

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

PHISON ELECTRONICS CORPORATION,

Petitioner

v.

VERVAIN, LLC,

Patent Owner

U.S. Patent No. 11,967,369

Post-Grant Review Case No. PGR2025-00010

DECLARATION OF CARL SECHEN, PH.D.

TABLE OF CONTENTS

I. OVERVIEW:	1
II. BACKGROUND AND QUALIFICATIONS:	4
III. MY UNDERSTANDING OF THE APPLICABLE LAW	8
IV. DOCUMENTS CONSIDERED IN FORMULATING MY OPINIONS: ..	12
V. RELEVANT STATE OF THE ART AS OF JULY 19, 2011	17
A. NAND FLASH MEMORY AND OPERATIONS	20
B. MLC AND SLC	45
C. HYBRID USE OF MLC AND SLC MEMORY	56
D. USE OF MLC MEMORY IN “SLC MODE” (“PSEUDO-SLC”)	62
E. “DATA INTEGRITY CHECKING” IN NAND FLASH MEMORY SYSTEMS	66
F. STRATEGIES FOR NAND FLASH MEMORY REMAPPING TO MANAGE SYSTEM RELIABILITY AND EFFICIENCY	78
VI. PERSON OF ORDINARY SKILL IN THE ART (POSITA)	82
VII. DISCLOSURE OF THE ‘369 PATENT SPECIFICATION	83
A. ‘369 SPECIFICATION BACKGROUND	84
B. PROFFERED EMBODIMENTS IN THE DETAILED DESCRIPTION	90
VIII. THE 2018 ABSTRACTION OF THE DISCLOSED SYSTEM	108
A. PRE-2018 PHYSICAL MLC AND SLC MODULES AND COUPLED CONTROLLER DEFINED BY L2P, DATA INTEGRITY TEST, AND WEAR LEVELING FUNCTIONS, WITH PREFERENTIAL TRANSFER TO SLC	109
B. 2018 ABSTRACTION OF “MEMORY SPACE” AND “MAXIMIZING LIFETIME.”	113
C. FURTHER CONFUSION OF THE CLAIMED SYSTEM IN THE ‘369 PATENT	123

D. THE CHALLENGED CLAIMS.....	127
E. CLAIM CONSTRUCTION.	128
IX. LACK OF ENABLEMENT AND DEFINITENESS OF THE ‘369 CLAIMS	132
X. PATENTABILITY OF THE ‘369 CLAIMS RELATIVE TO THE PRIOR ART	141
A. CLAIM 1 IS OBVIOUS OVER GAVENS, INCLUDING INCORPORATED REFERENCES, IN VIEW OF KNOWLEDGE OF THE POSITA.	141
1. [1Pre] A system for storing data comprising:.....	144
2. [1a][1a1] memory space containing logical and physical volatile memory space and nonvolatile memory space, [1a2] wherein the logical and physical nonvolatile memory space includes both multi-level cell (MLC) logical and physical memory space and single level cell (SLC) logical and physical memory space;	144
3. [1b] a bank of physical nonvolatile memory, including:.....	146
[1b1] a plurality of physical MLC memory modules, each including at least one physical MLC nonvolatile memory element having a physical address within the physical MLC nonvolatile memory space that can be mapped to a logical address in the MLC logical nonvolatile memory space, and.....	146
[1b2] at least one SLC memory module including at least one physical SLC nonvolatile memory element having a physical address within the physical SLC nonvolatile memory space that can be mapped into the SLC logical nonvolatile memory space;	146
4. [1c] at least one random access volatile physical memory element that exists in the logical volatile memory space;	148
5. [1d] at least one controller [1d1] to operate physical memory elements and associated memory space for associated Write access operations to the physical memory elements and Read access operations from the physical memory elements, [1d2] the controller having a controller memory associated therewith for storing received data therein;.....	150

6.	[1e] a flash translation layer (FTL), wherein the at least one controller, or the FTL, or a combination of both maintain an address table in one or more of the physical memory elements;	151
7.	[1f] [1f1] the controller controlling access of the physical MLC and SLC nonvolatile memory elements and the random access volatile physical memory elements for storage of the received data therein as stored data, [1f2] the controller, in at least a Write access operation transferring the stored received data from the controller memory to a given one of the physical MLC nonvolatile memory elements in an associated physical MLC memory module, [1f3] operable to store the received data in the given one of the physical MLC nonvolatile memory elements as stored data and retain such received data in the random access volatile physical memory element as retained data associated with stored data;	155
8.	[1g] the controller [1g1] performing a data integrity test on stored data in the given one of the physical MLC nonvolatile memory elements in the associated one of the physical MLC memory modules after the at least a Write access operation performed thereon by [1g2] reading the stored data to the controller memory and [1g3] comparing the stored data in the controller memory in the given one of the physical MLC nonvolatile memory elements to the retained data that was associated with the stored data in the random access volatile physical memory element by the controller during the Write access operation;	157
9.	[1h] wherein the address table maps logical addresses to physical addresses of the memory system, wherein the mapping is performed as necessitated by the memory system to maximize lifetime; and.....	159
10.	[1i] wherein a failure of the data integrity test performed on the stored data by the controller results in [1i1] a remapping of logical address space to a different physical range of addresses in the physical address space and [1i2] transfer of data corresponding to the retained data to those remapped physical addresses in the physical address space from those physical addresses in the physical address space determined to have failed the data integrity test.....	159
B. CLAIM 2 IS OBVIOUS OVER GAVENS, INCLUDING INCORPORATED REFERENCES, IN VIEW OF KNOWLEDGE OF THE POSITA.		161
1.	[2Pre] The system of claim 1, ...	161

2. [2] wherein the FTL flash translation layer is a software module, or a firmware module containing software updates.....	161
C. CLAIM 3 IS OBVIOUS OVER GAVENS, INCLUDING INCORPORATED REFERENCES, IN VIEW OF KNOWLEDGE OF THE POSITA.	162
1. [3Pre] The system of claim 1,.....	162
2. [3] wherein at least one of the random access volatile memory or the MLC and SLC nonvolatile memory elements are embedded in the at least one controller.	162
D. CLAIM 4 IS OBVIOUS OVER GAVENS, INCLUDING INCORPORATED REFERENCES, IN VIEW OF KNOWLEDGE OF THE POSITA.	163
1. [4Pre] The system of claim 1,.....	163
2. [4] wherein the physical MLC and SLC nonvolatile memory elements comprise flash memory.....	163
E. CLAIM 5 IS OBVIOUS OVER GAVENS, INCLUDING INCORPORATED REFERENCES, IN VIEW OF KNOWLEDGE OF THE POSITA.	164
1. [5Pre] The system of claim 1,.....	164
2. [5] wherein the random access volatile memory is dynamic random access memory.	164
F. CLAIM 6 IS OBVIOUS OVER GAVENS, INCLUDING INCORPORATED REFERENCES, IN VIEW OF KNOWLEDGE OF THE POSITA.	165
1. [6Pre] The system of claim 1,.....	165
2. [6] wherein the random access volatile memory is static random access memory.	165
G. CLAIM 7 IS OBVIOUS OVER GAVENS, INCLUDING INCORPORATED REFERENCES, IN VIEW OF KNOWLEDGE OF THE POSITA.	166
1. [7Pre] The system of claim 1,.....	166
2. [7] wherein the at least one physical SLC nonvolatile memory element includes a hard disk drive (HDD).	166

H. CLAIM 8 IS OBVIOUS OVER GAVENS, INCLUDING INCORPORATED REFERENCES, IN VIEW OF KNOWLEDGE OF THE POSITA.	167
1. [8Pre] The system of claim 1, ...	167
2. [8] wherein the at least one controller is associated with a processor external to the memory system.	167
I. CLAIM 9 IS OBVIOUS OVER GAVENS, INCLUDING INCORPORATED REFERENCES, IN VIEW OF KNOWLEDGE OF THE POSITA.	167
1. [9Pre] The system of claim 1, ...	167
2. [9] wherein the remapped physical addresses in the physical address space are located in the SLC memory module.	167
J. CLAIM 10 IS OBVIOUS OVER GAVENS, INCLUDING INCORPORATED REFERENCES, IN VIEW OF KNOWLEDGE OF THE POSITA.	168
1. [10Pre] The system of claim 1, ...	168
2. [10] wherein the FTL is associated with the MLC memory modules and the SLC memory modules in the bank of physical nonvolatile memory.	168

I, Carl Sechen, Ph.D., declare as follows.

I. Overview:

1. I am over the age of eighteen and otherwise competent to make this Declaration. I have been retained as an expert witness on behalf of petitioner Phison Electronics Corporation (“Petitioner”) for the above-captioned post grant review. I am being compensated for my time in connection with this review at my standard consulting rate, which is \$450.00 per hour. I have no personal or financial interest in the outcome of this proceeding. I make this Declaration on personal knowledge and on my expertise; where I state understandings with respect to patent law or procedure, the basis generally is discussion with counsel for Petitioner.

2. I have been asked to provide my expert opinions on a petition for post grant review (“PGR”) of U.S. Patent No. 11,967,369, entitled “LIFETIME MIXED LEVEL NON-VOLATILE MEMORY SYSTEM,” to Dr. G.R. Mohan Rao (the “‘369 patent”) (Ex. 1001). The ‘369 patent issued on Apr. 23, 2024, from U.S. Appl. No. 18/387,546, which was filed Nov. 7, 2023, as a continuation of U.S. Appl. No. 17/203,385 (now U.S. Patent No. 11,830,546, the “‘546 patent,” Ex. 1013), which was a continuation of U.S. Appl. No. 16/006,299, which was filed Jun. 12, 2018 (now U.S. Patent No. 10,950,300, the “‘300 patent,” Ex. 1011), as a continuation of U.S. Appl. No. 14/950,553, which was filed Nov. 24, 2015 (now U.S. Patent No.

9,997,240, the “’240 patent,” Ex. 1009), as a continuation of U.S. Appl. No. 14/525,411, filed on Oct. 28, 2014 (now U.S. Patent No. 9,196,385, the “’385 patent,” Ex. 1007), as a divisional of U.S. Appl. No. 13/455,267, filed Apr. 25, 2012 (now U.S. Patent No. 8,891,298, the “’298 patent,” Ex. 1005). All the patents incorporated by reference Dr. Rao’s U.S. Patent No. 7,855,916,¹ and claimed

¹ (’369 1:29-37, “Rao ’916,” Ex. 1004.) Rao ’916, entitled “NONVOLATILE MEMORY SYSTEMS WITH EMBEDDED FAST READ AND WRITE MEMORIES,” incorporated along with U.S. Patent No. 8,194,452 to Dr. Rao of the same title and specification (’369 1:29-37). The ’298, ’385, ’240, ’300 and ’369 patents, along with three other patents, U.S. Patent Nos. 11,830,546, 11,854,612, and 11,967,370, all based on the same specification (collectively, the “Lifetime Patents”) are currently asserted in *Vervain, LLC v. Phison Electronics Corp.*, No. 1-24-cv-259 (W.D. Tex.). My declarations were submitted in support of pending petitions in *Phison Electronics Corp. v. Vervain, LLC*, PGR2024-00047 (’546), IPR2024-00048 (’612), IPR2025-00212 (’298), IPR2025-00213 (’385), IPR2025-00214 (’240), and IPR2025-00215 (’300).

priority from and incorporated by reference U.S. Provisional App. No. 61/509,257,² filed on July 19, 2011 (“Provisional,” Ex. 1003). I understand that, according to USPTO records, the ‘369 patent is currently assigned to Vervain, LLC (“Patent Owner”).

3. I understand that, based on the Provisional filing date, the prior art references and state of the art which I should consider in evaluating the ‘369 patent’s patentable inventiveness (to the extent disclosed in the Provisional), should be July 18, 2011, or earlier.

4. In preparing this Declaration, I have reviewed the ‘369 patent and have considered each of the documents cited herein upon which I specifically rely. In formulating my opinions, I have relied upon my experience, education, and knowledge in the relevant art, including other background documents. I have also considered the viewpoint of a person of ordinary skill in the art (“POSITA”) of computer memories (as described below) at the time of the alleged invention of the ‘369 patent, that is, close to, but earlier than July 19, 2011.

² (‘369 1:25-28 [also incorporated by reference].) The Provisional was actually entitled, “IMPROVED LIFETIME MIXED LEVEL NON-VOLATILE MEMORY SYSTEM” (Ex. 1003 at 3, compare omission of “improved” in ‘369 title).

5. This Declaration is limited to my opinions on invalidity with respect to claims 1-10 of the '369 patent (the "Challenged Claims"). Those opinions, set forth in detail below, including factual bases and assumptions, are summarized as: The Challenged Claims, in rewriting ambiguously and incompletely described embodiments, which assume known structure and operations of two distinct types of NAND flash memory, "SLC" and "MLC," are unenabled and indefinite. To the extent that they are enabled and understood, the Challenged Claims are obvious over U.S. Patent No. 8,634,240 ("Gavens," Ex. 1045), including incorporated references, in view of the knowledge of a POSITA of the extensive background technology for joint operation of SLC and MLC memory, including U.S. Patent Appl. Pub. No. 2009/0327591 ("Moshayedi," Ex. 1043) and 2008/0140918 ("Sutardja," Ex. 1042).

6. While Gavens, its incorporated references, and other references cited below as background technology, including Moshayedi and Sutardja, form the primary bases for my obviousness opinion, I note that there are many other relevant references which further support a finding of invalidity as discussed herein. I reserve the possibility of amending or supplementing this Declaration should additional information or arguments develop or be presented which affect my opinions.

II. Background and Qualifications:

7. My qualifications and credentials are set forth herein as further evidenced by my curriculum vitae (addendum hereto).

8. I earned a B.E.E. in Electrical Engineering from the University of Minnesota in 1975, followed by an M.S. in Electrical Engineering from the Massachusetts Institute of Technology in 1977. I earned a Ph.D. in Electrical Engineering from the University of California, Berkeley in 1986.

9. I have been a Professor of Electrical Engineering for 38 years. Since August 15, 2005, I have been a Professor of Electrical and Computer Engineering at the University of Texas at Dallas. From July 1992 to August 14, 2005, I served as a Professor of Electrical Engineering at the University of Washington. From July 1986 through June 1992, I served as an Assistant Professor and then Associate Professor of Electrical Engineering at Yale University.

10. Over these years, my research has focused on the design and computer-aided design of digital integrated circuits, including the design of dynamic random-access memory (“DRAM”) and static random-access memory (“SRAM”) modules. I have also designed several chips that included various types of embedded DRAM (eDRAM). In this research work, I have designed multiple sense amplifier circuits for the various DRAMs, embedded DRAMs, and SRAMs. I have authored or coauthored over 200 papers and one book, the majority of which concern digital integrated circuit design and memory design.

11. As a professor, I have developed and taught numerous courses, including several courses that teach digital integrated circuit design and memory

design, including extensive coverage of flash memory design. I have taught these courses continuously for the past 29 years. I have taught digital integrated circuit design and memory design, including flash memory design, to undergraduate and graduate students at the University of Washington and at the University of Texas at Dallas. DRAM and embedded DRAM design is covered in these courses, as is the design of sense amplifier circuits for various types of memory. NAND-Flash memory design is taught in detail, including both SLC (single-level cell) and MLC (multi-level cell) variants.

12. I was elected a Fellow of the Institute of Electrical and Electronics Engineers (“IEEE”) in 2002 for contributions to placement and routing of ASICs. IEEE is the leading professional association for electrical engineers. The Board of Directors of the IEEE awards the rank of “Fellow” to individuals with an extraordinary record of accomplishments in any of the IEEE fields of interest. The total number of IEEE members who can be named Fellows in any one year cannot exceed one-tenth of one percent of the total voting IEEE membership.

13. I received several research and teaching awards during my career. I received the Semiconductor Research Corporation’s Inventor’s Recognition Award in 1988 and in 2001. I also received the Technical Excellence Award from the Semiconductor Research Corporation in 1994. While serving as Professor at the University of Washington, I received the Outstanding Research Advisor award from

the Department of Electrical Engineering in 2002. In 2008, I received the Distinguished Teacher of the Year Award from the Department of Electrical and Computer Engineering (ECE) at the University of Texas at Dallas. I also received the Distinguished Teaching Award for the Erik Johnson School of Engineering and Computer Science in 2014. In addition, I received the Graduate Teaching Excellence Award from the ECE Department at UT Dallas, May 2024.

14. Over the years, I have also received funding to conduct research in computer circuits and memory designs, including area-efficient and reliable embedded DRAM and SRAM design. Together with my graduate students, I have designed and fabricated various types of computational VLSI chips.

15. I am a co-inventor on four issued patents directed to transistor and computational logic technologies. Three of the patents concern transistor-level programmable logic.

16. In summary, I am an expert in the field of computer architecture, including memory design and management, generally and as applied within the context of the Lifetime Patents, including NAND flash and other non-volatile memories (“NVM”). As set forth above, my experience includes practical product development, including performance considerations and design choices involving the memory technology state of the art before the earliest, July 19, 2011, priority date applicable to the challenged claims.

17. As an expert, I am qualified to evaluate, interpret and rely on reports, patents, scholarly papers and professional opinions of others, within my areas of expertise, as to authenticity and accuracy. Moreover, I am qualified to provide an opinion as to what a person of ordinary skill in the art (POSITA) would have understood, known and concluded before July 19, 2011, including the context in which a POSITA would understand and construe teachings and terminology in written publications.

III. My Understanding of the Applicable Law

18. I am not an attorney. However, as outlined in my curriculum vitae addended hereto, I have formed and presented opinions according to my technology expertise and hypotheticals presented to me in other cases involving validity of United States patents, including for the defendant in *Vervain, LLC v. Western Digital Corp.*, No. 6:21-cv-00488 (W.D. Tex. filed May 10, 2021, settled Aug. 8, 2023), which asserted the four earliest Lifetime Patents, identified in paragraph 2 above, based on the same specification as the '369 patent. In those cases, as here, I was informed or reminded of the legal standards which are relevant to my analysis and opinions. I summarize here the understandings relevant for this declaration, noting that greater detail may be provided upon any cross-examination, and others may be provided for any amendment or supplementation reserved in paragraph 6 above.

19. **Written Description and Claiming Requirements.** I understand that, as part of the bargain with the public of providing information to advance “science and the useful arts” in return for patent rights claimed and granted with notice to the public, 35 U.S.C. § 112 requires, in subsection (a), an adequate written description that both enables a POSITA to practice, without undue experimentation, the invention as claimed and reasonably conveys to a POSITA that “the inventor had possession of the claimed subject matter as of the filing date” and, in subsection (b), that a patent’s claims be “concise” and, viewed in light of the specification and prosecution history, inform a POSITA about “the scope of the invention with reasonable certainty.” I also understand that, applied with other statutory requirements, there are instances in which (1) claim elements that are defined by a function may be found to be indefinite if there is inadequate structure disclosed for performing the function, (2) mixed product and process claims may be invalid for inadequate notice to patent-respecting manufacturers or users, and (3) a claim may be invalid for failing to recite a “useful” invention if it is manifestly inoperative.

20. **What Constitutes Prior Art that May Invalidate a Patent.** I understand that former 35 U.S.C. § 102 listed conditions, including what is considered “prior art,” that may preclude (or invalidate) a patent of the ‘369 patent’s vintage, the most pertinent here being:

(1) under subsection (a), if the invention was known or used by others in this country, or was patented or described in a printed publication in this or a foreign country, before the date of the invention of the asserted patent; or

(2) under subsection (b), if the invention was patented or described in a printed publication in this or a foreign country, or in public use or on sale in this country, more than one year before the effective filing date of the asserted patent; or

(3) under subsection (e), if the invention was described in (a) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (b) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent; or

I understand that one or more of these conditions apply to the art relied upon here for my invalidity analysis.

21. How the Prior Art Is Applied in the Invalidity Analysis:

Anticipation: I understand that a patent claim may be found to be invalid as “anticipated” under a subsection of 35 U.S.C. § 102 if a single prior art document (reference) (including any other prior art incorporated by reference in the prior art document) under that subsection describes every element of the claim explicitly or inherently (necessarily present) as understood by a POSITA to allow making and using the claimed invention.

22. How the Prior Art Is Applied in the Invalidity Analysis:

Obviousness: I understand that a patent claim may be invalid as “obvious” under 35 U.S.C. § 103 if the differences between the claimed invention and the prior art (including any other prior art incorporated by reference in the prior art document) are such that the claimed invention, as a whole, would have been obvious to a POSITA before the effective filing date of the claimed invention. The analysis calls for consideration of (a) level of ordinary skill in the pertinent art; (b) the scope and content of the prior art; (c) the differences between the prior art as a whole and the claim at issue; and (d) as appropriate, objective indicia of non-obviousness. The objective indicia, appropriate if there is a nexus with the claimed invention, include: (1) commercial success attributable to the invention; (2) long-felt but unsolved need for the invention; (3) failure of others; (4) skepticism by experts; (5) praise by others; (6) teaching away by others; (7) recognition of a problem which the claimed invention addresses; (8) copying by others; and (9) unexpected results.

23. I understand that obviousness may be found based on a single reference (and any of its incorporated references) considered in view of the knowledge of a POSITA, and that a patent that incorporates another document by reference includes the disclosure of the incorporated document as if it was explicitly contained in the patent. Where multiple references are proffered, I understand that while there is no requirement of an explicit teaching, suggestion or motivation to combine, generally

there needs to be put forth some reason why a POSITA (if only by common sense or ordinary creativity) would have made the particular combination and that there is a reasonable expectation of success of the combination.

IV. Documents Considered in Formulating My Opinions:

24. In formulating my opinions, I have considered all documents cited in this Declaration, including, in particular, the documents in the chart below:

Exhibit	Description
1001	U.S. Patent No. 11,967,369, titled “LIFETIME MIXED LEVEL NON-VOLATILE MEMORY SYSTEM,” to Rao
1003	U.S. Provisional Application No. 61/509,257, titled “IMPROVED LIFETIME MIXED LEVEL NAND FLASH SYSTEM” by Rao
1004	U.S. Patent No. 7,855,916, titled “NONVOLATILE MEMORY SYSTEMS WITH EMBEDDED FAST READ AND WRITE MEMORIES,” to Rao
1005	U.S. Patent No. 8,891,298, titled “LIFETIME MIXED LEVEL NON-VOLATILE MEMORY SYSTEM,” to Rao
1006	Excerpts from prosecution history of ’298 patent
1007	U.S. Patent No. 9,196,385, titled “LIFETIME MIXED LEVEL NON-VOLATILE MEMORY SYSTEM,” to Rao
1008	Excerpts from prosecution history of ’385 patent
1009	U.S. Patent No. 9,997,240, titled “LIFETIME MIXED LEVEL NON-VOLATILE MEMORY SYSTEM,” to Rao
1010	Excerpts from prosecution history of ’240 patent

1011	U.S. Patent No. 10,950,300, titled “LIFETIME MIXED LEVEL NON-VOLATILE MEMORY SYSTEM,” to Rao
1012	Excerpts from prosecution history of ‘300 patent
1013	U.S. Patent No. 11,830,546, titled “LIFETIME MIXED LEVEL NON-VOLATILE MEMORY SYSTEM,” to Rao
1025	Excerpts from Rino Micheloni, et al., <i>INSIDE NAND FLASH MEMORIES</i> (Springer 2010)
1026	Excerpts from <i>NONVOLATILE MEMORY TECHNOLOGIES WITH EMPHASIS ON FLASH: A COMPREHENSIVE GUIDE TO UNDERSTANDING AND USING NVM DEVICES</i> (Joe E. Brewer & Manzur Gill eds. Wiley-IEEE Press 2008)
1027	Excerpts from <i>MICROSOFT COMPUTER DICTIONARY</i> (5 th ed. 2002)
1030	Yu Cai et al., <i>Program Interference in MLC NAND Flash Memory: Characterization, Modeling and Mitigation</i> (IEEE 2013)
1031	Kang-Deog Suh et al., <i>A 3.3 V 32 Mb NAND Flash Memory with Incremental Step Pulse Programming Scheme</i> , 30 IEEE J. Solid State Circuits 1149 (Nov. 1995)
1032	Eitan Yaakobi et al., <i>Error Characterization and Coding Schemes for Flash Memories</i> (2010 IEEE Globecom Workshops, Miami, FL at 1856-60)
1033	Evans, Chris, <i>Enterprise MLC: How flash vendors are boosting MLC write endurance</i> (June 3, 2011)
1034	Nelson Duann, <i>SLC & MLC Hybrid</i> (Silicon Motion Flash Memory Summit, Santa Clara, CA, Aug. 12, 2008)
1035	Ashish Jagmohan et al., <i>Adaptive Endurance Coding for NAND Flash</i> (2010 IEEE Globecom Workshops, Miami, FL at 1841-45)

1036	Sungjin Lee et al., <i>FlexFS: A Flexible Flash File System for MLC NAND Flash Memory</i> (2009 USENIX Annual Technical Conference, San Diego, CA, June 14-19, 2009)
1037	Minyoung Sung & Kanghee Kim, <i>Asymmetric Flash Volume Management</i> , 58 <i>IEEE Transactions on Consumer Electronics</i> 455 (May 2012)
1038	Greg Atwood et al., Intel StrataFlash™ <i>Memory Technology Overview</i> , Intel Technology J. (Q4 1997)
1039	Taehee Cho et al., “A 3.3 V 1 Gb multi-level NAND flash memory with non-uniform threshold voltage distribution,” <i>2001 IEEE International Solid-State Circuits Conference. Digest of Technical Papers</i> , ISSCC (Cat. No.01CH37177) (2001)
1040	U.S. Patent No. 6,456,528, titled for “SELECTIVE OPERATION OF A MULTI-STATE NON-VOLATILE MEMORY SYSTEM IN A BINARY MODE” to Chen
1041	U.S. Patent No. 8,078,794, titled “HYBRID SSD USING A COMBINATION OF SLC AND MLC FLASH MEMORY ARRAYS,” to Lee et al.
1042	U.S. Patent Appl. Pub. No. 2008/0140918, titled “HYBRID NON-VOLATILE SOLD STATE MEMORY SYSTEM” by Sutardja
1043	U.S. Patent Appl. Pub. No. 2009/0327591, titled “SLC-MLC COMBINATION FLASH STORAGE DEVICE,” by Moshayedi
1044	U.S. Patent Appl. Pub. No. 2010/0172179, titled “SPARE BLOCK MANAGEMENT OF NON-VOLATILE MEMORIES,” by Gorobets et al.
1045	U.S. Patent No. 8,634,240, titled “NON-VOLATILE MEMORY AND METHOD WITH ACCELERATED POST-WRITE READ TO MANAGE ERRORS,” to Gavens et al.

1046	U.S. Patent No. 8,806,301, titled “DATA WRITING METHOD FOR A FLASH MEMORY, AND CONTROLLER AND STORAGE SYSTEM USING THE SAME,” to Yu et al.
1047	U.S. Patent Appl. Pub. No. 2010/0115192, titled “WEAR LEVELING METHOD FOR NON-VOLATILE MEMORY DEVICE HAVNG SINGLE AND MULTI LEVEL MEMORY CELL BLOCKS,” by Lee
1048	U.S. Patent Appl. Pub. No. 2011/0271043, titled “SYSTEM AND METHOD FOR ALLOCATING AND USING SPARE BLOCKS IN A FLASH MEMORY,” to Segal et al.
1049	U.S. Patent Appl. Pub. No. 2011/0153912, titled “MAINTAINING UPDATES OF MULTI-LEVEL NON-VOLATILE MEMORY IN BINARY NON-VOLATILE MEMORY,” by Gorobets et al.
1050	U.S. Patent No. 5,930,167, titled “MULTI-STATE NON-VOLATILE FLASH MEMORY CAPABLE OF BEING ITS OWN TWO STATE WRITE CACHE,” to Lee et al.
1051	U.S. Patent Appl. Pub. No. 2010/0172180, titled “NON-VOLATILE MEMORY AND METHOD WITH WRITE CACHE PARTITIONING,” by Paley et al.
1052	U.S. Patent No. 8,225,035, titled “SYSTEMS AND METHODS FOR OPERATING A DISK DRIVE,” to Colligan
1056	U.S. Patent App. Pub. No. 2009/0172267, “REFRESH METHOD OF A FLASH MEMORY,” by Oribe et al.
1057	U.S. Patent No. 5,936,971, titled “MULTI-STATE FLASH EEPROM SYSTEM WITH CACHE MEMORY, to Harari et al.
1058	PCT Pub. No. WO 03/027828, “METHOD OF WRITING DATA TO NON-VOLATILE MEMORY,” by Gorobets et al.

1061	Duane H. Oto et al., <i>High-Voltage Regulation and Process Considerations for High-Density 5 V-Only E²PROM's</i> , SC-18 IEEE J. Solid State Circuits 532 (Oct. 1983)
1062	Gheorghe Samachisa et al., <i>A 128K Flash EEPROM Using Double-Polysilicon Technology</i> , SC-22 IEEE J. Solid State Circuits 676 (Oct. 1987)
1063	Masaki Momodomi et al., <i>An Experimental 4-Mbit CMOS EEPROM with a NAND-Structured Cell</i> , 24 IEEE J. Solid State Circuits 1238 (Oct. 1989)
1064	Tae-Sun Chung et al., <i>A survey of Flash Translation Layer</i> , 55 J. Systems Architecture 332-43 (2009)
1065	Yoshiba Iwata et al., <i>A High-Density NAND EEPROM with Block-Page Programming for Microcomputer Applications</i> , 25 IEEE J. Solid-State Circuits 417 (Apr. 1990)
1066	Simona Boboila and Peter Desmoyers, <i>Write Endurance in Flash Drives: Measurements and Analysis</i> , Usenix Conference (Feb. 2010)
1067	SiliconSystems, <i>Increasing Flash SSD Reliability</i> , StorageSearch.com (April 2005)
1068	Taehee Cho et al., <i>A Dual-Mode NAND Flash Memory: 1-Gb Multilevel and High-Performance 512-Mb Single Level Modes</i> , 36 IEEE J. Solid State Circuits 1700 (Nov. 2001)
1069	Ken Takeuchi, <i>A Multipage Cell Architecture for High-Speed Programming Multilevel NAND Flash Memories</i> , 33 IEEE J. Solid-State Circuits 1228 (Aug. 1998)
1070	Garth Goodson & Rahul Iyer, <i>Design Tradeoffs in a Flash Translation Layer</i> , HPCA [High Performance Computer Architecture] WEST 2010 Conference, Bangalore, India

V. Relevant State of the Art as of July 19, 2011

25. I describe here the relevant state of the art prior to July 19, 2011, of NAND flash memory technology, which is the only non-volatile memory (“NVM”) explained in any detail in the ‘369 patent. As shown at Section VI below, the ‘369 specification, common to all the Lifetime Patents, only described a known NAND flash memory system composed of three components: (1) a bank of multiple physical modules of “MLC” NAND flash memory, (2) a bank of multiple physical modules of “SLC” NAND flash memory (5:6-10), and (3) a “controller” that “controls access” to the memories (5:32-38) including by “maintain[ing] a translation table/address map” (5:39-40) acknowledged to be the established way to use NAND flash memory (3:1-33 [logical-to-physical, “L2P,” mapping, flash translation layer “FTL”]; 5:39-54). No part of the disclosed and claimed memory system is even suggested to be new – just two optional methods of operating the system. One is “a method for utilizing a NAND flash memory system” shown in Figs. 3A and 3B (6:11-48), where a “data integrity verification check” compares data to be written (but “retained”) to “stored” data read after writing and, upon failure to match, results in remapping to “SLC NAND flash memory.” The second optional method is not illustrated and only functionally stated in two sentences (6:49-61) as (1) “allocat[ing] ‘hot’ blocks, . . . that receive frequent writes, into the SLC NAND flash module”

and “‘cold’ blocks, . . . that receive infrequent writes, into the MLC NAND flash module” and (2) “on a periodic basis, . . . transfer[ring] the contents of those blocks [written to most frequently] into the SLC NAND flash module” as one or possibly two “wear leveling” techniques. The two (possibly three) disclosed operations, purportedly “increasing the reliability and lifetime” of the system or module (1:41-48; 3:65-67; 5:13-16; 7:12-15), disclosed and claimed as distinct MLC and SLC NAND flash memory modules³ by redirecting, upon the function conditions, data otherwise directed to MLC memory to SLC memory. The first three patents, based on the 2012 common specification, claimed the known hybrid MLC-SLC system with the two/three optional functions. Beginning in 2018, six years after the filing of the common specification, the remaining Lifetime Patent claims shed the MLC-SLC physical structure of the specification and the claims of the first three patents and claimed instead abstract “memory spaces” and “memory elements,” terms not used in the specification, but recited for the first time along with a “random access volatile memory” (“RAVM”), mentioned only ambiguously in the specification

³ Both use the same type of memory cells, which require erase before write (program) in contrast to the proposed substitute technologies, namely, RRAM, PCM and MAGRAM, which “allow data to be written over existing data (without prior erase of existing data)” (2:19-23 [emphasis added]).

explanation of the data integrity test and in the BACKGROUND as “volatile memory for buffering, caching etc” (3:61-62). While pinning the “retained” data of the data integrity test to the RAVM, the Challenged Claims do not even recite the ambiguous wear leveling option, making it even more indefinite how the recited system benefits “lifetime.” I review here details of pre-existing NAND flash technology that show that no advance in the art is recited in ‘369 patent claims.⁴ I will follow in Section VII with a review of the ‘369 patent disclosure and in Section VIII with a review of the prosecution history which untethered the Challenged Claims from the disclosed physical structure, leaving them unenabled and indefinite

⁴ This Section V bridges material gaps in the Lifetime Patents specification which incompletely proposes functionality for its recited SLC and MLC NAND flash memory and discloses only through the incorporated Rao ‘916 patent some high-level structure for an SLC NAND flash memory and controller. I review in this declaration twelve (Exs. 1034, 1038, and 1061-1070) of the forty “information disclosure statement” (“IDS”) documents filed by Dr. Rao with his April 25, 2012, ‘298 application (cited here by “IDS#_” in reference to their listing on the three IDSs filed that day [Ex. 1006 at 26-36]). I understand these documents to provide both admitted prior art and, importantly, the use of terms in the art as Dr. Rao used in the ‘300 specification and claims.

(Section IX). In Section X, I show how just one published multi-modal NAND flash memory system, Gavens (Ex. 1045) (including its incorporated references), in view of the knowledge of the POSITA, discloses or render obvious the Challenged Claims.

A. NAND Flash Memory and Operations

26. **The Base NAND Flash Memory Cell:** Unlike other nonvolatile memory (NVM) systems such as rotating magnetic disk storage, in which information could be written into a location and then written over (“direct write” or “write in place”), NAND flash evolved from Read-Only Memory (“ROM”) which were only written once (perhaps on manufacture) to Erasable Programmable Read-Only Memory (“EPROM”) which required erasing (originally by ultraviolet radiation) of constituent memory cells before they could be programmed. Electrically Erasable Programmable Read-Only Memory (EEPROM), such as NAND flash memory, use floating-gate field-effect transistors that may be electrically erased and written:

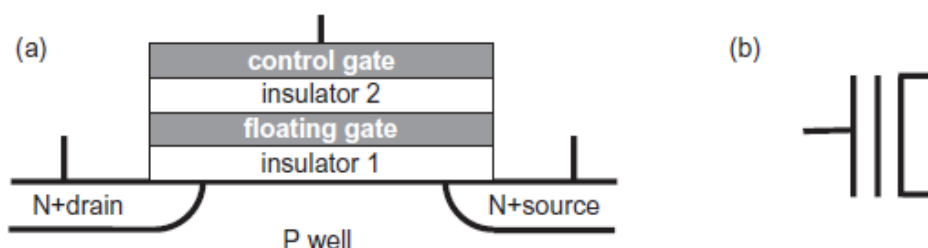


Figure 1.1. Floating-gate transistor: (a) elements of the transistor structure and (b) circuit symbol.

(NONVOLATILE MEMORY TECHNOLOGIES WITH EMPHASIS ON FLASH: A COMPREHENSIVE GUIDE TO UNDERSTANDING AND USING NVM DEVICES (Joe E. Brewer & Manzur Gill eds. Wiley-IEEE Press 2008) [“Brewer,” Ex. 1026] at 3.) At this cell level, data is represented by a charge stored on the floating gate to accomplish nonvolatile data storage; the charge determines whether the transistor will conduct when a fixed set of “read” bias conditions is applied. *Id.* The threshold voltage (related to the charge Q) is shown in the following relationship to the conduction of current:

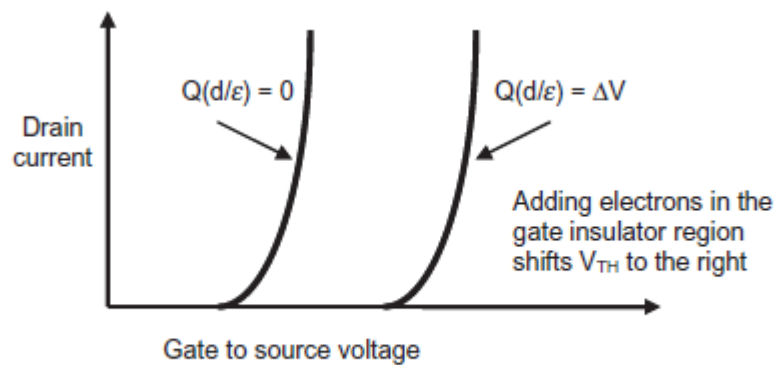


Figure 1.3. Shift of current-voltage characteristics because of inserted charge.

(Brewer at 4.) Thus, a threshold voltage can be programmed into the cell by injection of electrons (charge) along an analog continuum, which charge is held in the floating gate. From the early decades of NAND flash, the two threshold voltage levels or states shown were used to represent the two possible states – one binary “bit” – of information as “erased” or “programmed”:

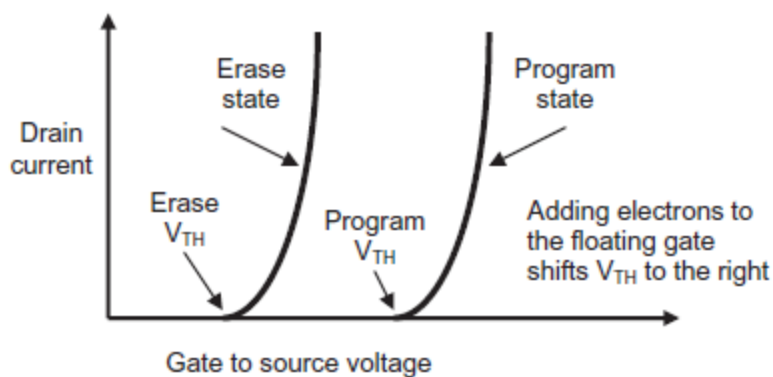


Figure 1.10. Erase state and program state current-voltage characteristics.

(Brewer at 11.) NAND flash uses Fowler-Nordheim (“FN”) tunneling to inject electrons to shift the threshold voltage curve from the “erase” (customarily representing a binary “1”) to the right to a “program” state (customarily representing a binary “0”) (*e.g.*, Brewer at 64; Rino Micheloni, et al., INSIDE NAND FLASH MEMORIES (2010) [“Micheloni,” Ex. 1025] at 24); in this write process, the charge can only be increased. One cannot by the same process move the curve to the left; it must be erased in a different, slower and more destructive process before programming again (thus “erasable programmable read-only” versus “write in place”). The tunneling operation, along with other processes (including “disrupts” by operations on nearby cells organized in groups), even sensing (reading), progressively degrades the functionality of the EEPROM transistor much faster than

the storage components of other NVM are degraded by their processes.⁵ The NAND erase process also applies FN tunneling (Brewer at 64), typically applied involving additional areas of the semiconductor substrate (Micheloni at 25-26). Although the NAND cells need to be erased before being programmed (two tunneling stresses), these operations (each a few milliseconds) typically are not performed serially on a particular cell, but erasing is typically done in the background (slow, as much as a second), and already erased cells are written to (thus not “write in place”). Before the priority date of the challenged patent, basic strategies had been settled upon and many others proposed to optimize programming throughput and balance degradation to put off failure of enough groups of cells that would compromise the intended use

⁵ (See Duane H. Oto et al., *High-Voltage Regulation and Process Considerations for High-Density 5 V-Only E²PROM's*, SC-18 IEEE J. Solid State Circuits 532, 533 Fig. 2 (Oct. 1983) [“Oto,” IDS#13, Ex. 1061] [showing the difference between threshold voltages for charged and discharged states of a FLOTOX {floating gate tunneling oxide} cell at 10-100K “clear/write cycles” shrinking to about the threshold voltage for the discharged state]; Gheorghe Samachisa et al., *A 128K Flash EEPROM Using Double-Polysilicon Technology*, SC-22 IEEE J. Solid State Circuits 676, 680 Fig. 10, Table I (Oct. 1987) [“Samachisa,” IDS#14, Ex. 1062] [voltage margin v. program/erase cycles].)

of a NAND memory system. These solutions involve both hardware structure as well as different ways to operate the hardware. A NAND memory cell by itself is not “memory” as it does not even provide for retrieval of stored information, essential to its usefulness.⁶ “While the heart of a semiconductor memory is the cell, the surrounding circuitry is the mechanism that makes it usable.” Brewer at 5.

27. **NAND memory cell arrays.** EEPROM memory cells are organized in a matrix to optimize the use of silicon area. (*E.g.*, Micheloni at 20.) In the NAND configuration, which became the prevalent configuration by the time of the 2011 priority date of the patent at issue, multiple EEPROM storage cells are arranged in strings with selection transistors at either end⁷ along a *bitline*. (See Masaki Momodomi et al., *An Experimental 4-Mbit CMOS EEPROM with a NAND-Structured Cell*, 24 IEEE J. Solid State Circuits 1238, 1238-39 Fig. 2 (Oct. 1989) [“Momodomi,” IDS#8, Ex. 1063]). In the standard array, each of the storage cells

⁶ (See MICROSOFT COMPUTER DICTIONARY 333 (5th ed. 2002) [Ex. 1027]: “**memory** *n.* A device where information can be stored and retrieved.”)

⁷ This saves circuitry on a chip – such as a transistor – associated in other arrangements for reading and writing of *each* storage cell. (See Oto at 534 Fig. 6; Samachisa at 676 [separate select transistor per cell].)

are connected across *wordlines* that define physical “rows,” sometimes called “pages,” of typically 4KB or 32K cells:

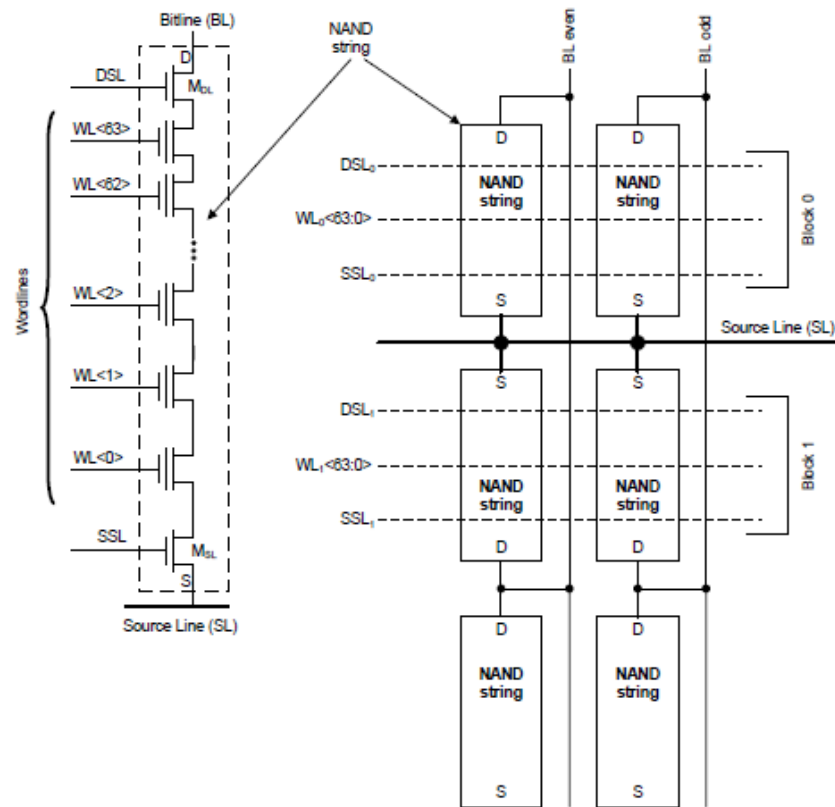


Fig. 2.2. NAND string (left) and NAND array (right)

(Michelsoni at 21.)

The wordlines are used to select the transistor of interest in the string. To access data from a string, the reading process requires that all nonselected transistors be turned on while only the selected transistor is allowed to influence [by its stored charge/threshold voltage] the current flow through the string. . . . For a single write operation the word present in an input/output register is used to determine the data inserted into the cells for that word [also called a page].

(Brewer at 6.) In long-standard NAND operation, the smallest programmable unit of NAND flash is a row of cells constituting a “page” (traditionally a binary string that represents approximately a page of type), where the parallel programming of the cells in a page is such that a cell may be left in the logical 1 state from erasure or a cell may be programmed to logical 0. In the latter case, trapped charge is added to the floating gate, and in this situation, the detection circuitry in the module will find largely an absence of current conducting from drain to source for the nominally applied control gate voltage.

28. **NAND flash memory – erasable blocks.** In advancing EEPROM technology, Toshiba in 1985 coined the term “Flash memory” for its new device erased in a “flash.” (Brewer at xvii.) “Flash memory is an EEPROM where the entire chip or a subarray within the chip may be erased at one time.” (Brewer at 2 [note omitted]; Samachisa at 676; Momodomi at 1239.) A NAND flash “block,” as shown in Micheloni Fig. 2.2 (para. 27 above), is an organization of multiple (typically 64 or 128) “pages” or rows of (typically 32K) data storage cells controlled by respective wordlines. (Spare cells may be included for such functions as error correction code, “ECC”.) A depiction of a typical NAND memory array is:

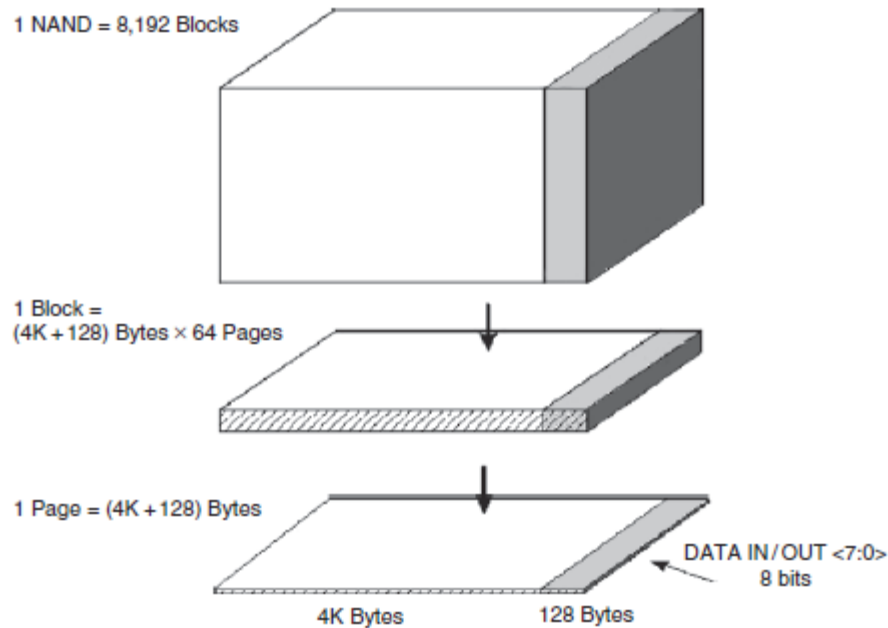


Fig. 2.7. NAND memory logic organization

The throughput (sometimes called “performance”) strategy here is to write 4KB pages (32K+ cells) in parallel (taking on the order of a few milliseconds), but erase a whole block of 2,048K+ cells (taking on the order of several seconds, and typically stressing the cells in the block more than writing) in the background in between reading and writing operations.

29. A memory cell array or a block organized within the array to store multiple pages of information as in a NAND flash memory, is not “memory.” To make a memory cell array a “memory,” one needs the circuitry to program the information into the cells, strings and blocks. (*E.g.*, Momodomi at 1241 [data latch circuitry and charge-pumping circuitry, on-chip high-voltage generator]; Brewer at 236-37.) The memory also needs the basic ability to sense and retrieve what is

stored, requiring a sense amplifier (SA) and one or more data latches per cell (one per bit stored in a cell) of a page (*e.g.*, Micheloni at 225, Brewer at 237-38), that may be shared with the programming operation in a page buffer (Brewer at 239, Fig. 6.20). There are also required interfaces for commands and addresses for which cells are to be programmed, read or erased and input/output of the data programmed or read. Thus, a **NAND flash memory** is shown as:

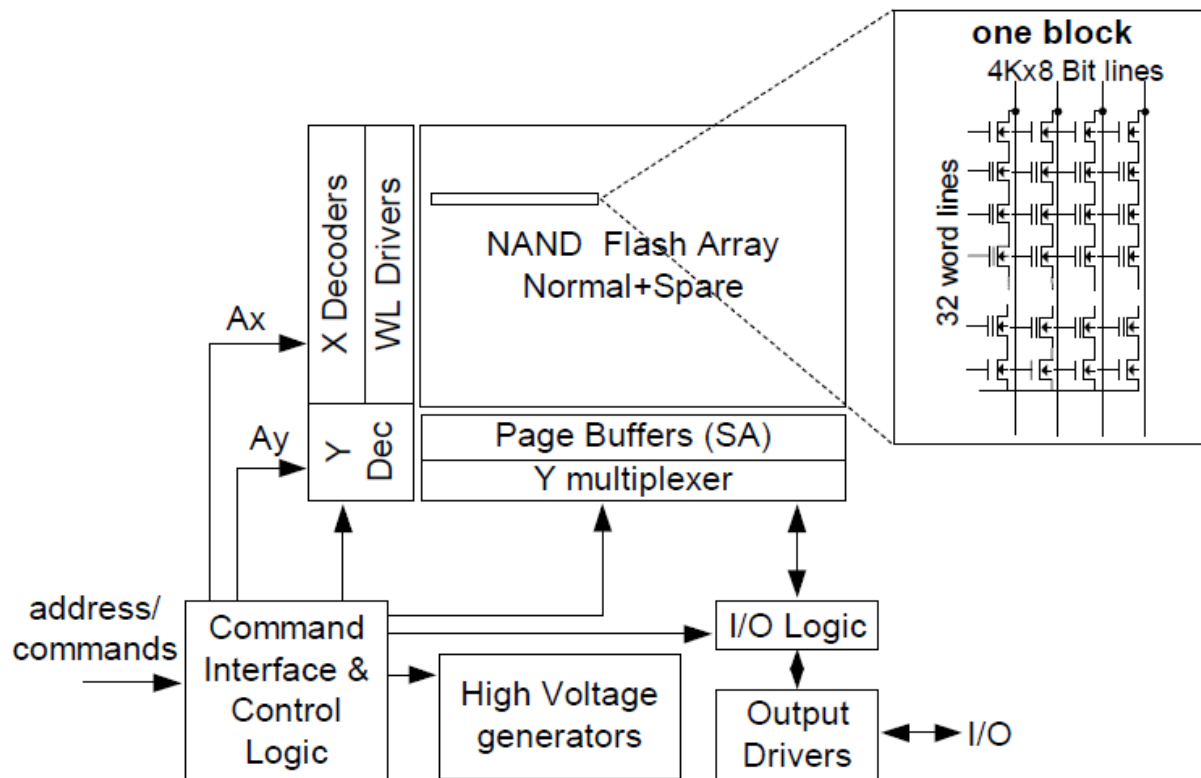


Fig. 15.1. NAND Flash memory block diagram

(Micheloni at 424.) A POSITA would understand this to be a flash memory that includes memory cells, strings and blocks in an array, none of which themselves constitute a “memory.”

30. **NAND flash memory modules.** While this arrangement may constitute a memory embedded with other circuitry, it will typically be understood as its own chip or module⁸ that is self-contained in its function and that can be interchanged with other NAND flash memory modules such as a NAND flash memory card or SSD (solid state drive) including an array or banks of flash memories (sometimes simply called “flash”). A typical SSD is shown here:

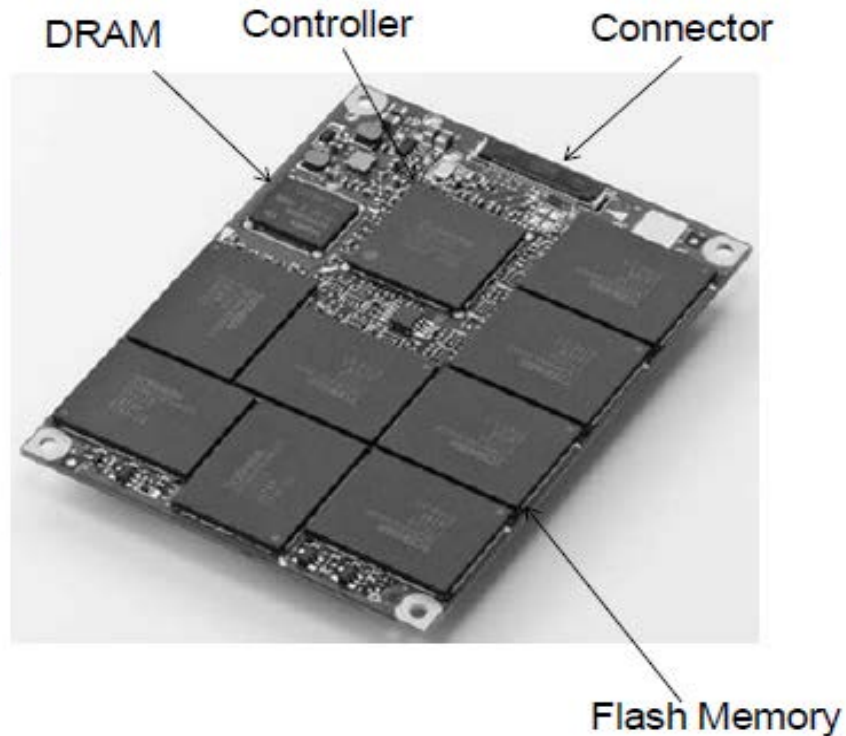
⁸ The MICROSOFT COMPUTER DICTIONARY 346 (Ex. 1027) provides the following definition for “module”:

In hardware, a self-contained component that can provide a complete function to a system and can be interchanged with other modules that provide similar functions. *See also* memory card, SIMM.

Similarly, from the Merriam-Webster online dictionary, <https://www.merriam-webster.com/dictionary/module:>

3b: a usually packaged functional assembly of electronic components for use with other such assemblies

A “flash memory module,” as used for embedded applications, “can readily be integrated into the host logic device” and includes a bus interface unit distinguished from the flash memory core, which includes a memory control as well as a flash memory array. (Brewer [Ex. 1026] at 395-96, Fig. 9.13.)



(Micheloni at 14, Fig. 1.16.)

31. The memory **device controller** typically has a microprocessor with some internal volatile memory (typically RAM for data operations) and often external volatile memory (as the DRAM shown in Micheloni Fig. 1.16 above) to support its functions of interfacing with a host computer to receive, read, write and transmit pages of data with optimal performance (transfer speed) and reliability (data integrity and retention), including appropriate data correction. Micheloni's functional block diagram, below, shows the host and SSD with a memory controller interfacing with banks of NAND flash modules:

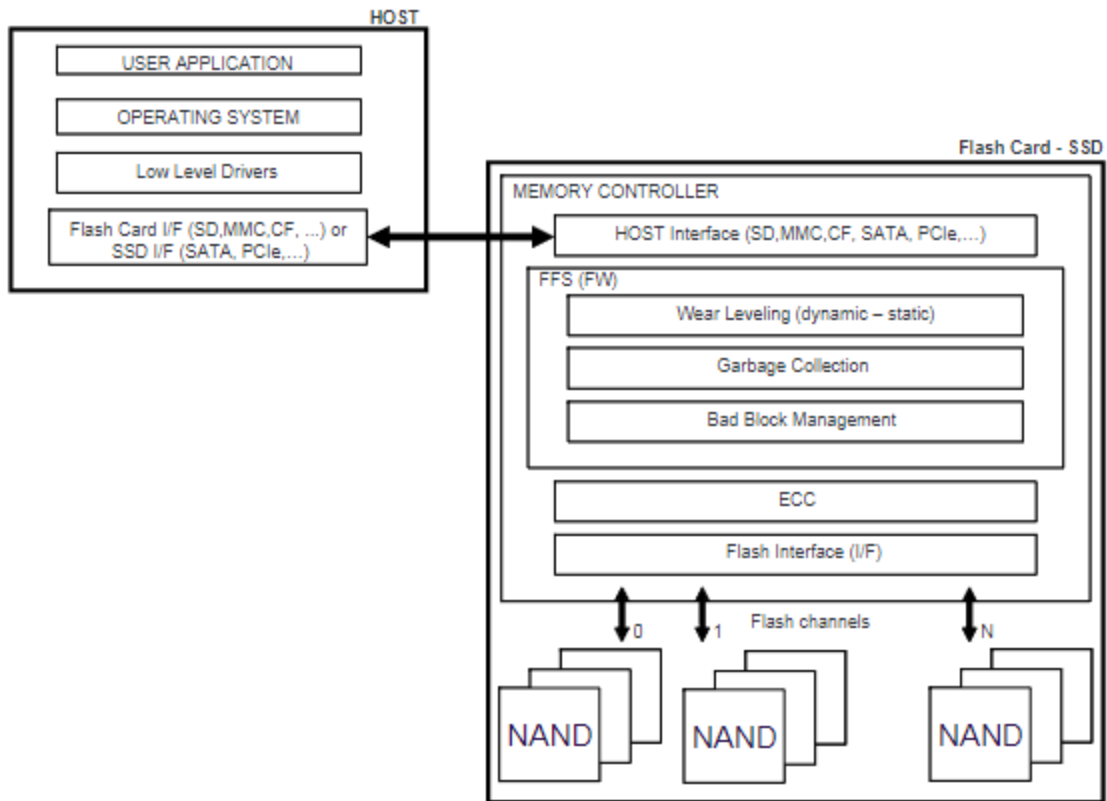


Fig. 2.33. Functional representation of a Flash card (or SSD)

(Micheloni at 39.)

32. In addition to the interfaces with the host (providing commands and “logical addresses” as described below for data to be written or read) and with the banks of NAND flash modules (with “physical addresses” as described below), the controller may include an ECC processor that detects and corrects data errors (up to a performance trade-off discussed below).

33. Crucial to the operation of flash memory is a **flash file system** (FFS) that performs the matching (or translation) to physical memory cells of host commands and addresses visible to the host such as file names associated with a

“logical address.” As described in paragraph 26 above, NAND flash programming can only increase the threshold voltage of a cell representing data and must be returned to the erased state before programming; thus, updating (or changing) a logical file (or page) needs to be made to physical cells in an erased state. Flash memory operation (shown in functional sublayers in Micheloni Fig. 2.33 at ¶ 31 above) also includes garbage collection (to consolidate valid data and to erase blocks with invalid data to free them up for new writes), wear leveling (to balance wear on blocks to avoid premature failure of overused blocks) and bad block management (to retire blocks not considered reliable for continued use). (Micheloni at 40.) “A [flash] filing system can use a standardized translation protocol such as FTL (Flash translation layer) to transform disk-style allocation table (FAT) file sectors to locations in the linear flash memory” (Brewer at 35.)

34. To perform these functions, tables, implemented as a software **Flash Translation Layer** or FTL, are widely used in order to map sectors (traditional organization for hard disk drives) and pages from logical (host-visible) locations (addresses) to physical locations (addresses) in the flash memory.⁹ (Micheloni at

⁹ There is a mismatch between the traditional HDD “sector” of 512 bytes of an HDD and the NAND flash “block” that comprises multiple (*e.g.*, 64) “pages,” each

40.) This is illustrated as follows, with the upper row being the host-logical view of the memory and the lower row being the physical flash memory:

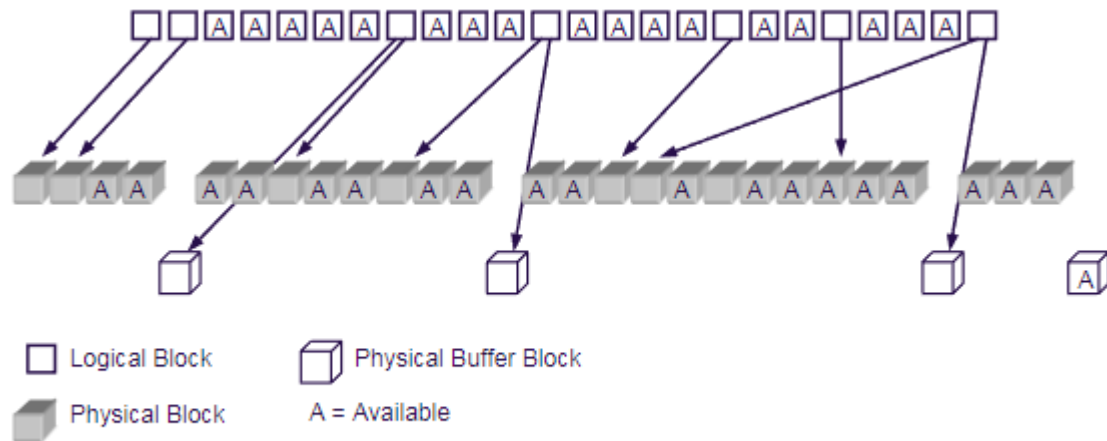


Fig. 2.34. Logical to physical block management

(Michelsoni at 41.) Flash memories lack the “update in place” capability of other NVMs, such as magnetic disk: programming of a flash memory page must start from its erased state, and the entire block must be erased to erase a page that is written, and erasing a block would substantially delay writing to that block. Thus, the updated data needs to be written to a different location from the outdated data,

typically of multiple (*e.g.*, 8) “sectors.” Various strategies were considered for mapping of bits of a data file (its name visible to the host/user, associated with a logical address, “page” or “blocks”) to the NAND flash physical location (groups of addressable cells). (*E.g.*, Tae-Sun Chung et al., *A survey of Flash Translation Layer*, 55 J. Systems Architecture 332-43 (2009) [“Chung,” IDS#39, Ex. 1064] [page, block or hybrid mapping].)

generally to another block. To make this process transparent to the host, which only “sees” a logical address target, the FTL allows emulation of independent modification of NAND pages through “logical to physical remapping.” (*Id.* at 492; also called “L2P”). When the host sends a command to write or update a certain logical page, the FTL writes the associated data (received from the host) to a different available physical page than currently mapped from the logical address and updates the table storing the logical to physical mapping. (*Id.*) The physical page containing the old data is marked as invalid. (*Id.*) Thus, in NAND flash operation, every update or write to a previously mapped physical location that has content must be remapped to another empty or erased location. If the update is to modify only part of the data in the targeted (currently mapped) page, the remaining data in that page, not to be modified (thus, considered valid), is read and combined with the update and written to the remapped location. In both cases, the previously mapped page is marked invalid. The “garbage collection” process works to coagulate the invalid pages into entire blocks of invalid pages which are then erased in background operation when the memory device is not occupied with reading or writing. These operations typically used **linked lists** to make available information for the controller’s media management (FFS or FTL). (Brewer at 30-31; Micheloni at 40.) Different strategies are applied through the FTL for the necessary NAND flash operations. (*E.g.*, Garth Goodson & Rahul Iyer, *Design Tradeoffs in a Flash*

Translation Layer, HPCA [High Performance Computer Architecture] WEST 2010 Conference, Bangalore, India [“Goodson,” IDS#23, Ex. 1070].) While the L2P translation is performed with a copy of the address table in controller RAM, to avoid loss if power is interrupted, “[t]he required translation tables are always stored on the memory card [flash memory]” (Micheloni at 40; *also* Goodson, p. 3 [LBA “periodically checkpointed to flash”]).

35. **Wear leveling.** It is considered important to keep flash memory aging (the number of program/erase cycles) of each block in a NAND flash memory system to a minimum and as uniform as possible. (Micheloni at 41.) As reviewed at paragraph 26 above, a NAND flash cell degrades (ages) to some extent after each (program/erase) cycle, comprising an initial erase and a subsequent number of writes (programs). “*Endurance* is the term used to describe the ability of a device to withstand [stresses of normal processes of writing and/or erasing cells that will eventually degrade the properties of the memory or disturb the contents of the memory] and is quantified as a minimum number of erase-write cycles or write-read cycles that the chip can be expected to survive.” (Brewer at 13.) This endurance is a

rating guarantee of the manufacturer; most often, “the device will continue operating well after the max cycling spec has been exceeded.” (*Id.* at 30.)¹⁰

36. A wear leveling software module makes sure all NAND blocks [of the same endurance expectation] are cycled an identical number of times, avoiding some blocks ending their cycle life [and compromising the useful life of the system] while others are only cycled only a few times. (Micheloni at 494.) Some wear leveling is accomplished by ordinary NAND logical-to-physical programming: “that is, each time the host application requires updates to the same (logical) sector, the memory

¹⁰ The determination of endurance is not exact. Various simulations have been performed with assumptions that each cycle comprises one simultaneous block erase and eight successive page program cycles. (Yoshida Iwata et al., *A High-Density NAND EEPROM with Block-Page Programming for Microcomputer Applications*, 25 IEEE J. Solid-State Circuits 417, 419 (Apr. 1990) [IDS#5, Ex. 1065].) In an investigation of system (“SLC” or “MLC” devices) endurance, an “end-of-life signature” was discerned in increased latency of operations. (Simona Boboila and Peter Desmoyers, *Write Endurance in Flash Drives: Measurements and Analysis* p. 4, Table 1, p. 8, Fig. 13, Usenix Conference (Feb. 2010) [“Boboila,” IDS#16, Ex. 1066].) In that investigation, “nominal” endurances provided by manufacturers were exceeded by nearly a factor of 100. (*Id.* p. 4.)

controller dynamically maps the sector onto a different (physical) sector – keeping track of the mapping either in a specific table or with pointers. The out-of-date copy of the sector is tagged as both invalid and eligible for erase. In this way, all the physical sectors are evenly used, thus keeping the aging under a reasonable value. (*Id.* at 41.)

There are two levels of wear leveling. In the *first level wear leveling* [also called dynamic] new data [directed by the host to a logical block] is programmed into a free [physical] block with the fewest write/erase cycles. Two techniques to achieve this are the chain method and the array method. (*Id.* at 494.)

In the *chain method*, the pool of available blocks is organized in a chain A simple round robin algorithm selects one block after another. When a block is erased, it is added to the chain. (*Id.*)

In the *array method* an age vector stores the number of each block’s program/erase cycles. This information can be used by the wear leveling algorithm to ensure a uniform exploitation of all the blocks. (*Id.*; see Brewer at 552 [“metering record of the usage of each block” continually sorted to provide a priority list of least-used blocks for next use].)

In the *second level wear leveling* [also called static wear leveling] from time to time blocks containing long-lived [static] data are moved (copied) into other blocks even though they never received an update command. The original block can then be used to store new incoming data. The second level wear leveling is triggered when the difference

between the maximum and the minimum number of cycles per block reaches a predetermined threshold. (Michelsoni at 495 [emphasis added].)

Both dynamic (at the time of writing) and static (in the background between write and read operations) wear leveling may be used in a system.¹¹ “Flash cards that use

¹¹ (*E.g.*, Goodson [Ex. 1070], pp. 2, 4 [both required].) It is important to understand information flow in NAND flash operation relative to high-level descriptions such as the ambiguous use of the term “block” above. Dynamic wear leveling involves writing of new data (updated data) to a location currently pointed to by the logical block address (LBA). The original data at the physical block address (PBA) pointed to by the LBA in the FTL is placed in a page buffer in the controller, and the page buffer is updated with the new data. Finally, the updated page buffer is written to a new, favored location. Static wear leveling “moves” all of the pages from an original (physical) block with a low erase count to a new (physical) block that has a high erase count. This achieves wear leveling since the data in the original block is presumed to have been updated relatively rarely (the erase count lagging other blocks), and so when placed in (by remapping to) the new block, it will typically result in the erase count of the new block stagnating at its current value. Meanwhile,

dynamic wear leveling algorithm only write across dynamic or free data areas. By far the best endurance is provided by static wear leveling, where the data is written equally to all blocks of the storage device.” (SiliconSystems, *Increasing Flash SSD Reliability*, StorageSearch.com at 4 (April 2005) [“SiliconSystems,” IDS#18, Ex. 1067]; see also Boboila [Ex. 1066] p. 9 [devices employing static wear leveling took longer to wear out than those only employing dynamic wear leveling].)

37. **Garbage collection.** Because in NAND flash a logical page may only be written to a physical page that is in an erased state, and because erasure is time-consuming, an update to a logical page (mapped to and represented) in a physical page of a physical block results in the logical page being remapped to another physical block whose corresponding page is in an erased state. This leaves the data in the previously mapped physical location (page) obsolete or “invalid.” An increasing portion of the memory is thus filled with pages comprising invalid data, which will need to be erased to make room for new data. This is resource-intensive because, before an entire block can be erased, valid data in the block needs to be copied to a new physical location. The “garbage collection” function of FTL is the

the original block is erased and placed on the free block list. (*Id.*, p. 4 [free and active lists, swaps].)

consolidation of valid data in a new physical block while the prior physical block(s) are erased to make available blocks for new data.

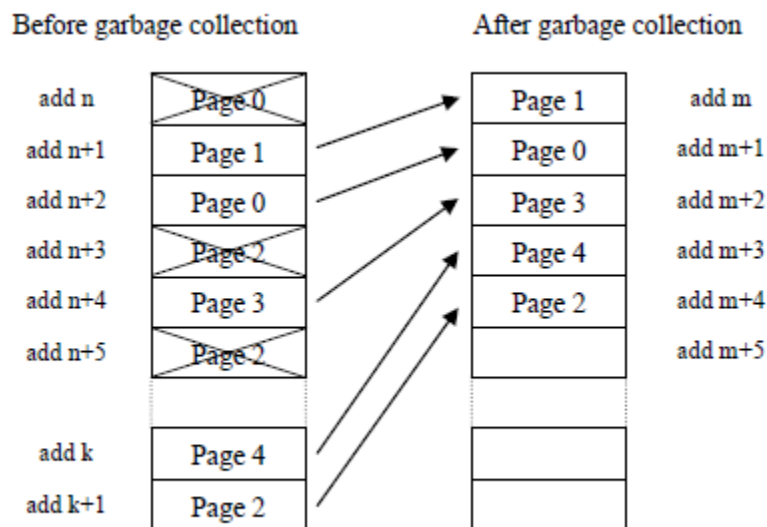


Fig. 17.7. Garbage collection

(Michelsoni at 493.) Higher garbage collection efficiency, measured by the ratio of the number of invalid pages to the total number of pages in a physical block, is desirable in order to reduce the number of rewrites of valid data while erasing and opening the block to new data. (*Id.*)

Hot data are the most frequently updated data. *Cold data* (or non-hot data) are seldom updated data. In a given block, it is advantageous to gather (cluster) all hot data or all cold data, because blocks with hot data will soon become garbage. Cleaning such blocks is beneficial because they have high garbage collection efficiency.

(*Id.* at 494.)

38. **Bad block management and ECC.** “Bad blocks” are “blocks which contain one or more locations whose reliability is not guaranteed.” (Micheloni at 42.) Thus, a “*Bad Block Management* (BBM) module creates and maintains a map of bad blocks.” (*Id.*) “[T]his map is created during factory initialization of the memory card, thus containing the list of bad blocks already present during the factory testing of the NAND Flash memory modules.” (*Id.*) The map is updated as the memory device is used when a block is determined to be bad (*id.*) and thereby retired. It is obvious that that determination may be made in various ways, depending, among other things, on what measure of unreliability is used. As described in paragraph 51 below, there are various “data integrity checks” that may be applied, with different actions taken in response to the results of those checks. One expects some less than optimally performing cells in any given block of 64 pages each of 32K cells. Rather than take out of service an entire block of millions of cells for a few unreliable cells, with ECC, the suspect cells can be kept in service and the data represented therein corrected upon retrieval (*see* ¶ 52 below). As explained in the prior art discussed below, there is a trade-off as to how many erroneous bits can be corrected through the selected levels of ECC code and

processing.¹² These functions and those reviewed in paragraphs 31 to 37 above are included in following diagram of the FTL implementation of the FFS component of the NAND flash controller shown in Micheloni Fig. 2.33 (reproduced and described at paragraphs 31 and 33 above):

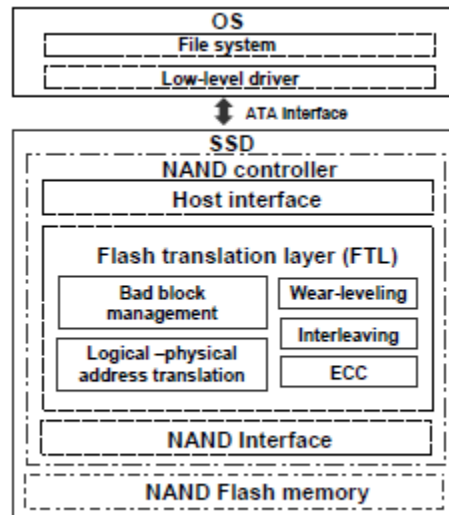


Fig. 18.3. Software architecture of SSD

(Micheloni at 516-17 {“FTL is the central part of the NAND controller [of an SSD]”}.) A POSITA understands that a NAND flash controller uses an FTL to perform these functions.

¹² Some manufacturers rate their flash product endurance including consideration of the number of bits of error correction implemented. (E.g., SiliconSystems, pp. 3-4 [including chart comparing SLC NAND and MLC NAND, the number of bits corrected by ECC and the application of no, dynamic, or static wear leveling].)

39. **System Life:** The ‘369 patent specification presents “embodiments of *a NAND flash storage system that provides long lifetime* (endurance) storage at low cost” (3:65-67 [emphasis added]) or relating “to a system and method of increasing the reliability and lifetime of a NAND flash storage system . . . through the use of a combination of multi-level cell (MLC) and single level cell (SLC) NAND flash storage” (7:12-16). Unlike for ordinary NAND flash wear leveling where “blocks that are frequently written . . . are not rendered unusable much faster than other blocks” (3:15-17; *compare* ¶¶ 35-36 above), the Lifetime Patents provide no explanation how their two-or-three embodiments increase reliability or lifetime. There is only recitation of the context of MLC and SLC flash and their known relative endurance:

MLC NAND flash SSDs are slowly replacing and/or coexisting with SLC NAND flash in newer SSD systems. . . . Generally, MLC NAND flash enjoys greater density than SLC NAND flash, at the cost of a decrease in access speed and lifetime (endurance). . . MLC NAND flash has a much lower lifetime (endurance) [3-10K writes] than SLC NAND flash [50-100K writes].

(3:34-48 [emphasis added].) It is a straightforward, obvious proposition to favor the use of a memory with higher (expected) endurance over a memory with lower endurance in order to “increase” the (average) endurance of a system that includes both, relative to greater use of the lower endurance memory. Endurance tests for the

cell level are not exact (note 6 above), and the tests for devices have resulted in as much as an order of magnitude increase in actual use relative to the manufacturer-stated endurance (note 10 above). Any write operation (particularly if preceded by an erase) in a NAND flash system will marginally decrease the lifetime of the written part of the system, *i.e.*, a page of cells. However, the wear leveling outlined in paragraphs 35-36 above “increases” useful lifetime of the flash memory system by minimizing unnecessarily early failure caused by greater than average wear on parts of the system.¹³ The clustering of “hot” or “cold” data for rewriting in garbage collection as described in paragraph 37 above is another strategy to avoid unnecessary writes and erases. The use of ECC described in paragraph 38 above clearly avoids unnecessary writes and erases. These NAND flash system life-enhancing strategies were well-known before the July 19, 2011, priority date of the

¹³ “A device with no wear leveling wears out faster because data is written to the same physical block.” (SiliconSystems [Ex. 1067], p. 4.) The Boboila investigation concluded that “wear-leveling techniques lead to a significant increase in the endurance of the whole device, compared to the endurance of the memory chip itself, with static wear-leveling providing much higher endurance than dynamic wear leveling.” (Boboila, [Ex. 1066] p. 9 [both SLC and MLC devices investigated].)

‘369 patent. Those strategies had already been applied to hybrid systems including both SLC and MLC memories.

B. MLC and SLC

40. “SLC” as an abbreviation for “single level cell” is somewhat of a misnomer, as it actually describes the use in “original” (previous subsection) NAND flash memory of not one, but two, floating gate charge or threshold voltage levels in the cell (¶ 26, Fig. 1.10 above; see Brewer [Ex. 1026 (2008)] at 87, 89 [“traditional Flash memories with two threshold levels per cell”]). In my investigation (reviewed at ¶ 42 below), the earliest use I found of “SLC” was in 2001, and it referred to the older “single level *program* cell NAND flash memory” as distinguished from the then new “MLC” or “multi-level *program* cell NAND flash memory; the latter, as described in paragraph 43 below, allowed *programming* of multiple, “lower” and “upper”, logical pages of data into a single row (a single physical page) of NAND flash cells. In fact, the storage “cells” of both NAND flash memories (SLC and MLC) are the same: the floating gate transistors shown in paragraph 26 above, which may be programmed to different charge ranges. As explained by the team that developed what they called the “Multi-Level-Cell (M.L.C.) concept”:

The charge storage ability of the flash memory cell is a key to the storage of multiple bits in a single cell. The flash cell is an analog storage device not a digital storage device. It stores charge (quantized at a single electron) not bits. By using a controlled programming

technique, it is possible to place a precise amount of charge on the floating gate. If charge can be accurately placed to one of four charge states (or ranges), then the cell can be said to store two bits. Each of the four charge states is associated with a two-bit data pattern. Figure 7 illustrates the threshold voltage distributions for a 1/2Mc block for two bit per cell storage.”

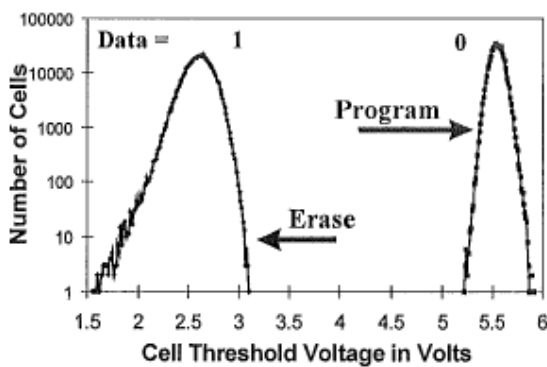


Figure 6: Single bit/cell array threshold voltage histogram

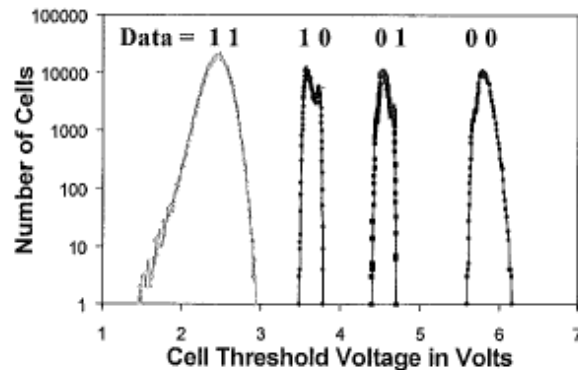


Figure 7: Two bit/cell array threshold voltage histogram

(Greg Atwood et al., Intel StrataFlash™ *Memory Technology Overview* at 4, Intel Technology J. (Q4 1997) [“Atwood,” IDS#29, Ex. 1038] [emphasis added].) “M-level cells” allow storage of the representation of multiple (m) bits of information in a cell by allowing charging of the floating gate to 2^m possible threshold voltage levels, each level representing a logical m -bit (-place) binary number. (The representation assignment often is made by Gray code, where adjacent threshold voltage levels represent binary numbers differing by only one bit in order to minimize the number of erroneous bits if a level is read as its neighboring level. Thus, rather than the descending binary order shown in Figure 7 above, the bit-pair

assignments to the threshold levels would instead be, for example, in left to right order: 11 10 00 01.)

41. The full, continuous (analog) range (window) of threshold voltages/charges possible to be programmed into (and stored in) a floating gate cell need not be divided into binary powers of two levels to represent a binary number. Following is diagram of non-binary seven memory states (seven voltage threshold levels) from U.S. Patent Appl. Pub. No. 2010/0172179 by Gorobets et al. (“Gorobets ‘179,” Ex. 1044) [Fig. 3, ¶ [0061]:

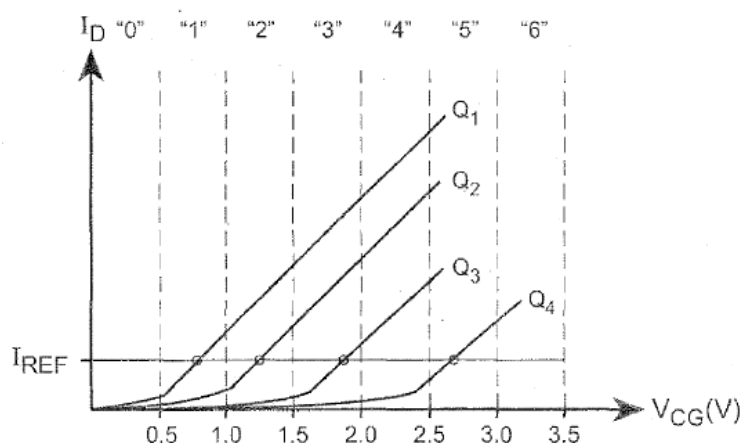


FIG. 3

The diagram also shows another important aspect of memory: how to sense the information representation using a reference current I_{REF} to sense the charge Q associated with the logical state. Memory requires not only the ability to store, but the ability to retrieve information. (See ¶ 26 above, note 6.) While multi-level storage implies a capability to represent multiple bits, that is not enough to make a

“memory.” In addition to storage and retrieval, NAND flash memory operation requires the ability to erase previous read-only states to program new states in a different block. (See ¶¶ 27-29 above.)

42. The first reference to “SLC” I have found is in the article, Taehee Cho et al., “A 3.3 V 1 Gb multi-level NAND flash memory with non-uniform threshold voltage distribution,” *2001 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, p. 1, ISSCC (Cat. No.01CH37177) (2001) (“Cho,” Ex. 1039), where a Samsung Electronics team noted the difficulty of “multi-level program cell NAND flash memory (MLC)” to compete in the market with “single-level program cell NAND flash memory” (emphasis added). One of the disadvantages of MLC they noted was “[t]he 2b [MLC] NAND flash memory having a three-step programming sequence requires 3 times longer program time than that of a 1b [SLC] NAND flash memory.”¹⁴ (*Id.*) From this early reference and my

¹⁴ To improve performance, the team experimented with a sense-and-latch page buffer that allowed a “single-bit-per-cell option transistor.” (*Id.* pp. 1-2, Fig. 2.1.2.) A paper by the same team later that year – submitted by Dr. Rao with his application resulting in the ‘298 patent – also compared “multilevel program cell (MLC) technique” with “single-level program cell (SLC) technique”; they noted again the

knowledge of the history of NAND flash, I believe the term “SLC” was introduced to distinguish prior NAND flash memory from the “MLC” concept of programming. Although I have heard “MLC” applied to NOR flash memory, I don’t recall the term “SLC” applied to anything other than NAND flash memory.

43. **Program level.** The 2001 reference of the Cho team to “multi-level program cell NAND flash memory (MLC)” and “single-level program cell NAND flash memory (SLC)” may have reflected the by-then accepted use of MLC NAND cells to represent multiple bits of binary data, where the bits are sourced from distinct strings (logical pages), rather than the same string. The approach abandoned an early approach taking two bits of one logical page and “encoding” (or compressing) them to be represented by one of the four threshold voltage levels of an MLC NAND flash memory cell. Instead, one bit of one logical page is programmed into the cell as one of two threshold voltages much as in traditional (now called “SLC”) memories, but

market disadvantage of “MLC NAND flash memories” relative to “SLC NAND flash memories” but reported advances in a “dual-mode NAND flash memory having 1-Gb MLC and 512-Mb SLC modes.” (Taehee Cho et al., *A Dual-Mode NAND Flash Memory: 1-Gb Multilevel and High-Performance 512-Mb Single Level Modes*, 36 IEEE J. Solid State Circuits 1700, 1700 (Nov. 2001) [“Cho Dual-Mode,” IDS#11, Ex. 1068] [emphasis added].)

a second bit of a different logical page is later programmed into the cell by moving the threshold voltage previously programmed to a higher, final voltage threshold representing a two-bit number where one bit comes from one page and the other bit from the other page. The approach was proposed in Ken Takeuchi, *A Multipage Cell Architecture for High-Speed Programming Multilevel NAND Flash Memories*, 33 IEEE J. Solid-State Circuits 1228, 1230 Figs. 5-7 (Aug. 1998) (“Takeuchi,” IDS#8, Ex. 1069) (“conventional” four-level cell¹⁵ programs in the same operation two bits of data belonging to the same page, where “page” is “the group of memory cell that are programmed simultaneously):

¹⁵ Note that what was called “single-level program cell” by Cho in 2001 was at this time (1998) described as “two-level NAND cell.” (*E.g.*, Takeuchi at 1229, Fig. 2, 1231, Table 1.)

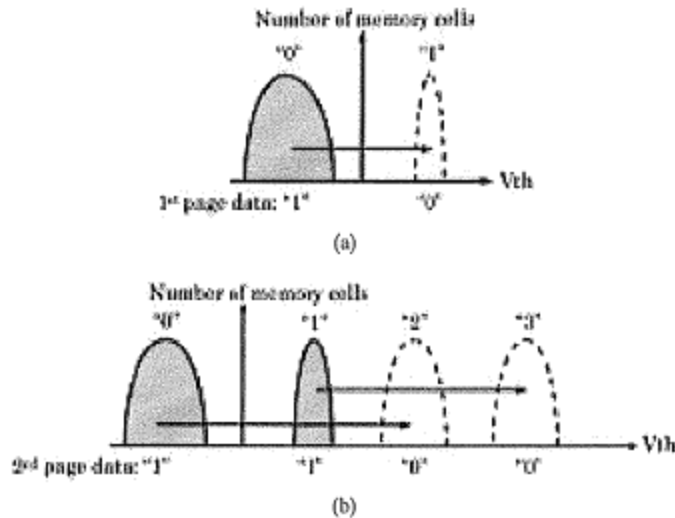
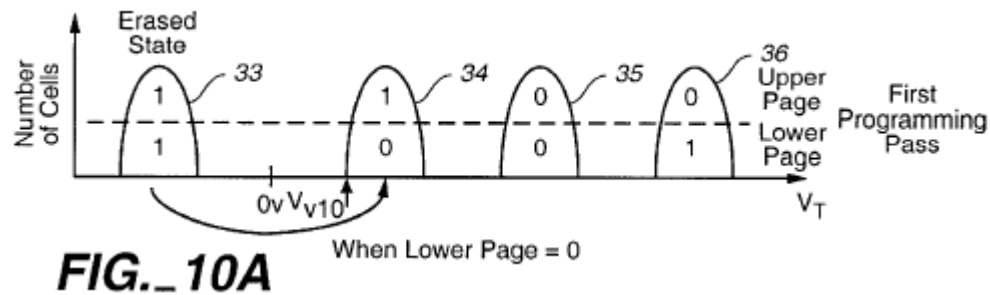


Fig. 6. Program operation of the proposed architecture. (a) First page program. (b) Second page program.

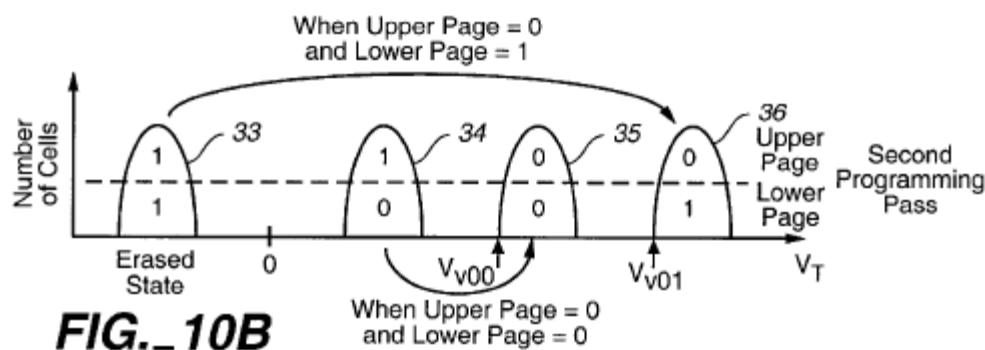
The two-pass approach has the advantage of automatically encoding the bit value of the first page with relatively little modification to traditional, 2-(threshold) level NAND flash programming (such as making room for the final, 2-bit threshold level) on the first pass and the bit value of the second page on the second pass. The threshold level sensed on reading needs to be decoded. For both one- and two-pass approaches, two latches are required for such decoding (*id.* at 1230, 1232, Fig. 12) as well as for the reading necessary prior to second pass programming (*id.* at 1233, Fig. 13). The Takeuchi approach was refined by the time of the 2001 Cho articles, reflected in the Sept. 17, 2001, application resulting in U.S. Patent No. 6,456,528 to Chen for “Selective Operation of a Multi-State Non-Volatile Memory System in a Binary Mode” (“Chen ‘528,” Ex. 1040): “Figs. 10A and 10B illustrate a specific

existing technique . . . In a first programming pass, the cell's threshold is set¹⁶ according to the bit from the lower logical page [the least significant bit of the two-bit representation].” (8:6-10 [emphasis added]).



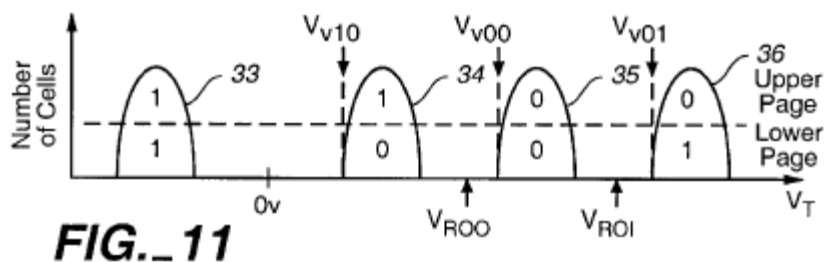
“In a second programming pass, the cell's threshold level is set according to the bit being stored in the cell from the upper logical page [the most significant bit of the two-bit representation].” (Chen ‘528 8:15-18 [emphasis added].)

¹⁶ Pulse programming is used, where between the stepped up voltage pulses “verify (read) operations are carried out,” where “the programmed level of each cell being programmed in parallel is read” to determine if it has reached or exceeded the verify level, and if so, it is taken off the pulse voltage line, while other cells are continued until they reach their verify levels. (7:52-8:5.)



The two passes result in the possible threshold levels shown in Fig. 11:¹⁷

¹⁷ Also shown in Fig. 11 are “the voltages used to read each cell in order to determine which of the four threshold states the cell lies,” including verification of the states “between repetitive programming pulses” (8:51-57) as well as reading from the memory array (8:60-65). Two registers are required for read and verification operations (6:38-7:8, Fig. 7) to decode the threshold values into 2-bit logical values. Note the pairs of latches in Takeuchi at 1232, Fig. 12. (*Also Micheloni at 275-77* [“Since two pieces of information must be stored, it is evident that at least two latches are needed inside the sensing circuit.”], 456 [“the number of latches inside a sensing circuit is (as a minimum) equal to the number n of bits stored inside the cell”].) The decoding of threshold levels representing bits from different logical pages requires two read passes. (*E.g.*, U.S. Patent No. 8,634,240 to Gavens et al., published April 28, 2011 [“Gavens,” Ex. 1045, Figs. 24A and 24B and associated



The lower page/bit and upper page/bit programming and threshold representation provide convenient encoding with Gray code mitigation of read errors. This characterization is consistent with the Cho reference to “multi-level program cell NAND flash memory (MLC)” and “single-level program cell NAND flash memory (SLC),” where the first allows for multiple-pass programming of multiple “levels” of different logical pages into a single physical row (page) of NAND flash cells, supported by hardware including at least a register or set of latches per level and the latter only having the hardware capable of programming only one “level” of a logical page.

44. **Endurance issues.** As explained at paragraph 26 above, through repeated stress of FN tunneling particularly in erasing but also in writing, the floating gate transistor memory cells used in NAND flash degrades at least by a narrowing

text at 26:31-27:2].) Reading one page from MLC requires different circuitry and has a different process for reading one page compared to the “original” NAND flash, “SLC.”

of separation (window) between the threshold voltages available to represent the fully erased and fully programmed states in 2-state operation. With 2^n states needed to represent n bits of binary data from n different logical pages (in NAND flash operation since at least 2001), the separation between threshold voltages shrinks exponentially, so that various “disturbs” or loss of charge may not be tolerated and program verification and retries increase time to program until failure. (See ¶ 26, note 6, ¶ 35, note 10.) In one empirical study published in 2010, Eitan Yaakobi et al., *Error Characterization and Coding Schemes for Flash Memories*, p. 1 (2010 IEEE Globecom Workshops, Miami, FL) (“Yaakobi,” Ex. 1032), it was remarked that “the observed lifetimes of the tested flash blocks were much longer than the lifetimes specified by the manufacturer . . . [a]n SLC block can typically tolerate 10^5 to 10^6 erasures whereas an MLC block 10^4 . . .” Raw bit error rates were presented:

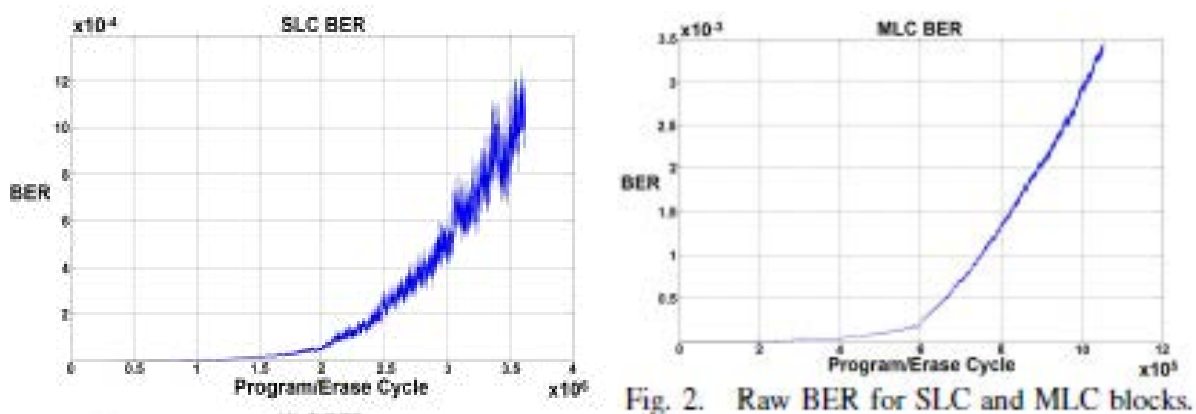


Fig. 2. Raw BER for SLC and MLC blocks.

These are factors towards the greater error probability and wear for MLCs compared to SLCs, leading to the known trade-off between shorter erase-write cycle life of MLCs but their lower cost per bit of memory.

45. By 2010, the comparative characteristics of SLCs and MLCs (often understood to be limited to 2 bits/cell, compared to TLC [3 b/c] and QLC [4 b/c] were known and summarized in the following graph (Micheloni at 9):

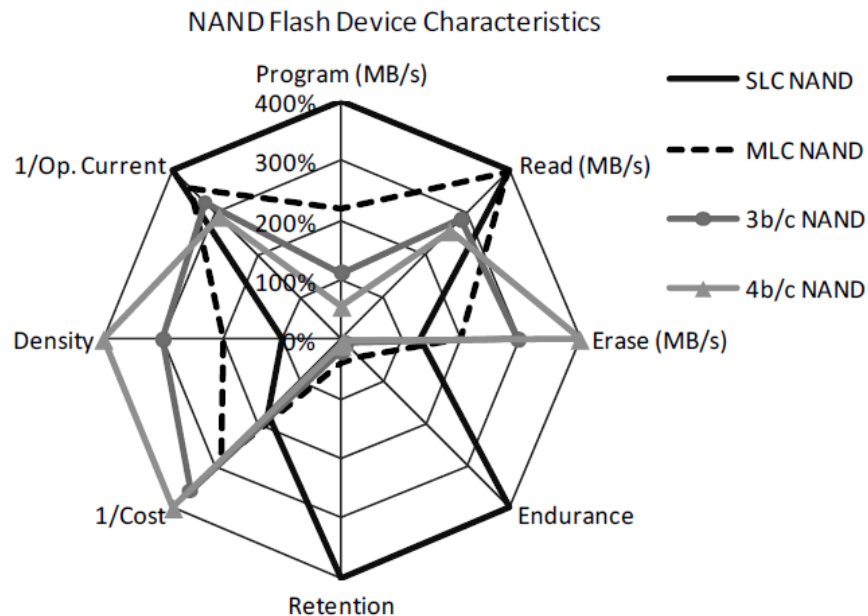


Fig. 1.10. NAND device characteristics (Source: Forward Insights)

Notably, the rated “endurance” for MLCs is an order of magnitude less than for SLCs.

C. Hybrid Use of MLC and SLC Memory

46. This reliability trade-off favored the use of MLCs for infrequent write applications such as memory cards for consumer digital cameras, an application which also tolerates bit errors. Enterprise uses demanding endurance and quality were often met with SLC NAND. (Micheloni at 169.) However, it was recognized that, for enterprise use, frequently programmed or critical data (such as chip or

memory system information) might be advantageously held in an SLC memory module, with less frequently programmed data (such as user data) held in an MLC system. (See Chris Evans, *Enterprise MLC: How flash vendors are boosting MLC write endurance*, ComputerWeekly.com (June 3, 2011) (Ex. 1033).) The hybrid SLC and MLC modules approach had been proposed by 2008. (See generally Nelson Duann, SLC & MLC Hybrid (Silicon Motion Flash Memory Summit, Santa Clara, CA, Aug. 12, 2008) [**“Duann,”** IDS#15, Ex. 1034].) U.S. Patent No. 8,078,794 to Lee et al., published Sept. 4, 2008 (**“Lee ‘794,”** Ex. 1041) disclosed the use of distinct “SLC flash memory array 226 [that] may include at least one flash memory chip” and “MLC flash memory array 228 [that] may include at least one MLC flash memory chip” (Lee ‘794 5:10-13, Fig. 2 [the “hybrid solid state device (SSD) 200 also includes a controller 210, a data read cache 203 and a data write cache 204, and a logical-to-physical address mapping memory 220”], 4:11-20). Lee ‘794 Fig. 3 shows mapping of logical block address (LBA) of system files and user directories (both frequently written) to SLC physical blocks 326 and user files to MLC physical blocks 328, where the physical blocks are physically part of their respective memory arrays (chips):

FIG. 3

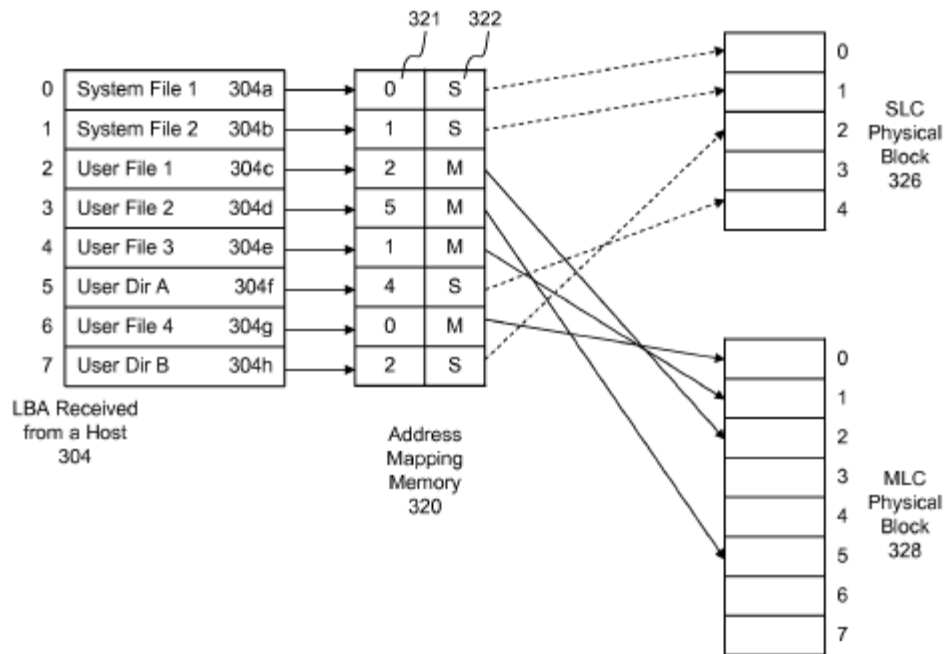


Fig. 5A shows decisions to remap system files and user directories – “hot data,” expected to be updated frequently – to the SLC blocks. Fig. 5B shows an additional decision

whether the user file has been accessed more than a threshold. The threshold may be defined as following examples: 1) the user file has been accesses [sic] more than a predefined number of times; 2) data blocks of the user file are found in the data write cache (i.e., have recently been accessed); 3) previously modified time of the user file indicates very recent access; or 4) an access counter of the user file shows a high number of accesses (i.e., more than a predefined threshold).

(Lee ‘794 6:58-66.) Thus, “hot data” determined by logical block count under thresholds 1) or 4) are dynamically (at the time of ordinary writing) sent to the SLC.

This is similar to dynamic wear leveling by the array method favoring a physical block by its age vector (§ 36 above, which is not as effective as static wear leveling, note 13 above) where a block in SLC likely will have much lower normalized (versus endurance) age than a block in MLC.

47. The hybrid use of SLC and MLC memory chips/modules to capture their respective advantages was also applied to other known wear-leveling techniques prior to June 19, 2011. *E.g.*, **Sutardja**, U.S. Patent Appl. Pub. No. 2008/0140918 “Hybrid Non-Volatile Solid State Memory System” (“Sutardja,” Ex. 1042). The Sutardja architecture included a first solid state nonvolatile memory 204 with relatively large capacity (such as MLC with a device usable lifetime of 10K write cycles) and a second solid state nonvolatile memory 206 with lower density and greater lifetime (such as SLC with a device usable lifetime of 100K write cycles). (*Id.* §§ [0006], [0106]-[0108].) Multiple, detailed criteria are provided for mapping data to one or the other memory, ranging from write cycle lifetimes, to ratios of use, to frequencies of writes, normalizing the wear on a physical block, using a write leveling software module and a write monitoring software module. (*Id.* §§ [0011]-[0083].) Sutardja disclosed detailed algorithms for application of these criteria. (*Id.* Figs. 7A-7E, 8 and explanations.) The base algorithm is similar to that of Lee ‘794, mapping logical addresses with low write frequency (as reported by the host) to the lower endurance memory and mapping logical addresses with high write

frequency to the higher endurance memory. (Sutardja Fig. 7A (506) & (508).) There are optional processes (not necessarily intervening, but at an appropriate “time”) that are based on actual wear to the physical blocks by (a) considering the number of writes during a predetermined time to the physical lower endurance memory and identifying the least used block in the higher endurance memory (Fig. 7C “data shift analysis”), (b) testing degradation of a physical addresses by successive writes and read backs (Fig. 7D “degradation analysis”), and (c) generating wear levels for both memories and mapping logical addresses to the other memory if the wear level of one exceeds a threshold (Fig. 7E). The data shift (*id.* ¶¶ [0126]-[0127]) resembles the traditional static wear leveling of moving static data to more worn (lower residual endurance) blocks (¶ 37 above), but also optionally includes biasing mapping a logical address associated with a block in the lower endurance memory which exceeds a number of writes in a predetermined time to a block in the higher endurance memory (Sutardja ¶ [0128]).

48. Also, **Moshayedi** U.S. Patent Appl. No. 2009/00327591 (“Moshayedi,” Ex. 1043), specifically showed four channels (banks) Fig. 1 (108) each connecting “five chips of flash (e.g., the first chip can be a K9K8G08 SLC flash, and the rest can be K9G8G08 MLC flash)” (¶ [0038]) in a system, where, similarly to the hybrid systems of Lee ‘794 and Sutardja sending “hot data” to SLC at the time of updating, the drive

keeps track of the number of times that data for each logical block address (LBA) has been written to the flash memory, and determines whether to store newly received data associated with a particular LBA in SLC flash or in MLC flash, depending on the number of writes that have occurred for that particular LBA.

(Moshayedi ¶ [0024].) As in Sutardja, there are other optional processes that are based on actual wear (erase count) or content of the physical blocks. (A variety of counts are disclosed using linked lists {*id.* ¶¶ [0029]-[0033], [0051]-[0058]}.) In a variation of the garbage collection that I reviewed at paragraph 37 above,

. . . a set of linked lists can be employed to track the number of valid segments in each block of flash memory . . . When empty blocks are needed to write new data from the host, [physical] blocks with the least number of valid segments are selected for garbage collection, where any remaining valid segments are copied to a new block and the old block is erased.

(*Id.* ¶ [0029].) In a variation of the static wear leveling that I reviewed at paragraph 37 above, data presumed to be relatively static or “cold” by their presence in blocks of relatively low erase counts:

A set of linked lists can be used for device 100 to track the relative number of times blocks have been erased based on the lowest erase count among the blocks. For example, one list may contain all blocks with erase count within 0 to 1000 of the erase count of the lowest count block, the next list may contain all blocks with erase counts that are

within 1001 to 2000 of the erase count of the lowest count block, etc. When a block reaches the highest erase count list, the data contained within that block is swapped with data from a block on the lowest erase count list, which is presumed to be more static since the block containing the data has a low erase count. This wear leveling is done to keep all of the flash memory chips within a specified range of erase counts.

(*Id.* ¶ [0030].) The physical block erase count list may be used to move data between MLC flash and SLC flash in another variant of traditional wear leveling:

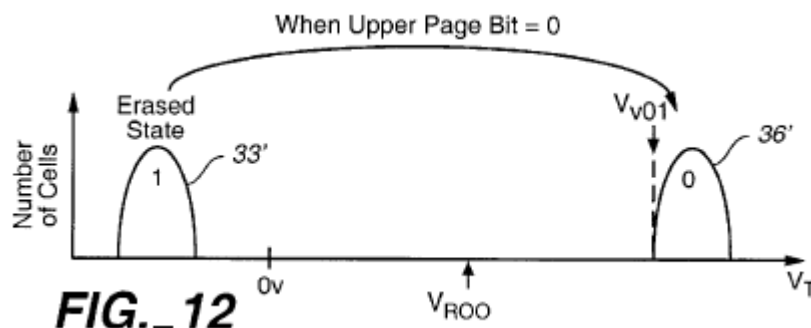
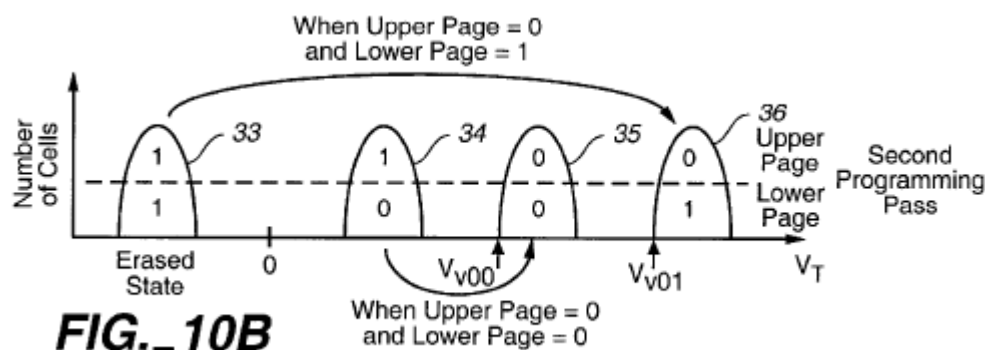
once a block in MLC flash reaches a threshold erase count (e.g., 500), the next write operation to that block triggers a swap where the data from the MLC flash block is written to a block in SLC flash. In this manner, data for an LBA that is frequently written and causes frequent erasures is moved to SLC flash which can perform more erase cycles than MLC flash.

(*Id.* ¶ [0032].)

D. Use of MLC Memory in “SLC Mode” (“Pseudo-SLC”)

49. Meanwhile, memory system manufacturers such as SanDisk provided memory systems that included MLC blocks that could be configured to function in one-bit (two-state, two threshold voltage level) mode, later sometimes called “pseudo-SLC” or “pSLC,” and this was disclosed as early as September 24, 2002, in the issuance of Chen ‘528,” (Ex. 1040) for “Selective Operation of a Multi-State Non-Volatile Memory System in a Binary **Mode**” (emphasis added), where the

“existing technique” of two page pass programming of MLC flash memory (§ 43 above) where only “the first two states 33 and 36” (in Chen ‘528 Fig. 10B) below are programmed as shown in Fig. 12:



In essence, the lower page is assumed to be a null “1” and not programmed, and only the upper page is programmed. Nonetheless, “[t]he reference voltage V_{v01} is used to verify the programming in the same manner as the multi-state case.” (Chen ‘528 9:22-24.) Recognizing the retention advantage of the two-state mode because of the increased margin and fewer programming passes (writes, wear) (*id.* 9:52-10:30), Chen suggested one use in which “memory cell blocks to which data are written most frequently are operated in two states while the remaining blocks that are

rewritten less frequently are operated in multi-state” (*id.* 10:41-52). Gorobets ‘179 (Ex. 1044) disclosed adaptive use of spare blocks in binary-operated and MLC pools and observed of binary operation (lower page/bit): “While this approach does not fully optimize the range of the threshold window as in the case of a single-level cell (‘SLC’) memory, it has the advantage of using the same demarcation or sensing level as in the operations of the lower bit of the MLC memory” (*id.* ¶ [0083]).¹⁸ Such configuration was disclosed in Gavens (Ex. 1045, 8:19-21 “[a] memory chip 100 includes a memory array 200 of memory cells with each cell capable of being configured as a multi-level cell (‘MLC’)]”) and the multiple patents it incorporated by reference. The benefits of split mode operation include similar trade-offs as between “actual” SLC and MLC modules or chips as in Lee ‘794, although the program cycle and processing overhead may be the same between an MLC operating in MLC mode and one operating in SLC mode. (Reading one page stored in pSLC still requires two sets of latches and two passes. [Note 17 above.]) An early appraisal of relative (Samsung) performance is provided, where “pseudo-SLC” is “MLC_{LSB}”:

¹⁸ Micheloni at 511 also notes that the greater distance between threshold voltage levels in an “SLC partition . . . allows speeding up program and read operations getting closer to native SLC memories.”

Table 1: Performance comparison of different types of cell programming (us)

Operation	SLC	MLC_{LSB}	MLC_{BOTH}
Read (page)	399	409	403
Write (page)	417	431	994
Erase (block)	860	872	872

(Sungjin Lee et al., *FlexFS: A Flexible Flash File System for MLC NAND Flash Memory* (2009 USENIX Annual Technical Conference, San Diego, CA, June 14-19, 2009) [Ex. 1036].) Yaakobi, p. 5 (Ex. 1032) remarked that the 2-state MLC technique can be made “adaptive by initially using the MLC device in its native mode, and then switching to SLC mode after a certain number of iterations, when the BER [bit error rate] has become larger due to device wear.” Another paper at the same conference, Ashish Jagmohan et al., *Adaptive Endurance Coding for NAND Flash* at 1841-45 (2010 IEEE Globecom Workshops, Miami, FL) (Ex. 1035), shows that 2-state operation of MLC has a BER after 20K p-e cycles that is an order of magnitude better than native-mode operation of MLC, but definitely not as good as SLC. This is confirmed in current commercial offerings that compare SLC as rated for 50-100K p/e cycles, pSLC for 20-30K and TLC for 3K. *E.g.*, Smart Modules Technologies, *Pseudo-SLC For Flash Storage*, <https://www.smartm.com/technology/pseudo-slc-pslc> (visited Nov. 12, 2024); Cactus Technologies, *Industrial pSLC Flash Storage*, <https://cactus-tech.com/products/industrial-pslc> (visited Nov. 12, 2024).

50. Prior to July 19, 2011, and since, system design considerations included different deployment of SLC or pSLC versus MLC memory, such as is exemplified in Fig 3 of Segal et al. U.S. Patent Appl. Publ. No. 2011/0271043 (“Segal,” Ex. 1048), for bad block replacement by adjusting MLC and SLC spare pools, including optimizing the diversion of MLC spares for SLC-mode spare usage, and similarly in Gorobets ‘179 (Ex. 1044) for MLC and pSLC spares.

E. “Data Integrity Checking” in NAND Flash Memory Systems

51. In addition to the routine threshold voltage programming checks mentioned above (*e.g.*, ¶ 43, notes 16 & 17 above) for readable writing (including for determining when to stop pulsing and for “read errors”), NAND flash systems – particularly near the end of useful life – often operated with an “integrity check” to determine whether the data written into a NAND flash page of cells (the smallest group for writing) is not only readably written, but is read (decoded) into a binary string identical to the one directed to be written. Different actions may or may not follow if bit errors (mismatches) are detected, including simply keeping the written copy to be corrected by ECC on future reads, retrying,¹⁹ or remapping and writing

¹⁹ Ordinarily it would be unduly disruptive to erase and retry the same physical location for the same directed write. However, various actions are performed during

to a different location. Excessive detected errors may lead to flagging a block as a “bad block” with respect to bad block management (BBM).

52. In most commercialized NAND flash memory systems, checks of data read against the data directed to be written are performed indirectly (if at all until end-of-rated-life) as part of the processing of customized ECC schemes. (ECC schemes have long been used for automated correction of digital communications, partly because such communications may suffer intervening interruptions such as radio frequency interference. Comparable interference occurs in NAND flash memory from writing into adjacent cells, with less error tolerance with higher-

a period between access (read/write) activities, *i.e.*, in the background. These include garbage collection, static wear leveling, “refreshes” of programmed threshold voltage levels that naturally drift over time (*e.g.*, U.S. Patent App. Pub. No. 2009/0172267, “REFRESH METHOD OF A FLASH MEMORY,” by Oribe et al. [“Oribe,” Ex. 1056]). For MLC memories, there is the possibility of reconfiguring and using a “failed” page’s block in a lower density, more tolerant mode in later writes (*e.g.*, Gorobets ‘179 [Ex. 1044] [partition of MLC memory into blocks with different *operating modes* of higher and lower endurance; higher endurance mode “spares” configured as necessary], Segal [Ex. 1048] [same, ¶ [0007]: “In MLC flash memory devices, memory blocks may be used as different type” [emphasis added].)

density [more closely spaced threshold voltage levels] use and greater wear.) A limited number of bad cells (those that cannot be read properly) in a page can be tolerated if ECC is used, thereby avoiding the stresses of erasing the block and writing to another, bringing both closer to rated “end of life.” In a typical NAND flash memory system, an ECC is generated for the data to be written, which may be stored with that data as written (as shown) or in a separate memory associated with that data. When the written data is read in response to a host request, a new ECC may be generated based on the read data and compared to the stored ECC (which may include embedded “EDC” or error detection code) to detect and correct erroneous data bits. Following is a diagram of an on-chip ECC from Brewer (Ex. 1026) at 87-88 showing the writing of user data D into the memory with generation by an ECC encoder of ECC correction bits C, stored in the memory, such that when the written data WD is read with the original C, a new code S is generated based on the read data WD to indicate which bits of WD are different from the original D to

allow correction of those erroneous bits to output D:

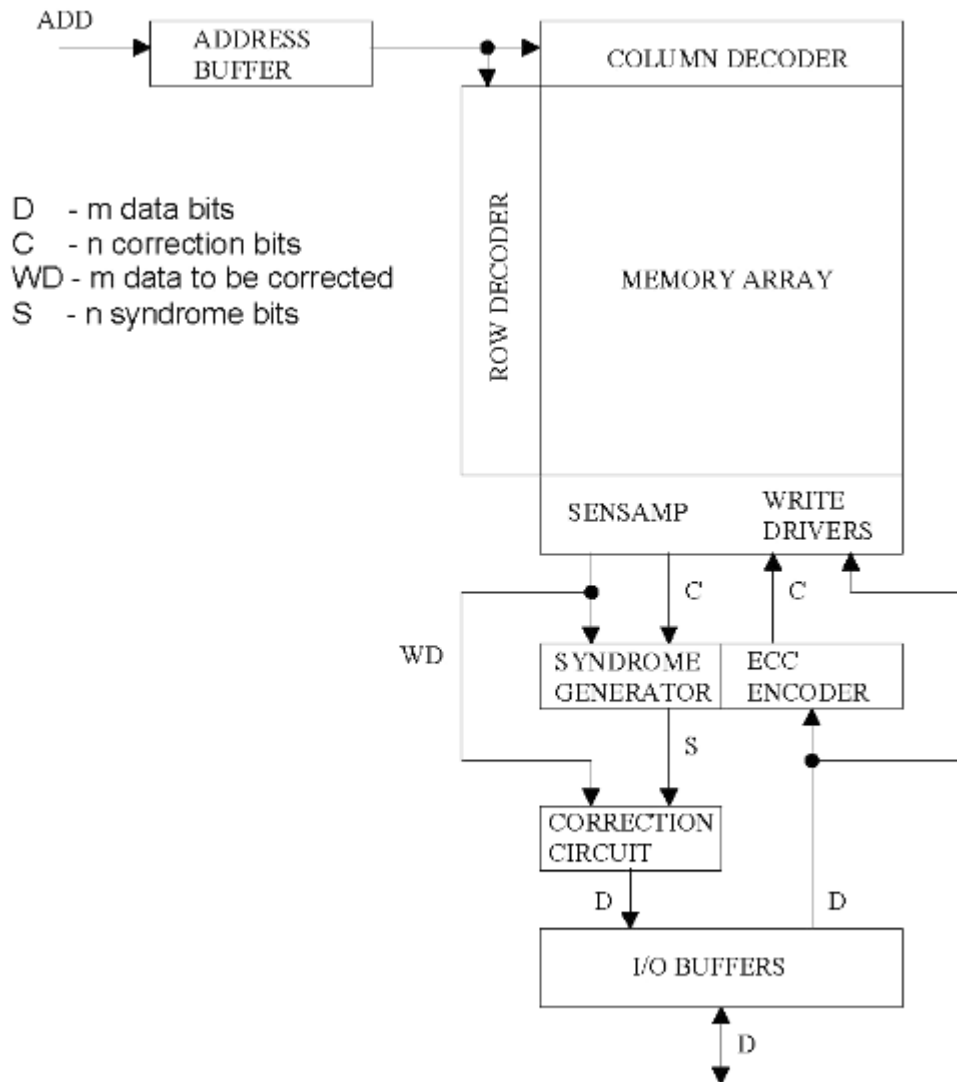


Figure 3.20. On-chip ECC for Flash memory.

EDC may be as simple as one parity bit per word or byte. More elaborately, if the number of EDC bits is the base-2 logarithm of the number of bits in a word, this enables a Hamming ECC to be applied to correct one erroneous bit in the word. The use of CRC enables the detection (not correction of appreciably more than one erroneous bit per word. (Brewer at 548-52; see Micheloni at 487 [cyclic redundancy

code or “CRC” for error detection].) Multi-bit errors can be corrected by using more sophisticated approaches such as “BCH” or Reed-Solomon codes. (*E.g.*, Micheloni at 44, 393-409.) Particularly for execution of the more complex algorithms, the ECC processor may be in the device controller (¶ 31 above, showing Micheloni Fig. 2.33; *also* Brewer at 551-52), and with auxiliary memory (Micheloni at 508, Fig. 17.21 [used to store code bits]). With enough processing resources and ECC space needed (which may be external to the chip) to correct multiple error bits, nothing needs to be done with the data (WD) with the error bits in the memory system, as what is read from that data will be corrected to what was received to be written (D). However, as a flash memory ages, the number of required error correction bits for a block may exceed the storage available for error correction bits and that block will have to be retired. As more and more blocks are retired, at some point the flash memory is no longer useful.

53. Gavens (Ex. 1045), published April 28, 2011, is one of a portfolio of SanDisk patents, many incorporated by reference in the context of a NAND flash memory system including “[a] memory chip 100 [that] includes a memory array 200 of memory cells with each cell capable of being configured as a multi-level cell.” (8:18-20.) Gavens discloses the use of ECC similar to that shown in the preceding paragraph:

In the types of memory systems described herein . . . the integrity of the data being stored is maintained by use of an error correction technique. Most commonly, an error correction code (ECC) is calculated for each sector or other unit of data that is being stored at one time, and that ECC is stored along with the data. The ECC is most commonly stored together with a unit group of user data from which the ECC has been calculated. The unit group of user data may be a sector or a multi-sector page. When this data is read from the memory, the ECC is used to determine the integrity of the user data being read. Erroneous bits of data within the unit group of data can often be corrected by use of the ECC. (3:27-39.)

As data is received from a host, a page of data is staged in the controller 102 and its ECC 76' is computed by the ECC processor 62. The data page incorporating the ECC is then written to the memory array 200. Typically, when the data page is read, the data page is latched in the data latches 430 and shifted out of the I/O circuits 440 to the controller 102. At the controller 102, the data page's existing ECC is compared to a second version of the ECC computed on the read data. The ECC typically includes an error detection code ("EDC") for rapid detection of any error in the data page. If the EDC indicates the existence of any error in the read data page, the ECC is invoked to correct erroneous bits in the read data page. (12:64-13:8; *also* 29:33-62.)

This is the preferred approach for checking for and correcting errors (without the need to repeat the process of programming (and erasing before the next programming)). However, although “[t]he ECC can be designed to correct any

number of bits[, t]he more bits it has to correct, the more complex and computationally intensive will the ECC be.”²⁰ (13:9-11.) Thus, in a section entitled “Enhanced Post-Write-Read Error Management” (“EPWR,” 19:37-24:33, Figs. 18-21), Gavens discloses the delay of application of error management to a device (including post-write-read²¹) until after a “hot count” of “the number of times [an] erase block has been cycled through erase and program operations” exceeds a

²⁰ “[F]or most of the lifetime of the device, the ECC is only marginally utilized, resulting in its large overheads being wasted and realizing no real benefits.” (Gavens 3:67-4:2.) Gavens explains that the design of ECC is “based on the expected worse case cell error rate (‘CER’) at the end of life (‘EOL’) of the memory device,” based on statistical likelihood of errors among a given population of cells. (13:11-35, Figs. 10A and 10B.) It reviews the sources of post-write errors for a sample memory device (13:40-15:17, Fig. 11) with number of “program-erase cycles” being the major factor and number of reads (disturbs correctable by “read scrubs”) a much smaller factor, and concluding that, near EOL (10K p-e cycles), “the ECC must be capable of correcting at least 21 error bits” (15:18-58, Fig. 12).

²¹ This is distinguished from write verification, ordinary read (retrieval of data, which may apply ECC), periodic refresh or “read scrubs” and other background processes such as wear-leveling and garbage collection.

predetermined threshold (5:31-6:2, 19:38-20:29, Fig. 18) and an embodiment in which data is first “staged” in a “less error prone, low density [1-bit] storage portion of the memory (D1)” and “subsequently folded into [one physical page of high density, 3-bit] (D3)” (20:30-35). (That is, three logical pages are folded into one physical (D3) page.) When the EPWR is enabled (Fig. 22A [820], Fig. 22B [820]),

... a current filled block in D3 is read back; and if the error rate exceeds a predetermined threshold, the current D3 block is rejected and a retry takes place with data being refolded into a new D3 block. The new D3 block is again read back and checked for excessive error rate. If the new D3 block passes, then it has good data and the original data in D1 is made obsolete. If the new D3 block again shows excessive error rate, the new D3 block is again discarded. If the excessive error rate persists after a predetermined number of retries, no further retry is attempted and the D1 to D3 folding operation is abandoned with the original data kept at D1. At this point the memory device is deemed too old for further programming operations and is made read-only to preserve the integrity of the existing data stored in the memory device.

(20:36-50; details and variations at Figs.20A-20C and explaining text at 19:51-23:29.) Optional process 864-865 retires a D3 block if it has been retried more than a pre-determined maximum number of retries (Fig. 22C explained at 24:11-25), serving bad block management.

54. While this embodiment describes the use of ECC to indirectly determine the number of bit errors, Gavens explicitly does not so limit its

embodiments. Gavens also discloses alternatives of direct comparison of the data written into memory (and then read back in “post-write read”) with a cached copy of what was to be written, among various embodiments under the rubric “Adaptively Rewrite Data from a Higher Density Memory Portion to a Lower Error Rate Memory Portion to Control Error Rate” (15:59-61). In one group of adaptive rewrite embodiments (Figs. 14 and 15 and associated text at 16:13-17:42), user data is written to the higher density (*e.g.*, MLC) portion as a “first copy.” (16:41-45.) Then, in a “post write read,” there is a check for errors “either by comparison with the original copy²² which may be cached **or** by checking the EDC portion of the ECC” (16:46-49 [emphasis added]). If the number of error bits is less than a predetermined amount, the first copy is deemed valid and subsequent reads of the data page will be from that copy with any errors corrected by ECC. (16:50-56.) If the number is greater, then a copy is written to the lower density, lower error rate (more robust, *e.g.*, “SLC”) portion; this is performed in one embodiment by copying from a cached

²² Obviously, this “original copy” is upstream of the “first copy” referenced in the preceding paragraph, likely off the memory chip in some cache (either in the controller or outside) from which the device controller directs data to be written to the memory chip as the “first copy” in its higher density portion.

copy (“of the original data”) or in another embodiment from application of ECC to correct error bits in the first copy. (17:9-14.)

55. In a second group of adaptive rewrite embodiments (Gavens Figs. 16 and 17 and associated text at 17:43-19:24), the more robust (“SLC”) portion is further divided into a first section for caching incoming data (simultaneously with a “first copy” written into the higher density “MLC” portion) and a second section for storing rewrites from the higher density MLC portion. (18:21-29.) In one of these embodiments,

the memory array is provided with a set of data latches on an integrated circuit chip, the checking of the error bits in the first copy is accomplished by loading the first copy and the cached copy into the set of data latches and making a comparison at the set of data latches.

(18:30-35 [emphasis added]; 18:40-44 [details].) If the number of error bits is less than a predetermined amount, the first copy is deemed valid and subsequent reads of the data page will be from that copy with any errors corrected by ECC. (18:45-52.) If the number is greater, then a copy is rewritten from the cached copy to the rewrite section of the more robust SLC portion. (18:53-66.) This then shows that when the number of errors in a copy written in MLC (for example) exceeds a certain amount, then that data is moved (rewritten) to SLC.

56. U.S. Patent No. 8,806,301 to Yu, published April 14, 2011 (“Yu ‘301,” Ex. 1046) uses an error checking and correcting unit 210 to determine the number of error bits in the data read after writing:²³

. . . when the memory management unit 204 receives a host write command from the host system 1000, the error checking and correcting unit 210 generates an error checking and correcting code (ECC code) corresponding to write dat[a] of the host write command, and the memory management unit 204 writes the write data and the ECC code into the flash memory chip 106. And, when the memory management unit 204 reads data from the flash memory chip 106, the memory management unit 204 simultaneously reads the ECC code

²³ Yu ‘301 discloses an “exemplary embodiment” of “a multi-level cell (MLC) NAND flash memory chip” as the “flash memory chip 106,” but notes that another embodiment would have that flash memory chip as the distinct “single-level cell (SLC) flash memory.” (7:5-9.) For each of those distinct memories, not disclosed to be used simultaneously, if the ECC unit (which may be on-chip, Fig. 6 [620]) finds an error bit and the number of error bits measured exceeds either of two thresholds (maximum correctable bits, Fig. 5 [S509] and Fig. 7 [S709] [on-chip ECC responding to confirm write command], and pre-determined threshold level, Fig. 5 [S517] and Fig. 7 [S717]), the physical block is marked a bad block (Fig. 5 [S521] and Fig. 7 [S721]).

corresponding the read data, and the error checking and correcting unit 210 performs the error checking and correcting process to the read data based on the ECC code corresponding to the read data. In the present exemplary embodiment, the maximum number of error bits which can be corrected by the error checking and correcting unit 210 is 26. . . .

(6:15-29.) However, “in another exemplary embodiment of the present invention, the memory management unit 204 may compare the read data and the writing data from the host system 1000 bit by bit to determine whether the read data has any error bit and the number of error bits thereof.” (11:49-53 [emphasis added]).

57. A bit-by-bit comparison of a page of user data to be written (held in some temporary memory, or cache) with the data read from the memory cells where it is written requires more storage and processing than typical ECC schemes using a controller. The ECC schemes do not require the storage of the “page of user data to be written,” and the number of bits used for ECC is a tiny fraction of the bits in a whole page of data. The ECC bits are generated and stored upon writing; then upon reading a page from the memory the ECC bits are also read and used to ascertain whether or not there are errors, and to determine which bits (if any) are erroneous. While there are the alternative bit-by-bit embodiments recited in Gavens and Yu – which necessarily (inherently) would require adequate memory (typically volatile) available to the controller to hold a to-be-written page to compare with a written-and-read page – I am unaware of any commercial use of bit-by-bit “comparison” for

read-after-write “integrity checking” for flash memory management. In my view, it would be an inefficient choice. Keeping in mind that a page of data is on the order of 8kB or 64k bits, it is a daunting task to compare two 64k-bit data items bit-by-bit. Typically, a subtractor is used to do the comparison, and this takes place using an adder, since subtracting two vectors ($A - B$) is equivalent to adding A to the complement of B . Now the largest adders handle about 64 bits (8 bytes). That means the adder would have to be applied 1000 times! Or you would need 1000 such adders to accomplish the task in the time it takes for one 64-bit adder to execute. A combination of the two approaches (serial and parallel) could also be attempted. It is also necessary to accumulate the total number of errors, so each 64-bit addition result must be accumulated (all 64-bit adder results must be added together) in a memory or register. Overall, this takes an enormous amount of resources and time, and a POSITA would not expect this method to be used in industry, even though it is theoretically a possible approach.

F. Strategies for NAND Flash Memory Remapping To Manage System Reliability and Efficiency

58. Prior to July 19, 2011, there were already many related strategies for avoiding enough memory components of a NAND flash memory system reaching the end of their lives before other components thereby degrading whole system reliability and its effective “lifetime.” An obvious general strategy as discussed at

paragraph 35 and 39 above, is to avoid overstressing components with lower endurance by sending data that are likely to stress components by frequent updating to components with higher endurance. As reviewed at paragraphs 46 to 48 above, Lee '794 (Ex. 1041), Sutardja (Ex. 1042) and Moshayedi (Ex. 1043), each applied the strategy of dynamically (at the time of writing) sending to SLC memory data associated with logical addresses likely to be frequently updated. (Also Duann [Ex. 1034] at 10-12.)²⁴ Another strategy (also disclosed in Sutardja and Moshayedi) is to make sure that all blocks in the system reach their end-of-life at the same time by maintaining uniform wear. With a system using a homogenous type of memory, traditional static wear leveling, moving apparently “cold data” to more worn blocks, has been found to be most effective. (Note 13 above.) Systems with memories of different endurances present complexities, but there are many solutions proposed in the prior art.

59. U.S. Patent Appl. Pub. No. 2010/0115192 to Lee (“Lee ‘192,” Ex.

²⁴ This strategy may be analogized to the array method of dynamic wear leveling explained at paragraph 36 above, in which each new write is sent to the least worn block of an available (erased) block list, where SLC blocks are the least worn, but only data expected to be updated frequently is allocated to SLC which is more expensive per bit of storage than multi-level memory.

1047, Samsung assignee) disclosed wear leveling in a memory system with both SLC and MLC blocks (Fig. 2), such that if the sum of “SLC mode usage” plus “MLC mode usage” minus an average usage of all blocks is greater than a threshold set for the mix of applications for the SLC (faster and more frequent writes) and MLC blocks, then wear leveling is performed (Fig. 4), in which the mapping of the LBA is changed to a new physical location, defining a new logical/physical memory block relationship (Lee ‘192 ¶ [0023]). Duann taught the use of global wear leveling, treating “all NAND components as one memory unit” (Ex. 1034 at 5, 9).

60. As reviewed at paragraphs 47 and 48 above, Sutardja (Ex. 1042) and Moshayedi (Ex. 1043), disclosed embodiments for preferential (both “dynamic” at the time of updating and “static” in the background) writing into SLC or MLC modules that also considered actual wear on memory blocks as a predictor of future wear. As explained in an article contemporaneous with the filing of the application leading to the ‘298 patent (upon which the ‘369 patent is based), Minyoung Sung & Kanghee Kim, *Asymmetric Flash Volume Management*, 58 *IEEE Transactions on Consumer Electronics* 455 (May 2012) (“Sung,” Ex. 1037), the asymmetry between SLC and MLC is not just endurance, but in the organization of page and block

capacity to be considered relative to user data.²⁵ Thus, an aspect of preferential writing is “to maximize the overall read/write performance of the logical volume by (1) hot data identification and (2) periodic data migration between SLC and MLC [with the objective of providing] the highest overall read/write throughput for the next period.” (*Id.*) In one implementation, “write operations from a process with a high storage update frequency are marked ‘hot’ before forwarded to the flash storage system.” (*Id.* at 456; *compare* Lee ‘794 [Ex. 1041] discussed at ¶ 46 above.)

61. In addition to these strategies of preferential mapping and wear leveling, at least another group of tools was available to avoid unnecessarily early failure of a NAND flash system: error management by remapping (and writing) to binary memory when the bit-errors in writes to one multi-level block or successive

²⁵ At the time of the ‘298 patent application (2012), a marketable trade-off between cost, performance, and endurance was a hybrid use of SLC to MLC modules in the proportion of 1:4 (*E.g.*, Duann at 14 [“72 GB hybrid SSD (8GB SLC + 64GB MLC)”]; Moshayedi Fig. 1 [four channels (banks) 108 each connecting “five chips of flash (e.g., the first chip can be a K9K8G08 SLC flash, and the rest can be K9G8G08 MLC flash)” {¶ [0038]}{same as cited in Duann at 14}].) These proportions of SLC and MLC flash memory are reflected in Fig. 4 of the ‘369 patent and its suggested proportion (5:18-20).

writes to multi-level blocks reach predetermined thresholds. (*E.g.*, Gorobets ‘179 [Ex. 1044], Segal [Ex. 1048], Gavens [Ex. 1046, *see* ¶¶ 53-55 above]; *see also* Brewer [Ex. 1026] at 87, 89-90 [ECC to increase robustness of multilevel flash, correcting errors at each level {page}].) The strategies are used compatibly, not exclusively; indeed, an objective of wear leveling was to avoid the overwhelming of ECC by excessive errors (Brewer at 549-52).

62. These strategies and knowledge of NAND flash systems and operations prior to July 19, 2011, make obvious the independent claim 1 of the ‘369 patent.

VI. Person of Ordinary Skill in the Art (POSITA)

63. I am informed by counsel that a “person of ordinary skill in the art at the time of the inventions” (“POSITA”) is a hypothetical person who is presumed to be familiar with the relevant field and its literature at the time of the inventions. This hypothetical person would also be a person of ordinary creativity, capable of understanding the scientific principles applicable to the relevant field.

64. In view of this instruction, my experience with the issues, and the ‘369 specification, in my opinion, a person of ordinary skill in the art at the time of the claimed inventions would have a bachelor’s degree in computer engineering, electrical engineering, computer science, or a closely related field, along with at least two years of experience in the design, development, implementation, or management of memory devices and systems. A person with an advanced degree in a relevant

field, such as computer or electrical engineering, would require less experience in the development and use of memory devices and systems. As is common, one could obtain equivalent knowledge and perspective from other life experiences as well.

VII. Disclosure of the ‘369 Patent Specification

65. Having read many patents, the specification of ‘369 patent, which incorporates Rao ‘916, struck me as disclosing nothing that might even be considered to be new, and included only an imprecise or inaccurate description of two variants of well-known, optional NAND flash operations. The purportedly inventive memory system was disclosed in the common specification as well-known MLC NAND flash memory modules and well-known, distinct SLC NAND flash memory modules in the known useful proportion (note 25 above), with the well-known superior endurance of SLC over MLC. A third component of the disclosed memory system, “coupled” to the other two, is a “controller” that is assumed (without explanation of structure such as memory or algorithm) to perform four functions in the DETAILED DESCRIPTION, the first being L2P/FTL mapping that is known and essential for any NAND flash memory. The second function is shown as a flowchart (Figs. 3A and 3B) of a “method for utilizing a NAND flash memory,” explained at a level omitting details of NAND flash structure and operations, such as any reason to remap to the SLC module upon “failure” of a “data integrity test.” The third and fourth functions are even higher level restatements of two sentences

in the specification that describe what may be considered dynamic *or* static wear leveling. As I review in the next section, there is no more. There is no structure disclosed other than by implicit assumption of known structures and necessary and common operations of MLC and SLC NAND flash memory modules and the controller with other ordinary memory – certainly no innovative structure. There is no explanation of how the controller performs the recited functions, only a reliance on known NAND flash memory systems, admitted in the BACKGROUND to have been operated using L2P/FTL mapping and forms of wear leveling.

A. ‘369 Specification BACKGROUND

66. The specification of the ‘369 patent, admits a “BACKGROUND” state of the art that includes bits of relevant NAND flash background that I reviewed in Section V above, particularly that it “is well known in the art, any ‘write’ or ‘program’ to a block in NAND flash (floating gate) requires an ‘erase’ of a block before ‘write.’” (Ex. 1001 3:48-51). The concepts and vocabulary of the disclosure is established, starting with the history of HDD organization into 512-byte sectors having a unique “physical (memory) address” where sectors are combined to form a “logical block” having a unique “logical address.” (2:28-36.)

. . . A computer [operating system] organizes **data** as “files.” Each file may be **located (stored)** in either a single **logical block** or a plurality of logical blocks, and therefore [by mapping], the location of files typically traverses the boundaries of individual (physical) sectors.

(2:39-43 [emphasis added].) It was recognized that generally NAND flash memory is organized in blocks of 32 to 64 pages, each with 2-4 Kbits (2:55-57). The erase-before-write limitation of NAND flash is described:

the process of writing data to a NAND flash memory device is complicated by the fact that, during normal operation of, for example, **single-level storage (SLC)**, erased bits (usually all bits in a block with the value of `1`) **can only be changed** to the opposite state (usually `0`) **once before the entire block must be erased**. Blocks can only be erased in their entirety, and, when erased, are usually written to `1` bits. However, if an erased block is already there, and if the addresses (block, page, etc.) are allowed, data can be written immediately; if not, a block has to be erased before it can be written to.

(2:57-67 [emphasis added].) L2P mapping is also described:

FTL is the driver that works in conjunction with an existing operating system (or, in some embedded applications, as the operating system) to make linear flash memory appear to the system like a disk drive, i.e., it emulates a HDD. This is achieved by creating "**virtual**" **small blocks of data**, or sectors, **out of flash's large erase blocks** and managing data on the flash so that it appears to be "write in place" when in fact it is being stored in different locations in the flash. FTL further manages the flash so that there are clean/erased places to store data.

Given the limited number of **writes** that individual blocks within flash devices can tolerate, **wear leveling** algorithms are used within the flash devices (as firmware commonly known as **FTL** or managed by a

controller) to attempt to ensure that **"hot" blocks**, i.e., blocks that are frequently written, are **not rendered unusable much faster than other blocks**. This task is usually performed within a flash translation layer. In most cases, the controller maintains a lookup table to **translate the memory array physical block address (PBA) to the logical block address (LBA) used by the host system**. The controller's wear-leveling algorithm determines which physical block to use each time data is programmed, eliminating the relevance of the physical location of data and enabling data to be stored anywhere within the memory array and **thus prolonging the service life** of the flash memory. Depending on the wear-leveling method used, the controller typically either writes to the available **erased block** with the lowest erase count (**dynamic wear leveling**); **or** it selects an available target block with the lowest overall erase count, erases the block if necessary, writes new data to the block, and ensures that **blocks of static data** are **moved** when their block erase count is **below** a certain threshold (**static wear leveling**).

(3:1-33 [emphasis added].) These actions are not stated to be sequential or relating to the same data or block; the static wear leveling can be done in the background. The description of dynamic wear leveling is consistent with the “first level” (dynamic) wear leveling using the “array method” (tracking “age”), which I described in paragraph 36 above. The description of static wear leveling is consistent with the “second level” (static) wear leveling described in the same paragraph as a “move” of “static data” triggered on the condition of a difference

between maximum and minimum p-e cycles among blocks reaching a predetermined threshold. The specification's threshold would not make sense for any particular physical block, since they each have increasing p-e cycle counts. It is considered beneficial to move static ("cold") data in a physical block less worn than average to one more worn than average where the data, *presumed* to be mapped to a logical address (logical data or block) not often updated, and hopefully can stay in the more worn physical block without being updated. (An update would result in further wear on erasure in garbage collection.)

67. The BACKGROUND states that the prior art wear leveling of the preceding paragraph, by "determin[ing] which physical block to use each time data is programmed" to be written, and thus "eliminating the relevance of the physical location of the data and enabling data to be stored anywhere within the memory array," thereby – without further explanation – "prolong[s] the service life of the flash memory." (3:21-26.)²⁶ The BACKGROUND states that "various embodiments

²⁶ For general wear leveling, there is stated the objective that "hot" blocks not to be "rendered unusable much faster than other blocks" (3:14-17). However, without other explanation, the specification recites a need for "writing data optimally for improved life time (endurance) of flash memory" (3:63-65), embodiments of NAND

of a NAND flash storage system that provides long lifetime (endurance) storage at low cost are described herein.” (3:65-67.) But nowhere is such long lifetime at low cost explained, other than by implication from the background establishment that “MLC NAND flash SSDs are slowly replacing and/or coexisting with SLC NAND flash” (3:34-35) in 2012, where the “MLC NAND flash enjoys greater density than SLC NAND flash [implying MLC’s lower cost per memory cell], at the cost of a decrease in access speed and lifetime (endurance)” (3:39-42). It is implied from juxtaposition of the background’s stated endurance of SLC NAND flash as 50-100K

flash “that provides long lifetime (endurance) storage” (3:65-67), the use of SLC “to boost the lifetime (endurance) of the storage system (5:13-15), and of a system and method of “increasing the reliability and lifetime of a NAND flash storage system, module, or chip” (7:13-14). I understand from my experience and the literature, and believe that a POSITA in computer memory would understand, that the improvement sought in the field and referenced in the ‘369 patent specification is to avoid premature failure (actual or rated, such as the “hot blocks” at 3:14-17) of enough memory components to result in degradation of a hybrid SLC-MLC NAND flash system making it no longer reliable for its intended use. This “improvement” is relative to a hybrid NAND system configured for projected uses (such as enterprise versus consumer) balanced with cost and performance.

writes and of MLC NAND flash as 3-10K writes (3:42-48²⁷; see ¶¶ 44, 49 above). Juxtaposed against a summary of remapping to the SLC module after failure of the ‘369 patent data integrity test, it is stated, without explanation, that “[a]s the SLC NAND flash is used to boost the lifetime (endurance) of the storage system, it can be considerably lesser in amount than the MLC NAND flash” (5:15-18). But if lifetime were the only consideration, its extension (or rather, avoidance of premature ending/failure) would be boosted by a *greater* amount of SLC NAND flash. It is suggested, without explanation, that SLC memory storage space be 1:8 to 1:4 of MLC memory storage space (5:16-18), which is also reflected in Fig. 4 (2+ SLC memory modules [62a and 62b] to 8+ MLC memory modules [60a and 60b]) and conforming with Duann (one of the IDS documents submitted with the ‘298 patent application) at 14 (“72 GB hybrid SSD (8GB SLC + 64GB MLC)”). These ratios, relevant to the purported balance of cost, performance, and endurance, were not recited in the claims to limit the system to a configuration of useful balance of cost and lifetime. Instead, what was disclosed by the ‘369 patent were methods of using

²⁷ The text also explains that writing to a floating gate NAND flash block requires prior erasure (3:49-51), without explaining that it would be disruptive to erase immediately before write, an important consideration in performance and endurance management (*e.g.* ¶¶ 36-39 above).

a system of at least one MLC module and at least one SLC module, with a controller performing the L2P/FTL functions essential for NAND flash operation.

B. Proffered Embodiments in the DETAILED DESCRIPTION

68. The only embodiment described in the ‘369 patent specification “DETAILED DESCRIPTION” is a system of known SLC memory modules, known MLC memory modules (their relative expected endurances given in the BACKGROUND at 2:25-27 and 3:42-48) and an abstract controller *not* limited to a known hardware or software module – but assumed to have known functionality.²⁸ Instead, the specification and claims attempt to cast the claimed system (rather than its operation) as new and unobvious by functionally defining the abstract controller

²⁸ The specification states that the controller *may* be a “chipset” (5:22) but does not otherwise disclose the controller’s structure or how it is adapted to perform the claimed functions. Incorporated by reference Rao ‘916 (Ex. 1004) at least provided (1) a functional block diagram of a “monolithic” (single chip) controller used with NVMS flash memory (Fig. 3, compare with Micheloni Fig. 2.33 reproduced at ¶ 31 above) and (2) a circuit block diagram of a commercial “NVMS Flash Memory” (Fig. 6 [by its Fig. 8 rated endurance of 100K p-e cycles, an SLC module], compare with Micheloni Fig. 15.1 reproduced at ¶ 29 above). Nothing like this was further provided in the ‘369 specification.

as performing (1) admitted, ordinary, necessary L2P/FTL operations (5:39-6:10, 6:64-67), (2) a two-sentence variant of admitted, ordinary wear leveling (6:49-61), and (3) a facially straightforward “data integrity test” with remapping to the SLC module upon failure, shown by Figs. 3A and 3B and a two-paragraph explanation of that “method for utilizing a NAND flash memory system” (6:11-28). The “embodiments” of replacement of the “SLC module” (4:33-38) with “flash alternatives” with greater endurance at least because they “allow data to be written over existing data” (2:19-23) are suggested without other support such as any modifications needed to the NAND flash memory or controller structure (even L2P/FTL) for non-flash operations.

69. SLC and MLC modules’ physical and operational foundation:

The proffered innovative system is stated in the “DETAILED DESCRIPTION” as the combination of POSITA-understood physical SLC and MLC NAND flash memory modules and an abstract controller:

Generally, and in particular regard to NAND flash memory, two separate banks of NAND flash are maintained by a controller. One bank contains economical MLC NAND flash, while a second bank contains high endurance SLC NAND flash.

(5:6-10.) The only alternative embodiment is shown in Fig. 4 and described as

two NAND flash memory banks **56** and **58**, whereby memory bank **56** comprises a plurality of MLC NAND flash memory

modules 60a and a plurality of SLC NAND flash memory modules 62a. Memory bank 58 comprises a plurality of MLC NAND flash memory modules 60b and a plurality of SLC NAND flash memory modules 62b.

(6:66-7:5 [emphasis added].)²⁹ In language reflecting the first ‘298 patent claim 1 (‘298 7:8-8:9), the “SUMMARY” states,

there is provided a system for storing data which comprises at least one MLC nonvolatile memory module (hereinafter referred to as “MLC module”) and at least one SLC non-volatile memory module (hereinafter referred to as “SLC module”), each module comprises a plurality of individually erasable blocks. The data storage system according to one embodiment of the present disclosure further comprises a controller for controlling both the at least one MLC module and the at least one SLC module. In particular, the controller maintains an address map comprising a list of individual logical address ranges each of which maps to a similar range of physical addresses within either the at least one MLC module or the at least one SLC module. After each write to (flash) memory, the controller conducts a data integrity check to ensure that the data was written correctly. When the

²⁹ The term “module” is used throughout the specification, including in the context of alternative physical items: “NAND flash storage system, module or chip” (1:44-45, 7:14 [emphasis added], *also* Fig. 7 in Rao ‘916 (Ex. 1004) 1:31-32 [“IC’s mounted on a memory card or module”).

data was not written correctly, the controller modifies the table so that the range of addresses on which the write failed is remapped to the next available range of physical addresses within the at least one SLC module.

(4:15-33 [emphasis added].) A POSITA would understand “module” in this context to be a NAND flash memory chip or package such as the labeled “NAND” or “FLASH” in the block diagram (Micheloni Fig. 15.1) reproduced at paragraph 29 above and the photograph (Micheloni Fig. 1.16) reproduced at paragraph 27 above and in Figs. 6a and 7 of the incorporated Rao ‘916³⁰ (Ex. 1004). The “module” comprises the NAND flash memory array and input/output (including addressing, writing and sensing) circuitry. This understanding of “module” is compelled by computer electronics usage (*see* ¶ 30, note 8 above) and the inclusion of multiple individually erasable blocks, understood by a POSITA as a defining, characteristic structure of NAND flash memory. In fact, as explained at paragraph 43 above, “SLC memory” and “MLC memory” are physically distinct structures required for distinct programming and reading, even without additional compartmentalization by

³⁰ Figs. 6-8 show a “current” Samsung NAND flash chip K9F1G08XOA with a “1024M + 32M BIT NAND FLASH ARRAY.” With “ENDURANCE: 100K PROGRAM/ERASE CYCLES” (Fig. 8), this is clearly an SLC module. (*See* ¶¶ 44, 49 above.)

modularization. There is no subdivision of flash storage cells that could be usefully coupled to the controller without the input/output circuitry. A POSITA would also understand not only from their distinct recitation, but from essentially different capacity, organization and incremental writing in passes, that an SLC module is different from an MLC module. Operation of part (such as a block) of an MLC module in 2-state (although sometimes loosely called “SLC”) mode (§ 49 above) does not transform it into an SLC module both because of the distinct ordinary usage of “mode” versus “module” (a block of connected cells is not by itself a memory, much less a “module”) and the substantial operational, performance and endurance differences between “pseudo-SLC” and real SLC (*id.*). The only suggestion in the ‘369 specification of a beneficial hybrid SLC-MLC memory system called for “SLC NAND flash equal to 12.5% or 25% of MLC NAND flash” (5:20-22), which is consistent with Fig. 4 showing two banks each of 2+ SLC modules compared to 8+ MLC modules.³¹ The ‘369 patent uses the term “SLC module” to describe a distinct

³¹ See also Duann and Moshayedi ratios at note 25 above. This also corroborates a POSITA’s understanding that SLC modules comprise all SLCs as distinguished from MLC modules, rather than any partition for 2-state operation of MLCs: the specification for the first ‘298 patent was filed on April 25, 2012, nearly ten years

memory device for proposed replacement by alternative technologies – including comparison to “rotating storage media (e.g., HDDs).”³² This is consistent with ta

after issuance of Chen ‘528 (Ex. 1040), which disclosed the 2-state operation of MLC, along with at least the 2001 Cho Dual-Mode article (Ex. 1068), which Dr. Rao filed with his application resulting in the ‘298 patent. Despite this, the ‘298 patent specification described and claimed the controller-directed use of the distinct MLC and SLC modules mention nothing about such dual mode operation, even with dependent claims to specific alternative technologies. There is no suggestion of a partition of MLC modules for 2-state operation. The word “module” appears throughout, but “mode” not at all. This is the same for the ‘369 patent.

³² The SUMMARY states that “SLC module can be (NAND) flash, PCM, RRAM, MAGRAM or any other solid-state non-volatile memory as long as it has endurance that is superior to that of MLC flash, and it provides data access speeds that are faster than that of MLC flash or rotating storage media (e.g., HDDs).” (4:33-38 [emphasis added].) The ‘369 patent does not and cannot define “SLC” to include the alternative technologies, but merely posits that an “SLC module” may be a NAND “flash” or any other “solid state [physical] non-volatile memory” faster and having more endurance than MLC “flash” and hard disk drives. The clear implication is that these systems are physically complete and interchangeable with the SLC module.

POSITA's understanding of "SLC module" as a "self-contained, interchangeable" chip (note 8 above).

70. **The controller and defining, SLC remapping function:** Tying the subject SLC and MLC foundational modules together in the NAND flash memory system is a "controller." The term by itself is abstract, merely suggesting some "control" of the physical components, namely SLC and MLC NAND flash modules. The '369 patent specification might have recited a POSITA-understood NAND flash memory controller with the functions outlined at paragraph 31 above or in incorporated Rao '916 Fig. 3; it instead maintained its abstraction, stating in the SUMMARY reproduced above that in "one embodiment" there is a controller for "controlling" the SLC and MLC modules (4:22-23), particularly "maintain[ing]" a logical to physical address map (4:23-27). The function further defining this embodiment is "conduct[ing] a data integrity check" after "each write" "to ensure that data was written correctly," and if not, "modif[ying]" the L2P address map to remap into the SLC module (4:27-33). "[M]emory within the device controller" is recited for the integrity check (6:17), but nothing is suggested other than the *assumed* memory in or available to an ordinary NAND flash controller memory (§ 31 *supra*) (contrast the *modifications* suggested as "invention" at Rao '916, *e.g.*, 1:59-60).

71. **The Data Integrity Test operational feature:** The DETAILED DESCRIPTION states that "[t]he controller conducts a data integrity test after every

write” (5:10-11), but a further description of this “read-modify-write” function at 5:56-60 qualifies “every” with “as time permits.” (There is no suggestion when these dynamic checks are commenced, unlike the more efficient approach, as in note 29 above, of starting only near “end-of-life.”) The only structure for the function is set forth in the high-level method steps of Figs. 3A and 3B and the description at 6:11-48.³³ According to that description, in Fig. 3A (100) a command is received (presumably from the host) to write a “quantum of data stored in DRAM”³⁴ to a

³³ Figs. 3A and 3B of the ‘369 patent are virtually identical to the hand-drawn Figs. 3a and 3b in the Provisional (Ex. 1003, sheets 2 and 4) except for a renumbering. The descriptions are close (*compare* ‘369 6:11-48 with Ex. 1003 at 7-8, ¶¶ [0022] and [0023]) with some exceptions noted below.

³⁴ A POSITA would understand that (1) the abstract “quantum” refers to data from the host directed to a logical NAND block address (with sub-addresses) for that data (established on first writing), and (2) a NAND flash device controller would assign (on first writing or reassign for an update) a physical NAND block address and issue appropriate memory commands to be executed at the SLC NAND flash and/or MLC NAND flash module level according to their distinct physical structures and operations. The known write unit for NAND flash is a page. (¶ 27 above.) None

particular location in NAND flash (not limited to the MLC module or the SLC module). In step 102, that data is “read from DRAM into memory within the device controller (which acts as the memory controller)”³⁵ (6:16-19 [emphasis added]).

of this is disclosed for the alternative technologies. The specification also does not disclose how the quantum of data gets to DRAM in the disclosed system, which might be assumed to be DRAM 20 (see 5:56-57).

³⁵ The “controller” is connected in Fig. 1 (described at 5:19-38) as device controller 14 by lines (“coupl[ing]”) to processor 12 (host computer), DRAM 20, MLC flash 26, SLC flash 28 (such as “NAND flash memory modules, confirming their physical distinction) and other discrete devices such as disks 24. There is no explanation how it “controls” any of these devices beyond admitted background L2P mapping for the disk 24 and NAND flash. Other than the just quoted description of “memory within the device controller,” there is no description of the structure of a controller having internal memory into which contents of “DRAM” may be “read.” A POSITA would expect a NAND device controller to be a separate chip (see photograph at ¶ 30 above, Rao ‘916 [Ex. 1004] Fig. 3) with some volatile memory for cache and logic operations using the same chip die (*e.g.*, data registers in Rao ‘916 Fig. 3) and “external” DRAM (photograph at ¶ 30 above, DRAM 20 in ‘369 Fig. 1). Block/step

(Steps [102] and [104] of the flowchart in ‘369 Fig. 3A both refer to “memory of device controller.” The incorporated Rao ‘916 [Ex. 1004] Fig. 3 shows a “Data Reg.,” but it is unclear whether it would large enough to hold more than a flash page (~4KB) of data for the ‘369 Figs. 3A-3B operations.) In step 104, “both the logical address range and the NAND physical address range to which [the data] is to be written is read into memory of the device controller” (6:18-21). In step 106 that data (in that “controller memory”) is “combined with the contents of the NAND flash memory”³⁶ (6:21-23); it is unstated where and how this is done and where the results

102 and block/step 104 in Fig. 3A both refer to “memory of device controller,” without specifying it as “random access volatile memory,” what kind, how large it is and how data (host or address) is input, output or compared, as would be expected if this were offered as some advance in the art. (*Compare* Rao ‘916 1:45-49 [page buffers].) Yet, the ‘369 patent Challenged Claims are pinned on “retained data” in a newly recited “random access volatile memory” where such volatile memory is disclosed only as external DRAM 20 and the background “volatile memory for buffering, caching” (3:61-62).

³⁶ It is unspecified where these “contents” are. The Provisional stated that the data to be written “is combined with the present contents of the entire address range that

are held. In step 108, that “NAND flash physical address range to be written is erased.”³⁷ (6:23-24). Then, in step 110, “the combined data is written to the appropriate NAND flash physical address range.” (6:24-26.) In step 112, the combined data that was written in step 110 into NAND “is read into device controller memory”³⁸ (6:26-28 [emphasis added]) – which, again, is not otherwise identified

the data will be written to.” ¶ [0021]. In NAND flash operation, that doesn’t make any sense, since updates will ordinarily be to an erased location. The DETAILED DESCRIPTION leaves this to the NAND flash understanding of the POSITA and fails to disclose how the “appropriate” address range in step 110 is determined.

³⁷ This step is highly inefficient as a step close in time to the writing. To erase the address targeted, the entire block must be erased, and before that, valid (non-obsolete) data rewritten (decoded, buffered, encoded) to other blocks – disrupting, adding latency to system operations. Normally the erasure would be part of an optimized garbage collection process that could be performed in the background.

³⁸ It is unclear whether “device controller memory” is the previously referenced “memory within the device controller,” which is not explained as part of the device controller physical structure (note 35 above), or DRAM 20, which is shown as coupled to the device controller and thus might be considered “device controller

in the specification. Then, in step 114 “the NAND flash physical address range that was read into device controller memory is compared to the retained data representing the combination of the previous contents of the physical address range and the quantum of data to be written”³⁹ (6:29-34 [emphasis added]). If at branch 116 there is no “match” in the step 114 comparison,⁴⁰ and if there is available SLC memory,

memory” or “memory of device controller” of Fig. 3A steps 102 and 104. Nonetheless, the ‘369 patent Challenged Claims are pinned on “retained data” in a “random access volatile memory” first recited in the ‘300 patent application in 2018.

³⁹ The text fails to explain, as in block 114 of Fig. 3B that the data written in that physical address range is compared, not the address range. It is not specified where the “retained data representing the combination” is held or compared or how. It can be surmised that it is in “controller memory” (whatever that is) where the data to be written was read **from** DRAM and combined with valid data from flash, since that combination is not in DRAM and no reason is provided for writing it back to DRAM.

⁴⁰ This “match” language strongly suggests in context that the ‘369 patent’s read-after-write integrity check is a bit-by-bit comparison between data. As I explained at paragraph 57 above, bit-by-bit comparison is inefficient. It is particularly inefficient if made after “every write” (5:10-11), severely multiplying latency.

“the failed NAND flash physical address range is remapped⁴¹ to the next available quantum of SLC NAND flash memory” and the write process from step 110 repeated. (6:36-48 [emphasis added].) Nothing is said (for example, for bad block management) about flagging or retiring the physical address range in which data was written and found not to match the data that was mapped to be written there: if that range is not retired, there may be problems in the future resulting in wasteful checking, remapping and rewriting, adding to wear. On the other hand, if the range is retired early in device lifetime, that is also wasteful, taking memory out of service due to a single error. That would lead to earlier exhaustion of system capacity (and thus life) compared to prior art ECC schemes, which may be deployed later in device life, allowing use (corrected reading) of even incorrect data until error rates exceed chosen thresholds of correctability (leading to bad block identification and retirement) (¶¶ 51-53 above). Nowhere is the specification-described data integrity test shown or explained to actually contribute to reliability or lifetime. The

⁴¹ It doesn’t make sense that the physical address range is remapped to another physical location, since mapping is between logical addresses and physical addresses. There is confusion among the “physical address range” as data (or metadata), the information represented in the “physical address range,” and the memory cells of the “physical address range.”

description ignores the necessary NAND flash operations, which include writing by encoding digital data into analog memory cells and reading by decoding, even before comparison is made, and, for MLC operations, multiple steps for each writing and reading (§ 43 above). The described integrity test embodiment is inoperable without rewriting the description to resolve ambiguous and inconsistent subjects and objects of its steps and assuming physical structures (such as the controller memory) and NAND flash operations. Any claim to novel, non-standard components, different from well-known hybrid MLC-SLC flash system, was unsupported in the DETAILED DESCRIPTION by *any* disclosure, much less enabling disclosure.

72. **Known, essential L2P/FTL as an “additional embodiment”:** Fig. 4 is described as “another embodiment” where “FTL” logic is “entirely resident” in a NAND flash module that includes two banks of NAND flash memory banks of SLC and MLC NAND flash modules. (6:63-67.) There was nothing new about FTL. I explained at paragraph 34 above the prior art FTL’s and linked lists roles in NAND flash-required L2P mapping, wear leveling and garbage collection. FTL is admitted prior art in the BACKGROUND with both L2P mapping and wear leveling functionality in a software layer (3:1-33). Nothing is added by the Fig. 4 “alternative” embodiment – only labeling a disembodied “logic” in a physical module that includes MLC and SLC modules. Such a module is typically a NAND flash memory card as depicted at paragraph 31 above (Micheloni Fig. 2.33) to

include a flash file system “implemented as firmware operating in different functional sublayers for wear leveling, garbage collection and bad block management” such as a Flash Translation Layer.

73. Wear leveling operational feature as an additional “embodiment”:

Another “embodiment” is proposed in a two-sentence DETAILED DESCRIPTION possible restatement of the BACKGROUND “wear leveling” (3:21-33) as applied to the hybrid SLC-MLC module system. The first sentence states:

Another application of one embodiment of the present disclosure, not depicted in any of the drawings, is to allocate “hot” blocks; i.e., those blocks that receive frequent writes, into the SLC NAND flash memory module **28**, while allocating “cold” blocks; i.e., those blocks that only receive infrequent writes, into the MLC NAND flash memory module **26**.

6:49-55.) Although this resembles the BACKGROUND dynamic wear leveling (3:27-28), the DETAILED DESCRIPTION leaves undetermined whether the referenced blocks are the *logical* blocks to which a host directs data frequently or infrequently (like “hot” or “cold” *data* as explained for garbage collection⁴²) or

⁴² As reviewed at paragraph 37 above, the Micheloni textbook concept of “hot data” relates to clustering for garbage collection efficiency often-updated data associated with logical blocks.

physical blocks that may be worn by frequent or infrequent writing of *different* data (since updating data generally means writing to a different physical block). On its face, this embodiment executes the prior art strategy of directing – dynamically, at the time of host-requested writing – to SLC or MLC “hot” or “cold” *data*, respectively.⁴³ Nor is it explained what allocation “into” SLC or MLC modules means and when and how it is performed.⁴⁴ The second sentence appears to be a sub-embodiment within the same process described in the first sentence:

⁴³ At paragraph 46 above, Lee ‘794 (Ex. 1041) is shown to write frequently written (“hot”) data such as system data and user directories (with associated logical blocks) to physical SLC blocks and less frequently written data such as user data to physical MLC blocks, also directing “hot data” by various logical block counts. (Also the base writing to SLC or MLC by Sutardja and Moshayedi as explained at paragraphs 47 and 48 above.)

⁴⁴ The Provisional favors the logical block or “data” interpretation, describing a “use of the invention to segregate hot blocks; i.e., those that are changed the most” (Ex. 1003 ¶ [0023]. “Change” aligns with “update” of a logical file, in contrast to the non-provisional specification of “those blocks that receive frequent writes” which suggests physical wear on a physical block, although neither specification explicitly

This could be accomplished within the device controller **14** described above, which could simply maintain a count of those blocks that are accessed (written to) most frequently, and, on a periodic basis, such as, for example, every 1000 writes, or every 10,000 writes, transfer the contents of those blocks into the SLC NAND flash memory module **28**.

(6:55-61 [emphasis added].) This maintains the first sentence’s ambiguity as to logical or physical blocks and adds further ambiguity as to whether “maintain[ing] a count of those blocks” means counting (somewhere) blocks that are written (equated to “accessed”) *most* frequently or maintaining a write count for each block. (Both can be done using linked lists [¶ 34 above] and controller logic [*e.g.*, Rao ‘916 {Ex. 1004} Fig. 3 {ALU – arithmetic logic unit}].) The first clause of the second sentence is consistent with dynamic writing of “hot data” (frequently written logical blocks) to SLC as in Lee ‘794, Sutardja, and Moshayedi (note 43 above). However, it is also consistent with counting writes of the physical block (which is also done in parallel operations in Sutardja [¶ 47 above] and Moshayedi [¶ 48 above]). The second

calls this feature wear leveling. The non-provisional specification also changed “a count of those blocks that are accessed to the most” to “those blocks that are accessed (written to) most frequently.” However, both Provisional and non-provisional formulations are consistent with benefiting system lifetime by “periodic transfer” of the contents (*i.e.*, of frequently updated logical files (blocks)) to SLC memory.

clause of the second sentence presents at least two additional ambiguities. First, the term, “periodic” suggests a time regularity (but none is provided), which conflicts conceptually with the recited thresholds reached on a random basis, but may mean “occasional,” repeated reaching of thresholds. (The specification’s exemplary threshold of “10,000 writes” [6:59] makes no sense, since the specification assumes MLC endurance of 3K-10K writes [3:48-50].) Second, “transfer the contents of those blocks” may refer to the data *stored* in physical blocks or to the *data associated* with logical blocks *to be* written. Ordinary static wear leveling as reviewed at paragraph 36 above and in the ‘369 patent background (3:29-33, ¶¶ 66-67 above), involves a move of “cold data” to a relatively worn block.⁴⁵ Thus, the second sentence could describe static wear leveling. However, the rationale for moving static or “cold” data in static wear leveling doesn’t apply to moving the data in a physical block that is pushed over a “hot” threshold number of writes. If that data happens to be infrequently updated data, the embodiment would wastefully move infrequently written data to more expensive (and much smaller – such as the 12.5%

⁴⁵ That move would be understood by a POSITA to involve sensing and decoding the data from its previous encoding in a previous block, holding the digital data in a buffer, and then writing into a new block. Terms like “move” and “transfer” could be understood to involve other buffers or holding places for the data (or “content”).

recommendation, 5:16-18) SLC memory.⁴⁶ In my understanding, most physical blocks have similar erase counts because wear leveling has been effectively implemented, and if enough reach a 1,000 threshold, the proposed transfer would overwhelm the SLC module; again, much depends on particular usage patterns. The two-sentence embodiment is unclear in its operation and effect.

VIII. The 2018 Abstraction of the Disclosed System

74. This Section reviews the abstraction in the '369 patent from the pre-2018 patents' claimed memory system of physical, understood MLC and SLC modules "coupled to" a "controller" defined by functions recited in the claims and

⁴⁶ One can't determine from counting erases of the physical block (PBA) to which data (LBA) is currently mapped, whether that LBA mapped data is frequently or infrequently updated, since p-e cycles of a PBA will likely reflect many different LBA mappings. One might reasonably infer that if the erase count of a PBA falls substantially behind that of other blocks, it is currently mapped to an infrequently updated LBA, and its "contents" may be beneficially moved to a more worn block (§ 36 and note 11 above). One cannot reasonably make the converse inference that a frequently erased PBA is currently mapped to a frequently updated LBA, since a frequently updated LBA keeps getting its data moved (and mapped) to different PBAs.

at most minimally supported by the “embodiments” reviewed at paragraphs 69 to 73 above, with preferential mapping to the SLC module suggested in the specification to improve useful NAND flash system lifetime. The abstraction was manifested in claims unanchored in physical or even algorithmic structure apparently to cover the technologies referred to in paragraph 68, without any enablement in the specification for the substitution of those technologies for the SLC module. That abstraction included (1) calling physical NAND flash modules and blocks and their logical addresses aggregate “memory spaces” without specification support (but elevating an ambiguously described RAVM, later distinguished from the also ambiguously described “controller memory”) and (2) requiring “mapping” to “maximize lifetime” not even limited by the DETAILED DESCRIPTION preferences for mapping into SLC modules.

A. Pre-2018 Physical MLC and SLC Modules and Coupled Controller Defined by L2P, Data Integrity Test, and Wear Leveling Functions, with Preferential Transfer to SLC.

75. The first, ‘298 patent, application filed April 25, 2012 (Ex. 1005 cover) was issued on an independent claim that corresponds to the “embodiments” reviewed in Section VII(B):

A system for storing data comprising:

at least one MLC non-volatile memory module comprising a plurality of individually erasable blocks;

at least one SLC non-volatile memory module comprising a plurality of individually erasable blocks; and

a controller coupled to the at least one MLC non-volatile memory module and the at least one SLC non-volatile memory module wherein the controller is adapted to:

- a) maintain an address map of at least one of the MLC and SLC non-volatile memory modules, the address map comprising a list of logical address ranges accessible by a computer system, the list of logical address ranges having a minimum quanta of addresses, wherein each entry in the list of logical address ranges maps to a similar range of physical addresses within either the at least one SLC non-volatile memory module or within the at least one MLC non-volatile memory module;
- b) determine if a range of addresses listed by an entry and mapped to a similar range of physical addresses within the at least one MLC non-volatile memory module, fails a data integrity test, and, in the event of such a failure, the controller remaps the entry to the next available equivalent range of physical addresses within the at least one SLC non-volatile memory module;
- c) determine which of the blocks of the plurality of the blocks in the MLC and SLC non-volatile memory modules are accessed most frequently by maintaining a count of the number of times each one of the blocks is accessed; and

d) allocate those blocks that receive the most frequent writes by transferring the respective contents of those blocks to the at least one SLC non-volatile memory module.

(‘298 7:8-8:9). The first two elements of physical SLC and MLC modules are physically coupled to a third element of a controller defined by four functions that roughly correspond to the DETAILED DESCRIPTION disclosures reviewed at paragraphs 70, 71 and 73 above. Controller function a) roughly corresponds to the FTL/L2P functions admitted in the BACKGROUND (‘298 2:30-3:13). Controller function b) roughly corresponds to the DETAILED DESCRIPTION Figs. 3A and 3B SLC remapping upon failure of the comparison test reviewed at paragraph 71, but with the test applied only to initial writes to MLC. Controller functions c) and d) restate the two sentences of the DETAILED DESCRIPTION shown at paragraph 73 to describe ambiguously one or two forms of “wear leveling.” In prosecution, controller function d), of “allocat[ing]” by “transferring” the “contents” of “those blocks that receive the most frequent writes” proved to be the deciding (inventive) limitation for the Examiner. (Sept. 2, 2014, [‘298] Notice of Allowability at 2 [Ex. 1006 at 80]; June 17, 2014, [‘298] Office Action at 7-8 [Ex. 1006 at 45-46] [finding the base system with functions a) and b) anticipated by Gorobets ‘179 [Ex. 1044] and function c) obvious over Gorobets ‘179 in view of Segal [Ex. 1048].)

76. This core physical memory system of SLC and MLC memory modules (understood to include read/write/erase circuitry) coupled with a controller defined by the recited functions is recited virtually identically in the only independent claim 1 of the next, ‘385 patent ([Ex. 1007], 7:11-8:4, replacing “controller” with “FTL”). “Transfer” function d) was also considered the point of invention. (Sept. 15, 2015, [‘385] Notice of Allowability at 3 [Ex. 1008 at 44].)

77. The same core physical system of SLC and MLC memory modules coupled with a controller with the L2P/FTL functions and “adapted” to perform the data integrity test and wear leveling functions (stated with a different mixing-and-matching of the specification language) was claimed in independent claims 1 and 6 of the third patent, the ‘240 patent ([Ex. 1009] 7:30-8:7 and 8:20-59). The Examiner allowed the ‘240 patent over Gorobets ‘179 and Duann (Ex. 1034) with the further limitation that the “transfer” in a restated wear leveling function d) was performed “after reaching the predetermined count value [of accesses/writes].” (Apr. 16, 2018, [‘240] Notice of Allowability at 2-3 [Ex. 1010 at 110-11].)

78. Thus, the ‘298, ‘385 and ‘240 patents’ claims conformed *functionally* to the Lifetime Patents’ DETAILED DESCRIPTION of *operation* of a *known* hybrid system of physical NAND MLC and SLC modules, with remapping to SLC modules after failure of a post-write read comparison test or reaching a threshold of “frequent writes.” Although those earlier claims explicitly recited mapping into SLC modules,

they were not expressly tied to improving hybrid SLC-MLC NAND system lifetime. The fourth, '300 patent, filed June 12, 2018 (Ex. 1001 cover), nearly seven years after the July 19, 2011, priority date, claimed a vague objective of “maximiz[ing] lifetime,” while untethering the claims from the known physical structure of the system described in the '369 specification and common to all the Lifetime Patents.

B. 2018 Abstraction of “Memory Space” and “Maximizing Lifetime.”

79. **Abstraction of the base terms:** Based on the same 2012 Lifetime Patents DETAILED DESCRIPTION of a known MLC-SLC hybrid system with two (or three) data integrity test and wear leveling operational features (Section VII(B)), the opening independent claim 1 of the 2018 application that resulted in the fourth patent, the '300 patent, read:

A system for storing data comprising:

memory space containing volatile memory space and nonvolatile memory space, wherein the nonvolatile memory space includes both multilevel cell (MLC) space and single level cell (**SLC**) space;

at least one controller to operate memory elements and associated memory space;

at least one MLC nonvolatile memory element;

at least one SLC nonvolatile memory element;

at least one random access volatile memory element;

an FTL flash translation layer, wherein the at least one controller, or FTL, or a combination of both maintain an address table in one or more of the memory elements; and

wherein the address table maps logical and physical addresses adaptable to the system, wherein the mapping is performed as necessitated by the system to maximize lifetime, and wherein the mapping maps blocks, pages, or bytes of data in either volatile or nonvolatile, or both, memories.

(June 12, 2018, [‘300] Application at 14 [Ex. 1012 at 14] [emphasis added]).

80. Untethering from the Relied-Upon Physical MLC-SLC Memory System; Elevating RAVM as “Element,” then Casting Aside: Departing from the first three Lifetime Patents’ reliance on well-known physical MLC and SLC *memory modules* and NAND flash operation they assume (Section VII(B)), the ‘300 patent application took “MLC” and “SLC” as undefined adjectives and applied them to abstract “memory space.” The originally filed ’300 claim 1 claimed an abstract, conceptual “memory space” with undefined “memory elements” generically “mapped” (as known for “secondary memories,” *e.g.*, ‘300 2:30-48 [HDD]) “as necessitated by the system to maximize lifetime.” With “MLC” and “SLC” as proxies for “less enduring” and “more enduring” respectively, the claim was thus to “map” different endurance “memory elements” to “maximize lifetime.” That erasing of the physical boundaries established in the specification had no basis. The word

“space” had been used in the specification only in the context of the market space for hard disk drives, “HDD space” (‘300 2:49), and a quantity of “total non-volatile memory space” (5:20-22; *also* incorporated Rao ‘916 [Ex. 1004] 4:50-51 [“available Storage space”]). In this context, a POSITA would understand “memory space” to be the *storage capacity property* of physical memory. “Element” is only mentioned in the incorporated patents describing Fig. 2 as an “exemplary NAND memory integrated circuit as one element of the NVMS” (Rao ‘916 2:43-45). Given the broad understanding of the word “element,”⁴⁷ a POSITA would not understand whether it refers to a storage cell (perhaps most elemental), a writable word or page or an erasable block (as in the first three patents) or even “bytes” recited in the ‘300 mapping “of blocks, pages, or bytes of data in either volatile or nonvolatile, or both, memories” (8:2-4 [issued claim 1], 8:67-9:2 [issued claim 12]) – and whether that granularity is uniform for the recited MLC, SLC and random access volatile

⁴⁷ *Microsoft Computer Dictionary* [Ex. 1027] at 189: “**1.** Any stand-alone item within a broader context.” Merriam-Webster online: “**f:** a distinct part of a composite device.” <https://merriam-webster.com/dictionary/element>.

memory⁴⁸ (“RAVM”) “elements,” where NAND flash mapping is not directly applicable to RAVM. The abstraction facially allowed a reach to other-than NAND flash non-volatile memory technology, such as PCM, RRAM, MAGRAM operating on fundamentally different physics (‘369 2:19-27 [writing over existing data, erasable at a byte level]). But the abstraction also introduced uncertainty in the claims across other axes of choice: location, continuousness or discreteness in abstract “space,” physical or logical nature, and different physical operations of the different technologies. No operations were identified, much less enabled, for other than NAND flash writing, itself only barely explained (and sometimes mis-assumed) in the specification. Unlike NAND flash “modules,” which a POSITA would

⁴⁸ That elevation of “random-access volatile memory” as an “element” on par with the core SLC and MLC “elements” is contrived. As explained in notes 35 and 38 above, the only mention in the specification of “volatile memory” is the “external” DRAM 20 (Fig. 1) (where host data to be written to flash is first held) and the background “volatile memory for buffering, caching” (‘369 3:61-62). The “memory within the device controller” is assumed to be part of the known and relied-upon SLC-MLC NAND flash system. Its location and association with the device controller and its nature as volatile, non-volatile or random access is not disclosed, much less explained as some inventive feature of the claimed memory system.

understand to include known read/write/erase circuitry, technology-agnostic “element” (which may not itself be qualified to be functioning “memory,” rather than just a component of such memory, such as a NAND flash *cell*) includes no structure to define, much less enable, the claimed “system for storing data.” On an amendment in the ‘300 prosecution, the MLC and SLC elements were qualified respectively as those “that can be mapped into the [MLC/SLC] memory space”; the new RAVM “element” was not so limited, but its status as “element” removed (Oct. 8, 2020, [‘300] Amendment at 2 [Ex. 1012 at 71] [emphasis added]). The newly recited map-ability is strained even for NAND flash, where the logical (host-visible) data address is mapped to a physical address, the latter updated with each updating of data; however, as explained at note 49 below, “memory space” could be considered “logical address space” to which addressable physical memory, through its physical address may be thus “mapped.” But the newly elevated (as a base component) RAVM was taken out of this framework. The “memory space” construct has no practical purpose or effect – it merely serves as verbal camouflage for unsupported technology and terms like RAVM barely noted in the specification elevated in an attempt to distinguish among the ten so-far issued Lifetime Patents, if not from the prior art (which, as shown in Section X below, it does not). As shown in this Section VIII, the “memory space” construct has been made even more

inconsistent through subsequent amendment with additional terms, resulting in more verbal camouflage, but even less structural integrity and practicality.

81. **Extreme Abstraction of “Lifetime” Goal/Effect.** The ‘300 patent application not only abstracted the disclosed physical MLC and SLC modules and blocks to “spaces” and “elements,” it abstracted out the disclosed and previously claimed preferential mapping or remapping to the SLC modules (and particularly abandoned recitation of wear leveling “transfer” of “contents” of “hot blocks” that was the point of invention for the prior patents [¶¶ 75-78 above]), the specification-suggested innovation of the Lifetime Patents. In original claim 1 of the ‘300 application (¶ 79 above), the only limitation on mapping among the recited memory spaces was that mapping is to be “performed as necessitated by the system to maximize lifetime” on “blocks, pages, or bytes of data in either volatile or nonvolatile or both, memories.” There is no mention or availing of any endurance advantage of SLC memory. There is no recitation, much less supporting disclosure of when and how “the system” “necessitates” “maximiz[ation of] lifetime” and of what. There is no indication whether what is modified by this recitation is (1) each mapping, where ordinary mapping (and writing) to MLC modules would itself incrementally lessen lifetime, at least of the physical cell or block written, multiplied in the erase-before-write NAND flash premise (unlike the alternative “direct write” technologies, ‘300 2:21-29, note 32 above); or (2) mapping applied after some event

and thereafter continuously, “periodically,” or not at all. In stark contrast are the known phased strategies to avoid premature reliability failure of NAND flash system components based on statistics (note 20 above). In my view, it is impossible to maximize the lifetime of a particular complex memory system with random stresses without knowing *a priori* its potential lifetime. As reviewed at note 26 above, the lifetime (endurance) improvement of the Lifetime Patents specification, postulated without further explanation to favor superior-endurance SLC modules, would be understood by a POSITA to be an attempt to avoid premature lowering of reliability of a hybrid SLC-MLC NAND system configured for a projected use – such as by using known wear leveling. If the recited “maximization” were construed to be simply some action at some time during the lifetime of a memory system to avoid premature failure, that is done by ordinary, known wear leveling and other strategies (*e.g.*, ‘300 3:13-35; ‘369 3:9-33; ¶¶ 35-36, 58-61 above).

82. In response to the Examiner’s obviousness rejection over Gorobets ‘179 (Ex. 1044) in view of Goodson (Ex. 1070, FTL) (Jan. 24, 2019, [‘300] Office Action at 3-5 [Ex. 1012 at 20-22], which made sense with the abandonment of the wear leveling “transfer” previously found to distinguish Gorobets ‘179), an amendment to claim 1 added a limitation as abstracted steps 114 to 126 of Fig. 3B (‘300 7:57-60, 8:5-12 [as issued]) similar to the ‘298 [Ex. 1005] 7:24-33 “data integrity test” controller function b) found anticipated by Gorobets ‘179, ¶ 75 above)

– but with remapping of “address space” only to a “different physical range of addresses from those determined to have failed the data integrity test to achieve enhanced endurance” (June 26, 2019 Amendment at 2 [Ex. 1012 at 30] [emphasis added]) – not remapping to the SLC module that provided physicality and a superior endurance rationale in the specification and the ‘298 patent. The amendment introduced the ambiguity of whether the maximizing lifetime mapping limitation applied to all mappings differently from the new remapping for failure of a data integrity test, in addition to whether “to achieve enhanced endurance” limits the remapping or the data integrity test. Those ambiguities remain in the issued ‘300 patent claims 1 and 12. (‘300 8:5-11, 9:3-9.) The goal “to achieve enhanced endurance” was removed from the failed data integrity test remapping of the Challenged Claims (‘369 8:29-37), but the ambiguity remains whether “maximize lifetime” applies as general wear leveling (not explicitly claimed) or to the failed test remapping, both or neither.

83. Data Integrity Test Compares “Stored” and “Retained” Data:

After the amendment to add the data integrity test, which included then claim 19 (without recitation of FTL, issued claim 12), the Examiner maintained an obviousness rejection over Gorobets ‘179 in view of Oribe (Ex. 1057). (Dec. 13, 2019, [‘300] Office Action at 4-8, 10-12 [Ex. 1012 at 40-44, 46-48]. An amendment was made to add “transfer of data” to the results of failure of the data integrity check. (Mar. 23,

2020, [‘300] Amendment at 2, 4-5 [Ex. 1012 at 56, 58-59].) After the Examiner continued to maintain the rejection, the data integrity test was amended to recite it as controller action to both “store data in the MLC nonvolatile memory element” and “retain such stored data in the [RAVM]” then “comparing the stored data to the retained data in the [RAVM].” (Oct. 8, 2020, [‘300] Amendment at 2, 5 [Ex. 1012 at 71, 74].) However, reference to RAVM as an “element” was expressly removed, while the MLC and SLC “elements” were qualified, “that can be mapped into the [MLC/SLC or nonvolatile] memory space”⁴⁹ (*id.* at 2, 4 [Ex. 1012 at 71, 73]). The following argument was made:

. . . This operation is not some operation that can be carried out by a conventional cache, as that cache is typically only designed to handle

⁴⁹ This amendment recast “memory space” (with “memory” ordinarily associated with physical memory, *e.g.*, note 6 above) as “logical address space” (ordinarily associated with “data”) by being mapped to the clearly physical MLC and SLC “memory elements” (parts of the disclosed physical MLC-SLC NAND flash system), and where “mapping” is recited in the original claim 1 (reproduced at ¶ 79 above and carried through to the issued claim 1 [7:66-67] and 12 [8:64-65] as between logical and physical addresses, where previously the claim only recited a controller “to operate memory elements and associated memory space” (carried over

one block of data. In the nonvolatile memory, in order to facilitate a compare operation, if an entire block is being written to with new data, that entire block has to be retained in one portion of the memory and then written to the memory, followed by a subsequent Read of the memory, requiring that block of data to be placed in an additional area of the nonvolatile [sic] memory, thus requiring at least two blocks of memory space in the nonvolatile [sic] memory. . . .

(*Id.* at 9 [Ex. 1012 at 78] [emphasis added].) This confirmed that the disclosed data integrity check placed “retained data” in the controller memory where it was to be

into the issued claims, 7:43-42 and 8:44-45) (emphases added). However, the newly recited, physical RAVM, stripped of its “element” nature, was no longer even “associated” by the controller with “volatile memory space.” The same amendment rewrote previously presented dependent claim 25 (*id.* at 5-6 [Ex. 1012 at 74-75]) into issued claim 10 “wherein the contents of frequently accessed portions of the memory space are periodically moved from the MLC space to the SLC space” (‘300 8:34-36 [emphasis added]). This adds new terms to the logical-physical “block”, “content” and “move”/“transfer” ambiguities of the DETAILED DESCRIPTION of a wear leveling embodiment (6:53-65, ¶ 73 above). However, at least for MLC and SLC memory “elements,” which are mapped to MLC and SLC memory “space,” the amended claims clearly indicate a physical/logical distinction between “element” and “space” in the Challenged Claims instead of a pointless identity.

combined with valid flash data (note 39 above) and then “stored data” was read from the written flash location back into the same “controller memory.” Again, the DETAILED DESCRIPTION of the data integrity test failed to clearly identify locations and failed to explain the processes (§ 71 above). The Examiner specifically cited the two amendment pages including the above quotation as his reason for allowance. (Jan. 27, 2021, [‘300] Notice of Allowability at 2 [Ex. 1012 at 86].) The amendment argument, as the Lifetime Patents generally, hides behind the ambiguity as to whether reference to a “block” is to a flash block of 64 pages versus the ordinary writing of one page at a time (in view of the page buffer for flash memory) or any other “block” referenced in the specification. The specification description of the data integrity test did not disclose details (§ 71 and notes 35 & 38 above), much less reading after writing of an entire block, particularly multiple pages in MLC.

C. Further Confusion of the Claimed System in the ‘369 Patent.

84. Inconsistent Restoration in ‘369 of “Memory Module” and RAVM as a “Memory Element”: In the ‘546 patent, followed by the ‘369 patent, the terms “MLC memory module” and “SLC memory module” returned as part of one “bank of nonvolatile memory” (‘546 7:37-41, ‘369 7:41-52). This departed from the DETAILED DESCRIPTION primary embodiment of two separate banks, one with only MLC modules and another one with only SLC modules (‘369 5:6-10) and also from an alternative embodiment (‘369 Fig. 4, 6:64-7:5) showing two banks, each

including 8+ MLC modules and 2+ SLC modules.⁵⁰ The reintroduction of “module” was not physically anchored to the POSITA-understood “plurality of [NAND flash] individually erasable blocks” (‘298 7:9-12) but qualified only as each including “at least one MLC nonvolatile element” (‘546 7:38-39, ‘369 7:43-44 [qualified as “physical”]) and “at least one physical SLC nonvolatile element” (‘546 7:40-41, ‘369 7:48-49 [same]). The ‘546 patent also recited, for the first time in the chain of continuing applications, and the ‘369 patent retained, “controller memory”: “the controller having associated controller memory for storing received data” (‘546: 7:47-48, ‘369 7:60-61) corresponding to data integrity test step 102 (‘369 6:16-18), “transferring the stored received data from the controller memory” to MLC (‘546 8:1-5, ‘369 8:5-8) corresponding to step 110 (‘370 6:37-39 [of combined data]), and, with ambiguous references, “reading the stored data [in MLC] to the controller memory and comparing the stored data in the controller memory in the [MLC] to the

⁵⁰ A POSITA would understand that the memory module depicted in Fig. 6a of the incorporated Rao ‘916 (Ex. 1004) was an SLC NAND flash chip of 1,024 blocks each of 64 pages (each of 2K bytes plus 64 bytes for spares or system) or 1056 Mbits. (See note 30 above.) The number each of MLC and SLC modules in ‘369 Fig. 4 is consistent with the DETAILED DESCRIPTION recommendation that SLC memory would be 12.5-25% of MLC capacity (5:36-38).

retained data that was associated with the stored data in the [RAVM] by the controller” (‘546 8:15-21 [emphasis added], ‘369 8:19-24), although, as explained paragraph 71 above, the DETAILED DESCRIPTION provided no clarity in its steps and locations of data, much less clear algorithms for those steps. in neither patent is “controller memory” recited to be part of the “memory space” like NAND flash memory and the recited RAVM. This adds inconsistency. The ‘369 memory space is further divided into “logical and physical volatile memory space” (7:34-35), which pointlessly and confusingly conflicts with a reasonable construction in the ‘300 patent that the MLC and SLC nonvolatile elements mapped to the logical address space that would be the “memory space” (note 49 above). And the RAVM is restored to a “physical element” which paradoxically “exists in the logical volatile memory space” (7:53-55) – but no logical or physical address is disclosed in the DETAILED DESCRIPTION for DRAM 20 or any other RAVM. Although by separate mention in the claims, “controller memory is suggested to be distinct from RAVM, with neither detailed description or claim limitation of each, one of the “at least one [RAVM] that exists” (7:53-55) may be the ordinary RAM in or associated with NAND flash controllers (and thus, controller memory) (*e.g.*, ¶ 31 above, ‘369 3:61-62 [all memory systems, “volatile memory for buffering, caching, etc]). Claim 1 recites that “retained data” is in the RAVM (8:11-13 and 8:21-23), and if the RAVM is external DRAM 20, from which data to be written is combined with valid

flash data, the data written (“transferred” in the claim language) to flash is not in DRAM 20 (note 39 above) unless written back from controller memory back to DRAM 20. Although ‘369 claim 1 provides that the controller is “operable” to “retain such received data” in RAVM (8:9-13), there is no disclosure of an operation to write from DRAM to “controller memory” to flash (as “stored data”) AND DRAM (as retained data) and then both back to “controller memory” for comparison, and then from DRAM to a different physical location if the comparison fails. Such a limitation would be enabled only if admitted for ordinary NAND flash operation. Otherwise, there is the inference from the DETAILED DESCRIPTION of Figs. 3A and 3B that the retained data remained in the controller memory (§ 71 and note 39 above). If “controller memory” were considered one of the “at least one” RAVMs in memory space, that could reconcile the inconsistency that “controller memory” is not explicitly included in the “memory space” construct. The ‘369 patent’s recitation of new categories of “logical and physical” memory spaces that somehow include RAVM but not “controller memory” is a reshuffling of words, including ones absent from the 2012 common specification, that does not better

specify any innovation but is merely camouflage for the same and only the 2012 proposed data integrity testing feature for a known MLC-SLC flash system.

D. The Challenged Claims.

85. The '369 patent has one independent claim 1 (7:34-8:36) that recites a system for storing data, comprising (1) an abstract "memory space", (2) a bank of (2a) a plurality of physical MLC memory modules, each including at least one MLC "memory element" and (2b) at least one SLC nonvolatile "memory element," (3) at least one RAVM element – possibly including "controller memory," (4) at least one controller "to operate" the physical memory elements, having "controller memory" for storing received data, (5) FTL with the controller maintaining an address table, (6) mapping to "maximize lifetime," and (7) performs a data integrity check by comparing "stored data" (in the controller memory) to "retained data" (in the RAVM) and remaps and "transfers" "data corresponding to the stored data" to a different location. Remaining claims 2-10 all depend from claim 1. Claim 2 calls for the FTL to be a software or firmware module. Claim 3 calls for at least one of the memory elements to be "embedded" in a controller. Claim 4 calls for the MLC and SLC elements to be flash memory. Claims 5 and 6 respectively call for the RAVM to be DRAM and SRAM. Claim 7 calls for the SLC element to include an HDD. Claim 8 calls for the controller to be associated with a processor external to the

memory system. Claim 9 calls for the remapping to SLC. Claim 10 calls for the FTL to be associated with the MLC and SLC modules.

E. Claim Construction.

86. As reviewed in Section VII(B), the ‘369 patent DETAILED DESCRIPTION leaves substantial ambiguities. No new hardware structure or software algorithm or structure is disclosed or claimed, only well-known (before July 19, 2011) MLC and SLC NAND flash memory modules (that include known differences in structure, endurance and operation, including mitigating wear on their common floating gate cells programmed and read with different circuitry and algorithms) and a controller to perform ordinary and necessary NAND flash memory operations like reading, writing and erasing, including L2P/FTL addressing and mapping (using internal or external random access volatile memory). Although the pre-2018 Lifetime Patents claimed that known system of MLC and SLC modules with optional “data integrity test” and “wear leveling allocation” features (Section VIII(A) above), the ‘369 patent conjures from those modules technologically purposeless abstract memory spaces having no applicability to the ordinary RAVM and controller memory elevated post-2018.

87. **MLC and SLC memory/element/space:** Dr. Rao proposed in the incorporated Rao ‘916 a new use of the existing Samsung SLC NAND memory module (Ex. 1004 Fig. 6, *e.g.*, 1:60 [“modify the already existing ‘buffers’”]); his

IDS submissions with the '298 patent application distinguished between MLC memory, SLC memory and portions of MLC memory used in 1-bit mode (*e.g.*, Cho Dual-Mode [Ex. 1068], Takeuchi [Ex. 1069]). As I explained in paragraphs 40-45 above, even though “original” EEPROM (called SLC after 2001) and MLC share the same kind of floating gate cell, that cell or group of cells, even a flash “block,” do not constitute a “memory,” which requires at least circuitry allowing retrieval/decoding of information stored. That retrieval and decoding of the two levels of logical pages programmed into an MLC memory row of cells (a physical page) requires different circuitry for processing than for the single level of pages programmed into an SLC memory row of the same type of cells. (*Id.*) MLC can be operated to store and retrieve only one logical page per physical page, but SLC cannot be operated to store and retrieve more than one logical page in one physical page.⁵¹ (*Id.*) With the claim terms’ recitation of “memory” and “module,” terms

⁵¹ The '369 specification explains that in “single-level storage (SLC), erased bits . . . can only be changed [programmed] to the opposite state [programmed] once before the entire block must be erased.” (2:59-64.) Thus, a row (page) of SLC cells cannot be programmed for a second, upper logical page as can a page of MLC cells. This is at least because SLC memory lacks a second page buffer, the associated circuitry

that each have meaning, a POSITA would have the ordinary and customary understanding (1) of the specifically recited “SLC memory” and “MLC memory” as distinct types of NAND flash memories (with circuitry that make them memories, ¶¶ 29 and 41-43 above) and (2) of “SLC memory modules” and “MLC memory modules” as self-contained, interchangeable manifestation of those memories (¶ 30 above). Reflecting my opinion of a POSITA’s understanding, compact, but accurate constructions (proposed by Phison in the court litigation) are:

- **“MLC memory modules”** means “modules comprising MLC non-volatile memory” *where*
“MLC non-volatile memory” means “non-volatile memory cells arranged with circuitry capable of storing multiple logical pages in a single physical page of cells” and
- **“SLC memory module”** means “module comprising SLC non-volatile memory” *where*
“SLC non-volatile memory” means “non-volatile memory cells arranged with circuitry incapable of storing multiple logical pages in a single physical page of cells”

I also understand that Phison has taken a position in the court litigation that the terms **“memory element”** and **“memory space”** are indefinite across their usage in the

needed to operate the two buffers, as well as the circuitry to enable the encoding of two bits of information in a single cell.

Lifetime Patents,” at least for the reasons I explained in paragraph 80 above. “MLC memory element” and “SLC memory element” should be construed as NAND-mappable/addressable physical components (such as a flash page or block) that are those distinct memory types and not just one-bit (1 programming level, 2 threshold voltage levels) versus n -bit (n programming level, 2^n threshold voltage levels) storage cells that are not themselves memories and addressable; and “memory space” should be construed as logical address space, subdivided by mapping to physical MLC and SLC memory.

88. **Maximize Lifetime:** If the term is not found unenabled or indefinite, “maximize **lifetime**” should be construed in the general understanding of the POSITA as an act to avoid premature failure of a NAND flash component (§ 81 and note 26 above).

89. **Controller/FTL:** I understand that Phison has taken the position in the litigation the terms “controller” and “FTL” (across multiple Lifetime Patents) are indefinite for lack of structure (algorithms) disclosed for recited controller functions. Those arguments apply to the Challenged Claims.

90. **Data Integrity Test/retained data/stored data:** I understand that Phison has taken the position in the court litigation that this term (across multiple Lifetime Patents) means “a test that compares stored data to retained data.” At that level, this not an issue for the ‘369 patent, however, the particular locations of the

relevant “stored data” and “retained data” and how they got there were not disclosed, particularly any rewriting of “retained data” from “controller memory” to DRAM (paragraph 84 and notes 38 & 39 above) and “failure” of the “data integrity test” is only disclosed when “the retained data does not match the newly stored data” (6:37, ¶ 71 and note 40 above [bit-by-bit comparison understood]). Thus, any steps and apparatus to perform the otherwise unexplained functions recited must be those of common NAND flash systems and operations prior to July 19, 2011.

IX. Lack of Enablement and Definiteness of the ‘369 Claims

91. **Element [1a] (“memory space”) is abstract, unsupported and indefinite.** The unsupported “memory space” construct [1a] (7:34-40), carried over from the ‘300 patent, is abstract, unsupported and indefinite as explained at paragraph 80 above. As reviewed at paragraph 80 above, the term was used in the specification to refer to storage capacity of a memory system; in the claim it appeared in the ‘300 patent to describe abstract logical address space, and in the ‘369 patent to describe some combination of abstract logical space and physical things. In any of these cases, the recited “memory space” and subdivisions are “disembodied concepts,” not allowing a POSITA to understand what part of physical memory is associated to what part of the memory space. Importantly, “memory space” adds nothing to the claimed structure or operation of the system other than possible extension to other than the base NAND flash technology. Nothing in claim element

[1a] provides any practical application, only a conceptual inclusion of other technology that can be called “SLC.” The recited categorization of memory space is no more than identification of physical memory as MLC nonvolatile or SLC nonvolatile or RAVM – and not even categorizing “controller memory.” Even more than other abstract ideas that I understand are ineligible for patenting, the recited taxonomy of five types of spaces can be performed by the human mind without even pad and pencil. As explained at paragraph 84 above, a rationalization for the ‘300 patent as “memory space” being logical address space mapped to physical MLC and SLC memory (note 49 supra) is challenged by the recital that the “memory space” includes both “logical and physical memory space.” Not only does the “memory space” construct of claim element [1a] have no practical import, it interferes with understanding of other claim elements. The abstraction and logical-physical ambivalence of “memory space” and “memory element” confuses the physical-versus-logical boundaries in the other claim elements, for example, detracting from the specificity required of the functionality of the controller recited in element [1b]. As explained at paragraphs 68, 69 and 75 above, where recitation of NAND flash modules in the first three patents provided the POSITA with some understanding how the recited components of the claimed systems worked together for preferential mapping into an SLC module, the technology-agnostic untethering of “memory space” (in the specification it’s a *capacity*) and “memory elements” leaves gaps in

how data is read, written or erased and thus it is not clear how the recited claim components could work together as a useful system. The additional division of RAVM (§ 84 above) – the element that has been elevated to some kind of innovation from presumed internal or external “controller memory” -- into both physical and logical space makes no sense; nowhere in the DETAILED DESCRIPTION is a logical or physical address suggested for RAVM. The further introduction of “controller memory” in the claims as distinct from RAVM leaves unexplained what “controller memory” is in this “memory space.”

92. **Element [1b] (“modules” of “memory elements”) is unsupported indefinite, and inoperative.** The unsupported “memory element” construct [1b] (7:41-52), as reviewed at paragraph 80 above, is both unsupported and indefinite because it doesn’t specify relevant elements for the operation of the claim. As explained at paragraph 84, the recitation of the elements as merely one element required for each of the MLC and SLC modules makes the claim inoperative with respect to the recited goal of the operation of the claimed system, which is to “maximize lifetime” by mapping and remapping on failure of the data integrity test.

93. **Element [1c] (RAVM) is unsupported.** (7:53-55) As explained at notes 35 & 38 above, “random access volatile memory” is not mentioned in the specification except for mention of DRAM 20 and “volatile memory for buffering, caching” – and the ambiguously referenced “controller memory” is not explained,

certainly not as an advance in the art. The failure to specify leaves the use of the RAVM unenabled and indefinite – for example the ‘300 prosecution statement that “conventional cache” is not adequate (§ 83 above). Moreover, as explained at note 39 and paragraph 84 above, there is no support in the specification (or antecedents for claim elements [1e] and [1f]) for RAVM being the location of “retained data” other than that the controller is “operable” to make it such – and no such operation is disclosed.

94. **Element [1d] (controller “to operate” and having associated memory) is unsupported and indefinite.** (7:56-62) The structure of the controller is never disclosed other than as a possibility that it may be implemented in a chip set. (§ 70 above.) Only function is recited, none in any detail appropriate for the coordination of distinct memory types – MLC and SLC, much less the alternative technologies. For example, a “Read access operation” is added to the claim (7:59) with no mention in the DETAILED SPECIFICATION, much less any explanation that the read operation for MLC is substantially different from that for SLC. I understand that when a general purpose computer or processor is claimed in terms of functions, the specification must disclose sufficient structure corresponding to the functions performed by that element; for general purpose computers, that would be special programming, that is, an algorithm. The functions recited in claim 1 for the controller include operating memory elements and associated memory space [1d1]

(7:56-60), controlling access of a physical memory element for storage [1f1] (8:1-4), transferring stored received data [1f2] (8:4-8), operable to store and retain data [1f3] (8:9-13), performing a data integrity test [1g1] (8:14-17), reading the stored data to controller memory [1g2] (8:17-18), comparing the stored data [1g3] (8:18-23); whereupon failure of the data integrity test, a result, presumably attributable to the controller, are the functions of remapping [1i2] (8:30-32) and transferring [1i3] (8:32-36) retained data to a different physical address. The controller, in conjunction with FTL also maintains an address table [1e] (7:63-66) and performs mapping [1h] (8:24-27). As reviewed in paragraphs 70 and 71 above, these functions are described in no more detail than recited in the claims; there is no special programming disclosed. I also understand that where method steps such as “operate”, “storing”, “controlling”, and “performing” are mixed with an apparatus claim, the claim is indefinite. Much of ‘369 claim 1, as in the Lifetime Patents, particularly since the abandonment of the wear leveling claims, is directed to Figs. 3A and 3B “illustrating a method for utilizing a NAND flash memory system” (6:5-6) and thus clearly claim method steps. Further, the recitation of “having a controller memory associated” with the controller (7:60-61) provides no structure for the controller. Moreover, it does not appear that “controller memory” is part of the “memory space” and whether it is volatile, nonvolatile, physical or addressable as other memory.

95. **Element [1e] (“a flash translation layer”) is abstract as purely software and indefinite and unsupported for other than NAND flash operation.**

I understand that software disembodied from physical implementation may be ineligible subject matter. FTL is software and in its address mapping function of claim element [1e], can be performed by the human mind with pencil and paper albeit at a different scale. Claim element [1e] requires that the “FTL... maintain an address table in one or more of the physical memory elements”, but this amounts to nothing more than specifying the type of “pen and paper” that must be used for record keeping and does not preclude the FTL from being performed as a mental process.) Moreover, the recitation of FTL does not limit it to being a non-transitory signal. The limitations of the claim element are “well-understood, routine, and/or conventional” at least for NAND flash operation. The claim element is both unsupported and indefinite for technologies other than NAND flash (which has its distinguishing erase-before-write requirement). (See note 3 above.) Further, the functions recited for FTL, “maintaining an address table” that maps “as necessitated by the system to maximize lifetime,” are not supported by disclosure of special programming or algorithms, and thus the claim element is indefinite.

96. **Element [1f] (controller “controlling access” and “transferring”) is unsupported.** (7:63-66) As explained at paragraph 94 above for element [1d], the structure of the controller is never disclosed nor the NAND flash operations that

differ for MLC and SLC and for RAVM, the recited functions are not supported by structure, and impermissibly recite method steps. Further, it is neither stated in the specification nor in the claim how “retained data” finds its way into the RAVM, if it is different from “controller memory” – as it is in the specification and the claim.

97. **Element [1g] (controller “performing data integrity test . . . after . . . a Write access” and “reading stored data”, “comparing”) is unsupported.**

(8:1-13) Again, as explained in paragraph 94 above, the structure of the controller is never disclosed nor the NAND flash functions differing for MLC and SLC, and the recited functions are not supported by structure disclosed in the specification. Here, “reading into” is a process never explained. . Element [1g] has a fundamental contradiction in “comparing the stored data in the controller memory in the given one of the physical MLC memory elements” (8:18-20 [emphasis added]): there is no antecedent that the controller memory is in the MLC memory element. The recited **timing** of the data integrity test “**after**” a **write** access is problematic. The DETAILED DESCRIPTION is ambiguous, stating in one place that a data integrity test in the disclosed system is performed “after every write” (5:10-11) and in another that it is performed after “each write to an address within a particular address range . . . as time permits” (5:57-59). The latter formulation is indefinite because it doesn’t specify what particular logical or physical address provides the condition. A POSITA, manufacturer or user is faced with the question whether the claimed,

resource-heavy bit-by-bit matching (§ 57 above) of “stored data” with “retained data” (and transferring into SLC upon failure (unlike common ECC schemes tolerating some error in the stored data, *e.g.*, note 20 above) are required (for infringement) with “every write” or “as time permits” and for the continuous lifetime of the system or otherwise. As I explained at paragraph 57, the claimed comparison is impractical, if not inoperative.

98. **Element [1h] (address table . . . mapping as necessitated . . . to maximize lifetime) is unsupported and indefinite.** (8:14-24) Both the syntax and the content of this limitation are indefinite. As reviewed in paragraph 81 for the same limitation in ‘300 (with some additional explanation that mapping could be of different “elements”), “maximizing” is not mentioned, much less explained in the common specification, and there is indefiniteness in multiple dimensions, including what is necessitated and what it means to maximize when every writing in NAND flash (and necessary erases) decreases lifetime. (Also, the syntax suggests that it is the address table that performs the mapping, even though the element [1f] FTL/controller “maintains the table.”)

99. **Element [1i] (“failure of the data integrity test”) is unsupported and indefinite.** (8:28-36) To the extent that “failure” is defined by something other than the “match” disclosed in the specification (6:34-36), this term is unsupported and indefinite.

100. **Claim 3**, “wherein at least one of the random access physical volatile memory or the MLC and SLC nonvolatile memory elements are embedded in at least one controller” (8:41-44) is unsupported insofar as there is no disclosure in the specification of any of these in the device controller (or of any physical content of the controller).

101. **Claim 7**, “wherein the at least one SLC physical nonvolatile memory element includes a hard disk drive (HDD)” (8:57-59) is on its face nonsensical as that “physical element” is recited to be a subcomponent of an SLC memory module (claim 1, 7:48-49). APOSITA would understand that it is physically impossible to install or somehow include a hard disk drive (HDD) on an SLC chip (module) or some smaller “element” since the HDD is many orders of magnitude larger than a chip. Thus, claim 7 is independently indefinite and unsupported.

102. **Claim 10**, “wherein the FTL is associated with the MLC memory modules and the SLC memory modules in the bank of physical nonvolatile memory” (8:63-65); the abstract “association,” is disembodied just as the FTL recited in claim element [1e] (§ 95 above), and was unsupported and indefinite in that association other than appearance as a functional block 54 in Fig. 4. Claim 10 is thus abstract, unsupported and indefinite.

X. Patentability of the ‘369 Claims Relative to the Prior Art

103. To the extent understandable, independent claim 1 of the ‘369 patent claims a memory system that would describe known hybrid MLC-SLC flash memory systems as described in the DETAILED DESCRIPTION (Section VII(B)) with known, necessary L2P mapping operations and at some time performs a “data integrity test” by comparing data written to flash with the data to be written to flash, with a general operation to avoid premature failure. Multiple alternative “data integrity checks” and approaches to avoid premature unreliability of NAND flash systems, including SLC-MLC hybrids, have been detailed in multiple prior art textbooks, articles and patents, as reviewed in Section V(C). Gavens and its incorporated references disclose or render obvious the Challenged Claims in view of the knowledge of the POSITA.

A. Claim 1 Is Obvious Over Gavens, Including Incorporated References, in View of Knowledge of the POSITA.

104. Gavens (Ex. 1045), among the SanDisk portfolio and incorporating by reference others of that portfolio,⁵² is a published reference that discloses a complete

⁵² See, e.g., U.S. Patent No. 6,456,528 to Chen (“Chen ‘528”), Ex. 1040, incorporated by reference at Gavens 16:29-40 (binary mode operation); U.S. Patent Appl. Pub. No. 2011/0153912 by Gorobets et al. (“Gorobets ‘912”), Ex. 1049,

multi-modal NAND flash system with the ordinary L2P/FTL function of its controller, a variety of post-write read error detection and management schemes, and the capability of and compatibility for employing other known tools (disclosed in incorporated documents and exemplified in paragraphs 58-61 above) used to avoid premature NAND system failure.

105. Since the 2001 (*e.g.*, Cho, Ex. 1039) emergence of the terms “multi-level program cell” (MLC) memory and (thereby distinguished) “single level

incorporated by reference at Gavens 20:53-59 (post-write-read error management with binary mode portion); U.S. Patent No. 5,930,167 to Lee et al. (“Lee ‘167”), Ex. 1050, incorporated by reference at Gavens 17:57-67 (MLC with binary mode cache); U.S. Patent Appl. Pub. No. 2010/0172180 by Paley et al. (“Paley ‘180”), Ex. 1051, incorporated by reference at Gavens 8:41-42 (block management system). In turn, both Gorobets ‘912 (¶¶ [0020], [0024]) and Paley ‘180 (¶¶ [0023], [0027]) incorporate by reference U.S. Patent No. 5,936,971 to Harari et al. (“Harari”), Ex. 1057, for using RAM in write cache for flash, and PCT Pub. No. WO 03/027828 by Gorobets et al. (“Gorobets PCT”), Ex. 1058, for buffering mapping information. Hereinafter, it should be understood that reference to a specific portion of any of the above references is understood to be disclosed by Gavens due to incorporation by reference, as if the incorporated document was explicitly contained in Gavens.

program cell” (SLC) memory, there have been implementations of hybrid MLC memory modules and SLC memory modules availing of the endurance advantages of SLC. (*E.g.*, Lee ’794 [Ex. 1041], Sutardja [Ex. 1042], and Moshayedi [Ex. 1043].) Many NVM implementations disclosed in the SanDisk patent portfolio involve the partitioning and configuration of some MLC (or TLC) blocks for one-level (two states) or pseudo-SLC (pSLC) operation. As reviewed at paragraph 49 above and in Gavens itself (*e.g.*, 12:6-10) the pSLC mode of operation enjoys superior endurance compared to MLC in multi-level mode – though not as superior or as fast as “real” or “native” SLC memory which uses different circuitry. It would be obvious to a POSITA to have one MLC module operating entirely in pSLC mode (and thus be considered a pSLC memory module, which, as mentioned in paragraph 49, are marketed as such) with an expectation of success for the relative superiority to MLC in multi-level mode⁵³ or use a “real” or “native” SLC module as understood by a POSITA from the embodiments actually disclosed in the specification. Like a memory cell, a block of memory cells is not memory – it does not have the necessary circuitry to program and read the threshold voltages that represent the logical data.

⁵³ This would meet the ‘300 patent specification statement that an “SLC module” may be “any other solid-state non-volatile memory as long as it has endurance that is superior to that of MLC flash” and faster (4:36-41).

106. I show here that Gavens and its incorporated references, in view of the knowledge of the POSITA, disclose the limitations recited in claim 1 as set forth in the groupings as printed in the ‘369 patent:

1. [1Pre] A system for storing data comprising:

107. I understand the question of whether a preamble is limiting is a question of claim construction. I do not provide an opinion as to whether the preamble of claim 1 is limiting.

108. To the extent the preamble is limiting, Gavens discloses “a system for storing data,” namely, the “memory device 90” that includes memory chip 100 with a memory array 200 and other circuitry, to which the host 80 “sends data to be stored” or from which it “retrieves data by reading” (Gavens Fig. 1 (90), 8:13-17).

2. [1a][1a1] memory space containing logical and physical volatile memory space and nonvolatile memory space, [1a2] wherein the logical and physical nonvolatile memory space includes both multi-level cell (MLC) logical and physical memory space and single level cell (SLC) logical and physical memory space;

109. As explained at paragraphs 80 and 91 above, the entire ‘369 patent claims’ construct of “memory spaces” and “memory elements” has no basis in the ‘369 patent specification and is abstract and ambiguous. To the extent understood, that they refer to volatile and nonvolatile components that have logical and physical addresses, Gavens satisfies claim element [1a] by disclosing MLC and SLC memory

components that have such addresses and RAVM (assumed to have such addresses) as explained in Sections X(A)(3)-(4) below.

110. **[1a1]** The recited “memory space” is disclosed by the existence (and addressability) the memory components shown in Sections X(A)(3)-(4), where “volatile memory space” is satisfied by the RAVM of Section X(A)(4) and the “nonvolatile memory space” is satisfied by the MLC and SLC memories of Section X(A)(3).

111. **[1a2]** The recited “multilevel cell (MLC) logical and physical memory space” is disclosed by the existence (and addressability) of the “MLC nonvolatile memory element” [1b1] of Section X(A)(2), and the recited “single level cell (SLC) logical and physical memory space” is disclosed by the existence (and addressability) of the “SLC nonvolatile memory element” [1b2] of Section X(A)(2).

112. Thus, Gavens and its incorporated references, in view of the admitted prior art and the knowledge of the POSITA, disclose or make obvious element **[1a]**.

3. [1b] a bank of physical nonvolatile memory, including:

[1b1] a plurality of physical MLC memory modules, each including at least one physical MLC nonvolatile memory element having a physical address within the physical MLC nonvolatile memory space that can be mapped to a logical address in the MLC logical nonvolatile memory space, and

[1b2] at least one SLC memory module including at least one physical SLC nonvolatile memory element having a physical address within the physical SLC nonvolatile memory space that can be mapped into the SLC logical nonvolatile memory space;

113. **[1b]** Gavens discloses a bank of non-volatile memory chips including MLC (NAND flash) memory. (Fig. 1; 8:13-21). **[1b1]** The base memory chip of Gavens meets the characteristics of the POSITA-understood MLC flash memory module of cells, circuitry and programming explained in paragraphs 26-30, 42 and 43 above. The array of memory cells 200 is partitioned into a first portion 410 configured for lower density (*e.g.*, pseudo-SLC) operation and a second portion 420 configured for high density (*e.g.*, native MLC, having each cell storing multiple bits) operation. (Fig. 14A, 16:15-22.) It particularly meets Phison's multiple level page programming requirement (distinguishing from SLC) set forth in paragraph 86 above, as programming of those levels of logical pages is disclosed in Fig. 23 (25:18-26:30). The Gavens controller maps logical "memory spaces" (addresses) to the physical memory (addresses) (8:36-39).

114. **[1b2]** Gavens does not disclose the use of a "native" SLC flash memory module or chip as described at paragraphs 26-30 above because the Gavens memory

module or chip 100 is capable, with its circuitry, of programming more than one logical page into a physical page, as explained in paragraph 86 above. However, Gavens discloses partitioning, configuration and use of portions of its memory array 200 in 1-bit mode, the “pseudo-SLC” mode explained at paragraph 49 above. Specifically, Gavens discloses, as a base system for multiple operational embodiments, “[a] first portion has each memory cell storing one bit of data and [a] second portion has each memory cell storing more than one bit of data” (4:32-34, 12:6-8 [calling memory operating in 1-bit mode “binary” or “SLC” memory]). Regarding one aspect of the difference, it is explained that the first portion, being “configured as lower density storage[,] . . . operates with a wider margin of error than that of the second portion” (18:16-20, also 12:6-10, 21:6-9). As I explained in paragraph 105 above, it would be obvious to a POSITA to substitute a “native” SLC flash memory module for the Gavens first portion, with a similar, if not wider margin of error than binary mode operation of MLC (*see* ¶ 49 above) and achieve similar, if not better results with the same NAND flash controller Fig. 1(102) with great expectation of success. Again, the Gavens controller maps logical “memory spaces” to the physical memory (8:36-39) and would be capable of writing to different NAND flash modules, as would be presumed in its memory array. As it is the memory module that performs the physical reading and writing based on controller commands and addresses, the NAND flash controller would be capable of writing

to both MLC and “native” SLC (*e.g.*, Moshayedi [Ex. 1043] Figs. 1, 2, ¶¶ [0036]-[0038] [controller 102 controlling virtual-to-physical mapping with RAM 214 for four channels of commercial SLC chips 112 and MLC chips 114]).

115. Thus, Gavens, in view of the POSITA’s knowledge of NAND flash, renders obvious claim element **[1b]**.

4. [1c] at least one random access volatile physical memory element that exists in the logical volatile memory space;

116. It is well-known to use RAVM (also known as “RAM”) in a NAND flash controller as a staging or cache location for data to be written to the flash memory. (*E.g.*, Rao ‘916 [Ex. 1004] Fig. 3 [“Data Reg.”].) Gavens incorporated by reference Gorobets ‘912 (Ex. 1049) and Paley ‘180 (Ex. 1051), the latter expressly for block management (note 53 above). Both share Fig. 1 showing a controller 100 with RAM 130, shown in Gorobets ‘912 Fig. 11 and Paley ‘180 Fig. 15 as “Cache (RAM) 102” within the controller. Gorobets ‘912 explains the controller cache 102 as one of a series of “high speed storage for temporarily storing data being passed between a higher speed component of the system” (¶ [0109]) and as a “volatile buffer memory . . . which is typically part of the controller” for host-originated “logically consecutive data” before staging in the binary portion of MLC that it shares with the Gavens embodiment. Paley ‘180 similarly explains that “[t]ypically high-speed volatile RAM are employed . . . in a controller cache 102 of the memory controller.”

(¶ [0197].) Both Gorobets ‘912 and Paley ‘180 incorporated by reference (note 53 above) (1) Harari (Ex. 1057), which shows as part of its Fig. 8 “cache buffer 705” as part of a cache system 701 “as part of the controller,” explained at 12:20-35, 13:24-42 and 12:58-14:11, and (2) Gorobets PCT (Ex. 1058), which explained:

The SRAM block 30 stores all temporary information [thus volatile memory] within the controller 16, this storing function includes [1] the buffering of [logical] sector data and [2] storage of control data structures and variables, as well as firmware code.

. . . .

During the operation of the controller all [logical] sector data being transferred between the logical interface 14 to host system 12, and the physical interface 18 to Flash memory 20 is buffered in the SRAM 30. Sufficient capacity in the SRAM 30 is allocated for buffering of two sectors

(11:10-12:3 [emphasis added].) A POSITA would further understand that there would be random access volatile memory for controller functions in the controller, including mapping (*e.g.*, Rao ‘916 [Ex. 1004] Fig. 3 [“InstReg.” and “Add- Map”]; Moshayedi [Ex. 1043] Fig. 2 [RAM 214]) as disclosed in Gavens, at least in incorporated references Gorobets PCT 11:12-13 (quoted above, [2]), Gorobets ‘912 ¶ [0018] and Harari Fig. 8 (memory 709). However, it is also well-known to use DRAM (external to a controller chip) in a memory system such as shown on an SSD in Micheloni Fig.1.16 (Ex. 1025 at 14, reproduced at ¶ 30 above) in the same detail

as DRAM 20 in ‘369 Fig. 1. As an example, Fig. 1 of Moshayedi (Ex. 1043) shows DRAM 106 external to controller 102 and controlling banks of SLC modules (112) and MLC modules (114), much as ‘369 Fig. 1. Moshayedi explains that “DRAM 106 may be used to buffer data to be written into the flash array 108 . . . which may be incoming data from a host system/and or data being rewritten from one portion of the flash array 108 to another portion of the flash array 108.” ([¶ [0028]].) The DRAM can also be used for tables and linked lists which may be stored to SLC on power-down. (*Id.* ¶ [0053].) Either qualify as “at least one” RAVM. Because only a certain number of parallel bits may be received from the host or sent to a memory array, there may be multiple caches (including the flash memory chip page buffer).

117. Thus, Gavens and its incorporated references disclose or, in view of the POSITA’s knowledge of NAND flash, renders obvious claim element [1c].

5. [1d] at least one controller [1d1] to operate physical memory elements and associated memory space for associated Write access operations to the physical memory elements and Read access operations from the physical memory elements, [1d2] the controller having a controller memory associated therewith for storing received data therein;

118. [1d1] Gavens Fig. 1 shows controller 102. The controller 102 manages one or more memory chips 100 in the memory device 90. (8:17-18, 32-35.) Taking the “memory space” as logical addresses, in Gavens, “[a] memory block

management system implemented in the controller stages the [logical] sectors and maps and stores them to the physical structure of the memory array” (8:36-39.)

119. **[1d2]** As explained at paragraph 116 above, Gavens and its incorporated reference, Gorobets PCT (Ex. 1058) discloses in-controller RAVM for storing (caching) received data. Also explained there, off-controller DRAM is common for data caching use (*e.g.*, ¶ 31 above, Moshayedi Fig. 1(106), ¶ [0028].

120. Thus, Gavens and its incorporated references disclose or make obvious in view of the knowledge of the POSITA the ordinary NAND flash L2P mapping and writing operation of claim element **[1d]**.

6. [1e] a flash translation layer (FTL), wherein the at least one controller, or the FTL, or a combination of both maintain an address table in one or more of the physical memory elements;

121. An FTL would be understood by the POSITA as the software layer supporting the L2P functions of a NAND controller (¶¶ 31-38 above, Micheloni [Ex. 1025] at 40, 492-98, and 516-17 [Fig. 18.3]), and thus is included within a NAND controller performing such functions as disclosed in Gavens. Gavens discloses the ordinary NAND controller function: “A memory block management system implemented in the controller stages the sectors and maps and stores them to the physical structure of the memory array” (8:36-39.) Gavens discloses updating of “a directory in a block management system embedded in the firmware of the controller (see Fig. 1)” to direct accesses to one or the other portions of the memory (17:17-

19, 18:47-49, 18:64-66). Gavens, for further explanation of block management, incorporated by reference Paley ‘180 (note 52 above):

122. Incorporated reference Paley ‘180 (Ex. 1051) provides further detail:

A memory-side memory manager 300 is implemented in the controller 100 of the memory system 90 to manage the storage and retrieval of the data of host logical sectors among metablocks of the flash memory 200. The memory manager comprises a front-end system 310 and a back-end system 320. The front-end system 310 includes a host interface 312. The back-end system 320 includes a number of software modules for managing erase, read and write operations of the metablocks. The memory manager also maintains system control data and directory data associated with its operations among the flash memory 200 and the controller RAM 130.

(¶ [0155] [emphasis added]; *also* Gorobets ‘912 ¶ [0095].) Paley explains the use of “the currently prevalent LBA interface” (¶ [0020]). “The memory manager 300 is preferably implemented in the controller 100. It translates logical addresses received from the host into physical addresses within the memory array, where the data are actually stored, and then keeps track of these address translations.” (¶ [0150].) Referring to Fig. 9, “the media management layer 330” (¶ [0157]) and “dataflow and sequencing layer 340” (¶ [0158]) combine to perform the role of FTL. Specifically, the media management layer handles “updates, tables and lists used for logical-to-physical address translation and free block management, and wear

leveling based on hot counts” (§ [0201] [emphasis added]). As further explained in the incorporated Gorobets PCT (Ex. 1058), the media management layer is implemented in firmware “control[ing] the transfer of data sectors between the logical interface 14 to host 12 and the physical interface 18 to Flash memory 20” (Gorobets PCT 13:5-8 [emphasis added]) with the following algorithms:

The data structures required to support the Cyclic Storage [7:20-24: “method of ensuring uniformity of distribution of use”⁵⁴] media management algorithms [18:1-10: (1) Data Write location to write host data, (2) Block Erase controlling erasure of obsolete data, (3) Block Sequencing of flash blocks used for storing data, and (4) Address Translation controlling “mapping of host logical addresses to physical memory addresses”] are stored principally in flash memory 10 together with the host data sectors, with only a very limited amount of control data being held temporarily in the control processor’s volatile RAM 30. Information held in the volatile memory 30 is non-critical, and can be reconstructed from flash memory if the power supply is interrupted.

⁵⁴ An objective is “to obtain even use of blocks in the non-volatile memory for storage of data over a long period of operation, and to avoid ‘hot-spots’ in non-volatile memory usage, thereby increasing the reliability of the memory system.” (Gorobets PCT 2:4-8 [emphasis added].)

(27:22-28:5 [emphasis added].) As I explained at paragraph 34, it was well-known, as corroborated by Goodson (submitted with the 2012 ‘298 application), that the address tables were loaded in controller RAVM and backed up in the flash memory. The NAND flash memory system disclosed in Gavens as further detailed in its incorporated references (note 52 above), explained:

The top level of hierarchy of the mapping structure [Fig. 5 schematic of the “address translation process”] is the sector address table 52, which is a master table containing a physical address for every logical sector stored in the system 10 and which is stored in Flash memory 20. . . .

The sector address record 56 is a list stored in the controller’s memory volatile memory SRAM 30 It may also be reconstructed during device initiation from the sequence of recently written sectors which may be traced in the Flash memory 20. The intermediate temporary sector address table 54 is contained in flash memory 20 and is updated with the contents of the sector address record 56

This hierarchy of mapping structures . . . efficiently supports logical to physical address translation . . . even if electrical power is unpredictably removed from the system 10.

(Gorobets PCT [Ex. 1058] 26:11-27:21 [emphasis added].) Gavens thus discloses an FTL that maintains address tables in both the RAVM and the flash memory.

123. Thus, Gavens discloses claim element **[1e]**.

7. [1f] [1f1] the controller controlling access of the physical MLC and SLC nonvolatile memory elements and the random access volatile physical memory elements for storage of the received data therein as stored data, [1f2] the controller, in at least a Write access operation transferring the stored received data from the controller memory to a given one of the physical MLC nonvolatile memory elements in an associated physical MLC memory module, [1f3] operable to store the received data in the given one of the physical MLC nonvolatile memory elements as stored data and retain such received data in the random access volatile physical memory element as retained data associated with stored data;

124. I incorporate my explanation in Sections X(A)(3)-(6) above regarding the L2P, writing, reading and block management operations of the Gavens controller/FTL, including use of address tables to map logical and physical addresses that may be blocks or pages in ordinary NAND flash operation. These are common to the NAND flash background art ('369 2:28-3:10 [writing host data to NAND flash nonvolatile memory]; ¶¶ 30-38 above).

125. [1f1] It was shown in Sections X(A)(3)-(6) that Gavens discloses a controller which “cooperates with the [NAND flash] memory chip [MLC partially operable in 2-state, pSLC mode, obviously substitutable with ‘real’ SLC] and controls and manages high level operations,” for example, a host write translated in logical sectors to the memory array (8:32-36) [thus storage of data received from the host 80 in Fig. 1]; the operation obviously involves control of the RAM data cache or buffer disclosed by Gavens (Section X(A)(4), *e.g.*, incorporated Paley ‘180 ¶

[0197]) where the received data may be received as recited in claim element [1d] and temporarily stored (or retained).

126. **[1f2]** In one variant of the Gavens post-write-read error management embodiment of Figs. 14 and 15 (explaining text at 16:13-17:42), user data is written to the higher density (*e.g.*, MLC) portion as a “first copy.” (Gavens 16:41-45.) It is obvious that the controller controls this operation, transferring data “from” a memory or cache associated with it (§ 119 [1d2] above). With the POSITA understanding that the transferred “stored received data” is the data written to (and “to be stored” in) MLC, the recited “Write access” operation is satisfied by the Gavens error management embodiment by “storing” data in the MLC **[1f3]** while operating the system to maintain the “original copy which may be cached” (16:46-49 [emphasis added]) as the “retained data” in the disclosed on-controller or associated external RAVM (or both). (Section X(A)(4) above.) As explained at paragraphs 71 and 93 above, no more (but less) detail is provided in the ‘369 DETAILED DESCRIPTION.

127. Thus, Gavens and its incorporated references disclose, or in view of POSITA knowledge, make obvious claim element **[1f]**.

8. [1g] the controller [1g1] performing a data integrity test on stored data in the given one of the physical MLC nonvolatile memory elements in the associated one of the physical MLC memory modules after the at least a Write access operation performed thereon by [1g2] reading the stored data to the controller memory and [1g3] comparing the stored data in the controller memory in the given one of the physical MLC nonvolatile memory elements to the retained data that was associated with the stored data in the random access volatile physical memory element by the controller during the Write access operation;

128. As reviewed at paragraphs 53-55 above, Gavens discloses multiple techniques for error management, efficiently deployed near the end-of-life of a memory system. One discloses the ‘369 patent data integrity test as recited in element [1g].

129. In one variant of the Gavens post-write-read error management embodiment of Figs. 14 and 15 (explaining text at 16:13-17:42), user data is written to the higher density (*e.g.*, MLC) portion as a “first copy.” (Gavens 16:41-45.) Then, in a “post write read,” there may be a check for errors [1g1], [1g3] “by comparison with the **original copy** which may be cached” (16:46-49 [emphasis added]) – which may be in the RAVM of the controller memory or associated DRAM (¶¶ 116 and 122 above). If the number of error bits is less than a predetermined amount, the first copy is deemed valid and subsequent reads of the data page will be from that copy with any errors corrected by ECC. (16:50-56.) If the number is greater, then a copy is written to the lower density, lower error rate (more robust, *e.g.*, pSLC) portion;

this is performed in one embodiment by copying from a cached copy “of the original data.” (17:9-11.) In the Gavens NAND flash technology corpus, “[u]sing RAM in a write cache operating with flash memory has been disclosed” (*e.g.*, Gorobets ‘912 [Ex. 1049] ¶ [0024], Paley ‘180 [Ex. 1051] ¶ [0027]). The “original data” temporarily stored (“cached” in RAM, understood to be RAVM) is compared to the stored (in flash) data, and if the stored data fails to match, the “original data” (temporarily stored or “retained”) is sent to the more robust pSLC. Thus, this error management variant discloses a data integrity test on written (stored) data that compares it against retained (cached) “original data” in RAM cache. It would be obvious that the “stored data” to be compared would be **[1g2]** written to the controller memory (where the Gavens/Gorobets cache has space for at least two write units, nominally, pages) for comparison using logic on the controller chip (*e.g.*, incorporated Rao ‘916 [Ex. 1004] Fig. 1 [“ALU”]).

130. It is understood that these write and compare actions are directed by the controller. The use of “real” SLC memory instead of a binary-mode portion of the MLC memory does not affect element [1h] and does not increase overall latency, since “copying” of the “stored data” is from the MLC, involving relatively slow decoding (reading), and reading of the “retained data” is from relatively fast RAM; any subsequent “transfer” (write) for element [1i] would be faster for “real” SLC than pSLC (*see* notes 11 & 45 above).

131. Thus, Gavens and its incorporated references disclose claim element [1g].

9. [1h] wherein the address table maps logical addresses to physical addresses of the memory system, wherein the mapping is performed as necessitated by the memory system to maximize lifetime; and

132. I incorporate my explanation in Section X(A)(6) regarding Gavens's use of address tables to map logical and physical addresses that may be blocks or pages in ordinary NAND flash operation. To the extent that the "maximize lifetime" limitation can be understood (see ¶ 81 above), Gavens's (1) flash system media management wear leveling (¶ 122 and note 54 above) and (2) error management disclosing (Section X(A)(8)) claim elements [1g] and [1i] remapping and transfer for data integrity failure each satisfy this limitation by enhancing lifetime.

133. Thus, Gavens and its incorporated references disclose claim element [1h].

10.[1i] wherein a failure of the data integrity test performed on the stored data by the controller results in [1i1] a remapping of logical address space to a different physical range of addresses in the physical address space and [1i2] transfer of data corresponding to the retained data to those remapped physical addresses in the physical address space from those physical addresses in the physical address space determined to have failed the data integrity test.

134. If this element is not found unsupported, indefinite and inoperative, ignoring the misplaced modifier requiring the transfer of data at physical addresses

found to have failed the data integrity test and construing instead claim element [1i] as requiring both remapping and transferring of the retained “original” data that was written to an originally mapped physical location from which a copy failed a data integrity test comparison, as explained in Section X(A)(8), Gavens, in at least one disclosed error management technique, disclosed claim element [1i].

135. As explained in Section X(A)(8), in the Gavens embodiment, after the data stored in the less robust (MLC) portion fails a data integrity test by exceeding a predetermined number of bit errors, the “original data” is rewritten to the more robust (pSLC) portion (Gavens 17:9-12). For that to occur, the original data must be remapped [1i1], and any “transfer” [1i2] in this context is the writing of that original data to the remapped – different – location (physical address), thereby satisfying the remapping and transfer limitation of claim element [1i].

136. Thus, Gavens discloses or makes obvious, in view of the knowledge of the POSITA, claim element [1i].

* * * * *

137. In summary, Gavens and its incorporated references make obvious, in view of knowledge of the POSITA, each limitation of ‘369 claim 1.

B. Claim 2 Is Obvious Over Gavens, Including Incorporated References, in View of Knowledge of the POSITA.

1. [2Pre] The system of claim 1,

138. I incorporate here my opinions set forth in Section X(A) for claim 1.

2. [2] wherein the FTL flash translation layer is a software module, or a firmware module containing software updates.

139. As explained in Section X(A)(6), Gavens, though its reference for block management (8:40-43), Paley ‘180 (Ex. 1051) (§§ [0155]-[0159], [0201]), which further incorporated Gorobets PCT (Ex. 1058) (note 52 above), disclosed the use of a media management layer and a dataflow and sequencing layer that perform: a) the L2P mapping/translation, and b) the wear leveling and block management functions of known FTL (*compare* ‘300 3:3-35, §§ 34-38 above). The next sentence in Gavens (8:44-45) refers to firmware 60 to implement the functions of the controller, and Gorobets PCT disclosed its sector sequencer, media management layer and flash controller layer, “control[ling] the transfer of data sectors between the logical interface 14 to host 12 and the physical interface 10 to Flash memory 20,” specifically as firmware (12:22-13:8), clearly updatable as located in SRAM (11:10-13).

140. Gavens and its incorporated references, Paley ‘180 and Gorobets PCT, disclose claim 2.

C. Claim 3 Is Obvious Over Gavens, Including Incorporated References, in View of Knowledge of the POSITA.

1. [3Pre] The system of claim 1,

141. I incorporate here my opinions set forth in Section X(A) for claim 1.

2. [3] wherein at least one of the random access volatile memory or the MLC and SLC nonvolatile memory elements are embedded in the at least one controller.

142. As explained in Section X(A)(4), Gavens discloses, through incorporated references, the embedding of RAVM in the controller both as a data cache and as a temporary location for data for other controller functions (*e.g.*, respectively, Gorobets '912 [Ex. 1049] ¶ [0197], [0018]; *also*, Gorobets PCT [Ex. 1058] 11:10-13 [both]). Although there was an argument made during prosecution of the '300 patent application that two “blocks” of “nonvolatile [sic, probably ‘volatile’] memory” were required for the data integrity test and that a “conventional cache” was not that large (¶ 83 above), there is no support in the specification for any hardware departure from the known hybrid MLC-SLC memory system and no claim is made to such a new hardware combination. Indeed, incorporated Gorobets PCT (Ex. 1058) explained that “[s]ufficient capacity in the SRAM 30 is allocated for buffering of two sectors” (11:24-12:2).

143. Thus, Gavens and its incorporated references disclose **claim 3**.

D. Claim 4 Is Obvious Over Gavens, Including Incorporated References, in View of Knowledge of the POSITA.

1. [4Pre] The system of claim 1,

144. I incorporate here my opinions set forth in Section X(A) for claim 1.

2. [4] wherein the physical MLC and SLC nonvolatile memory elements comprise flash memory.

145. Gavens, which would be understood by a POSITA to describe a NAND flash system with multiple MLC NAND flash modules, expressly describes its MLC non-volatile memory modules as including MLC NAND flash memory elements, which by definition are flash memory. (Fig. 4, 9:49-50 [page of memory cells in the memory array 200 is organized in the NAND configuration], 1:18-19, 4:14-18 [flash memory having a first portion and a second portion storing higher density compared to the first portion], 2:65-66, 8:65-67 [flash memory devices with NAND string structures; Fig. 1 (90) [flash memory device].)

146. As explained at paragraphs 105 and 114 above, a POSITA would readily and successfully substitute for Gavens's partitions of MLC memory elements configured for 2-state pseudo-SLC operation an SLC non-volatile memory module which is understood to include SLC NAND flash memory elements. As to Gavens's MLC NAND flash memory operated in pseudo-SLC mode, they are also described as flash memory. (Fig. 4, 9:49-50 [a page of memory cells in the memory array 200 is organized in the NAND configuration], 1:18-19, 4:14-18 [flash memory having a

first portion and a second portion storing higher density compared to the first portion], 2:65-66, 8:65-67 [flash memory devices with NAND string structures], Fig. 1 [flash memory device 90].)

147. Thus, Gavens and incorporated references, in view of the knowledge of the POSITA, discloses **claim 4**.

E. Claim 5 Is Obvious Over Gavens, Including Incorporated References, in View of Knowledge of the POSITA.

1. [5Pre] The system of claim 1,

148. I incorporate here my opinions set forth in Section X(A) for claim 1.

2. [5] wherein the random access volatile memory is dynamic random access memory.

149. In view of a POSITA's understanding that dynamic random access memory (DRAM) is a common type of random access volatile memory (RAVM) that may be used for purposes of temporarily holding data to be compared as is the recited role of RAVM in claim 1, it would be an obvious choice for Gavens's cache of "original data" for comparison (16:47-49). The background art supports such use of DRAM for cache and other NAND flash functions. (*E.g.*, Moshayedi [Ex. 1043] Fig. 1 (106), ¶ [0028] [Dynamic Random Access Memory 106 shown external to but "can be present in controller 102" and "may be used to buffer data to be written into the flash array 108"], *also* ¶ [0004] [DRAM used in management of read and write operations of flash, contain addressing table correlating logical and physical

addresses], ¶ [0053] [DRAM for linked lists, saved on SLC], ¶ [0054] [DRAM for tables and 4K logical blocks of data cache], ¶ [0067] [suitable amounts of DRAM for different SLC-MLC combinations]; Duann (Ex. 1034), p. 6 [utilize DRAM as cache].)

150. Thus, Gavens, in view of the knowledge of the POSITA, makes obvious **claim 5.**

F. Claim 6 Is Obvious Over Gavens, Including Incorporated References, in View of Knowledge of the POSITA.

1. [6Pre] The system of claim 1,

151. I incorporate here my opinions set forth in Section X(A) for claim 1.

2. [6] wherein the random access volatile memory is static random access memory.

152. Similar to claim 5, in view of a POSITA's understanding that static random access memory (SRAM) is common type of random access volatile memory (RAVM) that may be used for purposes of temporarily holding data to be compared as is the recited role of RAVM in claim 1, it would be an obvious choice for Gavens's cache of "original data" for comparison (16:47-49), particularly on the controller die because of compatible fabrication processes. Gavens discloses the use of SRAM through incorporation of Gorobets PCT (Ex. 1058) (11:10-12:3 [buffering of logical data and storage of control data and tables], *quoted* at ¶ 109 above; *also* Paley '180 Fig. 17, ¶¶ [0423]-[0425], [0502], [0503], [0506] [SRAM for storing "group address

tables”]). Duann (Ex. 1034), p. 6, suggested “[b]igger internal SRAM as cache” for a hybrid SLC and MLC system.

153. Thus, Gavens and incorporated Paley ‘180, in view of the knowledge of the POSITA, make obvious **claim 6**.

G. Claim 7 Is Obvious Over Gavens, Including Incorporated References, in View of Knowledge of the POSITA.

1. [7Pre] The system of claim 1,

154. I incorporate here my opinions set forth in Section X(A) for claim 1.

2. [7] wherein the at least one physical SLC nonvolatile memory element includes a hard disk drive (HDD).

155. A POSITA would understand that it is physically impossible for an SLC flash memory chip (much less an included “element”) to include a hard disk drive, where the latter is orders of magnitude larger than a flash chip. If this and all the inconsistencies with claim 1 were ignored, including the antecedent of being a part of an “SLC” module and changing the nature of an “SLC nonvolatile memory element,” a POSITA would know that there are some useful combinations of flash and HDD such as disclosed in U.S. Patent No. 8,225,035 to Colligan (Ex. 1052) 2:33-36, 5:40-6:6, 7:67-8:2, and 8:23-25 (flash memory could supplement HDDs for more frequent writes) and the POSITA could call a combination of disk(s) 24 and SLC flash 28 of ‘369 Fig. 1 an “SLC nonvolatile memory element” to replace bad MLC blocks as physical address destinations. Thus, Gavens, in view of the

knowledge of the POSITA, discloses claim 7 with an overstretched construction of “element.”

156. Thus, Gavens and its incorporated references, in view of the knowledge of the POSITA, make obvious **claim 7**.

H. Claim 8 Is Obvious Over Gavens, Including Incorporated References, in View of Knowledge of the POSITA.

1. [8Pre] The system of claim 1,

157. I incorporate here my opinions set forth in Section X(A) for claim 1.

2. [8] wherein the at least one controller is associated with a processor external to the memory system.

158. Gavens Fig. 1 shows communication of the controller 101 with a host (computer/processor) 80 external to the memory system 90.

159. Thus, Gavens and its incorporated references, in view of the knowledge of the POSITA, make obvious **claim 8**.

I. Claim 9 Is Obvious Over Gavens, Including Incorporated References, in View of Knowledge of the POSITA.

1. [9Pre] The system of claim 1,

160. I incorporate here my opinions set forth in Section X(A) for claim 1.

2. [9] wherein the remapped physical addresses in the physical address space are located in the SLC memory module.

161. As shown in Section X(A)(6), Gavens remaps to the lower density, pSLC memory, and it would be obvious (§ 101) to substitute “real” SLC.

162. Thus, Gavens and its incorporated references, in view of the knowledge of the POSITA, make obvious **claim 9**.

J. Claim 10 Is Obvious Over Gavens, Including Incorporated References, in View of Knowledge of the POSITA.

1. [10Pre] The system of claim 1,

163. I incorporate here my opinions set forth in Section X(A) for claim 1.

2. [10] wherein the FTL is associated with the MLC memory modules and the SLC memory modules in the bank of physical nonvolatile memory.

164. As shown in Section X(A)(6), Gavens's FTL, maintaining the address maps for the MLC and SLC modules, clearly is associated with the MLC and SLC modules. Moreover, the address maps are also backed up in flash.

165. Thus, Gavens and incorporated references, in view of the knowledge of the POSITA, make obvious **claim 10**.

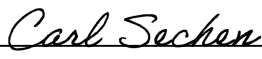
* * * * *

166. For the reasons given above, claims 1-10 of the '369 patent are invalid.

167. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful

false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: December 26, 2024



Carl Sechen, Ph.D.