IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent of Gonzalez *et al.*:	Petition for *Inter Partes* Review
U.S. Patent No. 7,012,835	Attorney Docket No.: 337722-000080.835
Issued: March, 14, 2006	Customer No.: 26379
Title: Flash Memory Data Correction and Scrub Techniques	Petitioner: Apple Inc. Real Party-in-Interest: Apple Inc.

PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 7,012,835

Mail Stop Patent Board
Patent Trial and Appeal Board
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Pursuant to 35 U.S.C. §§ 311-319, Apple Inc. ("Petitioner") hereby petitions the Patent Trial and Appeal Board to institute an *inter partes* review of claims 1, 4-10, 13, 15, 17, 18, 20-23, 25, 27 of United States Patent No. 7,012,835 (the "'835 patent") (Ex. 1001).

TABLE OF CONTENTS

Page

TABLE OF CONTENTS
(continued)

Page

TABLE OF CONTENTS
(continued)

Page

TABLE OF CONTENTS
(continued)

Page

TABLE OF CONTENTS
(continued)

Page

TABLE OF CONTENTS
(continued)

Page

TABLE OF CONTENTS
(continued)

Page

Exhibit Number	**Description**
1001	U.S. Patent No. 7,012,835 to Gonzalez
1002	Prosecution File History For 7,012,835 to Gonzalez
1003	Declaration of R. Jacob Baker
1004	U.S. Patent No. 5,341,339 to Wells
1005	U.S. Patent No. 6,151,246 to So
1006	U.S. Patent No. 6,396,744 to Wong
1007	U.S. Patent No. 5,838,614 to Estakhri
1008	Excerpts from *Designing with FLASH MEMORY: The Definitive Guide to Designing Flash Memory Hardware and Software for Components and PCMCIA Cards*, Brian Dipert & Markus Levy Annabooks (1993, 1994)
1009	U.S. Patent No. 5,485,595 to Assar
1010	PC Card Standard, Volume 7, Media Storage Formats Specification
1011	*A Floating Gate and Its Application to Memory Devices*, D. Kahng and S. M. Sze
1012	*New Ultra High Density EPROM and Flash EEPROM with NAND Structure Cell*, Fujio Masuoka *et. al*.
1013	Toshiba Web at flash25.toshiba.com
1014	U.S. Patent No. 5,418,752 to Harari
1015	U.S. Patent No. 6,362,049 to Cagnina
1016	Intel FDI User Manual

Exhibit Number	**Description**
1017	*Cleaning Policies in Mobile Computers Using Flash Memory*, M.-L. Chiang & R.-C. Chang
1018	*Managing Flash Memory in Personal Communication Devices*, Mei-Ling Chiang *et. al*.
1019	U.S. Patent No. 5,740,395 to Wells
1020	U.S. Patent No. 5,940,861 to Brown
1021	U.S. Patent No. 7,224,607 to Gonzalez
1022	U.S. Publication No. 2003/0046487 to Swaminathan
1023	Intel Series 2 Flash Memory Cards Data Sheet
1024	*Curriculum Vitae* of R. Jacob Baker
1025	U.S. Patent No. 5,532,962 to Auclair
1026	U.S. Patent No. 5,909,449 to So
1027	U.S. Patent No. 5,754,567 to Norman
1028	Excerpts from *Nonvolatile Semiconductor Memory Technology: A Comprehensive Guide to Understanding and Using NVSM Devices*, by William D. Brown and Joe E. Brewer, ed., IEEE Press (1998).

I. MANDATORY NOTICES

A. Real Party-in-Interest

Pursuant to 37 C.F.R. § 42.8(b)(1), the real party-in-interest is Apple Inc.

B. Related Matters

Pursuant to 37 C.F.R. § 42.8(b)(2), Petitioner states that Longitude Flash Memory Systems S.A.R.L. ("Patent Owner") is asserting U.S. Patent 7,012,835 (the "'835 patent") against the Real Party-In-Interest in a suit filed September 23, 2014, styled *Longitude Licensing Ltd., and Longitude Flash Memory Systems S.A.R.L. v. Apple Inc.*, Case No. 3:14-cv-4275, pending in the United States District Court for the Northern District of California (the "Related Litigation").

Petitioner has filed, or soon will file, petitions for *inter partes* review of U.S. Patent Nos. 6,510,488; 6,763,424; 6,831,865; 6,968,421; 7,224,607; 7,120,729; 7,181,611; 7,657,702; 7,818,490; 7,970,987; 8,050,095; and 8,316,177.

As of the filing of this petition, no other judicial or administrative matters are known to Petitioner that would affect, or be affected by, a decision in an *inter partes* review of the '835 patent.

C. Lead and Back-up Counsel

Lead counsel for this matter is Brent Yamashita (USPTO Reg. No. 53808), and back-up counsel for this matter is Edward Sikorski (USPTO Reg. No. 39478) and Harpreet Singh (USPTO Reg. No. 71842), all at the e-mail address: Apple-

Longitude-IPR@dlapiper.com. The postal and hand delivery address for both is DLA Piper LLP (US), 2000 University Avenue, East Palo Alto, California, 94303, and the telephone and fax numbers are (650) 833-2348 (for phone) and (650) 687-1206 (for fax).

D. Service Information

Pursuant to 37 C.F.R. § 42.8(b)(4), papers concerning this matter should be served on the following email address: Apple-Longitude-IPR@dlapiper.com.

II. GROUNDS FOR STANDING

Pursuant to 37 CFR § 42.104(a), Petitioner certifies that the ’835 patent is available for *inter partes* review, and Petitioner is not estopped or barred from requesting *inter partes* review challenging the ’835 patent on the grounds identified in this petition.

III. RELIEF REQUESTED

Petitioner asks that the Board review the accompanying prior art and analysis, institute a trial for *inter partes* review of claims 1, 4-10, 13, 15, 17, 18, 20-23, 25, 27 of the ’835 patent, and cancel claims 1, 4-10, 13, 15, 17, 18, 20-23, 25, 27 as invalid for the reasons set forth below.

IV. THE REASONS FOR THE REQUESTED RELIEF

The full statement of the reasons for relief requested is as follows:

A. Summary of Reasons

- **Challenge #1: Claims 1, 4, 13, 15, 17, 18, 20-23, 25, 27 of the ’835 patent are obvious under pre-AIA 35 U.S.C. § 103(a) in light of U.S. Patent No. 6,151,246 (“So”).**

- **Challenge #2: Claims 5 – 8 of the ’835 patent are rendered obvious by U.S. Patent No. 6,151,246 (“So”) in view of U.S. Patent No. 5,740,395 (“Wells”).**

- **Challenge #3: Claims 9 & 10 of the ’835 patent are rendered obvious by U.S. Patent No. 6,151,246 (“So”) in view of U.S. Patent No. 6,396,744 (“Wong”).**

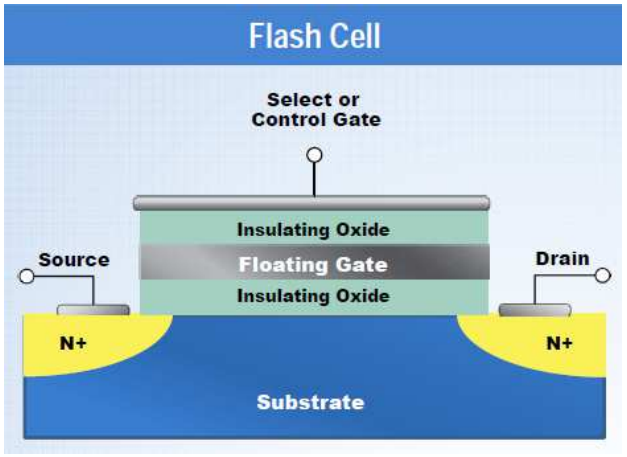
B. Relevant Background Technology

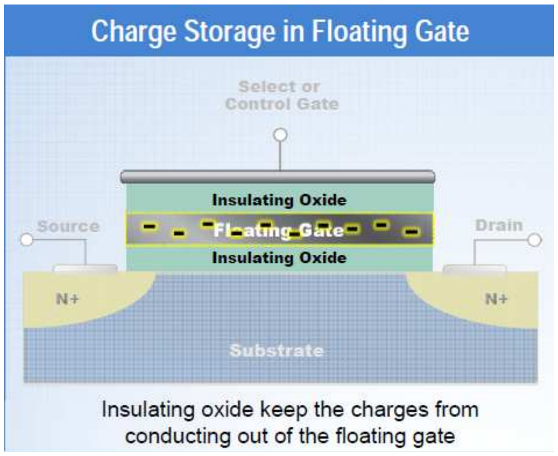
1. Overview of Flash Memory

Flash memory is a type of solid state semiconductor non-volatile memory. These devices are now ubiquitous in consumer electronic devices as data storage devices, even replacing magnetic disk drives in desktop computers. Ex. 1003 at ¶ 15, Declaration of Dr. Jacob Baker (“Baker Decl.”).

Flash memory typically comprises an array of flash memory cells organized in rows and columns, as in conventional memory systems (such as DRAM or SRAM). Each flash memory cell utilizes a floating gate within a field effect transistor (“FET”) to store electrical charge. Ex. 1003 at ¶ 19.

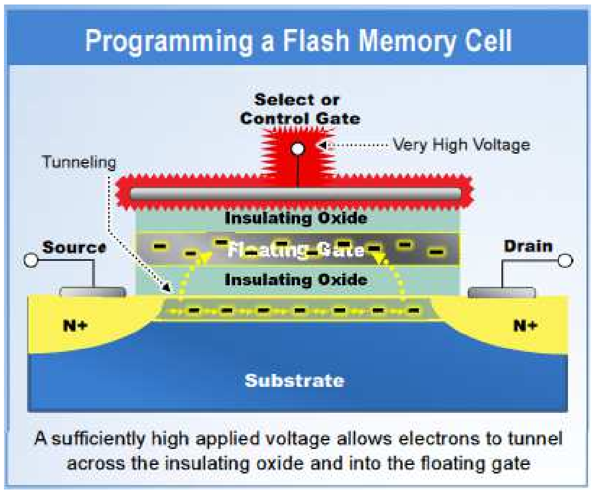
Shown below is an illustration of a typical flash memory cell with a floating gate added to a standard FET structure, from Ex. 1003 at ¶¶ 20 and 21.

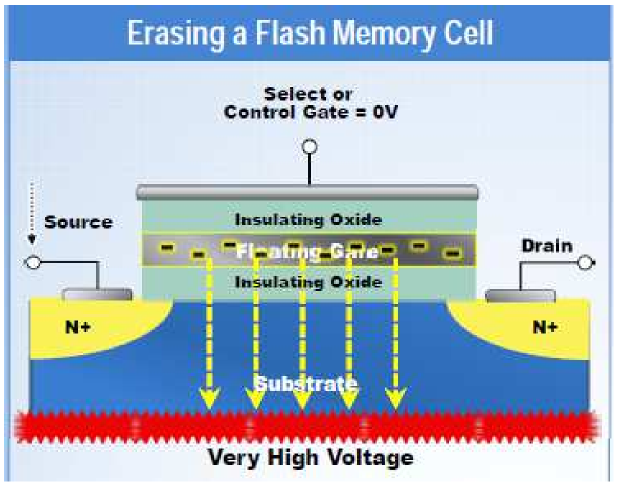



Charge Storage in Floating Gate
Select or Control Gate
Insulating Oxide
Floating Gate
Insulating Oxide
Source
Drain
N+
N+
Substrate
Insulating oxide keep the charges from conducting out of the floating gate

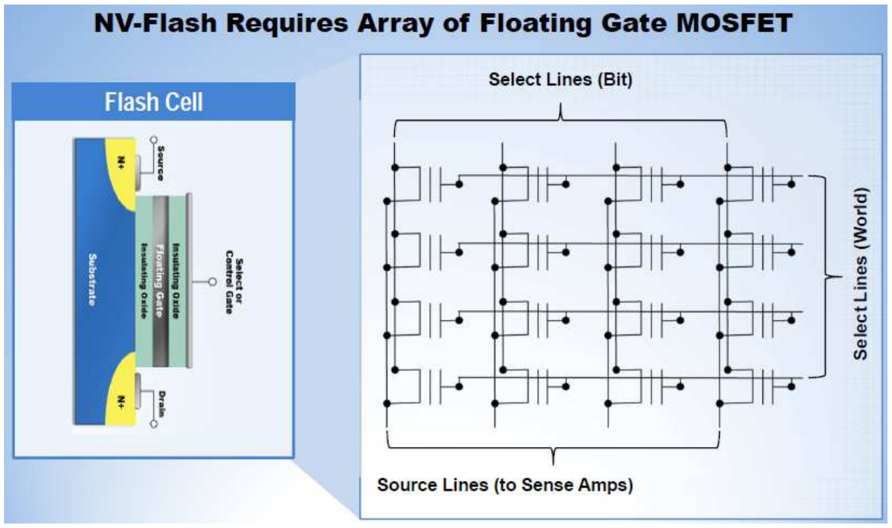
The amount of electrical charge stored in the floating gate can be used to represent data bits ("1" or "0"). Ex. 1003 at ¶¶ 21, 22. Since the "floating gate" is electrically insulated from the terminals of the FET, charge cannot readily conduct into or out of the floating gate, which allows long-term storage of the charge even when power is removed from the device. *Id.* at ¶ 19.

In order to utilize such floating gate FET's as memory cells, there must be a way to controllably add or remove charge from the floating gate. This can be accomplished by applying high voltage differences across the terminals of the memory cell. *See e.g.*, Ex. 1008 at 27, 28, 33, 34, 36. Adding charge to the floating gate is termed "programming" (changing the memory from "1" state to "0" state) and removing charge is termed "erasing" (changing from "0" to "1"). Ex. 1003 at ¶¶ 23-24.


Programming a Flash Memory Cell
Select or Control Gate
Very High Voltage
Tunneling
Insulating Oxide
Floating Gate
Insulating Oxide
Source
Drain
N+
N+
Substrate
A sufficiently high applied voltage allows electrons to tunnel across the insulating oxide and into the floating gate

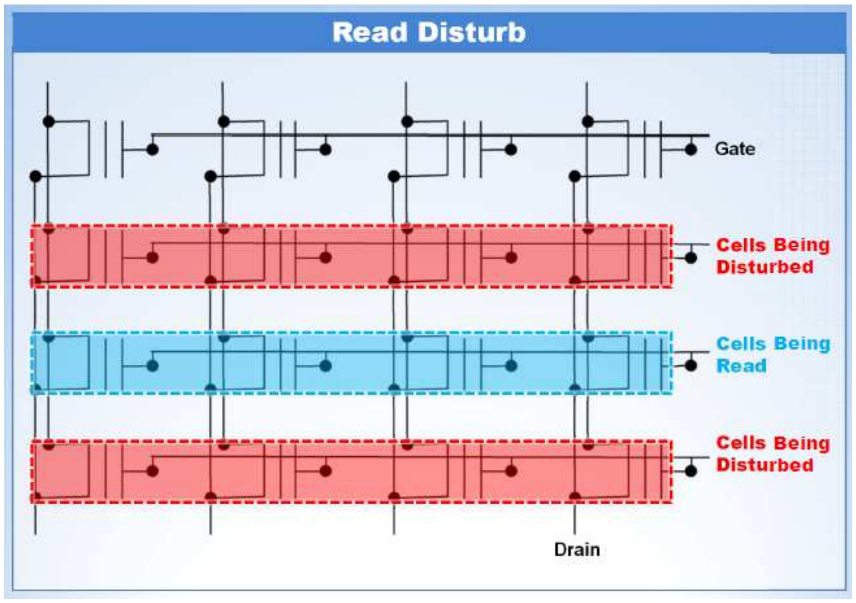


In real-world products, it is advantageous to integrate as many memory cells into as small an area as possible to maximize the storage density. *Id.* at ¶ 47. Therefore, a large number of cells are electrically interconnected into rows and columns of cells in close proximity to one another.



Ex. 1003 at ¶ 24.

This highly integrated arrangement of memory cells causes greater electrical coupling between neighboring cells, since they now share common word lines or bit lines. As was well-known at the time of the ’835 patent, reading and/or writing to one part of the memory array inevitably exposes neighboring cells (that are not being accessed) to high voltages that can disturb the charges in the floating gates in those cells.



Ex. 1003 at ¶ 48.

If enough charge inadvertently leaks out of (or into) the floating gate of a cell, this will appear as a bit error (i.e., a "1" will turn into a "0," or vice-versa) during a read operation. This phenomenon is well-known in the art, and is often referred to as a read/write disturb error. *See also* Ex. 1028 at 193, 213-216, 222-223, 227 and 244. To minimize the likelihood of read/write disturb errors, it was also known to "refresh" or "scrub" the stored data periodically. This generally involves reading the stored data and correcting any errors before they become uncorrectable. The read data (possibly corrected) is thereafter programmed back into the flash memory device to restore the nominal charge levels. *Id*. at ¶¶ 49-52.

Some systems incorporate error correction algorithms in combination with the refresh operation. In general, error correction codes ("ECC") can detect whether there are errors in a stored string of data by applying a special algorithm to the stored data (when it is first stored and therefore error free). Ex. 1003 at ¶ 55. The algorithm will return a code that can be stored in a part of the memory that is associated with the data. At a later time, when the data is read, the ECC circuitry can apply the algorithm to the retrieved data, and compare the resulting code with what was previously obtained. *Id.* If the two codes do not match, then this indicates an error has occurred. Depending on the sophistication of the ECC scheme, the specific bits within the string where the error occurred can be determined, and thus corrected. *Id.* In systems employing ECC, the refresh operations can be less frequent, thereby allowing the system to dedicate more time for essential tasks. *Id.* at ¶ 55. Applying error correction techniques to flash memory was also customary in the art. *See e.g.* Ex. 1027 at 8:61-9:15, 13:56-14:67.

In general, these operations (*i.e.* data scrub/refresh) are executed by the flash memory controller in the background and are implemented in a way that does not interfere with the memory access demands of the host system (e.g., normal host read and write operations). *Id.* at ¶¶ 56-58. If desired, the system can be designed to interrupt and pause or postpone such background operations until other timesensitive operations are completed. *Id.* Moreover, it was well known that the use of additional circuitry such as memory buffers and independent sense circuits can allow for parallel operations in different parts of the flash memory array so that background operations can run concurrently with host operations. *Id.* at ¶¶ 58-59. Including a plurality of memory sub-arrays each with dedicated erase/write/read circuitry allows for parallel operations to take place in the memory system, which can provide higher bandwidth in the system. *See e.g.,* Ex. 1008 at 175-184, 201-204; *see also* Ex. 1006 at 4:31-38; Ex. 1019 at 17:63-18:4, 20:23-29. Using buffers for temporary storage can improve throughput and provide greater flexibility to the overall system (such as movement of stored data in the background in parallel, or data error handling). *See e.g.*, Ex. 1008 at 66-67, 168; Ex. 1019 at 6:60-66, 24:51-25:7, 30:45-54, 30:61-31:35, 32:39-44.

Memory system designers readily understood and recognized that *e.g.* a host read request should in general be a high-priority task, because it directly affects the end user's perception of the system performance. This affects, for example, the speed with which a software application can be loaded, or how quickly a document can be opened. It was already a common practice before 2002 to design flash memory controllers that ensured prompt and prioritized servicing of host commands. Ex. 1003 at ¶¶ 57-59. This was needed because of the relative duration for the various operations involved. Ex. 1008 at 170. In one commercially available implementation of flash memory, a read operation would take only 60 ns, a program operation would take at least 6 µs (*i.e.* 100x the read time), and an erase operation would take at least 0.3 s (5,000,000x the read time). *Id*. at 32. Therefore, it was well-known to delay or pause the slower operations (*e.g.*, erase operation) and allow the relatively faster operations to take place (*e.g.*, read operation).

C. Overview of the ’835 Patent

The ’835 patent, titled “Flash Memory Data Correction and Scrub Techniques,” was filed on October 3, 2003. The ’835 patent was issued on March 14, 2006 to Carlos J. Gonzalez and Kevin M. Conley.

The ’835 patent relates to flash memory scrubbing/refreshing operations which are used to correct errors in storage levels in a memory cell caused by read/write disturb effects. The ’835 patent is related “to techniques of refreshing and correcting data stored therein, particularly in memory systems having very large memory cell blocks.” Ex. 1001 at 1:7-10. Describing the well-known write disturb phenomenon, the ’835 patent states that “programming of one set of memory cells sharing a line or circuit with a second set of memory cells can disturb the charge levels of the second set” and that “it is beneficial to restore shifting charge levels back to the centers of their state ranges from time-to-time, before disturbing operations cause them to shift completely out of their defined ranges, in which case erroneous data are then read." Ex. 1001 at 3:57-60; 4:3-7. The '835 patent seeks to correct this problem by a scrub process which "entails reading data in areas that have received exposure to potentially disturbing signals, and performing some corrective action if this data is determined to have been disturbed." Ex. 1001 at 4:42-45. The claims of the '835 patent are directed to embodiments of a system which operates the scrub process.

The different independent claims being challenged are directed to varying aspects of the disclosed data scrub technique process, none of which are novel. The specification of the '835 patent itself cites to prior art patents which disclose scrubbing techniques to reduce errors in non-volatile memory. Such Applicant Admitted Prior Art (AAPA) includes U.S. Patent Nos. 5,532,962 (Ex. 1025) and 5,909,449 (Ex. 1026):

> As the memory device is used, the threshold level of a cell not subjected to erase or program operations may lose margin, thereby producing a soft error not readily detectable by normal operations of the device. The invention provides a scheme for continually "scrubbing" the sectors in the array to maintain all cells within the proper margins.

Ex. 1025 at 3:59-64.

> In accordance with another aspect of the invention, a non-volatile memory performs a refresh cycle in which memory cells are read and

> threshold voltages of the memory cells are reprogrammed to an allowed state.

Ex. 1026 at 2:29-32.

Hence, the ’835 patent itself admits that such scrub or refresh processes were generally well-known and understood in the art. The claims being challenged in this petition merely claim well-known variations of scrubbing techniques. For example, claims 1, 13, and 23 additionally require copying uncorrected data into the block which contains the newly refreshed data to “consolidate” the corrected and uncorrected data. Ex. 1001 at 29:14-17; 31:50-55; 32:8-13. This is nothing more than an express recognition that host data is generally managed at the granularity of a “sector” (typically, of 512 bytes), rather than by individual bits (the granularity at which errors may occur). *See e.g.,* Ex. 1019 at 4:4-8; Ex. 1007 at 1:65-2:2; 2:63-65; 5:52-56; Ex. 1009 at 2:5-8; 2:66-3:1; Ex. 1014 at 5:12-18; 14:20-25. Copying data from one location to another would naturally preserve this sector level of granularity. Another well-known step in the refresh process involves temporarily storing data in a buffer. Claim 7 additionally requires storing other information (e.g., information about the location) along with the data from that location. Claim 9 requires allowing the system access to other parts of the array during one step of the refresh process. As shown below, all these minor variations to the scrub/refresh process were already well-known in the art before the ’835 patent priority date.

D. Level of Ordinary Skill in the Art

A person of ordinary skill in the art (“POSITA”) is a hypothetical person who is presumed to have known the relevant art at the time of the alleged invention. *Custom Accessories, Inc. v. Jeffrey-Allan Indus., Inc.*, 807 F.2d 955, 962 (Fed. Cir. 1986). Petitioner submits that a person of ordinary skill in the art at the time of the ’835 patent would have a minimum of a Bachelor of Science degree in electrical engineering, computer science, computer engineering, or a related field, and at least two years of experience working in the field of semiconductor memory design, or equivalent. Ex. 1003 at ¶ 79. Such a person would have been capable of understanding the ’835 patent and applying the prior art references as explained in this Petition. *Id*.

E. Claim Construction

Pursuant to 37 C.F.R. §§ 42.100(b) and 42.204(b)(2), this petition presents claim analysis that is consistent with the broadest reasonable construction in light of the specification. Claim terms are given their ordinary and accustomed meaning as would be understood by one of ordinary skill in the art, unless the inventor, as a lexicographer, has set forth a special meaning for a term. *Multiform Desiccants,* *Inc. v. Medzam, Ltd.*, 133 F.3d 1473 (Fed. Cir. 1998); *York Prods., Inc., v. Central Tractor Farm & Family Ctr.*, 99 F.3d 1568, 1572 (Fed. Cir. 1996).

Accordingly, using the broadest reasonable interpretation standard, the terms should be given their ordinary and customary meaning as understood by a person of ordinary skill in the art and consistent with the disclosure. Such understanding for some terms are discussed below.

1. "scrub trigger event" (claims 1, 7 and 9)

Independent claims 1, 7, and 9 recite a method comprising "identifying when a scrub trigger event has occurred…" Ex. 1001 at 28:50; 29:31; 29:53. The '835 provides several examples of scrub trigger events:

> 1. When a data read, data write or erase operation occurs within a given block or other unit of the array that may disturb the charge levels-of other units. The intensity and/or duration of the operation may be important in determining whether to trigger a scrub operation, as well as the susceptibility of the array to disturbs (such as when the memory is operating in multi-state with narrow charge level ranges defining the individual states).
> 2. When a normal system read operation of a given unit reads data with at least one or pre-set number of bit errors.
> 3. When margin reads (with reference levels set to read a narrower programmed distribution than the normal read)

> show that the threshold levels of the programmed cells, although no bit errors exist, are not optimal.
>
> 4. After a predefined interval of time has passed since the last scrub operation.
> 5. When the host initiates a scrubbing operation.

Ex. 1001 at 20:1-20.

> A scrub trigger event may further be limited to occur in a deterministic, random or pseudorandom manner:
>
> (a) After a specified number of host operations;
>
> (b) After a specified number of physical read, write and/or erase operations;
>
> (c) After a specified time period;
>
> (d) Based upon usage characteristics of the host; or
>
> (e) A random or pseudo-random sequence, the generation and checking of which may be tied to any of the above.

Ex. 1001 at 20:66-21:8.

The specification describes detecting or monitoring a scrub trigger event, which thereafter triggers the scrub operation: "Once a scrub trigger event is detected, a next step 93 determines locations within the memory array for performing a scrub operation;" "when a scrub trigger event is detected;" "the memory is monitored for a scrub trigger event." *Id.* at 20:24-26; 24:43-44; 24:46-47, respectively.

In addition, the ’835 patent uses the terms scrub and refresh synonymously: “[s]uch a process, termed data refresh or scrub is described…” Ex. 1001 at 4:7-8. One skilled in the art reviewing the ’835 patent would understand that “scrub” as used in the ’835 patent is the same as what is more commonly referred to as a refresh process. Consequently, one skilled in the art reviewing the ’835 patent would understand that “scrub trigger event” is an event that signals the need for a scrub or refresh operation. Ex. 1003 at ¶¶ 84-85.

2. “flash memory cell array” (claims 1, 7 and 9)

Independent claims 1, 7, and 9 include limitations that involve “a flash memory cell array”. Ex. 1001 at 28:45; 29:29; 29:51. The ’835 patent uses the term “array” broadly. For example, the specification states that “ the array is typically divided into sub-arrays, commonly referred to as planes, which contain their own data registers and other circuits to allow parallel operation such that sectors of data may be programmed to or read from each of several or all the planes simultaneously.” Ex. 1001 at 2:56-60. Furthermore, an array can span multiple integrated circuits as an “array on a single integrated circuit may be physically divided into planes, or each plane may be formed from a separate one or more integrated circuit chips.” *Id.* at 2:60-64. *See also, id*. at 1:38-65. One skilled in the art reviewing the ’835 patent would understand that a “flash memory cell array” refers an organized grouping of flash memory cells, including one or more sub-arrays or planes, located in one or more integrated circuit chips. Ex. 1003 at ¶¶ 86-87.

F. Challenge #1: Claims 1, 4, 7, 8, 13, 15, 17, 18, 20-23, 25, 27 of the ’835 patent are obvious under pre-AIA 35 U.S.C. § 103(a) in light of U.S. Patent No. 6,151,246 (“So”).

1. Overview of So

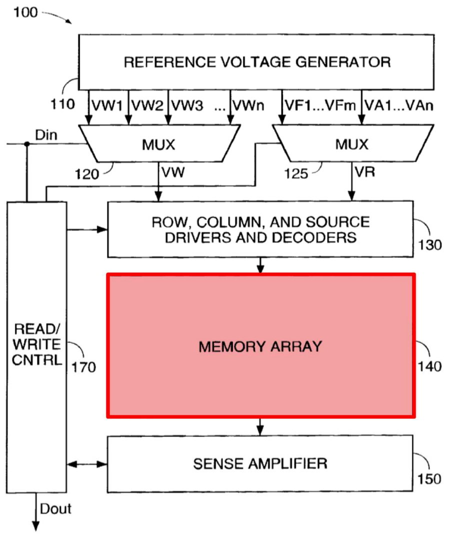
So, in combination with the knowledge of a POSITA, renders obvious claims 1, 4, 13, 15, 17, 18, 20-23, 25, 27 of the ’835 patent. So was filed on November 25, 1998 as a continuation-in-part application of Ser. No. 08/924,909 filed September 8, 1997, and issued on November 21, 2000. Therefore, So is prior art against the ’835 patent under pre-AIA 35 U.S.C. §§ 102(a), 102(b), and 102(e).

So discloses a refresh process for use in a non-volatile memory. Ex. 1005 at 2:35-38. So seeks to solve the well-known problem of unintended threshold voltage changes in memory cell which occur over time as programming and reading of cells is carried out, *i.e.* read or write disturb effects. *Id.* at 1:56-2:2. So discloses an error detection circuit used to detect disturbances to the memory cells. The disturbances are detected by determining whether the threshold voltage for a cell is in a predetermined allowed state or in a forbidden zone. Ex. 1005 at 2:13-25. This is similar to what the ’835 patent refers to as “margin read” to determine the state of stored data. *See e.g.*, Ex. 1001 at 16:64-17:4; 20:13-16. So seeks to solve the read/write disturb problem by using a refresh cycle (i.e., scrubbing). So discloses that an error detection of a sector results in that sector being marked as requiring a refresh. Ex. 1005 at 8:60-65. The refresh operation can be delayed to run during a period of inactivity or during startup procedures. *Id.* at 8:66-9:5. The refresh operation begins by reading the affected data sector and temporarily storing the data in a buffer before erasing the affected data sector. *Id.* at 9:6-11; 9:24-29. Next, the refresh operations can correct the data by using error detection and correction codes. *Id.* at 9:40-43. Afterwards, the corrected data can be written back into the original sector or to a different sector. *Id.* at 9:48-49.

2. So renders obvious claim 1:

a. Claim 1.[pre]: A method of operating a flash memory cell array that is organized into sub-arrays with the sub-arrays including blocks of a minimum number of memory cells that are erasable together and the blocks storing a number of units of data, comprising

If the Commission finds that the preamble is limiting, then So discloses the features recited in this preamble. So discloses that it "relates to non-volatile semiconductor memory" and that its disclosure is "not limited to flash EPROM of the exemplary embodiment but can be employed in a variety of memory architectures including but not limited to



EPROM, E^2PROM, and flash E^2PROM." Ex. 1005 at 1:12-13; 4:41-44.

Further, "[o]ne specific embodiment of the invention is a non-volatile semiconductor memory that includes: an array of memory cells where each memory cell that stores data has a threshold voltage that identifies a multibit data value…" Ex. 1005 at 2:58-61. So further discloses "[i]n an exemplary embodiment of the invention, memory 100 is a flash EPROM, and array 140 includes **hundreds or thousands of rows or columns of N-channel floating gate transistors (memory cells) organized into independently erasable sectors**." *Id.* at 4:17-22, emphasis added. *See also, id.* at 2:58-61; 3:23-25; 4:15-17; FIG. 1. Ex. 1003 at pp. 65-67 (Table 1, Claim 1.[pre]).

b. Claim 1.[a]: identifying when a scrub trigger event has occurred for data stored in at least one of the units of data in a first one of the blocks,

So discloses identifying when a scrub trigger event has occurred for data stored in at least one location of the array. So discloses various events that signal the need for a refresh process. Such scrub trigger events can include "… time since the last refresh of the sector, the number of erase/write cycles associated with a sector, or even the threshold voltage read during the last read cycle." *Id.* at 8:31-36. The '835 patent discloses similar parameters used to determine scrub trigger events (*see e.g.*, Ex. 1001 at 20:1-20; 20:66-21:8). So further discloses that the refresh process may be periodic and "[s]uch periods are typically on the order of at least weeks or months for current non-volatile memory but more frequent refreshes having a period of on the order of a day or less can be used." Ex.1005 at 10:35-38. *See also*, *id.* at 8:45-55; 10:21-51; Ex. 1003 at pp. 67-68 (Table 1, Claim 1.[a]).

c. Claim 1.[b]: reading the identified at least one unit of data from the first block,

So discloses reading the data stored in said at least one location in the array, reading the data from a sector of memory cells, including one or more memory cells with errors. After the scrub trigger event identifies a memory location, the "refresh controller 620 reads the identified sector." Ex. 1005 at 9:6-7. *See also*, *id.* at 2:29-34; 9:33-35; Ex. 1003 at p. 69 (Table 1, Claim 1.[b]).

d. Claim 1.[c]: correcting any errors in the data read from the first block to provide corrected first block data,

So discloses correcting any errors in the data read during the refresh operation: "data errors can be corrected by reading the data values from a sector of memory cells including one or more memory cells containing one or more errors, erasing the sector, and then programming the sector with corrected data values." Ex. 1005 at 2:29-34. In the refresh operation, after making the determination of an error, "the control circuit writes a corrected threshold voltage that corrects the error that the error detection circuit detected." *Id.* at 3:2-4. Additionally, So discloses a data correction circuit having an error correction code to identify the corrected threshold voltage. *Id.* at 9:37-43. The timing of the error correction in the overall refresh process can be modified as desired. *Id.* at 9:44-47; Ex. 1003 at pp. 69-70 (Table 1, Claim 1.[c]).

e. Claim 1.[d]: writing the corrected first block data as at least one unit of data to a second one of the blocks,

So discloses writing the corrected data into a second block. After the data has been corrected and stored in the buffer, and data has been programmed/read to/from another location in the array, the corrected data is written back into the array. "Refresh control 620 controls refresh operations that read the content of a data sector into buffer 610, correct the data, and write data from buffer 610 back to memory array 140." Ex. 1005 at 9:33-35. The correct data can be written back into the original memory location, or alternatively, to a different location (i.e., a second block). *Id.* at 3:5-8; 9:48-49. The system may select a different location in which to write the corrected data if the block has already been erased too often. *Id.* at 9:48-10:4. *See also* Ex. 1003 at ¶¶ 36-46; *see id.* at p. 70 (Table 1, Claim 1.[d]).

f. Claim 1.[e]: thereafter copying uncorrected data units of the first block into the second block, thereby to consolidate in the second block corrected and uncorrected units of data originally of the first block.

So discloses that the corrected data "can be written to the original memory cell containing the error or another memory cell that replaces the original memory cell after the refresh operation." Ex. 1005 at 3:5-8. *See also, id.* at 9:33-35; 9:44-

49; 9:54-61. A POSITA would readily recognize that the original memory sector contained some data (i.e., only a few cells) which required correction, but most data would not require correction. As is known in the art, there is a limit to the number of bits that can be corrected using error correction schemes. *See* Ex. 1003 at pp. 70-71 (Table 1, Claim 1.[e]). Accordingly, if there is to be a "replacement" for data found to have errors in the "original" memory cell, then the entire new memory sector will include not only the corrected data found from those cells, but also have the uncorrected data from the other cells in the block. A POSITA would further understand that it would be disadvantageous to refresh ONLY the bits that have errors, as this would lead to fragmentation of the data. *Id*. It would also prevent the erasure and reclamation of the original memory sector. Therefore, to the extent So does not inherently disclose consolidation of the corrected and uncorrected data into a second block, it would have been obvious to POSITA. *See also* Ex. 1003 at pp. 70-71 (Table 1, Claim 1.[e]).

3. So renders obvious claim 4: The method of claim 1, wherein the scrub trigger event includes an event disturbing said at least one of the data units stored in the first block.

The error detection circuit disclosed in So "can detect the error by finding a threshold voltage in a zone forbidden to threshold voltages corresponding to data or from an error detection code stored when the threshold voltage was written." Ex. 1005 at 3:12-15. The detected drifts in threshold voltages are recognized by the controller as the scrub trigger events that disturb the data units. Over time "charge tends to leak from the floating gates of memory cells and change the threshold voltages of the cells" and such "[c]hanges in the threshold voltage are a problem because the state of the memory cell and the data value stored in the memory cell can change and create a data error." Ex. 1005 1:56-60; 1:64-2:2. The controller can detect these changes and mark cells as requiring a refresh. Ex. 1005 at 8:45-52. *See also* Ex. 1003 at p. 72-73 (Table 1, Claim 4).

4. So renders obvious claim 13

a. Claim 13.[pre]: A method of operating groups of re-programmable non-volatile memory cells that store data as levels of charge therein, wherein individual ones of the groups store a plurality of units of data, and further wherein

If the Commission finds that the preamble is limiting, then So discloses the features recited in this preamble. So discloses that it "relates to non-volatile semiconductor memory" and that its disclosure is "not limited to flash EPROM of the exemplary embodiment but can be employed in a variety of memory architectures including but not limited to EPROM, E^2PROM, and flash E^2PROM." Ex. 1005 at 1:12-13; 4:41-44.

Further, So discloses a plurality of re-programmable non-volatile memory cells that store data as levels of charge. So discloses "[i]n an exemplary embodiment of the invention, memory 100 is a flash EPROM, and array 140 includes hundreds or thousands of rows or columns of N-channel floating gate transistors (memory cells) organized into independently erasable sectors." *Id.* at 4:17-22. *See also, id.* at 2:58-61; 3:23-25; 4:15-17. Ex. 1003 at ¶¶ 18-25; *see also id.* at pp. 73-75 (Table 1, Claim 13.[pre]).

b. Claim 13.[a]: in response to the occurrence of at least one predefined condition, data are read from at least one unit of a first group of memory cells,

So discloses that in response to a predefined condition, data are read from the group of memory cells. So also discloses various predefined conditions that signal the need for a refresh process, such as "time since the last refresh of the sector, the number of erase/write cycles associated with a sector, or even the threshold voltage read during the last read cycle." *Id.* at 8:31-36. These are similar to conditions disclosed by the '835 patent. *See* Ex. 1001 at 20:1-20; 20:66-21:8. In response to the predefined conditions, So discloses reading the data stored in said at least one location in the array. "[R]efresh controller 620 reads the identified sector." Ex. 1005 at 9:6-7. *See also, id.* at 2:29-34; 8:45-55; 9:33-35; 10:21-23; 10:30-31; 10:44-46; Ex. 1003 at pp. 76-78 (Table 1, Claim 13.[a]).

c. Claim 13.[b]: it is then determined whether there are any errors in the read data,

So discloses thereafter determining whether there are any errors in the read data. The error detection circuit determines errors "by finding a threshold voltage in a zone forbidden to threshold voltages corresponding to data or from an error detection code stored when the threshold voltage was written." Ex. 1005 at 3:12-15. So discloses that refresh operations "read the content of a data sector into buffer 610, correct the data, and write data from buffer 610 back to memory array 140." *Id.* at 9:33-35. As would be understood by POSITA, correcting the data would first involve a determination of whether there are any errors in the data read from that sector. So discloses using error correction codes for determining these errors in the stored data. *Id.* at 9:34-43. *See also, id.* at 2:58-63;3:19-22;8:45-55; *See also*, Ex. 1003 at pp. 78-79 (Table 1, Claim 13.[b]).

d. Claim 13.[c]: in response to at least errors being determined to exist in the read data, an effort is made to recover the data erroneously read from said at least one unit of the first group,

This limitation is largely identical to the limitation recited in claim 1.[c] discussed in Section IV.F.2.d above, and the same analysis applies. *See also*, Ex. 1003 at pp. 79-80 (Table 1, Claim 13.[c]).

e. Claim 13.[d]: if recovered, the recovered data are written into at least one unit of a second group of memory cells different from the first group of memory cells, and

This limitation is largely identical to the limitation recited in claim 1.[d] discussed in Section IV.F.2.e above wherein the first and second groups of the instant claim correspond to the first and second blocks of Claim 1.[d] and the same analysis applies *See also*, Ex. 1003 at pp. 80-81 (Table 1, Claim 13.[d]).

f. Claim 13.[e]: data read without errors from other units of the first group of memory cells are copied into units of the second group of memory cells other than its said at least one unit, thereby to consolidate in the second block data read without errors and recovered data originally of the first block.

This limitation is largely identical to the limitation recited in claim 1.[e] discussed in Section IV.F.2.f above wherein the first and second groups of the instant claim correspond to the first and second blocks of Claim 1.[e] and the same analysis applies. *See also*, Ex. 1003 at pp. 81-82 (Table 1, Claim 13.[e]).

5. So renders obvious claim 15: The method of claim 13, wherein the effort to recover the erroneously read data by using an error correction code read along with the data to recover the erroneously read data.

So discloses that "[e]rror detection and correction codes can be used to identify data errors and generate corrected data for refresh operations." Ex. 1005 at Abstract. So further discloses that "error detection and correction codes can be generated and stored for a section, row, column, or other part of a memory and used to correct data errors." Ex. 1005 at 11:49-52. As discussed in the background section of this Petition, the error codes are recalled and used to compare data values. *See also* Ex. 1003 at pp. 82-83 (Table 1, Claim 15).

6. So renders obvious claim 17: The method of claim 13, wherein said at least one predefined condition includes any one or more of programming, reading or erasing memory cells having at least one conductor in common with at least some of the memory cells of said at least unit of the first group of memory cells.

So discloses that "[e]ach row of memory cells has control gates coupled to a row line for the row, and each column of memory cells has drains coupled to a column line for the column. Each erasable sector has a source line coupled to the sources of memory cells in the sector. Row, column, and source drivers and decoders 130 are coupled to memory array 140 and generate voltages that are applied to selected row, column, and source lines in memory array 140 for erase, write, and read operations." Ex. 1005 at 4:16-30. *See also* Ex. 1003 at p. 83 (Table 1, Claim 17).

7. So renders obvious claim 18: The method of claim 13, wherein said at least one predefined condition to include receiving a command from a host to which the groups of memory cells are operably connected.

The read/write control of So controls the refresh process. Ex. 1005 at 5:37-39. One skilled in the art would know that the read/write control receives commands from a host to know what to read and write. Since the read/write control receives commands from a host and controls refresh process, it is inherent that the read/write control receives commands from the host regarding the refresh process. Furthermore, So also discloses that the system can use a timer for systematic memory refreshes. This timer can be off-chip. Ex. 1005 at 10:21-24. One skilled in the art would know that a host can serve as the timer and therefore the controller can receive a command from the host to trigger the refresh. *See also* Ex. 1003 at pp. 83-84 (Table 1, Claim 18).

8. So renders obvious claim 20: The method of claim 13, wherein said at least one predefined condition includes identification of said at least one unit of the first group of memory cells by a deterministic or random sequence.

So discloses a system that is “capable of performing scheduled or delayed refreshes of sectors.” Ex. 1005 at 8:52-54. A scheduled performance corresponds to a deterministic approach to identifying memory cells. Alternatively, So states that “[w]hen an error is detected, error detection circuit 655 marks the sector as requiring a refresh…” *Id.* at 8:60-61. Since the error detection is not scheduled or deterministic, So also is disclosing a random sequence for identifying a memory cell. Furthermore, So states that “[t]he corrected data can be written back into the original sector or to a different sector. Using a different sector helps to "randomize" the number of write/erase cycles for each sector.” *Id.* at 9:48-51. *See also* Ex. 1003 at pp. 84-85 (Table 1, Claim 20).

9. So renders obvious claim 21: The method of claim 13, further wherein the memory cells in individual ones of the groups are simultaneously erased.

So discloses wherein the controller further operates to simultaneously erase the memory cells in individual ones of the groups. So discloses that "memory 100 is a flash EPROM, and array 140 includes hundreds or thousands of rows or columns of N-channel floating gate transistors (memory cells) organized into **independently erasable sectors**." Id. at 4:17-22, emphasis added. *See also, id.* at 2:58-61; 3:23-25; 4:15-17. Ex. 1003 at pp. 85-87 (Table 1, Claim 21).

10. So renders obvious claim 23:

a. Claim 23.[pre]: A method of operating groups of re-programmable non-volatile memory cells that store data as levels of charge therein, wherein individual ones of the groups store a plurality of units of data, and further wherein:

This preamble is largely identical to the preamble of claim 13 discussed in Section IV.F.2.a above, and the same analysis applies *See also,* Ex. 1003 at pp. 88-90 (Table 1, Claim 23.[pre]).

b. Claim 23.[a]: data are read from at least a first group of memory cells,

This limitation is largely identical to the limitation recited in claim 1.[b] discussed in Section IV.F.2.c above wherein a first group of the instant claim corresponds to a first block of claim 1.[b] , and the same analysis applies. *See also,* Ex. 1003 at p. 91 (Table 1, Claim 23.[a]).

c. Claim 23.[b]: it is then determined whether there are any errors in the data read from at least one unit of the first group of memory cells,

This limitation is largely identical to the limitation recited in claim 13.[b] discussed in Section IV.F.4.c above, and the same analysis applies. *See also*, Ex. 1003 at pp. 91-93 (Table 1, Claim 23.[b]).

d. Claim 23.[c]: in response to at least errors being determined to exist in the read data, an effort is made to recover the data erroneously read from said at least one unit of the first group,

This limitation is largely identical to the limitation recited in claim 13.[c] discussed in Section IV.F.4.d above, and the same analysis applies. *See also*, Ex. 1003 at p. 93 (Table 1, Claim 23.[c]).

e. Claim 23.[d]: if recovered, the recovered data are written into at least one unit of a second group of memory cells different from the first group of memory cells, and

This limitation is largely identical to the limitation recited in claim 13.[d] discussed in Section IV.F.4.e above, and the same analysis applies. *See also*, Ex. 1003 at pp.93-94 (Table 1, Claim 23.[d]).

f. Claim 23.[e]: data read without errors from other units of the first group of memory cells are copied into units of the second group of memory cells other than its said at least one unit, thereby to consolidate in the second block data read without errors and recovered data originally of the first block.

This limitation is largely identical to the limitation recited in claim 13.[e]

discussed in Section IV.F.4.f above, and the same analysis applies. *See also,* Ex. 1003 at pp. 94-95 (Table 1, Claim 23.[e]).

11. Claim 25. The method of claim 23, wherein the effort to recover the erroneously read data includes use of error correction code read along with the data.

This limitation is largely identical to the claim 15 discussed in Section IV.F.5 above, and the same analysis applies. *See also,* Ex. 1003 at pp. 95-96 (Table 1, Claim 25).

12. Claim 27. The method of claim 23, further wherein the memory cells in individual ones of the groups are simultaneously erased.

This limitation is largely identical to the claim 21 discussed in Section IV.F.9 above, and the same analysis applies. *See also,* Ex. 1003 at pp. 96-98 (Table 1, Claim 27).

G. Challenge #2: Claims 5 – 8 of the ’835 patent are obvious under pre-AIA 35 U.S.C. § 103(a) in view So in light of U.S. Patent No. 5,740,395 (“Wells”).

1. Overview of Wells

Claims 5 – 8 are rendered obvious by So in light of Wells. Wells was filed on October 30, 1992 and issued on April 14, 1998. Therefore, Wells is prior art against the ’835 patent under pre-AIA 35 U.S.C. §§ 102(a), 102(b), and 102(e).

Wells discloses a method of cleaning-up a non-volatile semiconductor memory. Ex. 1019 at Abstract; 1:44-51; 22:47-52. Clean up is a background process that is triggered when the amount of invalid memory exceeds a threshold level. Ex. 1019 at 3:46-50. One well-known constraint in real-world flash memory devices is that an entire block of flash memory cells must be erased before individual cells within the block can be reprogrammed. Ex. 1003 at ¶ 24. Therefore, when the host system seeks to overwrite existing data, the new data is usually stored in a new physical memory location, and the data in the old location is not immediately erased. Ex. 1003 at ¶¶ 25-26. Over time, this leads to sectors with outdated "dirty" data that needs to be erased to make the sector capable of receiving new data. *Id.*at ¶ 56.

Wells discloses a clean-up process which has three major tasks. "First, a block is selected as the focus of clean-up. Second, sectors of valid user data are copied from the focus block into other blocks, referred to as destination blocks. Third, after all valid sectors have been copied out of it, the focus block is erased, converting dirty sectors into free memory." Ex. 1019 at 22:48-52. These are tasks that are substantially similar to the tasks that are performed during a refresh or scrub operation. *See e.g.*, Ex. 1003 at ¶ 101.

Wells discloses a controller including a top level scheduler which initiates the clean-up process. Ex. 1019 at 10:5-16; 23:17-26. The top level scheduler allocates time to the clean-up. *Id.* Time for clean-up is allocated during periods of host inactivity. *Id.* at 22:53-66. Wells divides the clean-up process into "stages," and at the end of each stage, a pointer to the next state is set and the clean-up process pauses. *Id.* at 22:58-66; 23:17-29. At the next window of time that the top level scheduler allocates for the clean-up process, the next stage commences and this process resumes. *Id.*

2. The combination of So and Wells renders obvious claims 5 and 6

a. Claim 5: The method of claim 1, additionally comprising pausing the operation after reading the data but before correcting any errors thereof, until other higher priority operations are performed.

and

b. Claim 6: The method of claim 1, additionally comprising pausing the operation after correcting any errors thereof but before writing the corrected data, until other higher priority operations are performed.

The combination of So and Wells renders obvious claims 5 and 6, as shown in the analysis in this section. So renders obvious claim 1, as shown above in Challenge 1. So further discloses that the refresh operation can "occur immediately, periodically, during the next startup of the memory, or when the memory becomes inactive for a period of time." Ex. 1005 at 2:66-3:4. So recognizes that refresh operations are lower priority than other system operations and teaches that refresh operations should run during periods of inactivity of memory. *Id.* at 8:65-9:5.

Similarly, Wells discloses that its clean-up process is a background process which is executed when the host interface is inactive. Ex. 1019 at 22:64-66. During this time of host inactivity, the top level scheduler allocates time to a given stage of the clean-up process. *Id.*at 23:17-26. Upon completion of that stage, a pointer to the next stage is generated and the clean-up process pauses giving control back to the system so that other operations can take place. *Id.* at 22:61-63; 23:21-26. The top level scheduler then checks for any host activity, and absent the need to service the host, proceeds with the next clean-up stage. *Id.*

It would have been obvious to combine the teachings of Wells with the refresh operation disclosed in So. Ex. 1003 at ¶ 103. Since So teaches waiting for a period of inactivity of memory before initiating a refresh operation to allow more important operations to occur first, a person skilled in the art would recognize that the disclosure of Wells can improve upon this general desire to allow more important operations to proceed first. *Id.* at ¶ 103. Wells teaches the concept that a time-consuming background process can be broken down into multiple discrete but interruptible "stages," and then there should be small windows of time allocated for executing such background tasks in these stages. At the conclusion of those windows, the overall background task is interrupted and paused so that controller/scheduler can revert attention back to see if the host has a higher priority task. Ex. 1006 at 22:53-66; 23:17-26. One skilled in the art would recognize the benefit of applying Wells' technique to So's refresh process to yield predictable results. Ex. 1003 at ¶ 103. Specifically, by modifying the So refresh operation to run in discrete stages and periodically revert back to perform other host operations, it essentially pauses the refresh operation after various stages (i.e., before/after reading, error correction, writing, etc.) to service the host. This would result in an improved refresh process which can even more readily service higher priority host operations, as desired by So. *Id.* Hence, the controller can operate to pause refresh operations after reading the data but before proceeding with the error correction operation (which can be one "stage") and allow higher priority operations to proceed. Alternatively, the controller can also pause the refresh operation after error correction but before writing data. The timing of the pause can vary depending on the timing of the requirements of the host since the host takes precedence. *See also* Ex. 1003 at pp. 102-107 (Table 2, Claim 5); Ex. 1003 at pp. 107-112 (Table 2, Claim 6).

3. The combination of So and Wells renders obvious claim 7:

a. Claim 7.[pre]: A method of operating a flash memory cell array, comprising:

This preamble is largely identical to the preamble of claim 1 discussed in Section IV.F.2.a above, and the same analysis applies *See also,* Ex. 1003 at pp. 112-114 (Table 2, Claim 7.[pre]).

b. Claim 7.[a]: identifying when a scrub trigger event has occurred for data stored in at least one location of the array,

This limitation is largely identical to the limitation recited in claim 1.[a] discussed in Section IV.F.2.b above, and the same analysis applies. *See also*, Ex. 1003 at pp. 115-116 (Table 2, Claim 7.[a]).

c. Claim 7.[b]: reading the data stored in said at least one location in the array,

This limitation is largely identical to the limitation recited in claim 1.[b] discussed in Section IV.F.2.c above, and the same analysis applies. *See also,* Ex. 1003 at pp. 116-117 (Table 2, Claim 7.[b]).

d. Claim 7.[c]: temporarily storing information about said at least one location and the data read therefrom,

So discloses temporarily storing information about said at least one location and the data read therefrom because the data from the location is temporarily stored in a buffer, and So also discloses storing error detection codes for the location which are then used for error correction. "Refresh control 620 controls refresh operations that read the content of a data sector into buffer 610, correct the data, and write data from buffer 610 back to memory array 140." Ex. 1005 at 9:32-35. So further discloses storing error detection and correction codes for a sector and using the codes to correct data errors, "error detection and correction codes can be generated and stored for a section, row, column, or other part of a memory and used to correct data errors". *Id.* at 11:49-52. *See also* Ex. 1003 at p. 117 (Table 2, Claim 7.[c]).

e. Claim 7.[d]: while this information remains stored, programming or reading other data to or from at least locations within the array other than said at least one location,

So discloses that the refresh operations can take place at a convenient time – when system resources are not tied up in other operations. Ex. 1005 at 2:66-3:2. Further, during the refresh operation the refreshed data can be written in the same original location or to a different location. *Id.* at 9:54-61; 10:2-5.

Similarly, Wells discloses that its clean-up process is a background process which is executed when the host interface is inactive. Ex. 1019 at 22:64-66. During this time of host inactivity, the top level scheduler allocates time to one stage of the clean-up process. *Id.*at 23:17-26. Upon completion of that stage, a pointer to the next stage is generated and the clean-up process pauses giving control back to the system so that other operations can take place. *Id.* at 22:61-63; 23:21-26. Then during the next period of inactivity, the top level scheduler proceeds with the next clean-up stage. *Id.*

It would have been obvious to combine Wells with the refresh operation disclosed in So. Ex. 1003 at ¶ 103. Since So teaches waiting for a period of inactivity of memory before initiating a refresh operation to allow more important operations to occur first, a person skilled in the art would recognize that Wells fulfills this desire to allow more important operations to proceed first. *Id.* Wells teaches the concept of allocating small windows of time for such background tasks, and at the conclusion of those windows, the controller/scheduler should revert attention back to see if the host has a higher priority task. Ex. 1006 at 22:53-66; 23:17-26. One skilled in the art would recognize the benefit of applying Wells' known technique (*i.e.* to break down a background process into multiple discrete but interruptible "stages") to So's refresh process to yield predictable results. With the So/Wells combination, upon temporarily storing the information, the So/Wells process would revert control back to the host. If the host has other program or read operations to run at other locations, then these would take place. Then upon the next period of inactivity, the So/Wells refresh process would continue with the next step in the refresh process. *See also* Ex. 1003 at pp. 118-122 (Table 2, Claim 7.[d]).

f. Claim 7.[e]: thereafter determining whether there are any errors in the read data read from said at least one location in the array,

So discloses thereafter determining whether there are any errors in the read data. The data is read, and then an error detection circuit is used to determine whether there are any errors in the data. The error detection circuit determines errors "by finding a threshold voltage in a zone forbidden to threshold voltages corresponding to data or from an error detection code stored when the threshold voltage was written." Ex. 1005 at 3:12-15. So discloses that refresh operations "read the content of a data sector into buffer 610, correct the data, and write data from buffer 610 back to memory array 140." *Id.* at 9:33-35. As would be understood by POSITA, correcting the data would first involve a determination of whether there are any errors in the data read from that sector. So discloses using error correction codes for determining these errors in the stored data. *Id.* at 9:34-43. *See also, id.* at 2:58-63;3:19-22;8:45-55; Ex. 1003 at pp. 122-123 (Table 2, Claim 7.[e]).

g. Claim 7.[f]: utilizing the stored information, correcting any errors in the data read from said at least one location in the array, and

So utilizes stored data and a correction circuit to correct errors. "During the refresh operation, the control circuit writes a corrected threshold voltage that corrects the error that the error detection circuit detected." Ex. 1005 at 3:19-22. The data correction circuit can use error correction codes to correct data values and threshold voltages. *Id.* at 9:37-43. As discussed in the background section of this Petition, error correction codes are stored in memory and recalled to conduct a comparison between the read data in its current state and when it was first written. "Alternatively, data errors can be corrected by reading the data values from a sector of memory cells including one or more memory cells containing one or more errors, erasing the sector, and then programming the sector with corrected data values." Ex. 1005 at 2:29-34. *See also, e.g., id.* at 11:49-52; Ex. 1003 at pp. 123-125 (Table 2, Claim 7.[f]).

h. Claim 7.[g]: writing the corrected data into the array.

So teaches writing the corrected data into the array. After the data has been corrected and stored in the buffer, and data has been programmed/read to/from another location in the array, the corrected data is written back into the array. "Refresh control 620 controls refresh operations that read the content of a data sector into buffer 610, correct the data, and write data from buffer 610 back to memory array 140." Ex. 1005 at 9:33-35. The correct data can be written back into the original memory location or to a different location. *Id.* at 3:5-8; 9:48-49. *See also* Ex. 1003 at p. 125 (Table 2, Claim 7.[g]).

4. Claim 8: The memory system of claim 7, wherein the scrub trigger event includes an event disturbing stored in said at least one location in the array.

This limitation is largely identical to the limitations recited in claim 1.[a] and claim 4 discussed in Section IV.F.2.b and Section IV.F.3, respectively above, and the same analysis applies. *See also* Ex. 1003 at pp. 126-127 (Table 2, Claim 8).

H. Challenge #3: Claims 9 & 10 of the '835 patent are rendered obvious by U.S. Patent No. 6,151,246 ("So") in view of U.S. Patent No. 6,396,744 ("Wong").

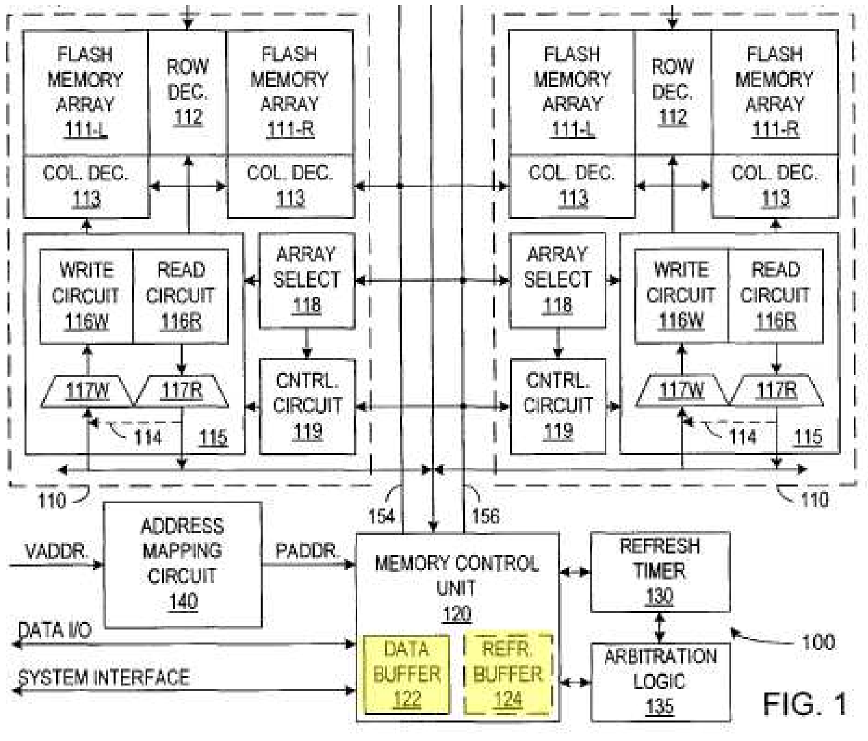
1. Overview of Wong

The combination of So and Wong renders claims 9 & 10 obvious. Wong was filed on April 25, 2000 and issued on May 28, 2002. Therefore, Wong is prior art against the ’835 patent under pre-AIA 35 U.S.C. §§ 102(a), 102(b), and 102(e).

Similar to So, Wong describes a method to refresh contents of a memory cell to prevent errors resulting from changes in threshold voltage. Ex. 1006 at 1:59-67. These read/write disturb errors can be caused by voltage leakage from operation occurring on neighboring cells. *Id.* at 1:24-26. As with So, Wong seeks to overcome these read/write disturbs by employing a refresh operation which periodically reads and re-writes content of all or selected portions of the memory. Wong’s refresh operation starts in response to a refresh timer but the operation can be delayed until other pending operations are first completed. *Id.* at 6:1-8. The refresh process then “reads the data from a sector, stores the data in buffer 122, erases the sector, and re-writes the data from buffer 122 back into the erased sector.” *Id.* at 6:44-46. Alternatively, Wong teaches that the data can be written back to another sector. *Id.* at 7:3-5. These aspects of refresh operations were also discussed in So and were generally well-known in the art. Ex. 1003 at ¶ 49; *See also* Section F above.

Wong likewise realizes that the refresh operation may prevent other higher priority host memory operations from continuing during the refresh operation. Ex.

1006 at 6:1-8. Wong overcomes this hurdle by disclosing the use of an additional buffer which allows for data input/output during (i.e., in parallel with) the refresh operation: "Alternatively, memory 100 can include an I/O buffer data buffer 122 and a refresh buffer 124 to permit data input or output during a refresh operation. For example, the arrays 111 other than the array containing a sector being refreshed, can be accessed normally through I/O buffer 122 during refresh of the requested sector." *Id.* at 6:57-62. Using this method, the system can program (or read) data to (or from) another memory location while the refresh operation is on-going. This helps make the system more efficient because the system's access to the memory array is not held up by the refresh operation. Ex. 1003 at ¶ 59.



Ex. 1006 at FIG.1, excerpted, with the two buffers annotated in yellow.

In effect, a person of ordinary skill in the art would appreciate that the teaching in Wong is that having a single buffer serve as both the input/output and the refresh buffer fails to fully utilize the ability to conduct parallel operations across different parts of the array, because the one buffer becomes the bottleneck at the "gate" to the memory system. Ex. 1003 at ¶ 113. Having a dedicated buffer for refresh, and an additional separate buffer for the input/output interface between the host and the memory array, would allow refresh operations to occur in parallel with other host memory access operations, and is a clear improvement over the state of the prior art. *Id.*

So recognized this same problem of having to delay refresh operations until the memory is "inactive for a period of time," because refresh operations are lower priority than other system operations. Ex. 1005 at 2:66-3:4; 8:65-9:5. Incorporating the use of a dedicated refresh buffer in addition to an I/O buffer as taught in Wong would have allowed So to overcome the need to delay the refresh process. Ex. 1003 at ¶ 114. One of ordinary skill in the art would recognize that applying Wong's teaching regarding the use of separate buffers (one for input/output with the host, and another dedicated for temporarily storing data being refreshed) would allow So's refresh operations and other host-to-memory operations to occur concurrently. *Id.* Even though refresh operations are lower priority, they are important to maintaining system performance, and adding

Wong's teaching would allow So's refresh operation to take place without delays. *Id.*

2. The combination of So and Wong renders obvious claim 9:

a. Claim 9.[pre]: A method of operating a flash memory cell array, comprising:

If the Commission finds that the preamble is limiting, then So discloses the features recited in this preamble.

So discloses that it "relates to non-volatile semiconductor memory" and that its disclosure is "not limited to flash EPROM of the exemplary embodiment but can be employed in a variety of memory architectures including but not limited to EPROM, E^2PROM, and flash E^2PROM." Ex. 1005 at 1:12-13; 4:41-44.

Likewise, Wong discloses "[a] conventional non-volatile memory such as a Flash memory…" Ex. 1006 at 1:11-13; FIG.1.

So discloses an array of flash memory cells. So discloses that "[i]n an exemplary embodiment of the invention, memory 100 is a flash EPROM, and array 140 includes hundreds or thousands of rows or columns of N-channel floating gate transistors (memory cells) organized into independently erasable sectors." *Id.* at 4:17-22. *See also, id.* at 2:58-61; 3:23-25; 4:15-17; Figure 1.

Likewise, Wong discloses that "[m]emory 100 includes multiple memory banks 110 containing memory arrays 111. In FIG. 1, each memory bank 110 contains two memory arrays 111. Alternatively, each memory bank 110 could include a single memory array 111 or more than two memory arrays 111. Each memory array 111 includes rows and columns of memory cells. Each memory cell can be a conventional Flash memory cell (e.g., a floating gate transistor), an EEPROM cell, or an EPROM cell." Ex. 1006 at 4:9-12.

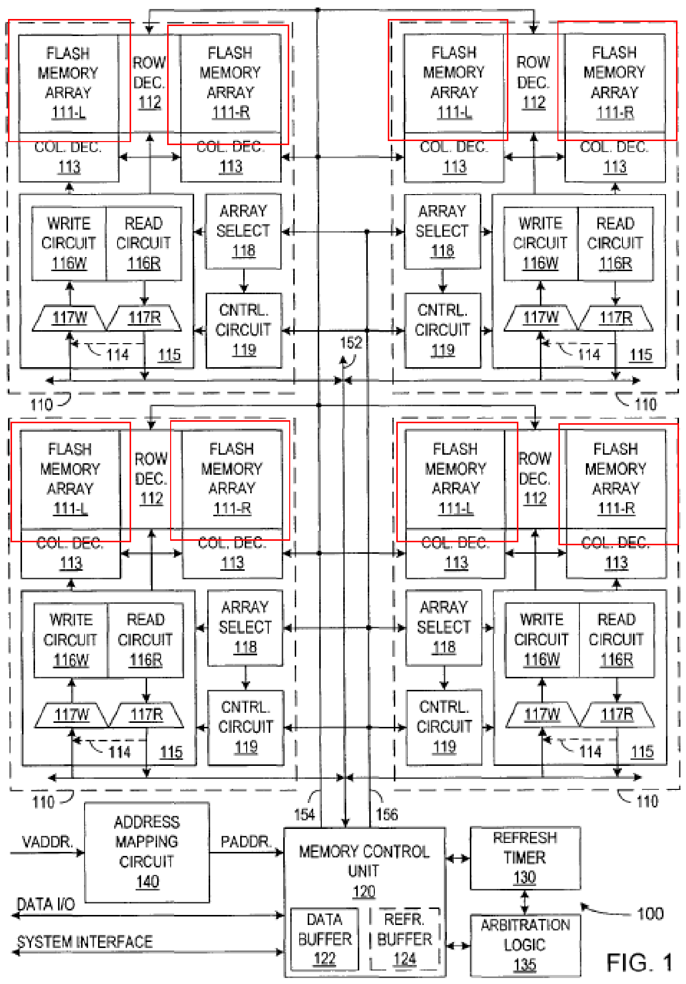


FIG. 1

Ex. 1006 at Fig. 1, annotation added.

Under the broadest reasonable interpretation of "flash memory cell array" as discussed in Section IV.E.2 *supra*, the "array" can include the group of all memory cells across the different memory banks disclosed in Wong, including *e.g.* the cells outlined in red in Fig. 1 above. *See also* Ex. 1003 at pp. 133-138 (Table 3, Claim 9.[pre]).

b. Claim 9.[a]: identifying when a scrub trigger event has occurred for data stored in at least one location of the array,

So discloses that the controller identifies when a scrub trigger event has occurred for data stored in at least one location of the array. So describes that threshold voltages of cells can drift overtime and create errors. Ex. 1005 at 1:56-2:2. So discloses detecting these threshold voltage drifts in the memory cells using the error detection circuit. "A memory cell detected as having a threshold voltage in one of the forbidden zones indicates a data error that can be automatically corrected during a read or reported as an error for subsequent correction and refresh procedure." *Id.* at 4:11-14. So also discloses various trigger events that signal the need for a refresh process. Such triggers can include "…time since the last refresh of the sector, the number of erase/write cycles associated with a sector, or even the threshold voltage read during the last read cycle." *Id.* at 8:31-36. The '835 patent discloses similar parameters used to determine scrub trigger events (*see e.g.,* Ex. 1001 at 20:1-20; 20:66-21:8). So further discloses that the refresh process may be periodic and "[s]uch periods are typically on the order of at least weeks or months for current non-volatile memory but more frequent refreshes having a period of on the order of a day or less can be used." Ex. 1005 at 10:35-38. *See also*, *id.* at 8:45-55; 10:21-51; Ex. 1003 at pp. 138-140 (Table 3, Claim 9.[a]).

c. Claim 9.[b]: reading the data stored in said at least one location in the array,

So discloses reading the data stored in the array. After the scrub trigger event identifies a memory location, the "refresh controller 620 reads the identified sector." Ex. 1005 at 9:6-7. *See also, id.* at 2:29-34; 9:33-35.

Wong also discloses that "a refresh operation reads the content of each memory cell and writes the read value back into the same or a different location in the memory." Ex. 1006 at 1:59-61. In one example, data can be read from one sector and written to another sector, or alternatively, data can be read from one sector, written to a buffer while other operations are performed, and written back into the same sector. *Id.* at 2:8-12; 2:36-41; 2:58-67; 3:5-7; 3:23-32; 6:37-46. *See also* Ex. 1003 at pp. 140-142 (Table 3, Claim 9.[b]).

d. Claim 9.[c]: thereafter determining whether there are any errors in the data read from said at least one location in the array

So discloses thereafter determining whether there are any errors in the read data. The data is read, and then an error detection circuit is used to determine whether there are any errors in the data. The error detection circuit determines errors "by finding a threshold voltage in a zone forbidden to threshold voltages corresponding to data or from an error detection code stored when the threshold voltage was written." Ex. 1005 at 3:12-15. So discloses that refresh operations "read the content of a data sector into buffer 610, correct the data, and write data from buffer 610 back to memory array 140." *Id.* at 9:33-35. As would be understood by POSITA, correcting the data would first involve a determination of whether there are any errors in the data read from that sector. So discloses using error correction codes for determining these errors in the stored data. *Id.* at 9:34-43. *See also, id.* at 2:58-63;3:19-22;8:45-55; Ex. 1003 at pp.142-145 (Table 3, Claim 9.[c]).

e. Claim 9.[d]: correcting any errors in the data read from said at least one location in the array

So discloses correcting any errors in the data read during the refresh operation: "data errors can be corrected by reading the data values from a sector of memory cells including one or more memory cells containing one or more errors, erasing the sector, and then programming the sector with corrected data values." Ex. 1005 at 2:29-34. In the refresh operation, after making the determination of an error, "the control circuit writes a corrected threshold voltage that corrects the error that the error detection circuit detected." *Id.* at 3:2-4. Additionally, So discloses a data correction circuit having an error correction code to identify the corrected threshold voltage. *Id.* at 9:37-43. The timing of the error correction in the overall refresh process can be modified as desired. *Id.* at 9:44-47. *See also* Ex. 1003 at pp. 145-146 (Table 3, Claim 9.[d]).

f. Claim 9.[e]: temporarily storing the corrected data

So discloses temporarily storing the corrected data in a buffer 610. "To perform a refresh operation on an identified sector, refresh controller 620 reads the identified sector and temporarily stores the results in a buffer 610 while the identified data sector is erased. Buffer 610 can be on-chip or off-chip volatile memory such as SRAM or DRAM or non-volatile memory that stores digital or analog data." Ex. 1005 at 9:6-11. *See also, id.* at 9:18-24; 9:33-35. In its disclosure of various embodiments, So discloses that the buffer can be used to temporarily store corrected data. As shown in Figure 6, in one embodiment, data is read from the array ("Read Circuit 650"), the data is then corrected ("Data Correction 615"), and the corrected data is then stored in the buffer ("Buffer 610"), as noted by the directionality of the arrows. Ex. 1005 at Annotated Fig. 6. *See also,* Ex. 1005 at 9:44-47.

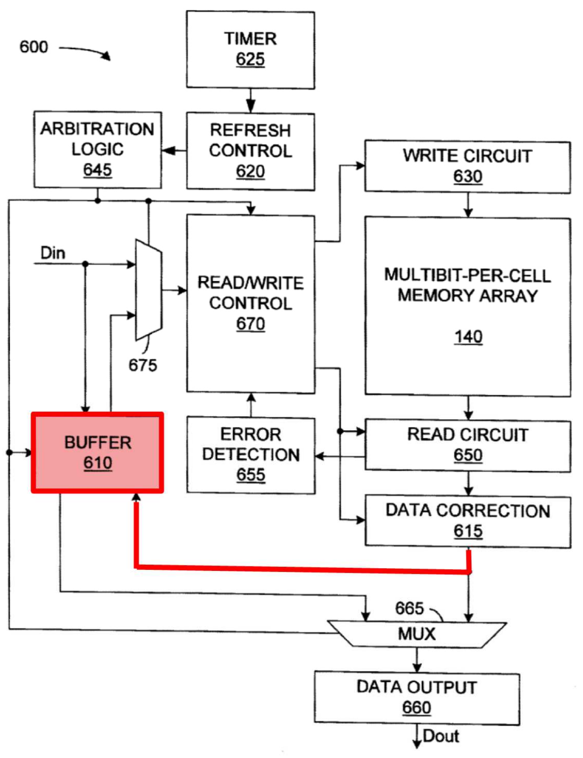


FIG. 6

Wong also discloses using a buffer to temporarily store data during the refresh operation. "The memory management unit optionally includes a data buffer that the memory management unit uses for data externally transferred and for the data read during the refresh operation for writing back into the memory cells." Ex. 1006 at 2:30-34. *See also*, *id.* at 6:37-46.

In another exemplary aspect, Wong further discloses using a dedicated "refresh buffer 124 to permit data input or output during a refresh operation. For example, the arrays 111 other than the array containing a sector being refreshed, can be accessed normally through I/O buffer 122 during refresh of the requested sector." *Id.* at 6:57-62. The use of additional buffers taught in Wong, allow for simultaneous operations to take place in the memory system. *See* Ex. 1006 at Fig. 1, excerpt and annotated.

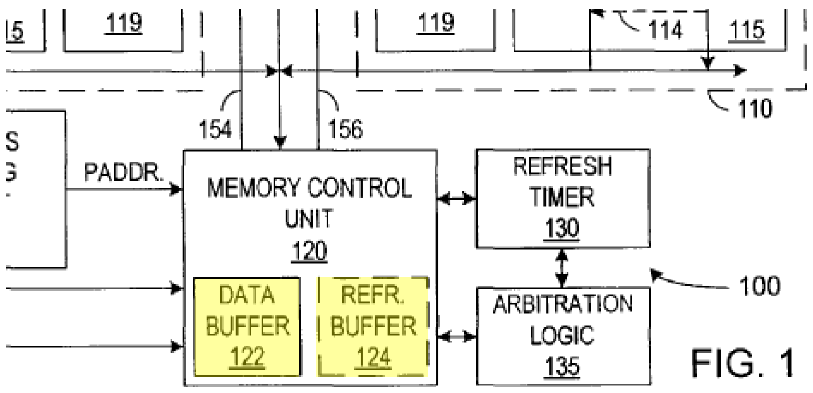


FIG. 1

One of ordinary skill would recognize that using additional buffers allows for more functionality in the system (*e.g.*, multiple operations running simultaneously because the buffers are each dedicated to different function). Ex. 1003 at ¶ 113. It would have been obvious to incorporate the teaching of multiple buffers from Wong into So. *Id.* at ¶ 114. The system of So has to choose between executing the refresh operation, or to delay the refresh and allow for higher priority operations to occur. Ex. 1005 at 8:65-9:2; 10:30-46. The motivation to combine So with Wong is clear because with such a combination, So's refresh operations would not need to be delayed. Therefore, it would have been obvious to one of ordinary skill in the art to combine the teachings of So and Wong. *See also* Ex. 1003 at pp. 146-151 (Table 3, Claim 9.[e]).

g. Claim 9.[f]: while the corrected data remain stored, programming or reading other data to or from at least locations within the array other than said at least one location

So in combination with Wong disclose programming/reading other data to/from locations in the array other than said one location while the corrected data remains temporarily stored in the buffer. So discloses that the refresh operations can take place at a convenient time – when system resources are not tied up in other operations. Ex. 1005 at 2:66-3:2. Further, during the refresh operation the refreshed data is generally written back to the same original location (unless there is a need for *e.g.* wear leveling). *Id.* at Abstract; 9:48-10:4. *See also* Ex. 1003 at ¶¶ 36-46. Writing the refreshed data back into the same original location from which it was read first requires a sector erase operation. Ex. 1005 at 9:24-29. This erase-before-write constraint is well known in the art for flash memory. Ex. 1003 at ¶¶ 24-25. However, erase operations are known to be much slower than programming or read operations. *Id.* at ¶ 57. Due to such long erase times, it was generally known in the art to postpone or suspend erase operations and allow the system to conduct *e.g.* the faster and higher priority read operations. *Id.* at ¶ 58. A POSITA would have been motivated to allow other faster and more important host operations to be performed while the original sector is being erased (during which the data has been temporarily stored in the buffer, waiting to be rewritten to the original sector when the erase operation is complete). Furthermore, So expressly teaches that during the refresh operation, the arbitration logic can temporarily store data and an address in the buffer and can reroute data accesses to output data from the buffer if is contains data corresponding to a read address. Ex. 1005 at 10:8-20. Accordingly, So expressly teaches this general desire to allow read operations to other parts of the array during a refresh operation.

Wong expressly discloses the means to allow programming/reading other data to/from locations in the array other than said one location while the corrected data remains stored in the buffer. Wong discloses that "the arrays 111 other than the array containing a sector being refreshed, can be accessed normally through I/O buffer 122 during refresh of the requested sector." Ex. 1006 at 6:59-62. Wong also discloses performing parallel operations in general: "When the selected and following sectors are in different arrays, reading, writing, and erasing can be performed in parallel." *Id.* at 2:61-64. Such parallel operations allow the system to program or read other data to or from other locations within the array. Specifically, Wong teaches a memory architecture having both an I/O data buffer, as well as a separate refresh buffer, allowing the host to access the memory and perform operations through the I/O buffer in parallel with a refresh operation (such as while the refresh buffer is temporarily storing refresh data). *Id.* at 6:54-7:2. *See also, id.* at 7:19-22; 8:44-47; 9:3-11; 10:14-16.

It would have been obvious to combine So with the teachings enabling parallel memory access as disclosed in Wong. Ex. 1003 at ¶ 114. For example, So discloses "[t]o perform a refresh operation on an identified sector, refresh controller 620 reads the identified sector and temporarily stores the results in a buffer 610 while the identified data sector is erased." Ex. 1005 at 9:6-9. A POSITA implementing this So "original sector" refresh would recognize that during the erase operation (*i.e.*, while the corrected data is temporarily stored in the buffer), the additional I/O buffer based on Wong's teachings allows the host system to access other parts of the array as desired. With the express teaching in Wong that enables this full parallelism of memory access, one of ordinary skill in the art would recognize that incorporating the teaching of Wong into So would allow the refresh operations and other operations to occur concurrently, thus improving overall system performance. *See also* Ex. 1003 at pp.151-155 (Table 3, Claim 9.[f]).

h. Claim 9.[g]: thereafter writing the corrected data into the memory cell array.

So teaches thereafter writing the corrected data into the array. After the data has been corrected and stored in the buffer, and data has been programmed/read to/from another location in the array, the corrected data is written back into the array to complete the refresh operation. "Refresh control 620 controls refresh operations that read the content of a data sector into buffer 610, correct the data, and write data from buffer 610 back to memory array 140." Ex. 1005 at 9:33-35.

Wong also teaches thereafter writing the corrected data into the array. After the data has been refreshed, the corrected data is written back into the array. "[A] refresh operation reads the content of each memory cell and writes the read value back into the same or a different location in the memory." Ex. 1006 at 1:59-61. *See also, id.* at 2:8-12; 2:36-41; 2:58-67; 3:3-7; 3:23-32. Wong teaches using a buffer in a refresh operation to temporarily store the data before writing it back from the buffer into the erased sector. *Id.* at 6:37-46. *See also id.* at 11:56-12:4. *See also* Ex. 1003 at pp.155-158 (Table 3, Claim 9.[g]).

3. The combination of So and Wells renders obvious claim 10: The method of claim 9, wherein the scrub trigger event includes an event disturbing the data stored in said at least one location in the array.

So and Wells render obvious claim 9, as shown above. So further discloses a scrub trigger event that includes an event disturbing the data. The error correction circuit of the controller detects errors in the threshold voltages (i.e., events disturbing data) and signals for a refresh operation. When the data is disturbed, it is flagged as an error for subsequent correction and refreshing. So states that "over time, charge tends to leak from the floating gates of memory cells and change the threshold voltages of the cells." Ex. 1005 at 1:56-60. Such "[c]hanges in the threshold voltage are a problem because the state of the memory cell and the data value stored in the memory cell can change and create a data error." *Id.* at 1:64-2:2. The error detection circuit detects errors in the threshold voltage and signals for the refresh operation to occur in the sector. *Id.* at 3:12-15; 4:11-14; 8:45-52. Then "a data correction circuit 615 identifies any threshold voltages that are not in an allowed state 420 and replaces such threshold voltages with the correct one of target levels VWl to VWn. If error detection and correction codes are used instead of or in addition to forbidden zones, data correction circuit 615 can also use such codes to determine or confirm the correct data value or correct threshold voltage." Ex. 1005 at 9:37-43. *See also* Ex. 1003 at pp. 158-160 (Table 3, Claim 10).

V. CONCLUSION

Claims 1, 4-10, 13, 15, 17, 18, 20-23, 25, and 27 of the ’835 patent are rendered obvious by the prior art as discussed above.. There is a reasonable likelihood that Petitioner will prevail as to each of the claims. Petitioner respectfully requests that the Patent Office initiate an *inter partes* review of claims 1, 4-10, 13, 15, 17, 18, 20-23, 25, and 27 that it find those claims invalid in light of the prior art, and that it cancel those claims.

Respectfully submitted,

Dated: September 17, 2015

/s/ Brent Yamashita
Brent Yamashita
Registration No. 53,808

DLA PIPER LLP (US)
2000 University Avenue
East Palo Alto, CA 94303-2214
Telephone: 650.833.2348
Facsimile: 650.833.2001

CERTIFICATE OF SERVICE

The undersigned certifies service pursuant to 37 C.F.R. 37 C.F.R. §§ 42.6(e) and 42.105(b) on September 17, 2015, by UPS Overnight Delivery of a copy of this Petition for *Inter Partes* Review and supporting material at the following correspondence address of record for the ’835 Patent:

Davis Wright Tremaine LLP – Sandisk Corporation
IP Docketing Department
1201 Third Avenue, Suite 2200
Seattle, WA 98101

Dated: September 17, 201

/s/ Brent K. Yamashita
Brent K. Yamashita
Registration No. 53,808

DLA PIPER LLP (US)
2000 University Avenue
East Palo Alto, CA 94303-2215
Telephone: 650.833.2348
Facsimile: 650.687.120