECC encoder/decoder 1310 can count how many corrections, if any, it makes. This count can be a good approximation of the number of errors and related to a bit

error rate (BER) actually existing in the page stripe.”).

219. As discussed at [5.0] and [17.0], the combination of Li and Cideciyan renders obvious [25.0], because it renders obvious a circuit (e.g., using SSD controller 200 including elements implementing the read-storage-data path, including the Code Library, Error Statistics Collecting/Tracking 180, Universal Encoder 120 and Universal Decoder 160) that performs the function of *reading data from the region of flash memory*.

[25.1] wherein the determining means further comprises means for estimating the bit error rate based at least partly on a number of errors corrected by ECC in connection with reading the data from the region.

220. The combination of Li and Cideciyan renders obvious [25.1] for the reasons discussed at [5.1] and [17.1].

221. First, a POSITA would have understood that “*the determining means further comprises means for estimating the bit error rate based at least partly on a number of errors corrected by ECC in connection with reading the data from the region*” to have the structure and function presented below:

222. Function: estimating the bit error rate based at least partly on a number of errors corrected by ECC in connection with reading the data from the region.

223. Structure: a circuit including an ECC encoder/decoder. Ex.1001, '085 patent, 4:38-40 (“**a first circuit** configured to determine a bit error rate

associated with a region associated with a plurality of flash memory pages or integer fractions thereof”); 53:1-7 (“Whenever a read occurs, **the ECC encoder/decoder 1310** performs ECC decoding 1122, 1124, 1126 to perform error detection and correction as needed as previously described. As it does so, the ECC encoder/decoder 1310 can count how many corrections, if any, it makes. This count can be a good approximation of the number of errors and related to a bit error rate (BER) actually existing in the page stripe.”).

224. As discussed at [5.1] and [17.1], the combination of Li and Cideciyan renders obvious [25.1], because it renders obvious a circuit (ECC 261 implementing Error Statistics Collecting/Tracking 180, Universal Encoder 120 and Universal Decoder 160) for determining (*see* [24.1.1]) further performs the function of *estimating the bit error rate based at least partly on a number of errors corrected by ECC in connection with reading the data from the region.*

13.Claim 29

[29.0] *The apparatus of claim 24, wherein the gear includes a lower-latency ECC decode option and one or more higher latency ECC decode options, wherein the gear is selected such that performing ECC decoding using the lower-latency ECC decode option alone is sufficient to provide corrected data at least 90% of the time.*

225. The combination of Li and Cideciyan renders obvious [29.0] for reasons discussed at [22.0].

14.Claim 30

[30.0] *The apparatus of claim 24, further comprising a plurality of flash*

memory devices, wherein the apparatus comprises a flash drive.

226. The combination of Li and Cideciyan renders obvious [30.0] for reasons discussed at [23.0].

15.Claims 31, 32, and 33

[31.0/32.0/33.0] *The method of claim 1/apparatus of claim 15/apparatus of claim 24, wherein the data payload and corresponding primary ECC parity symbols comprise primary codewords,*

227. The combination of Li and Cideciyan renders obvious [31.0/32.0/33.0].

228. Continuing with the example the combination of Li and Cideciyan describes in claims 1, 15, and 24, as shown in FIG. 4 below, Cideciyan describes that *data payload* (input data) *and corresponding primary ECC parity symbols* (C1 parity symbols) *comprise primary codewords* (C1 codeword). Ex.1006, Cideciyan, 10:60-64 (“The EC1 encoder 13 systematically encodes each row in accordance with the C1 code to generate (n1 -k1) parity symbols for that row. This **C1 parity code is added to the k1 input data symbols to produce an n1-symbol C1 codeword.**”); 11:9-14 (“The array of encoded symbols shown in the FIG. 4 is **written in a single stride** as described above with reference to FIG. 3. The process is designed to give a particular configuration of C1 and C2 codewords in individual logical subdivisions (here storage channels 8) and across the set of subdivisions as a whole.”).

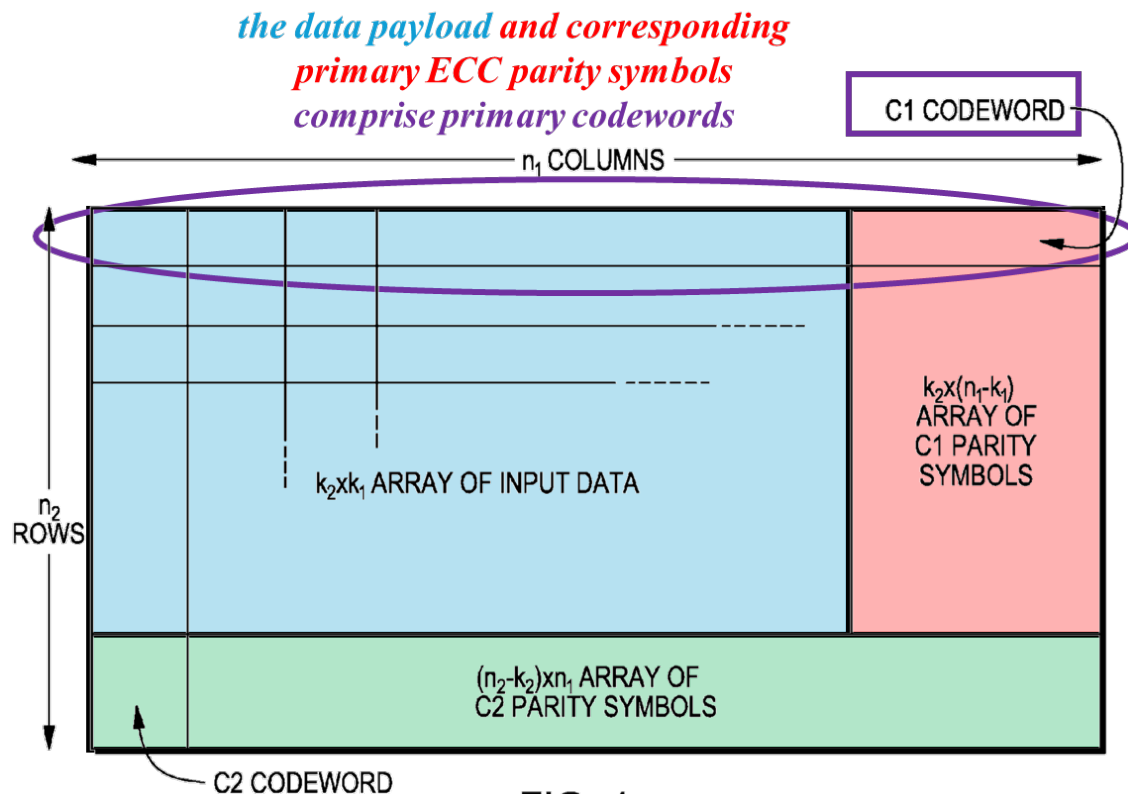


FIG. 4

Ex.1006, Cideciyan, FIG. 4, annotated.

[31.1/32.1/33.1] wherein the data payload and corresponding secondary ECC parity symbols comprise secondary codewords,

229. The combination of Li and Cideciyan renders obvious

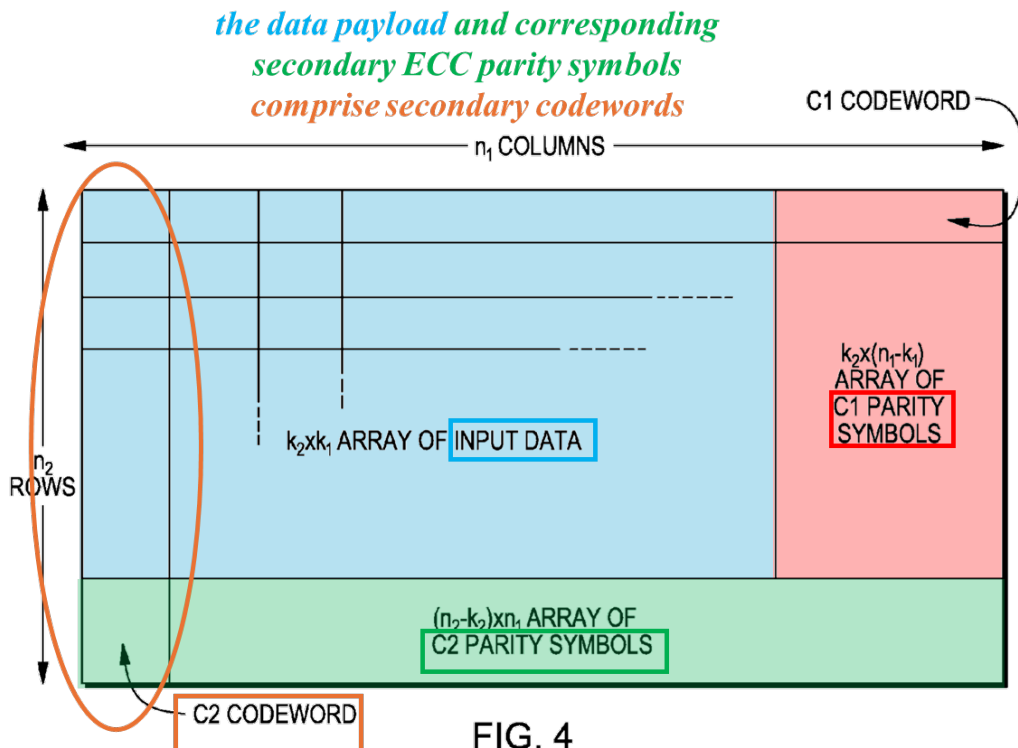
[31.1/32.1/33.1].

230. Continuing with the example the combination of Li and Cideciyan, as shown in FIG. 4 below, Cideciyan describes that *data payload* (input data) and *corresponding secondary ECC parity symbols* (C2 parity symbols) comprise *secondary codewords* (C2 codewords). Ex.1006, Cideciyan, 11:1-5 (“The C2

encoding performed by EC2 encoder 14 systematically encodes each column in accordance with the C2 code to generate **(n2-k2) parity symbols** for that column.

This C2 parity code is added to the k2 column symbols to produce an n2-symbol C2 codeword."); 11:9-14 (“The array of encoded symbols shown in the FIG. 4 is **written in a single stride** as described above with reference to FIG. 3.

The process is designed to give a particular configuration of C1 and C2 codewords in individual logical subdivisions (here storage channels 8) and across the set of subdivisions as a whole.”).



[31.2/32.2/33.2] wherein the secondary ECC parity symbols of each secondary codeword are programmed across the region,

231. The combination of Li and Cideciyan renders obvious

[31.2/32.2/33.2].

232. Continuing with the example the combination of Li and Cideciyan, Cideciyan describes that “while the C2 parity codes are stored in the Nth channel here, **any channel could be used for the parity.**” Ex.1006, Cideciyan, 11:39-42 (citing Ex.1006, Cideciyan, 11:9-14 (“The array of encoded symbols shown in the FIG. 4 is **written in a single stride** as described above with reference to FIG. 3. The process is designed to give a particular configuration of C1 and C1 codewords in individual logical subdivisions (here storage channels 8) and **across the set of subdivisions as a whole.**”); 17:50-55 (“It will be appreciated that **numerous further changes and modifications** can be made to the above embodiments. By way of example, while the logical subdivisions of storage 6 for C2 coding are aligned with the storage channels in SSD 5, in general these subdivisions could be any subset of the available storage.”)).

233. As shown in annotated FIG. 5 below, a POSITA would have understood that one implementation of Cideciyan’s teaching that “**any channel** could be used for the [C2] parity” is programed *across the region* (across the set of subdivisions/across the channels) by using all channels (CH(1) through CH(N)) of the region for the C2 parity symbols, for the benefits of balancing the workload of both reading and writing operations, reducing the burden on any single channel or chip. It was well-known that distributing the C2 parity symbols across all channels

can improve the overall fault tolerance of the SSD. If a failure occurs in one of the channels, the data can still be reconstructed using the parity information from the other channels.

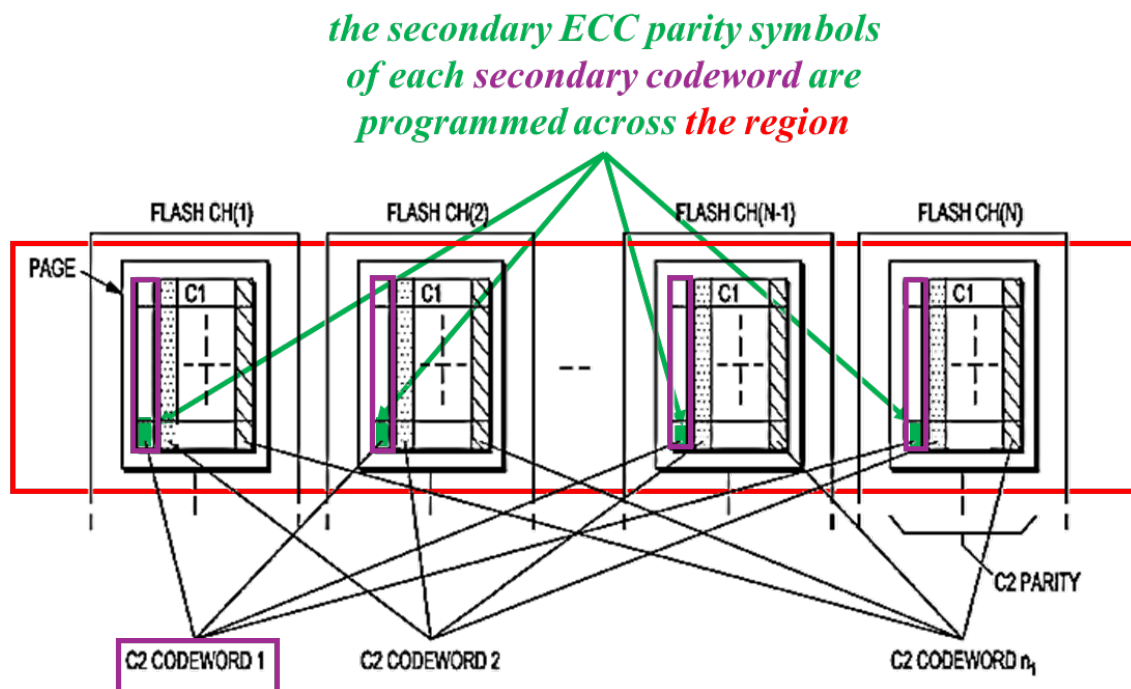


FIG. 5

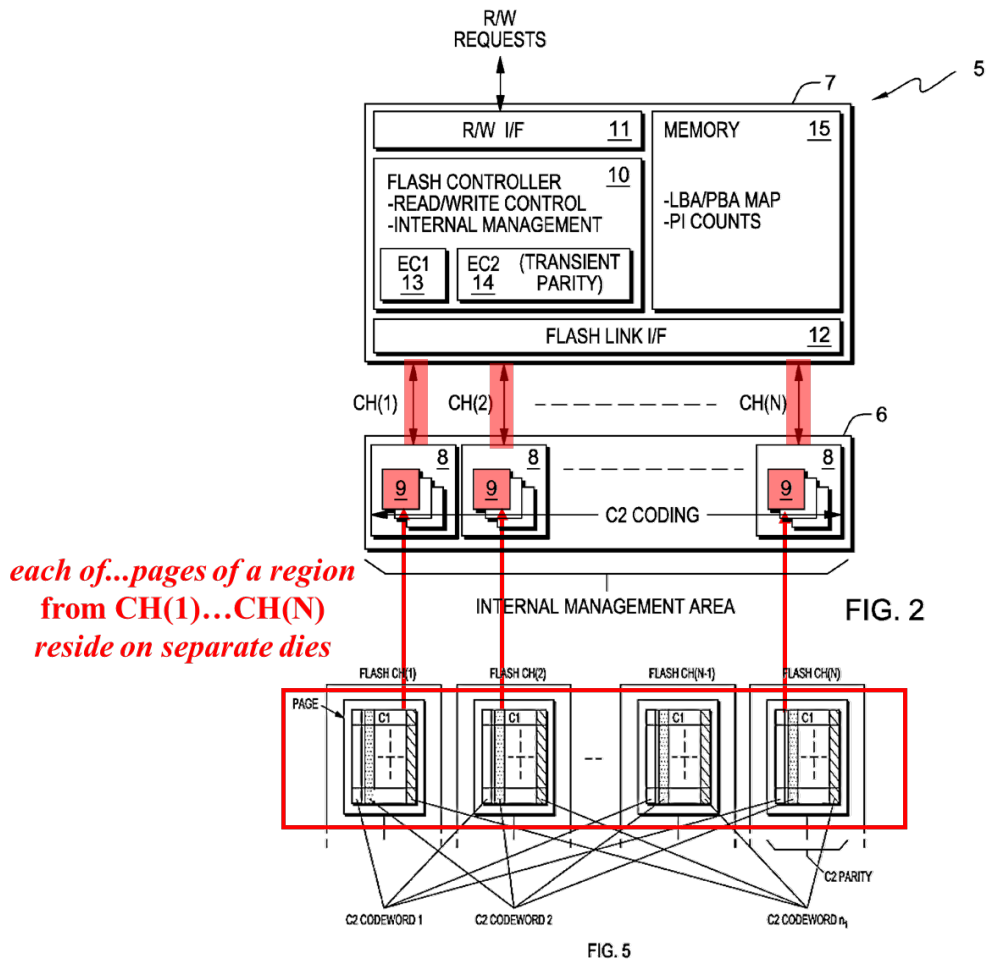
Ex.1006, Cideciyan, FIG. 5, annotated.

[31.3/32.3/33.3] wherein each of the two or more flash memory pages of a region reside on separate dies.

234. The combination of Li and Cideciyan renders obvious
[31.3/32.3/33.3].

235. Continuing with the example of the combination of Li and Cideciyan, Cideciyan describes that each of the two or more flash memory pages of a region reside on separate dies (a flash storage die 9 corresponding to each of the N

channels (CH(1)...CH(N)). Ex.1006, Cideciyan, 8:53-59 (“FIG. 2 is a block diagram of an exemplary SSD 5 embodying the invention showing the main elements involved in the data management operations to be described. The SSD 5 has solid state storage 6 and control apparatus indicated generally at 7. In this example, **storage 6 consists of N storage channels, Ch(1) to Ch(N), each providing a bank 8 of storage units, here flash storage dies 9.”).**



Ex.1006, Cideciyan, FIGS. 2 and 5, annotated.

C. Ground 2: Claims 1-2, 4-5, 14-17, 22-25, and 29-33 are obvious in view of Li, Cideciyan, and Frost.

1. Claim Analysis

236. The claim analysis for Ground 2 is substantially the same as that of Ground 1, with the additional analysis below explaining that Li combined with Cideciyan and Frost renders [1.1.1] of Claim 1 obvious. *See* analysis in Ground 1.

[1.1.1] *determining a bit error rate associated with a region comprising a fixed number of two or more flash memory pages or integer fractions thereof, wherein the two or more flash memory pages of a region can be read simultaneously,*

237. The combination of Li, Cideciyan, and Frost renders obvious [1.1.1].

238. As discussed at Ground 1:[1.1.1], Li alone or in combination with Cideciyan renders obvious that *the two or more flash memory pages of a region* (the group (“stride”) including N pages distributed across N channels) *can be read simultaneously*. Ex.1005, Li, [0111], [0112].

239. To the extent that it is argued that the combination of Li and Cideciyan’s teaching that a group of N pages from across N channels (which is a traditional “page stripe”) that can be accessed in parallel does not explicitly teach that those pages “*can be read simultaneously*,” the further combination with Frost renders it obvious, because the feature that pages from different channels “*can be read simultaneously*” as recited in [1.1.1] is a well-known and conventional implementation detail as taught by Frost.

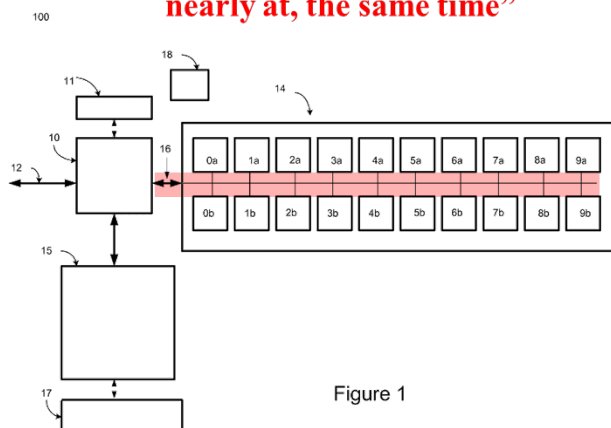
240. Specifically, referring to FIG. 1 reproduced below, Frost describes

that:

each group of two chips forms a **‘Lane,’** also sometimes referred to as a **‘Channel,’**....Because each communication bus 0-9 for a given Lane is independent of the communication buses for the other Lanes, the controller 10 can issue commands and send or **receive data across the various communication buses at the same time** such that the system controller can **access the memory chips corresponding to the individual Lanes at, or very nearly at, the same time.**

Ex.1007, Frost, 6:46-67; *see also* Ex.1007, Frost, 11:51-60 (all of the pages from separate lanes “can be written to or **read from the memory array 14 simultaneously or near-simultaneously,**” which “allows for relatively **quick read and write operations** and allows data to be stored to and retrieved from the memory array 14 **in an efficient and effective manner.**”).

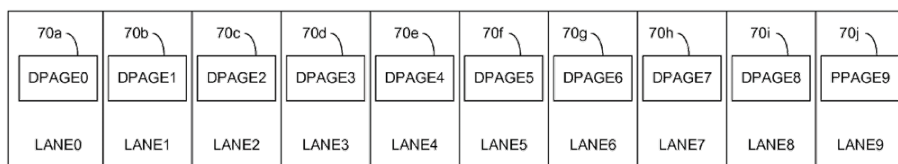
“the system controller can access the memory chips corresponding to the individual Lanes at, or very nearly at, the same time”



Ex.1007, Frost, FIG. 1, annotated.

241. As one example, referring to FIG. 7B below, Frost describes that in its system, all of the pages associated with a page **stripe** can be read simultaneously across the multiple Lanes/Channels. Ex.1007, Frost, 11:51-60.

“all of the Pages associated with Page Stripe 700 can be written to or read from the memory array 14 simultaneously or near-simultaneously.”



Ex.1007, Frost, FIG. 7B, annotated.

242. As such, a POSITA would have been motivated to implement Frost’s teaching for reading the pages of a stripe simultaneously in the combination of Li

and Cideciyan, which provides the benefit of “relatively quick read[] operations.”
of Ex.1007, Frost, 11:51-60.

243. Thus, the combination of Li, Cideciyan, and Frost renders obvious determining a bit error rate associated with a region comprising a fixed number of two or more flash memory pages or integer fractions thereof, as taught by Li and/or in combination with Cideciyan, and renders obvious wherein the two or more flash memory pages of a region can be read simultaneously as taught by Frost.

Reasons to Combine Li, Cideciyan, and Frost

244. First, a POSITA would understand that parallel access (including read operations) in Cideciyan (and Li) would be implemented simultaneously as shown in Frost.

245. Second, it would have been obvious to a POSITA to combine Frost with the combination of Li and Cideciyan, because the combination of Li and Cideciyan teaches a region including pages distributed across multiple channels, and Frost provides the implementation detail of how channels provide simultaneous access to those pages. Applying Frost’s teaching of reading the pages of different channels simultaneously in the combination of Li and Cideciyan would provide the obvious, predictable, and beneficial result of “relatively quick read and write operations” and efficient and effective data storage and retrieval, as taught by Frost.

246. Third, as discussed at [1.1.1], Cideciyan teaches that the region (a group of pages (“stride”)) includes N pages from N channels, and a POSITA would have understood such a group of pages to be a traditional “page stripe.” Frost provides the implementation detail of how pages in a stripe are read simultaneously. *See, e.g.*, Ex.1007, Frost, FIG. 7B. As such, a POSITA would have been motivated to applying Frost’s teaching of implementation detail of a “stripe” in the combination of Li and Cideciyan.

247. Additionally, the results of the proposed combination would have been predictable and a POSITA would have had a reasonable expectation of success, because the combination of Frost with Li and Cideciyan merely applies known technique (channels and/stripes providing simultaneously read of pages) to a known device (flash memory based storage system of the combination of Li and Cideciyan) ready for improvement to yield the predictable results of quick read and write operations.

248. Thus, the combination of Li, Cideciyan, and Frost renders obvious [1.1.1].

D. Grounds 3 and 4: Claims 6-7 and 18 are rendered obvious by each of Combination of Li, Cideciyan, and Leung (Ground 3) and Combination of Li, Cideciyan, Frost, and Leung (Ground 4).

1. Reasons to Combine of Leung with Each of Grounds 1 and 2.

249. Li alone or in combination with Cideciyan and/or Frost, renders

obvious that selecting a gear from the set of predetermined gears for the region, based on the determined bit error rate. Leung provides an obvious way of implementing the selection of the gear, providing that the selection of the gear is based on both *a predetermined acceptable uncorrectable bit error rate (aUBER)* and the bit error rate, because doing so would provide the benefit of allowing the configuration of a predetermined desired outgoing error rate based on design needs, with optimal ECC scheme selection and memory array configuration with suitable code word sizes based on the predetermined desired outgoing error rate. Ex.1008, Table 1 (below, illustrating selecting a gear (corresponding to a row of Table 1) based on different incoming BERs to maintain a predetermined output error rate of 1E-15), [0033] (, “Table 1, below, illustrates how a memory array may be reconfigured for various codeword sizes, depending on the error rate of the memory device”).

TABLE 1

Bit error rate level	Information bits per codeword	Parity per codeword	Codeword size	Information bits per 524 B	Capacity	Code rate	Incoming BER to maintain 1E-15 output error rate
1	512 B	12 B	524 B	512 B	100%	97.7%	3.4E-5
2	250 B	12 B	262 B	500 B	97.7%	95.45	6.3E-5
3	119 B	12 B	131 B	476 B	93.0%	90.8%	11.6E-5

Ex.1008, Leung, Table 1

250. The results of the proposed combinations would have been predictable and a POSITA would have had a reasonable expectation of success, because the combination merely applies known technique (Leung's selecting a gear based on a predetermined acceptable uncorrectable bit error rate (aUBER), in addition to the bit error rate) to a known device (the flash memory based storage system implementing adaptive ECC schemes as provided by the combination of Li and Cideciyan and the combination of Li and Cideciyan and Frost) ready for improvement to yield the predictable results of providing a predetermined desired outgoing error rate based on design needs.

2. Claim 6

[6.0] *The method of claim 1, wherein selecting further comprises selecting the gear that has the highest data payload size and sufficient correction capability for the determined bit error rate such that memory reads from the region can be corrected with no more than a predetermined acceptable uncorrectable bit error rate (aUBER).*

251. Leung as combined with Grounds 1 and 2 each renders obvious [6.0].

252. First, Li teaches or suggests selecting the gear that has the highest data payload size and sufficient correction capability for the determined bit error rate based on both **a predetermined acceptable uncorrectable bit error rate (aUBER)**, and the predetermined raw BER, because of its suggestion to provide more “[s]pace available to store data.” Ex.1005, Li, [0104].

253. Second, Leung describes choosing a gear (operating at bit error levels

1, 2, or 3) with the lowest error correction capability for satisfying the output error rate requirement. Specifically, the rows corresponding to bit error levels 1 through 3 have increasing correction capability, with decreasing code rate. Ex.1008, Leung, Table reproduced below.

TABLE 1

Bit error rate level	Information bits per codeword	Parity per codeword	Codeword size	Information bits per 524 B	Capacity	Code rate	Incoming BER to maintain 1E-15 output error rate
1	512 B	12 B	524 B	512 B	100%	97.7%	3.4E-5
2	250 B	12 B	262 B	500 B	97.7%	95.45	6.3E-5
3	119 B	12 B	131 B	476 B	93.0%	90.8%	11.6E-5

Ex.1008, Leung, FIG. 1.

254. Leung describes choosing the lowest gear (e.g., choosing bit error rate level 1 for incoming BER 3.4E-5), which has the lowest error correction capability among the three gears, where all three gears satisfy the output error rate requirement (e.g., 1E-15). Ex.1008, Leung, [0034] (“Initially, data may be read and written using **bit error rate level 1**. When the detected error rate approaches 3.4e-5, the device may attempt to change codeword sizes and operate at **bit error level 2**. ... When the detected error rate approaches 6.3e-5, the device may switch to operate at **bit error level 3**.”).

255. As discussed at IX.D.1, a POSITA would have implemented Leung’s teaching in the combination of Li, Cideciyan, and Frost for maximizing data

payload size while satisfying the aUBER requirement.

256. Thus, the combination of Li, Cideciyan, and Leung (Ground 3) and the combination of Li, Cideciyan, Frost, and Leung (Ground 4) each render obvious [6.0].

3. Claim 7

[7.0] *The method of claim 6, wherein the aUBER is 1E-16.*

257. Leung as combined with Grounds 1 and 2 each renders obvious [7.0].

258. Specifically, Leung describes that the aUBER (outgoing error rate) is 1E-16. Ex.1008, Leung, FIG. 2, [0035] (“a graph (200) that illustrates the **outgoing error rate** as a function of code rate at various incoming bit error rates,” where the outgoing error rates include 1E-16).

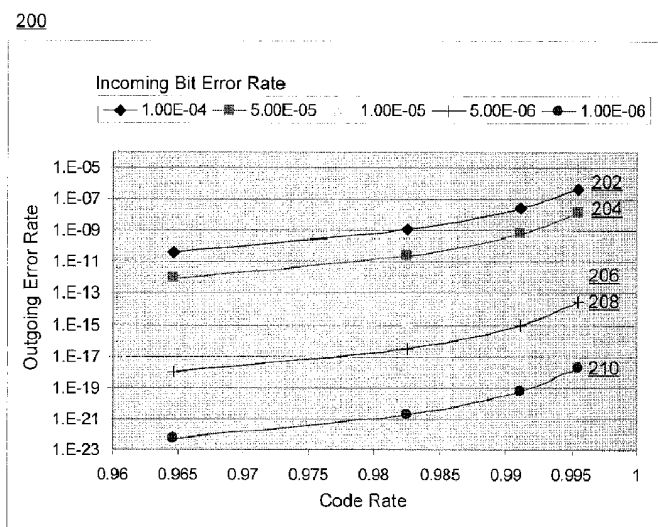


Fig. 2

259. Thus, the combination of Li, Cideciyan, and Leung (Ground 3) and

the combination of Li, Cideciyan, Frost, and Leung (Ground 4) each render obvious [7.0].

4. Claim 18

[18.0] *The apparatus of claim 15, wherein the processor is further configured to select a gear that has the highest data payload size and sufficient correction capability for the determined bit error rate such that memory reads from the region can be corrected with no more than a predetermined acceptable uncorrectable bit error rate (aUBER).*

260. Leung as combined with Grounds 1 and 2 each renders obvious [18.0].

261. First, Leung as combined with Grounds 1 and 2 renders obvious selecting a gear as recited for reasons discussed at Grounds 3 and 4:[6.0].

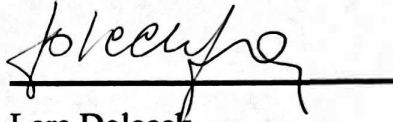
262. Second, Li combined with Cideciyan and Leung renders that the “processor” is configured to perform selected functions including selecting a gear function. Ex.1005, Li, [0137]; [0131]; see also Ex.1006, Cideciyan, 9:29-31.

263. Accordingly, Li combined with Cideciyan and Leung renders obvious [18.0].

X. DECLARATION

263. I declare that all statements made herein of my own knowledge are true, that all statements made on information and belief are believed to be true, and that these statements were made with knowledge that willful false statements so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code.

Dated: January 29, 2025


Lara Dolecek