IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF WISCONSIN

- -

SANDISK CORPORATION,

OPINION and ORDER

Plaintiff,

v.

10-cv-243-bbc

KINGSTON TECHNOLOGY CO., INC. and
KINGSTON TECHNOLOGY CORP.,

Defendants.

- -

This lawsuit is the third such action brought by plaintiff SanDisk Corporation against defendants Kingston Technology Co., Inc. and Kingston Technology Corp. for infringement of plaintiff's patents related to flash memory technology. In this lawsuit, at issue are six patents belonging to plaintiff: United States Patents Nos. 7,397,713 ('713 patent); 7,492,660 ('660 patent); 7,657,702 ('702 patent); 7,532,511 ('511 patent); 7,646,666 ('666 patent); and 7,646,667 ('667). These patents all share specifications with patents brought in two previous lawsuits, Cases Nos. 07-cv-605-bbc and 07-cv-607-bbc. Now before the court are defendants' motion for summary judgment of noninfringement, plaintiff's motion to strike portions of that motion and to strike portions of a rebuttal expert report, motions from both parties for leave to file additional briefing on the motion to strike, and defendants' motion for leave to file corrected versions of expert reports. I will grant the motions for leave to file additional briefing but deny the motion to strike and grant the motion to file corrected versions of expert reports. I will also grant defendants' motion for summary judgment of noninfringement in full. As a result, it is not necessary to address the question of damages, and in my discretion I decline to exercise jurisdiction over the remaining invalidity claims and will thus deny the motion for summary judgment of invalidity.

I. MOTION TO STRIKE AND RELATED MOTIONS

Plaintiff seeks to strike portions of defendants' rebuttal expert report and portions of defendants' motion for summary judgment on the ground that defendants failed to disclose certain witnesses to plaintiff. (The parties also seek to supplement their briefing on this matter; those motions will be granted and their supplemental briefs considered.) In particular, defendants' expert, Melvin Ray Mercer, states in his rebuttal report that he "considered and relied on telephone interviews conducted with certain engineer personnel at the controller companies." Dkt. #269-1, ¶ 16. However, he did not identify these personnel and defendants allegedly delayed responding to requests for this information until it would not have been practicable to depose them. In addition, defendants' counsel refused to accept service for any subpoena related to these individuals or one of the controller companies.

Although it appears that defendants failed to comply with the letter of Fed. R. Civ.

P. 26 by failing to properly disclose these individuals in advance, their actions are far from troubling when examined carefully. Defendants provided the names within two days of plaintiff's request for them, and of the six individuals not disclosed, plaintiff had deposed four of them extensively in the previous action, the discovery for which plaintiff successfully sought to consolidate with this action. As for the remaining two individuals, Mercer interviewed them in response to a recent inclusion by plaintiff in *its* expert report two weeks earlier addressing reverse engineering performed on 3S products.

It is understandable that defendants' counsel refused to accept the service of subpoenas for the individuals or even the controller company being served a subpoena. Defendants' counsel did not have the authority to accept service of the subpoenas for the six individuals. Although it represented the controller company in other matters, it was concerned that accepting service of the subpoena would waive the company's right to challenge personal jurisdiction. In short, defendants' actions do not suggest gamesmanship.

Along these lines, although Mercer indicates that he relied on these conversations, with only one exception, plaintiff fails to identify any information he may have used that they would have liked to have challenged or explored but could not. For the most part, the parties simply do not dispute the particular features of the accused products; plaintiff does not suggest in any instance that Mercer describes the features of the products improperly as a result of one-sided conversations. The one exception to this is Mercer's reliance on testmony by Jackie Hsu that, despite the presence of code showing that Phison controllers use GigaDevice SRAM, they do not actually do so. As to that testimony, striking it is unnecessary: Hsu's testimony does not provide a ground for granting summary judgment because it simply creates a factual dispute between the parties as to whether the controllers use that SRAM. Moreover, as explained below, that issue becomes moot because the matter of noninfringement is resolved on the absence of other limitations in the claim.

Under these circumstances, striking Mercer's report or the declarations of the interviewees or any portion of defendants' motion for summary judgment is unwarranted. Plaintiff's motion to strike will be denied.

II. DEFENDANTS' MOTION FOR SUMMARY JUDGMENT

Before turning to the facts for the motion for summary judgment, it is necessary to address plaintiff's argument that most of defendants' facts must be disregarded. Plaintiff points out that defendants failed to submit a declaration or affidavit from their expert, Mercer, when submitting their motion for summary judgment. Instead, defendants simply submitted Mercer's rebuttal expert report and cited that directly in support of many of their facts. Defendants challenge whether the expert report itself is inadmissible, pointing out that Mercer states in the report that he would testify in accordance with his statements found therein. However, it is not necessary to decide whether this submission suffices to support a motion for summary judgment. Defendants have moved for leave to file corrected versions of the expert reports that include declarations by the respective experts and plaintiff fails to identify any prejudice that would warrant denying such a motion. The corrected versions are adopted. They suffice to support defendants' factual propositions.

UNDISPUTED FACTS

A. Converting Logical Address from Host to Physical Address for Flash Memory

Logical Block Addresses, or LBAs, are the logical addresses received from the host. In the accused systems, each LBA from the host represents one 512-byte data sector. Depending on the size of a page in a flash memory system, an LBA could refer to a single page, multiple pages or a portion of a page. Generally, controllers must convert the logical address received from the host into a new address that corresponds with where the data will be written to the flash memory. Logical block numbers (LBNs) and logical page offsets or page numbers are logical addresses associated with specific physical locations in flash memory where data can be stored. For example, a logical block number points to a particular physical block of memory in the flash device. A Physical Block Number (PBN) identifies a specific physical block in the flash memory.

1. Phison

Products incorporating Phison USB controllers divide flash blocks into "zones" according to the total number of blocks within the flash. Phison's Flash controllers use Mother Blocks to store most user data. A Mother Block is a block that is completely filled with original user data. A Child Block also stores user data. When all of the pages in a Child Block have been written, that Child Block becomes a Mother Block. The Phison USB controller uses a record to determine a logical block number by dividing the logical address received from the host system by one or more constant numbers. The logical block number helps determine which group, or zone will be used. The controllers determine zone by using a set of three constants, X_ZonePerDevice, XdeviceLBlock and X_Flash_number. These three constants store the number of zones on each chip, the number of LBNs in each zone and the number of chips within the memory system, respectively. In Phison controllers, the ending address for each zone can be calculated with the formula Ending Address = (ZoneNumber * X_DeviceLBlock)-1.

Once the LBN has been calculated, the controller compares the LBN against a list of LBNs for which update blocks have been associated. It then uses a "LBN-to-Virtual Physical Block Table" or L2VP table, to identify the correspondence between each LBN and its corresponding Mother Block. Each zone for the USB controller has a separate L2VP table, stored in a block that does not include any user data. The L2VP table lists the virtual-physical block numbers that correspond with certain logical block numbers. For LBNs having an associated update block, additional mapping tables such as the "RN-Table" will be used to keep track of updates made to logical pages originally stored in the Mother Block. The RN-Table lists physical page addresses that correspond with logical page numbers where updated data has been placed. Where updated data has not been placed, the table lists an invalid page offset, indicating that the page in the Mother Block is still valid for that page. The RN-Table indicates both whether a more up-to-date version of the data exists than the version stored in the Mother Block and provides the physical page offset of the most up-to-date version corresponding to a given logical page number. The entries in the LBN-to-VPN and RN-tables act as pointers to physical blocks and pages. The controller also loads a bad block replacement table. The L2VP tables, the RN-Table and the bad block replacement table are separate from the record containing the constants used to determine the zone. Each of these data structures is treated as a unit and stored in the controller SRAM for separate operations.

The Phison controllers rebuild an L2VP table for a given zone each time the zone is accessed following power-up by reading firmware bytes and logical block numbers stored in the reserve fields of the blocks. When a given L2VP table is being used, it is in the SRAM of the controller. After reading a given L2VP table, the controllers maintain a copy of the table in SRAM during operation so that it may be quickly accessed in response to a host command. However, when the table is no longer being used or a table for another zone is being used, the table not in use is stored in a flash memory block by an operation called "SaveMappingTable."

Accused products containing Phison CF controllers use a similar method to that of products with Phison USB controllers to determine a block physical address from a logical address. These CF controllers also have a separate mapping table for each zone in the memory system. The CF controllers use an Hpage variable, a VIR unit and Fzone variables. The Fzone variables determine which zone mapping table should be used. Each zone mapping table is treated as a unit and separately loaded into SRAM as part of the process of determining the block address within the zone once the zone has been determined. The CF controllers also use a bad block replacement table as part of the process of determining a block address within a zone. This table is also treated as a separate unit.

The physical block number stored is not the physical address supplied by the controller to the flash memory device. The flash memory devices require an address in the form of separate row and column addresses. Converting the physical block number into the actual row and column addresses requires either additional computation, additional data structures or both. The Phison controllers use a hardware conversion or translation to obtain the column address and row address.

2. Skymedi

Accused products with Skymedi controllers divide the physical address space of a memory system into multiple groups called "windows." A Windows Table stores the number of logical blocks in each window. This table is treated as a unit. The information found in the Windows Table could be used to calculate the ending range of logical block addresses that would be assigned to the window. Each window is associated with a particular group of physical blocks in the non-volatile memory. User data is stored in blocks called W-Blocks, C-Blocks and R-Blocks. An "H-Block" is also assigned to each window, where those H-blocks are stored in designated physical blocks within the Flash memory. Blocks capable of storing W-, C- or R-Blocks are also capable of storing an H-Block instead. A W block stores original data, and is a block that is completely full of user data stored in a sequential manner.

Products using a Skymedi controller performs Boolean operations on the bits of a logical block address received from a host system to determine a logical block number, logical page number and logical partition number for the address. The logical block number is used to determine the appropriate window to access based on the receiving address. Each window used by Skymedi controllers includes an information block called an "H-Block." This includes mapping information called a "Logical Block ID table" or a "link table" and data structures called "Rlink" and "Clink." Each mapping table is treated as a unit, such as when the table is loaded into the controller's SRAM.

After a logical block number is determined, the controller finds the logical block number in the Rlink table to be able to find the corresponding R and W blocks. The Rlink table keeps track of updates made to logical pages originally stored in the W Block which are now stored in an updated R Block by listing physical page addresses that correspond with logical page numbers where updated data has been placed and providing an invalid page address for logical page numbers where no updated data has been placed. The Rlink Table indicates both whether a more up-to-date version of the data exists than the version stored in the W Block and provides the physical page offset of the most up-to-date version corresponding to a given logical page number. The Logical Block ID table identifies the physical block that corresponds to a particular logical block ID. The controller knows which logical block ID corresponds to a given physical block information according to the location in which that physical block information is stored in the controller's SRAM. The link table for a given window or zone is stored in the controller's SRAM when the controller is performing an operation on that zone. The Skymedi controller also uses a Defect Replacement Table to identify replacement physical addresses for bad blocks, and in some situations uses a table known as the RBLK_TABLE.

The physical block number stored in the tables in Skymedi controllers is not the physical address supplied to the flash memory device. The flash memory devices require an address in the form of separate row and column addresses. Converting the physical block number into the actual row and column addresses requires either additional computation, additional data structures or both.

3. Silicon Motion

Silicon Motion controllers divide the physical address space of a memory system into multiple zones, also called management units, memory units or "MU." Each MU is a group of flash blocks. The controllers use different types of user data blocks including Mother blocks, Child blocks and FAT blocks.

The Silicon Motion controllers convert the LBA received from the host into an "H block" number, which is then converted to correspond to its zone to determine which block link table to use. The controller determines the zone by accessing a record known as the LOADDATBLK, which stores the number of logical blocks in each zone. This information could be used to calculate the ending address of logical block addresses that would be assigned to each zone. The LOADDATBLK is treated as a unit. After determining the zone, the controller uses a block Link Table to determine the corresponding block address within the MU. A Link Table is assigned to each zone, and stored in designated physical blocks within their Flash memory. The Link table lists the physical block numbers that correspond to certain logical block numbers. Each Link Table is treated as a unit and is separately stored in the controller's RAM. The block that includes the Link Table does not include any user data and the blocks that store user data do not store logical to physical conversion information.

A FAT table, or "page link table," keeps track of updates made to logical pages originally stored in the Mother Block which are now stored in FAT Blocks by listing physical page addresses that correspond with logical page numbers where updated data has been placed and providing an invalid page address for logical page numbers where no updated data has been placed. The FAT Table indicates both whether a more up-to-date version of the data exists than the version stored in the Mother Block and provides the physical page offset of the most up-to-date version corresponding to a given logical page number. The entries in the Link and FAT tables act as pointers to physical blocks and pages, respectively.

Each physical block is programmed sequentially; random page address programming is not allowed. The physical block number stored in the tables in Silicon Motion controllers is not the physical address supplied by the controller to the flash memory device. The flash memory devices require an address in the form of separate row and column addresses. Converting the physical block number into the actual row and column addresses requires either additional computation, additional data structures or both.

B. Defective Blocks

In the accused products, if a block is deemed to be bad, then no other data is programmed into the block after a bad block marker is programmed into it. For example, an accused product using a flash memory device described in the Flash Datasheet for K9XXG08UXM will check the I/O 6 bit to determine whether the page operation is "ready" or "busy." If there is an error during programming of a block of a flash memory device described in the Samsung Flash Datasheet for K9XXG08UXM, the block will be marked as having failed to program and the data stored in that block will be copied over into another replacement block. An "invalid block" table or "other appropriate scheme" will indicate that the block is not to be erased or programmed to. That marker informs the controller that the block should no longer be used to store data. The bad block markers of the accused products do not indicate where any defects exist within the block or what type of defect is present. None of the products accused of infringing the '660 or '713 patents are capable of remapping individual flash memory cells to other flash memory cells. None of the flash memories used in the accused products can remap defective flash memory cells on a cell-by-cell basis.

Plaintiff's expert, Rhyne, stated that it is his opinion that storing defect data in user data blocks "would not achieve any of the stated advantages of the '667 inventions." In particular, Rhyne explained that "defect data is not bound to change each time the corresponding user data changes; it only changes when a block or bit line becomes defective.

Thus, storing defect data in the same block as user data would not achieve the stated advantage of avoiding rewriting physical characteristics data unnecessarily." Dkt. #165-13, ¶ 294. He adds that "[w]hile it is unclear exactly which physical characteristic data cannot be stored in the same blocks as user data, it is my opinion that defect data cannot be stored in the same blocks as user data." *Id.*, ¶ 295. Rhyne made these statements in the context of challenging an invalidity claim in light of a patent that included "bad sector" information in every user data sector and did not preclude writing valid user data to any remaining good sectors.

1. Phison

Kingston products incorporating Phison controllers all store some type of good/bad block indicator in the same group of blocks designated for storing user data. Phison's USB controllers store information in the header bytes of designated user data blocks that identifies whether a block is good or bad. They store a "bad block marker" in the sixth firmware bite, and store a "P" in the first first byte using the function "Fill_LBAB3." In a bad block, the first byte will not be filled in with a "P." Phison's CF controllers store information in the header bytes of designated user data blocks that identifies whether a block is good or bad. In particular, the controller records certain markers in the firmware bytes of Mother Blocks and Child Blocks. If the block is good, the firmware bytes will contain an FF. If a block is deemed defective when it leaves the factory, the controller records 00 in the firmware bytes, and if a block is later identified as having a defect, the controller will mark either an F0 or a C0 in the fimrware bytes.

2. <u>Skymedi</u>

During a write or program operation, the accused products incorporating a Skymedi controller receive a logical address from a host and may determine that the page program operation "failed." The controller will transfer into a new block the valid data in the block where the page program operation failed and program the data into that new block. Once a program operation has failed, the header of the block containing defective user data will be marked with a block status byte indicating that it is a bad block.

3. <u>Silicon Motion</u>

If a page program operation fails in a product using a Silicon Motion controller, the physical block containing the page that failed to program is marked bad or "mapped out" and any valid data in the block is moved into a new block.

C. <u>Writing Data</u>

In accused products using Skymedi, Phison and Silicon Motion controllers, the offset positions of the pages storing updated data will always overlap with the offset positions of pages storing original data. The offset positions of any of the particular pages may not be the same in the updated data as the offset position of the corresponding page in the original page, but collectively, one or more of the pages storing updated data will always have an offset position identical to the offset position of one or more of the pages storing original data.

When using the multi-page program command, the accused products simultaneously write into the flash memory array until at least a full page is written into each of a plurality of blocks. A full page includes multiple sectors. For example, according to a Toshiba NAND flash datasheet, 2K bits of data are simultaneously written into each of the plurality of blocks in the "metablock." NAND Flash memory manufacturers typically require that a block in NAND memory be programmed from its lowest page to its highest page.

1. <u>Phison</u>

Phison controllers continuously write all of the data received from the host in a write request. The write operation does not end after a different sector and error correction code is written into a block in each plane. The products use a"dual-plane" write operation to simultaneously write multiple sectors in a single operation into each block of a metablock.

The amount of data received in a host write request from a system running the Windows operating system is 128 sectors.

Each of the accused products containing a Phison controller can perform partial block updates. Such updates can occur, for example, when a user opens a file from the flash memory and saves it again after making edits. When an associated host requests that a product using a Phison controller store enough data to fill a complete metablock, those data are subsequently stored into a Mother Block. When the host computer writes to a small number of sectors having LBAs within the LBA range of a Mother Block, the new data is written into a non-sequential Random Block, a Child Block. For example, page offsets 0-2 of the updated Child Block can store updated data replacing data stored in page offsets 3-5 of the original Mother Block. The NAND Flash memories used in the accused products typically have 64 or 128 pages in each block. The overwhelming majority of small updates will be programmed into pages of a Random Block that have a different offset from the physical page offset in the Mother Block that holds the superseded original data.

2. Skymedi

Skymedi controllers continuously write all of the data received from the host in a write request. Each of the accused products using a Skymedi controller can perform partial block updates. They can occur when a user opens a file on their drive and saves it again after making edits. When an associated host requests a product using a Skymedi controller to store enough data to fill a complete metablock, those data are stored in a W Block. When a host computer writes to a small number of sectors having LBAs within the range of a W block, the new data is written into a non-sequential R Block. Thus, for example, physical page 5 of an R Block can store the updated data replacing the data stored in page offset 2 of the W Block. The programming of an R Block begins at physical page 0 of a NAND block regardless of what logical page address is associated with original data and contemporary NAND Flash memories typically have 64 or 128 pages per block. The overwhelming majority of small updates will be programmed into pages of an R Block that have a different offset than the physical page offset in the W Block that holds the superseded original data.

3. Silicon Motion

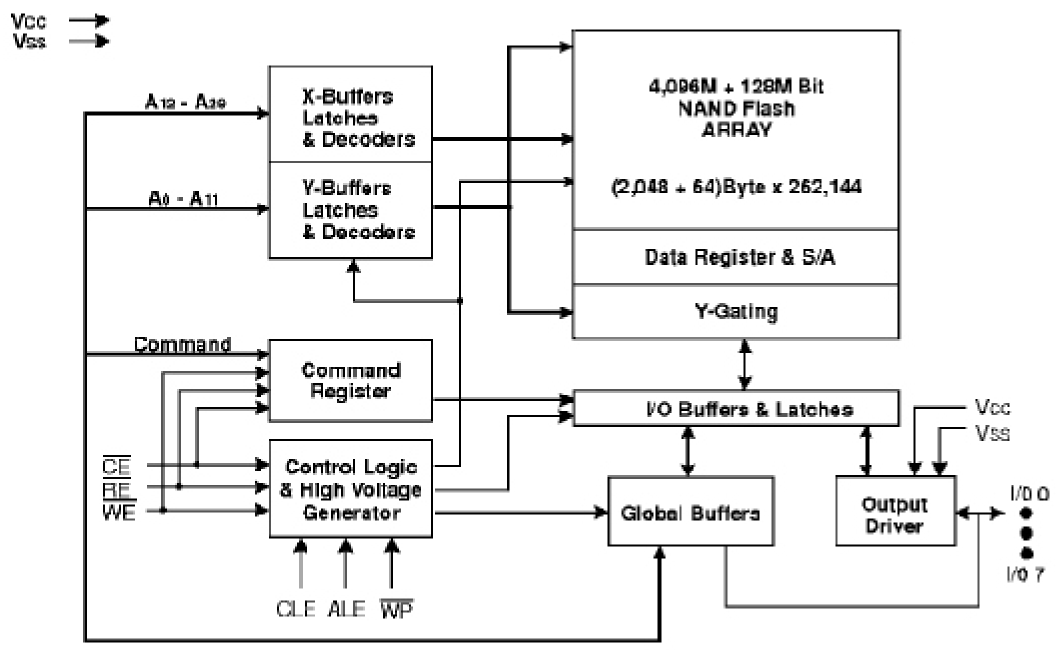
Each of the accused products containing a Silicon Motion controller can perform partial block updates. Such updates can occur, for example, when a user opens a file from the flash memory and saves it again after making edits. When an associated host requests that a product using a Silicon Motion controller store enough data to fill a complete metablock, those data are subsequently stored into a Mother Block. When the host computer writes to a small number of sectors having LBAs within the LBA range of a Mother Block, the new data is written into a non-sequential FAT Block. In the case in which a

Mother Block is associated with LBA values of 0 through 511 and a write command is received with LBA 4 and a transfer length of 8, but no FAT block yet exists, a new FAT block will be used to store LBA 4 through 11. The new data could be stored in pages 0 and 1 of the FAT Block.

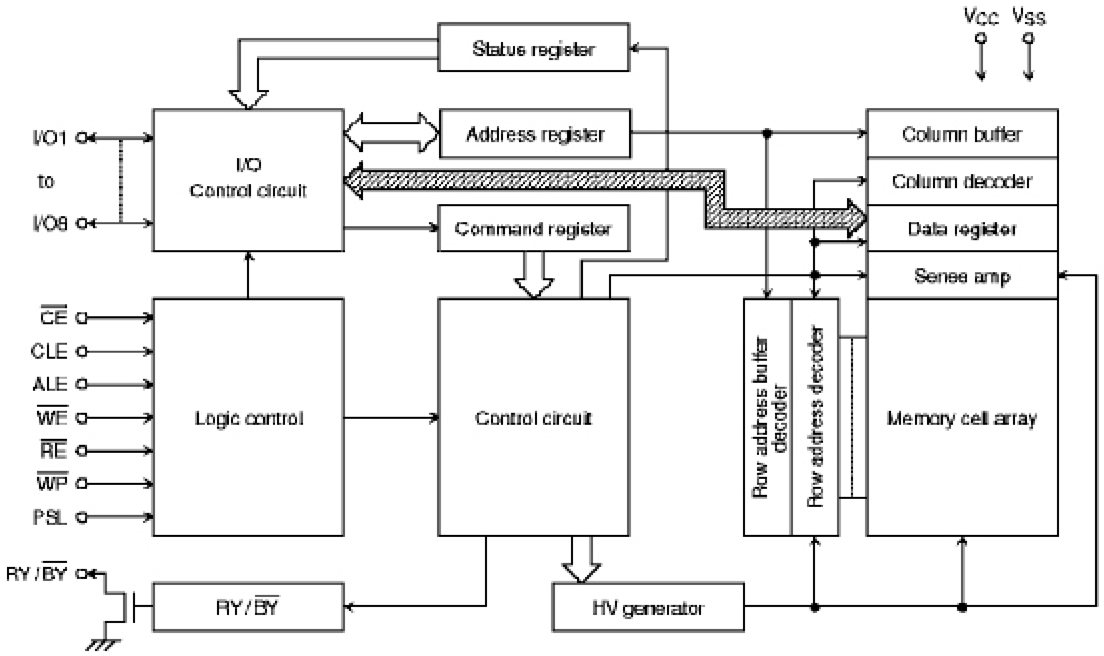
D. Circuitry

In accused products with either Samsung or Toshiba flash memory, there are several circuits between the memory controller and the blocks of flash memory. Products with Samsung flash memory are configured as follows.

Figure 1-1. K9G4G08X0A Functional Block Diagram


Vcc
Vss
A12 - A29
A0 - A11
X-Buffers Latches & Decoders
Y-Buffers Latches & Decoders
4,096M + 128M Bit NAND Flash ARRAY
(2,048 + 64)Byte x 262,144
Data Register & S/A
Y-Gating
Command
Command Register
I/O Buffers & Latches
Vcc
Vss
CE
RE
WE
Control Logic & High Voltage Generator
Global Buffers
Output Driver
I/O 0
I/O 7
CLE ALE WP

Products with Toshiba flash memory are configured as follows.



The images show "decoding circuitry" between the point at which a Physical Block Number (PBN) is supplied to the products and the flash memory blocks. The addressing information sent between the memory controller and the Flash memory is always in the form of a physical block number that identifies a specific physical block within the memory. The decoding circuitry found between the controller and the flash memory blocks decodes each physical block number supplied by the controller into a set of row and column signals identifying a specific physical block.

Given the large number of blocks of storage elements in the Flash memories used in the accused products, there must be some form of address decoding between the source of a Logical Block Number (LBN) and the specific physical block that the LBN identifies and the location of that circuitry. The location of that circuitry, whether on the controller chip or flash memory chip, is not critical to the operation of the memory. In either instance, the decoding circuitry performs the same function of decoding a given PBN so as to supply data and operational voltages to a specific block. Address decoding on a controller is accomplished through logic circuits that cause or provide connections to a specific physical block when given signals representing a specific LBN. Either decoding addresses on a controller or decoding them on the chip will cause the data to be programmed into or read out from the appropriate blocks of the Flash memory.

A direct connection between a controller and memory blocks requires the controller to be programmed to work with those specific blocks. If a controller instead uses a standard interface, it can work with multiple types of flash memory blocks.

When the priority application of the ‘702 patent was filed, persons of ordinary skill in the art recognized that it was common to have a Flash memory system with a controller and Flash memory block that connected to one or more Flash memory devices over a bus. In such a configuration, the controller chip issues certain programming commands, address information and the data to be programmed. This type of programming was common around the time the priority application of the ‘702 patent was filed. In the context of the

‘702 patent, a person of ordinary skill in the art would recognize that address decoding and other peripheral circuitry is required to allow the system to function properly.

E. Controller RAM Speed

Defendants’ expert has stated that “it would not surprise” him to learn that to read something from flash memory takes “on the order of 1,000 to 10,000 times” longer than accessing something from the RAM in a 4 2091/2092 controller. (Plaintiff also submits testimony from Dr. John Hayes, who stated in the context of discussing a method for assembling data, that assembling the data in flash memory “would be slower.” Aside from the fact that the significance of this statement is far from clear, Hayes was not disclosed as an expert and could not give an opinion on this matter. Plaintiff does not attempt to explain how Hayes could testify as a fact witness on the matter.)

(The parties dispute whether Phison controllers use GigaDevice SRAM, which operates faster than flash memory. Plaintiff points out that the Phison 2233 RTL code, a computer language describing computer hardware, indicates that the controller uses GigaDevice SRAM. Defendants submits testimony that, despite the presence of this code, neither this controller nor any other Phison controllers use GigaDevice SRAM. According to defendant, the reference to the GigaDevice SRAM found in the code is a remnant from an older project that Phison never sold to defendant.)

F. Infringement by Products Incorporating 3S Controllers

In support of its theory of infringement of the 3S controllers, plaintiff relies on tests performed on the 3S controllers by TechInsights and additional tests that plaintiff conducted. These tests involve use of a logic analyzer. Logic analyzers are typically used for digital measurements involving numerous signals and may be used to trace embedded software execution. The analyzer captures and analyzes signals using an acquisition probe, which connects to pins on the device under test. In the tests plaintiff identifies, the probe was connected to the external pins of the flash memory chip in accused products containing 3S controllers.

The test reports show that the tested product with a 3S controller, a 3S 6690 controller, contained two memory chips. The controller used the first of the two dies for the lower half of the logical address space and used the second of the two dies for the upper half of the logical address space. The controller must contain some information for determining on which die the physical address corresponding to a received logical address resides.

The test results also show that the controller copied previously programmed but unaltered portions of a page into corresponding locations in a page buffer and that the third 1070 byte increment of data included the updated user data received from the host. In one test, a host computer provided starting write address LBA 6,000,163 and associated user data to the controller. The controller translated that logical address into physical address

5C08050E00, the address for plane 0, block 15, page 5 of the flash memory chip used for testing. A subsequent read operation showed that the user data was programmed into this location.

The tests also show that when the products using 3S controllers read data, they do so without reading any other information from the flash memory, showing that the controller was able to make the logical-to-physical address translation using only information stored within its own memory. In addition, after third and fourth updates to data, in response to a request for the 3S controller to read data, the controller read only the most up-to-date version of the data.

TechInsights sent a request to an accused product containing a 3S controller to perform a total of four partial block updates, two each to LBA 6,000,163 and LBA 6,000,164. All four partial updates were programmed into block 14 ("0E00"). The updates were successively programmed into pages 3, 5, 7 and 9 of block 14. At most two of the four updates could have the same physical offset as their respective original pages. There are more than 100 pages in each block in the Flash memory chips used in the accused products containing 3S controllers.

In a complaint brought to the International Trade Commission, plaintiff alleged that certain products of defendants infringed a separate patent, relying on a test performed by Semiconductor Insights, which is now TechInsights. When the claims brought before the

ITC were brought before this court in a previous lawsuit between the parties, plaintiff's infringement claims were ultimately rejected . At no point in the ITC investigation or in the determination of noninfringement in this court did a court call into question the reliability of reverse engineering test reports or the tests used in particular.

OPINION

The parties have stipulated that only the following claims are at issue in this case:

a. independent claims 1 and 11 and dependent claims 3, 4 and 12 of the '713 patent;
b. independent claim 1 and dependent claim 3 of the '660 patent. Dkt. #215.
c. independent claims 1, 16, 24 and 33 and dependent claims 3, 9, 10, 12, 21, 22, 28, 34-38 and 44 of the '702 patent;
d. independent claim 14 and dependent claims 18 and 21 of the '511 patent;
e. independent claim 1 and dependent claims 5, 7 and 9-12 of the '667 patent;
f. independent claim 1 and dependent claims 2 and 4-8 of the '666 patent; and

In defendants' motion for summary judgment, none of the challenges are particular to the dependent claims and many address the requirements for multiple independent claims.

Before turning to the particular claims at issue, I note that I describe below the "accused products" generally in many places because their features are similar. This glosses over the fact that much about the 3S controllers is uncertain because plaintiff was not able to receive direct evidence about the internal workings of the controller but instead relied on external testing to draw conclusions, and on occasion even resorted to simply attributing features to the controller that are common to modern controllers. It is not necessary to address the shortcomings of the evidence of infringement by the 3S controller. Plaintiff does not pretend to offer any evidence of the 3S controller that would distinguish it from the other controllers at issue in this case; at most, plaintiff seeks to prove that the controller functions as the other accused controllers do. As explained below, the record shows that these other controllers are not infringing. By the same measure, the 3S controllers are not infringing either, even assuming they contain features similar to those of the other controllers at issue.

A. <u>“Defect” and “Defective” (Cls. 1 and 11, ‘713 Pat. and Cl. 1, ‘660 Pat)</u>

Defendants contend that the accused products do not infringe the asserted claims of either the ‘713 patent or the ‘660 patent because none of the products perform cell-by-cell remapping of defective blocks.

Claim 1 of the ‘713 patent requires that the claimed flash memory be “electrically programmable to store . . . address information for defect mapping.” Claim 11 of the ‘713 patent requires that the claimed controller include “a register file to store defect mapping data.” Claim 1 of the ‘660 patent requires that the claimed method of accessing a flash memory array include “determining that the logical address corresponds to a defective memory location.” According to defendants, each of these claims’ reference to “defect mapping" or "defective memory locations" requires cell-by-cell remapping of the memory unit in question before it becomes "defective."

As plaintiff points out, only with respect to the '660 patent has the court determined that a memory unit cannot be deemed "defective" until cell mapping capacity for the memory unit is reached. In the order construing claims, I considered the scope of the terms "defective memory location" ('660 pat., cl. 1) and "defective location" ('660 pat., cl. 15). I concluded that " before a memory location is deemed defective, the memory location's cell defect mapping 'capacity' must be reached, and in the meantime memory location must 'essentially' be remapped cell-by-cell using an 'error correction scheme.'" Dkt. #131, at 65. I did not construe the meaning of "defect mapping" as used in the claims of the '713 patent.

Although plaintiff is correct that the construction does not include the '713 patent, the issue was not before the court. Now that it is, I conclude that the same requirement applies to the term "defect mapping" in the '713 patent and to the term "defective" in the '660 patent. In particular, I conclude that an operation cannot be "defect mapping" until a location is deemed "defective." In light of the language found in the specification of the '713 patent, a memory location cannot be deemed defective unless the location's cell defect mapping capacity is reached. In the meantime the memory location must "essentially" be remapped cell-by-cell using an "error correction scheme."

The principal reason for imposing such a limitation on the term "defect" in the context of the '660 patent was the language in the specification, which emphasized avoiding wastage as a key aspect of the invention. *Id.* at 15-17. The '713 patent specification is identical to the specification for the '660 patent, so the same concern for avoiding wastage must shed the same light on use of "defect mapping" in the '713 patent. In addition, although the claim language at issue in the '713 patent is different from the language construed from the '660 patent, the difference is unimportant. The '660 patent refers to "defective" memory locations and the '713 patent refers to "defect mapping"; in either instance, the conclusion that a memory location cannot be deemed "defective" unless certain conditions are met would attach limitations to the claim terms at issue. In the '713 patent, the claimed storage of "address information for defect mapping" assumes that memory locations can be made defective. The specification explains that

> [a]s soon as a hard error is detected, defect mapping is used to replace the defective cell with a spare cell in the same sector block. Only when the number of defective cells in a sector exceeds the defect mapping's capacity for that specific sector will the whole sector be replaced as in a conventional disk system.

'713 pat., col. 8, lns. 18-23.

The claimed "defect mapping" address information relates to a defective memory location's mapping capacity. Allowing address information to be considered "defect mapping" information simply because the controller has deemed a location defective would mean that the claimed invention could discard a memory location of any size even after finding a single bad cell. As explained in the claims construction order, such a construction would disregard the key aspect of the invention of avoiding wastage. Dkt. #131 at 15-16. Thus, I conclude that for address information to be considered "defect mapping" information, it must relate to a "defective" memory location, meaning a location for which the flash memory's cell defect mapping "capacity" has been reached and for which the memory location has "essentially" been remapped cell-by-cell using an "error correction scheme."

Next, plaintiff argues that the court's previous construction should be "clarified" as applying only to "read" operations, not "write" operations. According to plaintiff, such a construction would be consistent with the court's ruling that the "claimed [defective] 'memory location' is not limited to a 'block.'" Plaintiff's argument is not persuasive. As defendants point out, the specification does not draw such a distinction, but rather groups read and write operations together:

> Defective cells are detected by their failure to program or erase correctly. Also during read operation, defective cells are detected and located by the ECC. As soon as a defective cell is identified, the controller will apply defect mapping to replace the defective cell with a space cell located usually within the same sector.

'660 pat., col. 7, ln. 66-col. 8, ln. 5. As mentioned above, the specification describes the "defect mapping" as relating to the memory location's "capacity" for replacing cells within it.

On a related note, plaintiff emphasizes the fact that the claimed memory location need not be a block, suggesting that this may pave a way to infringement. However, the accused products discard a block after a single bit is found defective. In the order construing claims, I concluded that such an arrangement would not satisfy the requirements of the patent. I explained that in light of the specification's description of the "present invention" as "conserv[ing] as much memory as possible," the presence of "one single defective bit [in an entire block] would not suffice" to deem the entire block "defective." Dkt. #131, at 16.

Because the accused products all find blocks defective after a single bit is found to have failed, none of them meet the "defective" memory location or "defect mapping" requirements found in the '660 and '713 patent. Defendants' motion for summary judgment will be granted as to the independent and dependent claims in these patents containing these requirements. This conclusion makes it unnecessary to address defendants' two additional argument about the claims arising under the '660 patent, that is, that the accused products do not infringe because they do not store both user data and logical address to physical address conversion in each block of the flash memory array. Even if defendants are wrong when they assert that their products do not infringe claim 1 of the '660 patent in these respects, plaintiff could not show that the products infringe the claims in all respects, which it must do to prove infringement.

B. Linking Physical and Logical Addresses (Cls. 1 and 16, '702 Pat.)

Claim 1 of the '702 patent requires "maintaining an updatable address data structure that links one or more physical addresses of [a] first group of pages with one or more of the logical addresses associated with the data stored therein." Claim 16 requires

> maintaining updatable address information that links physical addresses of the first and second blocks storing original and updated data with the logical addresses associated with the stored data, wherein the address information includes physical addresses of multiple blocks for individual logical addresses of original data having updated versions thereof.

In the order construing claim terms, I concluded that

> For "updatable data structure" ('702 pat., cl. 1) and "updatable address information" ('702 pat., cl. 16): (1) the terms "updatable data structure" and "updatable address information" are not synonymous; (2) the claimed "linking" must be more than simply storing a pointer in the physical address that points to the location of the logical address; (3) there is no requirement that the claimed "linking" be at the page level.

Dkt. #131 at 66. The parties' disagreement now is whether the claimed "linking" must link logical block addresses or could otherwise link other sorts of logical addresses, in particular logical page numbers and logical block numbers.

It is undisputed that the accused products do not link physical addresses with the logical address received from the host; indeed, there is no evidence that the products even store that address at all beyond their initial receipt. Instead, the record shows that accused products convert the logical block address received from the host into a logical block number, the logical block number is linked with a physical block number, and specific logical page numbers are linked with physical pages. At times the parties suggest that the question to answer is "do the claims at issue require the claimed updatable address information or updatable data structure to link a logical block address received from the host?" However, they also both identify the more important question, which is, "are logical block numbers or logical page numbers a form of 'logical addresses associated with the data stored therein'?"

Plaintiff's response to this question is simply a conclusory assertion that this matter is one for the jury. However, the scope of the term "associated with the data" is not a jury question, but one the court must construe. Plaintiff's failure to address this matter in its brief is reason alone to find for defendants on this matter. (Plaintiff also argues that no additional claim construction should be allowed on the matter, but the parties' unresolved disputes about the scope of claim terms cannot be ignored or sent to the jury. O2 Micro Intern. Ltd. v. Beyond Innovation Technology Co., Ltd., 521 F.3d 1351, 1361-62 (Fed. Cir. 2008).) At any rate, a proper construction of the term "associated with the data" excludes logical addresses pointing to physical locations in the flash memory.

Defendants contend that the term "associated with the data" requires that the logical addresses be LBAs, citing two statements in the specification suggesting as much:

> The subsystem controller in a large block system performs a number of functions including the translation between logicall addresses (LBAs) received by the memory sub-system from a host. . .

‘702 pat., col. 1, lns. 61-64. In addition:

> Data are written to the page location in the block corresponding to the low order bits of its logical address (LBA).

‘702 pat., col. 7, lns. 58-60. However, neither statement suggests that “logical address” as it is used in the patent must *always* be an LBA. Thus, I am not persuaded that the claim must be construed so narrowly.

On the other hand, the term cannot be construed so broadly as to include any “association” with data. Thus, the term “associated with the data” must require more than simply being listed in a table with physical addresses or providing a manner to find user data. If a logical address could be “associated with data” simply because it is linked to it in a table, the claim would not have had to include the term “associated with the data.” That term would be superfluous because the claim already requires linking the logical and physical addresses, which would allow the data to be found using the logical address.

Moreover, the term would be all but meaningless if any happenstance relationship between a particular logical address and user data could be considered an “association.” Thus, an address cannot be considered to be “associated” with data simply because it has some relationship with another address that is associated with the data.

To give any meaning to the term “associated with the data,” it should be read to exclude logical addresses that point to physical locations, as defendants suggest. This is because at most the relationship between the logical address and the user data is indirect. In the present case, for example, the logical page numbers and logical block numbers simply point to the physical location where that data will end up. Aside from that association, the only relationship the block number has with the user data is that the logical block address that is associated with that user data provides a path to the logical block number by way of a calculation by the controller. Because these logical addresses serve only to point to physical locations, I conclude that the accused products do not maintain updatable address data structures or information linking physical addresses of pages or blocks with the "logical addresses associated with the data stored therein."

Because I so conclude, it is unnecessary to address defendants argument is, that the accused products do not infringe claims 1 and 16 of the '702 patent because they do not program updated versions of data in each block of the flash memory array at a different offset position.

C. Direct Connection Between Controller and Flash Memory (Cls. 24 and 33, '702 Pat.)

Claims 24 and 33 of the '702 patent require that the product's memory controller be "connected with . . . the plurality of blocks of storage elements." In the order construing claims, I concluded that this meant that the "'memory controller' must be connected directly to the blocks, but it need not apply voltage to the blocks." Dkt. #131, at 66.

The accused products all include decoding circuitry between the controller and the flash memory blocks. Despite this, plaintiff contends that the connection between the controller and the memory blocks is "direct." First, plaintiff challenges the court's construction, pointing to the "preferred embodiment" of the '702 patent as including circuitry between the controller and the flash memory of a similar sort. Figure 1 shows this embodiment:

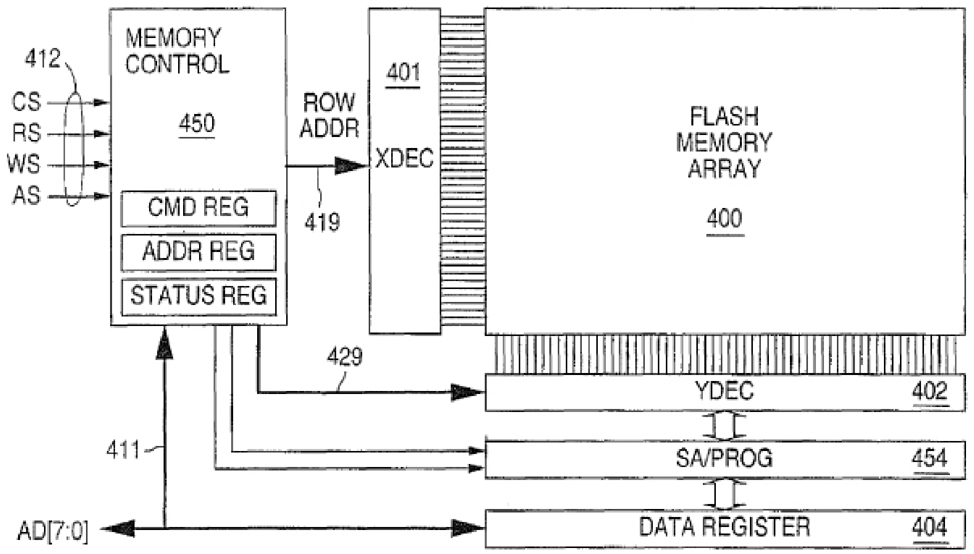


FIG. 1
(PRIOR ART)

Figure 2 further illustrates the "flash media interface" "in the form of a bus." Dkt. #220, at 64.

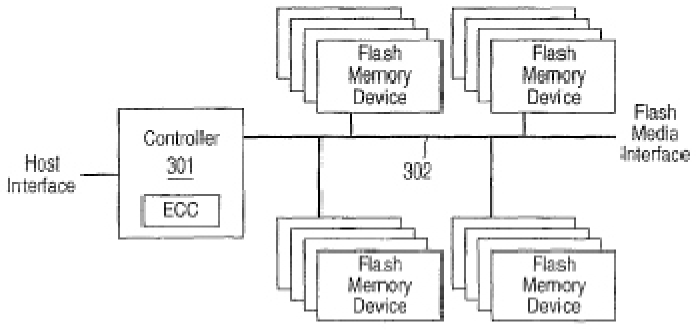


FIG. 2
(PRIOR ART)

As plaintiff acknowledges, the images are both labeled "PRIOR ART," but according to plaintiff, "[p]art of the brilliance behind the '702 patent is that its partial block update techniques and systems could be implemented with then-existing hardware." Dkt. #220, at 64. However, plaintiff offers no evidence to support that broad statement. Instead, plaintiff simply points out that the patent refers to Figure 2 when describing an embodiment, stating that for a configuration for partial block updating, "multiple chips may be connected to a single bus (as shown in FIG. 2)." '702 pat., col. 11, ln. 67, col. 12, ln. 3. The specification also mentions in the "Background of the Invention" section that the "controller also manages the low level flash circuit operation through a series of commands that it issues to the flash memory devices via an interface bus." Id., col. 11, ln. 67, col. 2, ln. 2.

Neither of the two passages plaintiff identifies is conclusive. Neither of the descriptions suggests that the patent is describing a "preferred" embodiment as opposed to just *an* embodiment of one or more claims of the invention, not necessarily relating to the claims at issue. Also, there appears to be no reason why a controller could not have both direct and indirect connections with the flash memory block.

More important, plaintiff does not address the reason I concluded that the patent required a direct connection when I construed the claim term the first time. I explained that the claim language specified that the controller must connect to a host-to-controller interface and a flash memory block, but did not mention any such connection to a flash interface. Because the possibility of using flash interfaces was known to be an option at the time of the invention and the patentees mentioned connecting to a host interface but not to a flash interface, I concluded that the claim language showed that the patentee intended to require a direct connection. Dkt. #131 at 36. Plaintiff's citations to the specification fail to support a broader construction.

Plaintiff also contends that the claim term should be construed to allow either direct or indirect connections between the controller and the flash memory because the claim uses the term "comprising" before claiming a "memory controller connected with . . . the plurality of blocks of storage elements." This argument does not travel far. Plaintiff cites no authority suggesting that a claimed element can be disregarded simply because it is listed as a requirement in an open-ended list.

Plaintiff's arguments fail to support a broader construction than that determined previously, which means the memory controllers in the accused products must directly connect to the flash memory blocks. Plaintiff makes one last effort to save this claim, arguing that there *is* evidence that such a direct connection exists. In particular, plaintiff's expert asserts that the connection through the interface is still "direct" because circuitry used to decode a physical block number does not "count" as interfering with a direct connection. This creative read of the term "direct" in the court's construction finds no support in the patent. Plaintiff's evidence that people of ordinary skill in the art knew that decoding circuitry *could* be placed between the controller and the flash memory does not show that the inventor meant not to "count" such circuitry. More to the point, plaintiff has no evidence that the inventor believed that decoding circuitry *must* be placed between the controller and flash memory. There is no evidence, for example, that people of ordinary skill thought the decoding circuitry could not have been placed elsewhere, such as within the controller, or that a system without decoding circuitry could not have been implemented. Without such a showing, plaintiff's argument amounts to a rehash of its argument that the controller should not be required to connect directly to the flash memory block. I have explained why that argument is wrong.

Because the accused products include decoding circuitry between the controller and the flash memory, plaintiff cannot show that defendants' products infringe this limitation literally. However, plaintiff also contends that product infringes under the doctrine of equivalents. The evidence fails to support this theory. It is undisputed that whether the decoding circuitry is located on the controller chip or flash memory chip is not critical to the operation of the memory. In either instance, the decoding circuitry performs the same function of decoding a given physical block number so as to supply data and operational voltages to a specific block. Indeed, it is even undisputed that either decoding addresses on a controller or decoding them on the chip will cause the data to be programmed into or read out from the appropriate blocks of the Flash memory. However, it is also undisputed that there is an important difference between a controller that is directly connected to a flash memory device and one that is connected through an interface. A direct connection between a controller and memory blocks requires the controller to be programmed to work with those specific blocks. If a controller uses a standard interface instead, it can work with multiple types of flash memory blocks. This is fatal to plaintiff's theory of equivalence, under either the insubstantial differences test or the function-way-result test. Cf. Voda v. Cordis Corp., 536 F.3d 1311, 1326 (Fed. Cir. 2008) (describing requirements for each test). The different nature of a direct connection means it is not an "insubstantial difference" just as it means it does not perform "in substantially the same way."

Because plaintiff has failed to show that the accused products contain a "memory controller connected with . . . the plurality of blocks of storage elements" as required in claims 24 and 33 of the '702 patent, defendants' products do not infringe these claims or the claims that depend from them and defendants are entitled to summary judgment as to these claims.

D. Bad Block Data Stored in First Group of Blocks (Cl. 14, '511 Pat. and Cl. 1, '667 Pat.)

Claim 14 of the '511 patent claims a method of operating a memory system and claim 1 of the '667 patent claims a memory system, each involving a "first group of blocks" generally for storing "user data" and a "second group of blocks" generally for storing information about the "physical characteristics of the first group of blocks." Despite the fact that claim 14 of the '511 patent is a method claim and claim 1 of the '667 patent is an apparatus claim and each uses slightly different language, the parties agree that the arguments defendants raise apply equally to both claims.

In the order construing claims, I concluded that not any type of physical characteristics data could be stored in the first group of blocks but some such types could be. Dkt. #131 at 66-67. I rejected both parties' proposed constructions. Defendants had sought a requirement that no physical characteristics data could be stored in the first group of blocks, but the specification provided that a "pre-erase bit" could be stored in the first group of blocks: a pre-erase bit is a sort of physical characteristics data. At the other extreme, plaintiff pursued a construction that would allow any physical characteristics data in the first group of blocks. As I explained, this could not be right, either, because it would "conflict with the stated advantages of out-of-block storage of this data generally and with the specifications' repeated statements emphasizing that physical data should generally be stored elsewhere." Id. at 54.

Defendants contend that "bad block" data such as that found in the accused products is one sort of physical characteristics data that cannot be included in the first group of blocks. This theory finds support in the specification, which distinguishes the "present invention" from prior art as follows:

> Prior memory systems, particularly those which emulate a disk drive by storing 512 byte sectorsof user data, have also stored, in the individual user data blocks, information about the block's characteristics in terms of experience cycles, numbers of pulses or voltages required to program or erase the block of cells, defects within the block, and like information about the storage media, in addition to an ECC generated from the data stored in the block. As part of the present invention, however, this type of information about the physical block is stored in another block.

E.g., '511 pat., col. 14, lns. 6-16. Thus, according to the specification, the "present invention," unlike the prior art, will not include in individual user data blocks the "type of information" that includes information about "defects within the block." Id. As defendants point out, the specification's description of the "present invention" is evidence that "strongly suggests" that the claims should be limited as described. Trading Technologies International, Inc. v. Espeed, Inc., 595 F.3d 1340, 1356-57 (Fed. Cir. 2010).

Against this, plaintiff has three arguments. First, plaintiff contends that defendants fail to identify the correct step in the claim language because they point to the step that involves "storing, in individual ones of the second group of said blocks." This hypertechnical argument must be rejected. Plaintiff was on notice that defendants believed that the "first group of blocks" contained information beyond the scope of the claim language because that language was construed in the order construing claims.

Second, plaintiff contends that the court has already rejected defendants' theory because it presented the "present invention" argument at the claims construction stage. However, the question then was whether *all* physical characteristics information should be excluded from the first group of blocks. Now, the question is simply whether bad block information should be. For that question, the "present invention argument" is more persuasive.

Third, plaintiff contend that, in the accused products, the use of a bad block marker "achieve[s] at least one of the stated advantages set forth by the Court." Dkt. #220, at 76. In particular, according to plaintiff, use of the bad block marker "avoids having to rewrite "the overhead data each time the user data is rewritten into the block," at least when the bad block marker is used as the accused products use it. This is because the accused products stop writing to and reading from a block once a bad block marker is written to the block. It is unclear whether this argument is available to plaintiff. Its expert, V. Thomas Rhyne, has stated in a different context that it is his opinion that storing defect data in user data blocks "would not achieve any of the stated advantages of the '667 inventions" and that because "defect data is not bound to change each time the corresponding user data changes," "storing defect data in the same block as user data would not achieve the stated advantage of avoiding rewriting physical characteristics data unnecessarily." Dkt. #165-13, ¶ 294. He further stated that it was his "opinion that defect data cannot be stored in the same blocks as user data." *Id.*, ¶ 295. Plaintiff emphasizes that Rhyne made these statements in the context of challenging an invalidity claim in light of a patent that included "bad sector" information in every user data sector; nonetheless, the expert's statements were not so qualified. As defendants point out, plaintiff's expert cannot take one position for the sake of invalidity and an opposing one for the sake of infringement. Dkt. #277 at 38-39 (citing W.L. Gore & Associates, Inc. v. Garlock, Inc., 842 F.2d 1275, 1279 (Fed. Cir. 1988)).

However, even if plaintiff were correct and I could find that the use of bad blocks achieves one of the purposes of the patents, this does not undermine the specifications' more specific statement that the "present invention" will not store defect data in user data blocks. Above and beyond the general purposes of the patent, the patentee made it clear that it believed that defect data should be excluded from the first block.

Plaintiff also argues that because bad blocks are never marked defective at the same time any of the user data is in use, bad block data does not fall within the meaning of "physical characteristics." However, it is not clear why bad block data would not be "physical characteristic" data about a user data block simply because the user data block will no longer store data. The bad block data still provides information about the block itself, in particular that there is a defect within the block. This is still physical characteristics data, and as explained above, it does not belong in user data blocks.

Plaintiff appears to argue that the block receiving a bad block marker is no longer a "user data" block because no additional user data will be stored in that block. However, plaintiff fails to identify any basis in the patent for treating a block that has stored user data as no longer "storing" that data simply because the data from the block will not be read from or written to after the bad block marker is placed in it.

Finally, plaintiff contends that the bad block markers used in the accused products are not "defects within the block" and therefore not "physical characteristics" information because although the bad block marker indicates that the block contains a defect and should not be used, the marker does not "indicate where any defects exist within the block or what type of defect is present." Dkt. #220 at 80. Plaintiff does not offer any explanation why this additional information about the defect must be provided before the information providing that a block contains a defect could be considered "physical characteristics" information.

None of plaintiff's arguments succeed. Because the products all store bad block markers in blocks of data containing user data, they do not meet the requirements of claim 14 of the ‘511 and claim 1 of the ‘667 patent or their dependent claims. Therefore, defendants are entitled to summary judgment on these claims as well. Again, this conclusion makes it unnecessary to address defendants’ other argument for finding that plaintiff has not shown that their products infringe claim 14 of the ’511 patent, which is that they do not write user data “until at least a different sector of the user data . . . are written into each of the plurality of blocks.” For the same reason, it is unnecessary to address defendants’ argument that claim 1 of the ‘667 patent is not infringed because plaintiff has no evidence that the accused products incorporate a controller memory that is faster than the non-volatile memory in the products.

E. Record of Logical Block Addresses (Cl. 1, ‘666 Pat.)

Claim 1 of the ‘666 patent requires the claimed memory system contain “a record stored in the memory system that contains nonoverlapping ranges of logical addresses of the designated blocks of memory cells within each of the at least two groups.” In the order construing claim terms, I concluded that “(1) ‘a record’ is limited to one record; (2) the record need not be sufficient to determine a physical address; and (3) the recited ‘logical address’ need not be an address received from the host.” Dkt. #131 at 66.

Plaintiff raises several criticisms of defendants’ proposed construction and application of the term at issue, but it is not necessary to address these criticisms. In response to defendants' contention that plaintiff could not identify any "record" of the accused products, plaintiff points to the following data structures for each accused controller: in Phison controllers, it is the set of constants used for zone determination, X_ZonePerDevice, X_DeviceLBlock and X_Flash_number; in Skymedi controllers, it is the Windows table; in Silicon Motion controllers, it is the LOADDATBLK. As defendants point out, none of these data structures "contain" any address at all, let alone a "nonoverlapping range" of such addresses. Instead, each data structure identified is a constant or set of constant numbers such as the number of zones in each device or the number of blocks in each zone or window. Plaintiff points out that for each of these "records," the ending logical address for each zone could be calculated from that "record." However, the ability to calculate an address is not the same as "containing" an address. Because plaintiff has failed to identify any aspect of the accused products that could be said to actually "contain" a nonoverlapping range of logical addresses, it cannot establish literal infringement of the '666 patent.

Plaintiff asserts a doctrine equivalents argument as well. The whole of plaintiff's theory of equivalence is drawn from the following paragraph of its expert, V. Thomas Rhyne:

> It is my opinion that records containing data sufficient to calculate the ending LBAs of ranges of logical addresses are insubstantially different from the claimed "record" limitation of claim 1 of the '666 patent. This is illustrated by the exemplary record of Figure 14 of the '666 patent. The record of Figure 14 stores the ending LBA of each range of LBAs in the memory system. It is my

> opinion that data sufficient to calculate the ending LBAs of ranges of logical addresses is insubstantially different from data for the actual ending LBAs because such data is ultimately used to calculate the bounds of ranges. Thus, records containing data sufficient to calculate ending LBAs meet the "record" limitation under the doctrine of equivalents.

Dkt. #224, ¶ 23. (In Rhyne's original report, he simply stated that any difference between the accused products and "any claim limitation . . . may be insubstantial" so he "reserve[d] the right to supplement [his] opinions" with respect to equivalence. Dkt. #284-12, ¶ 418.)

Thus, the whole of plaintiff's theory of equivalence comes down to Rhyne's assertion that the use of the constants in the accused products "is ultimately used to calculate the bounds of ranges," which he says makes it insubstantially different from the embodiment in figure 14, which stores the ending logical block addresses. However, the record shows that the constants are used not to "calculate the bounds" of the range of addresses but rather to convert a given logical block address into a logical block number. The constant and a calculation together *could* identify the "bounds," but the nature of the "record" simply does not require those bounds to be identified at all. Rhyne's cursory discussion is not persuasive because it does not address the apparent differences between storing end ranges of logical block addresses and storing a constant and using a calculation that bypasses any need to compare a given logical block address with end ranges. No reasonable jury could find equivalence in light of the sparse evidence in support of it. Because plaintiff's evidence falls short of supporting its claim of infringement of the '666 patent either literally or by equivalence, defendants are entitled to summary judgment on that ground as well.

Because I have concluded that none of the accused products are infringing any of the asserted claims, defendants' additional arguments for noninfringement are moot, as are their arguments for reduced damages. Less clear is whether their argument for invalidity is moot, but it is not necessary to determine as much. The Court of Appeals for the Federal Circuit has held that a district court has the discretion to dismiss invalidity counterclaims upon a grant of summary judgment of non-infringement. Phonometrics, Inc. v. Northern Telecom Inc., 133 F.3d 1459, 1468 (Fed. Cir. 1998); Cardinal Chemical Co. v. Morton Int'l, Inc., 508 U.S. 83, 95 (1993) (in addressing motion for declaratory judgment district court has discretion to decide whether to exercise jurisdiction even when established). It is appropriate for a district court to address only the infringement issue when non-infringement is clear and invalidity is not plainly evident. Phonometrics, Inc., 133 F.3d at 1468 (citing Leesona Corp. v. United States, 530 F.2d 896, 906 n.9 (Ct. Cl. 1976)).

Rarely is an invalidity question "plainly evident" and there is no reason to think this case is any exception. The only invalidity challenges defendants have raised at this stage involve parsing the language of the claims to determine whether certain language should be construed to require method steps within apparatus claims. Defendants have not suggested that any of their remaining invalidity claims are particularly straightforward. Under these circumstances, I conclude that it is appropriate to decline to exercise jurisdiction over defendants' remaining invalidity counterclaims.

ORDER

IT IS ORDERED that:

1. The parties' joint stipulation to reduce claims and simplify issues, dkt. #215, is GRANTED.

2. The motion to strike filed by plaintiff SanDisk Corporation, dkt. #216, is DENIED.

3. The motion for leave to file corrected versions of expert reports filed by defendants Kingston Technology Co., Inc. and Kingston Technology Corp., dkt. #267, is GRANTED.

4. The motions for leave to file additional briefing in the form of a reply in support of the motion to strike, dkt. #286, and a sur-reply in opposition to the motion to strike, dkt. #295, are GRANTED.

5. Defendants' motion for summary judgment, dkt. #154, is GRANTED with respect to plaintiff's claims of infringement and DENIED in all other respects.

Entered this 12th day of August, 2011.

BY THE COURT:

/s/

BARBARA B. CRABB
District Judge