

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION,
Petitioner

v.

BiTMICRO LLC,
Patent Owner.

IPR2023-00782
U.S. Patent No. 8,010,740

**DECLARATION OF DR. CARL SECHEN, PH.D.
IN SUPPORT OF INTER PARTES REVIEW OF U.S. 8,010,740**

TABLE OF CONTENTS

I.	INTRODUCTION	1
A.	Qualifications and Expertise	3
B.	Compensation.....	6
C.	Previous Expert Witness Experience	7
D.	Materials Considered and Basis of Opinions	7
E.	Level of Ordinary Skill in the Art	11
F.	Understanding of Relevant Legal Principles.....	12
II.	TECHNOLOGY OVERVIEW.....	15
A.	Flash Memory.....	15
B.	Address Mapping	17
C.	Large-Capacity Flash Memory Systems	27
D.	Interleaving.....	30
III.	THE CHALLENGED '740 PATENT	33
A.	Summary of the '740 Patent.....	33
B.	Prosecution History of the '740 Patent	36
C.	Claim Construction	38
1.	“LBA”	39
2.	“LBA set”.....	39
IV.	GROUND 1: CLAIMS 1 AND 14 ARE OBVIOUS OVER BRUCE '006.....	41
A.	Overview of Bruce '006.....	41
B.	Independent Claim 1	44

1.	Claim 1 Preamble.....	45
2.	Claim 1, Limitation [A]	58
3.	Claim 1, Limitation [B].....	66
4.	Claim 1, Limitation [C].....	72
C.	Dependent Claim 14.....	76
V.	GROUND 2: CLAIMS 1 AND 14 ARE OBVIOUS OVER THE COMBINATION OF BRUCE '006 AND SBC.....	78
A.	Overview of SBC	78
B.	Motivation to Combine Bruce '006 with SBC.....	80
C.	Independent Claim 1	87
1.	Claim 1 Preamble.....	87
2.	Claim 1, Limitation [A]	93
3.	Claim 1, Limitation [B].....	93
4.	Claim 1, Limitation [C].....	94
D.	Dependent Claim 14.....	98
VI.	GROUND 3: CLAIMS 1, 9-15, 32, AND 34 ARE OBVIOUS OVER THE COMBINATION OF BRUCE '006 AND BRUCE '251	99
A.	Overview of Bruce '251	99
B.	Motivation to Combine Bruce '006 with Bruce '251	101
C.	Independent Claim 1	112
1.	Claim 1 Preamble.....	112
2.	Claim 1, Limitations [A]-[C]	115
D.	Dependent Claim 9	115

E.	Dependent Claim 10.....	122
F.	Dependent Claim 11	123
G.	Dependent Claim 12.....	123
H.	Dependent Claim 13	123
I.	Dependent Claim 14.....	124
J.	Dependent Claim 15.....	124
	1. Claim 15, Limitation [A]	124
	2. Claim 15, Limitation [B].....	126
K.	Independent Claim 32	128
	1. Claim 32 Preamble.....	128
	2. Claim 32, Limitation [A]	133
	3. Claim 32, Limitation [B].....	133
	4. Claim 32, Limitation [C].....	134
	5. Claim 32, Limitation [D]	135
	6. Claim 32, Limitation [E].....	136
L.	Dependent Claim 34.....	137
VII.	GROUND 4: CLAIMS 1, 9-15, 32, AND 34 ARE OBVIOUS OVER THE COMBINATION OF BRUCE '006, BRUCE '251, AND SBC	138
A.	Motivation to Combine Bruce '006, Bruce '251, and SBC	138
B.	Independent Claim 1	141
	1. Claim 1 Preamble.....	141
C.	Dependent Claims 9-15	145

D.	Independent Claim 32	146
1.	Claim 32 Preamble.....	146
2.	Claim 32, Limitations [A]-[D].....	147
3.	Claim 32, Limitation [E].....	147
E.	Dependent Claim 34.....	148

I. INTRODUCTION

1. My name is Carl Sechen, and I have been retained by Perkins Coie LLP on behalf of Intel Corporation (“Intel” or “Petitioner”) in the above-referenced *inter partes* review proceeding to evaluate United States Patent No. 8,010,740 to Arcedera et al. (“the ’740 Patent”) against certain references that predate the earliest possible priority date of June 8, 2006 (the “Earliest Priority Date”) for the ’740 Patent. The ’740 Patent is attached as Exhibit INTEL-1001 to Intel’s Petition for *Inter Partes* Review of U.S. Patent No. 8,010,740 based on United States Patent No. 6,000,006 (“Bruce ’006”), United States Patent No. 5,822,251 (“Bruce ’251”), and *American National Standard for Information Technology - SCSI-3 Block Commands* (“SBC”). I am informed that Intel seeks review of claims 1, 9-15, 32, and 34 of the ’740 Patent (collectively, the “challenged claims”). I have also been informed that the ’740 Patent is currently assigned to BiTMICRO LLC (“BiTMICRO” or “Patent Owner”). As detailed in this declaration, it is my opinion that each of the challenged claims is disclosed and/or rendered obvious by prior art references that predate the Earliest Priority Date.

2. I am over 18 years of age. I have personal knowledge of the facts stated in this Declaration and if requested by the Patent Trial and Appeal Board

(“PTAB” or “Board”), I am prepared to testify competently about my opinions expressed herein.

3. I have reviewed and am familiar with the specification and the claims of the ’740 Patent. In general, I will cite to the specification of a United States patent using the following formats: (Patent No. at Col:Line Number(s)) or (Patent Publication No. at Paragraph Number(s)). For example, the citation (’740 Patent at 1:1-10) points to the ’740 Patent specification at column 1, lines 1-10. Also, for convenience, I use italics to denote limitations from the challenged claims.

4. All of the opinions contained in this Declaration are based on the documents I reviewed and my knowledge and professional judgment. In forming the opinions expressed in this Declaration, I reviewed the documents listed below. I have also reviewed and am familiar with the other materials referred to in this Declaration. To the best of my knowledge, the documents mentioned above are true and accurate copies of what they purport to be and are the kinds of materials that an individual with expertise in this field at the relevant time period would reasonably rely on in formulating opinions, such as those set forth in this declaration.

5. I have been asked to provide my technical opinions regarding how a person of ordinary skill in the art (“POSITA”) would have understood the claims of the ’740 Patent at the time of the alleged invention. For purposes of whether the

teachings of the prior art anticipate or render obvious the claims of the '740 Patent, I have been asked to assume the date of June 8, 2006, the filing date of the earliest patent in the priority chain of the '740 Patent, for the analysis in this document. I refer to that date as the Earliest Priority Date. I have also been asked to provide my technical opinions on how concepts in the '740 Patent specification relate to claim limitations of the '740 Patent. In reaching the opinions provided herein, I have considered the '740 Patent, its prosecution history, and the references cited above and have drawn as appropriate on my own education, training, research, knowledge, and personal and professional experience.

A. Qualifications and Expertise

6. I am an expert in the field of memory system design. In formulating my opinions, I have relied on my knowledge, training, and experience in the relevant field, which I will summarize briefly. A more detailed summary of my background, education, experience, and publications is set forth in my curriculum vitae ("CV"), which is attached as Exhibit INTEL-1004 to the Petition. My CV also includes a list of all other cases in which, during the previous 4 years, I testified as an expert at trial or by deposition.

7. I earned my B.E.E. in Electrical Engineering from the University of Minnesota in 1975, followed by my M.S. in Electrical Engineering from the

Massachusetts Institute of Technology in 1977 and then my Ph.D. in Electrical Engineering from the University of California, Berkeley in 1986.

8. I have been a Professor of Electrical Engineering for 36 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.

9. 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.

10. 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 and non-volatile memory design. I have taught these courses continuously for the past 27 years. I have taught digital

integrated circuit design and memory design, including flash and non-volatile memory design, to undergraduate and graduate students at the University of Washington and at the University of Texas at Dallas. NAND-Flash memory design is taught in detail, including both SLC and MLC variants.

11. 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.

12. 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 Engineering 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.

13. 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.

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

B. Compensation

15. In connection with my work as an expert, I am being compensated at an hourly rate of \$400 for consulting services including time spent testifying at any hearings that may be held. I am also being reimbursed for reasonable and customary expenses associated with my work in this case. I receive no other form of compensation related to this case. My compensation is not dependent or otherwise contingent upon the specifics of my testimony or the outcome of this action.

C. Previous Expert Witness Experience

16. My CV attached as Exhibit INTEL-1004 to the Petition contains a list of cases in which I have testified at trial, hearing, or by deposition within the preceding four years.

D. Materials Considered and Basis of Opinions

17. In forming the opinions set forth in this Declaration, I have considered and relied upon my education, experience, and knowledge of the relevant fields. I have also reviewed and considered the '740 Patent and its prosecution history ("’740 FH"). The '740 Patent and its prosecution history are attached as exhibits INTEL-1001 and INTEL-1002, respectively, to the above-referenced IPR petition. Additionally, I have relied on and reviewed the following documents and materials cited in this Declaration, all of which are attached as exhibits to the above-referenced IPR petitions.

Exhibit	Reference
1001	U.S. Patent No. 8,010,740 to Arcedera et al. ("’740 Patent")
1002	File History of the '740 Patent ("’740 FH")
1004	CV of Dr. Carl Sechen, Ph.D.
1005	U.S. Patent No. 6,000,006 to Bruce et al. ("Bruce '006")
1006	U.S. Patent No. 5,822,251 to Bruce et al. ("Bruce '251")

Exhibit	Reference
1007	“ <i>American National Standard for Information Technology - SCSI-3 Block Commands (SBC)</i> ,” American National Standards Institute (1998) (“SBC”)
1008	United States District Courts — National Judicial Caseload Profile
1009	The IEEE Standard Dictionary of Electrical and Electronics Terms (6th Ed. 1996)
1010	U.S. Patent No. 5,519,844 to Stallmo (“Stallmo”)
1011	Microsoft Press Computer Dictionary (3rd Ed. 1999)
1012	U.S. Patent No. 7,299,316 to Chou et al. (“Chou”)
1013	Summons with Proof of Service, <i>BiTMICRO LLC v. Intel Corp.</i> , No. 6:22-cv-00335-ADA (W.D. Tex.), Dkt. 9
1014	Declaration of June Munford in Support of Petition
1015	Claim Construction Order and Memorandum In Support Thereof, <i>BiTMICRO LLC v. Kioxia America, Inc., Kioxia Corporation</i> , No. 6:22-cv-00331-ADA (W.D. Tex.), Dkt. 54
1016	U.S. Patent App. Pub. No. 2003/0163630 to Aasheim et al. (“Aasheim”)
1017	Wu et al., <i>An Efficient B-Tree Layer for Flash-Memory Storage Systems</i> , RTCSA 2003: Real-Time and Embedded Computing Sys. and Applications: 9th Int’l. Conf., Revised Papers, Vol. 2968, pp. 409-430 (February 18-20, 2004) (“Wu”)
1018	Gal et al., <i>Algorithms and Data Structures for Flash Memories</i> , ACM Computing Survs., Vol. 37, No. 2, pp. 138-163 (June 2005) (“Gal”)

Exhibit	Reference
1019	Kim et al., <i>A Space-Efficient Flash Translation Layer for CompactFlash Systems</i> , IEEE Transactions on Consumer Elecs., Vol. 48, No. 2, pp. 366-375 (May 2002) (“Kim”)
1020	Pavan et al., <i>Flash Memory Cells—An Overview</i> , Proc. of the IEEE, Vol. 85, No. 8, pp. 1248-1271 (1997) (“Pavan”)
1021	Yim, <i>A Novel Memory Hierarchy for Flash Memory Based Storage Systems</i> , JSTS: Journal of Semiconductor Technology and Science, Vol. 5, No. 4, pp. 262-269 (2005) (“Yim”)
1022	U.S. Patent No. 5,479,638 to Assar et al. (“Assar”)
1023	U.S. Patent No. 7,000,063 to Friedman et al. (“Friedman”)
1024	Huffman et al., <i>A Standard Interface for NAND Flash</i> , Intel Developer Forum (March 7, 2006) (“Huffman”)
1025	Chang et al., <i>An Adaptive Striping Architecture for Flash Memory Storage Systems of Embedded Systems</i> , Proceedings. Eighth IEEE Real-Time and Embedded Tech. and Applications Symp. pp. 187-196 (September 2002) (“Chang”)
1026	Park et al., <i>A High Performance Controller for NAND Flash-based Solid State Disk (NSSD)</i> , 2006 21st IEEE Non-Volatile Semiconductor Memory Workshop, pp. 17-20 (May 15, 2006) (“Park”)
1027	U.S. Patent No. 7,506,098 to Arcedera et al. (“Arcedera”)
1028	Tsur, <i>Trends in COTS Storage Solutions for Data Acquisition Systems</i> , Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications III, Vol. 6209, pp. 45-55 SPIE (May 5, 2006) (“Tsur”)
1029	Ryu et al., <i>Improvement of Space Utilization in NAND Flash Memory Storages</i> , Embedded Software and Systems: Second International Conference, ICESS 2005, Xi’an, China, Proceedings 2, pp. 766-775 (December 16-18, 2005) (“Ryu”)

Exhibit	Reference
1030	Chung et al., <i>LSTAFF: System Software For Large Block Flash Memory</i> , Lecture Notes in Computer Science, 3398 pp. 704-710 (2005) (“Chung”)
1031	U.S. Patent No. 5,404,485 to Ban (“Ban”)
1032	Peterson, <i>Designing A VME-To-SCSI Adapter</i> , The System Engineers Handbook, pp. 311-319 (1992) (“Peterson”)
1033	American National Standards Institute (ANSI) X3.131-1986, <i>Small Computer System Interface (SCSI)</i> (1986) (“SCSI-1986”)
1034	U.S. Patent No. 6,381,677 to Beardsley et al. (“Beardsley”)

18. I reserve the right to supplement this Declaration and rely upon any additional information, deposition testimony, documents, or materials that may be provided to me or that are relied upon by any of Respondent’s experts or witnesses.

19. I also reserve my right to supplement this Declaration and rely upon additional information that becomes available to me.

20. In connection with live testimony in this proceeding, should I be asked to provide it, I may rely on visual aids and demonstratives to demonstrate the bases of my opinions, such as claim charts, patent drawings, excerpts from patent specifications, file histories, interrogatory responses, deposition transcripts and exhibits, as well as charts, diagrams, videos and animated or computer-generated video. The demonstrative exhibits I may use to accompany any testimony I may give may also include documents, testimony, and other evidence cited in my

Declaration, portions of such evidence, and other portions of my Declaration and the attached exhibits.

E. Level of Ordinary Skill in the Art

21. I am told that certain issues relating to the validity of the '740 Patent must be judged from the perspective of a person of ordinary skill in the relevant art, as I discuss below. I have been asked to define the level of a “person of ordinary skill in the art” or “POSITA” at the time the alleged invention as claimed was made. In deciding the level of ordinary skill, I have considered the following factors, which I have been informed are relevant to the determination of ordinary skill in the art:

- the levels of education and experience of persons working in the field;
- the types of problems encountered in the art;
- the prior art solutions to these problems;
- the rapidity with which innovations are made; and
- the sophistication of the technology.

22. In my opinion, for the purposes of the '740 Patent, a POSITA, at the Earliest Priority Date (June 8, 2006), would have had at least a bachelor's degree in electrical engineering or computer engineering and at least two years of experience in memory system design, including solid-state memory, or equivalent combined education, work, and/or experience.

23. My opinion below explains how a POSITA would have understood the technology described in the references I have identified in this declaration as of the Earliest Priority Date.

F. Understanding of Relevant Legal Principles

24. I am not a lawyer and will not provide any legal opinions. Although I am not a lawyer, I have been advised that certain legal standards are to be applied by technical experts in forming opinions regarding the meaning and validity of patent claims.

25. I understand that a patent claim is invalid if it is anticipated or obvious in view of the prior art, and that a claim can be unpatentable even if all of the requirements of the claim cannot be found in a single prior-art reference. I further understand that invalidity of a claim requires that the claim be anticipated or obvious from the perspective of a person of ordinary skill in the art at the time the invention was made.

26. I have been informed that a patent claim is invalid if it would have been obvious to a person of ordinary skill in the art. In analyzing the obviousness of a claim, I understand the following factors may be considered: (1) the scope and content of the prior art; (2) the differences between the prior art and the claims; (3) the level of ordinary skill in the art; and (4) any so called “secondary considerations” of non-obviousness if they are present. Such secondary

considerations include: commercial success of products covered by the patent claims; a long-felt need for the invention; failed attempts by others to make the invention; copying of the invention by others in the field; unexpected results achieved by the invention as compared to the closest prior art; praise of the invention by the infringer or others in the field; the taking of licenses under the patent by others; expressions of surprise by experts and those skilled in the art at the making of the invention; and the patentee proceeded contrary to the accepted wisdom of the prior art. I am not aware of any evidence of secondary considerations of non-obviousness relevant to the '740 Patent. I reserve the right to supplement this Declaration if Patent Owner introduces evidence of secondary considerations of non-obviousness.

27. I understand that to prove that prior art or a combination of prior art renders a patent obvious, it is necessary to:

- (1) identify the particular references that, singly or in combination, make the patent obvious;
- (2) specifically identify which elements of the patent claim appear in each of the asserted references; and
- (3) explain why a person of ordinary skill in the art would have combined the references, and how they would have done so, to create the

inventions claimed in the patent. I further understand that exemplary rationales that may support a conclusion of obviousness include:

- combining prior art elements according to known methods to yield predictable results;
- simple substitution of one known element for another to obtain predictable results;
- use of known technique(s) to improve similar devices (methods or products) in the same way;
- “obvious to try” – choosing from a finite number of identified, predictable solutions with a reasonable expectation of success;
- known work in one field of endeavor may prompt variations of the work for use in either the same field or a different field based on design incentives or other market forces if the variations are predictable to a person of ordinary skill in the art; and
- some teaching, suggestion, or motivating in the prior art that would have led a person of ordinary skill in the art to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

28. I have been informed that, in considering obviousness, hindsight reasoning derived from the patent-at-issue may not be used.

II. TECHNOLOGY OVERVIEW

A. Flash Memory

29. Rotating mechanical hard disk drives (“HDDs”) have been widely used for persistent (*i.e.*, non-volatile) storage in computing systems since their introduction by IBM in 1956. HDDs, however, are mechanical devices with many moving parts. As a result, they are large, slow, consume a lot of power, and prone to failure. These characteristics reduce HDD’s suitability as a portable storage medium.

30. Flash memory was first invented in the early 1980s by Japanese engineer Masuoka Fujio as a technology that would replace existing data-storage media such as HDDs. Like the HDD, flash memory is non-volatile. A semiconductor-based non-volatile storage medium with no moving parts, flash memory is smaller, hardier, and consumes less power than HDDs. Flash memory has thus gained use in a wide range of applications, particularly those requiring portability or high performance.

31. There are two types of flash memories: NOR and NAND. NOR flash is better suited for random data access as it uses no shared components and connects individual memory cells in parallel. NAND, on the other hand, is better suited for serial data access as it stores data in “blocks” of “pages.” Bruce ’006 at 1:20; Bruce ’251 at 1:46; Chang at 1; Aasheim at ¶[0030]; Wu at 411. Toshiba

commercially launched the first NAND flash memory in 1987 and Intel introduced the first commercial NOR type flash chip in 1988.

32. While flash memory is smaller, hardier, and consumes less power than HDDs, flash memory does have two downsides as compared to HDDs. First, flash memory must be erased before it can be rewritten, an attribute sometimes referred to as “erase-before-write.” Bruce ’006 at 1:26-28, 5:60-61; Bruce ’251 at 1:53-54; Chang at 1; Wu at 411-412; Aasheim at ¶¶[0033]-[0039]; Kim at 366; Yim at 262. Erase operations are typically performed on regions of fixed size greater than are required for read/write operations. Ryu at 768; Bruce ’006 at 1:28-31; Bruce ’251 at 1:53-57. For example, NAND flash—the most popular type of flash memory—is read and written a **page** at a time but must be erased a **block** at a time. Yim at 262-263; Wu at 411.

33. Second, flash memory has a finite number of erase/write cycles, meaning the memory cells will eventually “wear out.” Bruce ’006 at 1:34-36; Friedman at 1:19-24; Assar at 1:37-38; Gal at 138; Wu at 3-4. Put differently, there is a limited number of times that any particular block of flash memory can be erased and rewritten, typically between 10,000 and 1,000,000 cycles as of the Earliest Priority Date. Gal at 138; Aasheim at ¶[0037]; Assar at 1:39-43; Chang at 1.

B. Address Mapping

34. Before the Earliest Priority Date, flash memories were commonly used as non-volatile storage replacements for HDDs in “host” systems such as personal computers, mobile phones, digital cameras, PDAs, and portable music players. Bruce '251 at 1:24-34 (“[F]lash-memory systems are frequently used as a mass-storage replacement for a hard disk on a personal computer and are thus sometimes referred to as a flash ‘disk’, even though no rotating disks are used.”); Assar at Abstract; Pavan at 1248.

35. The unique read/write/erase behaviors of flash memory caused compatibility issues with systems designed for use with other types of data-storage devices. Ban, a U.S. patent filed in 1993 titled “Flash File System” and originally assigned to M-Systems Flash Disk Pioneers Ltd., acknowledges this problem in the Background of the Invention section:

In a typical computer system, the operating system program is responsible for data management of the data storage devices that are a part of the system. A necessary, and usually sufficient, attribute of a data storage device to achieve compatibility with the operating system program is that it can read data from, and write data to, any location in the data storage medium. Thus, **flash memories are not compatible with typical existing operating system programs, since data cannot be written to an area of flash memory in which data has previously been written, unless the area is first erased.**

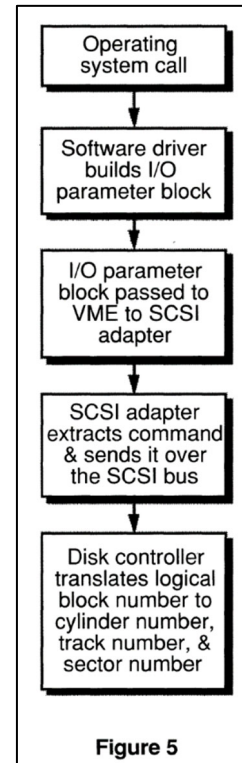
Ban at 1:29-39 (emphasis added).

36. Well-known techniques before the Earliest Priority Date allowed host systems using storage-management techniques designed for other types of memory devices like HDDs to better interface with flash-memory systems. Gal at 138-139; Ryu at 766-770. Some of these techniques were invented specifically for flash memories, but many were adapted from older techniques that were originally invented for other storage devices. “Address mapping,” sometimes called “logical-to-physical” mapping, is an example of the latter. Chang at 2; Aasheim at ¶[0005].

37. Ban describes address mapping in a flash-memory system. Ban teaches a “virtual memory map” that “control[s] and manage[s] access to a flash memory so that the flash memory appears to the computer operating system as a data storage device in which it is possible to read data from, and write data to, any flash memory location.” Ban at Abstract, 1:52-57. Ban explains that the “**virtual memory map is a table**” in which virtual block addresses are mapped to physical addresses in the flash memory. Ban at 2:20-28 (emphasis added). The flash controller in Ban uses the virtual map to “establish a correspondence” between a “logical address” and a “physical address,” after which a read/write operation is performed at the physical address block location. Ban at 4:56-59, 5:29-35, 5:47-52.

38. By using address mapping, a host system need not know the physical location of a piece of information (which may change over time), but only the logical location. Aasheim at ¶[0056].

39. The concept of address mapping is old in the art and actually predates the introduction of flash memory. For example, the Small Computer System Interface (“SCSI”), first introduced in the 1980s, facilitates data communication among computers and peripherals using block addressing. In other words, the SCSI interface dealt with logical blocks instead of the cylinders, heads, and sectors (“CHS”) used in traditional HDDs. **Figure 5** of Peterson shows that “the flow of control in a SCSI-based subsystem” includes translation from “logical block number” to “cylinder number, track number, & sector number.”



Peterson at 313. Use of the SCSI standard together with flash-memory devices was widespread before the Earliest Priority Date. Tsur at 6 (“In 2001, Ultra Wide SCSI solid-state flash disks were introduced[.]”); Park at 1 (Figure 2).

40. As can be seen, by the Earliest Priority Date, use of address mapping was well known, including in the context of flash-memory systems. In fact, address mapping was so well known by that point it was used for sophisticated applications like memory optimization. Chang, titled *An Adaptive Striping*

Architecture for Flash Memory Storage Systems of Embedded Systems and published in September 2002, describes the use of address mapping to achieve a “striping architecture to introduce I/O parallelism to [NAND] flash-memory storage systems.” Chang at 2. Chang explains that its flash-memory storage system “provides the file system a collection of logical blocks for reads/writes,” each of which “is identified by a unique logical block address (LBA).” Chang at 3. Chang goes on to note that “each logical block is stored in a page of flash memory” and clarifies that “**when we use the term ‘logical block’, it is for a logical block viewed by the file system.**” Chang at 3 (emphasis added). In other words, the “block” in LBA has no relation to a “block” in the flash-memory storage system. *See also* Wu at 3 (“Note that a ‘logical block’ usually denotes a disk sector.”), 4 (“Figure 2: The logical block address ‘i’ is mapped to the physical page number ‘1024’ by the translation table.”).

41. The dynamic address translation between an LBA and its physical location in flash memory is accomplished through the use of an “address translation table ... indexed by LBA,” shown in **Figure 3**. Chang at 4. The address translation table is part of a “flash translation layer” (“FTL”) driver that “provides

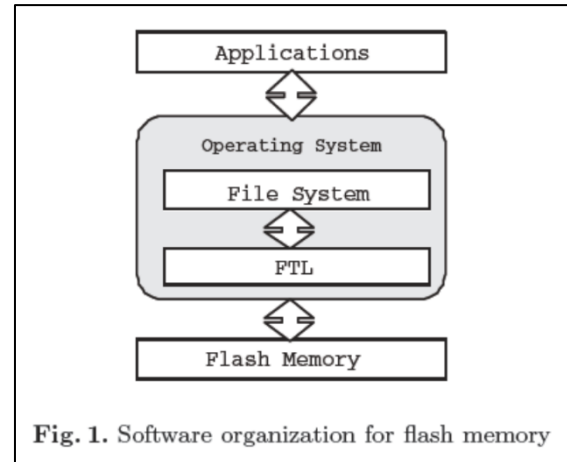
Figure 3. Chang at 4. The address translation table is part of a “flash translation layer” (“FTL”) driver that “provides

transparent access for file systems and user applications via block device emulation.” Chang at 3-4.

42. FTL was well known in the art before the Earliest Priority Date. FTL is an intermediate software layer between a host system and flash memories that performs address mapping between logical and physical addresses. Ryu at 768; Yim at 263; Kim at 366-368; Aasheim at ¶¶[0049]-[0066]. FTL “provides a common block I/O interface to the host which is similar to the hard disks” and uses logical addressing to “logically hide[] the erase operation of the flash.” Yim at 263, 265.

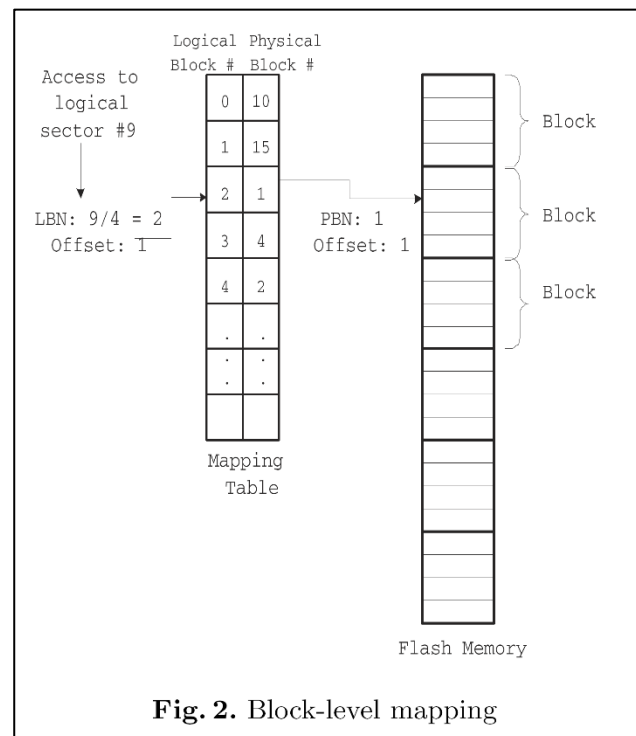
43. Ryu, a 2005 paper titled *Improvement of Space Utilization in NAND Flash Memory Storages*, also teaches an address mapping table as part of an FTL. Ryu at 767-768. The FTL “works in conjunction with [the] file system to make flash memory device[s] appear to the system like a disk drive.” Ryu at 767.

Figure 1 of Ryu shows “a typical software organization for NAND-type flash memory” where the FTL sits between the “File System” and the “Flash Memory.” Ryu at 767.



44. “The FTL usually maintains the address mapping table in order to map logical address of I/O requests to physical address in flash memory.” Ryu at 768. “The address translation table is indexed by logical block address (LBA), and each entry of the table contains the physical address of the corresponding LBA.” Ryu at 768.

45. Unlike Chang, which teaches mapping only at the page level, Ryu explains that “[t]he mapping can be maintained either at the page level or at the block level.” Ryu at 769. Ryu explains that in page-level address mapping, “a logical page can be mapped to any physical page in flash memory.” Ryu at 769. Ryu cautions, however, that “this mapping requires a large amount of space to store the needed mapping table.” Ryu at 769. As an alternative, Ryu describes block-level address mapping, shown in **Figure 2**. Ryu at 768. In block-level mapping, “the logical address is divided into a logical block address and a block offset, and only the logical block address is translated into a physical block address in flash memory.” Ryu at 769.

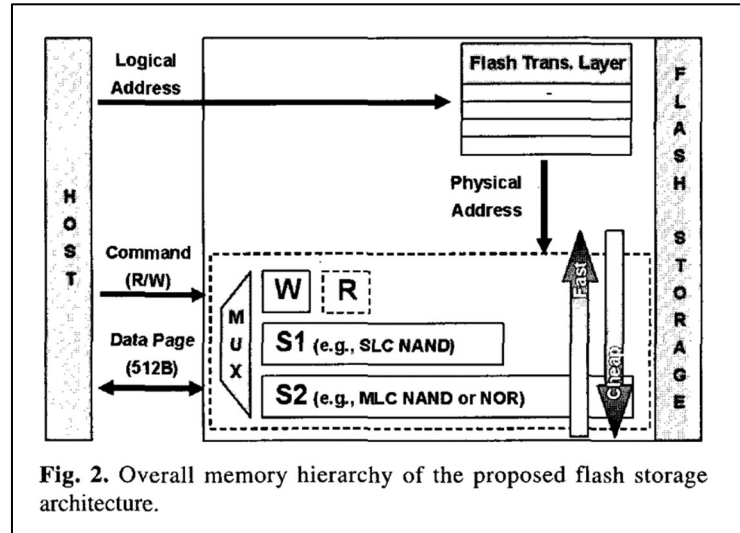


46. Chung, titled *LSTAFF: System Software for Large Block Flash Memory* and published in 2005, teaches an FTL mapping algorithm for a “large block flash memory system.” Chung at 4. Chung states that existing “FTL algorithms are designed for small block flash memory in which the size of a physical page of flash memory is same to the size of data sector of operating

system.” Chung at 1. Chung notes, however, that recently, “major flash vendors have produced large block flash memory” in which “the basic chunk for reading and writing is larger than operating system data sector sizes,” for example, “2K or 1K bytes,” as opposed to a “data sector size [for] ... most operating systems [of] 512 bytes.” Chung at 1.

47. Chung’s proposed solution, called LSTAFF (Large State Transition Applied Fast Flash Translation Layer), “is designed to provide maximum performance in large block flash memory whose page size is larger than file system’s data sector size.” Chung at 9. Chung teaches a “hybrid mapping” technique that uses a “block mapping technique to find the physical block corresponding to the logical block” followed by “sector mapping techniques ... to find an available sector within the physical block.” Chung at 3. The algorithm **“first finds the physical block number from the logical block number according to the mapping table**, and then, by reading the logical sector numbers within the physical block, it can **read the requested data.**” Chung at 3 (emphasis added), 5 (describing “three levels of logical to physical mapping ... in the large block flash system: sector, page, and block mapping”).

48. Similar disclosures can be found in Yim, a 2005 paper that describes a flash storage architecture that maintains a logical address for a piece of information used by the host computer, as shown in **Figure 2**. Yim at 265. A translation (or mapping) is then used to track the correlation between the logical address and the physical location of the data. Yim at 263, 265.



49. As can be seen, logical-to-physical address mapping, specifically in the context of flash-memory systems, was well known in the art before the Earliest Priority Date. In fact, BitMICRO described the use of a mapping table to translate logical addresses to physical address in a flash system in Bruce '006, filed almost nine years before the Earliest Priority Date:

FIG. 3 is a diagram illustrating re-mapping of all accesses to a flash memory System. Logical block addresses (LBA's) from a user or host System are received and used as an index into a re-map table. The re-map table includes a physical block address (PBA) for each possible logical block address. Since all accesses first pass through the re-map table, logical addresses can be arbitrarily translated into physical addresses for the flash devices. The physical addresses do not have to correspond to the logical addresses, allowing more-efficient wear-leveling to occur. Flash blocks can be swapped and frequently-written

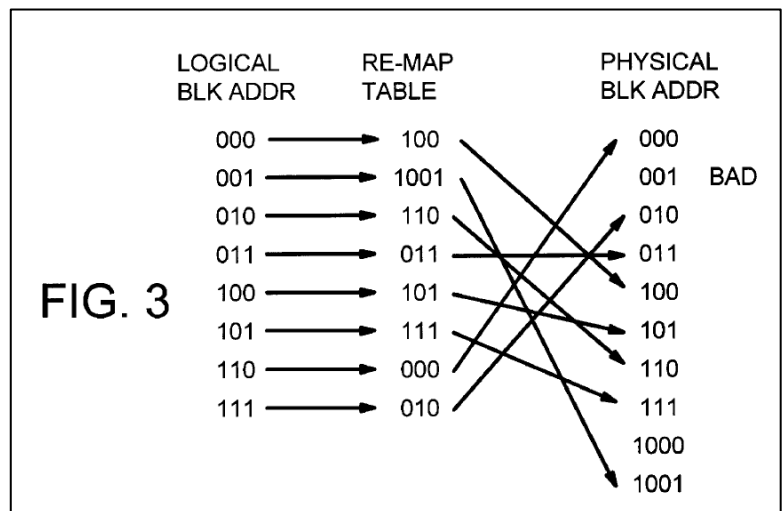
addresses allocated to different flash blocks over time without regard to the logical address. The access time is the same whether a block is re-mapped or not.

Bruce '006 at 5:39-52.

50. More specifically, Bruce '006 teaches a “re-map table” for “re-mapping ... all accesses to a flash memory system.” Bruce '006 at 5:39-42.

Logical block addresses (LBAs) from a host system are “received and used as an index into a re-map table.” Bruce '006 at 5:40-42. The re-map table includes a physical block address (“PBA”) for each possible logical block address. Bruce

'006 at 5:42-43. “Since all accesses first pass through the re-map table, logical addresses can be arbitrarily translated into physical addresses for the flash



devices.” Bruce '006 at 5:44-46. The re-map table is shown in **Figure 3**.

51. One well-known benefit of address mapping is the ability for a system using flash memory to perform a technique known as “wear leveling.” Bruce '006 at 2:57-59; Aasheim at ¶[0091], claim 1; Assar at Abstract, 4:6-11. Wear leveling is a technique—intended to address the two deficiencies of flash memory discussed in ¶¶ 32-33 above—that equalizes the “wear” (*i.e.*, the number of erase operations)

of a flash memory by using all areas in the flash memory in an equal manner. Friedman at 1:24-28; Aasheim at ¶[0099]. When data stored in a block of flash memory is updated, the system avoids an expensive erase-before-write operation by writing the updated data to an unused block of memory and re-directing subsequent read accesses instead of erasing and overwriting the old data with the updated data. Friedman at 1:28-33; Assar at 2:52-55. The old data is marked as “old” and is not erased until the block has to be reused. Friedman at 1:33-34.

52. Address mapping correlates the host logical address to a physical location in memory so that wear-leveling may occur with systems that are not natively configured to track continually-changing physical locations of data files. Assar at 2:56-63; Bruce '006 at 5:48-51 (“Flash blocks can be swapped and frequently-written addresses allocated to different flash blocks over time without regard to the logical address.”); Aasheim at ¶[0056] (“The logical-to-physical sector mapping module 404 permits data to be flexibly assigned to any physical location on the flash memory medium, which provides efficiency for other tasks, such as wear-leveling[.]”).

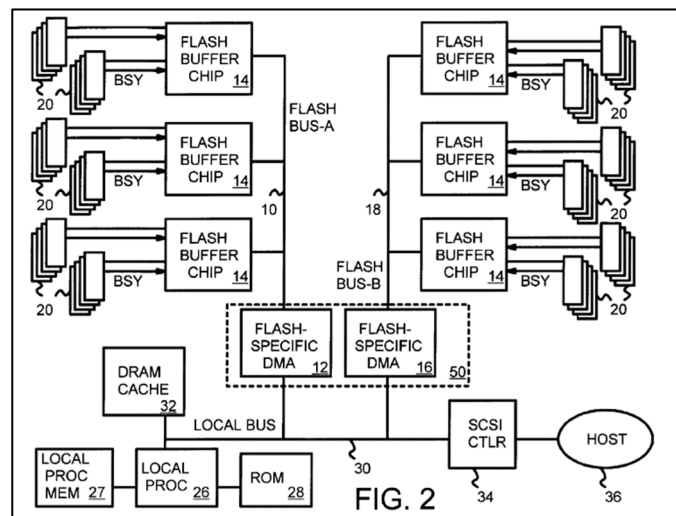
C. Large-Capacity Flash Memory Systems

53. Flash memory has found widespread use in mobile and embedded systems (e.g., digital cameras, cell phones, PDAs) because it is non-volatile, shock-resistant, and power-economic. Chang at 1. As users demanded higher-quality

real-time data, flash-memory technology has evolved to support larger storage capacities with faster I/O speeds. Yim at 262 (“As the system users want to manipulate high-quality and large-size data in real-time, flash storage shall provide a larger capacity with a faster I/O speed. ... [F]lash storage contains more many numbers of NAND flash chips (e.g., from 2 to 16) than the previous product.”); Pavan at 1248 (a major application for flash memories “is to create storing elements, like memory boards or solid-state hard disks, made by Flash memory arrays which are configured to create large-size memories to compete with miniaturized hard disks”); Bruce ’251 at 1:24-34.

54. BiTMICRO itself described such an expandable flash memory system in Bruce ’251, filed almost nine years before the Earliest Priority Date. Bruce ’251

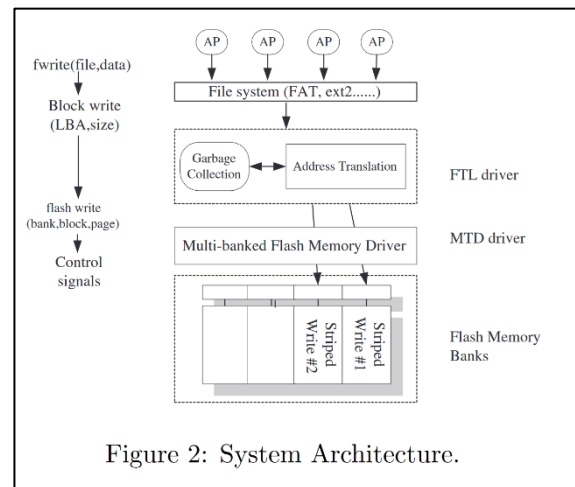
teaches an expandable flash memory system comprising flash memory-specific direct memory access (“DMA”) controllers that control access to the flash memory by generating a sequence of command bytes and address bytes to



initiate data transfer from the flash memory. Bruce ’251 at Abstract, 2:49-55. A flash bus is coupled to the DMA controller to transmit the command and address

bytes to flash buffer chips connected to banks of flash memory chips. Bruce '251 at Abstract, 3:4-8. **Figure 2** of Bruce '251 depicts an expandable flash memory system with two flash-specific DMAs, each connected to three flash buffer chips which are each connected to multiple flash memory chips.

55. Chang similarly teaches a NAND flash storage system comprising multiple NAND flash banks with an FTL driver that “provides transparent access for file systems and user applications via block device emulation.” Chang at 3. The system architecture comprising multiple flash-memory banks is illustrated in **Figure 2**.



56. Yim, a 2005 paper titled *A Novel Memory Hierarchy For Flash Memory Based Storage Systems*, teaches a flash-memory system that combines “many types of flash memories” “in order to maximize the I/O performance while reducing the cost per byte ratio.” Yim at 263, 264. Yim further teaches that “I/O parallelism” is one of “two fundamental techniques [in I/O subsystems] for improving the I/O speed.” Yim at 263 (“[I]n flash storage, multiple flash chips are already in use in order to partially parallelize the I/O operations[.]”). Yim

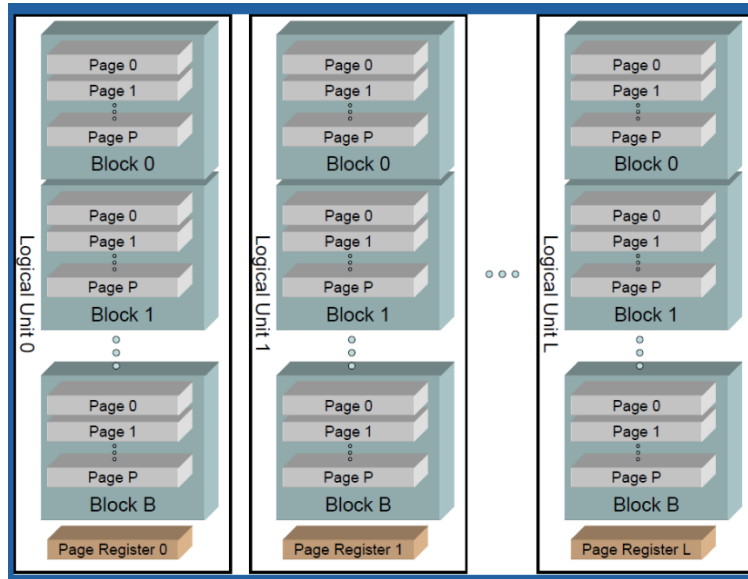
identifies “interleaved memory” as a type of parallel I/O architecture. Yim at 263.

57. A POSITA as of the Earliest Priority Date would have understood that a flash-memory system like the one described in Bruce '251 would be ideally suited for interleaving because the DMA engines, flash-buffer chips, and buses disclosed therein would permit parallel memory operations across the flash-memory chips. Interleaving is described in further detail below.

D. Interleaving

58. Interleaving was another well-known technique as of the Earliest Priority Date. Interleaving refers to executing certain flash-memory operations concurrently and/or overlapped in time. Bruce '006 at 11:20-22; Chang at 2-3; Park at 1. Interleaving significantly increases the speed and/or throughput of certain flash memory operations.

59. Open NAND Flash Interface (“ONFI”) was unveiled in early 2006 to develop a “standardized NAND Flash interface that allows interoperability between NAND devices.” Huffman at 20. Under the ONFI standard, blocks are grouped into “logical units,” or “LUNs,” which are grouped into independent units with their own chip enables. Huffman at 26-27.



Huffman at 27.

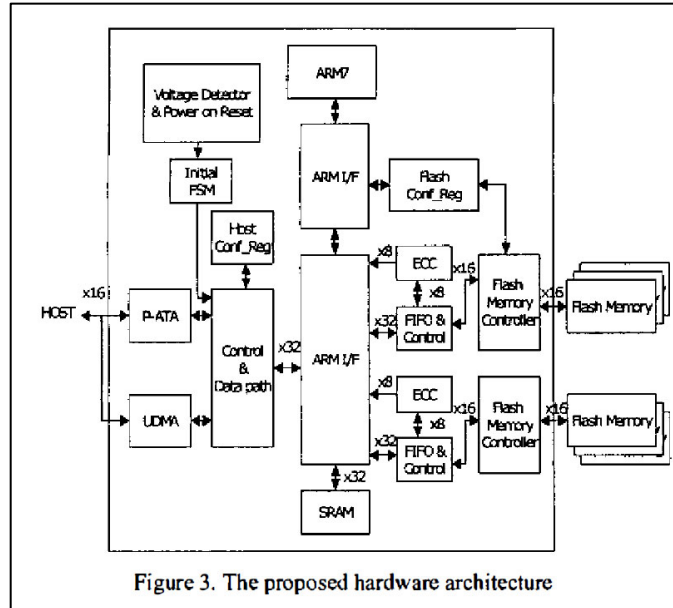
60. The ONFI standard further discloses that “[e]ach LUN may support interleaved addressing for increased parallelism” for read, program, and erase operations. Huffman at 27. “Interleaving may be used to complete the same operation on additional blocks on a per logical unit basis to enhance performance.” Huffman at 33 (“Concurrent interleaving: Operations to all of the blocks is issued at the same time and then executes in parallel”).

61. Park, presented in February 2006 and published in May 2006, describes a hardware architecture for a NAND flash memory-based solid-state disk. Park at 1. The NAND SSD “can be interfaced directly with the Host using the IDE standard interface (ATA-5), [and] supports up to Ultra DMA Mode 4.” Park at 1. Park explains that “the total capacity can range from 256MBytes to

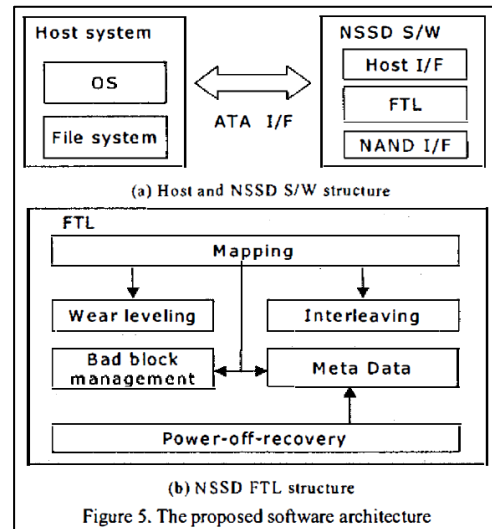
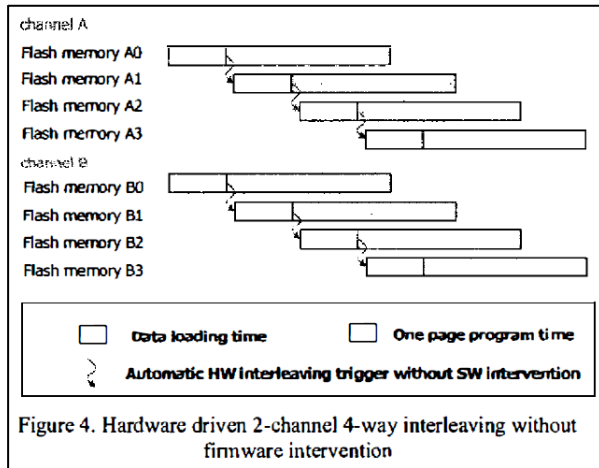
32GBytes” and “[t]he flash memory chips are interfaced to each channel to support parallel write operations.” Park at 1. The proposed hardware architecture is shown

in **Figure 3**. Park at 2.

62. Park explains that “the key factor of improving NSSD performance is to accelerate sequential write operation using interleaving scheme of multiple NAND flash memories.” Figure 4



shows the “automatic interleaving hardware logic” that “maximize[s] the interleaving effect, while the firmware intervention is minimized” and Figure 5 shows the proposed software architecture, including the flash translation layer (FTL) that “converts the logical address from the host into the physical address of the Flash memory”:



Park at 2.

63. BiTMICRO itself acknowledged the well-known performance improvement associated with the use of interleaving in Bruce '006:

Interleaving can be combined with the invention. **Interleaving is desirable since multiple accesses can occur in parallel to the flash devices, improving performance.** Interleaving is accomplished by altering the physical address bits assigned to decoders for the flash devices and the physical address bits assigned to decode blocks and pages within a flash device.

Bruce '006 at 11:20-26 (emphasis added).

III. THE CHALLENGED '740 PATENT

A. Summary of the '740 Patent

64. The '740 Patent describes a mapping table that associates LBAs from “[a] host logical addressing system to the device addressing system used by” a storage device. '740 Patent at 6:42-44. As noted below in ¶76 75, the '740 Patent defines “LBA” or “logical block address” as “an address that is part of a logical

addressing system ... used by a host.” ’740 Patent at 6:37-40. The mapping table, illustrated in annotated Figure 2 below, “includes a set of fields, named logical fields 26” (annotated in **blue**) “and a set of PBA fields 28” (annotated in **red**).
’740 Patent at 5:29-30.

9

Logical Fields 26	PBA Fields 28			
	Access Parameter Fields 30			Memory Index Fields 32
	Bus Id Fields 34	FDE Id Fields 36	Group Id Fields 38	
40 LBAs 00-07	8-0	22-01	15-0	46 0x00
42 LBAs 08-15	8-0	22-02	15-1	48 0x08
44 LBAs 16-23	8-1	22-11	15-0	50 0x00
45 LBAs 24-31	8-1	22-12	15-1	51 0x08
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮

’740 Patent at Figure 2.

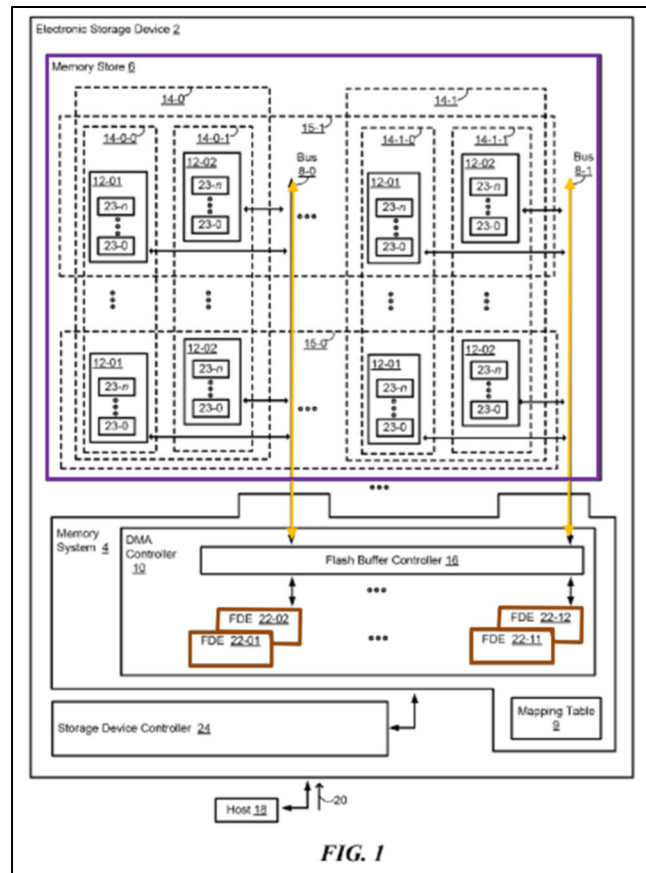
65. The ’740 Patent teaches that the logical fields 40, 42, 44, and 45 in set of logical fields 26 each represents a “LBA set” and, as noted below in ¶77, the ’740 Patent defines each LBA set as a “set[] of consecutive LBAs” (annotated in **green**). ’740 Patent at 6:46-51.

66. The ’740 Patent further teaches that each PBA field 46, 48, 50, and 51 represents “a unique addressable physical memory location, named PBA, in memory store 6” (annotated in **purple** in **Figure 1** below). ’740 Patent at 5:30-33.

For each PBA field, the '740 Patent indicates that “th[e] physical memory location may be represented by using a set of access parameters fields 30 and a memory index field 32.” '740 Patent at 5:33-35. “The set of access parameter[] fields 30 includes a bus identifier field 34, a FDE identifier field 36, and a group identifier field 38.” '740 Patent at 5:36-38.

67. The '740 Patent states that “[l]ogical fields in mapping table 9 may be initialized to represent LBA sets that are serially arranged in mapping table 9 so that LBAs from adjacent LBA sets are contiguous” and that “any two adjacent PBAs differ by at least one access parameter.” '740 Patent at 6:55-57, 7:5-6. The '740 Patent explains that for “[a]n I/O transaction that is associated with consecutive LBAs [which] is sometimes referred to as a sequential I/O transaction,” “[s]ince mapping table 9 has been initialized to map these LBAs 00-07 and LBAs 08-15 with LBA sets 40 and 42, respectively, DMA controller would more than likely be able to perform memory operations required by I/O transaction request 20 in an optimal manner by interleaving these memory operations across different buses, different FDEs, different flash memory devices, or any combination of these.” '740 Patent at 9:28-37.

68. For memory store, the '740 Patent explains that it “may be configured to include a set of solid state memory devices that are coupled to a set of buses [(annotated in orange)] and that are controlled by a set of FDEs [(annotated in brown)].” '740 Patent at 4:54-56. The '740 Patent further describes an FDE as “a device that is capable of controlling a flash memory device and performing DMA operations on the flash memory device in response to commands generated by storage device 2 through storage device controller 24.” '740 Patent at 3:35-39.



B. Prosecution History of the '740 Patent

69. On November 25, 2008, the '461 application was filed as a continuation-in-part of U.S. Patent Application No. 11/450,005 (“the '005 application”). The '005 application was filed on June 8, 2006 and issued as U.S. Patent No. 7,506,098 on March 17, 2009.

70. The '461 application was originally filed with three independent claims and 37 dependent claims. '740 FH at 32-43. On December 30, 2008, the

applicant preliminarily amended the specification to (1) remove a priority claim to U.S. Patent Application No. 11/450,023 (“the ’023 application”), filed on June 8, 2006, (and issued on November 3, 2009, as U.S. Patent No. 7,613,876), and (2) expressly “incorporate[] by reference” the ’005 and ’023 applications. ’740 FH at 57-58.

71. On October 25, 2010, the Examiner issued a Non-Final Office Action rejecting all claims under 35 U.S.C. § 112 ¶ 1 for lack of written description support because “[t]he specification does not describe how the mapping table causes the storage device to perform optimized memory operation on memory location in such a way that is understood by one having ordinary skill in the related art.” ’740 FH at 77-78.

72. On February 25, 2011, the applicant responded that “the Specification teaches how a ‘mapping table’ causes a ‘storage device to perform optimized memory operations,’” and identified paragraphs [0023]-[0025], [0034], and [0036] of the as-filed specification as “disclos[ing] a mapping table, that when used by a storage device, **causes** the storage device to perform optimized memory operations.” ’740 FH at 129-132 (emphasis in original). The applicant further argued that paragraphs [0034] and [0036] supported pending claim 32 (“*wherein said mapping table increases the likelihood that that [sic] the operational load imposed on the storage apparatus during the processing of the I/O transaction*

request will be optimally distributed across storage device resources”) because those paragraphs “teach[] how a ‘mapping table increases the likelihood that the operational load imposed on the storage apparatus’ ... ‘will be optimally distributed across storage device resources.’” ’740 FH at 133-135.

73. On April 13, 2011, the Examiner allowed the claims. ’740 FH at 147-150. On July 12, 2011, the applicant filed an amendment after allowance making minor changes. ’740 FH at 218-219, 233. On July 28, 2011, the Examiner allowed the amendment. ’740 FH at 245. On August 30, 2011, the ’740 Patent issued.

C. Claim Construction

74. I understand that “claim construction” is the process of determining a patent claim’s meaning. I understand that the Board applies the same claim construction standard used in district courts which seeks to give claim terms their plain and ordinary meaning to a POSITA at the time of the claimed invention in view of the claims, specification, and prosecution history. I also understand that if the patent applicant gave a term a special meaning in the specification or during prosecution, then the special meaning should be used. Lastly, I understand that claim construction is a matter of law and that the Board will ultimately determine the final claim construction.

75. In this section, I identify the claim terms for which the '740 Patent includes express definition to which I have applied in this Declaration. For the claim terms “optimizing memory operations” and “optimally distributed,” I am aware that a district court for another co-pending litigation has construed them to have a plain and ordinary meaning which I have applied in this Declaration. Claim Construction Order and Memorandum In Support Thereof at 31, 34-35. For other claim terms, I give the terms their plain and ordinary meaning to a POSITA unless otherwise noted.

1. “LBA”

76. The '740 Patent teaches that “[t]he term ‘LBA’, which may also be referred to herein as a logical block address, is intended to represent an address that is part of a logical addressing system (not shown) used by a host.” '740 Patent at 6:37-40.

2. “LBA set”

77. The '740 Patent teaches that “[t]hese LBAs may be grouped into sets of consecutive LBAs, named ‘LBA sets.’” '740 Patent at 6:46-48.

78. I note here that although the term “block” appears in “logical block address,” the '740 Patent teaches that the size of the claimed logical block address (LBA) or LBA sets are not tied to the size of a physical flash block. The '740 Patent first provides an example of a flash block within a flash memory device

with a “block size of 16 KB, [and] an LBA set size of 4KB.” ’740 Patent at 7:25-26. As evident in this example, the LBA set size (4KB) is smaller than the size of a physical block (16 KB). Since each LBA set is a set of consecutive LBAs, each LBA size must also be smaller than a flash block size. Therefore, the ’740 Patent teaches that the size for the claimed LBAs and LBA sets can be smaller than the size of a physical flash block. The ’740 Patent also teaches that the size for the claimed LBAs and LBA sets can be the same or larger than the size of a physical flash block. The ’740 Patent teaches that “[i]n the example shown in FIG. 2, the LBA set size selected is set equal to a section size ... The term ‘section size’ is the size of the minimum portion in which data may be transferred or relocated between physical memory locations in memory store 6. In addition, the section size may be limited to a size that is at least equal to the product of a positive integer multiplied by the native page size used by a flash block. For example, if the native page size for flash block 23-0 is 4KB, then the section size that may be used would be limited to 4 KB, 8KB, 12K, and so on.” Since the ’740 Patent teaches that each LBA set size is set equal to a section size and the section size can be equal to the product of any positive integer multiplied by the native page size, the ’740 Patent teaches that the LBA set size can be equal to the size of a flash block. For example, if there are 16 pages in each flash block, by setting the section size to be 16 times of a page size, the LBA set would also be set to 16 times of a page size

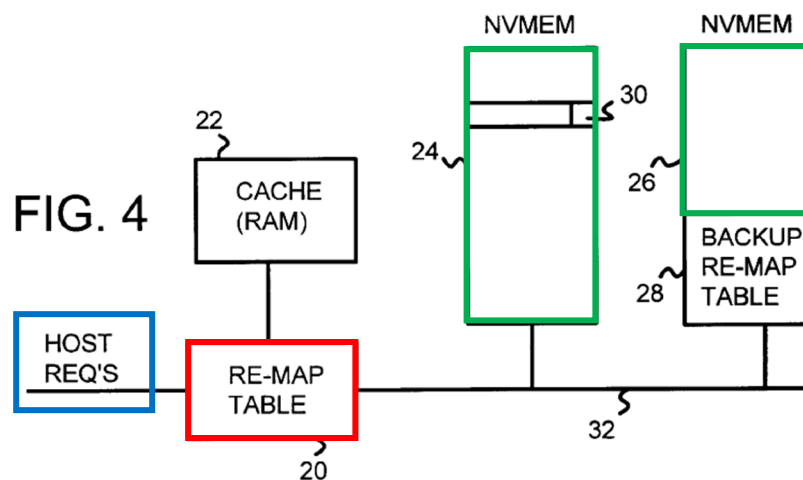
which would be equivalent to the size of a flash block. Similarly, by setting the integer multiplier used for determining the section size to be greater than the number of flash pages within a flash block, the size of each LBA set can be greater than the size of a flash block. Similarly, depending on how many LBAs are grouped into a LBA set, the integer multiplier used for determining the section size can be adjusted such that each claimed LBA size can be smaller, the same, or larger than a physical block.

IV. GROUND 1: CLAIMS 1 AND 14 ARE OBVIOUS OVER BRUCE '006

A. Overview of Bruce '006

79. Bruce '006 teaches a “more efficient and exact wear-leveling scheme” using a “unified table for re-mapping, wear-leveling, and caching flash memories.” Bruce '006 at 2:54-59. Bruce '006 recognizes that “EEPROM cells wear out as they are written and erased because some charge is trapped in the insulating oxide layer surrounding the floating gate. Eventually this trapped charge increases until it prevents an applied voltage from sufficiently reading the cell. Thus a scheme is needed to identify such defective memory cells and replace them with good cells.” Bruce '006 at 1:33-39. As I discuss above in ¶ 33, “wear out” was a well-known disadvantage of flash-memory technology before the Earliest Priority Date and techniques for dealing with it, such as wear-leveling, were commonplace.

80. Bruce '006 discloses a flash-memory system as shown in Figure 4 below. Bruce '006 states that the host requests (annotated in blue) “contain a logical address which includes a logical-block address (LBA) and an offset address that identifies a page within a flash block and perhaps a byte within a page for a read access.” Bruce '006 at 6:13-17. Bruce '006 further teaches that the “[f]lash memory is contained in several non-volatile (NV) flash memories 24, 26 [(annotated in green)], which can be individual flash-memory chips, or cards of flash-memory chips such as flash PC cards.” Bruce '006 at 6:38-41.



Bruce '006 at Figure 4.

81. For the re-map table (annotated in red), Bruce '006 teaches that it is “used to arbitrarily re-map all logical addresses from a host system to physical addresses of flash-memory devices.” Bruce '006 at Abstract, 6:19-20. As I discuss above in ¶¶ 34-52, address mapping was well known in the art before the

Earliest Priority Date, and Bruce '006 was one of many references that disclosed techniques for mapping between logical addresses in a host system and physical addresses in a flash storage system. Bruce '006 provides an exemplary re-map table, as shown in annotated Figure 7A below, with “7 entries (rows) ... for 7 sequential logical block addresses (LBA’s).” Bruce '006 at 8:3-5.

LBA	PBA	TOT_WR'S	INCR_WR'S
001	001	50,000	50,000
010	101	20,000	20,000
011	111	250,001	250,001
100	011	350,000	15,000
101	100	125,000	125,000
110	000	200,000	200,000
111	010	300,000	12,000

TOT_THRESH = 250,000
INCR_THRESH = 25,000

FIG. 7A

Bruce '006 at Figure 7A.

82. Bruce '006 explains that host requests containing a logical address which includes a logical-block address (LBA) (annotated in blue) “are first re-mapped by unified re-map table 20.” Bruce '006 at 6:12-17. Bruce '006 then explains that the re-map table includes a physical block address (PBA) (annotated in red) “for each possible logical block address.” Bruce '006 at 5:42-43.

83. Bruce '006 describes including in its re-map table “two write-counters for wear-leveling.” Bruce '006 at 3:15. Once an entry in the re-map table is determined to be ready for wear-leveling, “[t]he wear-leveling swap [] moves the data for the heavily-used LBA [] to the light-used PBA [], while the existing data in [lightly-used PBA] for [lightly-used LBA] is transferred to the cache and then to the heavily-written PBA [].” Bruce '006 at 8:29-39. As I discuss above in ¶¶ 51-52, wear-leveling was well known in the art before the Earliest Priority Date. Bruce '006's description of wear-leveling is consistent with the understanding that a POSITA would have had of that technique as of the Earliest Priority Date.

84. Bruce '006 also teaches that “[i]nterleaving can be combined with the invention.” Bruce '006 at 11:20. As I discuss above in ¶¶ 58-63 interleaving was well known in the art before the Earliest Priority Date. Bruce '006 explains that “[i]nterleaving is desirable since multiple accesses can occur in parallel to the flash devices, improving performance.” Bruce '006 at 11:20-22. Bruce '006 further recognizes that such performance improvement can result “when sequential blocks are accessed.” Bruce '006 at 11:39-40.

B. Independent Claim 1

85. It is my opinion that Bruce '006 discloses and renders obvious claim 1 of the '740 Patent.

1. Claim 1 Preamble

A mapping table for optimizing memory operations performed by an electronic storage device in response to receiving an I/O transaction request initiated by a host

86. I understand that language in a claim's preamble is generally assumed to be duplicative of the language found in the body of the claim and to merely provide context for the claim absent any indication to the contrary in the claim. To the extent this preamble is found to be limiting, it is my opinion that Bruce '006 discloses and renders it obvious.

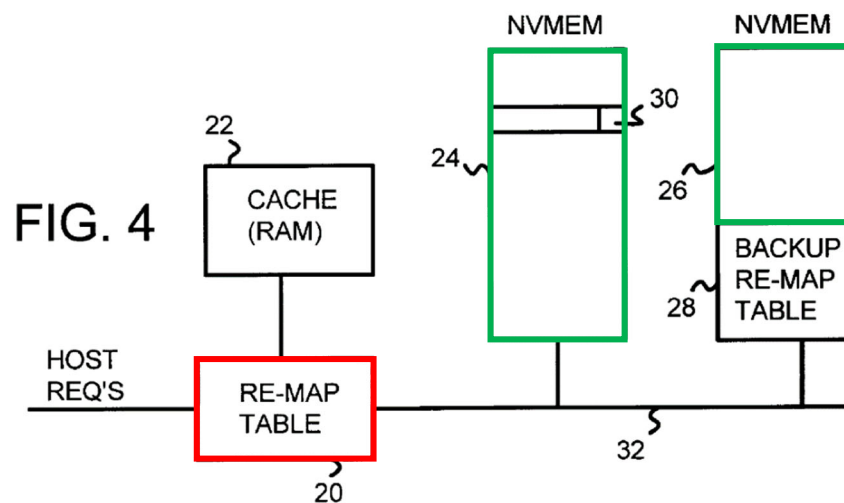
87. Bruce '006 is titled "Unified Re-Map and Cache-Index Table with Dual Write-Counters for Wear-Leveling of Non-Volatile Flash RAM Mass Storage." Bruce '006 at Cover Page. As the title indicates, Bruce '006 teaches a re-map table for a non-volatile flash RAM mass storage. For the reasons explained below, the re-map table in Bruce '006 corresponds to the claimed "*mapping table*."

88. Bruce '006 teaches that host requests for access to flash storage are first re-mapped by the re-map table. Bruce '006 at 5:15-17, 6:12-13, 12:40-43. The re-map table receives a logical address and provides the corresponding physical block address (PBA) of the data in flash memory. Bruce '006 at 5:33-35.

89. Bruce '006 describes the re-map table the same way the '740 Patent describes the claimed mapping table, namely, converting a logical address for an I/O transaction request initiated by a host to a physical address used to access a

storage device. '740 Patent at 6:37-44 (“[T]his host may use one or more LBAs in an I/O transaction request, such as I/O transaction request 20. Mapping table 9 associates LBAs from this host logical addressing system to the device addressing system used by storage device 2.”). As I discuss above in ¶¶ 34-52, mapping logical addresses from a host system to physical addresses in a flash-memory storage device was well known in the art before the Earliest Priority Date.

90. Bruce '006 further teaches using the re-map table in a flash-memory system. Bruce '006 at 2:62-63, 6:7-8. The flash-memory system taught in Bruce '006 is shown in annotated Figure 4 below:



Bruce '006 at Figure 4.

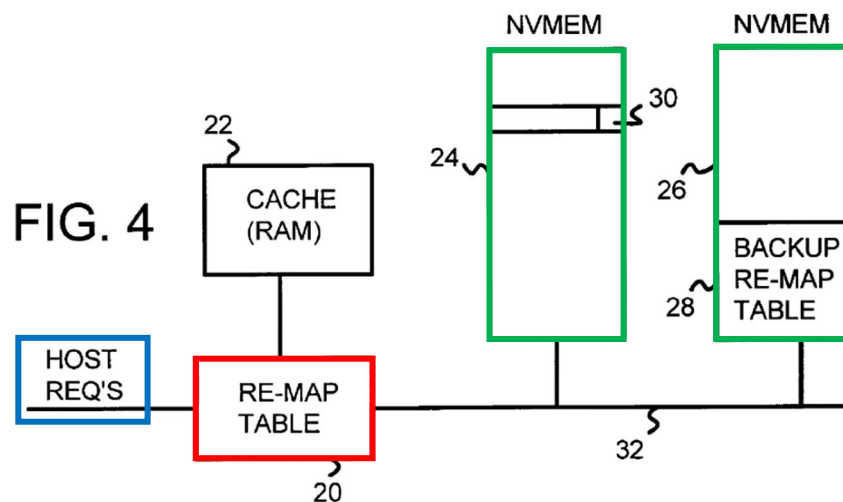
91. The re-map table is annotated in **red**. The flash-memory system is described as including “several non-volatile (NV) flash memories 24, 26” (annotated in **green**), “which can be individual flash-memory chips, or cards of

flash-memory chips such as flash PC cards.” Bruce ’006 at 6:38-41. Bruce ’006 describes its flash-memory system as providing “solid-state **storage** as a replacement to a hard disk.” Bruce ’006 at Abstract (emphasis added), 1:7-8 (“This invention relates to non-volatile **storage** system” (emphasis added)), 6:8-9 (“Such a system is useful as a mass-**storage** replacement to a hard disk on a personal computer.” (emphasis added)).

92. The ’740 Patent states that “[e]lectronic storage devices ... are sometimes referred to as solid state storage devices” and “employ a memory system that includes memory devices, memory chips, or modules that use non-volatile memory cells.” ’740 Patent at 1:28-32. The ’740 Patent provides as an example “solid state storage devices that use NAND flash memory, also referred to as ‘flash drives.’” ’740 Patent at 1:42-43. The Bruce ’006’s flash-memory system, which is a flash-storage device that comprises “flash-memory chips or cards of flash-memory chips,” is therefore an “*electronic storage device*.”

93. The flash-memory system in Bruce ’006 performs memory operations in response to “host requests,” including reads and writes. Bruce ’006 at 6:19-42, 8:51-52, 9:30-32. The host requests (annotated in **blue** in Figure 4 below) “contain a logical address which includes a logical-block address (LBA) and an offset address that identifies a page within a flash block and perhaps a byte within a page for a read access.” Bruce ’006 at 6:13-17. Bruce ’006 further teaches that the

LBAs are “**from** a user or **host** system.” Bruce ’006 at 5:40-41 (emphasis added). Bruce ’006 further teaches that its flash-memory system “is useful as a mass-storage replacement to a hard disk.” Bruce ’006 at 6:8-9. A POSITA would consider a hard disk to be an input/output device; hence, a POSITA would also consider Bruce ’006’s flash-memory system, which is taught to be a replacement for a hard disk, to be an input/output device. *See, e.g.*, Microsoft Press Computer Dictionary at 252 (“input/output device” is defined as “[a] piece of hardware that can be used both for providing data to a computer and for receiving data from it, depending on the current situation. A disk drive is an example of an input/output device. ...”) Thus, a POSITA would understand Bruce ’006’s host requests to correspond to the claimed input/output “I/O” transaction requests initiated by a host.



Bruce ’006 at Figure 4.

94. Bruce '006 goes on to teach that for each host request received by the flash-memory system, “[t]he entry for the current LBA is read from unified re-map table 20, and a cache-valid bit is checked to determine if the LBA’s data is in cache 22.” Bruce '006 at 6:12-13, 6:20-22, 7:26-31. “When the cache valid bit indicates that the data is not stored in cache 22, then a PBA field in the entry received from unified re-map table 20 is used **to access the data in flash memory.**” Bruce '006 at 6:35-38 (emphasis added), 7:28-33.

95. Bruce '006 thus teaches its flash-memory system performing *“memory operations ... in response to receiving I/O transaction request[s] initiated by a host.”* The Bruce '006’s description of performing memory operations in response to receiving host I/O transaction requests mirrors that of the '740 Patent. The '740 Patent describes its invention as “**optimizing memory operations performed** by or in an electronic storage device **in response to receiving an I/O transaction request initiated by a host[.]**” '740 Patent at 2:25-27 (emphasis added). The '740 Patent teaches that the LBAs in the host I/O transaction request are used by the mapping table to determine “which addressable physical memory locations to use **when performing memory operations**” requested by the host through the I/O transaction request. '740 Patent at 9:14-19 (emphasis added). Like the '740 Patent, Bruce '006 describes a flash-memory system that uses a re-map table to perform memory operations (“access[ing] the

data in flash memory”) in response to I/O transaction requests initiated by a host (“host requests”).

96. Bruce '006 also teaches that a **single** “*I/O transaction request initiated by a host*” could lead to **multiple** “*memory operations*” performed by the flash-memory system. Bruce '006 teaches this in the context of wear-leveling, for example.

97. Bruce '006 implements wear-leveling by including in the re-map table “two write-counters” associated with each entry in the table: total-write count (TOT_WR’S) and incremental-write count (INCR_WR’S). Bruce '006 at 3:4-8, 3:15, 3:41-49, Figures 7A-7E. Bruce '006 explains that “[e]ach time a block is written, its **total-write count** is compared to a total-write threshold, while the block’s **incremental-write count** is compared to an incremental-write threshold.” Bruce '006 at 4:65-5:5 (emphasis added). The total-write count represents the “total number of writes to the physical block since system manufacture.” Bruce '006 at 7:4-6. On the other hand, the incremental-write count represents the “number of writes since this PBA was allocated to the LBA, or since it was last wear-leveled,” and “is preferably cleared when physical flash blocks are re-assigned to a LBA, but the total-write count from total-write-counter field 46 is transferred over from the old entry to the entry for the newly allocated LBA.” Bruce '006 at 7:6-12.

98. A wear-leveling operation is triggered when: (1) “the total number from the entry in the unified re-mapping table exceeds the total-write threshold” and (2) “the incremental number from the entry in the unified re-mapping table exceeds the incremental-write threshold.” Bruce ’006 at 3:16-41.

99. Bruce ’006 illustrates in Figure 7A a wear-leveling operation for entry 90 (annotated in **red**) where the total-write threshold is 250,000 and the incremental-write threshold is 25,000 (annotated in **blue**). Bruce ’006 teaches that these threshold values can be stored in memory registers for use by the entire flash-memory system. Bruce ’006 at 3:18-19, 3:25-28. A POSITA would understand that the determination that entry 90 is ready for wear-leveling (Bruce ’006 at 8:29-32) is “in response to” a host’s write request to LBA 011 (annotated in **purple**) because the threshold comparisons are done “each time a block is written” (Bruce ’006 at 4:65):

LBA	PBA	TOT_WR'S	INCR_WR'S
001	001	50,000	50,000
010	101	20,000	20,000
011	111	250,001	250,001
100	011	350,000	15,000
101	100	125,000	125,000
110	000	200,000	200,000
111	010	300,000	12,000

90

TOT_THRESH = 250,000
INCR_THRESH = 25,000

FIG. 7A

Bruce '006 at Figure 7A.

100. Bruce '006 then describes a wear-leveling controller searching for a “least-used flash block,” which is “PBA 101, with 20,000 total writes.” Bruce '006 at 8:32-34. Because PBA 101 “is currently allocated to LBA 010,” the wear-leveler “moves the data for the heavily-used LBA 011 to the lightly-used PBA 101, while the existing data in PBA 101 for LBA 010 is transferred to the cache and then to the heavily-written PBA 111.” Bruce '006 at 8:35-39. Figure 7B below shows the revised mapping table “after the wear-leveling swap of LBA’s 010 and 011.” Bruce '006 at 8:40-41.

LBA	PBA	TOT_WR'S	INCR_WR'S
001	001	50,000	50,000
010	111	250,002	1
011	101	20,001	1
100	011	350,000	15,000
101	100	125,000	125,000
110	000	200,000	200,000
111	010	313,001	25,001

TOT_THRESH = 250,000
INCR_THRESH = 25,000

FIG. 7B

92

Bruce '006 at Figure 7B.

101. Bruce '006 thus teaches the flash-memory system performing at least **three memory operations** in response to the write request initiated by a host which triggered the wear-leveling example above: (1) a **read memory operation** to transfer the data from lightly-used PBA 101 to the cache, (2) a **write memory operation** to write the data from the heavily-used LBA 011 to the lightly-used PBA 101, and (3) a **write memory operation** to move the data from the cache to heavily-written PBA 111. Bruce '006 therefore discloses or renders obvious *“memory operations performed by an electronic storage device in response to receiving an I/O transaction request initiated by a host.”*

102. Bruce '006 also teaches that the re-map table is used by the flash-memory system for *“optimizing memory operations.”* Bruce '006 describes using its re-map table to interleave data across multiple flash memory. Specifically, the

block-interleaving scheme taught in Bruce '006 teaches that a sequence of four blocks (64 pages) is interleaved among four flash devices. Bruce '006 at 11:37-38. Interleaving is performed so that “multiple accesses can occur in parallel to the flash devices, improving performance.” Bruce '006 at 11:19-22, 11:37-40. Bruce '006 further states that performance increases from interleaving are especially pronounced “when sequential blocks are accessed.” Bruce '006 at 11:39-40.

103. The Figures 7A and 7B used in Bruce '006 to illustrate the wear-leveling operations described above demonstrate how a re-map table is used in Bruce '006 to optimize memory operations. Bruce '006 teaches that “[i]nterleaving can be combined with the invention.” Bruce '006 at 11:20. A POSITA reading the '740 Patent would understand this to mean that interleaving can be combined with the re-map tables taught in Bruce '006. For one, Bruce '006 specifically teaches that “[t]he **interleaving used** can be flexible by multiplexing different logical address bits **to the unified re-map table.**” Bruce '006 at 11:66-67 (emphasis added).

104. As I have explained above, Figure 7B in Bruce '006 shows the revised mapping table “after the wear-leveling swap of LBA’s 010 and 011.” Bruce '006 at 8:40-41. LBAs 010 and 011 are sequential LBAs. As part of Bruce '006 teaching on interleaving, Bruce '006 explains that “[p]erformance is improved when sequential blocks are accessed and the flash devices are slow.” Bruce '006 at

11:38-40. A POSITA would understand this to teach that for wear-level operations involving sequential LBAs, interleaving the corresponding memory operations among different flash devices would improve performance as it would increase the likelihood that those operations can be distributed across different resources and therefore proceed in parallel. Again, Bruce '006 teaches that “[i]nterleaving is desirable since multiple access can occur in parallel to the flash devices, improving performance.” Bruce '006 at 11:20-23. Bruce '006’s teaching of increasing the likelihood that multiple access can occur in parallel mirrors the '740 Patent’s description for optimizing memory operations. '740 Patent at 1:16-24 and 3:12-21.

105. Bruce '006 further teaches that for block-interleaving, “a sequence of four blocks are interleaved among four flash devices.” Bruce '006 at 11:37-38. Bruce '006 went on to provide the following bit assignment for the physical addresses when block-interleaving is used: Bruce '006 at 11:22-25.

3:0 Page within a block
5:4 Interleave among four devices
17:6 Block within a flash device
31:18 Device within a bank; bank within the system, etc.

106. When combining block-interleaving with the re-map table in Bruce '006, a POSITA would understand that the PBAs in the re-map table would include bits [31:18] of the physical addresses. This is because Bruce '006 teaches that the re-map table includes “a physical block address PBA field which ...

includes a portion of the address that identifies a flash memory chip or card, and another portion that identifies a block within a flash chip.” Bruce ’006 at 6:60-65. Based on this teaching, a POSITA would understand that bits [31:18] of the physical addresses used to identify a “[d]evice within a bank; bank within the system, etc.” should be included in the PBA field as well as bits [17:6] for identifying a “[b]lock with a flash device.” A POSITA would further understand that the PBAs in the re-map table would also include bits [5:4] of the physical addresses. This is because claim 9 of Bruce ’006 requires “each entry contains a physical-block-address for an interleave of at least four physical blocks, the at least four physical blocks being in at least four of the flash memory chips.”

107. Therefore, when block-interleaving is combined with the re-map table in Figure 7A, a POSITA would understand the lower 2 bits in the PBA fields to correspond to bits [5:4] of the physical addresses which are used for identifying “[i]nterleave among four devices” according to the bit assignment above. Therefore, the lower 2 bits of the PBA fields indicates, among the four interleaved flash devices, the flash memory device in which the data for the corresponding LBA is stored. Note, a POSITA would further understand from Bruce ’006’s teaching at 6:3-5 that there would be more than 3 bits for the PBA fields in an actual system so the PBA fields would have additional bits not shown in Fig. 71 corresponding to the rest of the bits [31:4] of the physical addresses as I have

explained above. A POSITA would understand the reason for using only 3 bits in the exemplary re-map tables is to simplify the presentation and discussion of the re-map tables, that is, simply for pedagogical reasons.

108. Therefore, for the wear-leveling operations described above, Fig. 7A shows that the write memory operation to write the data from the heavily-used LBA 011 to the lightly-used PBA 101 will be routed to flash device “01” which is the lower 2 bits of the PBA field 101. Similarly, the write memory operation to move the data from the cache to heavily-written PBA 111 will be routed to flash device “11” which is the lower 2 bits of the PBA field 111. A POSITA would understand that having the memory operations routed to different flash devices means they can be interleaved. Again, interleaving matches how the ’740 Patent describes optimizing memory operations.

109. In sum, Bruce ’006 teaches using its re-map table to interleave data across multiple flash devices so that “multiple accesses can occur in parallel to the flash devices, [thereby] improving performance,” which corresponds to the ’740 Patent’s description of “*optimizing memory operations.*” Bruce ’006 therefore teaches a “*mapping table for optimizing memory operations.*”

110. Bruce ’006’s disclosures of optimizing memory operations described above match the ’740 Patent’s disclosures. The ’740 Patent states that “optimizing memory operations includes increasing the likelihood that ... memory operations

will be optimally distributed across different storage device resources, such as by interleaving or parallel memory operations.” ’740 Patent at 2:14-19.

111. Thus, Bruce ’006 discloses and renders obvious the preamble of claim 1 of the ’740 Patent.

2. Claim 1, Limitation [A]

a set of logical fields, including a first logical field and a second logical field, and said logical fields respectively disposed for representing a plurality of LBA sets, including said first logical field disposed for representing a first LBA set and said second logical field disposed for representing a second LBA set, said first and second LBA sets each representing a set of consecutive LBAs

112. I will first note how certain terms in this limitation are defined in the ’740 Patent. The ’740 Patent defines “LBA” (logical block address) as “an address that is part of a logical addressing system ... used by a host.” ’740 Patent at 6:37-40. The ’740 Patent also defines “LBA sets” as “sets of consecutive LBAs,” in other words, sets of consecutive logical block addresses. ’740 Patent at 6:46-48. As shown in annotated Figure 2 below, the ’740 Patent shows a mapping table with logical fields annotated in **blue**, where “each logical field in set of logical fields 26 is respectively disposed to represent a LBA set (annotated in **green**).” ’740 Patent at 6:48-51. Therefore, for claim 1, the claimed first and second logical fields in a re-map table each represents a first and a second LBA set, and the first and second

LBA sets each represents a group of consecutive logical addresses in the host's addressing system.

Logical Fields 26	PBA Fields 28			
	Access Parameter Fields 30			Memory Index Fields 32
	Bus Id Fields 34	FDE Id Fields 36	Group Id Fields 38	
40 LBAs 00-07	8-0	22-01	15-0	0x00
42 LBAs 08-15	8-0	22-02	15-1	0x08
44 LBAs 16-23	8-1	22-11	15-0	0x00
45 LBAs 24-31	8-1	22-12	15-1	0x08
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮

FIG. 2

'740 Patent at Figure 2.

113. Bruce '006 uses somewhat different terminologies but a POSITA would understand Bruce '006 to disclose or render obvious a mapping table with a set of logical fields, each representing a claimed LBA set where each LBA set in turn represents a set of claimed consecutive LBAs.

114. Bruce '006 describes its re-map table as “contain[ing] entries preferably for all LBA's.” Bruce '006 at 6:19-20. Those LBAs are annotated in blue in annotated Figure 7A below, which Bruce '006 uses to demonstrate “examples of wear-leveling operations using dual write counters for total and incremental writes to flash blocks.” Bruce '006 at 4:31-33. Annotated Figure 7A

below “shows 7 entries (rows) in the unified re-map table for **7 sequential logical block addresses (LBA’s)**” that “are mapped to various physical blocks in flash memory by the PBA fields.” Bruce ’006 at 8:3-6 (emphasis added). The LBA column annotated in **blue** below thus corresponds to the claimed “set of logical fields,” including a first logical field in **orange** and a second logical field in **purple**:

LBA	PBA	TOT_WR'S	INCR_WR'S
001	001	50,000	50,000
010	101	20,000	20,000
011	111	250,001	250,001
100	011	350,000	15,000
101	100	125,000	125,000
110	000	200,000	200,000
111	010	300,000	12,000

90

TOT_THRESH = 250,000
INCR_THRESH = 25,000

FIG. 7A

Bruce ’006 at Figure 7A.

115. Bruce ’006 also teaches that each of the LBAs in the re-map table represents a set of logical addresses in the host’s addressing system. First, although Figure 7A depicts three address bits for the LBA and PBA entries, Bruce ’006 explains with regard to Figure 3—which also uses three address bits for the LBA fields, as well as the PBA fields—that “[i]n actual systems, the LBA’s and PBA’s have more address bits than shown . . . , such as 32 bits for the LBA and 32 bits for the PBA.” Bruce ’006 at 6:3-5. A POSITA would have understood the

same to be true of the other figures in Bruce '006 that use 3-bit addresses, including Figure 7A.

116. Bruce '006 then describes the host requests “from a user or host system” as including “a logical address which includes a logical-block address (LBA) and an **offset address that identifies a page within a flash block.**” Bruce '006 at 5:40-42, 6:12-16 (emphasis added). A POSITA would understand from Bruce '006's teaching of including an offset address in the logical address to identify a page within a flash block to mean that each logical address corresponds to one page within a flash block. In other words, each page is represented by the logical-block address (LBA) together with an offset address, both of which are contained in the host request and are part of a logical address in the host's addressing system. Bruce '006 then teaches that a flash block “is typically composed of 16 pages.” Bruce '006 at 5:64, 10:16-17. Therefore, each logical-block address (LBA) in Bruce '006 represents 16 consecutive pages within a flash block. Therefore, each LBA entry in the re-map table corresponds to and represents the logical addresses for a group of 16 consecutive pages that belong to the corresponding re-mapped physical blocks. As such, the LBAs taught in Bruce '006 corresponds to the claimed “LBA sets” in the '740 Patent.

117. Second, a POSITA would also understand from Bruce '006's teaching that “it is more efficient to keep **block-based** ... re-mapping information” to mean

that each LBA in the re-map table represents a set of 16 consecutive pages within a flash block. Bruce '006 at 5:61-62 (emphasis added). Bruce '006 explains that “**blocks** rather than **pages** are re-mapped by the unified re-map table.” Bruce '006 at 5:63-64 (emphasis added). As explained above, Bruce '006 further teaches that “[a] block is typically composed of 16 pages.” Bruce '006 at 5:64-65. Bruce '006 explains that mapping entire blocks is beneficial because “the block is the smallest unit for erase, while the page is the smallest unit for write,” and thus block-based mapping is “more efficient.” Bruce '006 at 5:59-65. As I discuss above in ¶¶ 34-52, by the Earliest Priority Date, address-mapping techniques had achieved a level of sophistication such that a POSITA would have known that it was beneficial to take system specifications into consideration when deciding the granularity at which to map addresses—for example, at the page level or at the block level. *E.g.*, Ryu at 768-769 (describing a similar block-level address mapping scheme where “the logical address is divided into a logical block address and a block offset, and only the logical block address is translated into a physical block address in flash memory” as an alternative to page-level address mapping where “a logical page can be mapped to any physical page in flash memory” thereby “requir[ing] a large amount of space to store the needed mapping table”). Bruce '006’s teaching of doing the re-mapping at the block-level further supports that each LBA entry in the re-map table represents the logical addresses for a group of 16 consecutive pages

that belong to the corresponding re-mapped physical block and that the LBAs taught in Bruce '006 corresponds to the claimed “LBA sets” in the '740 Patent.

118. Lastly, the block-interleaving scheme taught in Bruce '006 teaches that an LBA corresponds to 16 consecutive pages within a flash block. In block interleaving, Bruce '006 teaches that a sequence of four blocks (64 pages) is interleaved among four flash devices. Bruce '006 at 11:37-38. Block interleaving across four devices has the following bit assignment for the physical address:

3:0 Page within a block
5:4 Interleave among four devices
17:6 Block within a flash device
31:18 Device within a bank; bank within the system, etc.

Bruce '006 at 11:31-36. Bruce '006 explains that only bits 31 to 4 (annotated in **green**) “are routed to the re-map table [as the LBA] for block interleaving.” Bruce '006 at 11:67-12:6. Each entry in the re-map table thus corresponds to a single 16-page flash block, with the four bits in positions 3 to 0 of the host logical address (annotated in **red**) corresponding to a page within the flash block. This again supports that each LBA entry in the re-map table represents the logical addresses for a group of 16 consecutive pages that belong to the corresponding re-mapped physical blocks and that the LBAs taught in Bruce '006 corresponds to the claimed “LBA sets” in the '740 Patent.

119. In sum, with reference to annotated Figure 7A below, Bruce '006 includes in its re-map table a set of LBAs in the LBA column (annotated in **blue**) including a first LBA (annotated in **orange**) and a second LBA (annotated in **purple**). Each LBA in Bruce '006 corresponds to the claimed "LBA set" and each LBA in Bruce '006 is represented by a logical field in Bruce '006's re-map table.

LBA	PBA	TOT_WR'S	INCR_WR'S
001	001	50,000	50,000
010	101	20,000	20,000
011	111	250,001	250,001
100	011	350,000	15,000
101	100	125,000	125,000
110	000	200,000	200,000
111	010	300,000	12,000

90

TOT_THRESH = 250,000
INCR_THRESH = 25,000

FIG. 7A

Bruce '006 at Figure 7A.

120. The teachings of Bruce '006 also match the illustrative sizes that the '740 Patent provides for LBA (*i.e.*, "logical address" in Bruce '006) and LBA set (*i.e.*, group of logical addresses in Bruce '006). In Bruce '006, "[a] block is typically composed of 16 pages[, and a] page is usually 512 bytes in size." (Bruce '006 at 5:64-65.) With the block-based re-mapping of Bruce '006, the disclosed logical addresses (which correspond to the claimed LBAs) are for pages and each page would be 512 bytes in size. Thus, the Bruce '006 LBAs, which correspond to

the claimed LBA sets as I explain above, would be 8KB in size (*i.e.*, 512B per page x 16 pages per block = 8KB).

121. The '740 Patent suggests an exemplary LBA size of 512B and an LBA set associated with 8 LBAs, yielding an LBA set size of 4KB. '740 Patent at 7:22-25. The '740 Patent then explains that the LBA set size may be set to “a size that is **at least equal to the product of a positive integer multiplied by the native page size** used by a flash block.” '740 Patent at 7:35-38 (emphasis added). For example, “if the native page size ... is 4 KB, then the [LBA set size] that may be used would be limited to **4 KB, 8 KB, 12K, and so on.**” '740 Patent at 7:38-40 (emphasis added). A POSITA would therefore understand that the '740 Patent teaches greater than 8 LBAs (*e.g.*, 16 LBAs), in an LBA set, so long as the LBA set size is “at least equal to the product of a positive integer multiplied by the native page size used by a flash block.” Thus, Bruce '006 teaches an LBA set size of 8KB, which is within the range that the '740 Patent discloses.

122. For the reasons explained above, Bruce '006's logical addresses correspond to the claimed LBAs. And each LBA in the re-map table of Bruce '006 is respectively disposed to represent a set of 16 consecutive individually-addressable pages, which is a set of consecutive addresses that are part of a logical addressing system used by a host, and thus is an “LBA set” as that term is defined in the '740 Patent. And Bruce '006 teaches re-map tables with LBA entries that

correspond to the claimed “logical fields.” Bruce ’006 therefore teaches “*a set of logical fields, including a first logical field and a second logical field, and said logical fields respectively disposed for representing a plurality of LBA sets, including said first logical field disposed for representing a first LBA set and said second logical field disposed for representing a second LBA set, said first and second LBA sets each representing a set of consecutive LBAs.*”

123. Thus, Bruce ’006 discloses and renders obvious claim 1, limitation [A] of the ’740 Patent.

3. Claim 1, Limitation [B]

a set of PBA fields, including a first PBA field and a second PBA field, said set of PBA fields respectively disposed for representing a set of PBAs, including a first PBA disposed for representing a first set of access parameters and a second PBA disposed for representing a second set of access parameters, said PBAs each associated with a physical memory location in a memory store, said set of logical fields and said set of PBA fields disposed to associate said first and second LBA sets with said first and second PBAs; and

124. Bruce ’006 teaches “a set of PBA fields, including a first PBA field and a second PBA field, said set of PBA fields respectively disposed for representing a set of PBAs.” Referring again to Figure 7A below, Bruce ’006 teaches that “[e]ach entry in the unified re-map table contains a physical block address (PBA) of the flash memory allocated to the logical address.” Bruce ’006 at Abstract. The LBAs in the re-map table “are mapped to various physical blocks in

flash memory by the PBA fields.” Bruce ’006 at 8:5-6. Thus, the entries in the PBA column (annotated in **red**) correspond to the claimed “set of PBA fields,” which includes “a first PBA field” (annotated in **brown**) for representing a first PBA 101 and “a second PBA field” (annotated in **light blue**) for representing a second PBA 111.

LBA	PBA	TOT_WR'S	INCR_WR'S
001	001	50,000	50,000
010	101	20,000	20,000
011	111	250,001	250,001
100	011	350,000	15,000
101	100	125,000	125,000
110	000	200,000	200,000
111	010	300,000	12,000

90

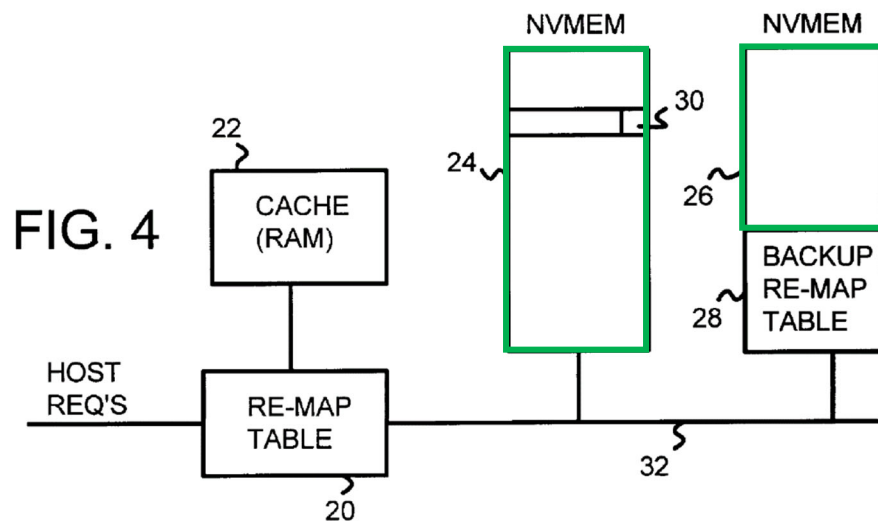
TOT_THRESH = 250,000
INCR_THRESH = 25,000

FIG. 7A

Bruce ’006 at Figure 7A.

125. Moreover, Bruce ’006 teaches that each of the PBAs in the re-map table is associated with a physical memory location in a memory store. As I explained in the context of the preamble, Figure 4 of Bruce ’006 below is a block diagram of a flash-memory system containing multiple non-volatile flash memories, such as “individual flash memory chips (in **green**).” Bruce ’006 at 6:7-8. The ’740 Patent teaches that “[m]emory store 6 may be configured to include a set of solid state memory devices ...” ’740 Patent at 4:54-55. The ’740 Patent explains that “these memory devices ... are implemented in the form of flash

memory devices 12-01 and 12-2 ...” and that “[t]he term ‘flash memory device’ is intended to include any form of non-volatile memory, including those that use blocks of non-volatile memory cells, named flash blocks.” ’740 Patent at 4:56-62. Thus, a POSITA would have understood the “several non-volatile (NV) flash memories 24, 26, which can be individual flash-memory chips, or cards of flash-memory chips such as flash PC cards” to correspond to the claimed “*memory store.*”



Bruce '006 at Figure 4.

126. Bruce '006 also teaches that the “PBA field in the entry retrieved from unified re-map table 20 is used to access the data in flash memory.” Bruce '006 at 6:36-38. The re-map table’s “physical block address PBA field contains the physical address of the flash memory device containing the data of the LBA.” Bruce '006 at 6:60-62. “This physical address includes a portion of the address

that identifies a flash memory chip or card, and another portion that identifies a block within a flash chip.” Bruce ’006 at 6:62-65. A POSITA would have thus understood that the PBAs disclosed in Bruce ’006 are “*each associated with a physical memory location in a memory store.*”

127. Bruce ’006 further discloses that “[a] single lookup of the re-map table for a logical address produces the physical block address (PBA) of the data in flash memory[.]” Bruce ’006 at 5:33-35. Thus, the first and second PBA fields (annotated in **brown** and **light blue**, respectively) are associated with corresponding first and second LBA sets (annotated in **orange** and **purple**, respectively):

LBA	PBA	TOT_WR'S	INCR_WR'S
001	001	50,000	50,000
010	101	20,000	20,000
011	111	250,001	250,001
100	011	350,000	15,000
101	100	125,000	125,000
110	000	200,000	200,000
111	010	300,000	12,000

90

TOT_THRESH = 250,000
INCR_THRESH = 25,000

FIG. 7A

Bruce ’006 at Figure 7A.

128. Bruce ’006 also teaches in connection with its block-interleaving scheme that each of the first and second PBAs is “*disposed for representing a ... set of access parameters.*” Bruce ’006 at 11:31-40, 11:67-12:2. Specifically,

Bruce '006 explains that for block interleaving, the physical address bits are assigned such that bits 31 to 18 denote the “device within a bank; bank within the system, etc.”; bits 17 to 6 denote the “block within a flash device”; and bits 5 to 4 denote the “interleave among [the] four devices.” Bruce '006 at 11:31-40.

129. The '740 Patent states that “access parameters” are parameters used to locate a particular flash-memory device within a flash-memory system:

Using their respective **access parameters**, PBA 46 is associated with a physical memory location that is **within a flash memory device 12-01 coupled to bus 8-0, controlled by FDE 22-01, and in group 15-0**, while PBA 50 is associated with another physical memory location that is **within a flash memory device 12-01 coupled to bus 8-1, controlled by FDE 22-11, and in group 15-0** in FIG. 1.

'740 Patent at 6:29-36 (emphasis added).

130. Bruce '006 teaches that when block-interleaving is used, “each entry contains a physical-block-address for an interleave of at least four physical blocks, the at least four physical blocks being in at least four of the flash memory chips.” Bruce '006 at 16:53-59, 17:6-12. Bruce '006 further teaches that each entry in the re-map table also “contains a physical block address PBA field which ... includes a portion of the address that identifies a flash memory chip or card, and another portion that identifies a block within a flash chip.” Bruce '006 at 6:60-65. These teachings would indicate to a POSITA that bits 31 to 4 of the physical addresses for block-interleaving are included in the PBAs in the re-map table.

131. A POSITA would further recognize that, at minimum, bits 31 to 18 (device within a bank; bank within the system, etc.) and bits 5 to 4 (interleave among [the] four devices) in the physical address in Bruce '006's block-interleaving scheme correspond to the access parameters described in the '740 Patent because they are also used to locate a particular flash memory chip within a flash-memory system based on their association in the re-map table with corresponding LBAs. Bruce '006 at 11:19-26, 11:31-40, 11:67-12:10. Thus, those bits that are part of the first and second PBAs described above are disposed to each represent a set of access parameters.

132. Bruce '006 therefore discloses *a set of PBA fields, including a first PBA field and a second PBA field, said set of PBA fields respectively disposed for representing a set of PBAs, including a first PBA disposed for representing a first set of access parameters and a second PBA disposed for representing a second set of access parameters, said PBAs each associated with a physical memory location in a memory store, said set of logical fields and said set of PBA fields disposed to associate said first and second LBA sets with said first and second PBAs.*

133. Thus, Bruce '006 discloses and renders obvious claim 1, limitation [B] of the '740 Patent.

4. Claim 1, Limitation [C]

wherein, in response to receiving the I/O transaction request, said mapping table causes the electronic storage device to perform optimized memory operations on memory locations respectively associated with said first PBA and said second PBA, if the I/O transaction request is associated with said first and second LBA sets.

134. As I discuss above in relation to claim 1's preamble, Bruce '006 teaches a re-map table that performs wear-leveling operations comprising multiple memory operations in response to a write request initiated by a host.

135. As I discuss above in ¶101, Bruce '006 shows a wear-leveling operation with reference to Figure 7A wherein a host write request to LBA 011 results in a wear-leveling sequence that requires at least three memory operations to complete: 1) a read memory operation to transfer the data from the lightly-used PBA 101 (annotated in **brown**) to the cache; 2) a write memory operation "move[] the data for the heavily-used LBA 011 to the lightly-used PBA 101 (annotated in **brown**); and 3) a write memory operation to move the data from cache to the heavily-written PBA 111 (annotated in **light blue**). Bruce '006 at 8:35-39.

LBA	PBA	TOT_WR'S	INCR_WR'S
001	001	50,000	50,000
010	101	20,000	20,000
011	111	250,001	250,001
100	011	350,000	15,000
101	100	125,000	125,000
110	000	200,000	200,000
111	010	300,000	12,000

90

TOT_THRESH = 250,000
INCR_THRESH = 25,000

FIG. 7A

Bruce '006 at Figure 7A.

136. Bruce '006 therefore teaches that the memory operations are performed in response to the host request on memory locations respectively associated with a first PBA (PBA 101, annotated in **brown**) and a second PBA (PBA 111, annotated in **light blue**). Bruce '006 further illustrates the revised mapping table “after the wear-leveling swap of LBA’s 010 (annotated in orange) and 011 (annotated in purple)” as shown in Figure 7B below.

LBA	PBA	TOT_WR'S	INCR_WR'S
001	001	50,000	50,000
010	111	250,002	1
011	101	20,001	1
100	011	350,000	15,000
101	100	125,000	125,000
110	000	200,000	200,000
111	010	313,001	25,001

92

TOT_THRESH = 250,000
INCR_THRESH = 25,000

FIG. 7B

As shown, Bruce '006 teaches a write request initiated by a host to LBA 011 that is associated with a first LBA set (LBA 010, annotated in **orange**) and a second LBA set (LBA 011, annotated in **purple**).

137. Moreover, as I explain above for claim 1, limitation [B], Bruce '006 teaches using the re-map table for block interleaving. “With such block-interleaving, a sequence of four blocks are interleaved among four flash devices,” and “[p]erformance is improved when sequential blocks are accessed and the flash devices are slow.” Bruce '006 at 11:38-40. In other words, Bruce '006 achieves improved performance through parallel reads and writes of blocks spread among four interleaved devices using the following bit assignment for the physical address:

3:0 Page within a block

5:4 Interleave among four devices

17:6 Block within a flash device

31:18 Device within a bank; bank within the system, etc.

Bruce '006 at 11:31-37. As I also explain above for claim 1, limitation [B], a POSITA would understand that in actual systems, more than three bits are used for addressing LBAs. Bruce '006 at 6:3-5.

138. When block interleaving is used as taught in Bruce '006, the lower two bits of the PBA fields correspond to bits 5 to 4 of the physical address bit assignment (annotated above in **blue**), which identify one of the four interleaved

flash devices that is associated with each LBA set. Bruce '006 at 11:67-12:2, 12:6-8. As a result, during a wear-leveling sequence like the one described in relation to Figure 7A, the write memory operation that moves the data from the cache to the heavily-written PBA (Bruce '006 at 8:37-39) would be routed to a flash device identified as "11" while the write memory operation to write the data from the heavily-used LBA to the lightly-used PBA would be routed to a different flash device such as "01." "With such block-interleaving, a sequence of four blocks are interleaved among four flash devices," and "[p]erformance is improved when sequential blocks are accessed and the flash devices are slow." Bruce '006 at 11:38-40. As shown in Figure 7B below, the wear-leveling operations described in Bruce '006 are in response to a write request initiated by a host that is associated with adjacent LBA sets 010 and 011. Given Bruce '006's teaching that interleaving sequential block accesses can lead to performance improvement by allowing parallel memory accesses, a POSITA would understand Bruce '006 to teach that interleaving the corresponding wear-leveling memory operations among different flash devices increases the likelihood that those operations can be distributed across different resources and proceed in parallel. Bruce '006 at 11:20-26, 15:53-59.

139. Bruce '006 teaches that by having the two memory operations directed at different flash devices, "multiple accesses [can] occur in parallel to the

flash devices, improving performance.” Bruce ’006 at 11:21-22. As I discuss above, this is precisely the type of optimized memory operation that the ’740 Patent teaches. ’740 Patent at 2:14-19. Moreover, Bruce ’006’s re-map table includes “dual write counters ... along with address translation information.” Bruce ’006 at 5:14-16. Bruce ’006 then explains that “[e]ach time a block is written, its total-write count is compared to a total-write threshold, while the block’s incremental-write count is compared to an incremental-write threshold.” Bruce ’006 at 4:53-5:5. When both the total-write count and incremental-write count exceed the system-wide thresholds, wear-leveling is performed. Bruce ’006 at 7:34-54. Thus, a POSITA would have understood that the re-map table in Bruce ’006 causes multiple memory operations to be performed in response to receiving an I/O transaction request initiated by a host, as required by this limitation.

140. Thus, Bruce ’006 discloses and renders obvious claim 1, limitation [C] of the ’740 Patent.

C. Dependent Claim 14

The mapping table of claim 1, wherein said first and second PBA respectively include a first group identifier and a second group identifier.

141. As I discuss above in ¶¶128-131 for claim 1, limitation [B], Bruce ’006’s re-map table includes *a first PBA disposed for representing a first set of access parameters and a second PBA disposed for representing a second set of*

access parameters. More specifically, Bruce '006 discloses a flash-memory system comprising several non-volatile flash memories, which can be individual flash-memory chips.” Bruce '006 at 6:38-41. Bruce '006 teaches using bits of the physical address (PBA) as access parameters for identifying an interleaved flash device and the “[d]evice within a bank, bank within the system, etc.” Bruce '006 at 11:27-37.

142. The fact that physical address bits are required to identify a “[d]evice within a bank” and a “bank within the system” would indicate to a POSITA that each bank includes multiple flash devices. Otherwise, there would be no need to identify a particular flash device within a bank and a bank within the system via the physical address bits when performing a memory operation. This understanding comports with Bruce '006’s broad disclosure of a general flash-memory system that includes several non-volatile flash memories “which can be individual flash-memory chips, or cards of flash-memory chips such as flash PC cards.” Bruce '006 at 6:38-41.

143. The '740 Patent teaches that a “group identifier” is used to identify a group of flash memory devices. '740 Patent at 6:11-21. A POSITA would have understood the teaching in Bruce '006 of flash-memory devices grouped “within a bank” within the flash-memory system (Bruce '006 at 11:27-37) to be a group identifier because those bits are similarly used to identify a group of flash-memory

chips in a bank. Bruce '006 therefore teaches the *mapping table of claim 1, wherein said first and second PBA respectively include a first group identifier and a second group identifier*, where the first and second group identifier are the portions of the physical address bits that identify the “[d]evice within a bank, bank within the system, etc.” for each of the *first and second PBA*, respectively.

144. Thus, Bruce '006 renders obvious claim 14 of the '740 Patent.

V. GROUND 2: CLAIMS 1 AND 14 ARE OBVIOUS OVER THE COMBINATION OF BRUCE '006 AND SBC

A. Overview of SBC

145. SBC relates to “SCSI” or the “Small Computer System Interface,” which is a standardized interface that allows an application running on a small computer system (e.g., a personal computer) to communicate with peripheral devices. Since its initial standardization in 1986, SCSI has evolved to operate with a multitude of interfaces and device, including flash memory. The first iteration of SCSI, known as SCSI-1, was standardized by ANSI in 1986 and introduced a parallel interface allowing for 8-bit data transfers and up to 8 devices on a single bus. SCSI-2, approved in 1994, brought improvements like faster data transfer rates, support for more devices (up to 16), and new “command sets,” which are collections of specified codes and coded values used to communicate requests and responses between a host and a SCSI device.

146. The SBC “standard defines the command set extensions to facilitate operation of SCSI block devices.” SBC at 1. A “block device” in the SBC standard is “[a] device that is capable of containing data stored in blocks that have a unique logical block address.” SBC at 3. A stated objective of the SBC standard is to “[p]ermit an application client to communicate with a logical unit that declares itself to be a direct-access device” SBC at 1. SBC identifies flash to be a typical variation of a direct-access block device. SBC at 15-16.

147. SBC defines numerous “commands unique to the type of SCSI block devices,” including READ(10) and WRITE(10) commands. SBC at 1, 40. The READ(10) command requests the device to “retrieve the blocks of data” to the host and includes the fields shown in Table 15 below. SBC at 7, 40.

Table 15 - READ(10) command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (28h)							
1	Reserved			DPO	FUA	Reserved		RELADR
2	(MSB)							
3								
4	LOGICAL BLOCK ADDRESS							
5	(LSB)							
6	Reserved							
7	(MSB)							
8	TRANSFER LENGTH (LSB)							
9	CONTROL							

SBC at Table 15.

148. Within the above command format, a “relative address (RELADR) bit of zero indicates that the LOGICAL BLOCK ADDRESS field specifies the first logical block of the range of logical blocks for this command.” SBC at 37, 41.

“The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred.” SBC at 41. “A (FUA) bit of one indicates that the device server shall access the media in performing the command prior to returning GOOD status. Read commands shall access the specified logical blocks from the media (i.e., the data is not directly retrieved from the cache).” SBC at 40.

149. The WRITE(10) command requests the device to “store the blocks of data” from the host and includes identical fields as the READ(10) command except that it uses a different OPERATION CODE. SBC at 7, Table 35.

Table 35 - WRITE(10) command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Ah)							
1	Reserved			DPO	FUA	Reserved	Reserved	RELADR
2	(MSB)							
3	LOGICAL BLOCK ADDRESS							
4								
5								
6	Reserved							(LSB)
7	(MSB)							
8	TRANSFER LENGTH							(LSB)
9	CONTROL							

SBC at Table 35.

B. Motivation to Combine Bruce '006 with SBC

150. A POSITA would have been motivated to combine SBC's disclosure of SCSI block commands with the flash-memory system of Bruce '006 and would have had the skills to make such a combination work.

151. SCSI is a 1998 ANSI standard that defines the SCSI command set extensions to “facilitate operation of SCSI block devices.” SBC at 1, 4, 6-7, 10. SBC defines a SCSI block device as a device that (i) conforms to the SBC standard and (ii) addresses data “in groups referred to as logical blocks.” SBC at 6. This logical block addressing is demonstrated by, for example, the READ(10) and WRITE(10) commands I discuss above in ¶¶ 147-149, both of which include a LOGICAL BLOCK ADDRESS field that specifies the first logical block of the range of logical blocks for the command, as well as a TRANSFER LENGTH field that specifies “the number of contiguous **logical blocks** of data that shall be transferred.” SBC at 40-41 (emphasis added), 56-57.

152. SBC further teaches that the SCSI block command set is designed for use with direct-access memory devices such as flash. SBC at 15-16. SBC further describes its command set as permitting “**efficient peer-to-peer operation by an OS by block transfer.**” SBC at viii (emphasis added). As I discuss above in ¶ 35, the compatibility issues between flash memories and existing OS programs based on the unique read/write/erase behaviors of flash memory were well known before the Earliest Priority Date. SBC’s ability to “efficient[ly]” mediate block transfers between the OS and flash memory would have appealed to a POSITA as a solution for this well-known compatibility problem.

153. Bruce '006 teaches a flash-memory system that receives host requests and explains that “[s]uch a system is useful as a mass-storage replacement to a hard disk on a personal computer[.]” Bruce '006 at 6:7-9; *see also* 1:12-15 (noting in the Background of the Invention section that “[o]ften the computer’s hard disk is replaced by an array of ... non-volatile flash memories”). Bruce '006 identifies as an example SCSI hard disks and further observes that flash memory and SCSI hard disks share the same page/sector size. Bruce '006 at 1:21-24, 5:64-65. As I discuss above in ¶ 39, flash-memory-based SCSI systems were known before the Earliest Priority Date. A POSITA would recognize Bruce '006’s mention of “match[ing]” page/sector size between flash memory and SCSI, together with its teaching of substituting hard disks with its flash-memory system, as an implicit acknowledgement that integrating different addressing schemes (*e.g.*, CHS and SCSI) was known.

154. A POSITA would have also recognized that Bruce '006 teaches a flash-memory system that receives I/O transaction requests for access to flash storage from a host system but does not specify an interface/format for the requests. Bruce '006 at Abstract, 2:62-64, 5:39-42, 6:7-13, 8:45-47, claims 1, 10, 17. A POSITA would have known that implementing the teachings of Bruce '006 as a functional system would require an interface between the host and flash-memory systems. The POSITA would have further found it beneficial to choose

an interface format that conforms with an industry-wide technical standard to maximize compatibility. A standard developed under the auspices of an organization such as ANSI, whose accreditation signifies that the procedures used by the standards-setting organization meet the institute's requirements for openness, balance, consensus, and due process, would be ideal. SBC is such a standard.

155. A POSITA would have recognized that the flash-memory system of Bruce '006 is a type of direct-access memory devices expressly described as supported in SBC. SBC at 6-7.

156. Bruce '006 also teaches using its inventive flash-memory system in place of an HDD such as a SCSI hard disk. Bruce '006 at Abstract, 6:7-11. A POSITA would have been aware of the need to use the SCSI format/interface for host requests in the modified system. As I discuss above in ¶ 34, the use of flash memory as an HDD replacement was commonplace long before the Earliest Priority Date. A POSITA would have therefore been familiar with the concept of replacing an HDD with a flash-memory drive, and further with the well-known need for ensuring that the flash-memory drive is able to communicate with the host system using the existing interface (*e.g.*, SCSI).

157. It was also well known by the Earliest Priority Date that SCSI storage devices implemented the SBC block commands. For example, Beardsley, a U.S.

patent titled “Method And System For Staging Data Into Cache,” filed in 1998 and originally assigned to IBM, teaches an embodiment that uses SCSI, and specifies that “SCSI commands are described in ‘Information Technology--SCSI-3 Block Commands (SBC),’ published by ANSI”:

In SCSI, a request to transfer data is accomplished with commands such as PRE-FETCH, READ(6), READ(10), and READ LONG. All these commands request a device server, e.g., the storage controller 8, to transfer data from a device, e.g., the DASD 6, to a client, e.g., the host systems 4a, b, c. These commands include a logical block address and transfer length fields. The logical block address provides the address of the first fixed block address in the data request. The transfer length specifies the number of contiguous logical blocks of data to be transferred starting from the logical block address. **SCSI commands are described in “Information Technology--SCSI-3 Block Commands (SBC),” published by ANSI, reference no. ANSI NCITS 306-199x (Rev. 8c, November 1997),** which publication is incorporated herein by reference in its entirety.

Beardsley at 4:48-63 (emphasis added). Beardsley was directed at direct access storage devices (“DASD”) and specifically teaches that a DASD “may be a string of hard disk drives or **other non-volatile storage devices ...**” Beardsley at 1:11-15 (emphasis added).

158. Combining SBC’s block command set with Bruce ’006’s flash-memory device would have thus required nothing more than use of a known technique to improve a known device in a known way. A POSITA would have known that SBC and SCSI are standards designed to be used together.

Implementing the host requests disclosed in Bruce ’006 using SBC’s block

commands would have thus been straightforward and resulted in achieving the benefit described in SBC, namely, an “efficient peer-to-peer operation by an OS by block transfer”—a predictable result. SBC at viii.

159. A POSITA would have had a reasonable expectation of success implementing SBC’s SCSI block commands in Bruce ’006’s flash-memory system. First, SBC expressly discloses that it is designed for use with block devices such as Bruce ’006’s flash-memory system. SBC at 1, 15-16. The teachings of SBC therefore show that by the Earliest Priority Date, performing I/O transactions between a host and flash-memory system was known in the art.

160. Second, the notion of performing I/O transactions between a host and a peripheral device using logical-block addressing was old in the art even by the Earliest Priority Date. For example, the first SCSI standard released in 1986 “use[d] logical rather than physical addressing for all data blocks.” SCSI-1986 at 12. Like SBC, the SCSI-1986 command set “assume[s] a data structure providing the appearance at the interface of **a contiguous set of logical blocks** of a fixed or explicitly defined data length.” SCSI-1986 at 51 (emphasis added), 53 (“**6.2 Command Descriptor Block**. A request to a peripheral device is performed by sending a command descriptor block to the target.”). The SCSI-1986 commands, like those in SBC (SBC at viii), specify a LOGICAL BLOCK ADDRESS:

Bit:	7	:	6	:	5	:	4	:	3	:	2	:	1	:	0	:	
Byte:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	
0	:	Operation Code														:	
1	:	Logical Unit Number				:	Reserved							:	RelAdr		:
2	:	Logical Block Address (if required) (MSB)														:	
3	:	Logical Block Address (if required)														:	
4	:	Logical Block Address (if required)														:	
5	:	Logical Block Address (if required) (LSB)														:	
6	:	Reserved														:	
7	:	Transfer Length (if required) (MSB)														:	
8	:	Transfer Length (if required) (LSB)														:	
9	:	Control Byte														:	

SCSI-1986 at 55. SCSI-1986, like SBC, further teaches use of a TRANSFER LENGTH that “indicates the number of blocks that shall be transferred”:

6.2.5 Transfer Length. The transfer length specifies the amount of data to be transferred, usually the number of blocks. For several commands the transfer length indicates the requested number of bytes to be sent as defined in the command description. For these commands the transfer length field may be identified by a different name. See the following descriptions and the individual command descriptions for further information.

Commands that use one byte for transfer length allow up to 256 blocks of data to be transferred by one command. A transfer length value of 1 to 255 indicates the number of blocks that shall transferred. A value of zero indicates 256 blocks.

Commands that use two bytes for transfer length allow up to 65,535 blocks of data to be transferred by one command. In this case, a transfer length of zero indicates that no data transfer shall take place. A value of 1 to 65,535 indicates the number of blocks that shall be transferred.

SCSI-1986 at 57. SBC’s command set would not have presented any challenges to a POSITA to incorporate into the system of Bruce ’006 because the techniques it uses were well known by the Earliest Priority Date.

161. Bruce ’006 requires that the host requests include a logical address but otherwise imposes no restriction on the protocol used. Bruce ’006 at 5:39-42,

6:12-17. SBC teaches host requests compatible with flash-memory systems that include a logical address. SBC is designed for use with SCSI systems and Bruce '006 teaches replacing a SCSI hard disk with its flash-memory system. And SCSI has taught the use of logical addressing since its inception, further cementing the compatibility of SBC with Bruce '006 and its use of a re-map table that relies on logical addressing. A POSITA would thus have recognized that there are no technical barriers needed to incorporate SBC's command set into Bruce '006's flash-memory system. Therefore, a POSITA would have had a reasonable expectation of success implementing SBC's SCSI block commands in Bruce '006's flash-memory system

C. Independent Claim 1

162. It is my opinion that Bruce '006 combined with SBC renders obvious claim 1 of the '740 Patent. As I explain above in Ground 1, Bruce '006 alone renders obvious the preamble of claim 1. However, Bruce '006 combined with SBC also renders obvious, in the manner described below, certain portions of claim 1.

1. Claim 1 Preamble

A mapping table for optimizing memory operations performed by an electronic storage device in response to receiving an I/O transaction request initiated by a host

163. As I explain above, a POSITA would have been motivated to combine Bruce '006 with SBC. To the extent this preamble is found to be limiting, as I discuss above in Ground 1, Bruce '006 discloses and renders obvious a *mapping table for optimizing memory operations performed by an electronic storage device in response to receiving an I/O transaction request initiated by a host*. As explained below, the combination of Bruce '006 and SBC also renders obvious optimizing **more than one** memory operation *performed by an electronic storage device in response to receiving an I/O transaction request initiated by a host*.

164. SBC's read and write commands specify a maximum TRANSFER LENGTH large enough that in the combined system of Bruce '006 and SBC, read/write commands would easily span two or more sequential LBAs in the re-map table and therefore cause **more than one memory operation** to be performed on memory locations associated with the PBAs mapped to the two or more sequential LBAs that were part of the host request. SBC teaches that a "host" can be "[a]ny combination of initiators and application clients that form a device managing one or more peripheral devices." SBC at 4. SBC "defines [] command set extensions to facilitate operation of SCSI block devices," which include flash memory storage devices that are "capable of containing data stored in blocks that have a unique logical block address." SBC at 1, 3, 15-16.

165. SBC provides both read and write commands. For direct-access block devices, SBC teaches that “[a]n application client issues ... WRITE commands to store the blocks of data (write operations) and READ commands to retrieve the blocks of data (read operations).” SBC at 7. SBC specifies the READ(10) command, which includes the fields shown in Table 15:

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (28h)							
1	Reserved			DPO	FUA	Reserved		RELADR
2	(MSB)							
3								
4	LOGICAL BLOCK ADDRESS							
5	(LSB)							
6	Reserved							
7	(MSB)							
8	TRANSFER LENGTH							
9	CONTROL							

SBC at 40.

166. SBC explains that setting the RELADR (relative address) bit (annotated in **orange**) to zero indicates that the LOGICAL BLOCK ADDRESS field (annotated in **green**) “specifies the first logical block of the range of logical blocks [of data] for this command.” SBC at 37. The TRANSFER LENGTH field (annotated in **blue**) “specifies the number of contiguous logical blocks of data that shall be transferred.” SBC at 41. Each logical block of data in the SBC specification is, for example, 512 bytes (512B) of data, which is often called a “page” of data.

167. The TRANSFER LENGTH field occupies bytes 7 and 8 (2 total bytes, or 16 bits) of the READ(10) command. The READ(10) command can therefore request a data transfer of up to 64K (*i.e.*, 2^{16}) “contiguous logical blocks of data.” SBC at 41. This follows from the fact that a 16-bit binary string can represent numerical values up to 2^{16} ; for example, the binary value 0110011100010100 is equal to 26,388 in decimal:

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	1	0	0	1	1	1	0	0	0	1	0	1	0	0

168. SBC’s 2-byte TRANSFER LENGTH field therefore permits a transfer of up to 64K times 512B (the size of a logical block) of data. SBC further teaches that when the FUA bit (annotated in **red**) is set to one, the data is read from “the specified logical blocks from the media,” meaning the data is not retrieved from the cache. SBC at 40. Put differently, when the FUA bit is set to one, the data must be retrieved from flash-memory storage, and not from the cache.

169. As I explain above in ¶¶ 120-121, Bruce ’006 teaches a re-map table where each entry maps a logical block address (LBA) to 16 consecutive 512-byte pages, meaning each entry in the re-map table corresponds to 8KB of data. Bruce ’006 at 5:59-65, 6:13-17. Bruce ’006 also teaches that the LBA entries of the re-map table are sequential. Bruce ’006 at 8:3-5. A single host READ(10) command that accesses as many as 64K contiguous logical blocks of data would therefore

traverse more than one row in the re-map table to identify the different physical blocks of flash memory where the requested data is stored. A WRITE(10) command with comparable transfer length would likewise require multiple memory operations to the flash blocks of the system taught in Bruce '006. In both instances, more than one memory operation is performed on memory locations associated with the PBAs mapped to the two or more sequential LBAs that were part of the host request.

170. Furthermore, Bruce '006 teaches a block-interleaving scheme where the re-map table maps LBAs to physical blocks across four flash devices. Bruce '006 at 11:31-40. A host READ(10) command that accesses more than 16 and as many as 64K contiguous logical blocks would, as discussed above, necessarily traverse multiple sequential rows of the re-map table taught in Bruce '006, thereby triggering “multiple accesses [to] occur in parallel to the flash devices.” Bruce '006 at 11:20-22. A WRITE(10) command with comparable transfer length would likewise require multiple memory operations to the flash blocks of the system taught in Bruce '006. As I discuss above in ¶¶ 58-63, interleaving was a known technique by the Earliest Priority Date for speeding up memory operations by executing memory operations concurrently and/or overlapped in time. A POSITA would readily understand that interleaving memory operations performed in response to a READ(10) or WRITE(10) command spanning multiple sequential

LBAs in Bruce '006 among different flash devices would increase the likelihood of distributing memory operations across different resources, thereby improving performance.

171. Thus, performing more than one optimized memory operation in response to receiving an I/O transaction request initiated by a host was obvious over Bruce '006 in combination with SBC. Bruce '006 combined with SBC therefore discloses and renders obvious the preamble of claim 1 of the '740 Patent. Other documentary evidence confirms that a POSITA would have known how to convert a single host command when the transfer size is larger than the size of an entry in the re-map table. For example, this is taught in Chou at Figure 15 and 11:16-60. Figure 15 “is a flow chart of a read method in accordance with the invention.” Chou at 3:15-16. Chou at 11:19-20 that “[i]n step 1505 the logical block address register 625 and the sector count register may be loaded from the CBW.” Chou goes on to teach at 11:31-33 that “[i]f the sector is available, then in step 1515 the write look up table 640 may be used to translate the logical block addresses to physical block addresses.” Then Chou at 11:42-45 that “[a]f translation, the resultant physical address may be used to read data from flash memory 550, the sector count decremented and the sector number incremented in step 1525.” Chou then teaches at 11:53-60 that “[i]n step 1560 it may be determined if the sector count is equal to zero. If sector count equals zero then

process 1500 ends. In step 1565 the block boundary may be checked. If the block boundary has been reached, the logical block address may be incremented in step 1570 and the offset bits set to zero is step 1575. Since CBW only has a starting address, all intermediate address will be generated internally.” So Chou taught receiving a host command that includes only the starting address which is stored in the logical block address register and a transfer length which is stored in the sector count register. Chou then teaches algorithms to generate multiple intermediate addresses based on the transfer length and using those multiple intermediate address and a lookup table to translate the logical block addresses to physical block addresses.

2. Claim 1, Limitation [A]

a set of logical fields, including a first logical field and a second logical field, and said logical fields respectively disposed for representing a plurality of LBA sets, including said first logical field disposed for representing a first LBA set and said second logical field disposed for representing a second LBA set, said first and second LBA sets each representing a set of consecutive LBAs

172. As I explain above in Ground 1, Bruce '006 alone discloses and renders this claim limitation obvious.

3. Claim 1, Limitation [B]

a set of PBA fields, including a first PBA field and a second PBA field, said set of PBA fields respectively disposed for representing a set of PBAs, including a first PBA disposed for representing a first set of access parameters and a second PBA

disposed for representing a second set of access parameters, said PBAs each associated with a physical memory location in a memory store, said set of logical fields and said set of PBA fields disposed to associate said first and second LBA sets with said first and second PBAs; and

173. As I explain above in Ground 1, Bruce '006 alone discloses and renders this claim limitation obvious.

4. Claim 1, Limitation [C]

wherein, in response to receiving the I/O transaction request, said mapping table causes the electronic storage device to perform optimized memory operations on memory locations respectively associated with said first PBA and said second PBA, if the I/O transaction request is associated with said first and second LBA sets.

174. As I explain above in Ground 1, Bruce '006 alone discloses and renders this claim limitation obvious. This claim limitation was also obvious over Bruce '006 in combination with SBC.

175. I first note that SBC's READ(10) and WRITE(10) commands use the LOGICAL BLOCK ADDRESS field to "specif[y] the logical block where the read [or write] operation shall begin." SBC at 39, 40-41, 56-57. Bruce '006, on the other hand, discloses a host's "logical address which includes a logical-block address (LBA) and an offset address that identifies a page within a flash block." Bruce '006 at 6:13-17. Bruce '006 further teaches that each LBA is mapped to a physical block in flash memory and that a physical flash block "is typically composed of 16 pages." Bruce '006 at 5:64-65, 8:5-6. Bruce '006's logical

address therefore corresponds to SBC's "logical block address" because they both refer to the host's logical address. It follows that an LBA in Bruce '006 corresponds to a sequence of 16 logical block addresses in SBC. SBC at 39, 41, 57; Bruce '006 at 6:13-17. I show this equivalence in the chart below:

Bruce '006	SBC
logical address	logical block address
logical-block address (LBA)	16 logical block addresses

176. Bruce '006 teaches that "[i]nterleaving is desirable since multiple accesses can occur in parallel to the flash devices, improving performance." Bruce '006 at 11:20-22. Bruce '006 further recognizes that such performance improvement is especially pronounced "when sequential blocks are accessed" to satisfy a host's I/O transaction request. Bruce '006 at 11:39-40. A POSITA would interpret "sequential blocks are accessed" broadly, in particular because Bruce '006 places no limitation on the number of sequential accesses, and because bulk data transfers (i.e., data transfers comprising many blocks of data) were common by the Earliest Priority Date. Indeed, as I discuss above in ¶¶ 53-57, large-capacity flash-memory systems in existence by the Earliest Priority Date would require memory operations capable of transferring large amount of data in order to function. Although Bruce '006 does not explicitly detail a host request that would require accessing two or more sequential LBAs that are mapped to two or more

interleaved PBA entries in the re-map table, it is clearly implied. Host requests of that kind were well known at the time, as demonstrated by SBC's READ(10) and WRITE(10) commands.

177. As I discuss above in ¶¶ 167-168, SBC teaches that a host may send a READ(10) or WRITE(10) command to a connected block device to retrieve or store up to 64K contiguous logical blocks of data (given that the transfer length field is 16 bits wide and $2^{16} = 64K$) where each logical block of data comprises 512 bytes. SBC at 40-41, 56-57. A READ(10) or WRITE(10) command with the RELADR (relative address) bit set to 0, the FUA (force unit access) bit set to 1, and the TRANSFER LENGTH set to greater than 16 logical block address units of data, would necessarily be mapped to two or more entries in Bruce '006's re-map table. A POSITA would realize that Bruce '006 teaches mapping sequential LBAs to different flash devices in the re-map table to support parallel access.

178. As I explain above in ¶ 116, Bruce '006 discloses a host request comprising a logical address which includes a logical-block address (LBA) and an offset address that identifies a page within a flash block. Bruce '006 at 5:40-42, 6:12-16. Bruce '006 further explains that a flash block typically contains 16 pages and suggests mapping each set of addresses—represented by the LBA portion of the logical address—to an entire flash block rather than mapping each host address to an individual page of a flash block. Bruce '006 at 5:62-65. Thus, it is the LBA

portion of the host logical address in Bruce '006 that is mapped to PBAs in the re-map table.

179. SBC, on the other hand, uses the term “logical block address,” to refer to the host’s logical address, which in Bruce '006 is both the LBA portion and the page offset. Bruce '006 at 6:14-16. Consequently, in the combined flash-memory system of SBC and Bruce '006, receiving a READ(10) or WRITE(10) command with a TRANSFER LENGTH of more than 16 pages—*i.e.*, more than one physical flash block in length—will result in a data transfer that exceeds a physical flash-block boundary, and will thus require multiple read or write memory operations directed to memory locations mapped to two or more PBAs in the re-map table.

180. In other words, in the combined system of Bruce '006 and SBC when block interleaving across four flash devices (per Bruce '006) is used and the flash-memory system receives a WRITE(10) command to store more than 16 pages of data (per SBC), each successive PBA in the re-map table would cycle through each of four flash devices in repeating order via bits 5 to 4 of the physical address:

3:0 Page within a block

5:4 Interleave among four devices

17:6 Block within a flash device

31:18 Device within a bank; bank within the system, etc.

Bruce '006 at 11:31-37. A POSITA would have understood that this behavior necessarily follows from the design of Bruce '006’s re-map table.

181. A POSITA would further realize that a similarly configured re-map table would be used in response to a READ(10) command in the combined flash-memory system of SBC and Bruce '006—in the combined system, reads would benefit from the same performance enhancement as writes. Thus, in response to receiving a “bulk” I/O transaction request (*i.e.*, a READ(10) or WRITE(10) command with TRANSFER LENGTH greater than 16), the combined system of Bruce '006 and SBC renders obvious a mapping table that causes the flash-memory system to perform optimized memory operations on memory locations respectively associated with two PBAs when the I/O transaction is associated with two LBA sets.

182. In summary, Bruce '006 in combination with SBC also renders this claim limitation obvious.

D. Dependent Claim 14

183. As I discuss above in Ground 1, Bruce '006 discloses “*wherein said first and second PBA respectively include a first group identifier and a second group identifier.*” Thus, as Bruce '006 in combination with SBC renders claim 1 obvious, it also renders this claim obvious.

VI. GROUND 3: CLAIMS 1, 9-15, 32, AND 34 ARE OBVIOUS OVER THE COMBINATION OF BRUCE '006 AND BRUCE '251

A. Overview of Bruce '251

184. Bruce '251, which is directed to an expandable flash-memory system, is in the same patent family as Bruce '006 and shares three of the same inventors.

Bruce '251 at Cover Page, Abstract, 1:9-13. As shown in annotated **Figure 2**

below, Bruce '251 teaches “an expandable flash-disk system using dual DMA and dual flash busses.” Bruce '251 at 5:52-53. More specifically, Bruce '251 teaches a

flash-memory system comprising banks of flash-memory chips 20 (annotated in

purple) connected to flash

buffer chips 14 (annotated in

orange), which are connected

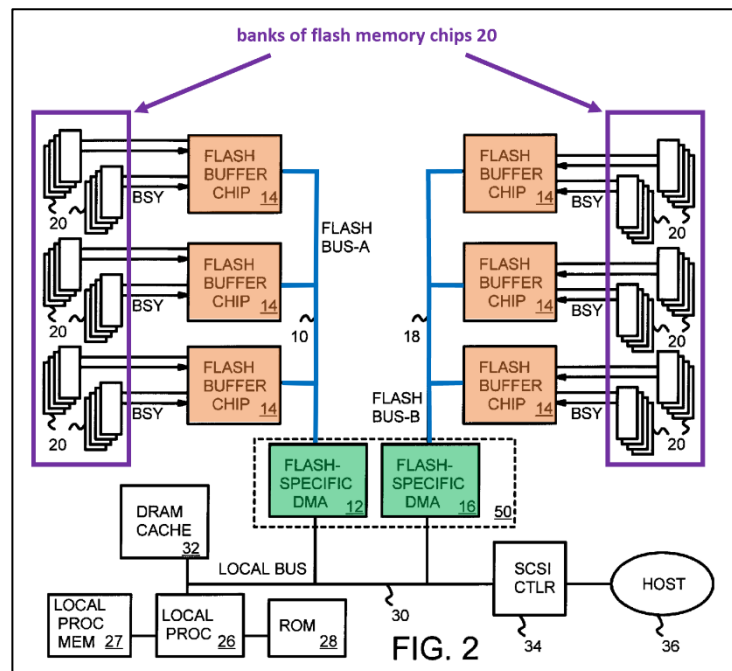
to flash busses 10 and 18

(annotated in blue). Bruce

'251 at 5:53-55. Bruce '251

explains that “[a] great many

flash-memory chips can be



added since each flash buffer chip can connect to four banks of 8 flash-memory

chips, a total of 32 flash-memory chips per flash buffer chip.” Bruce '251 at 5:58-

61. Bruce '251 further teaches connecting the dual flash busses to the local bus 30

via dual flash-specific DMA controllers 12 and 16 (annotated in green). The

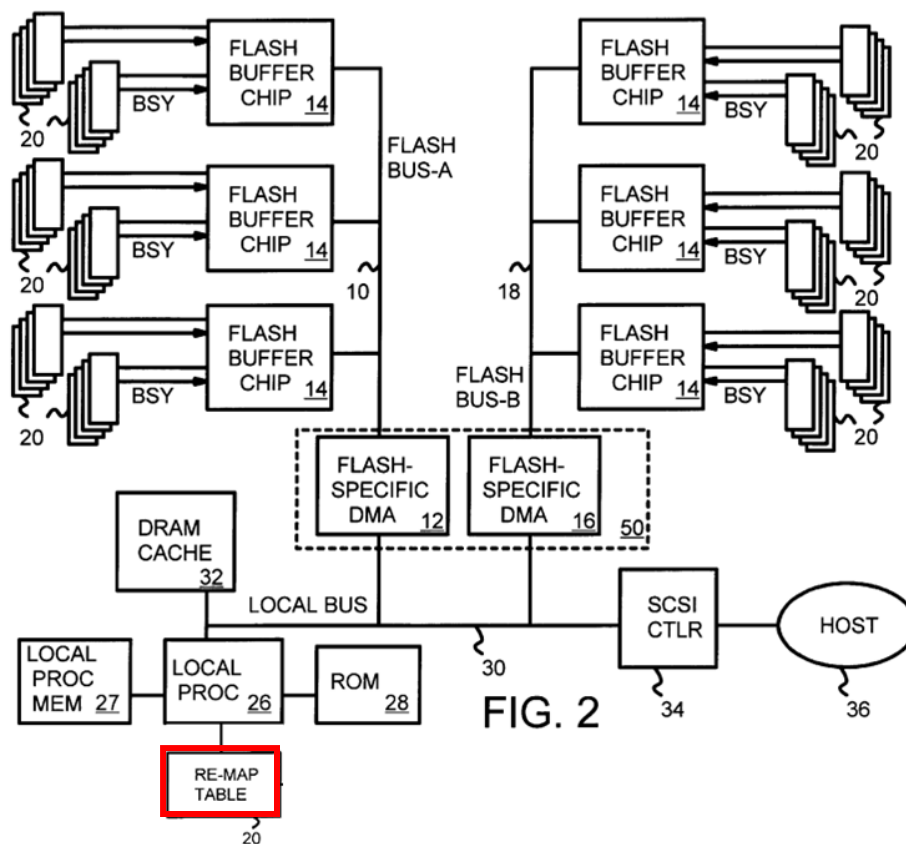
DMA controllers perform memory operations by “sending command and address bytes to the flash-memory chips through the flash buffer chips.” Bruce ’251 at 6:13-22.

185. Bruce ’251 explains that each of the DMA controllers 12 and 16 “contains four state machines 40, one for each of four banks of flash-memory chips,” thereby allowing “four flash-memory chips to be accessed at once for each flash bus, for a total of eight flash accesses.” Bruce ’251 at 6:53-57. The data in the flash-memory system of Bruce ’251 can therefore “be interleaved among the four banks for each flash buffer chip, and also interleaved among chips connected to the two flash busses.” Bruce ’251 at 6:57-60. Bruce ’251 also teaches that “additional flash busses can be added along with additional DMA controllers” that can “be dynamically assigned to flash busses or to interleaves rather than fixed.” Bruce ’251 at 14:10-12.

186. The expandable flash-disk system of Bruce ’251 includes a local processor 26 “that controls the flash-memory storage peripheral.” Bruce ’251 at 3:49-50, 5:65-6:5. Local processor 26 is coupled to ROM 28 that “contains firmware code of routines that execute on local processor 26 to respond to host requests[and o]ther system-maintenance routines ..., such as wear-leveling and copy-back routines.” Bruce ’251 at 6:5-9.

B. Motivation to Combine Bruce '006 with Bruce '251

187. A POSITA would have been motivated to combine the unified re-map table taught in Bruce '006 (highlighted red) with the expandable flash-memory system taught in Bruce '251, as illustrated in this annotated Figure 2 from Bruce '251.



188. Bruce '006 describes various advantages that its unified re-map table provides over other techniques for wear-leveling and data re-mapping in a flash-memory system, and those advantages would have been apparent to a POSITA. POSITAs would therefore have been motivated to utilize Bruce '006's unified re-map table in Bruce '251's expandable flash-memory system, thereby realizing the

benefits Bruce '006's unified re-map table provides for the wear-leveling and data re-mapping routines disclosed by Bruce '251 as aspects of its flash-memory system.

189. Bruce '251 describes that its expandable flash-memory system includes a local processor for executing firmware routines to perform maintenance, including "wear-leveling routines for re-mapping data from over-used or faulty memory blocks under-used or unused memory blocks." Bruce '251 at 3:48-55, 5:51-6:12, 16:59-64. Bruce '006 also addresses wear-leveling in a flash-memory system, as it discloses a unified re-map table for a flash-memory system that stores "dual write counters ... along with address translation information" for determining when wear-leveling is required. Bruce '006 at 2:60-3:14, 4:53-5:20, 7:22-44, 7:58-63.

190. According to Bruce '006, the disclosed unified re-map table provides certain advantages and benefits, including providing a "more efficient and exact wear-leveling scheme [that] minimizes excess writes to flash-memory while re-mapping addresses to pages of flash-memory." Bruce '006 at 12:11-15. Bruce '006 describes that the dual write counters include "[a] total-write counter [that] allows the wear-leveling firmware to determine which blocks have had the most writes and are most likely to wear out," and "[an] incremental-write counter [that] prevents thrashing, where a block is continually swapped back and forth during

successive wear-leveling operations.” Bruce ‘006 at 4:43-61, 5:3-20, 12:22-24, 12:26-29. By storing the dual write counters in the unified re-map table, along with address translation information, “[a] single lookup of the re-map table for a logical address produces the physical block address (PBA) of the data in flash-memory [and] the two write-counter values for wear-leveling,” eliminating the need for separate table accesses to obtain address translation information and wear-leveling information. Bruce ‘006 at 12:44-47, 6:6-18. Bruce ‘006 explains that “[t]his one-step re-mapping is faster for bad pages than the prior-art technique of mechanical mapping followed by reading re-map pointers of bad pages.” Bruce ‘006 at 5:18-20. Since Bruce ‘251’s expandable flash-memory system also performs wear-leveling and re-mapping, and therefore would be improved by the benefits suggested in Bruce ‘006, POSITAs would have been motivated by the benefits expressly suggested in Bruce ‘006 to incorporate Bruce ‘006’s unified re-map table into Bruce ‘251’s system.

191. A POSITA would further be motivated to look to Bruce ‘006 for the implementation details it provides for performing wear-leveling and re-mapping, which Bruce ‘251 lacks. As discussed above, Bruce ‘251 describes its expandable flash-memory system as performing various “routines” including “wear-leveling routines for re-mapping data.” Bruce ‘251 at 3:51-55, 6:5-9, claim 10. However, Bruce ‘251 does not specify **how** the “wear-leveling routines for re-mapping data”

should be performed. POSITA would have been motivated to turn to Bruce '006 for these missing implementation details, and in fact Bruce '251 suggests doing so. Bruce '251 is a continuation-in-part of Bruce '006 and shares three of the same named inventors. Bruce '251 mentions Bruce '006 at the beginning of the specification:

1

EXPANDABLE FLASH-MEMORY MASS-STORAGE USING SHARED BUDDY LINES AND INTERMEDIATE FLASH-BUS BETWEEN DEVICE-SPECIFIC BUFFERS AND FLASH-INTELLIGENT DMA CONTROLLERS

RELATED APPLICATION

This application is a continuation-in-part of the co-pending application for “Unified Re-Map and Cache-Index Table with Dual Write-Counters for Wear-Leveling of Non-Volatile Flash RAM Mass Storage”, filed Aug. 25, 1997, U.S. Ser. No. 08/918,203.

Bruce '251 at 1:9-13.

192. A POSITA reading Bruce '251 would therefore want to know how to implement wear-leveling routines and data re-mapping in the expandable flash-memory system of Bruce '251. The POSITA would notice the prominent citation to Bruce '006 at the beginning of the Bruce '251 specification, in particular the title: “**Unified Re-Map** and Cache-Index Table with Dual Write-Counters for **Wear-Leveling** of Non-Volatile Flash RAM Mass Storage.” Bruce '251 at 1:9-13 (emphasis added). The relevance of Bruce '006 (based on its relationship to Bruce

'251, its title, etc.) would not be a lost on a POSITA, who would be compelled to investigate further.

193. The POSITA would quickly realize that the unified re-map table referenced in the title of Bruce '006 provides an efficient and compatible method for implementing wear-leveling routines for re-mapping data in the system of Bruce '251. In fact, the POSITA would recognize Bruce '006's use of a unified re-map table as a known technique that would improve the expandable flash-memory system of Bruce '251 in a predictable way.

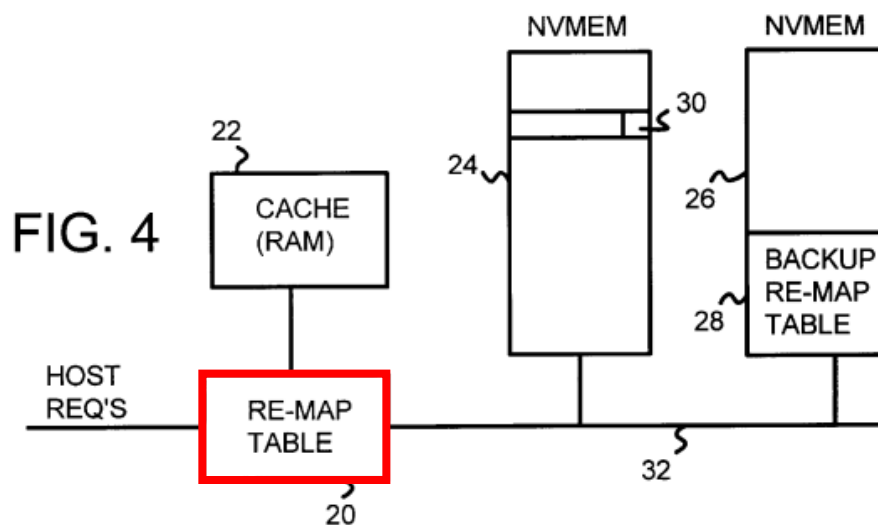
194. As I discuss above in ¶¶34-52, by the Earliest Priority Date it was well known to use a mapping table like the one in Bruce '006 to perform address translation in a flash-memory system, including for wear leveling. A POSITA would have thus been aware that wear-leveling a flash-memory system requires maintaining information about the logical-to-physical address translation and the amount of "wear" of the different physical locations in the system. *Supra* ¶¶ 51-52.

195. The prior art makes these requirements clear. Assar, a 1995 U.S. patent titled "Flash memory mass storage architecture incorporation **wear leveling technique**," teaches "a map to directly correlate a logical address of a block to a physical address of that block and a count register for each block." Assar at Abstract. Assar explains that its wear-leveling technique "would ordinarily not be

possible when using conventional mass storage because the central processor and commercial software available in conventional computer systems are not configured to track continually changing physical locations of data files.” Assar at 2:55-60. To that end, Assar “includes a programmable map to maintain a correlation between the logical address 308 and the physical address 408 of the updated information files.” Assar at 2:60-63. Assar describes using a “look-up” table to store this “address map,” which comprises “a mapping of the logical address 308, i.e., the address where the computer system believes the data file is stored to the physical address 408, i.e., the actual location the data file can be found is stored in the mass storage” and is “required” for the invention to accomplish wear leveling. Assar at 4:4-15.

196. A POSITA would therefore understand based on Bruce '251 and their general knowledge that logical-to-physical mapping information and wear-level information would need to be maintained in the system. Bruce '006 maintains both in its re-map table. Bruce '006 discloses a flash-memory system with a unified re-map table that stores “dual write counters ... along with address translation information.” Bruce '006 at 5:13-15. “All flash accesses are first routed through the unified re-map table to determine the physical block address of the data in flash memory. Bruce '006 explains that “[t]his one-step re-mapping is faster for bad pages than the prior-art technique of mechanical mapping followed

by reading re-map pointers of bad pages.” Bruce ’006 at 5:15-20. In other words, as shown in annotated Figure 4 below, Bruce ’006 describes a unified re-map table (annotated in **red**) that stores wear-leveling information (*e.g.*, dual write counters) and the logical-to-physical address translation. Bruce ’006 further teaches that the re-map table is used by “wear-leveling firmware” to “determine which blocks have had the most writes and are most likely to wear out” and “search for a replacement block that has both a low number of total writes and a low number of recent (incremental) writes.” Bruce ’006 at 5:3-12, 7:48-51, 8:45-60.



Bruce ’006 at Figure 4.

197. Bruce ’006 therefore maintains in its re-map table information regarding logical-to-physical address translation and the amount of “wear” relating to different physical locations in the system, which is what a POSITA would have understood the system of Bruce ’251 to require.

198. The re-map table of Bruce '006 is also compatible with the system of Bruce '251. First, both references describe conventional components. Bruce '006 at 6:7-50, Figure 4. Figure 2 of Bruce '251 shows that the expandable flash-memory system comprises banks of flash-memory chips, flash buffer chips, flash busses, flash-specific DMA controllers, a bus, a DRAM cache, a local processor, and a SCSI controller. Bruce '251 at 5:51-67. Bruce '251 does not describe any of these components as anything other than conventional, and I agree that they are conventional. As I discuss above in ¶¶ 53-57, by the Earliest Priority Date large-capacity flash-memory storage systems comprising multiple chip arrays, busses, and DMA engines were known. A POSITA would have recognized that the components described in the expandable flash-memory system of Bruce '251 are conventional and operate as expected. Bruce '251 at 6:1-13:15.

199. The systems of Bruce '251 and Bruce '006 would operate the same in combination as they do individually. Bruce '251 explains that its expandable flash-memory system includes a local processor with local memory. Bruce '251 at 5:51-6:12. Bruce '251 teaches that the local processor controls the flash-memory system and executes firmware code routines to respond to host requests or perform maintenance operations. Bruce '251 at 3:49-50, 6:5-9. Bruce '251 further teaches that the maintenance routines include wear-leveling routines for re-mapping data. Bruce '251 at 3:53-54, 6:5-9, claim 10. The data are mapped to locations in flash-

memory chips that are organized into banks and accessed via flash buffer chips, flash busses, and flash-specific DMA controllers. Bruce '251 at 5:51-67.

200. Bruce '006, in turn, discloses a unified re-map table with PBA fields each “containing the physical address of the flash memory device containing the data of the [corresponding] LBA.” Bruce '006 at 6:61-62. Bruce '006 further discloses that “[t]his physical address includes **a portion of the address that identifies a flash memory chip or card ...**.” Bruce '006 at 6:62-65, 11:27-46 (teaching to use portions of the physical address bits for identifying “[d]evices **within a bank; bank[s] within [a] system, etc.**” (emphasis added)). Bruce '006 additionally discloses at 5:16-18 that “[a]ll flash accesses are first routed through the unified re-map table to determine the physical block address of the data in flash memory,” where “[a] single lookup of the re-map table for a logical address produces the physical block address (PBA) of the data in flash memory [and] the write-counts for wear-leveling.” Bruce '006 at 5:33-35.

201. Thus, Bruce '006 discloses the use of a re-map table in a flash-memory system for logical-to-physical address translation where portions of the physical addresses in the re-map table are used to identify a specific flash-memory chip in a flash memory system. A POSITA would have understood that the Bruce '006 disclosure of using portions of the physical address bits for identifying “[d]evices within a bank; bank[s] within [a] system, etc.” is directly applicable to

the expandable flash-memory system of Bruce '251, which contains multiple DMA controllers, flash busses, and flash buffer chips used to access flash-memory chips. In the combined system, the physical address bits would simply associate with a specific flash-bus identifier, flash-specific DMA controller identifier, and a bank identifier, in the same way that is described in Bruce '006.

202. Doing so would have been within the knowledge of a POSITA. It was known by the Earliest Priority Date to use address bits to identify a location of a physical memory address in a flash-memory system. Chang at 4 (“In our multi-bank storage system, each entry in the table is a triple (*bank_num*, *block_num*, *page_num*), indexed by LBA. Such a triple indicates that the corresponding logical block resides at Page *page_num* of Block *block_num* of Bank *bank_num*.”); Huffman at 33 (“When using interleaving, the lowest order bits of the block address may be modified.”). A POSITA would have thus recognized that Bruce '006's use of physical address bits in the unified re-map table to identify flash memory chips would have predictably enabled identifying flash-buses, flash-specific DMA controllers, and banks of flash memory chips in the system of Bruce '251.

203. A POSITA would have had a reasonable expectation of success incorporating the unified re-map table taught in Bruce '006 into the expandable flash-memory system taught in Bruce '251. Bruce '251 specifically discusses

performing wear-leveling to re-map data in the disclosed expandable flash-memory system. Bruce '251 at 3:53-54, 6:5-9, claim 10. As I discuss above in ¶¶ 34-52, the use of mapping tables to maintain information regarding logical-to-physical address translations, thereby enabling wear-leveling, was also well known in the prior art. Moreover, Bruce '006 demonstrated that POSITAs knew how to implement a re-map table for wear-leveling and data re-mapping in flash-memory systems, including in those similar to the expandable flash-memory system disclosed by Bruce '251. Bruce '006 at 2:60-3:14, 4:53-5:20. Both Bruce '006 and Bruce '251 describe implementing wear-leveling through firmware routines, further demonstrating the compatibility of Bruce '006's unified re-map table in Bruce '251's expandable flash-memory system. Bruce '006 at 5:3-12, 7:48-51, 8:45-60; Bruce '251 at 3:48-55, 6:5-9. And Bruce '006 described that its unified re-map table was stored in SRAM. Bruce '006 at 6:43-44. POSITAs would have recognized that the ability to utilize Bruce '251's local processor's local memory 27, DRAM cache 32, or other memories internal to the local processor 26, for similarly storing the unified re-map table integrated into Bruce '251's system. Bruce '251 at 6:8-12.

204. Furthermore, storing identifiers in such tables to identify specific flash-memory chips and related physical components was similarly known in the prior art. Stallmo, a U.S. patent issued in 1996 titled "Logical Partitioning of a

Redundant Array Storage System,” is one example. Stallmo describes a storage system that includes “a set of physical storage units which are accessible in terms of block numbers.” Stallmo at Abstract. Each physical storage unit is configured as one or more “logical storage units” that are addressed in terms of (1) a channel number (*e.g.*, a bus among multiple buses), (2) a storage unit number (*e.g.*, a numbered physical storage unit along the bus identified by the channel number), and (3) a controller number. Stallmo at Abstract, 7:38-8:9, Figure 1, Figure 2A. A POSITA would have thus recognized that there are no technical barriers to surmount or modifications to be made to incorporate the unified re-map table of Bruce ’006 into the expandable flash-memory system Bruce ’251.

C. Independent Claim 1

1. Claim 1 Preamble

A mapping table for optimizing memory operations performed by an electronic storage device in response to receiving an I/O transaction request initiated by a host

205. To the extent this preamble is found to be limiting, it is my opinion that Bruce ’251 teaches an “*electronic storage device*.” Bruce ’251 describes its invention as relating to non-volatile memory storage systems, and more particularly to “expansion of flash-memory systems.” Bruce ’251 at 1:18-20, 16:34-17:35, 18:28-59, 19:13-45. As I discuss above in ¶ 184, Bruce ’251 teaches an expandable flash-memory system comprising banks of flash-memory chips

connected to flash busses via flash buffer chips. Bruce '251 at 5:53-55. In the combined system of Bruce '251 and Bruce '006, the re-map table of Bruce '006 resides in Bruce '251's expandable flash-memory system. The '740 Patent provides as an example of an electronic storage device "solid state storage devices that use NAND flash memory, also referred to as 'flash drives,'" and further specifies that they may "employ a memory system that includes memory devices, memory chips, or modules that use non-volatile memory cells." '740 Patent at 1:28-43. A POSITA would understand the combined Bruce '006-'251 flash-memory system to be a "flash drive," and further that the combined system comprises "memory devices, memory chips, or modules that use non-volatile memory cells." The combined system therefore corresponds to the claimed "*electronic storage device.*"

206. Bruce '251 teaches "*I/O [input/output] transaction request[s] initiated by a host.*" Bruce '251 describes its expandable flash-memory system as a "peripheral." Bruce '251 at 3:48-50, 16:54. The expandable flash-memory system "receives requests for access of flash-memory" through a "host interface" and in response "generates a sequence of command bytes followed by address bytes to initiate a data transfer from the flash-memory." Bruce '251 at 2:49-55, 6:1-7, 6:13-19. Bruce '251 further teaches that "flash-memory systems are frequently used as a mass-storage replacement to a hard disk on a personal

computer.” Bruce ’251 at 1:31-34. A POSITA would understand the host read/write requests in Bruce ’251 to be “*I/O [input/output] transaction request[s] initiated by a host*” because they are read/write (I/O) requests from a host for data stored in flash memory. I note that the ’740 Patent merely shows the I/O transaction request as an input to the electronic storage device illustrated in Figure 1 but does not include any specifics or implementation details. ’740 Patent at 9:26-27, Figure 1. Bruce ’251’s host read/write requests match the minimal disclosure of “*I/O [input/output] transaction request[s] initiated by a host*” in the ’740 Patent.

207. As I discuss in detail above in Ground 1, Bruce ’006 discloses and renders obvious “[a] *mapping table for optimizing memory operations*” in response to a host request that triggers wear-leveling operations. In the combined Bruce ’006-’251 system, the re-map table of Bruce ’006 performs memory operations in response to a host-initiated I/O transaction request. When a request triggers wear-leveling based on the counters in Bruce ’006’s re-map table, the table causes the combined system to perform optimized memory operations for the same reasons I describe above in Ground 1, ¶¶ 96-101, 114, and 135-136. Thus, to the extent this preamble is found to be limiting, Bruce ’006 combined with Bruce ’251 renders obvious the preamble of claim 1.

2. Claim 1, Limitations [A]-[C]

208. As I explain above in Ground 1, Bruce '006 alone discloses and renders these claim limitations obvious.

D. Dependent Claim 9

The mapping table of claim 1, wherein said set of access parameters includes a bus identifier, an FDE identifier and a group identifier.

209. As I explain above for claim 1, limitation [B] in Ground 1, the re-map table of Bruce '006 includes “a first PBA disposed for representing a first set of access parameters and a second PBA disposed for representing a second set of access parameters.” As I explain above for claim 14, Bruce '006 also discloses that “said set of access parameters includes a ...group identifier.”

210. Bruce '006 does not, however, explicitly disclose that said set of access parameters include a “bus identifier” or “an FDE identifier.” As I explain above, it would have been obvious to combine the teaching of Bruce '006 with the teaching of an expandable flash-memory system in Bruce '251. In that combination, the “set of access parameters includes a bus identifier[and]an FDE identifier.”

211. Bruce '251 discloses that “[r]ather than directly connecting individual flash-memory chips to a controller, flash buffer chips are used.” Bruce '251 at Abstract. “[E]ach flash buffer chip can connect to four banks of 8 flash-memory

chips.” Bruce ’251 at 5:59-60. Bruce ’251 therefore discloses grouping eight flash-memory chips into one bank and connecting four banks of memory devices to each flash buffer chip. Connecting groups of flash-memory devices into banks was a well-known technique before the Earliest Priority Date. As just one example, Chang similarly teaches a NAND flash storage system comprising multiple NAND flash-memory chips organized into banks. Chang at 3, Figure 2.

212. Bruce ’251 further describes flash busses 10 and 18, each of “which have 8 multiplexed data/address/command lines.” Bruce ’251 at 5:7-17. The flash busses use an “8-bit address/data/command signal for [sending] commands, addresses, and data to the flash-memory chips.” Bruce ’251 at 9:52-54. Bits 1 to 0 of the 8-bit address/data/command signal are used to identify one of the four banks of memory devices associated with each memory operation, as shown in the annotated table below:

bit 7	normal = 1, initialization = 0
bits 5:4	00 = data, 01 = command, 10 = address, 11 = disconnect
bit 3	1 = lock sequence
bits 1:0	Interleave selection, one of four banks

Bruce ’251 at 10:12-15. The ’740 Patent teaches that the claimed *group identifier* identifies a group of flash-memory devices. ’740 Patent at 6:11-15. In the combined system of Bruce ’006 and Bruce ’251, the re-map table of Bruce ’006 uses the two

“interleave selection” bits taught in Bruce ’251 as the bits in the PBAs for identifying the “bank within the system” taught in Bruce ’006. Bruce ’006 at 11:30, 11:36, 11:46. Like the ’740 Patent’s *group identifier*, the bits are used to identify a group of flash-memory devices, namely, the eight flash-memory devices residing on one of the four banks of memory devices that can be associated with any given memory operation. As such, Bruce ’006 in combination with Bruce ’251 also renders obvious “*wherein said set of access parameters includes ... a group identifier.*”

213. Bruce ’251 further teaches “an expandable flash-disk system using ... **dual flash busses.**” Bruce ’251 at 5:52-53 (emphasis added). More specifically, Bruce ’251 teaches having “[b]anks of flash-memory chips 20 connect to flash buffer chips 14, which are connected to **flash busses 10, 18.**” Bruce ’251 at 5:53-55 (emphasis added). The ’740 Patent similarly teaches using a “*bus identifier*” as “an identifier that is used for representing a bus interface, such as **bus 8-0 or 8-1.**” ’740 Patent at 5:39-41 (emphasis added). A POSITA would have understood flash busses 10 and 18 in Bruce ’251 to be equivalent to busses 8-0 and 8-1 in the ’740 Patent: they are flash busses for accessing flash-memory devices in their respective systems. A POSITA would thus understand the combined system of Bruce ’251 and Bruce ’006 to include a *bus identifier* that identifies the bus used to access each flash-memory device pursuant to the portions of the physical address

bits that identify the “[d]evices within a bank; bank[s] within [a] system, etc.” in Bruce ’006. Bruce ’006 at 11:27-46.

214. Bruce ’251 also discloses connecting multiple flash buffer chips to “[t]wo flash-specific DMA controllers.” Bruce ’251 at Abstract. More specifically, Bruce ’251 teaches:

Accesses of flash-memory chips 20 by local processor 26 are initiated by local processor 26 sending a high-level command to one of flash-specific DMA controllers 12, 16. These DMA controllers then generate sequences of command and address bytes, and transfer data.

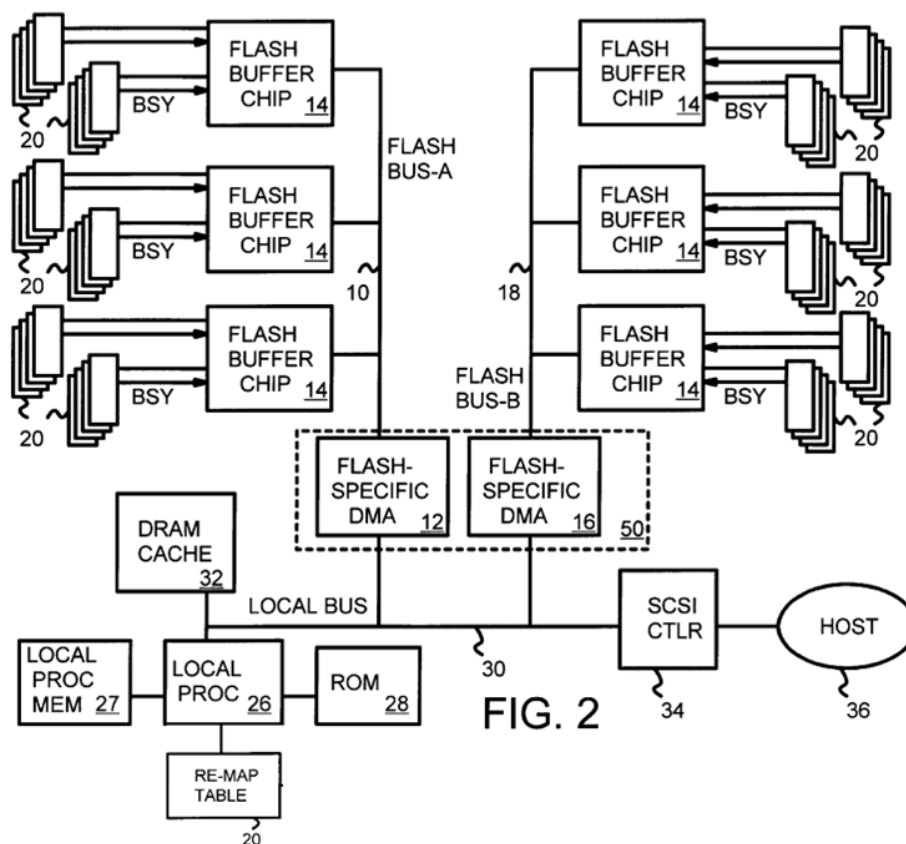
Bruce ’251 at 6:13-17. In other words, Bruce ’251 discloses the use of two flash-specific DMA controllers for accessing different flash-memory chips in its expandable flash-memory system.

215. The ’740 Patent describes “[a]n FDE [to be] a device that is capable of controlling a flash memory device **and performing DMA operations on the flash memory device[.]**” ’740 Patent at 3:35-38 (emphasis added). A POSITA would have understood that the flash-specific DMA controllers disclosed in Bruce ’251 correspond to the claimed FDEs in the ’740 Patent because like the FDEs described in the ’740 Patent, the flash-specific DMA controllers in Bruce ’251 “generate[] command, address, and data sequences” to perform DMA operations on the flash-memory devices. Bruce ’251 at 3:58-4:21 (“A first flash-specific DMA controller is coupled to the local processor. It generates command, address,

and data sequences to a first flash-memory chip in a format required by the first flash-memory chip. ... A second flash-specific DMA controller is coupled to the local processor. It generates command, address, and data sequences to a second flash-memory chip in a format required by the second flash-memory chip. ... Thus two flash-specific DMA controllers control access of flash-memory chips through flash buffer chips connected by two flash buses.”). A POSITA would thus understand the combined system of Bruce ’251 and Bruce ’006 to include an *FDE identifier* that identifies the flash-specific DMA controller used to access each flash-memory device pursuant to the portions of the physical address bits that identify the “[d]evices within a bank; bank[s] within [a] system, etc.” in Bruce ’006. Bruce ’006 at 11:27-46.

216. I note that Bruce ’251 also teaches alternative embodiments in which “DMA controllers could be dynamically assigned to flash busses or to interleaves rather than fixed.” Bruce ’251 at 14:10-12. “Dynamic” I/O refers to a communication method where addresses and configurations of devices on a bus are determined at runtime. This means that devices can be added, removed, or replaced, and the system will automatically discover their addresses and configure them accordingly. Bruce ’251 therefore teaches assigning DMA controllers to flash busses/interleaves without the need for manual reconfiguration. A POSITA would understand from the above disclosure that Bruce ’251 teaches that both the

flash-specific DMA controllers 12 and 16 in Figure 2 below may be connected to flash bus-A 10 and flash bus-B 18. A POSITA would further understand that Bruce '251 teaches that flash-specific DMA controller 12 can be dynamically configured to communicate via flash bus-A 10 **or** flash bus-B 18, and that flash-specific DMA controller 16 can also dynamically communicate via flash bus-A 10 **or** flash bus-B 18.



Bruce '251 at Figure 2.

217. Since there are more than one DMA controller 12,16 on local bus 30, and since more than one flash bus can be dynamically assigned to each DMA

controller (as I discussed above), a POSITA would understand that both the DMA controller number (*i.e.*, 12 or 16) **and** the flash bus number (*i.e.*, 10 or 18) must be used to identify a particular flash-buffer chip. Because the relationship between DMA controllers/flash busses is no longer fixed, but can vary due to dynamic reassignment, a POSITA would understand that in the combined system of Bruce '006 and Bruce '251, in addition to the 2-bit “*group identifier*” I discuss above in ¶ 212, additional physical address bits are needed (e.g., in the high-level command sent by local processor 26 over local bus 30) to identify the DMA controller and flash bus associated with a given flash-memory chip. Moreover, it was well-known to use multiple identifier bits to identify a device. For example, Stallmo described using both a Controller Number (used to identify a controller) and a Channel Number (used to identify a channel or bus), in order to access a storage unit on the identified bus accessible by the identified controller. Stallmo at 4:56-60, 7:44-48, 5-9. Accordingly, the POSITA would have recognized using bits to identify a DMA controller, and bits to identify a bus, as part of a command. A POSITA would further understand that Bruce '006 in combination with Bruce '251 renders obvious “*wherein said set of access parameters includes a bus identifier, an FDE identifier and a group identifier.*”

E. Dependent Claim 10

The mapping table of claim 9, wherein said first and second PBA include different bus identifiers, different FDE identifiers and different group identifiers.

218. As I explain above for claim 9, Bruce '251 teaches an expandable flash-memory system where different locations in physical memory can be accessed via different DMA controllers, flash busses, and banks. As I further explain above for claim 9, in the combined system of Bruce '006 and Bruce '251, each PBA has a corresponding *bus identifier*, *FDE identifier*, and *group identifier* for identifying the specific flash-memory chip to use in a memory operation.

219. I note that claim 10 does not require “*said first and second PBA*” to be any specific entries in a mapping table; they can correspond to any two entries in a mapping table. A POSITA would understand that any two locations in physical memory that are accessed via different DMA controllers, flash busses, and banks would accordingly have different bus identifiers, FDE identifiers, and group identifiers. Thus, a POSITA would understand that any such pair of PBAs in the re-map table of the combined system of Bruce '006 and Bruce '251 satisfy claim 10. Thus, Bruce '006 in combination with Bruce '251 renders obvious “*wherein said first and second PBA include different bus identifiers, different FDE identifiers and different group identifiers.*”

F. Dependent Claim 11

The mapping table of claim 9, wherein said first and second PBA respectively include different combinations of said bus identifiers, said FDE identifiers, and said group identifiers.

220. I note that claim 11 is substantially identical to claim 10, which requires “*wherein said first and second PBA include **different** bus identifiers, **different** FDE identifiers and **different** group identifiers.*” For the same reasons I discuss above for claim 10, Bruce ’006 in combination with Bruce ’251 renders claim 11 obvious.

G. Dependent Claim 12

The mapping table of claim 1, wherein said first and second PBA include different bus identifiers.

221. The recited claim limitation here is a part of claim 10 which requires “*wherein said first and second PBA include different bus identifiers, different FDE identifiers and different group identifiers.*” For the same reasons I discuss above for claim 10, Bruce ’006 in combination with Bruce ’251 renders claim 12 obvious.

H. Dependent Claim 13

The mapping table of claim 1, wherein said first and second PBA include different FDE identifiers.

222. The recited claim limitation here is a part of claim 10 which requires “*wherein said first and second PBA include different bus identifiers, **different** FDE identifiers and different group identifiers.*” For the same reasons I discuss

above for claim 10, Bruce '006 in combination with Bruce '251 renders claim 13 obvious.

I. Dependent Claim 14

The mapping table of claim 1, wherein said first and second PBA respectively include a first group identifier and a second group identifier.

223. I note that the recited claim limitation here is broader than the claim limitation in claim 10 requiring “*said first and second PBA include ... different group identifiers.*” For the same reasons described for claim 10, Bruce '006 in combination with Bruce '251 renders claim 14 obvious.

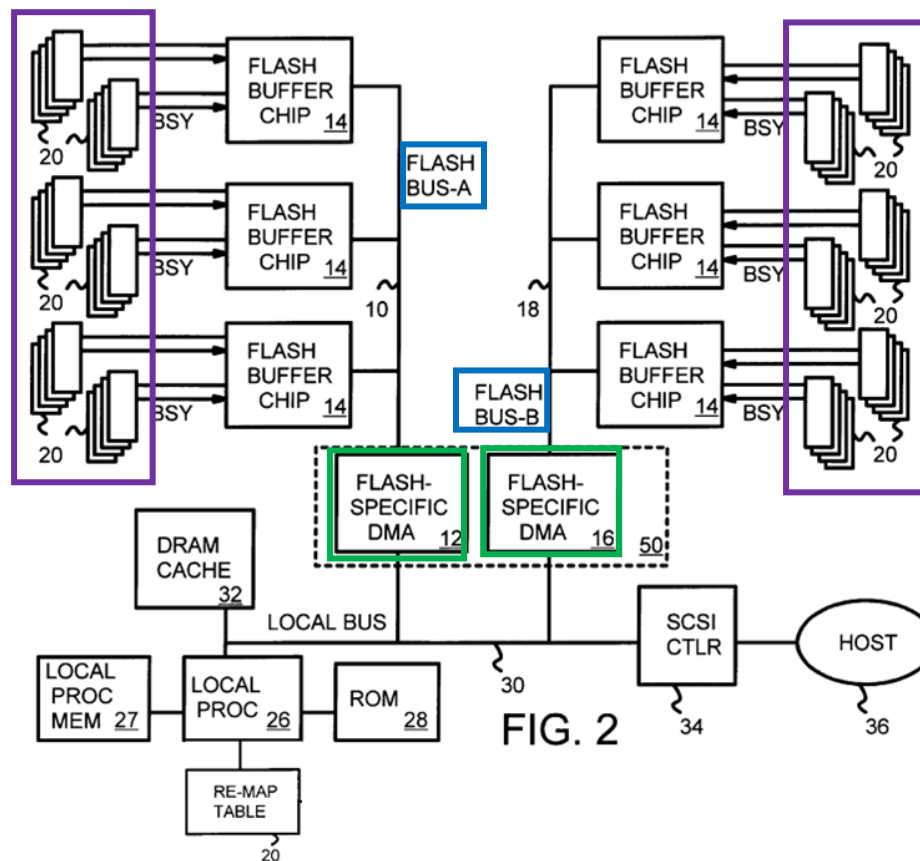
J. Dependent Claim 15

1. Claim 15, Limitation [A]

The mapping table of claim 1: wherein said memory store includes a set of solid state memory devices that are coupled to a set of buses and that are controlled by a set of FDEs.

224. As I explain above in ¶ 184, Bruce '251 teaches an expandable flash-memory system with “[b]anks of flash-memory chips 20.” Bruce '251 at 5:53. A POSITA would understand the flash-memory chips taught in Bruce '251 to be “solid state memory devices.” Bruce '251 at Abstract (“A flash-memory system provides solid-state mass storage.”). Thus, Bruce '006 in combination with Bruce '251 renders obvious “[t]he mapping table of claim 1: wherein said memory store includes a set of solid state memory devices.”

225. Bruce '251 further discloses “an expandable flash-disk system using **dual DMA and dual flash busses.**” Bruce '251 at 5:52-53 (emphasis added). As shown in the annotated Figure 2 below, the flash-memory chips (annotated in **purple**) are coupled to flash busses 10 and 18 (annotated in **blue**) and flash-specific DMA controllers 12 and 16 (annotated in **green**).



Bruce '251 at Figure 2.

226. Bruce '251 also explains that “**additional flash busses** can be added along with **additional DMA controllers.**” Bruce '251 at 14:9-10 (emphasis added). The DMA controllers control the flash-memory chips by “sending

command and address bytes to the flash-memory chips through the flash buffer chips.” Bruce ’251 at 6:13-22. Thus, Bruce ’006 in combination with Bruce ’251 renders obvious “*wherein said memory store includes a set of solid state memory devices that are coupled to a set of buses and that are controlled by a set of FDEs.*”

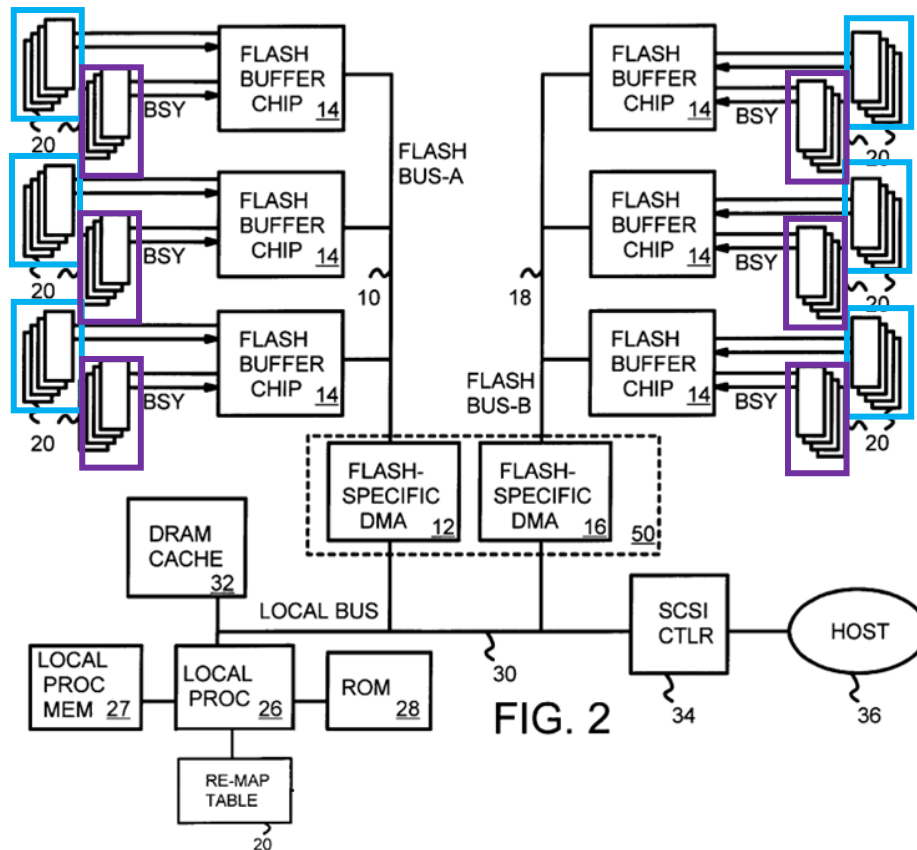
2. Claim 15, Limitation [B]

wherein said access parameter includes a group identifier for identifying at least a portion of said set of memory devices that are respectively controlled by different FDEs from said set of FDEs.

227. As I explain above for claim 9, Bruce ’006 alone or in combination with Bruce ’251 discloses “*wherein said access parameters includes a group identifier.*” Bruce ’251 further teaches that each group identifier identifies “*at least a portion of said set of memory devices that are respectively controlled by different FDEs from said set of FDEs.*”

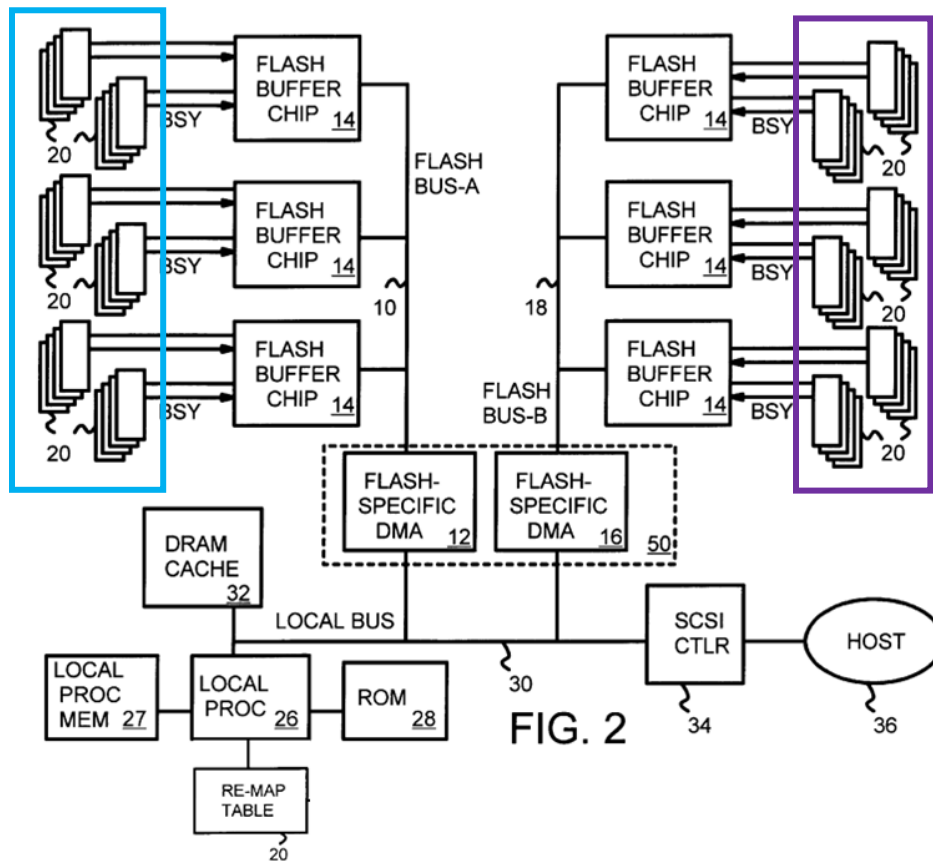
228. As shown below in annotated Figure 2 from Bruce ’251, two banks (*i.e.*, two groups) are attached to each flash-buffer chip 14 (Bruce ’251 at 5:28-29), where one bank identifier identifies half of all the flash-memory chips (annotated in **blue**) that are controlled by the two DMA controllers, and a second bank identifier identifies the half of the flash-memory chips (annotated in **purple**) controlled by the two DMA controllers. In other words, each group identifier identifies *a portion of memory devices controlled by a [sic] FDE* (where the FDE

is a “flash-specific DMA engine,” for the reasons discussed above in ¶ 215). Thus, Bruce '006 in combination with Bruce '251 renders this claim limitation obvious.



Bruce '251 at Figure 2.

229. Alternatively, and illustrated in the figure below, DMA controllers 12 and 16 each identify a set of flash-memory chips respectively controlled by the DMA controllers. For example, DMA controller 12 identifies the flash-memory chips highlighted **light blue**, while DMA controller 16 identifies the flash-memory chips highlighted **purple**. They therefore also illustrate the claimed “group identifier” for the reasons I already address with respect to claim 10.



Bruce '251 at Figure 2.

230. Thus, Bruce '006 combined with Bruce '251 renders this claim limitation obvious.

K. Independent Claim 32

1. Claim 32 Preamble

An electronic storage device that includes a memory system that uses a memory table for increasing the likelihood that an operational load imposed on the storage apparatus during the processing of an I/O transaction request will be optimally distributed across storage device resources, said I/O transaction request received by said storage device from a requesting host.

231. To the extent this preamble is found to be limiting, it is my opinion that the combination of Bruce '251 and Bruce '006 renders obvious the preamble. The re-map table of Bruce '006 together with the two flash-specific DMA controllers 12 and 16 in the combined system of Bruce '006 and Bruce '251 correspond to the claimed “*memory system*” in the '740 Patent. The '740 Patent describes the *memory system* as including a “mapping table” and “a DMA controller 10 ... for performing DMA operations on memory devices”:

[T]he present invention pertains to solutions that increase the likelihood which, in response to an I/O transaction initiated by a host, the operational load imposed on **the memory system** by these memory operations will be optimally distributed across resources used by the memory system, reducing memory operation latency, increasing operational device efficiency, or both. **This memory system, such as memory system 4**, may be used with or implemented in an electronic storage device 2, in the manner illustrated in FIG. 1 in accordance with one embodiment of the present invention. **Memory system 4** is coupled to a memory store 6 via a set of buses, such as buses 8-0 and 8-1, and **uses a mapping table 9. Memory system 4 also includes a DMA controller 10 that includes a flash buffer controller 16 and a set of direct memory access engines, which are suitable for performing DMA operations on memory devices used by memory store 6.**

'740 Patent at 3:15-31 (emphasis added). The '740 Patent further specifies that the *memory system* “is coupled to a memory store 6 via a set of buses[.]” '740 Patent at 3:25-26. Memory system 4 is highlighted in Figure 1 below:

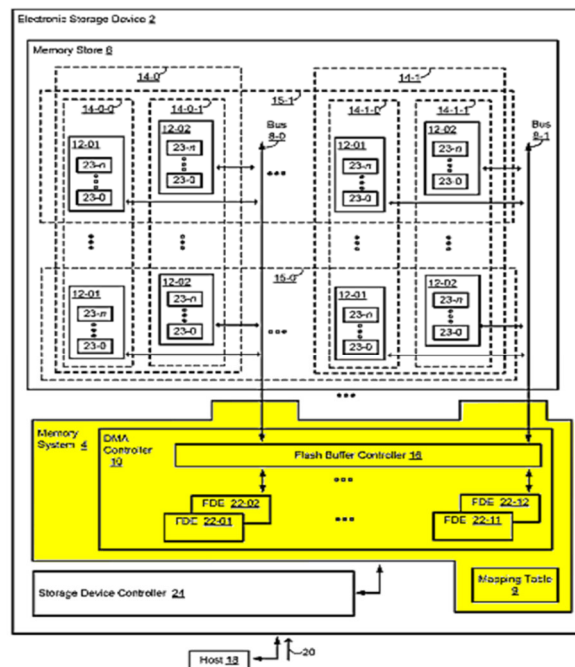
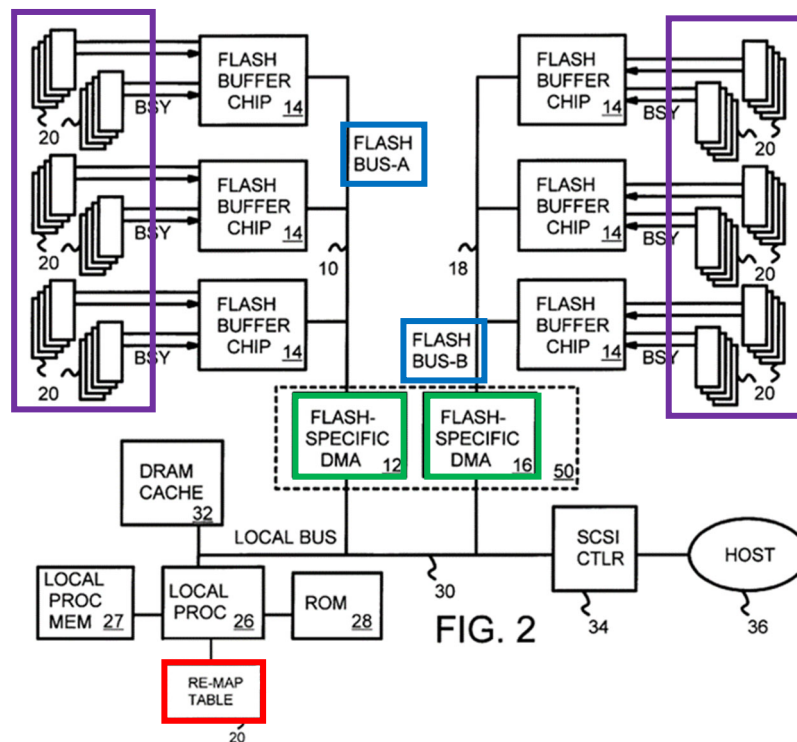


FIG. 1

'740 Patent at Figure 1.

232. In the combined system of Bruce '251 and Bruce '006, the re-map table of Bruce '006 together with the two flash-specific DMA controllers 12 and 16 correspond to the *memory system* described in the '740 Patent. As in the '740 Patent, the two flash-specific DMA controllers 12 and 16 (annotated in green) are coupled to the *memory store* (i.e., the various flash-memory chips, annotated in purple) by flash busses 10 and 18 (annotated in blue). The combined system of Bruce '006 and Bruce '251 therefore discloses a “*memory system*.”



Bruce '251 at Figure 2.

233. The preamble of claim 32 further requires that the memory system “uses a memory table ... during the processing of an I/O transaction request ..., said I/O transaction request received by said storage device from a requesting host.” This language is substantially similar to claim 1’s preamble, which requires “a mapping table for ... memory operations performed by an electronic storage device in response to receiving an I/O transaction request initiated by a host.” In my opinion, a POSITA would not have discerned any meaningful difference between this claim language in claim 1’s preamble and claim 32’s preamble. Thus, Bruce '006 discloses and renders obvious this portion of the preamble for the same reasons provided for claim 1’s preamble above.

234. Bruce '006 in combination with Bruce '251 further discloses the remaining claim limitation which requires that the memory table is for “increasing the likelihood that an operational load imposed on the storage apparatus during the processing of an I/O transaction request will be optimally distributed across storage device resources.” The '740 Patent describes “optimizing memory operations” as including “increasing the likelihood that, in response to an I/O transaction initiated by a host, **the operational load imposed on the storage device by these memory operations will be optimally distributed across different storage resources, such as by interleaving or parallel memory operations.**” '740 Patent at 2:14-19 (emphasis added). As I explain above for claim 1’s preamble, Bruce '006 also taught interleaving and performing multiple accesses in parallel. Bruce '006 at 11:19 (“**[i]nterleaving** can be combined with the invention.”), 11:20-22 (“**[i]nterleaving** is desirable since **multiple accesses can occur in parallel** to the flash devices, improving performance.”). And as I further explain above for claim 10, a POSITA would understand that two PBAs in the re-map table of the combined system of Bruce '006 and Bruce '251 include different bus identifiers, FDE identifiers, and group identifiers. A POSITA would therefore understand that the operational load imposed on the expandable flash-memory system will be optimally distributed across different storage resources in the combined system of Bruce '006 and Bruce '251 when an I/O transaction is mapped to two PBAs with

different bus, FDE, and/or group identifiers. As I discuss above in ¶¶ 58-63, techniques for executing flash-memory operations concurrently and/or overlapped in time were well known by the Earliest Priority Date. Thus, Bruce '006 in combination with Bruce '251 discloses the same manner of increasing the likelihood that the operational load imposed by memory operations will be optimally distributed across different storage resources as the '740 Patent.

2. Claim 32, Limitation [A]

a mapping table which maps a first LBA set to a first PBA, and a second LBA set to a second PBA, said first and second LBA sets including respective sets of consecutive LBAs.

235. As I explain above for claim 1, limitation [B] in Ground 1, Bruce '006 discloses and renders obvious a mapping table “*disposed to associate said first and second LBA sets with said first and second PBAs.*” And as explained for claim 1, limitation [A] in Ground 1, Bruce '006 discloses and renders obvious “*said first and second LBA sets each representing a set of consecutive LBAs.*” Bruce '006 therefore discloses and renders obvious this limitation.

3. Claim 32, Limitation [B]

a memory store that includes a set of flash devices

236. As I explain above for claim 15, limitation [A], Bruce '251 discloses an expandable flash-memory system having “[b]anks of flash-memory chips 20.” And as I explain above for claim 1, limitation [B] in Ground 1, Bruce '006 teaches

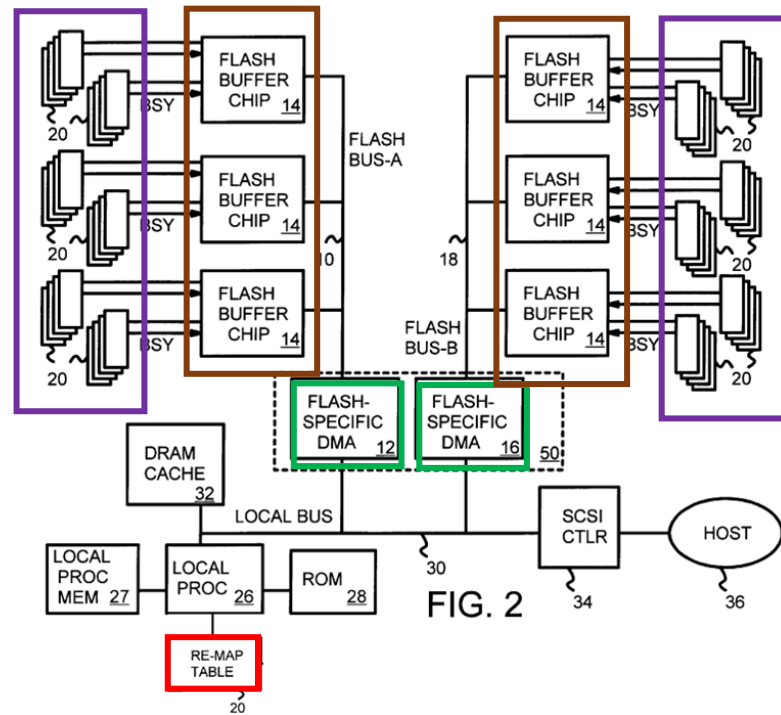
and renders obvious “a set of PBA fields ... said set of PBA fields respectively disposed for representing a set of PBAs, ... said PBAs each associated with a physical memory location in a memory store.” Bruce ’006 in combination with Bruce ’251 thus renders obvious “a memory store that includes a set of flash devices, said set of flash devices includes a first physical memory location and a second physical memory location that are addressable by said first and second PBAs respectively.”

4. Claim 32, Limitation [C]

a memory system coupled to said memory store and disposed to use said mapping table during a memory operation.

237. The claim limitation here requiring “a memory system ... disposed to use said mapping table during a memory operation” is substantially similar to the limitation in the preamble of claim 32, which requires “a memory system that uses a memory table ... during the processing of an I/O transaction request.” Thus, Bruce ’006 in combination with Bruce ’251 renders obvious this portion of the claim limitation for the same reasons described above for claim 32’s preamble.

238. Furthermore, as shown in annotated Figure 2 below for the combined system of Bruce ’006 and Bruce ’251, the memory system (i.e., the two flash-specific DMA controllers 12 and 16 (annotated in **green**) and the re-map table (annotated in **red**) is coupled to the memory store (i.e., the banks of flash memory chips, annotated in **purple**).



Bruce '251 at Figure 2.

239. The combination of Bruce '251 and Bruce '006 therefore discloses that the memory system is *coupled to said memory store*.

5. Claim 32, Limitation [D]

a first PBA disposed for representing a first set of access parameters and a second PBA disposed for representing a second set of access parameters.

240. As I explain above for claim 10, Bruce '006 in combination with Bruce '251 discloses “*said first and second PBA include different bus identifiers, different FDE identifiers and different group identifiers.*” The combination of Bruce '006 and Bruce '251 thus discloses “*wherein said first PBA includes a first*

set of access parameters and said second PBA includes a second set of access parameters, and at least one difference between said first and second PBAs.”

6. Claim 32, Limitation [E]

said mapping table increases the likelihood that that [sic] the operational load imposed on the storage apparatus during the processing of the I/O transaction request will be optimally distributed across storage device resources.

241. Bruce '006 combined with Bruce '251 renders obvious “*said mapping table increases the likelihood that that [sic] the operational load imposed on the storage apparatus during the processing of the I/O transaction request will be optimally distributed across storage device resources.*” As I explain above for claim 32’s preamble, the '740 Patent teaches optimally distributing the operational load imposed by an I/O transaction using interleaving or parallel memory operations by “distributing these memory operations across different storage device resources that are associated with these PBAs, such as across different FDEs, different buses, different flash memory devices, or any combination of these.” '740 Patent at 2:14-19. As I also explain above for claim 32’s preamble, the combined system of Bruce '006 and Bruce '251 teaches interleaving memory operations when an I/O transaction is mapped to two PBAs with different bus, FDE, and/or group identifiers. Thus, Bruce '006 in combination with Bruce '251 teaches increasing the likelihood that the operational load imposed by an I/O

transaction will be optimally distributed across different storage resources as described in the '740 Patent.

L. Dependent Claim 34

The electronic storage device of claim 32, wherein the processing of the I/O transaction request includes a first memory operation performed using said first PBA and a second memory operation performed using said second PBA; said first and second PBAs respectively associated with a first set of access parameters and a second set of access parameters; and said set of first and second set of access parameters differing by at least one access parameter.

242. As I explain above for claim 1, limitation [C] in Ground 1, Bruce '006 teaches and renders obvious “in response to receiving the I/O transaction request, said mapping table causes the electronic storage device to perform optimized memory operations on memory locations respectively associated with said first PBA and said second PBA.” Bruce '006 thus discloses and renders obvious “the processing of the I/O transaction request includes a first memory operation performed using said first PBA and a second memory operation performed using said second PBA.”

243. As I also explain above for claim 1, limitation [B] in Ground 1, Bruce '006 teaches and renders obvious “a first PBA disposed for representing a first set of access parameters and a second PBA disposed for representing a second set of access parameters.” Bruce '006 thus additionally discloses “said first and second

PBAs respectively associated with a first set of access parameters and a second set of access parameters.”

244. And as I explain above for claim 10, Bruce '006 in combination with Bruce '251 discloses “wherein said first and second PBA include different bus identifiers, different FDE identifiers and different group identifiers.” Bruce '006 in combination with Bruce '251 therefore further teaches “said set of first and second set of access parameters differing by at least one access parameter.”

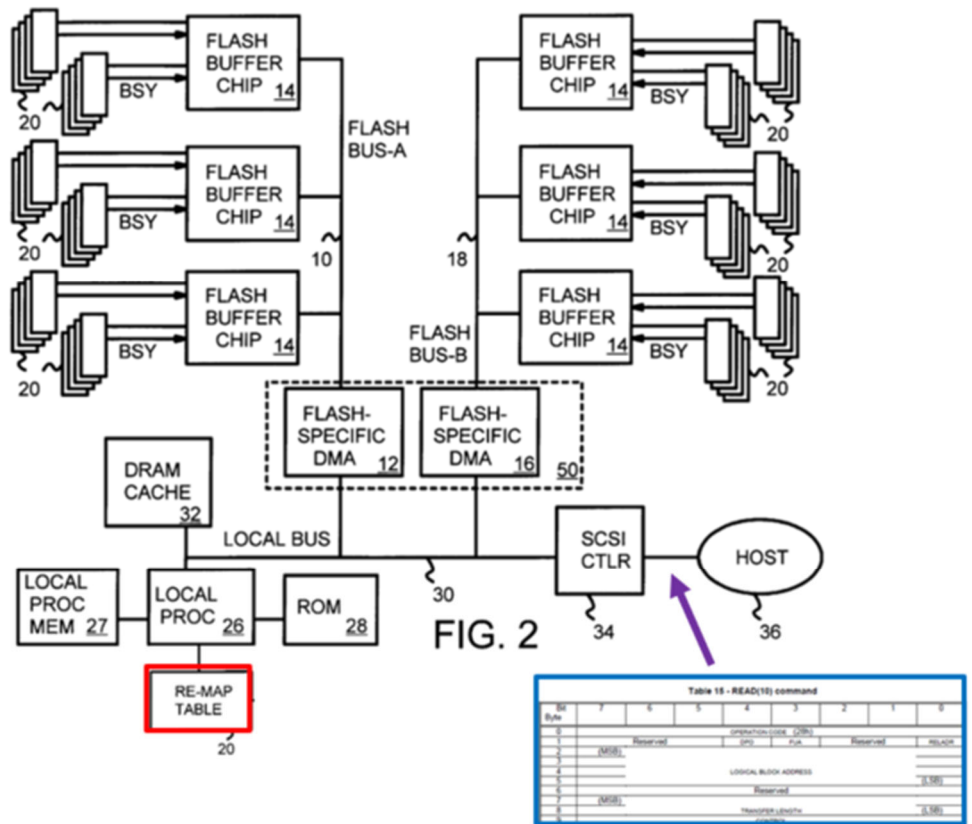
VII. GROUND 4: CLAIMS 1, 9-15, 32, AND 34 ARE OBVIOUS OVER THE COMBINATION OF BRUCE '006, BRUCE '251, AND SBC

A. Motivation to Combine Bruce '006, Bruce '251, and SBC

245. As I explain above in ¶¶187-204, a POSITA would have been motivated to combine the unified re-map table taught in Bruce '006 with the expandable flash-memory system taught in Bruce '251.

246. A POSITA would have also been motivated by Bruce '251's teachings to further incorporate SBC's SCSI block commands into the combined flash-memory system of Bruce '251 and Bruce '006 to arrive at the claimed invention. Bruce '251 expressly teaches using a SCSI interface for the host requests. Bruce '251 describes the expandable flash-memory system as including a “small-computer system interface (SCSI) controller 34” connected to the same local bus 30 that connects other components of the expandable flash-memory system (*e.g.*, local processor 26 and DMA controllers 12 and 16). Bruce '251 at

5:65-67, Figure 2. Bruce '251 explains that “[r]equests from host 36 are **received by SCSI controller 34** and driven onto local bus 30.” Bruce '251 at 6:1-2 (emphasis added). “Local processor 26 responds to these host requests by storing host data into DRAM cache 32 for writes[] or reading data from flash-memory chips 20 or from DRAM cache 32 for reads.” Bruce '251 at 6:2-5. As I discuss above in ¶157, it was well known by the Earliest Priority Date that SCSI storage devices implemented the SBC block commands. A POSITA would therefore have been motivated to use SBC’s SCSI block commands for host requests received by Bruce '251’s flash-memory system, which further incorporates Bruce '006’s unified re-map table as shown in the revised Figure 2 from Bruce '251 below. Bruce '251 at Figure 2 (revised).



247. A POSITA would have had a reasonable expectation of success combining Bruce '006's unified re-map table, SBC's disclosure of SCSI block commands, and Bruce '251's expandable flash-memory system. As discussed above in ¶¶159-161, a POSITA would have had a reasonable expectation of success incorporating Bruce '006's unified re-map table into the expandable flash-memory system disclosed by Bruce '251. Since Bruce '251 discloses using a SCSI interface for the host requests, a POSITA would also have had a reasonable expectation of success in further combining Bruce '251 and Bruce '006 with SBC.

B. Independent Claim 1

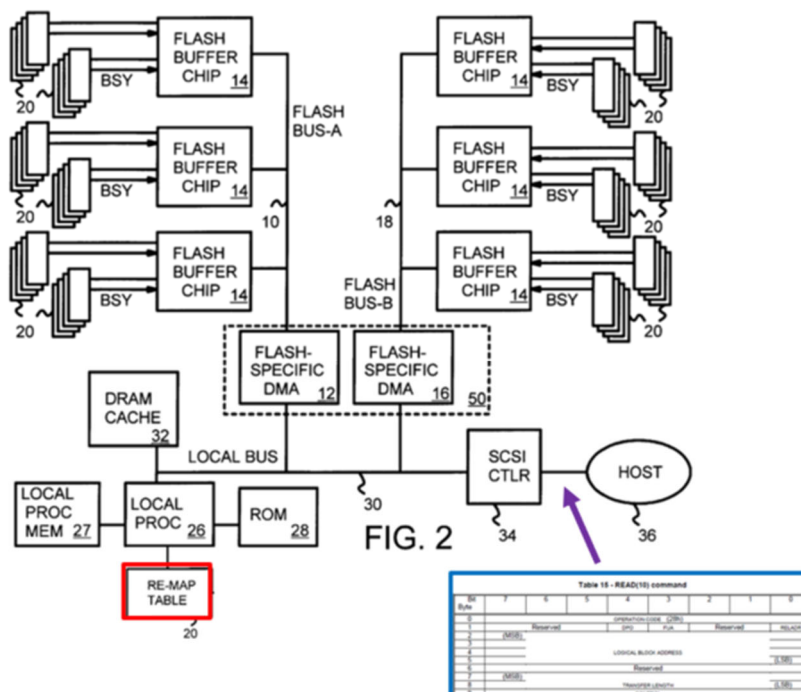
248. It is my opinion that Bruce '006 in combination with Bruce '251 and SBC renders obvious claim 1 of the '740 Patent.

1. Claim 1 Preamble

A mapping table for optimizing memory operations performed by an electronic storage device in response to receiving an I/O transaction request initiated by a host

249. To the extent this preamble is found to be limiting, it is my opinion that it is obvious in view of Bruce '251 in combination with Bruce '006 and SBC.

250. As explained above in ¶246, a POSITA would have been motivated to combine Bruce '006 with Bruce '251 and SBC as shown in the revised Figure 2 below.



251. I have also explained above at ¶205, Bruce '251 in combination with Bruce '006 discloses the claimed “*electronic storage device*.” By further incorporating SBC, the combined flash-memory system of Bruce '251, Bruce '006, and SBC corresponds to the claimed “electronic storage device” for the same reason that the flash-memory system taught in Bruce '251 discloses the claimed “*electronic storage device*.”

252. As I have further explained above at ¶206, Bruce '251 in combination with Bruce '006 also discloses “*an I/O transaction request initiated by a host*.” Using the block commands from SBC for the host requests taught in the combined flash-memory system of Bruce '251 and Bruce '006 does not change the fact that a POSITA would consider the host requests for the combined flash-memory system of Bruce '251, Bruce '006, and SBC to be an “*I/O transaction request initiated by a host*.” Bruce '251's disclosure that its flash-memory system could be a peripheral applies equally to the combined flash-memory system of Bruce '251, Bruce '006, and SBC. Thus, for the same reasons explained above ¶206, Bruce '006 in combination with Bruce '251 and SBC renders obvious “*an I/O transaction request initiated by a host*.”

253. The combined flash-memory system of Bruce '251, Bruce '006, and SBC for ground 4 differs from the SBC-Bruce-006 system I described above for ground 2 in that SBC block commands coming in as the I/O transaction requests

will now be received by the SCSI controller taught in Bruce '251 instead of being directly routed to the re-map table as shown in Figure 1 of Bruce '006. Bruce '251 at 6:1; Bruce '006 at Figure 1. However, this doesn't change how the re-map table taught in Bruce '006 would be used to perform memory operations in response to I/O transaction requests initiated by a host from what I described above for ground 2. A POSITA would understand that there is some logic in the Bruce '006 flash-memory system for converting a single I/O transaction request with transfer length greater than the size of the LBA into multiple LBAs before the multiple LBAs can be used to do the lookup using the re-map table taught in Bruce '006. As I have explained above in ¶171, it was well known in the art how to convert a single host command when the transfer size is larger than the size of an entry in the re-map table and a POSITA would understand this to be part of Bruce '006's flash-memory system but not explicitly shown in Bruce '006's Figure 1. A POSITA would further understand this logic to be part of the SCSI controller or any other components taught in Bruce '251 before a set of sequential LBAs are routed to the re-map table in the combined flash-memory system of Bruce '251, Bruce '006, and SBC. Therefore, the use of Bruce '006's re-map table to perform memory operations in response to SBC's block commands initiated by a host in the combined flash-memory system of Bruce '251, Bruce '006, and SBC remains identical to the way Bruce '006's re-map table is used in ground 2 above.

Therefore, for the same reasons explained above in ¶¶163-171, when the combined flash-memory system of Bruce '251, Bruce '006, and SBC receives a READ(10) or WRITE(10) command taught in SBC for reading or writing more than 16 contiguous logical block which would necessarily result in the combined system to traverse multiple sequential rows of the re-map table taught in Bruce '006, multiple optimized read or write memory operations will be performed. Thus, Bruce '006 in combination with Bruce '251 and SBC renders obvious “[a] mapping table for optimizing memory operations performed ... in response to receiving an I/O transaction request initiated by a host” for the same reasons I provided above for ground 2. As such, Bruce '006 in combination with Bruce '251 and SBC renders obvious the entire preamble, if limiting.

2. Claim 1, Limitation [A]

a set of logical fields, including a first logical field and a second logical field, and said logical fields respectively disposed for representing a plurality of LBA sets, including said first logical field disposed for representing a first LBA set and said second logical field disposed for representing a second LBA set, said first and second LBA sets each representing a set of consecutive LBAs

254. As I explain above in Ground 1, Bruce '006 alone discloses and renders this claim limitation obvious.

3. Claim 1, Limitation [B]

a set of PBA fields, including a first PBA field and a second PBA field, said set of PBA fields respectively disposed for

representing a set of PBAs, including a first PBA disposed for representing a first set of access parameters and a second PBA disposed for representing a second set of access parameters, said PBAs each associated with a physical memory location in a memory store, said set of logical fields and said set of PBA fields disposed to associate said first and second LBA sets with said first and second PBAs; and

255. As I explain above in Ground 1, Bruce '006 alone discloses and renders this claim limitation obvious.

4. Claim 1, Limitation [C]

wherein, in response to receiving the I/O transaction request, said mapping table causes the electronic storage device to perform optimized memory operations on memory locations respectively associated with said first PBA and said second PBA, if the I/O transaction request is associated with said first and second LBA sets.

256. As I explain above in ¶253, the use of Bruce '006's re-map table to perform memory operations in response to SBC's block commands initiated by a host in the combined flash-memory system of Bruce '251, Bruce '006, and SBC remains identical to the way Bruce '006's re-map table is used in ground 2 above. Thus, for the same reasons I provided above for ground 2, Bruce '006 in combination with Bruce '251 and SBC renders this claim limitation obvious.

C. Dependent Claims 9-15

257. Claims 10 and 11 depend on claim 9 which depends on claim 1. Claims 12 to 15 all depend on claim 1. As I explain above in ¶¶209-230, Bruce '006 in combination with Bruce '251 renders obvious the additional limitations in

claims 9 to 15. Thus, the combination of Bruce '251, Bruce '006, and SBC also renders claims 9-11 obvious.

D. Independent Claim 32

1. Claim 32 Preamble

258. To the extent this preamble is found to be limiting, as I explain above in Ground 3, it is my opinion that Bruce '006 in combination with Bruce '251 renders this preamble obvious. It is also my opinion that Bruce '006 and Bruce '251 in further combination with SBC also renders obvious the portion of the preamble referring to optimally distributing the memory load across storage device resources.

259. As I explain above in ¶253, the use of Bruce '006's re-map table to perform memory operations in response to SBC's block commands initiated by a host in the combined flash-memory system of Bruce '251, Bruce '006, and SBC remains identical to the way Bruce '006's re-map table is used in ground 2 above. And for the same reasons explained above in ¶¶163-171, when the combined flash-memory system of Bruce '251, Bruce '006, and SBC receives a READ(10) or WRITE(10) command taught in SBC for reading or writing more than 16 contiguous logical blocks which would necessarily result in the combined system traversing multiple sequential rows of the re-map table taught in Bruce '006, multiple read or write memory operations will be performed. The multiple read or

write memory operations will be directed to memory locations mapped to two or more PBAs in Bruce-006's re-map table as Bruce '006 teaches that each LBA in the re-map table maps to a corresponding PBA. Bruce '006 at 5:33-35, Fig. 2.

260. Moreover, as I explained above in ¶¶164-70, a POSITA would readily understand that interleaving memory operations performed in response to a READ(10) or WRITE(10) command spanning multiple sequential LBAs in Bruce '006 among different flash devices would increase the likelihood of distributing memory operations across different resources, thereby improving performance.

261. For this additional reason, it is my opinion that the combination of Bruce '251, Bruce '006, and SBC also renders this preamble obvious.

2. Claim 32, Limitations [A]-[D]

262. As I explain above in ¶¶112-133, 236-240, Bruce '006 alone and/or in combination with Bruce '251 discloses and renders these claim limitations obvious.

3. Claim 32, Limitation [E]

263. As I explain above in ¶241, Bruce '006 in combination with Bruce '251 renders this claim limitation obvious. As I further explain above in ¶260, Bruce '006 and Bruce '251 in further combination with SBC also teaches optimally distributing the memory load across storage device resources. Thus,

Bruce '006 in combination with Bruce '251 and SBC also renders this claim limitation obvious.

E. Dependent Claim 34

264. As I have explained above in ¶169, a single host READ(10) command that accesses more than 16 and as many as 64K contiguous logical blocks of data would traverse more than one row in the re-map table to identify the different physical blocks of flash memory where the requested data is stored. A WRITE(10) command with comparable transfer length would likewise require multiple memory operations to the flash blocks of the system taught in Bruce '006. In both instances, more than one memory operation is performed on memory locations associated with the PBAs mapped to the two or more sequential LBAs that were part of the host request.

265. As I explain above in ¶253, the use of Bruce '006's re-map table to perform memory operations in response to SBC's block commands initiated by a host in the combined flash-memory system of Bruce '251, Bruce '006, and SBC remains identical to the way Bruce '006's re-map table is used in ground 2 above. Therefore, it is my opinion that Bruce '006 in combination with Bruce '251 and SBC also renders obvious the claim limitation requiring "*wherein the processing of the I/O transaction request includes a first memory operation performed using said first PBA and a second memory operation performed using said second PBA.*"

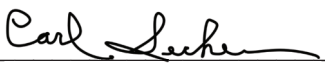
266. And as I have also explained above at ¶¶164-70, 260, a POSITA would understand that interleaving memory operations performed in response to a READ(10) or WRITE(10) command spanning multiple sequential LBAs in Bruce '006 among different flash devices would increase the likelihood of distributing memory operations across different resources. As I explained above at ¶¶128-131, Bruce '006 further teaches using portions of the PBAs in the re-map table for identifying “[d]evice within a bank” and those portions correspond to one of the claimed “access parameters.” Therefore, having memory operations on different flash devices mean the corresponding first and second PBAs have first and second sets of access parameters that differ by at least one access parameter (i.e. the access parameter used to identify the flash devices associated with the memory operation). Thus, it is my opinion that the combination of Bruce '251, Bruce '006, and SBC also renders obvious “*said first and second PBAs respectively associated with a first set of access parameters and a second set of access parameters, and said set of first and second set of access parameters differing by at least one access parameter.*” As such, Bruce '251 in combination with Bruce '006 and SBC renders claim 34 obvious.

In signing this declaration, I recognize that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

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.

Executed this 1st day of April in 2023 in Allen, TX.

Respectfully submitted,



Dr. Carl Sechen