

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION,
Petitioner

v.

BiTMICRO LLC,
Patent Owner.

IPR2023-00780 and IPR2023-00779

U.S. Patent No. 9,135,190

**DECLARATION OF DONALD ALPERT
IN SUPPORT OF INTER PARTES REVIEW OF U.S. 9,135,190**

Mail Stop PATENT BOARD
Patent Trial and Appeal Board
U.S. Patent & Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

INTEL-1003
9,135,190

TABLE OF CONTENTS

I. Qualifications	2
II. Understanding of Relevant Legal Principles	5
III. '190 Patent: Technical Background and Overview.....	7
A. Technical Background.....	7
1. Memory Types	9
a) Dynamic Random Access Memory (DRAM).....	10
b) Static Random Access Memory (SRAM).....	11
c) Non-Volatile Memory/Flash	11
2. Memory Controllers	23
3. Memory Access Patterns.....	25
B. The '190 Patent.....	27
C. Claim Construction.....	29
1. “Means for Performing a Memory Transaction”.....	31
2. “Means for Addressing”	32
3. “Interface Controller Means”	33
D. Level of Ordinary Skill in the Art	33
IV. Combination of Danilak and Cornwell.....	34
A. Overview of the Combination	34
1. Danilak	34
2. Cornwell	45
3. Combination of Danilak and Cornwell	49
a) Cornwell’s Non-Volatile Memory Structure	49
b) NAND and NOR Flash Devices	53
B. Independent Claims	57
1. Independent Claim 1 (Method)	57
a) Structural Limitations.....	60
(1) “Memory Device”	62
(2) “Memory Location”	63
(3) “Memory Store”	66

(4) “ <i>Controller</i> ”	68
(5) “ <i>Memory Bus</i> ”	75
(6) “ <i>Communication Path</i> ”	78
b) “ <i>Device Profile</i> ” Limitations.....	79
(1) “ <i>Device Profile</i> ”	80
(2) “ <i>Attributes</i> ”	84
(3) “ <i>Attribute Qualifier</i> ”	87
(4) “ <i>Device Profile is Optimal for a Data Type Subject to the Memory Transaction</i> ” [1A.6].	89
c) Preamble [1P]	96
d) Method Steps	100
(1) “ <i>Addressing a First Memory Location ...</i> ”	101
(2) “ <i>Identifying Command Details ...</i> ” [1B].....	103
(3) “ <i>Obtaining the First Set of Attributes</i> ” [1C] and “ <i>Using Attributes From Said First and Second Device Profiles</i> ” [1D].	105
(4) “ <i>Selecting a Transfer Size for the Memory Transaction</i> ” [1E].	108
e) “ <i>Read-Write</i> ” Limitations	111
(1) Controller Performs the Memory Read Transaction or Memory Write Transaction on a Memory Location.	112
(2) Write Data Flow Path.....	115
(3) Read Data Flow Path	117
f) Summary	119
2. Independent Claim 29 (Computer Program Product Claim)	124
a) Preamble [29P]	126
b) “ <i>Device Profile</i> ” Limitations.....	127
c) Action Steps	128
d) Summary	130
3. Independent Claim 59 (Memory Controller)	133
a) Preamble [59P]	135
b) “ <i>Interface Controller</i> ” [59A].	136
c) “ <i>Memory Store</i> ” [59B]	139

d) “Device Profile” Limitations.....	141
e) Action Limitations.....	142
f) Summary	144
4. Independent Claim 45 (Memory Controller)	148
a) Preamble [45P] and “ <i>Means for Performing</i> ” [45A].....	149
b) “ <i>Means for Addressing</i> ” Limitations [45A.1]/[45A.3a]/[45A.3c]/[45D]/[45E].....	151
c) “ <i>Interface Controller Means</i> ”	154
d) “ <i>Identifying Command Details</i> ” [45B]	155
e) Device Profile Limitations	155
f) Summary	156
5. Independent Claim 80 (Electronic Storage Device)	164
a) Preamble [80P]	165
b) “ <i>Memory Controller</i> ”	166
c) “Device Profile” Limitations.....	167
d) Action Limitations.....	168
e) Summary	170
6. Independent Claim 98	175
7. Independent Claim 100 (Method)	177
a) Preamble.....	178
b) Structural Limitations.....	179
c) “Device Profile” Group.....	180
d) Method Steps and Transaction Limitations	181
e) Summary	183
C. Dependent Claims.....	189
1. Claims 2, 33, 46, and 82.....	189
2. Claims 5, 36, 51, 71, and 84.....	190
3. Claims 6, 37, 52, 76, and 85.....	194
4. Claims 7, 38, 53, 77, and 87.....	196
5. Claims 8, 39, 54, 72, and 86.....	197
6. Claims 11, 42, 57, and 92.....	198

7. Claim 19	199
8. Claim 30	200
9. Claims 94-97	201
V. Conclusion.....	203

I, Donald Alpert, declare as follow:

1. I have been engaged by Perkins Coie LLP on behalf of Intel Corporation (“Intel” or “Petitioner”), to provide this Declaration concerning technical subject matter relevant to the petition for *Inter Partes* Review (“Petition”) of U.S. Patent 9,135,190 to Bruce, et al (“the ’190 patent”).

2. I am over 18 years of age. I have personal knowledge of the facts stated in this Declaration and could testify competently to them if asked to do so.

3. I have reviewed and am familiar with the specification and the claims of the ’190 patent. In general, I will cite to the specification of a United States patent using the following formats: (Patent No., Col:Line Number(s)) or (Patent No., Paragraph Number(s)). For example, the citation (’190 patent, 1:1-10) points to the ’190 patent specification at column 1, lines 1-10. Also, for convenience, I use italics to denote limitations from the challenged claims.

4. All of the opinions contained in this Declaration are based on the documents I reviewed and my knowledge and professional judgment. In forming the opinions expressed in this Declaration, I reviewed the documents in the Exhibit List in the attached appendix to this document. I have also reviewed and am familiar with the other materials cited in this declaration.

5. I have been asked to provide my technical opinions regarding how a person of ordinary skill in the art would have understood the claims of the ’190

patent at the time of the alleged invention, which I have been asked to assume is the 2009 timeframe. For purposes of whether the teachings of the prior art render the claims of the '190 patent obvious, I have been asked to assume the date of September 4, 2009. I have also been asked to provide my technical opinions on how concepts in the '190 patent specification relate to claim limitations of the '190 patent. In reaching the opinions provided herein, I have considered the '190 patent, its prosecution history, and the references cited above and have drawn as appropriate on my own education, training, research, knowledge, and personal and professional experience.

I. Qualifications

6. In formulating my opinions, I have relied on my knowledge, training, and experience in the relevant field, which I will briefly summarize. A more detailed summary of my background, education, experience, and publications is set forth in my curriculum vitae (CV), which is submitted as INTEL-1004.

7. I have personal knowledge of the facts and opinions set forth in this declaration and believe them to be true. If called upon to do so, I would testify competently thereto. I have been warned that willful false statements and the like are punishable by fine or imprisonment, or both.

8. I am being compensated for my time at my standard consulting rate. I am also being reimbursed for expenses that I incur during the course of this work.

My compensation is not contingent upon the results of my study and analysis, the substance of my opinions, or the outcome of any proceeding involving the Challenged Claims. I have no financial interest in the outcome of this matter or in any litigation involving the '190 patent.

9. I have 45 years of academic and industrial experience in applying, designing, studying, teaching, and writing about microprocessors and computer systems. I received an Electrical Engineering Ph.D. degree in 1984 from Stanford University. I earlier received an Electrical Engineering B.S. degree from MIT in 1973 and an Electrical Engineering M.S. degree from Stanford University in 1978. I have taught classes in computer architecture at Stanford, Tel Aviv, and Arizona State Universities.

10. From 1976 to 1977, I worked at Burroughs Corporation, where I designed peripheral interface controllers, including those for serial data communications based on Intel 8080 microprocessor components. From 1980 to 1989, I was the lead architect for the design of three high-performance microprocessors at Zilog and National Semiconductor. Later, at Intel, I was the lead architect of the Pentium® Processor from 1989 to 1992 and of the 815 chipset from 1999 to 2000, both of which became the most widely used PC components of their time. The 815 chipset comprised two components: (1) a memory controller hub (MCH) that included a graphics controller and memory controller with

interfaces to the CPU, 133 MHz SDRAM system memory modules, an optional, external graphics controller and (2) an I/O controller hub (ICH) that included various I/O controllers (e.g., network, hard drive, USB) for system peripheral devices and power management control registers. Additionally, I served as co-manager for the Itanium processor design from 1993-1997.

11. I am a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), and served as the chair of the IEEE Technical Committee on Microprocessors and Microcomputers from 1999 to 2000. I was the keynote speaker at the first Cool Chips conference, dedicated to the study of low-power microprocessors and systems. I have given invited lectures at several universities, and published ten papers in various professional journals and conference proceedings. My paper entitled “Architecture of the Pentium Processor,” was selected as best paper in IEEE Micro for 1993. I am a named inventor on over 30 U.S. patents that pertain to microprocessors, computer systems, and related technology.

12. I have reviewed the '190 Patent, and I am familiar with the patent's subject matter, which is within the scope of my education and professional experience. Based at least on my background in academia, industry, and consulting, I am familiar with the issues and technology relating to processors, chipsets, memory, and peripheral devices and their interoperation in computer

systems, I have personally analyzed, developed, and tested such computer components and systems. For example, the 815 chipset for which I was the lead architect at Intel implemented an cache memories, SDRAM controller, IDE interface to hard disks and solid-state disks (SSD), and firmware hub for BIOS ROM.

II. Understanding of Relevant Legal Principles

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

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

15. I have been informed that a patent claim is invalid if it would have been obvious to a person of ordinary skill in the art. In analyzing the obviousness of a claim, I understand the following factors may be taken into account: (1) the scope and content of the prior art; (2) the differences between the prior art and the claims;

(3) the level of ordinary skill in the art; and (4) any so called “secondary considerations” of non-obviousness, if they are present. I am not aware of any evidence of secondary considerations of non-obviousness relevant to the ‘190 patent. I reserve the right to supplement this Declaration if Patent Owner introduces evidence of secondary considerations of non-obviousness.

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

- (1) identify the particular references that, singly or in combination, make the patent obvious;
- (2) specifically identify which elements of the patent claim appear in each of the asserted references; and
- (3) explain why a person of ordinary skill in the art would have combined the references, and how they would have done so, to create the inventions claimed in the patent. I further understand that exemplary rationales that may support a conclusion of obviousness include:
 - combining prior art elements according to known methods to yield predictable results;
 - simple substitution of one known element for another to obtain predictable results;
 - use of known technique(s) to improve similar devices (methods or products) in the same way;
 - applying a known technique to a known device (method or product)

ready for improvement to yield predictable results;

- “obvious to try” – choosing from a finite number of identified, predictable solutions with a reasonable expectation of success;
- known work in one field of endeavor may prompt variations of the work for use in either the same field or a different field based on design incentives or other market forces if the variations are predictable to a person of ordinary skill in the art; and
- some teaching, suggestion, or motivation in the prior art that would have led a person of ordinary skill in the art to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

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

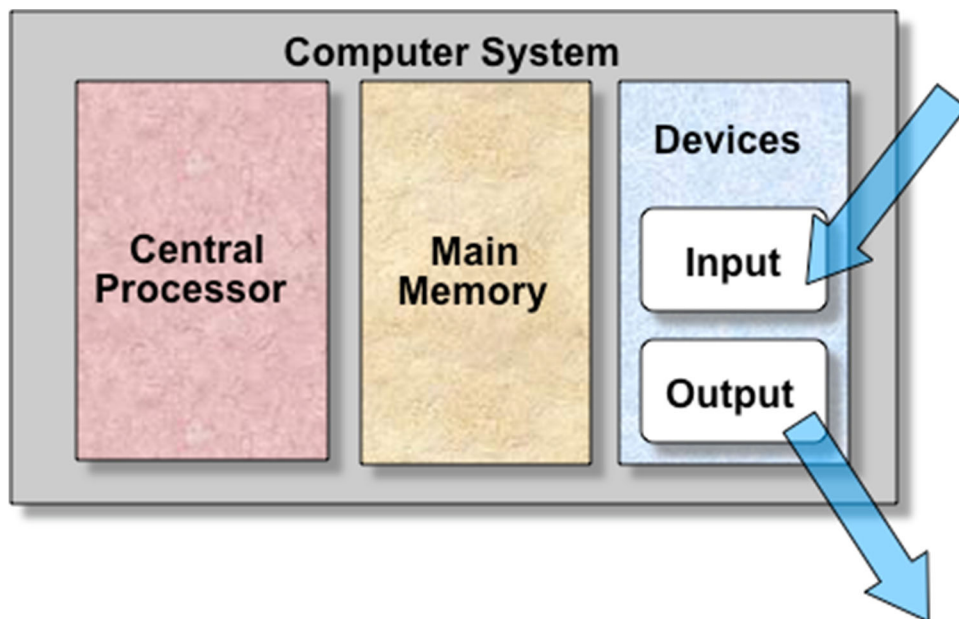
III. '190 Patent: Technical Background and Overview

18. The '190 patent is directed to memory having “multi-profile memory controllers” and to “computing devices, such as solid-state storage devices, that use these multi-profile memory controllers.” (INTEL-1001, 1:17-19.) In this section, I first provide a brief overview of computer memory and an overview of the '190 patent. I then discuss the level of skill in the art and terms requiring construction.

A. Technical Background

19. A system's hardware generally comprises three types of components, as described below and shown in my demonstrative figure below:

- **Central Processor:** The system's central processor (CPU) fetches and executes the instructions of application programs and operating system software.
- **Main Memory:** The main memory stores instructions executed by the central processor and data accessed by the central processor and I/O devices.
- **Input/Output (I/O) Devices:** I/O devices (also known as "peripheral devices") provide auxiliary storage (e.g., hard drive), communication between systems (e.g., network interface), and human interface (e.g., display).



Communication with the memory takes place on a bus which generally contains a set of DATA lines, a set of ADDRESS lines, and a set of CONTROL lines that specify what action is going on. I focus my discussion in this background section on memory.

1. Memory Types

20. Various types of memory are used in a computer system according to cost, speed, volatility, and other requirements. Volatility refers to the characteristic of a memory device's losing or retaining stored information when power is removed. A volatile device loses its information when power is removed. A non-volatile device, in contrast, retains its information.

21. An exemplary hierarchy for personal computer memory is illustrated in Figure 1.7 from Brewer (reproduced below). The memory closest to the CPU, the cache, provides very fast access. (INTEL-1008, 8.) The information stored in the cache is an “image of a portion of the data stored in the next level of memory.” (*Id.*) The main, or primary, memory stores “the programs or data to be processed at the present time.” (*Id.*) A computer system's main memory generally comprises DRAM (Dynamic Random Access Memory), a type of volatile memory, which I discuss briefly in the next section. The main memory “retrieves program and code from the secondary store and writes information to be retained in the form of files back to the secondary store.” (*Id.*) “The secondary memory is nonvolatile and block oriented.” (*Id.*) Although this figure does not call out nonvolatile memory, nonvolatile memory was recognized to be a lower cost option for secondary storage. (*See Id.*) For example, secondary storage for personal computers is often “small manually transportable storage that can be accessed via a universal serial

bus (USB) port.” (*Id.*) For physically small devices such as phones and digital cameras, nonvolatile secondary storage dominates. (See INTEL-1008, 8-9.) I discuss nonvolatile memory further below. Although the figure below only shows a cache associated with the processor’s accesses to primary memory, caches can be placed at other locations in the system, such as within secondary memory, to improve average access speed to slower devices.

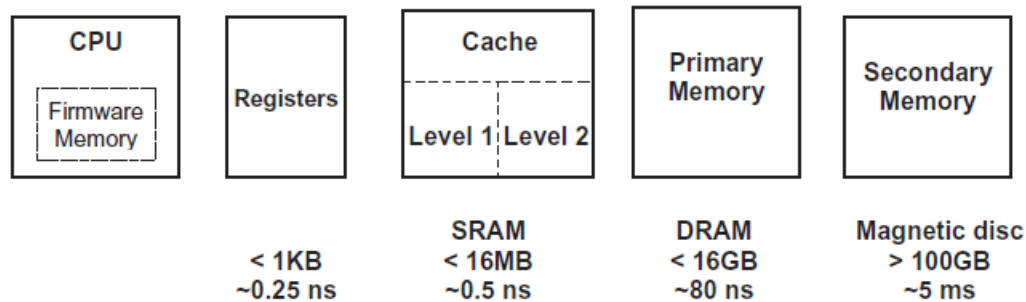


Figure 1.7. Memory hierarchy showing approximate latency and byte capacity bounds.

a) Dynamic Random Access Memory (DRAM)

22. DRAM is a type of volatile memory that can be read and written.

DRAM stores data by the presence or absence of an electrical charge in association with a single transistor that transfers the charge. The charge dissipates over time, so the stored data must be periodically refreshed by restoring the charge; this is the dynamic characteristic of DRAM. Random access refers to the characteristic that all data stored in the DRAM can be accessed in approximately the same time.

b) Static Random Access Memory (SRAM)

23. SRAM is another type of volatile memory that can be read and written. In comparison with DRAM, SRAM does not require refreshing to maintain stored data. SRAM is generally designed to operate faster than DRAM, and it requires multiple transistors per bit, which lead to higher cost per bit. A *cache* memory comprising SRAM is commonly placed between the processor and main memory, where the cache holds copies of recently accessed memory locations. (INTEL-1015, 93.)

c) Non-Volatile Memory/Flash

24. Nonvolatile memory in which data is retained in the absence of dc power, is essential for applications such as (a) start-up code and computer settings, (b) resident firmware for any device using an embedded processor, or (c) storage (e.g., USB drive or Solid-State Device (SSD)) of documents, images, or other files. One type of widely used nonvolatile memory is called Flash memory.

25. Flash memory “makes use of a charge stored on a floating gate to accomplish nonvolatile storage.” (INTEL-1008, 3.) In a Flash memory cell, a floating gate (shown in Figure 1.1 below) allows charge to be stored indefinitely, thereby affecting the threshold voltage for a control gate that is used to sense the stored value. (*Id.*) Because the floating gate is surrounded by insulators, it can

“retain charge for a long period of time independent of whether the circuit power supply voltage is present.” (*Id.*)

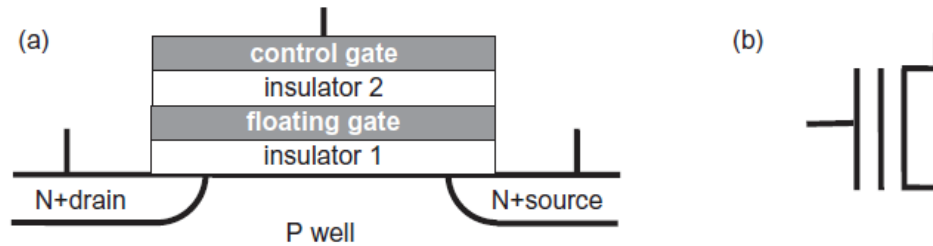


Figure 1.1. Floating-gate transistor: (a) elements of the transistor structure and (b) circuit symbol.

26. Flash memory cells can be categorized as single-level cell (SLC) or multi-level cell (MLC). A single-level cell distinguishes between a ‘1’ and ‘0’ by having no charge or charge present on the floating gate of the cell. Increasing the number of charge levels allows more than 1-bit per cell to be stored. For example, a two bits per cell (MLC) storage has four charge levels representing 00, 10, 01, and 11. MLC storage permits an increase in capacity without incurring an increase in process complexity.

27. There are many variants of Flash memory, but production is dominated by two types: NAND Flash and NOR Flash. (INTEL-1008, 2.) As the name implies, NOR flash is based on NOR gates; it is organized as a stack of parallel cells connected to a common ground (transistor source) and a bit line (transistor drain); one of the cells is selected to read by activating a word line connected to the control gate.

Comparison of NAND and NOR Flash Cells

	NAND	NOR
Cell Array		
Layout		
Cross Section		
Cell Size	$4F^2$	$10F^2$

(INTEL-1010, 3.)

28. NAND flash is based on NAND gates and organized as a series of cells connected between a bit line and ground; one of the cells is selected to read by activating a word line connected to its control gate, while the control gates of other cells are pulled high enough to ensure conduction. (NAND Flash is generally denser (more bits per area) than NOR because the cells in series do not require direct connection to a bit line or ground, but NAND flash is also generally slower

to access than NOR because the series cell connection provides lower conductance between the bit line and ground. (INTEL-1010, at 4.)

29. A number of basic principles of operation apply to both NOR and NAND Flash device. Brewer's Figure 3.2 illustrates basic features of a Flash memory. Cells are packed closely together in a planar array. (INTEL-1008, 5.) The X,Y array arrangement "contains the cells and conductive lines that allow access to each individual cell." (*Id.*) Lines running in the X direction (rows) are called wordlines and are "used to select a row of cells during the write or read operations." (*Id.*) Lines running in the Y direction (column) are called bitlines. (*Id.*) The "wordlines activate a specific group of cells in a row, and the bitlines for each intersection column connect those cells to read and write circuits." (*Id.*)

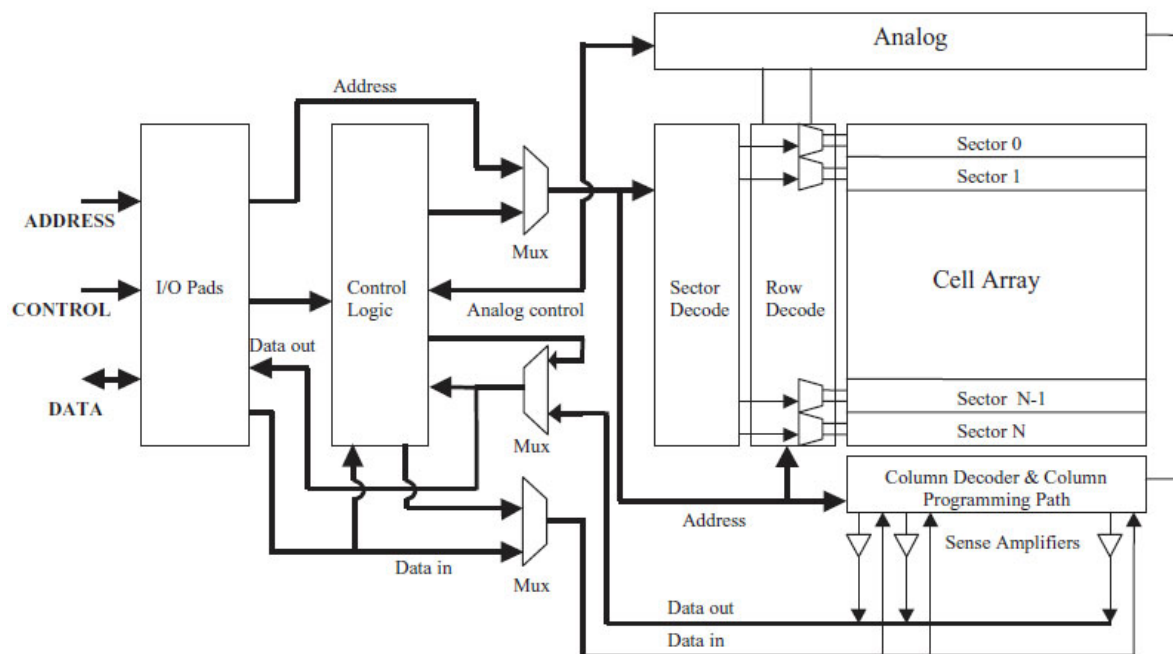


Figure 3.2. Flash memory circuit block diagram.

30. “While the heart of a semiconductor memory is the cell, the surrounding circuitry is the mechanism that makes it useable.” (INTEL-1008, 5.) The circuitry around the periphery of the array performs the basic operations on Flash cells: program, erase, and read. The array is often divided into relatively small blocks. (INTEL-1008, 67.) A block is the smallest erasable unit of the memory. (*See* INTEL-1010, 3.) Flash further groups a block into “pages” which are the smallest addressable unit for reading and writing. The memory input “consists of data, address, and control signals” and memory output “consists of data and status signals.” (INTEL-1008, 7.) The address indicates to the memory device where to perform the requested operation. To select a given row and column, the memory device is provided with a binary address word from external circuitry. (Brewer, 6.) This address word is routed to address decoder circuitry, designed to drive the word and bitlines. (*Id.*) A memory device also incorporates control logic “that accepts relatively simple commands as inputs and generates timed sequences of signals to accomplish writing, reading, and various other modes of operation.” (*Id.*)

31. A Flash device powers up in the read state. Therefore, algorithms are needed to perform functions such as programming a byte or series of bytes and erasing a block or series of blocks. (*See* INTEL-1008, 33.) For example, to write a telephone number and address to a Flash device, a program routine, such as

illustrated below, “must first be called that sends the program command to the Flash device and then writes the information to the desired location in the Flash.” (*Id.*) If not enough free space to write the information, an erase command must be called first to erase a block of Flash. (*Id.*) A status routine is called between each program command “to ensure that the data has been properly written to the device.” (*Id.*) After the data is written, “the read command must be sent to the device in order to read information from the device again.” (*Id.*)

```
Main()
{
  if (!(space_available)) {
    Erase_block (block_to_erase);
  }
  Program (beginning_address, data, number_bytes_to_program);
  Read();
}

Program (beginning_address, data, number_bytes_to_program)
{
  Flash_address = beginning address;
  while (number_of_bytes_to_program)
  {
    Send_command (Program_command, Flash_address, data_byte);
    while (Flash_status == not_done) {
      wait;
    }
    number_of_bytes_to_program--;
    Flash_address++;
  }
}
```

Figure 2.4. Example basic algorithm calling sequence.

32. Flash memory has enabled the creation of small consumer electronics devices such as smartphones, handheld computers, and digital music players “by providing storage for the program code that defines their functionality as well as

for the increasing amount of user data that must be carried.” (INTEL-1008, 19.)

Flash has become “the standard system for embedded designs because it provides benefits for both engineering, in quick code updates during development, and for production and support.” (INTEL-1008, 20.) The selection of either NAND or

NOR or both for a specific application is driven by the strengths and weaknesses of each device. The following Figure from Campardo summarizes the main applications of flash memory which I discuss further below. (See INTEL-1012, XXV.)

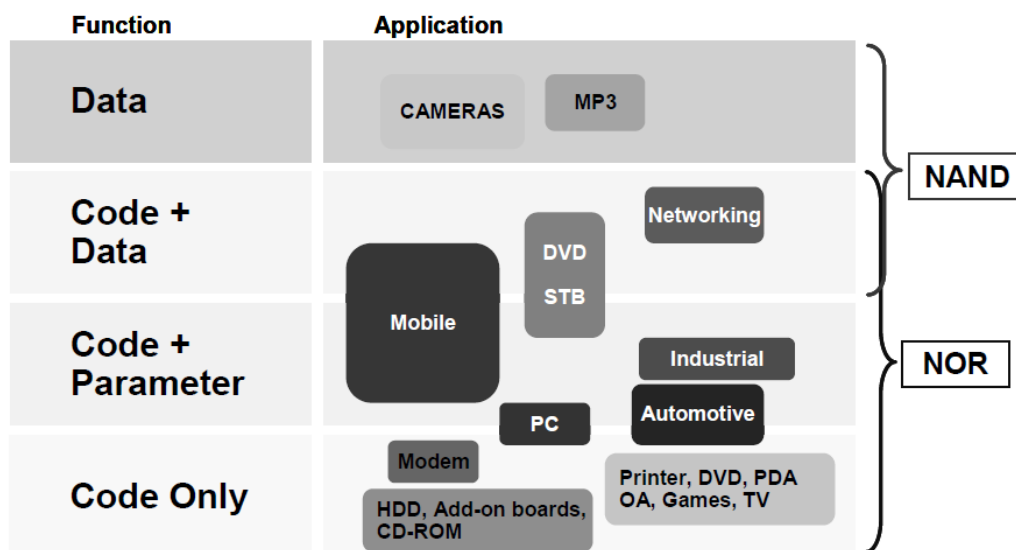


Fig. 6. Main applications of flash memories

33. “The strengths of NOR-type Flash memories are fast random-access reads and reliable storage.” (INTEL-1008, 22.) As I discussed above, NOR Flash memory employs a parallel array architecture where each cell may be accessed via a contact. This direct cell access is a contributing factor to the superior random

performance of NOR Flash. NOR flash therefore works well in any application where fast, reliable random-access reads are needed, like for code or combined code+data storage. (INTEL-1008, 22; *see also*, INTEL-1011, 3 “For a system that needs to boot out of Flash, execute code from the Flash, or if read latency is an issue, NOR Flash may be the answer”).) Additionally, NOR Flash guarantees 100% good bits “simplif[ying] implementation by removing the need for error correction.” (INTEL-1008, 22.)

34. As explained by Brewer, “[c]ode storage was the first usage and remains a large consumer of Flash memory.” (INTEL-1008, 38.) Code execution “requires random-access reads because code jumps from one memory location to another when branching.” (INTEL-1008, 22.) Virtually all types of microprocessor-based systems used Flash memory prior to the ’190 patent. (Brewer, 38.) In these systems, code storage often uses an execute in place (XIP) model. (INTEL-1008, 38.) XIP refers to the usage model “whereby executable code is stored in Flash memory, and the system microprocessor fetches instructions directly from the Flash memory subsystem for execution.” (INTEL-1008, 39.) Therefore, the primary attribute required for XIP is read access time, making NOR-type Flash ideally suited for code storage. (*Id.*) Specifically, random access gives NOR Flash its XIP functionality, which is often required in embedded applications. (*See* INTEL-1010, 4.)

35. “The strengths of NAND-type Flash memories are fast writes and high-density arrays.” (INTEL-1008, 23.) NAND FLASH “has poor random-access read performance, but sequential read performance is good.” (*Id.*) Unlike NOR, NAND does not guarantee 100% good bits, “but fixes this with error correction and cell redundancies.” (*Id.*) Memory cells in NAND Flash are organized serially. Random performance is slow due to the fact there are no contacts directly accessing the memory cells. However, because NAND Flash has only two contacts every 32 memory cells, the effective cell size is much smaller than NOR Flash. Therefore, the strengths of NAND-type Flash memories are fast writes and high-density arrays.” (*Id.*; *see also*, INTEL-1010, “The real benefits of NAND Flash are faster PROGRAM and ERASE times, as NAND Flash delivers sustained WRITE performance exceeding 5 BM/s. Block erase times are an impressive 2ms for NAND Flash compared with 750 ms for NOR Flash”).) Because of its fast write performance, NAND is suited to “data-only applications such as audio storage, video storage, or data logging where fast data writes are required and the data will be accessed sequentially.” (INTEL-1008, 23; *see also*, INTEL-1011, 3 “for storage applications, NAND Flash’s higher density, and high programming and erase speeds make it the best choice”).) For example, NAND is “well suited for storage of sequential data such as pictures, video, audio, or PC Data.” (INTEL-1010, 3.)

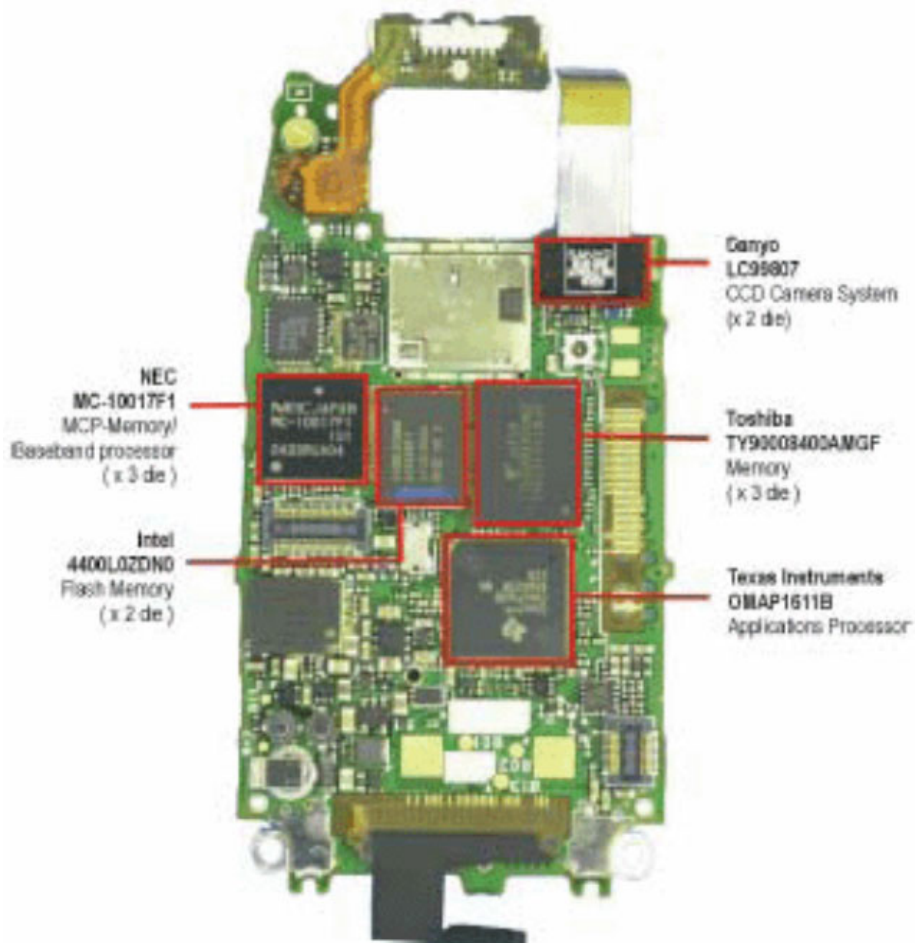
36. The following Table from NAND 101 summarizes the advantages and disadvantage of each type of device.

Table 1: NAND Flash vs. NOR Flash

	NAND	NOR
Advantages	Fast WRITES	Random access
	Fast ERASEs	Byte WRITES possible
Disadvantages	Slow random access	Slow WRITES
	Byte WRITES difficult	Slow ERASEs
Applications	File (disk) applications	Replacement of EPROM
	Voice, data, video recorder	Execute directly from nonvolatile memory
	Any large sequential data	

37. A single device or application is not limited to using only one type of Flash. For example, cell phone controllers in existence before the '190 patent included support for **both** NAND and NOR Flash. (See INTEL-1011, 1; 4 “When a system, such as a camera phone, has a requirement for code execution and high capacity data storage, designers may need to consider alternatives and tradeoffs, such as using both types of Flash memory”).) Toshiba explains that “data storage capacity and performance requirements in cell phones have increased significantly with the growth of feature-rich phones that incorporate camera, music, video, gaming and other functionality.” (INTEL-1011, 1.) NAND Flash is an “attractive alternative for the data storage aspects of today’s cell phones because of its higher speed write and erase performance as well as its low cost-per-bit.” (*Id.*) Thus, a POSITA would know how to configure such a phone during manufacture with code stored in NOR flash and large data files stored in NAND flash.”

38. To meet these demands, Toshiba offered a Multi Chip Package (MCP) that combined NAND flash, NOR flash, Pseudo SRAM, and/or low power DRAM in a single package using chip-stacking technology. (INTEL-1011, 4.) Indeed, memory combining multiple flash NOR and NAND with RAM in a single package was “widely” in use in cell phone applications at least 3 years before the ’190 patent. (INTEL-1013, 1.) DiPaolo notes that 15% of MCPs available for the handset market in 2005 incorporated NAND and most retained NOR for code execution including the NEC handset illustrated below which utilized Intel’s NOR flash memory and Toshiba’s NAND flash. (See INTEL-1009, 2-3.)



39. The above wireless phone applications were not the only applications to use both NAND and NOR devices together. A hybrid flash file system (HFFS) was designed based on both NOR flash and NAND flash memory prior to the '190 patent. (See INTEL-1014, 1002.) Lee explains that in embedded systems, “the major file operations are the updating or appending of small data, yet most of the capacity is occupied by large files.” (*Id.*) For example, in a mobile phone, “small records such as text messages and an address book are updated sporadically, while

comparatively large images and media files occupy most of the storage area.” (*Id.*)

A need therefore existed to optimize storage for both small and large data files.

40. The architecture below illustrates the use of both NAND and NOR flash in the HHFS. (INTEL-1014, 1004.) In this architecture, during a write operation, the size of the data is checked first. “Only data that is smaller than a predetermined threshold is stored as a log in the NOR flash; larger amounts of data are stored in the NAND flash.” (INTEL-1014, 1005.)

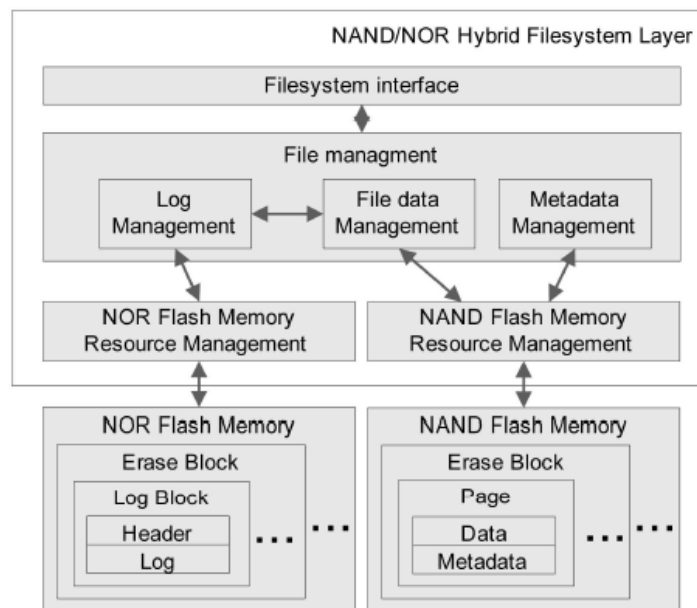


Fig. 2. The architecture of a hybrid flash file system (HHFS).

2. Memory Controllers

41. When the central processor (also called the "host") of a system accesses memory to read or write data, the access request is communicated in the form of a transaction. A transaction begins with the central processor sending an address and

a request to memory. The address specifies a location in memory. For a write access, the central processor also sends the data to be written. For a read access, the data is returned from the memory to the central processor. A memory transaction is commonly handled by a memory controller, which acts as an intermediary between the central processor and the memory and performs the physical memory access. (INTEL-1021, 1, Figure 1.) The use of a controller situated between the central processor and memory is beneficial because this arrangement hides many details of the physical memory from the processor, and allows the processor to be used in systems with different types of memory. (INTEL-1021, 1.) Additionally, in systems where additional processors and/or devices can also access the memory, the controller can optimize memory accesses that are interleaved from various sources, such as the central processor, a graphics processor, and peripheral devices (such as a disk or network interface).

42. Generally speaking, the controller handles flash management functions so they need not be handled by the host. (INTEL-1021, 1.) The flash management functions include the following:

- **Address translation.** The controller can keep track of address mappings between transaction addresses and the physical address used to access the flash memory chips. Translating addresses allows data in the flash to be moved from one location to another as it is re-written, or to be written to various locations, to improve speed, to delay erasure, to

improve reliability, or to perform wear leveling, as discussed below. The controller maintains tables to store the mappings for translating addresses.

- **Garbage collection.** The controller can track whether a block needs to be erased. The controller performs “garbage collection” (erasing blocks no longer in use) in the background as read and write accesses are performed to other locations. This improves the speed of the system.
- **Wear leveling.** The controller can track how often a block has been erased, and perform “wear leveling” (spreading usage across blocks, avoiding excessive erasure of particular blocks). (See INTEL-1021, 1.) This improves reliability.
- **Bad block reallocation.** The controller can track whether writes to a page are successful. In addition, the controller can track whether multiple bits in a page require error correction. If writes are unsuccessful or if multiple bits require error collection, this is an indication that the block may have deteriorating performance. The controller can use this information to identify bad blocks to reallocate (move to another location). (See INTEL-1021, 1.) This improves reliability.

3. Memory Access Patterns

43. Computer engineers have long known that memory accesses can exhibit patterns with different performance characteristics. For example, random and sequential accesses have different performance characteristics, and that

performance can be optimized according to the type of memory used and its parameters. Sequential access refers to accessing storage locations in a predefined order, such as from consecutive locations; random access refers to accessing storage locations with no predefined order. (*See, e.g., INTEL-1015¹, 621, 680.*) Generally, it has also been known that performance for random accesses benefits from transferring smaller amounts of data (just the data required) and performance for sequential accesses benefits from transferring larger amounts of data (prefetching additional data that will be used). Several techniques have been developed to optimize performance for random and sequential memory accesses; I discuss one of them below.

44. Storage systems for databases, file servers, and other applications allow control over the block size, which in this context, generally for hard disk drives (HDDs), means the smallest unit of data transfer between the storage system and main memory. A POSITA would have known that certain types of applications with small, random accesses, benefit from smaller block sizes that avoid reading and writing unneeded data associated with larger block sizes. (*See, e.g., INTEL-1016, 8* (“The ideal block size depends on the nature of the applications; large

¹ For ease of discussion, I cite to the page numbers from the original document and not the page numbers added.

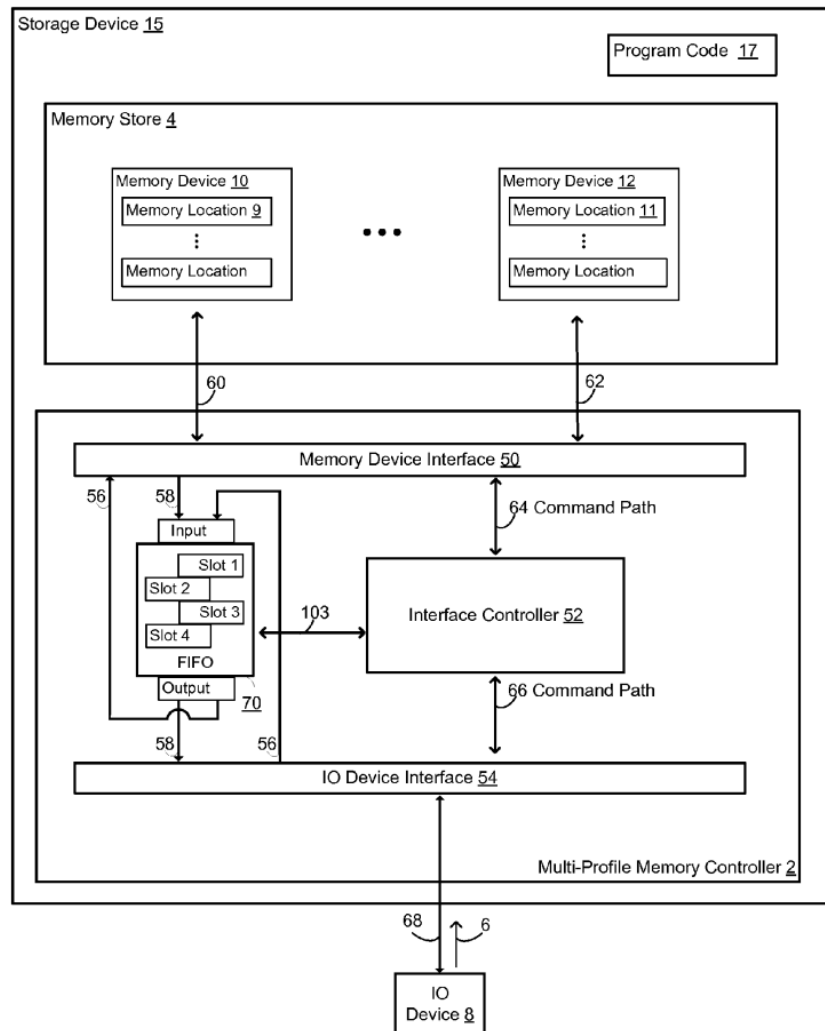
blocks are preferable for applications that do long sequential reads, such as streaming multimedia, and smaller blocks are useful for nonsequential applications, such as databases”).) Examples of such applications include (i) On-Line Transaction Processing (OLTP), where many records are stored, but each transaction only accesses individual, small records scattered throughout the database, and (ii) certain web servers that commonly handle large numbers of small files. In contrast, applications that handle large, sequential accesses benefit from larger block sizes, so that each access reads or writes large amounts of useful data. Common examples of such applications include databases for Decision Support System (DSS) and for Enterprise Data Warehouse (EDW).

B. The '190 Patent

45. The '190 patent “relates to multi-profile memory controllers and computing devices, such as solid-state storage devices, that use these multi-profile memory controllers.” (INTEL-1001, 1:17-19.) The multi-profile memory controller 2, illustrated in Figure 1 below, “perform[s] a memory transaction on a memory store 4 in response to receiving a memory transaction request 6” initiated by an IO device 8. (INTEL-1001, 2:50-55.) As shown in Figure 1, the memory store includes multiple memory devices which the '190 patent defines as “any type of memory device that can write to or read data from an addressable location in

response to a memory device command, such as a read or write command.”

(INTEL-1001, 4:34-38.)



'190 Patent, Figure 1

46. The '190 patent describes its memory devices may be “implemented using a NAND flash memory device that complies with the Open NAND Flash Interface Specification, commonly referred to as ONFI Specification.” (INTEL-1001, 4:50-53.) The specification also references other existing memory standards

such as those created by the Non-Volatile Memory Host Controller Interface (“NVMHCI”) working group. (INTEL-1001, 4:61-67.)

47. The memory store “includes addressable memory locations, such as memory locations 9 and 11, that are respectively associated with a set of at least one memory attribute.” (INTEL-1001, 2:63-65.) The set of memory attributes for a device is organized into a device profile, as shown in Figure 2A below. The attributes “include a page size 16, a block size 18, a memory protocol 20 supported by the memory device in which the memory location is addressable, a memory device type 22, and a partial page size 23.” (INTEL-1001, 3:25-29.) Figure 2B and 2C illustrate exemplary device profiles having shared attributes (page size, block size, memory protocol, device type, and partial page size). (*See, e.g.*, INTEL-1001, 4:8-11.) The exemplary profiles differ “by attribute qualifier for their sets of shared attributes.” (INTEL-1001, 4:11-4:14.) For example, page sizes, block sizes and partial page sizes “differ by their attribute values” and memory protocols and memory device types “differ by their attribute types.” (INTEL-1001, 4:14-17.)

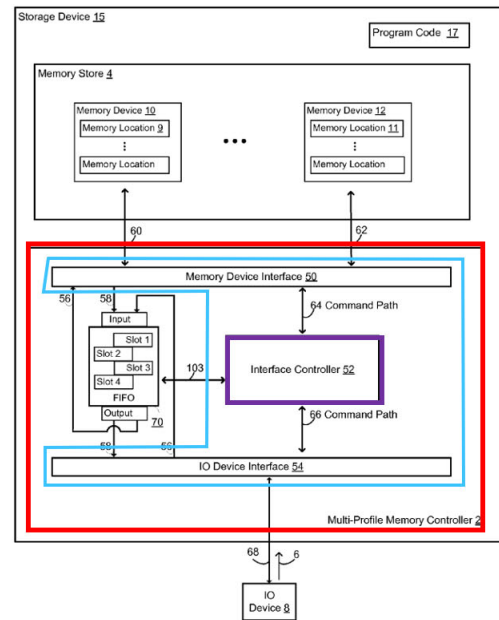
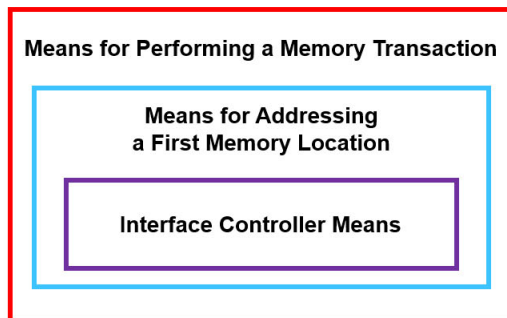
C. Claim Construction.

48. I understand that the Patent Office applies the claim construction standard used in district courts which seeks to give claim terms their plain and ordinary meaning to a person of ordinary skill in the art at the time of the claimed invention. I also understand that if the patent applicant gave a term a special

meaning in the specification or during prosecution, then the special meaning should be used.

49. Claim 45 is directed to a “*memory controller*” and recites a set of nested means-plus-function limitations: “*means for performing a memory transaction comprising a means for addressing a first memory location ... said means for addressing includes an interface controller means for processing memory transaction requests*” as illustrated below left.² I address the construction of those terms below. I apply the plain and ordinary meaning to each of the remaining claim terms in the challenged claims.

² I understand that a “means-plus-function” claim term is a term that recites a particular function but does not recite a specific structure for performing that function. I understand that means-plus-function terms are limited to the function recited in the claim(s) and the specific structure disclosed in the patent’s specification for performing that function, as well as structures equivalent to the disclosed structures.



1. “Means for Performing a Memory Transaction”

50. The function performed by the “means for performing a memory transaction” [45A] is “performing a memory transaction.” (INTEL-1001, 2:50-52.) Claim 45 further recites that “performing the memory transaction” includes “identifying command details for causing the memory transaction to be performed.” (INTEL-1001, 17:38-41 (claim 45).) Claim 45 does not expressly recite whether the “means for performing” performs this function or another recited component of the claims. (INTEL-1001, 17:38-41 (claim 45).)

51. The structure for the “means for performing” is the memory controller (indicated by red box in the above figures). (See, e.g., INTEL-1001, 2:50-52 (“FIG. 1 illustrates a multi-profile memory controller 2 for performing a memory

transaction on a memory store 4 ...”) I note this is consistent with the preamble which recites a “*memory controller for performing memory transactions.*”

2. “*Means for Addressing*”

52. The function of the “*means for addressing*” is “addressing a first memory location”, “obtain[ing] the first set of attributes after identifying the command details”, “us[ing] attributes from said first and second device profiles”, and “select[ing] a transfer size for the memory transaction.” (INTEL-1001, 7:27-30, 7:35-36 (addressing); 6:25-30; 6:50-63 (obtaining attributes), 8:39-42 (using attributes and selecting transfer size); 11:35-37 (selecting transfer size).)

53. The structure for the “*means for addressing*” is the interface controller, memory device interface, and IO interface (highlighted by blue box above). In the ’190 patent, the interface controller translates the received logical address into a physical address (7:35-36) and selects a transfer size for the memory transaction (*see, e.g.*, 11:35-37). Because the memory device interface provides signaling to the attached memory devices, the memory device interface also performs aspects of addressing a memory location and obtaining the set of attributes which are stored in the memory device. (*See*, INTEL-1001, 6:25-30; 6:50-63.) Therefore, the “*means for addressing*” includes the interface controller and the memory device interface. Because claim 45 further requires the “*means for addressing*” is also

“*directly coupled to a host*”, the structure further includes the IO interface to the host.

3. “*Interface Controller Means*”

54. The function of the “*interface controller means*” is “processing memory transaction requests.” (INTEL-1001, 7:21-24.)

55. The associated structure is an interface controller (highlighted by purple box above). (*See, e.g.*, INTEL-1001, 7:21-24 (“Interface controller 52 processes memory transaction request 6 by interpreting the memory transaction request and performing a memory transaction on a memory location.”); 7:35-36 (“interface controller 52 translates this logical address into a physical address”).)

D. Level of Ordinary Skill in the Art

56. I understand that certain issues relating to the validity of the ’190 patent must be judged from the perspective of a person of ordinary skill in the relevant art, as I discuss below. I have been asked to define the level of a “person of ordinary skill in the art” or “POSITA” that someone would have had at the time the alleged invention as claimed was made. In §III(B), I provided an overview of the ’190 patent. In deciding the level of ordinary skill, I have considered the following factors, which I have been informed are relevant to the determination of the level of ordinary skill in the art.

- the levels of education and experience of persons working in the field;

- the types of problems encountered in the art;
- the prior art solutions to these problems;
- the rapidity with which innovations are made; and
- the sophistication of the technology

57. A person of ordinary skill in the art (“POSITA”) would have had at least a bachelor’s degree in electrical or computer engineering and two years of experience working with storage systems (including storage systems made up of different types of storage devices), file management policies, and different data access patterns, or an equivalent combination of education and experience.

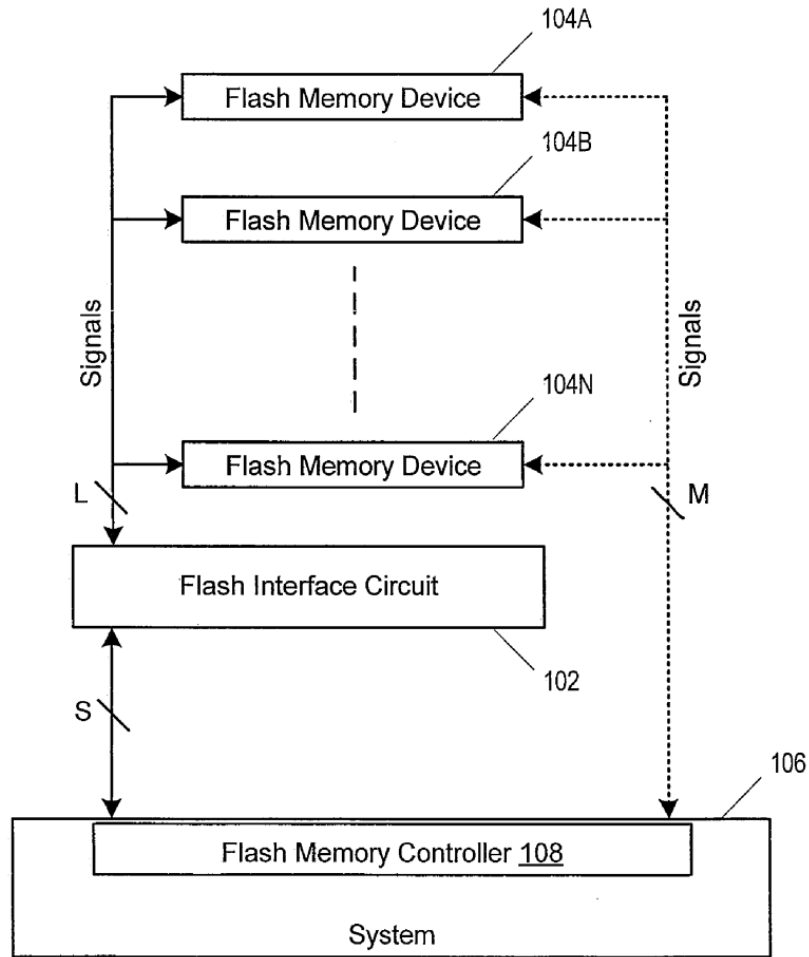
IV. Combination of Danilak and Cornwell

A. Overview of the Combination

1. Danilak

58. Danilak “relates to memory, and more particularly to enhanced capacity, performance, flexibility, and reliability in multiple flash memory circuit systems.” (INTEL-1005, ¶2.) Danilak discloses two embodiments of a memory system using a flash interface circuit. In the first embodiment, illustrated in Figure 1 below, several flash memory devices 104A-104N are “connected to a system 106 by way of a flash interface circuit 102.” (INTEL-1005, ¶17.) This system also includes a flash memory controller 108. The flash memory devices 104A-104N are connected to the flash interface circuit by a data bus, as shown in Figure 1. (*See*

INTEL-1005, ¶18.) The flash interface circuit is also connected to the host system 106 via a data bus. (*See Id.*)



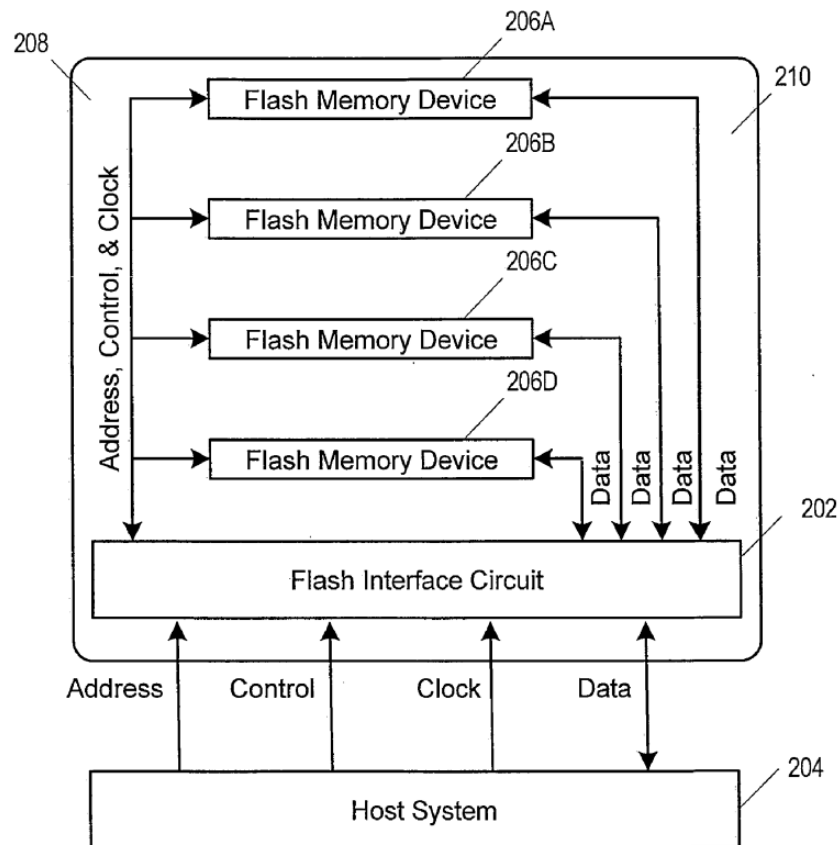
Danilak, Figure 1

59. Flash interface circuit 102 “exposes multiple flash memory devices attached to the flash interface circuit 102 as at least one flash memory device”, referred to as a “virtual flash memory device” to the rest of the system (e.g. the flash memory controller).” (INTEL-1005, ¶17.) “For example, the flash interface circuit may expose 1, 2, 4, or 8 attached flash memory devices 104A-104N to the

host system as 1, 2 or 4 flash memory devices.” (INTEL-1005, ¶19.) As such, “[o]ne or more attributes of the virtual flash memory device(s) may differ from the attributes of the flash memory devices 104A-140N.” (INTEL-1005, ¶17.) In an embodiment, “flash interface circuit 102 may provide vendor-specific protocol translation between attached flash memory devices and may present itself to host as a different type of flash, or a different configuration, or as a different vendor's flash device.” (INTEL-1005, ¶20.) In other examples, flash interface circuit 102 “present[s] a virtual configuration to the host system emulating one or more of the following attributes: a desired (smaller or larger) page size, a desired (wider or narrower) bus width, a desired (smaller or larger) block size, a desired redundant storage area (e.g. 16 bytes per 512 bytes), a desired plane size (e.g. 2 Gigabytes), a desired (faster) access time with slower attached devices, a desired cache size, a desired interleave configuration, auto configuration, and open NAND flash interface (ONFI).” (*Id.*)

60. In Danilak’s second embodiment, illustrated in Figure 2 below, “all signals from the host system are received by the flash interface circuit before presentation to the flash memory devices” and “all signals from the flash memory devices are received by the flash interface circuit before being presented to the host system 204.” (INTEL-1005, ¶22.) The primary difference between this embodiment and the embodiment of Figure 1 is that the system does not include a

separate memory controller in the host. Therefore, in this embodiment, the memory management functions performed by the memory controller are assumed to be performed in the flash interface circuit. As I discussed further in my analysis of the claims, Danilak discloses and/or motivates the inclusion of these functions in the flash interface circuit. I note that in this embodiment, the flash interface circuit acts as the sole “controller” for the flash memory devices.



Danilak, Figure 2

61. Danilak’s flash interface circuit, illustrated in Figure 5, includes an interface to the host 501, and interface to the flash memory devices 502,

configuration logic 503, configuration block 504, read-only memory (ROM) block 505, flash discover block 506, discovery logic 507, address translation unit 508, and unit for translations other than address translations 509. (*See* INTEL-1005, ¶55.) This flash interface circuit may be flash interface circuit 102 or flash interface circuit 202. (INTEL-1005, ¶23.)

62. Translation units 508/509 “translate host flash memory access and configuration requests into requests to one or more flash memory devices, and may translate flash memory replies to host system replies if needed.” (INTEL-1005, ¶56.) Translation units 508/509 “modify requests provided from the host system based on differences between the virtual configuration presented by the interface circuit 500 to the host system and the physical configuration of the flash memory devices, as determined by the discovery logic 507 and/or the configuration logic 503 and stored in the configuration block 504 and/or the discovery block 506.” (*Id.*) Configuration block 504, ROM block 505, and/or flash discovery block 506 “store data identifying the physical and virtual configurations.” (*Id.*)

63. In Danilak, the physical configuration of each flash memory device must be determined. Danilak teaches that “[t]here are many techniques for determining the physical configuration, and various embodiments may implement one or more of the techniques.” (INTEL-1005, ¶57.) First, configuration can be determined “using a discovery process implemented by the discovery logic 507.”

(*Id.*) This technique “may be selected using an auto configuration signal.” (*Id.*)
Second, “[f]ixed configuration information may be programmed into the ROM
block 505.” (*Id.*)

64. Configuration block (CB) 504 “stores the virtual configuration.”
(INTEL-1005, ¶58.) The embodiment of CB 504, illustrated in Figure 7 below,
“comprises 8 bytes of attributes, labeled 0 to 7” and “having various bit fields.”
(INTEL-1005, ¶61.)

700

Byte	Bits	Mnemonic	Description
0	0	AUTO	Auto discovery (No, Auto)
	1	ONFI	ONFI standard support (No, Yes)
	3:2	CHIPS	How many chip selects exposed (Auto, 1, 2, 4)
	7:4	—	Reserved
1	8	MKR	External maker code reported by flash interface circuit to host.
2	8	DEV	External device code reported by flash interface circuit to host.
3	1:0	CIPN	Internal chip number (1, 2, 4, 8)
	3:2	CELL	Cell type (2, 4, 8, 16 levels)
	5:4	SIMP	# of simultaneously programmed pages (1, 2, 4, 8)
	6	INTRL	Interleave between multiple chips (No, Yes)
	7	CACHE	Cache program (No, Yes)
4	1:0	PAGE	Page size (1, 2, 4, 8 KB)
	2	RSIZE	Redundant area size (8, 16 bytes / 512 bytes)
	7&3	SMIN	Serial access minimum (50/30, 25 ns)
	5:4	BSIZE	Block size (64, 128, 256, 512 KB)
	6	ORG	Organization (x8, x16)
5	1:0	—	Reserved
	3:2	PLANE	Plane number
	6:4	PLSIZE	Plane size (64, 128, 256, 512, 1Gb, 2Gb, 4Gb, 8Gb)
	7	—	Reserved
6	7:0	—	Reserved
7	7:0	—	Reserved

Danilak, Figure 7

- “**Byte zero** includes a auto discover bit (AUTO), indicating whether or not auto discovery is used to identify the configuration data; an ONFI bit indicating if ONFI is supported; and a chips field (CHIPS) indicating how many chip selects are exposed (automatic, 1, 2, or 4 in this embodiment, although other variations are contemplated).” (INTEL-1005, ¶62.)

- “**Byte one** is a code indicate the manufacturer (maker) of the device (or the maker reported to the host); and **byte two** is a device code identifying the particular device from that manufacturer.” (INTEL-1005, ¶62.)
- “**Byte three** includes a chip number field (CIPN) indicating the number of chips that are internal to flash memory system (e.g. stacked with the flash interface circuit or integrated on the same substrate as the interface circuit, in some embodiments).” (INTEL-1005, ¶63.)
 “Byte three also includes a cell field (CELL) identifying the cell type, for embodiments that support multilevel cells”; “simultaneously programmed field (SIMP) indicates the number of simultaneously programmed pages for the flash memory system”; “interleave bit (INTRL) indicates whether or not chip interleave is supported”; and the “cache bit (CACHE) indicates whether or not caching is supported.” (*Id.*)
- “**Byte four** includes a page size field (PAGE), a redundancy size bit (RSIZE) indicating the amount of redundancy supported (e.g. 8 or 16 bytes of redundancy per 512 bytes, in this embodiment), bits (SMIN) indicating minimum timings for serial access, a block size field (BSIZE) indicating the block size, and an organization byte (ORG) indicating the data width organization (e.g. $\times 8$ or $\times 16$, in this embodiment, although other widths are contemplated).” (INTEL-1005, ¶64.)
- “**Byte five** includes plane number and plane size fields (PLANE and PLSIZE).” (*Id.*)

65. ROM block 505 comprises “one or more instances of the configuration data in table 800 for various attached flash devices and for the configuration presented to the host system.” (INTEL-1005, ¶66.) The configuration data, illustrated in Figure 8 below, “is a subset of the data stored in the configuration block.” (*Id.*)

800

Byte	Bits	Mnemonic	Description
1	8	MKR	External maker code reported by flash interface circuit to host.
2	8	DEV	External device code reported by flash interface circuit to host.
3	1:0	CIPN	Internal chip number (1, 2, 4, 8)
	3:2	CELL	Cell type (2, 4, 8, 16 levels)
	5:4	SIMP	# of simultaneously programmed pages (1, 2, 4, 8)
	6	INTRL	Interleave between multiple chips (No, Yes)
	7	CACHE	Cache program (No, Yes)
4	1:0	PAGE	Page size (1, 2, 4, 8 KB)
	2	RSIZE	Redundant area size (8, 16 bytes / 512 bytes)
	7&3	SMIN	Serial access minimum (50/30, 25 ns)
	5:4	BSIZE	Block size (64, 128, 256, 512 KB)
	6	ORG	Organization (x8, x16)
5	1:0	–	Reserved
	3:2	PLANE	Plane number
	6:4	PLSIZE	Plane size (64, 128, 256, 512, 1Gb, 2Gb, 4Gb, 8Gb)
	7	–	Reserved
–	7:0	–	Reserved

Danilak, Figure 8

66. Danilak’s discovery block “comprise[s] one or more instances of the configuration data in table 900”, illustrated in Figure 9 below, “for various

attached flash devices.” (INTEL-1005, ¶67.) The configuration data “is a subset of the data stored in the configuration block.” (*Id.*) This information “is discovered using one or more read operations to the attached flash memory devices, initiated by the discovery logic 507.” (INTEL-1005, ¶68.)

900

Byte	Bits	Mnemonic	Description
0	0	—	Reserved
	1	ONFI	ONFI standard support (No, Yes)
	3:2	CHIPS	Number of stacked chips (1, 2, 4, 8)
	7:4	—	Reserved
1	8	MKR	External maker code reported by flash interface circuit to host.
2	8	DEV	External device code reported by flash interface circuit to host.
3	1:0	CIPN	Internal chip number (1, 2, 4, 8)
	3:2	CELL	Cell type (2, 4, 8, 16 levels)
	5:4	SIMP	# of simultaneously programmed pages (1, 2, 4, 8)
	6	INTRL	Interleave between multiple chips (No, Yes)
	7	CACHE	Cache program (No, Yes)
4	1:0	PAGE	Page size (1, 2, 4, 8 KB)
	2	RSIZE	Redundant area size (8, 16 bytes / 512 bytes)
	7&3	SMIN	Serial access minimum (50/30, 25 ns)
	5:4	BSIZE	Block size (64, 128, 256, 512 KB)
	6	ORG	Organization (x8, x16)
5	1:0	—	Reserved
	3:2	PLANE	Plane number
	6:4	PLSIZE	Plane size (64, 128, 256, 512, 1Gb, 2Gb, 4Gb, 8Gb)
	7	—	Reserved
6	7:0	—	Reserved

Danilak, Figure 9

67. In operation, illustrated in Figure 10 below, the flash interface circuit “wait[s] for the host system to attempt flash discovery.” (INTEL-1005, ¶77.) When

flash discovery is requested from the host, the flash interface circuit “perform[s] device discovery/configuration for the physical flash memory devices coupled to the flash interface circuit.” (*Id.*) The flash interface circuit “determine[s] the virtual configuration based on the discovered flash memory devices and/or other data (e.g. ROM data).” (*Id.*) The flash interface circuit then “report[s] the virtual configuration to the host”, “exposing the virtual configuration to the host rather than the physical configuration.” (*Id.*)

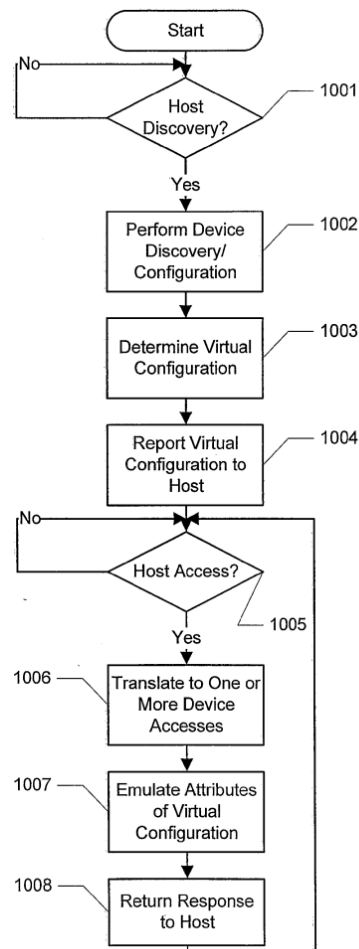


Fig. 10

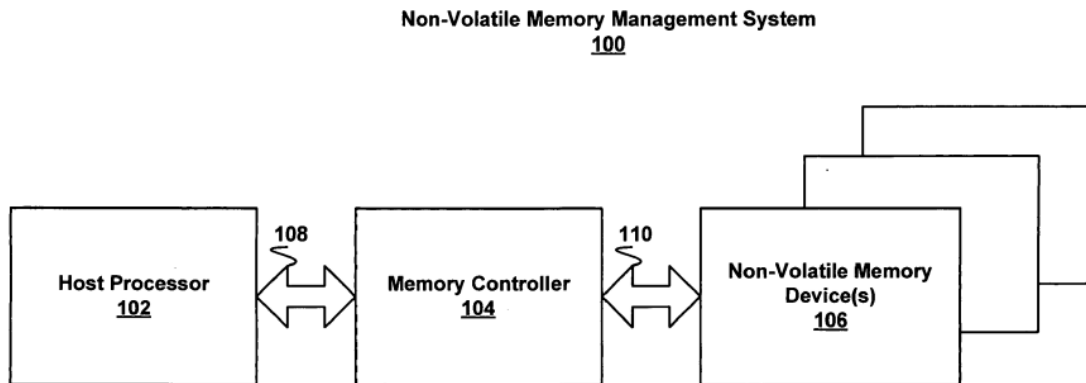
Danilak, Figure 10

68. When the host makes a memory access request (1005), the flash interface circuit “translate[s] the request into one or more physical flash memory device accesses (block 1006), emulate[s] attributes of the virtual configuration that differ from the physical flash memory devices (block 1007), and return[s] an appropriate response to the request to the host (block 1008).” (INTEL-1005, ¶78.)

2. Cornwell

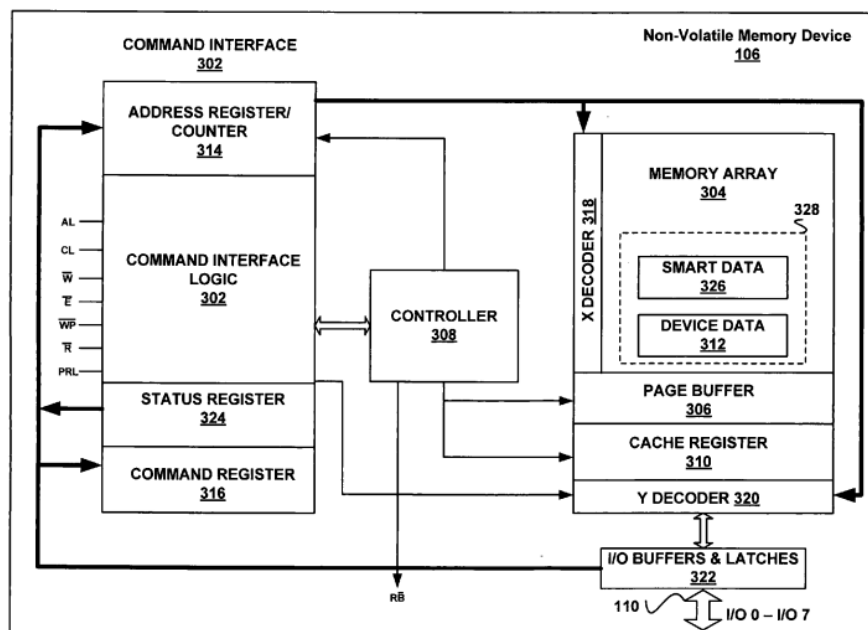
69. Cornwell relates to memory management, particularly in the context of non-volatile memory such as flash. (*See* INTEL-1006, ¶¶2, 5, Title.) Cornwell’s memory management system, illustrated in Figure 1 below, includes host 102, memory controller 104, and one or more non-volatile memory devices 106. (*See* INTEL-1006, ¶18.) Non-volatile memory devices 106 “include but are not limited to: NAND and/or NOR flash media, read-only memory (ROM), erasable, programmable ROM (EPROM), electrically-erasable, programmable ROM (EEPROM), Ferroelectric RAM (FeRAM), magnetoresistive RAM (MRAM), non-volatile static RAM (nvSRAM), and any other memory device that does not need its memory contents periodically refreshed and/or can retain information without power.” (INTEL-1006, ¶20.) Memory controller 104 “manages memory access.” (INTEL-1006, ¶19.) Specifically, memory controller 104 “recognizes control, address, and data signals transmitted on bus 108 by the host processor 102” and

“translates the control, address and data signals into memory access requests on memory devices 106.” (INTEL-1006, ¶21.)



Cornwell, Figure 1

70. Figure 3, reproduced below, is a block diagram of a non-volatile memory device 106. Device 106 generally includes command interface 302, memory array 304 and controller 308. (INTEL-1006, ¶34.) Command interface 302 includes command register 316, address register/counter 314 and status register 324. (*Id.*) Memory array 304 includes page buffer 306, optional cache register 310, x-decoder logic 318 and y-decoder logic 322. (*Id.*) I/O bus 110 includes eight I/O lines (I/O 0-I/O 7) “which are used to: (a) input a selected address, (b) output data during a read operation, or (c) input a command or data during a program operation.” (*Id.*) I note that the architecture of Figure 3 is consistent with well-known architecture for flash memory devices which I discussed in the background section.



Cornwell, Figure 3

71. Memory array 304 is accessed using x-decoder 318 and y-decoder 320. (INTEL-1006, ¶35.) “X-decoder 318 decodes input addresses in address register/counter 314 to determine a memory line to be accessed for the read or write operation.” (*Id.*) “Y-decoder 320 decodes signals from the command interface logic 302 for reading or writing data into the memory line determined by x-decoder 318.” (*Id.*)

72. Command register 316 “is configured to receive memory commands from the memory controller 104 via I/O bus 110.” (INTEL-1006, ¶37.) Address register/counter 314 “is configured to receive addresses from the memory controller 104 via I/O bus 110.” (*Id.*) “Thus, I/O bus 110 can receive either

command inputs or address inputs depending on the states of the AL and CL signals.” (*Id.*)

73. Controller 308 “is operatively coupled to the address register 314 and the command register 316 for receiving one or more input addresses and command inputs, which are used by the controller 308 in combination with control signals from the command interface logic 302 to carry out read and write operations on memory array 304.” (INTEL-1006, ¶38.)

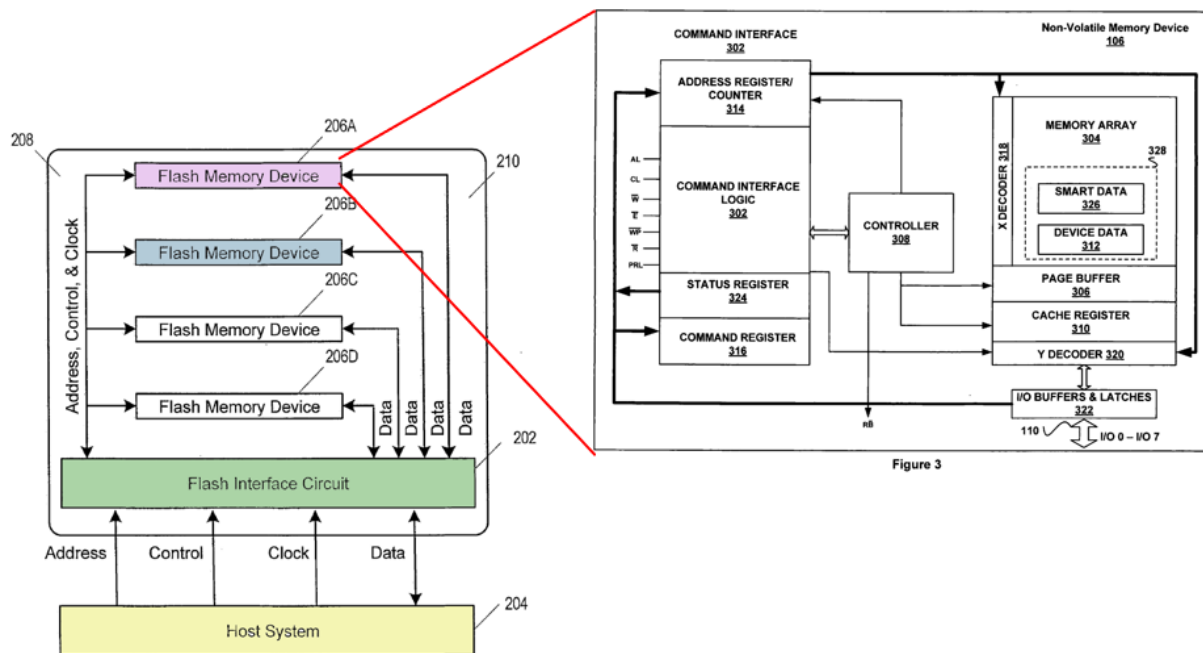
74. In Cornwell, device data “is stored in a spare area 328 of the memory array 304”, illustrated in Figure 3 above. (INTEL-1006, ¶42.) The device data “can be used to identify the memory device 106 and its manufacturer.” (*Id.*) The “data 312 can also include but is not limited to: device type (e.g., NAND, NOR, etc.), device density (e.g., 512 Mb, 1 Gb, 2 Gb, etc.), device operating voltage (e.g., 3.3 volts), page size (1 k, 2K, etc.), spare area size (e.g., 8, 16 bytes, etc.), sequential access time (e.g., 30, 50 nanoseconds, etc.), block size (e.g., 64 k, 128 k, etc.), bus width (e.g., ×8, ×16, etc.), bad block identification, and any other information that is associated with attributes, properties or characteristics of the memory device 106 (collectively, referred to herein as “attributes”).” (*Id.*) The device data is “transmitted to the memory controller 104 via the I/O bus 110 in response to a read command issued by the memory controller 104.” (INTEL-1006, ¶43.)

3. Combination of Danilak and Cornwell

75. A POSITA would have been motivated to combine the teachings of Danilak and the teachings of Cornwell. Specifically, a POSITA would have been motivated to combine Cornwell's teachings regarding the structure of its non-volatile memory with Danilak's flash memory devices and would have been motivated to use both NAND and NOR flash devices in the combined memory system. I address the motivations to make each of these combinations individually in this section.

a) Cornwell's Non-Volatile Memory Structure

76. A POSITA would have been motivated to combine Cornwell's teachings regarding the structure of non-volatile memory (including the use of device data stored in non-volatile memory) with Danilak's memory devices, as illustrated in the following figure which combines Danilak's Figure 2 with Cornwell's Figure 3. There are several reasons that a POSITA would have combined these teachings of Danilak and Cornwell.



Danilak, Annotated Figure 2 (left); Cornwell, Figure 3 (right)

77. First, Danilak discloses the use of flash memory devices 206A-206D but provides no details regarding the structure of its flash memory devices. A POSITA would have been therefore motivated to search for references providing these details and would have been led to Cornwell which describes non-volatile memory and the management of such memory. (See INTEL-1006, ¶¶2, 5, Title.)

78. Cornwell is in the same field as Danilak and the '190 patent—memory and memory device management. Danilak relates to “memory, and more particularly to enhanced capacity, performance, flexibility, and reliability in multiple flash memory circuit systems.” (INTEL-1005, ¶2.) Cornwell similarly “relate[s] to memory management” of non-volatile memory systems. (INTEL-1006, ¶2.) And the claims of the '190 patent are directed to a memory controller

that uses device profiles. (*See, e.g.*, INTEL-1001, claims 1, 29, 45, 59, 80, 98, 100.)

79. Second, a POSITA would have been motivated to combine the teachings of Cornwell with Danilak to create an operational memory system. For example, in addition to the lack of disclosure related to the structure of non-volatile memory, Danilak discloses configuration data for its flash memory devices “is discovered using one or more read operations to the attached flash memory devices.” (INTEL-1005, ¶68.) Danilak provides no detail regarding how the device-specific configuration data is discovered via a read of a flash device. This functionality is taught by Cornwell which describes storage of device data in a memory device. Danilak also teaches that its memory devices perform read and write operations but provides no specific details regarding where the device-specific data is stored or how a memory device processes these transactions. Cornwell’s non-volatile memory of Figure 3 and its corresponding discussion provides details on how a memory device handles such transactions. (*See, e.g.*, INTEL-1006, ¶¶34-43.)

80. Third, the integration of Cornwell’s non-volatile memory device architecture into Danilak’s system is merely the simple substitution of one known element (Cornwell’s non-volatile memory) for another (Danilak’s flash memory) to obtain predictable results. As I discuss in §§III(B)(1)-(6) below, Danilak

contains a memory device, computer program product, and method which differs from the device and method claimed in the '190 patent simply by the substitution of Cornwell's flash memory device for Danilak's flash memory device. A POSITA would have substituted Danilak's flash memory device for Cornwell's flash memory device as discussed above and the results of the substitution would have been predictable because Cornwell's flash memory devices use standard interfaces and perform flash memory operations in a predictable manner.

81. Indeed, I note that the ONFI specification, which the '190 patent mentions and acknowledges as prior art, describes a "standardized NAND Flash device interface" including support for a "Read Parameter Page" function. (INTEL-1007, 1, 91.) Specifically, the "Read Parameter Page function retrieves the data structure that describes the target's organization, features, timings and other behavioral patterns." (INTEL-1007, 91.) As shown in Table 38, the device data available in this data structure includes, for example, device identification information, number of pages per block, number of bytes per page, bits per cell, partial programming attributes. Similarly, flash devices, including NOR devices, that supported the CFI specification could be queried to return page and block size. (See INTEL-1023, 70, 73, 75.) This device data is consistent with what is stored in Cornwell's device data and Danilak's configuration data. A POSITA would have therefore understood, based on the existing standards and other available data, how

to access Cornwell's device data and perform memory operations on a flash device using Danilak's flash interface circuit.

b) NAND and NOR Flash Devices

82. A POSITA would have also been motivated to use both NAND and NOR Flash Devices in the combined memory of Danilak and Cornwell. Cornwell provides an express teaching that would have led a POSITA to use both NAND and NOR flash devices in the combined memory system. Specifically, Cornwell teaches examples of non-volatile memory devices used in its system “include but are not limited to: NAND and/or NOR flash media.” (INTEL-1006, ¶20.)

Cornwell provides a field in its device data for each memory device to identify its type: “Chip data 312 can also include but is not limited to: device type (e.g., **NAND, NOR, etc.**) ...” (INTEL-1006, ¶42.) The inclusion of the field further indicates an intent to support different types of devices in the same memory.

83. Danilak similar provides a suggestion to use different device types in its memory system. For example, Danilak teaches its “flash interface circuit 102 may provide vendor-specific protocol translation between attached flash memory devices and may present itself to host as a different type of flash, or a different configuration, or as a different vendor's flash device.” (INTEL-1005, ¶20.) Danilak specifically mentions support for both NOR and NAND protocols. Danilak explains that while previously, it was “difficult or impossible to design a flash

memory controller that is capable of controlling both NAND and NOR protocols” due to the fact that the protocols “generally are not compatible,” the invention of Danilak overcomes this problem because its “flash interface circuit may contain logic circuits that may translate between the NAND protocols that are in use by the flash memory and a NOR protocol that interfaces to a host system or CPU” and “an embodiment that provides compatibility with NAND flash may include a flash interface circuit that contains logic circuits to translate between the NOR protocols used by the flash memory and a NAND protocol that interfaces to a host system or CPU.” (INTEL-1005, ¶45.) The support for both protocols and the suggestion that flash memory devices could be either NAND or NOR would have at least suggested to a POSITA the inclusion of both types of memory in a memory system. As a further example, Danilak’s configuration data indicating the manufacturer and device code further suggests the use of different device types: “Byte one is a code indicate the manufacturer (maker) of the device (or the maker reported to the host); and byte two is a device code identifying the particular device from that manufacturer.” (INTEL-1005, ¶62.) The inclusion of manufacturer and device code by Danilak would have indicated to a POSITA support for different types of devices in the memory system because several vendors offered both NAND and NOR memory devices prior to the ’190 patent. I note that both Toshiba and Micron offered both NAND and NOR Flash devices

prior to the '190 patent. (*See, e.g.*, INTEL-1011, 2 (mentioning Toshiba's NAND and NOR Flash devices); INTEL-1010, 5 (mentioning Micron's NAND and NOR Flash devices).)

84. A POSITA would have been further motivated to use both NAND and NOR memory in a memory system to address a large and growing market in portable devices, particularly those requiring code execution and high-capacity storage, like smartphones. (See INTEL-1011, 4 (“When a system, such as a camera phone, has a requirement for code execution and high capacity data storage, designers may need to consider alternatives and tradeoffs, such as using both types of Flash memory”).) As I discuss in further detail in §III(A)(1) (Technical Background), it was well-known that the “strengths of NOR-type Flash memories are fast random-access reads and reliable storage.” (INTEL-1008, 22.) NOR flash, therefore, works well in any application where fast, reliable random-access reads are needed, like for code execution. (INTEL-1008, 22.) In contrast, the “strengths of NAND-type Flash memories are fast writes and high-density arrays”, making it well suited for data storage applications. (INTEL-1008, 23.) It was also known prior to the '190 patent that manufactures such as Toshiba offered Multi Chip Packages combining both NOR and NAND Flash in a single package. (See INTEL-1011, 4.) Indeed, memory combining multiple flash NOR and NAND with RAM in a single package was “widely” in use in cell phone applications at least 3

years before the '190 patent. (INTEL-1013, 1.) In 2005, 15% of MCPs available for the handset market incorporated NAND and most retained NOR for code execution including the NEC handset illustrated below which utilized Intel's NOR flash memory and Toshiba's NAND flash. (See INTEL-1009, 2-3.) Thus, a POSITA would have been motivated and understood how to combine NAND and NOR Flash devices in a single storage device to take advantage of their individual strengths.

85. A POSITA would have been further motivated to use both NAND and NOR Flash devices to support applications using both small data and larger files. (See, e.g., INTEL-1014, 1002) For example, in embedded systems, "the major file operations are the updating or appending of small data, yet most of the capacity is occupied by large files." (*Id.*) And in a mobile phone, "small records such as text messages and an address book are updated sporadically, while comparatively large images and media files occupy most of the storage area." (*Id.*) In such systems, applications that frequently update records having a much smaller size than the page size of a NAND Flash "greatly reduce the utilization of the medium" which in turn deteriorates the life span of the memory. (*Id.*) A POSITA would have been motivated to integrate NAND and NOR Flash devices together for a file system to optimize utilization, durability, and life span of the memory devices.

86. The combination is also nothing more than the combination of prior art elements (existing NAND flash and NOR flash devices in a memory system) according to known methods. Danilak expressly teaches that its flash interface circuit performs protocol translations for its attached memory devices. (*See* Danilak, ¶¶20, 45.) And Cornwell teaches that both NAND and NOR flash devices are supported in a memory system. (INTEL-1006, ¶20.) The '190 patent also admits that protocols for accessing non-volatile memory devices such as ONFI were known prior to its critical date. (*See*, INTEL-1001, 4:47-5:3.) The results of the combination would have been predictable because the interface protocols were known, Danilak supported both NOR and NAND protocols, and existing devices successfully integrated both NAND and NOR flash devices in the same memory system and supported protocols for both simultaneously.

B. Independent Claims

1. Independent Claim 1 (Method)

87. I have reproduced independent claim 1 below in its entirety with labels for each limitation reflecting the place of each limitation in the claim and its functional grouping. Independent claim 1 is a method claim having 5 method steps: [1A] “*addressing a first memory location*”, [1B] “*identifying command details*”, [1C] “*obtaining the first set of attributes*”, [1D] “*using attributes*” and [1E] “*selecting a transfer size.*” These method steps are applied in the context of a

memory store having recited structural limitations identified in the claim list as “structural”. Claim 1 also has 11 “*wherein*” clauses and 2 “*said*” clauses narrowing a prior limitation spread throughout the claims. These limitations of the claims can be grouped into 2 categories labeled in the claim as either “read-write” or “device profile”.

[1P] A method of performing memory transactions

[structural] on a set of memory locations that includes a first memory location and a second memory location,
the method comprising:

[1A/Action] performing a memory transaction comprising addressing a first memory location **[structural]** in a first memory device in a memory store,

[1A.1/Profile] said first memory location and a second memory location respectively associated with a first device profile and a second device profile,

[1A.2/Structural] wherein said second memory location is in a second memory device in the memory store,

[1A.3/Read-Write] wherein a memory read transaction or a memory write transaction is performed on at least one of the first memory location or second memory location,

[1A.4/Read-Write] wherein data is transferred to the first memory location or second memory location from a host for a memory write transaction,

[1A.5/Read-Write] wherein data is transferred from the first memory location or second memory location to the host for a memory read transaction;

[1A.6/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;

[1B/Action] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;

[1B.1/Profile] said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles;

[1C/Action] after identifying the command details, obtaining the first set of attributes;

[1C.1/Profile] wherein each attribute in the first set of attributes and second set of attributes is associated with a respective attribute qualifier that qualifies a respective memory location;

[1C.2/Structural] wherein the memory store is directly coupled to at least one memory bus and wherein the at least one memory bus is directly coupled to a controller, and

[1C.3/Read-Write] wherein the controller performs the memory read transaction and memory write transaction on the memory store;

[1C.4/Structural] wherein the host is directly coupled to the controller by a communication path **[Read-Write]** as the controller performs the memory read transaction or memory write transaction;

[1C.5/Profile] wherein the first device profile and second device profile are each stored in the memory store; and

[1D/Action] using attributes from said first and second device profiles; and

[1E/Action] selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.

88. For ease of discussion, after I address the structural limitations, I address the “device profile” limitation groupings followed by the preamble and method steps [1A]-[1E] and conclude with a discussion of the read-write limitations.

a) Structural Limitations

89. This section addresses the following structural limitations of claim 1:

[1P] *A method of performing memory transactions on a set of memory locations that includes a first memory location and a second memory location, the method comprising:*

...

[1A] *performing a memory transaction comprising addressing a first memory location in a first memory device in a memory store*

....

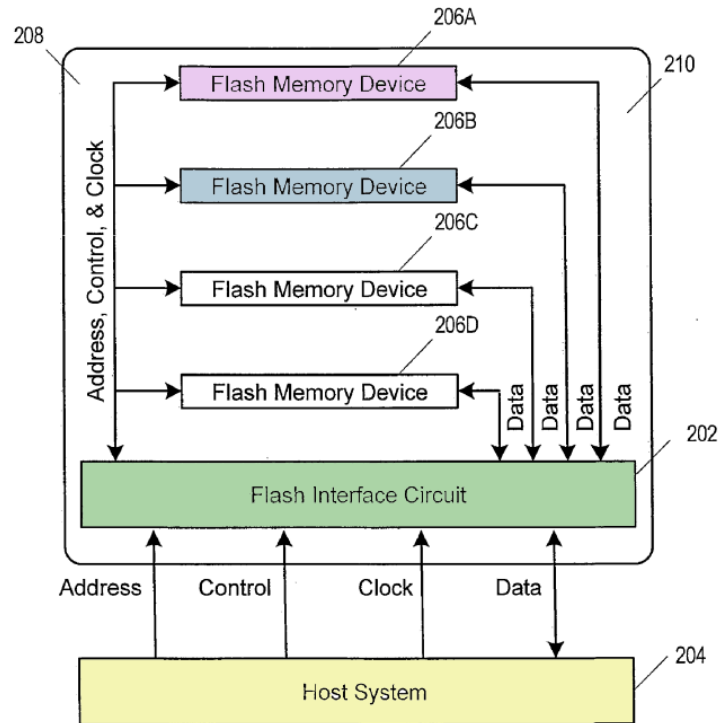
[1A.2/Structural] *wherein said second memory location is in a second memory device in the memory store,*

...

[1C.2/Structural] *wherein the memory store is directly coupled to at least one memory bus and wherein the at least one memory bus is directly coupled to a controller, and*

[1C.4/Structural] *wherein the host is directly coupled to the controller by a communication path as the controller performs the memory read transaction or memory write transaction;*

90. Danilak “relates to memory, and more particularly to enhanced capacity, performance, flexibility, and reliability in multiple flash memory circuit systems.” (INTEL-1005, ¶2.) An embodiment of Danilak’s memory is illustrated in Figure 2, reproduced below. As shown, the memory includes flash memory devices 206A-206D coupled to flash interface circuit 202 (shaded green) which is coupled to host system 204 (shaded yellow). (INTEL-1005, ¶22.)



Danilak, Figure 2

(1) “Memory Device”

91. Danilak’s flash memory devices are the recited “*memory device[s]*” of claim 1. One of these flash memory devices is a “*first memory device*” and one is a “*second memory device*.” For example, flash memory device 206A is the recited “*first memory device*” (shaded pink) and flash memory device 206B is the recited “*second memory device*” (shaded blue). (See, INTEL-1005, ¶22, Figure 2; see also, ¶¶17-18 (providing general discussion of memory system).) Danilak is consistent with the ’190 patent which states the “term ‘memory device’ refers to any type of memory device that can write to and read data from an addressable location in response to a memory device command, such as a read or write command.”

(INTEL-1001, 4:34-38.) Indeed, the '190 patent teaches that its memory devices can be NAND flash memory devices or NOR flash memory devices. (See INTEL-1001, 4:41-46.)

(2) “Memory Location”

92. Each of Danilak’s flash memory devices has one or more “*memory locations*.” The '190 patent uses the term “memory location” to refer to an addressable portion of a memory device: “Memory store 4 includes addressable memory locations.” (INTEL-1001, 2:63.) I agree that memory location encompasses an addressable portion of a physical memory device configured to store data.

93. Danilak teaches that a “flash memory is typically divided into sub-units, portions, or blocks.” (INTEL-1005, ¶24.) As shown in Figure 2 above, each flash memory device is coupled to an address bus over which the address of the byte or word to be read or programmed or the address of the sector to be erased is provided. (See INTEL-1005, ¶¶22, 69, INTEL-1006, 66-67.) Danilak also refers to “locations” within a memory device. For example, claim 7 of Danilak refers to “storage locations in the one or more flash memory devices.” (See INTEL-1005, claim 7.) And in its discussion of bad block relocation, Danilak states that “the combination of the flash interface circuit and flash memory together with the translation tables may act to ensure that the physical memory location that is

accessed is not in a bad block.” (INTEL-1005, ¶25.) Danilak therefore discloses that the “*first memory device*” has “*a first memory location*” and the “*second memory device*” has a “*second memory location.*”

94. Cornwell confirms that each of Danilak’s flash memory devices has one or more “*memory locations.*” Cornwell explains that “[f]lash memory is a type of non-volatile memory that stores information in an array of floating gate transistors called ‘cells’ which can store one or more bits.” (INTEL-1006, ¶3.) Flash memory “is divided into blocks.” (INTEL-1006, ¶3.) “A block is an array of memory cells organized into pages or sectors.” (*Id.*) Flash memory “can be read or programmed a byte or word at a time in a random access fashion” but “is usually erased a block at a time.” (INTEL-1006, ¶4.)

95. It is further inherent that flash memory devices (and in fact all memory devices) have one or more “*memory locations.*” The sole purpose of a memory device is to store information for an application and to provide that information as needed to an application. The ’190 patent in fact acknowledges that a “NAND flash memory device includes a set of addressable non-volatile memory cells, named flash blocks.” (INTEL-1001, 4:44-45.) As a further example, Brewer describes the architectural features of a basic flash memory in Figure 3.2 (below). Brewer explains that “[f]lash memory arrays are segmented into smaller pieces (subarrays), usually referred to as *blocks* or *sectors.*” (INTEL-1008, 66 (emphasis

in original).) The flash memory receives an “address of the byte or word to be read or programmed or the address of the sector to be erased” over the address bus shown below.

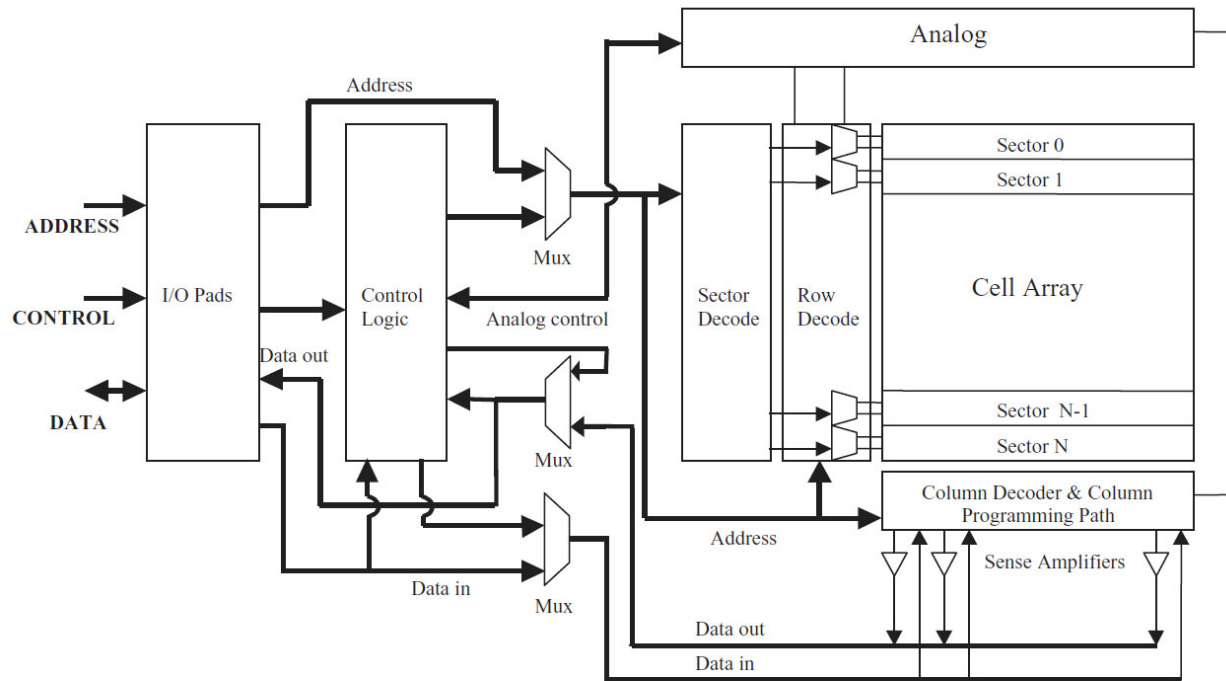


Figure 3.2. Flash memory circuit block diagram.

Brewer, Figure 3.2 Flash Memory Circuit Block Diagram

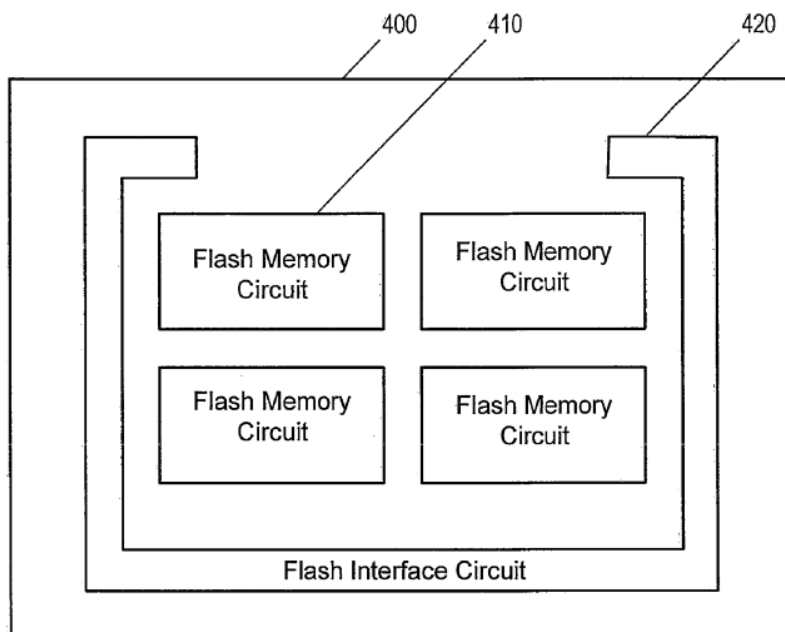
96. The Open NAND Flash Interface Specification, referred to as the “ONFI Specification”, also teaches that a flash memory device includes addressable “memory locations.” ONFI states that an “address is comprised of a row address and a column address.” (INTEL-1007, 2.) “The row address identifies the page, block, and LUN to be accessed” and the “column address identifies the byte or word within the page to access.” (INTEL-1007, 2.)

97. The combination of Danilak and Cornwell therefore discloses “*a first memory location in a first memory device*” [1A], “*a second memory location is in a second memory device*” [1A.2], and “*a set of memory locations that includes a first memory location and a second memory location*” [1P].

(3) “*Memory Store*”

98. In Danilak, the flash memory devices are exposed “as at least one flash memory device” to the rest of the system via flash interface circuit 102 (shaded green in Figure 2 above). (INTEL-1005, ¶17.) The set of flash memory devices exposed to the rest of the system is referred to as a “virtual flash memory device.” (INTEL-1005, ¶17; *see also*, ¶80 (“The interface circuit is configured to present at least one virtual flash memory device to the host system, and the interface circuit is configured to implement the virtual flash memory device using the one or more flash memory devices to which the interface circuit is coupled.”).) The virtual flash memory device comprised of multiple flash memory devices is a “*memory store*.”

99. Should an argument be made that the “*memory store*” must incorporate the memory devices into a single die, Danilak discloses such an embodiment. In Danilak, “the flash interface circuit may be integrated with one or more flash devices onto a single monolithic semiconductor die.” (INTEL-1005, ¶54.) Figure 4, reproduced below, illustrates “a die 400 including one or more flash memory circuits 410 and one or more flash interface circuits 420.” (*Id.*)



Danilak, Figure 4

100. Danilak’s disclosure regarding a “*memory store*” is consistent with the ’190 patent which discloses that a memory store includes memory devices “grouped so that each of the memory devices are associated with only one memory bus controller.” (INTEL-1001, 10:13-17.) The ’190 patent also indicates the “*memory store*” “includes addressable memory locations.” (INTEL-1001, 2:63.) The “*memory store*” “may be part of a computing device or equivalent device, such as a storage device 15, that includes multi-profile memory controller 2.” (INTEL-1001, 2:55-57.) Danilak is also consistent with the IEEE dictionary definition for a memory store: “3.2126 store. (1) A device into which data can be placed, in which they can be retained, and from which they can be retrieved.” (INTEL-1017, 89.)

(4) “*Controller*”

101. The controller in the ’190 patent “manage[s], control[s] or both the flow of data between the IO device 8 and memory store 4.” (INTEL-1001, 2:60-62.)

The flash interface circuit in the combined system of Danilak and Cornwell is the recited “*controller*” because it performs the same functions as the ’190 patent’s controller, managing and controlling access to memory devices of the memory store by the host, as well as performing other memory management functions.

The flash interface circuit manages and controls access to the memory devices in the memory store by the host.

102. Danilak discloses an embodiment, illustrated in Figure 2 above, having flash interface circuit 202 coupled between host system 204 and flash memory devices 206A-206D. (INTEL-1005, ¶22.) In this embodiment, “all signals from the host system are received by the flash interface circuit before presentation to the flash memory devices” and “all signals from the flash memory devices are received by the flash interface circuit before being presented to the host system 204.” (*Id.*) Unlike the embodiment of Danilak’s Figure 1, this embodiment does not include a separate flash memory controller. Therefore, a POSITA would have understood that memory controller functionality is implemented using the flash interface circuit. As I discuss further below, Danilak discusses the integration of this memory controller functionality into the flash interface circuit throughout its discussion.

103. As I discussed previously, Danilak's flash interface device exposes the set of flash memory devices to host as a single virtual flash memory device. (See INTEL-1005, ¶¶17, 51, 80.) Figure 10, reproduced below, depicts "flowchart illustrating one embodiment of a method of emulating one or more virtual flash memory devices using one or more physical flash memory devices having at least one differing attribute." (INTEL-1005, ¶14.) The discussion of Figure 10 refers to the "flash interface circuit." Danilak teaches the references to flash interface circuit in its specification "may be, in various embodiments, the flash interface circuit 102, the flash interface circuit 202, or other flash circuit embodiments (e.g., embodiments shown in FIGS. 3-6)." (INTEL-1005, ¶23.) Thus, flash interface circuit 202 of Figure 2 performs the operations discussed in Figure 10.

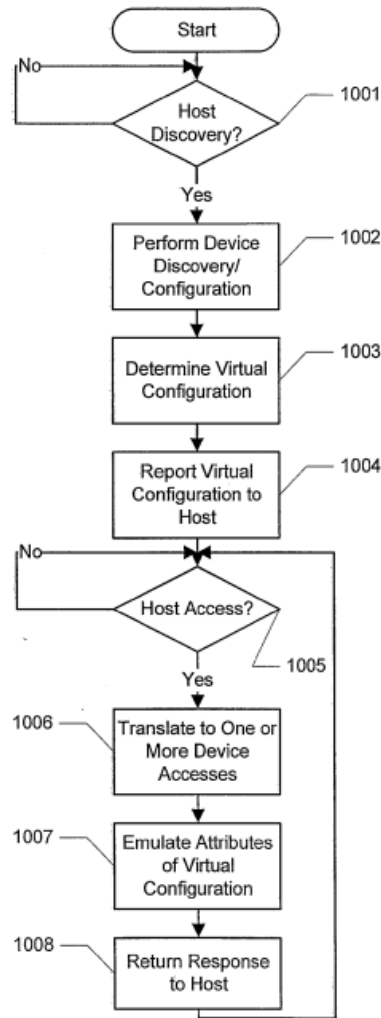


Fig. 10

Danilak, Figure 10

104. In operation, the flash interface circuit “translate[s] the request into one or more physical flash memory device accesses (block 1006), emulate[s] attributes of the virtual configuration that differ from the physical flash memory devices (block 1007), and return[s] an appropriate response to the request to the host (block 1008).” (INTEL-1005, ¶78; *see also*, ¶56.) The flash interface circuit may also, as needed, “translate a protocol implemented by the host system to a protocol

implemented by the one or more flash memory devices” and “translate the protocol implemented by the one or more flash memory devices to the protocol implemented by the host system.” (INTEL-1005, ¶80.)

105. Danilak teaches that the flash interface circuit performs the functions (or a subset of the functions) of the flash driver—making “flash memory appear to the host system as a read/write memory array” and supporting “basic file system functions (e.g. read, write, file open, file close etc.) and directory operation (e.g. create, open, close, copy etc.).” (INTEL-1005, ¶¶29-30.) Danilak explains that “[m]oving some or all of the flash driver functionality into a flash interface circuit enables the use of standard flash devices that do not have integrated flash driver capability and/or standard flash memory controllers that do not have integrated flash driver capability.” (INTEL-1005, ¶30.) Based on this express teaching in Danilak, a POSITA would have understood that the flash interface circuit 202 of Figure 2 includes flash driver functionality or would have been motivated to include such functionality in the flash interface circuit.

The flash interface circuit performs other memory management functions that control flow of data between host and memory devices.

106. Danilak also teaches the flash interface circuit performs memory management functions including increasing erase and modify bandwidth, reducing read latency by prefetching, increasing write bandwidth, and increasing bus bandwidth. (INTEL-1005, ¶¶35-42.) For example, to improve erase performance,

“logic circuits in the flash interface circuit may perform a pre-erase operation (e.g. advanced scheduling of erase operations, etc.).” (INTEL-1005, ¶36.) “[W]hen a future write operation is requested the block is already pre-erased and associated time delay is avoided.” (*Id.*) With prefetching, the flash interface circuit “read[s] data from the flash memory ahead of a request by the system.” (INTEL-1005, ¶38.) To increase write bandwidth, the flash interface circuit “hold[s] (e.g. buffer etc.) write requests in internal SRAM and write them into the multiple flash memory chips in an interleaved fashion (e.g. alternating etc.) thus increasing write bandwidth.” (INTEL-1005, ¶40.) And to increase bus bandwidth, the flash interface circuit “interleave[s] two 8-bit busses to create a 16-bit bus using one 8-bit bus from each of two flash memory devices.” (INTEL-1005, ¶41.) This “interleave two 8-bit busses to create a 16-bit bus using one 8-bit bus from each of two flash memory devices.” (*Id.*)

The flash interface circuit performs memory management functions typically performed by memory controllers.

107. Danilak further discloses that certain memory management functions are performed by the flash interface circuit in an embodiment including bad block detection, error detection and correction, garbage collection, and wear leveling in embodiments. (INTEL-1005, ¶¶24-34.) I note that these functions are typically performed by a flash memory controller. (*See* §III(A)(2).) A POSITA would have understood that in the embodiment of Figure 2 without a memory controller these

functions would have been performed in the flash interface circuit. Regardless, a POSITA would have been motivated to include these functions in the flash interface circuit for the reasons explicitly taught by Danilak and discussed briefly below.

108. In bad block detection, the flash interface circuit is “used to manage relocation of one or more bad blocks in a flash memory device transparently to the system and applications.” (INTEL-1005, ¶¶24-25.) Danilak teaches that the flash interface circuit “generate[s] and check[s] the ECC [error-correcting code]” to “enable[] the use of MLC flash memory devices in applications designed for the lower error rate of a SLC flash memory devices [sic].” (INTEL-1005, ¶28.)

109. Garbage collection “refer[s] to the process of using and then collecting, reclaiming, and reusing those areas of host memory.” (INTEL-1005, ¶31.) In flash memory, garbage collection “is the process of erasing these garbage blocks so that they may be reused.” (*Id.*) Danilak teaches that “[m]oving some or all of the garbage collection functionality into a flash interface circuit enables the use of standard flash devices that do not have integrated garbage collection capability and/or standard flash memory controllers that do not have integrated garbage collection capability.” (INTEL-1005, ¶32.)

110. Wear leveling “refers to the process to spread read and write operations evenly across a memory system in order to avoid using one or more areas of

memory heavily and thus run the risk of wearing out these areas of memory.”

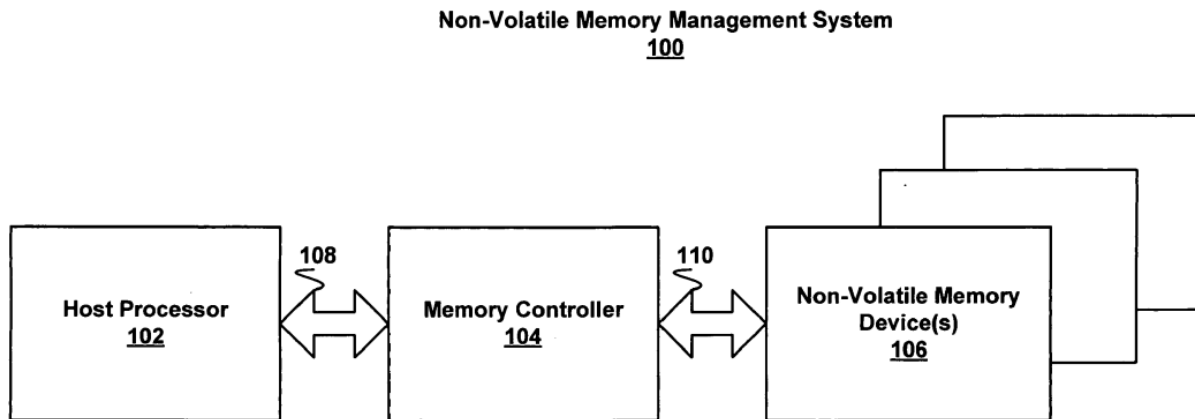
(INTEL-1005, ¶33.) Danilak teaches that “[m]oving some or all of the wear leveling functionality into a flash interface circuit enables the use of standard flash devices that do not have integrated wear leveling capability and/or standard flash memory controllers that do not have integrated wear leveling capability.” (INTEL-1005, ¶34.)

111. Based on these express teachings of Danilak, a POSITA would have also been motivated to incorporate these management functions into the flash interface circuit to improve system performance by offloading these functions from the host and to provide flexibility to select various types of memory devices to use while simplifying host software by hiding detailed characteristics of wear-leveling, error-correction, and other functions that require management.

Cornwell discloses a memory controller acting as an interface between a host and memory devices.

112. Cornwell similarly discloses a memory controller acting as an interface between a host and memory devices, like Danilak’s flash interface circuit. As shown in Figure 1 below, Cornwell discloses a memory controller “operatively coupled to the one or more non-volatile memory devices” and to a host processor. (INTEL-1006, ¶7.) Cornwell describes that a memory controller “can be any device that manages memory access.” (INTEL-1006, ¶19.) During operation of Cornwell’s memory controller, “one or more memory devices 106 receive signals

from the memory controller 104” which “enable[] the memory devices 106 to perform memory access requests (e.g., read or write operations).” (INTEL-1006, ¶24.)

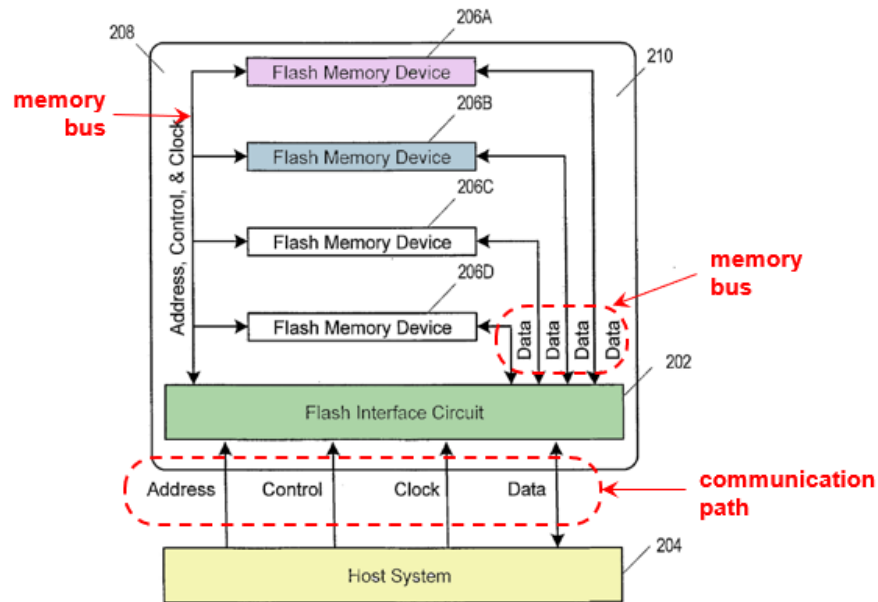


Cornwell, Figure 1

(5) “*Memory Bus*”

113. In Danilak’s system, “all signals from the host system are received by the flash interface circuit before presentation to the flash memory devices” and “all signals from the flash memory devices are received by the flash interface circuit before being presented to the host system 204.” (INTEL-1005, ¶22.) The connection carrying the signals between the flash interface circuit and the flash memory devices is a “*memory bus*.” As shown in Figure 2 below, the flash interface circuit 202 is directly connected to each flash memory device to communicate address, control, and clock signals (left) and data (right). (INTEL-1005, ¶22.) Although Figure 2 does not label the connections carrying these signals as a bus, a POSITA would understand that the connection is a “*memory bus*.” (See,

e.g., INTEL-1017, 13 (“3.236 bus. One or more conductors that are used for the transmission of signals, data, or power”), 57 (“3.1330 memory bus. A bus connecting memory to the devices which can access it, including the processor and peripheral devices”).) Danilak, in fact, expressly refers to a bus between the flash interface circuit and the memory devices. (INTEL-1005, ¶41 (describing 8-bit bus between each memory device and the flash memory circuit).) For example, Danilak describes “connections of data bus & control signals between flash memory devices 104A-104N and a flash interface circuit 102.” (INTEL-1005, ¶18.) In the section directed to “increasing bus bandwidth”, Danilak discloses its “virtual flash memory device comprises a data bus having a width equal to N times a width of a data bus of any one of the one or more flash devices, wherein N is an integer greater than one.” (INTEL-1005, ¶80.) In an embodiment, “the interface circuit is configured to interleave data on the buses of the one or more flash memory devices to implement the data bus of the virtual flash memory device.” (*Id.*) “In another embodiment, the interface circuit is configured to operate the data buses of the one or more flash memory devices in parallel.” (*Id.*)



Danilak, Annotated Figure 2

114. Cornwell similarly refers to a bus coupling the memory controller and memory device having “eight I/O lines (I/O 0-I/O 7) which are used to: (a) input a selected address, (b) output data during a read operation, or (c) input a command or data during a program operation.” (INTEL-1006, ¶34; *see also*, ¶24 (“one or more memory devices 106 receive signals from the memory controller 104 over Input/Output (I/O) bus 110”).) In addition to Danilak’s and Cornwell’s express teachings regarding a bus connecting the flash interface circuit/controller and memory devices, it was further known in the art that address, control, and data signals are all carried on a bus. (*See, e.g.*, INTEL-1008, 67.) The ’190 patent, in fact, acknowledges use of a memory bus was known—“[t]he means and method of

selecting a memory device or addressing a memory location from a set of memory devices **by using a single memory bus are known.**” (INTEL-1001, 6:25-27.)

115. Accordingly, in the combination of Danilak and Cornwell, the set of flash memory devices (*“the memory store”*) *“is directly coupled to at least one memory bus”* and *“the at least one memory bus is directly coupled to a controller”* (Danilak’s flash memory circuit) [1C.2].

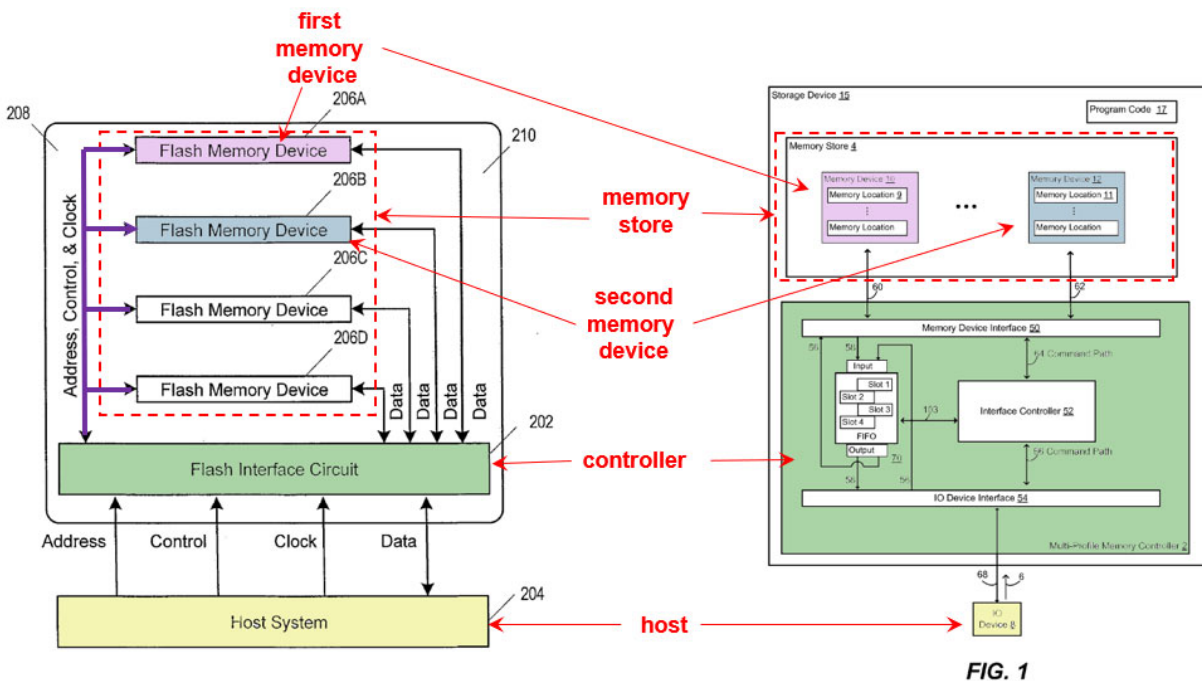
(6) “Communication Path”

116. Danilak’s system, illustrated in Figure 2 above, communicates address, control, clock and data signals between the host 204 and flash interface circuit 202. As I discussed above, in this embodiment, “all signals from the host system are received by the flash interface circuit before presentation to the flash memory devices” and “all signals from the flash memory devices are received by the flash interface circuit before being presented to the host system 204.” (INTEL-1005, ¶22.) Danilak teaches that Figure 1 includes a “data bus ... between the flash interface circuit 102 and a host system.” (INTEL-1005, ¶18.) A POSITA would have understood that such a bus also is shown in Figure 2, although not labeled as a “bus” in either diagram. Cornwell teaches that the connection between the host and memory controller is a bus. (See INTEL-1006, ¶21 (“the memory controller 104 recognizes control, address, and data signals transmitted on bus 108”).) Thus, the combination of Danilak and Cornwell discloses *“the host is directly coupled to*

the controller by a communication path” [1C.4]. This direct coupling exists throughout the operation of the memory system.

* * * *

117. The combination of Danilak and Cornwell discloses each of the structural limitations recited in independent claim 1. As shown below in the side-by-side depictions of Figure 2 of Danilak (left) and Figure 1 of the '190 patent (right), Danilak's system and the '190 patent's system have the same structure.



Danilak, Annotated Figure 2 (left); '190 Patent, Annotated Figure 1 (right)

b) “Device Profile” Limitations

118. In this section, I address the “Profile” group of “wherein” clause limitations, reproduced below:

[1A.1/Profile] *said first memory location and a second memory location respectively associated with a first device profile and a second device profile,*

[1A.6/Profile] *wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type*

[1B.1/Profile] *said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles*

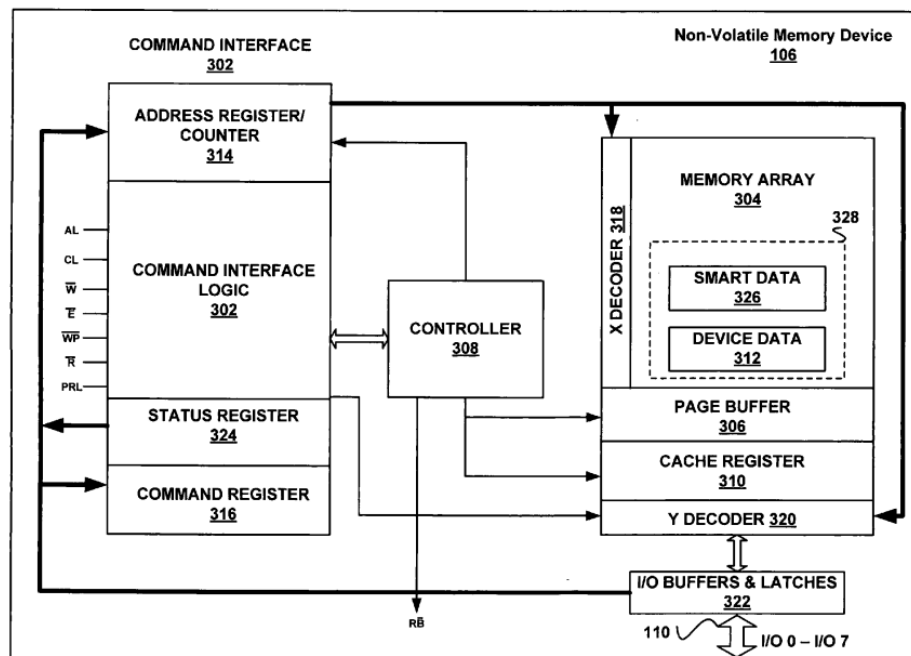
[1C.1/Profile] *wherein each attribute in the first set of attributes and second set of attributes is associated with a respective attribute qualifier that qualifies a respective memory location*

[1C.5/Profile] *wherein the first device profile and second device profile are each stored in the memory store*

(1) “Device Profile”

119. Device data stored in a non-volatile (flash) memory in the combination of Danilak and Cornwell is a “*device profile*.” The memory of the combination of Danilak and Cornwell, that I described in detail in §IV(A), stores device data in a profile in the non-volatile memory, as taught by Cornwell. Specifically, Cornwell discloses that “device data 312 is stored in a spare area 328 of the memory array 304” shown in Figure 3 of Cornwell below. The device data includes data “to identify the memory device 106 and its manufacturer.” (INTEL-1006, ¶42.) The

data 312 “also include[s] but is not limited to: device type (e.g., NAND, NOR, etc.), device density (e.g., 512 Mb, 1 Gb, 2 Gb, etc.), device operating voltage (e.g., 3.3 volts), page size (1 k, 2K, etc.), spare area size (e.g., 8, 16 bytes, etc.), sequential access time (e.g., 30, 50 nanoseconds, etc.), block size (e.g., 64 k, 128 k, etc.), bus width (e.g., ×8, ×16, etc.), bad block identification, and any other information that is associated with attributes, properties or characteristics of the memory device 106 (collectively, referred to herein as “attributes”).” (*Id.*)



Cornwell, Figure 3

120. The device data stored in each memory device in the combined system includes the configuration data used by Danilak’s flash interface circuit. Danilak describes that its flash discovery block (FB) 306 of the flash interface circuit

“store[s] configuration data discovered from attached flash memory devices.”

(INTEL-1005, ¶59.) This discovered configuration data, illustrated in Figure 9, includes overlapping device data with the stored device data including manufacturer, device, and chip identification information, page size, device density/plane size, sequential access time, and block size. A POSITA would have been motivated to store the non-overlapping configuration data of Danilak (e.g., cell type, number of simultaneously programmed pages, interleave between multiple chips, cache program) in the device data of a memory device. Cornwell expressly suggests storing this additional data in the device data, teaching that device data includes “any other information that is associated with attributes, properties or characteristics of the memory device 106.” (INTEL-1006, ¶42.) Danilak also suggests storage of its data in the device, disclosing the configuration data of Figure 9 is discovered from the attached memory devices. This teaching would indicate to a POSITA that the configuration information is stored at a memory device. As I discussed in the motivation to combine section, the ONFI specification also requires that device data be stored in a retrievable data structure in the memory device. (*See* INTEL-1007, 91-106, Table 38.) A POSITA would have understood the mandatory data specified in this table would also be included in, at least, ONFI compatible NAND devices.

900

Byte	Bits	Mnemonic	Description
0	0	—	Reserved
	1	ONFI	ONFI standard support (No, Yes)
	3:2	CHIPS	Number of stacked chips (1, 2, 4, 8)
	7:4	—	Reserved
1	8	MKR	External maker code reported by flash interface circuit to host.
2	8	DEV	External device code reported by flash interface circuit to host.
3	1:0	CIPN	Internal chip number (1, 2, 4, 8)
	3:2	CELL	Cell type (2, 4, 8, 16 levels)
	5:4	SIMP	# of simultaneously programmed pages (1, 2, 4, 8)
	6	INTRL	Interleave between multiple chips (No, Yes)
	7	CACHE	Cache program (No, Yes)
4	1:0	PAGE	Page size (1, 2, 4, 8 KB)
	2	RSIZE	Redundant area size (8, 16 bytes / 512 bytes)
	7&3	SMIN	Serial access minimum (50/30, 25 ns)
	5:4	BSIZE	Block size (64, 128, 256, 512 KB)
	6	ORG	Organization (x8, x16)
5	1:0	—	Reserved
	3:2	PLANE	Plane number
	6:4	PLSIZE	Plane size (64, 128, 256, 512, 1Gb, 2Gb, 4Gb, 8Gb)
	7	—	Reserved
6	7:0	—	Reserved

FIG. 9

Danilak, Figure 9

121. Each non-volatile memory device in Cornwell includes device data 312. This is consistent with Danilak’s disclosure teaching that “multiple instances of the configuration data may be stored in the discovery block.” (INTEL-1005, ¶68.) That is, every memory device has associated device data stored in the device which is discovered by the flash interface circuit and stored in a configuration block at the flash interface circuit. This is further consistent with ONFI specification requiring

each memory device has a data structure describing the “target’s organization, features, timings and other behavioral parameters.” (INTEL-1007, 91-106.)

122. The device data for a non-volatile memory is associated with every location in that memory device. (See Cornwell, ¶42.) As discussed in §§IV(B)(1)(a)(1)-(2), a first memory location is located in the first memory device and a second memory location is located in the second memory device in the combination. Therefore, the memory location in a first memory device is associated with the device profile for the first memory device and the memory location in a second memory device is associated with the device profile for the second memory device. The combination of Danilak and Cornwell therefore discloses “*said first memory location and a second memory location respectively associated with a first device profile and a second device profile*” [1A.1]. And because each device profile is stored in its associated non-volatile memory in the memory store, as disclosed by Cornwell, the combination of Danilak and Cornwell also discloses “*wherein the first device profile and second device profile are each stored in the memory store*” [1C.5].

(2) “Attributes”

123. Cornwell refers to the device data stored in the memory device as “**attributes**” of the device: “Chip data 312 can also include but is not limited to: device type (e.g., NAND, NOR, etc.), device density (e.g., 512 Mb, 1 Gb, 2 Gb,

etc.), device operating voltage (e.g., 3.3 volts), page size (1 k, 2K, etc.), spare area size (e.g., 8, 16 bytes, etc.), sequential access time (e.g., 30, 50 nanoseconds, etc.), block size (e.g., 64 k, 128 k, etc.), bus width (e.g., $\times 8$, $\times 16$, etc.), bad block identification, and any other information that is associated with attributes, properties or characteristics of the memory device 106 (collectively, **referred to herein as ‘attributes’**).” (INTEL-1006, ¶42.) Each device profile therefore has a “*set of attributes*” that applies to each of its memory locations. Accordingly, the combination of Danilak and Cornwell discloses “*said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location*” [1B.1].

124. The attributes used to describe the memory devices of the combined memory system of Danilak and Cornwell are consistent with the attributes disclosed in the ’190 patent, illustrated in Figure 2A below, including “page size 16, block size 18, memory protocol 20 supported by the memory device, and memory device type 22.” (INTEL-1001, 3:25-29.)

24		25		14
Page Size	<u>16</u>		(# of pages)	
Block Size	<u>18</u>		(# of blocks)	
Memory Protocol	<u>20</u>		(Protocol Type Supported)	
Mem. Dev. Type	<u>22</u>		(Mem. Type)	
Partial Page Size	<u>23</u>		(# of pages)	

'190 Patent, Figure 2A

125. The combination of Danilak and Cornwell discloses “*said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location*” [1B.1].

126. Danilak further teaches that “various attached flash memory devices may have differing attributes.” (INTEL-1005, ¶68.) The combined system of Danilak and Cornwell includes both NAND and NOR flash devices. (See §IV(A)(3).) Brewer recognized that “Flash memory possesses its own set of attributes that define the product category and each specific product implements those attributes in a slightly different way.” (INTEL-1008, 26.) The device profiles for NAND and NOR devices differ because the devices have different values and types for specified attributes (e.g., type, page size, access time, etc.). As one example, NAND and NOR devices will have different values for the “device type” attribute. NAND and NOR devices also, for example, have different densities and

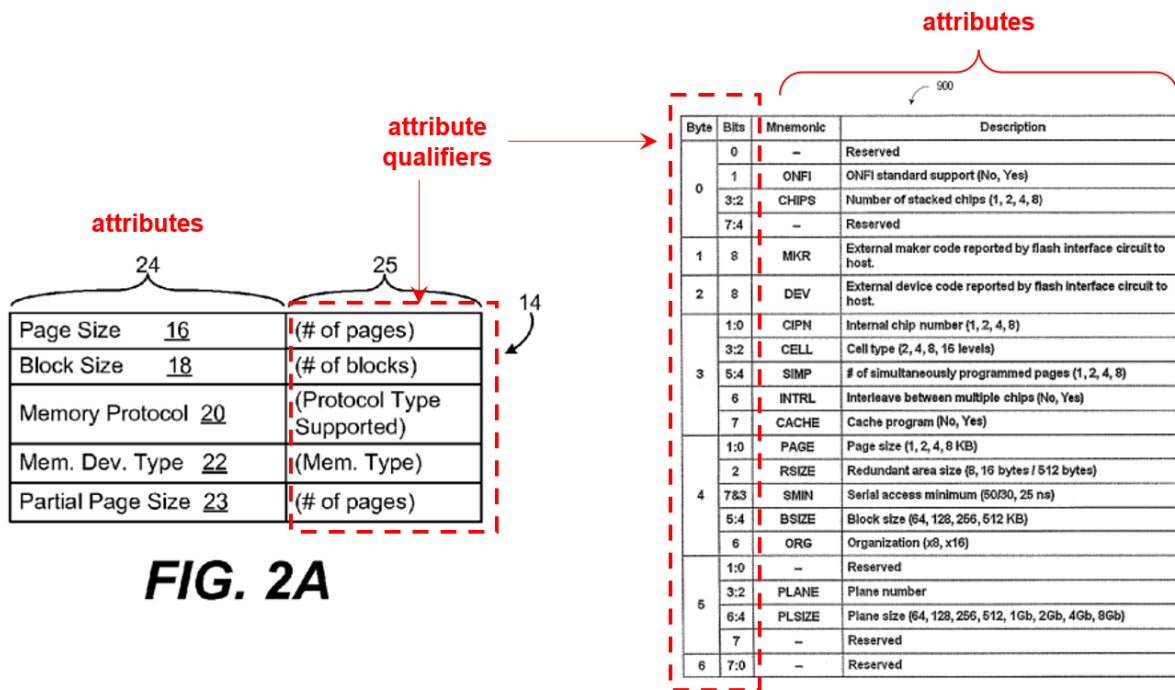
access times. (*See, e.g.*, INTEL-1011, 3 (noting differences between NAND and NOR devices); INTEL-1010, 5 (noting differences between NAND and NOR devices).) Cornwell also teaches, for example, that “[f]lash is available in a variety of block sizes.” (INTEL-1006, ¶48.) This teaching indicates that the attached memory devices have varying block sizes. And, I note Toshiba indicates that the number of levels and density also varies within NAND devices. (*See* INTEL-1011, 3 (listing density for SLC and MLC NAND devices).)

127. The combination of Danilak and Cornwell therefore discloses that “*a difference exists between said first and second device profiles*” [1B.1].

(3) “Attribute Qualifier”

128. The ’190 patent describes that “[e]ach attribute kept in a device profile is associated with data, named ‘attribute qualifier,’ that qualifies the attribute, such as an attribute value or attribute type.” (INTEL-1001, 3:16-18, Figure 2A.) As explained by the ’190 patent, the “attribute page size 16 corresponds to an **attribute value in bytes per page**, attribute block size 18 corresponds to an **attribute value in pages per block**, attribute memory protocol 20 corresponds to a memory device specification or protocol supported by the memory device associated with the memory location, **attribute memory device type 22 corresponds to a memory device type of the memory device** associated with the

memory location, attribute partial page size 23 corresponds to an attribute value in bytes per partial page.” (INTEL-1001, 3:30-39.)



’190 Patent, Figure 2A (left); Danilak, Figure 9 (right)

129. Danilak discloses that each of its attributes is associated with data qualifying the attribute for a specific device profile. The qualifying data is either value data (e.g., numerical) or type data (non-numerical). For example, byte zero of Danilak’s configuration data shown in Figure 9 above (right) includes a bit indicating whether ONFI is supported (yes or no). This non-numerical data is an example of an attribute type. As a further example, byte four “includes a page size field (PAGE)” that specifies the value for the page size of the memory device (1, 2, 4, 8 KB). (INTEL-1005, ¶64, Figure 9.) Cornwell similarly discloses page size is

indicated by a numerical value (e.g., 1k, 2k). (INTEL-1006, ¶42.) Byte four of Danilak also includes the “block size field (BSIZE) indicating the block size” (64, 128, 256, 512 KB).” (INTEL-1005, ¶64, Figure 9.) Cornwell similarly discloses block size is indicated by a numerical value (e.g., 512 MB, 1G, 2G). (INTEL-1006, ¶42.) Page size and block size are therefore example of attribute values (numerical). As a further example, the device type (NAND or NOR) is specified by a non-numeric value and is therefore an attribute type.

130. The combination of Danilak and Cornwell discloses that “*each attribute in the first set of attributes and second set of attributes is associated with a respective attribute qualifier that qualifies a respective memory location*” [1C.1].

(4) “Device Profile is Optimal for a Data Type Subject to the Memory Transaction” [1A.6].

131. Claim 1 recites the limitation “*said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type*” [1A.6]. The specification mentions the word “optimal” in a single sentence:

In another example, by using a multi-profile memory controller 2, which supports memory locations that have different device profiles, a host or IO device, such as IO device 8, can use a logical address that maps to at least one target memory location that is associated with a device profile that is **optimal** for the type of data subject to a memory transaction request initiated by IO device 8, and another

target memory location that is associated with another device profile that is **optimal** for the type of data subject to another memory transaction request initiated by IO device 8.

(INTEL-1001, 9:7-16.) The patent does not disclose or discuss what makes a target memory location “optimal” for a specific data type. I also note that sequential and random are typically associated with the type of memory access and not the data in the request.

132. I understand that PO has pointed to three examples included in two paragraphs of column 9 as providing example disclosure of this claim limitation. The first example implies that a device profile with a large block size and NAND flash memory type is desirable for a sequential access.

IO device 8 may use a memory write transaction request that uses a logical address that maps to at least one target memory location that is associated with a device profile that includes a block size attribute value that is the largest available from the other device profiles associated with a set of memory locations available for addressing by multi-profile memory controller 2. This transaction command detail may be desired if the data subject to the memory transaction request is of a sequential type, and the memory type attribute value also associated with the device profile is a NAND flash memory device.

(INTEL-1001, 6:16-26.) The second example implies that a device profile with a small block size is desirable for a random access.

In yet another example, IO device 8 may use a memory transaction request that uses a logical address that maps to at least one target memory location which is associated with a device profile that includes a block size attribute value that is the smallest available from the other device profiles associated with a set of memory locations available for addressing by multi-profile memory controller 2. This transaction command detail may be desired if the data subject to the memory transaction request is random nature.

(INTEL-1001, 9:27-35.) In the final example, the '190 patent implies that use of a device profile with a partial page size equal to the data size is desirable for a random access.

In yet another further example, IO device 8 may use a memory transaction request that uses a logical address that maps to at least one target memory location that is associated with a device profile that includes a partial page size attribute value. This transaction command detail may be desired if the data subject to the memory transaction request is random in nature, and if the data size of the data subject to the write transaction is at most equal to the partial page size attribute value.

The combination of Danilak and Cornwell discloses or at least renders this limitation obvious based on how PO appears to be interpreting “optimal” in the co-pending district court litigation.

133. In the combination of Danilak and Cornwell, the memory includes both NAND and NOR flash devices as I discussed in §IV(A)(3). The device data includes an attribute which specifies the memory device type (e.g., NAND or NOR), as taught by Cornwell. (See INTEL-1006, ¶¶42.) This data is in turn discovered and stored in a configuration file for the device in the flash interface circuit, as taught by Danilak. (See INTEL-1005, ¶¶56-60, 57-68, Figure 5, Figure 9.) Thus, in the combined system, the flash interface circuit has a configuration file associated with each memory device which specifies the data describing the device.

134. It was well-known prior to the '190 patent that NOR flash is best suited for code storage as I discussed in §III(A)(1). As explained by Brewer, “[c]ode execution generally requires fast random-access reads because code jumps from one memory location to another when branching. For this reason, the high speed random-access capability of NOR Flash architecture is ideal for code storage applications.” (INTEL-1008, 23.) In fact, the recognized strengths of NOR-type Flash memories “are fast **random-access** reads and reliable storage.” (*Id.*; *see also*, INTEL-1008, 38-39; INTEL-1010, 4; INTEL-1011, 3.)

135. It was also well-known prior to the '190 patent that NAND flash “has poor random-access read performance, but **sequential read** performance is good” as I discussed in §III(A)(1)(c). (INTEL-1008, 23; *see also*, INTEL-1010, 3, 5;

INTEL-1011, 3.) And NAND flash is “strongly suited” to data storage applications “such as audio-storage, video storage, or data logging where fast data writes are required and the **data will be access sequentially.**” INTEL-1008, 23.) And DiPaolo in an article published in 2005 (4 years before the earliest possible priority date of the ’190 patent) noted that “NAND can’t perform random access” but “[o]ne major benefit of NAND in a handset application is the write speed, and the time required to erase a previous block and write a new one.” (INTEL-1009, 2.)

136. In 2005, DiPaolo described a handset from NEC which included both NOR flash memory and NAND flash memory. Specifically, “[r]ead speed and the XIP architecture are reasons that NOR is used to store and run code” and “[d]ensity and faster program and erase speeds (write speeds) are the main reasons that NAND devices are used to store images and content.” (INTEL-1009, 3.) The combination of NAND and NOR, as described by the EE Times, allows a vendor “to get the best of both worlds.” (INTEL-1009, 3.)

137. Based on the well-known known strengths of NOR and NAND, a POSITA would have been further motivated to incorporate a memory management function that (1) selects NOR flash when a transaction is received which requires writing or reading code because NOR flash is “*optimal*” for code execution requiring fast random-access reads and (2) selects NAND flash when a transaction is received for writing and reading data (e.g., audio or image storage) which

requires faster sequential write and read accesses. A POSITA would have understood that the request from the host included an indication of the data type associated with the request (e.g., code or data) or alternatively would have been motivated to include such an indication to achieve optimal and desired performance of the dual NAND/NOR memory system. Several microprocessor buses existing prior to the '190 patent allowed the microprocessor to identify whether a bus transaction was for program (code) or data. (*See, e.g.*, INTEL-1019, 4-5 (identifying function code assignments of program or data); INTEL-1020, 3-14 (indicating “data or code”).) A POSITA would have understood that the incorporation of the selection of the optimal memory device (e.g., NOR or NAND) when handling a transaction was necessary in dual memory type systems to permit the vendor (and user) “to get the best of both worlds” offered by the inclusion of both types of memory. A POSITA would have therefore been motivated to include this selection of memory device type to optimize application and device performance. Indeed, mobile phones existing prior to the '190 patent used both NOR and NAND flash to achieve the optimizations offered by both in a single system. (*See, e.g.*, INTEL-1009, 3.)

138. Accordingly, the combined memory of Danilak and Cornwell includes a “*device profile*” [NOR device type] that “*is optimal for a data type subject to the memory transaction* [e.g., reading or writing code]” where the “*data type*” is “*a*

random data type” and a “*device profile*” [NAND device type] that “*is optimal for a data type subject to the memory transaction [e.g., writing data]*” where the “*data type*” is a “*sequential data type.*”

139. It was further known that the “ideal block size depends on the nature of the application.” (INTEL-1016, 8.) For example, large blocks and accordingly large page sizes³ are “preferable for applications that do long sequential reads.” (*Id.*) The ’190 patent acknowledges this known fact, admitting in the background section “[f]lash memory devices that have a relatively large block size generally provide better sequential memory access performance than flash memory devices that have a relatively small block size.” (INTEL-1001, 1:27-31.) Consistent with this knowledge, Cornwell teaches “[f]lash is available in a variety of block sizes” and both Cornwell and Danilak provide “block size” and “page size” as configurable attributes of a memory device. (INTEL-1006, ¶48, *see also*, ¶42; INTEL-1005, ¶59, Figure 9.) For example, NAND flash was available with small block sizes and large block sizes. Accordingly, based on the teachings of Cornwell and knowledge of a POSITA, a POSITA would have been motivated to select a

³ As I have discussed previously, page size is the relevant parameter for reading data, such as code, and block size is the relevant parameter for erasing data. (*See, e.g.*, INTEL-1007, 2-3.)

flash memory with a large block size/page size to optimize sequential access (e.g., select a NAND flash memory with large block size over a NAND flash memory with small block).

140. Accordingly, the combined memory of Danilak and Cornwell includes a “*device profile*” that “*is optimal [e.g., has large block size/page size] for a data type subject to the memory transaction*” where the “*data type*” is a “*sequential data type*.”

c) Preamble [1P]

141. The combination of Danilak and Cornwell discloses “*performing memory transactions⁴ ...*” [1P]. As I discussed in §IV(B)(1)(a)(2), the combination of Danilak and Cornwell includes “*a set of memory locations that includes a first memory location and a second memory location*.” In the combination of Danilak and Cornwell, the flash interface circuit (“*controller*”) receives a memory

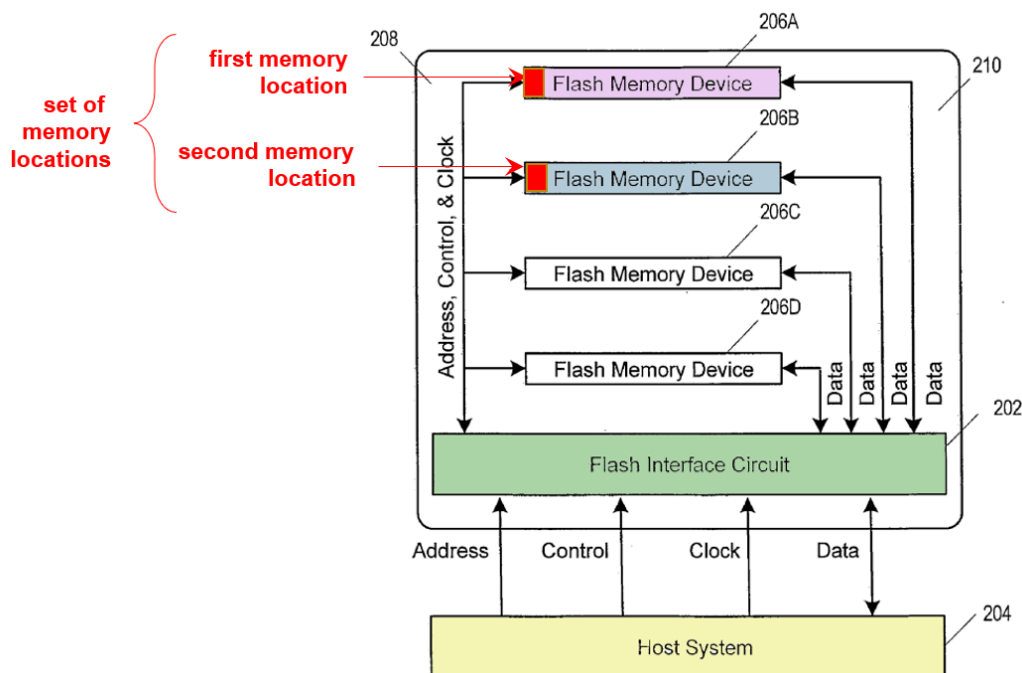
⁴ The ’190 patent does not define the term “transaction.” A “bus transaction” is a “sequence of bus operations that includes a request and may include a response, either of which may carry data. A transaction is initiated by a single request and may take many individual bus operations.” (INTEL-1018, 582.) Therefore, a transaction is equivalent to a request and associated data transfer, if any.

transaction request (e.g., read or write request) from the host formatted according to a defined protocol such as IDE/ATA. (INTEL-1005, ¶78; *see also, e.g.*, INTEL-1006, ¶¶25-27.) The memory transaction request from the host specifies the transaction type (e.g., read or write) and an address on which the transaction should be performed. (*See, e.g.*, INTEL-1007, 2-3, 111-112, 117-118.) For write transactions, the request will also include the data to be written to the specified address. (*See, e.g.*, INTEL-1007, 117-118.) Translation units 508/509 in the flash interface circuit perform the requested transaction by “translat[ing] the request into one or more physical flash memory device accesses.” (INTEL-1005, ¶78.)

142. The types of physical flash memory device accesses performed by Danilak include read, program (write), and erase operations. (*See, e.g.*, INTEL-1005, ¶¶33-40.) Cornwell similarly teaches that its memory controller “includes circuitry for enabling **read, program and erase operations** to one or more memory devices 106.” (INTEL-1006, ¶31.) I provide further discussion of the support of read and write transactions in the combination of Danilak and Cornwell in §IV(B)(1)(e).

143. Each operation required to perform the memory transaction requested by a host is performed on a “*memory location*” in a physical memory device. As I discussed in §IV(B)(1)(a), the combination of Danilak and Cornwell discloses a memory store having multiple memory devices each having a set of addressable

memory locations, as illustrated in Figure 2 below. In Danilak, while performing the memory transaction requested by the host, translation units 508/509 in the flash interface circuit translate the received address “to an address to be transmitted to the attached flash devices.” (INTEL-1005, ¶72; *see also*, ¶68 (“As mentioned above, the address translation unit 508 may translate addresses between the host and the flash memory devices”).) The address transmitted to the flash memory device indicates the memory location on which the requested transaction is to be performed.



Danilak, Annotated Figure 2

144. Cornwell confirms the translated address indicates the memory location on which the requested transaction is to be performed. When discussing operation of a non-volatile memory device, Cornwell describes that the memory array 304

“is accessed using x-decoder 318 and y-decoder 320.” (INTEL-1006, ¶35.) “X-decoder 318 decodes input addresses in address register/counter 314 to determine a memory line to be accessed for the read or write operation.” (*Id.*) Address register/counter “receive[s] addresses from the memory controller 104 via I/O bus 110.” (INTEL-1006, ¶37.) “Y-decoder 320 decodes signals from the command interface logic 302 for reading or writing data into the memory line determined by x-decoder 318.” (INTEL-1006, ¶35.)

145. Additionally, it would have been well-known to a POSITA that a memory system performed memory transactions, including read and write (program) operations, on a memory location. In fact, I note that the ’190 patent admits this well-known fact stating: “A flash memory device permits memory operations, such as a write or read operation, to be performed on these flash blocks according to a protocol supported by the flash memory device.” (INTEL-1001, 4:47-50.) The ’190 patent also teaches that its memory devices “comply with the Open NAND Flash Interface Specification, commonly referred to as ONFI Specification.” (INTEL-1001, 4:50-53.) The ONFI specification published prior to the ’190 patent describes read, program (write), and erase operations. (*See, e.g.*, INTEL-1007, 3 (“**page**: The smallest addressable unit for **read and program** operations”), 2 (“The address is comprised of a row address and a column address.

The row address identifies the page, block, and LUN to be accessed. The column address identifies the byte or word within a page to access”).)

146. Accordingly, the combination of Danilak and Cornwell discloses “performing memory transactions on a set of memory locations that includes a first memory location and a second memory location” [1P].

d) Method Steps

147. In this section, I address the method step limitations, reproduced below:

[1A/Action] *performing a memory transaction comprising addressing a first memory location in a first memory device in a memory store,*

[1B/Action] *said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;*

[1C/Action] *after identifying the command details, obtaining the first set of attributes;*

[1D/Action] *using attributes from said first and second device profiles; and*

[1E/Action] *selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.*

(1) “Addressing a First Memory Location ...”

148. The combination of Danilak and Cornwell discloses that “*performing a memory transaction*” includes “*addressing a first memory location in a first memory device in a memory store*” [1A]. In the combination of Danilak and Cornwell, the flash interface circuit (“*controller*”) receives a memory transaction request from the host. (See INTEL-1006, ¶¶25-27.) The memory transaction request from the host specifies the transaction type (e.g., by a read or write command) along with an address associated with the request. (See INTEL-1006, ¶¶25-27.) Translation units then translate the host request into the memory operations (data, address, and control signals) required to satisfy the memory transaction request. (See INTEL-1005, ¶77; INTEL-1006, ¶29.)

149. As I discussed in §IV(B)(1)(c), “*performing a memory transaction*” on a memory location requires addressing that memory location. In Danilak, during a memory transaction, translation unit 508 translates the address received from the host “to an address to be transmitted to the attached flash devices.” (INTEL-1005, ¶72; *see also*, ¶68 (“As mentioned above, the address translation unit 508 may translate addresses between the host and the flash memory devices”).) The address transmitted to the flash memory device indicates the memory location on which the requested transaction is to be performed.

150. Cornwell confirms the translated address sent to the memory device indicates the memory location on which the requested transaction is to be performed. When discussing operation of a non-volatile memory device, Cornwell describes that the memory array 304 “is accessed using x-decoder 318” which “decodes input addresses” received from the memory controller “to determine a memory line to be accessed for the read or write operation.” (INTEL-1006, ¶35, *see also*, ¶37.)

151. Additionally, it would have been well-known to a POSITA that performing a memory transaction on a memory location requires “*addressing*” that memory location. For example, the ONFI specification states that an “address is comprised of a row address and a column address” with the “row address identif[ying] the page, block, and LUN to be accessed” and the “column address identifies the byte or word within a page to access.” (INTEL-1007, 2.)

152. As discussed in §IV(B)(1)(a), the combination of Danilak and Cornwell discloses “*a first memory location in a first memory device in a memory store.*” A memory location is an addressable portion of memory on which operations such as read or write operations are performed. Accordingly, the combination of Danilak and Cornwell discloses “*addressing a first memory location in a first memory device in a memory store*” [1A] during the performance of a memory transaction.

(2) “Identifying Command Details ...” [1B].

153. The combination of Danilak and Cornwell discloses that “*performing the memory transaction*” includes “*identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device*” [1B].

154. The ’190 patent describes “command details” as the “details necessary to cause the interpreted memory transaction to be performed by multi-profile memory controller 2 through its components.” (INTEL-1001, 7:60-63.) The command details required to perform a memory operation on a memory device are specified by the protocol of the memory device. For example, in an ONFI device, to perform a read operation, the “read” command must be specified along with the address of the data to be read. (*See, e.g.*, INTEL-1007, 111-112.) And to perform a write (program) operation, the “write” command must be specified along with an address and data to write to that memory address. (*See, e.g.*, INTEL-1007, 117-118.) Cornwell similar confirms that a read or write operation for a memory device requires a command specifying the type of operation and the address. (*See, e.g.*, INTEL-1006, ¶¶34-38.) Therefore, at a minimum, the command details that must be identified by the controller when performing a memory transaction are the command type and address.

155. Translation units 508/509 in Danilak’s flash interface circuit identify details needed to cause a memory transaction requested by a host to be performed by one or more memory devices. In Danilak, during a memory transaction, translation units 508/509 in the flash interface circuit “translate host flash memory access and configuration requests into requests to one or more flash memory devices.” (INTEL-1005, ¶56.) To perform this translation, these translation units identify the address provided by the host. The address translation unit 508 then translates this address into an internal address. (INTEL-1005, ¶71.) The “internal address format for the flash interface circuit comprises a valid bit indicating whether or not a request is being transmitted; **a device field identifying the addressed flash memory device**; a plane field identifying a plane within the device, a block field identifying the block number within the plane; a page number identifying a page within the block; a redundant bit indicating whether or not the redundant area is being addressed, and column address field containing the column address.” (INTEL-1005, ¶70.) The internal address is subsequently translated by the flash interface circuit “to an address to be transmitted to the attached flash devices.” (INTEL-1005, ¶72.) Therefore, the received address and its internal address are both command details necessary to cause flash interface circuit to perform the memory transaction.

156. Accordingly, the combination of Danilak and Cornwell discloses that “performing the memory transaction” includes “*identifying command details for causing the memory transaction to be performed.*” And because the internal address includes a “device field identifying the addressed flash memory device,” the combination also discloses “*said command details comprising the first memory device*” [1B].

(3) “Obtaining the First Set of Attributes” [1C] and “Using Attributes From Said First and Second Device Profiles” [1D].

157. The combination of Danilak and Cornwell discloses “*after identifying the command details, obtaining the first set of attributes*” [1C] and “*using attributes from said first and second device profiles*” [1D].

158. In Danilak, the flash interface circuit exposes the attached physical flash memory devices to the host as one “virtual flash device.” (See INTEL-1005, ¶17; §IV(A)(1).) To emulate the virtual flash memory device, the flash interface circuit must first determine the virtual configuration for the virtual memory. To do this, the flash interface circuit “perform[s] device discovery/configuration for the physical flash memory devices coupled to the flash interface circuit.” (INTEL-1005, ¶77.) From the discovered data, the flash interface circuit “determine[s] the virtual configuration based on the discovered flash memory devices.” (*Id.*) The virtual configuration is reported to the host, “thus exposing the virtual configuration to the host rather than the physical configuration.” (*Id.*) And as I

discussed in §IV(B)(1)(b)(1), the discovered device data is stored locally at the flash interface circuit in configuration files associated with the attached memory devices.

159. Therefore, when performing a memory transaction requested by the host, translation units 508/509 in the flash interface circuit must modify the request “provided from the host system based on differences between the virtual configuration presented by the interface circuit 500 to the host system and the physical configuration of the flash memory devices.” (INTEL-1005, ¶56.) The translation units obtain and use attributes of the attached flash memory devices to determine and account for the differences between the virtual configuration and physical configurations of the memory devices.

160. Danilak describes an example of a translation required based on differences between the virtual and physical configuration of the memory devices. In this example, “the configured page size” for the virtual memory reported to the host “is larger than the discovered page size[s]” for the physical memory devices. (See INTEL-1005, ¶73.) This difference results in a memory transaction request from a host specifying a data size consistent with the virtual page size but larger than can be handled by a single memory device. (See, *Id.*) To account for this difference, translation units 508/509 in the flash interface circuit are configured to translate a host write request with a large data size into multiple write operations of

smaller sizes. As explained by Danilak, in this circumstance, the memory transaction requested by the host is “performed to multiple flash memory devices to form a page of the configured size.” (*Id.*) That is, the translation units identify the page sizes for available flash memory devices and selects two or more memory devices having page sizes which sum to the data size of the memory write transaction.

161. To identify the page sizes for the available flash memory devices during this translation, the translation units must obtain attributes for the physical memory devices. These attributes, discovered from the memory devices, are stored locally in the configuration files at the flash interface circuit as I mentioned above and discussed in further detail in §§IV(B)(1)(b)(1). And as I discussed in §IV(B)(1)(d)(2), as part of the translation to an internal address, the translation units identify a “device field identifying the addressed flash memory device.” (*See* INTEL-1005, ¶70.) Therefore, a POSITA would have understood the flash interface circuit obtains the attributes associated with the identified memory device. Thus, because the received host address is associated with the “*first memory device*” by the translated internal address (“*command detail*”), the attributes obtained by the translation unit would include the “*first set of attributes*” associated with that device. Danilak teaches or suggests “*obtaining the first set of attributes*” [1C].

162. However, because the page sizes associated with all the physical memory devices including the “*first memory device*” are less than the size needed to complete the memory write transaction, the translation units must obtain additional device profiles “to form a page of the configured size.” (INTEL-1005, ¶73.) For example, if the memory write request specified a data size of 4K and the first memory device has a page size of 2k, the translation units must identify one or more additional memory devices with page sizes summing to 2k. To do this, the translation units obtain the attributes of additional memory devices. These attributes are then used together with the attributes of the identified “*first memory device*” to select the set of memory devices having page sizes summing to the needed size to perform the requested memory transaction.

163. Accordingly, Danilak teaches or suggests the translation units in the flash interface circuit “*us[e] attributes [e.g, page size] from said first and second device profiles*” when processing the memory transaction.

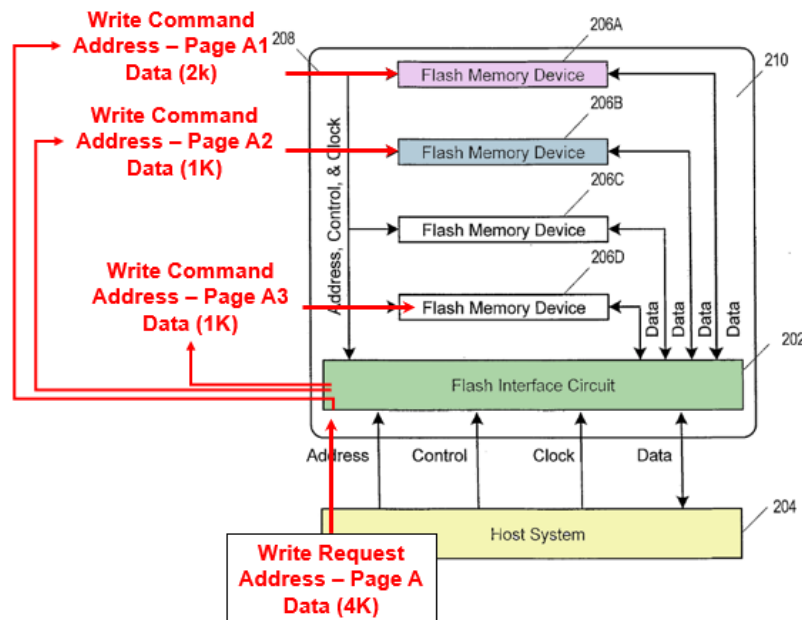
(4) “*Selecting a Transfer Size for the Memory Transaction*” [1E].

164. The combination of Danilak and Cornwell discloses “*selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes*” [1E]. A host memory transaction has a “data size” which is the amount of data to be read or written by the transaction. (For example, the data size for a memory transaction

initiated by a Pentium Pro processor specifies a data size using a combination of LEN# and BE# signals. (See INTEL-1020, 3-13—3-15.)) This host memory transaction is translated by a memory controller into one or more operations sent to the physical memory devices, as I have discussed previously. The amount of data read or written by an operation from/to an individual memory device is the “transfer size” for that operation. It was well-known that the transfer size of a memory operation to a memory device is dictated by the size of the largest addressable unit for that device (e.g., page size for flash memory). It was also well-known that the transfer size of a memory operation is a function of the data size of the transaction requested (e.g., transfer size is either equal to or a fraction of the data size). Danilak teaches these well-known relationships.

165. As Danilak explains, in the circumstance where the virtual page size (and hence the data size) is larger than the discovered page size, “the memory request may be performed to multiple flash memory devices to form a page of the configured size.” (INTEL-1005, ¶73.) That is, in Danilak’s size translation example, the data size for the requested memory transaction is larger than the transfer size any individual memory device can accept in a single operation, requiring multiple memory operations to be performed. (*See Id.*) The translation units in Danilak’s flash interface circuit therefore selects a set of available memory devices to perform the requested transaction based on their page sizes (transfer

size) and the data size of the transaction, as I illustrate in the following figure which adds annotations to Danilak's Figure 2. Therefore, the transfer size of each individual memory operation needed to perform the requested memory transaction is a function of the page size of the memory device (e.g., "*first set of attributes*") and the "*data size of the memory transaction*" (e.g., a fraction of the data size). In the example I illustrate below, memory device 206A has a page size of 2K, memory device 206B has a page size of 1K, and memory device 206D has a page size of 1K. Therefore, the memory operation to device 206A has a transfer size which is one-half the data size of the requested memory transaction received from the host. The memory operations to both devices 206B and D each have a transfer size which is one-quarter the data size of the requested memory transaction received from the host.



166. The size translation function performed by Danilak’s flash interface circuit is consistent with the example of transfer size selection described in the ’190 patent. In the ’190 patent’s example, “if the data size is greater than the page size attribute value in device profile 26, multi-profile memory controller 2 uses a transfer size equal to at least the page size attribute value for each memory operation transfer for the memory transaction requested until the remaining data not yet transferred from a requesting IO device is less than the page size attribute.” (INTEL-1001, 8:50-57.) This is exactly what is described by Danilak in its size translation example.

167. The combination of Danilak and Cornwell therefore discloses “*selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes*” [1E].

e) “Read-Write” Limitations

168. In this section, I address the “Read-Write” group of “*wherein*” clause limitations, reproduced below:

[1A.3/Read-Write] *wherein a memory read transaction or a memory write transaction is performed on at least one of the first memory location or second memory location,*

[1A.4/Read-Write] *wherein data is transferred to the first memory location or second memory location from a host for a memory write transaction,*

[1A.5/Read-Write] *wherein data is transferred from the first memory location or second memory location to the host for a memory read transaction;*

...

[1C.3/Read-Write] *wherein the controller performs the memory read transaction and memory write transaction on the memory store;*

[1C.4/Structural] *wherein the host is directly coupled to the controller by a communication path* **[Read-Write]** *as the controller performs the memory read transaction or memory write transaction;*

169. The “read-write” group of wherein clauses recite extremely well-known aspects of memory. In my opinion, the basic concept of reading and writing to a memory are inherent in flash memory. Indeed, I note the ’190 patent recognizes the well-known and incontrovertible fact that “[a] flash memory device permits memory operations, such as a write or read operation, to be performed on these flash blocks according to a protocol supported by the flash memory device.” (INTEL-1001, 4:47-50.)

(1) Controller Performs the Memory Read Transaction or Memory Write Transaction on a Memory Location.

170. As I discussed in §IV(B)(1)(c) for the preamble, the combination of Danilak and Cornwell discloses “*performing memory transactions on a set of*

memory locations.” Both Danilak and Cornwell disclose the memory transactions performed by the flash interface circuit (controller) include read and write transactions. For example, as I discussed above in §IV(B)(1)(c), Danilak’s flash interface circuit performs the functions of a flash driver. Specifically, it “makes the flash memory appear to the host system as a **read/write** memory array” and “supports basic file system functions (e.g., **read, write**, file open, file close etc.).” (INTEL-1005, ¶29.) Danilak also repeatedly references read and write operations in its discussion of the memory management functions performed by the flash interface circuit:

- “Garbage collection may be performed, for example, when the system is idle or after a **read/write operation.**” (INTEL-1005, ¶31.)
- “Typically, flash memory has a low bandwidth (e.g. for **read, erase and write operations**, etc.) and high latency (e.g. for **read and write operations**) that are limits to system performance.” (INTEL-1005, ¶35.)
- “The term leveling, and in particular the term wear leveling, refers to the process to spread **read and write operations** evenly across a memory system in order to avoid using one or more areas of memory heavily and thus run the risk of wearing out these areas of memory.” (INTEL-1005, ¶33.)

171. Cornwell similarly discloses that the memory controller performs both read and write operations. Specifically, Cornwell explains that “one or more memory devices 106 receive signals from the memory controller 104 over Input/Output (I/O) bus 110, which enables the memory devices 106 to perform memory access requests (e.g., **read or write operations**).” (INTEL-1006, ¶24.)

172. Both read and write transactions are performed on a “*memory location*” by the flash interface circuit (“*controller*”) as discussed in §§IV(B)(1)(c) (preamble) and §§IV(B)(1)(d)(1) (addressing).

173. Thus, the combination of Danilak and Cornwell discloses that “*a memory read transaction or a memory write transaction is performed on at least one of the first memory location or second memory location*” [1A.3] and “*the controller performs the memory read transaction and memory write transaction on the memory store*” [1C.3].

174. In Danilak, “all signals from the host system are received by the flash interface circuit before presentation to the flash memory devices” and “all signals from the flash memory devices are received by the flash interface circuit before being presented to the host system 204.” (INTEL-1005, ¶22.) As shown in Figure 2 above and discussed in §IV(B)(1)(a)(6), the signals between the host and the flash interface controller are carried over a “*communication path*.” When the host requests a memory access, the flash interface circuit “return[s] an appropriate

response to the request to the host” after the transaction is performed. (INTEL-1005, ¶78.) Thus, the combination of Danilak and Cornwell discloses “*the host is directly coupled to the controller by a communication path as the controller performs the memory read transaction or memory write transaction*” [1C.4].

(2) Write Data Flow Path

175. It was well-known long before the ’190 patent that “*data is transferred to the first memory location or second memory location from a host for a memory write transaction*” [1A.4]. Both Danilak and Cornwell disclose this well-known write data flow.

176. Danilak focuses on improvements to basic write operations and therefore does not describe a basic write operation. However, Danilak does disclose the basic flow of data for a write operation when discussing its improvements. Danilak discloses an example of improving write bandwidth. In this example, the “flash interface circuit may hold (e.g. buffer etc.) write requests” received from the host “in internal SRAM and **write them into the multiple flash memory chips** in an interleaved fashion (e.g. alternating etc.) thus increasing write bandwidth.” (INTEL-1005, ¶40.) Danilak further discloses in Figure 2 that data is transferred between the host and flash interface circuit. (See INTEL-1005, Figure 2.) A POSITA would have understood from this discussion that a normal write operation entails the flash interface circuit receiving a write request with data to be

written from the host and writing the data into the physical flash memory. Danilak therefore discloses “*data is transferred to the first memory location or second memory location from a host for a memory write transaction*” [1A.4] during normal operation. I also note that the claim limitation does not preclude buffering a write request before performing the write on the physical memory and therefore Danilak’s write bandwidth optimization embodiment also discloses this claim limitation.

177. Cornwell describes an exemplary page program operation to write a data sequence at a specified address. In non-volatile memory, the purpose of the program operation is to write a data sequence at a specified address. Cornwell explains that an application running on the host requests access to one or more memory devices, for example, to save data such as a song to memory. (INTEL-1006, ¶25.) The operating system of the host then formats and sends the request to a memory controller using a specified protocol such as ATA. (*Id.*) The IDE/ATA interface supports commands including a write command. (See INTEL-1006, ¶¶26-27.) The memory controller then sends a page program command to the non-volatile memory. The non-volatile memory stores the page program command in a command register. (INTEL-1006, ¶39.) The associated data stored in I/O buffers and latches is loaded into the page buffer. (*Id.*) The memory “programs the page into the memory array 304 at the address stored in address register 314 using x-

decoder logic 318 and y-decoder logic 320 for row and column address decoding, respectively.” (*Id.*) Cornwell therefore also discloses “*data is transferred to the first memory location or second memory location from a host for a memory write transaction*” [1A.4].

178. The ONFI specification confirms this well-known write data flow, repeatedly referring to a write as a data transfer from the host to the memory device. For example, ONFI describes the use of a strobe signal (DQS) which is “center aligned to the valid data window for data **transfers from the host to the device (writes)**.” As a further example, ONFI discloses that “[d]ata input cycle timing describes timing of **data transfers from the host to the device (i.e., data writes)**.” (INTEL-1007, 74.)

179. Danilak and Cornwell therefore both disclose “*data is transferred to the first memory location or second memory location from a host for a memory write transaction*” [1A.4].

(3) Read Data Flow Path

180. It was well-known before the ’190 patent that “*data is transferred from the first memory location or second memory location to the host for a memory read transaction*” [1A.5]. Both Danilak and Cornwell disclose this well-known read data flow.

181. Danilak focuses on improvements to basic read operations and therefore does not describe a basic read operation. However, Danilak does disclose the basic flow of data for a read operation when discussing its improvements. For example, Danilak discloses prefetching data by “read[ing] data from the flash memory ahead of a request by the system.” (INTEL-1005, ¶38.) The prefetched data is stored in the flash interface circuit. (INTEL-1005, ¶39.) When a “read request from the system is received, and the read request is for the prefetched data, the prefetched data may be returned by the flash interface circuit to the system without accessing the flash memory devices.” (*Id.*) From this discussion, a POSITA would have understood the well-known fact that without “prefetching” data is read from the flash memory when a request for the host system is received and returned to the host. Danilak therefore teaches “*data is transferred from the first memory location or second memory location to the host for a memory read transaction*” [1A.5] during normal operation. I also note that the claim language does not preclude storage at the “*controller*” for a period of time and therefore Danilak’s prefetch embodiment also discloses this claim limitation.

182. Cornwell describes read operations. Cornwell explains that an application running on the host requests access to one or more memory devices. (INTEL-1006, ¶25.) The operating system of the host send the request to the memory controller using the specified protocol (e.g., ATA). (*Id.*) The IDE/ATA

interface supports commands including the Read Sectors command. (INTEL-1006, ¶¶26-27.) In Cornwell’s page read operation, data stored in memory array 304 of the non-volatile memory is “transferred to the page buffer 306” and “is read out from the page buffer 306 sequentially (from selected column address to last column address) and stored in I/O buffers & latches 322, where the data can be read out by the memory controller 104.” (INTEL-1006, ¶40.) Cornwell therefore also discloses that “*data is transferred from the first memory location or second memory location to the host for a memory read transaction*” [1A.5].

183. The ONFI specification confirms this well-known read data flow. ONFI defines a “read request” as a “request from the host that results in a data transfer from the device to the host.” (INTEL-1007, 3.) As a further example, ONFI describes the use of a strobe signal (DQS) which is “aligned to the transition of the DQ bus for **data transfer from the device to the host (reads)**.” (INTEL-1007, 43.)

184. Danilak and Cornwell therefore both disclose “*data is transferred from the first memory location or second memory location to the host for a memory read transaction*” [1A.5].

f) Summary

185. In summary, following table presents the preamble and each limitation in order, the section or sections addressing the limitation, and the evidence cites.

Claim 1	
[1P] A method of performing memory transactions on a set of memory locations that includes a first memory location and a second memory location, the method comprising	§IV(B)(1)(a)(2) (“memory location”): INTEL-1005, ¶¶22, 24-25, 69, claim 7, Fig. 2 INTEL-1006, ¶¶3-4 INTEL-1008, 66-67, Fig. 3.2 INTEL-1007, 2 §IV(B)(1)(c) (“Preamble [1p]”): INTEL-1005, ¶¶33-40, 68, 72, 78, Fig. 2 INTEL-1006, ¶¶25-27, 31, 35, 37 INTEL-1018, 582 INTEL-1007, 2-3, 111-112, 117-118
[1A/Action/Structural] performing a memory transaction comprising addressing a first memory location in a first memory device in a memory store,	§IV(B)(1)(a)(1) (“memory device”): INTEL-1005, ¶¶17-18, 22, 22 §IV(B)(1)(a)(2) (“memory location”): INTEL-1005, ¶¶22, 24-25, 69, cl. 7, Fig. 2 INTEL-1006, ¶¶3-4 INTEL-1008, 66-67, Fig. 3.2 INTEL-1007, 2 §IV(B)(1)(a)(3) (“memory store”): INTEL-1005, ¶¶17, 51, 54, 80, Fig. 4 INTEL-1017, 89 §IV(B)(1)(d)(1) (“addressing a first memory location...”): INTEL-1005, ¶¶68, 72, 77 INTEL-1006, ¶¶25-27, 29, 35, 37 INTEL-1007, 2
[1A.1/Profile] said first memory location and a second memory location respectively associated with a first device profile and a second device profile,	§IV(B)(1)(b)(1) (“device profile”): INTEL-1005, ¶¶59, 68, Fig. 9 INTEL-1006, ¶¶42, Fig. 3 INTEL-1007, 91-106, Table 38

[1A.2/Structural] wherein said second memory location is in a second memory device in the memory store,	§IV(B)(1)(a)(1) (“memory device”): INTEL-1005, ¶¶17-18, 22, 22 §IV(B)(1)(a)(2) (“memory location”): INTEL-1005, ¶¶22, 24-25, 69, cl. 7, Fig. 2 INTEL-1006, ¶¶3-4 INTEL-1008, 66-67, Fig. 3.2 INTEL-1007, 2 §IV(B)(1)(a)(3) (“memory store”): INTEL-1005, ¶¶17, 51, 54, 80, Fig. 4 INTEL-1017, 89
[1A.3/Read-Write] wherein a memory read transaction or a memory write transaction is performed on at least one of the first memory location or second memory location,	§IV(B)(1)(e)(1) (“controller performs the memory read or memory write transaction on a memory location”): INTEL-1005, ¶¶29, 31, 33, 35 INTEL-1006, ¶24 §IV(B)(1)(c) (“Preamble [1p]”): INTEL-1005, ¶¶33-40, 68, 72, 78, Fig. 2 INTEL-1006, ¶¶25-27, 31, 35, 37 INTEL-1018, 582 INTEL-1007, 2-3, 111-112, 117-118 §IV(B)(1)(d)(1) (“addressing a first memory location...”): INTEL-1005, ¶¶68, 72, 77 INTEL-1006, ¶¶25-27, 29, 35, 37 INTEL-1007, 2
[1A.4/Read-Write] wherein data is transferred to the first memory location or second memory location from a host for a memory write transaction,	§IV(B)(1)(e)(2) (“write data flow path”): INTEL-1005, ¶40, Fig. 2 INTEL-1006, ¶¶25-27, 39 INTEL-1007, 74
[1A.5/Read-Write] wherein data is transferred from the first memory location or second	§IV(B)(1)(e)(3) (“read data flow path”): INTEL-1005, ¶¶38-39 INTEL-1006, ¶¶25-27, 40 INTEL-1007, 3, 43

memory location to the host for a memory read transaction;	
[1A.6/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;	§IV(B)(1)(b)(4) (“device profile is optimal for a data type subject to the memory transaction”): INTEL-1005, ¶¶56-60, 57-68, Fig. 5, Fig. 9 INTEL-1006, ¶42 INTEL-1008, 23 INTEL-1009, 2 INTEL-1019, 4-5 INTEL-1020, 3-14
[1B/Action] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;	§IV(B)(1)(d)(2) (“identifying command details”): INTEL-1005, ¶¶56, 70-72 INTEL-1006, ¶¶34-38 INTEL-1007, 111-112, 117-118
[1B.1/Profile] said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles;	§IV(B)(1)(b)(2) (“attributes”): INTEL-1005, ¶68 INTEL-1006, ¶¶42, 48 INTEL-1010, 5 INTEL-1008, 26 INTEL-1011, 3
[1C/Action] after identifying the command details, obtaining the first set of attributes;	§IV(B)(1)(d)(3) (“obtaining the first set of attributes [and] using attributes from said first and second device profiles”): INTEL-1005, ¶¶17, 51, 56, 70, 73, 77
[1C.1/Profile] wherein each attribute in the first set of attributes and second set of attributes is associated with a	§IV(B)(1)(b)(3) (“attribute qualifier”): INTEL-1005, ¶64, Fig. 9 INTEL-1006, ¶42

respective attribute qualifier that qualifies a respective memory location;	
[1C.2/Structural] wherein the memory store is directly coupled to at least one memory bus and wherein the at least one memory bus is directly coupled to a controller, and	§IV(B)(1)(a)(4) (“controller”): INTEL-1005, ¶¶14,17, 22-42, 51, 56, 78, 80, Fig. 2, Fig. 10 INTEL-1006, ¶¶7, 19, 24, 53-55, Fig. 1 INTEL-1010, 27 INTEL-1008, 58 §IV(B)(1)(a)(5) (“memory bus”): INTEL-1005, ¶¶18, 22, 41, 80, Fig. 2 INTEL-1006, ¶¶24, 34 INTEL-1005, ¶¶56-60, 57-68, Fig. 5, Fig. 9 INTEL-1008, 67
[1C.3/Read-Write] wherein the controller performs the memory read transaction and memory write transaction on the memory store;	§IV(B)(1)(e)(1) (“controller performs the memory read or memory write transaction on a memory location”): INTEL-1005, ¶¶29, 31, 33, 35 INTEL-1006, ¶24 §IV(B)(1)(c) (“Preamble [1p]”): INTEL-1005, ¶¶33-40, 68, 72, 78, Fig. 2 INTEL-1006, ¶¶25-27, 31, 35, 37 INTEL-1018, 582 INTEL-1007, 2-3, 111-112, 117-118 §IV(B)(1)(d)(1) (“addressing a first memory location...”): INTEL-1005, ¶¶68, 72, 77 INTEL-1006, ¶¶25-27, 29, 35, 37 INTEL-1007, 2
[1C.4/Structural] wherein the host is directly coupled to the controller by a communication path as the controller performs the memory read transaction or memory write transaction	§IV(B)(1)(a)(6) (“communication path”): INTEL-1005, ¶¶18, 22, Fig. 1, Fig. 2 INTEL-1006, ¶21

	§IV(B)(1)(e)(1) (“controller performs the memory read or memory write transaction on a memory location”): INTEL-1005, ¶¶22, 78, Fig. 2
[1C.5/Profile] wherein the first device profile and second device profile are each stored in the memory store; and	§IV(B)(1)(b)(1) (“device profile”): INTEL-1005, ¶¶59, 68, Fig. 9 INTEL-1006, ¶42, Fig. 3 INTEL-1007, 91-106, Table 38
[1D/Action] using attributes from said first and second device profiles; and	§IV(B)(1)(d)(3) (“obtaining the first set of attributes [and] using attributes from said first and second device profiles”): INTEL-1005, ¶¶17, 51, 56, 70, 73, 77
[1E/Action] selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes	§IV(B)(1)(d)(4) (“selecting a transfer size for the memory transaction”): INTEL-1005, ¶73, Fig. 2

2. Independent Claim 29 (Computer Program Product Claim)

186. I have reproduced independent claim 29 below in its entirety with labels for each limitation reflecting the limitations place in the claim and functional grouping. The method limitations recited in claim 29 are substantially identical to the limitations in claim 1. The primary difference between claims 1 and 29 is the language of the preamble. Claim 1 is a method claim and claim 29 is a computer readable medium claim. The combination of Danilak and Cornwell renders this claim obvious.

[29P] A non-transitory computer readable medium comprising computer executable instructions adapted to cause a method of performing memory transactions on a first memory location and a second memory location, said method comprising:

[29A] performing a memory transaction comprising addressing a first memory location,

[29A.1/Profile] said first memory location and a second memory location are respectively associated with a first device profile and a second device profile;

[29A.2/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;

[29B] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;

[29B.1/Profile] said first device profile representing a first set of attributes of said first memory device, and said second device profile representing a second set of attributes of said second memory device, and a difference exists between said first and second device profiles;

[29C] after identifying the command details, obtaining the first set of attributes; and

[29D] using attributes from said first and second device profiles; and

[29E] selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.

a) Preamble [29P]

187. The combination of Danilak and Cornwell discloses “*a method of performing memory transactions ...*” as I discussed in §IV(B)(1)(c). Translation units 508/509 of Danilak’s flash interface circuit perform the recited “action” steps of the method claims [29A]-[29E] which are identical to the method steps of claim 1. (*See* §§IV(B)(1)(d)(1)-(4).)

188. Danilak discloses translation units 508/509 “may be implemented in any desired fashion (combinatorial logic circuitry, pipelined circuitry, processor-based software, state machines, various other circuitry, and/or any combination of the foregoing).” (INTEL-1005, ¶55.) A POSITA would have been motivated to implement the translation units and other management functions in the flash interface circuit using “processor-based software” as taught by Danilak because it was well-known that Flash requires software to operate. (*See* INTEL-1008, 32.) For example, a flash device powers on in a ‘read’ state and requires a series of commands to put it into another state such as program or erase. (Brewer, 32.) Software is therefore needed to put the attached flash devices in the proper mode. Additionally, a POSITA would have been motivated to use software to implement translation units 508/509 to permit updates to management functions performed by the flash interface controller. A POSITA would have understood that using

software to implement some of the flash management functions provided a strong advantage compared to a hardware-only implementation, because the algorithms could be changed to accommodate differences between flash vendors, technology evolution to denser chips, and characteristics for wear leveling and error correcting. Cornwell confirms that it was known to implement memory controller functions using software, teaching its controller is implemented with “firmware.” (INTEL-1006, ¶29.)

189. A POSITA would have understood that “processor-based software” is stored on “*non-transitory computer readable medium*” and comprises “*executable instructions*.” A POSITA would have understood that all software comprises “executable instructions” and would further have understood that while running, programs are stored in memory, which is a non-transitory computer readable medium. (See INTEL-1022, 20.)

b) “Device Profile” Limitations

190. The device profile limitations of claim 29 are substantially identical to the device profile limitations of claim 1. The only difference occurs in limitations [1A.1/Profile] and [29A.1/Profile], indicated by underline, and this difference is non-material. Thus, for the reasons I discuss in §§IV(B)(1)(b)(1)-(2) and IV(B)(1)(b)(4) for limitations [1A.1/Profile], [1A.6/Profile], and [1B.1/profile], the

combination of Danilak and Cornwell also discloses limitation [29A.1/Profile], [29A.2/Profile], and [29B.1/Profile].

Claim 1	Claim 29
[1A.1/Profile] said first memory location and a second memory location respectively associated with a first device profile and a second device profile,	[29A.1/Profile] said first memory location and a second memory location <u>are</u> respectively associated with a first device profile and a second device profile,
[1A.6/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;	[29A.2/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;
[1B.1/Profile] said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles;	[29B.1/Profile] said first device profile representing a first set of attributes of said first memory device, and said second device profile representing a second set of attributes of said second memory device, and a difference exists between said first and second device profiles;

c) Action Steps

191. The action limitations recited in independent claim 29 are substantially identical to the action limitations of claim 1. The sole difference is in limitations [1A] and [29A]. Limitation [1A] recites that the “*first memory location*” is “*in a first memory device in a memory store.*” Limitation [29A] does not include this language, making this claim language broader. Addressing a first memory location

in a first memory device in a memory store meets the limitation of “*addressing a first memory location.*” Thus, for the reasons I discuss in §§IV(B)(1)(d)(1)-(4) for limitations [1A], [1B], [1C], [1D], and [1E], the combination of Danilak and Cornwell also discloses limitation [29A], [29B], [29C], [29D], and [29E].

Claim 1	Claim 29
[1A] performing a memory transaction comprising addressing a first memory location <u>in a first memory device in a memory store</u> :	[29A] performing a memory transaction comprising addressing a first memory location,
[1B] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device	[29B] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;
[1C] after identifying the command details, obtaining the first set of attributes;	[29C] after identifying the command details, obtaining the first set of attributes; and
[1D] using attributes from said first and second device profiles; and	[29D] using attributes from said first and second device profiles; and
[1E] selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.	[29E] selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.

d) Summary

192. In summary, following table presents the preamble and each limitation in order, the section or sections addressing the limitation, and the evidence cites.

Claim 29	
[29P] A non-transitory computer readable medium comprising computer executable instructions adapted to cause a method of performing memory transactions on a first memory location and a second memory location, said method comprising:	§IV(B)(2)(a) (“Preamble [29P]”): INTEL-1005, ¶55 INTEL-1006, ¶29 INTEL-1008, 32 §IV(B)(1)(a)(2) (“memory location”): INTEL-1005, ¶¶22, 24-25, 69, claim 7, Fig. 2 INTEL-1006, ¶¶3-4 INTEL-1008, 66-67, Fig. 3.2 INTEL-1007, 2 §IV(B)(1)(c) (“Preamble [1p]”): INTEL-1005, ¶¶33-40, 68, 72, 78, Fig. 2 INTEL-1006, ¶¶25-27, 31, 35, 37 INTEL-1018, 582 INTEL-1007, 2-3, 111-112, 117-118
[29A] performing a memory transaction comprising addressing a first memory location,	§IV(B)(2)(c) (“action steps”), §IV(B)(1)(a)(2) (“memory location”):

	INTEL-1005, ¶¶22, 24-25, 69, claim 7, Fig. 2 INTEL-1006, ¶¶3-4 INTEL-1008, 66-67, Fig. 3.2 INTEL-1007, 2 §IV(B)(1)(d)(1) (“addressing a first memory location...”): INTEL-1005, ¶¶68, 72, 77 INTEL-1006, ¶¶25-27, 29, 35, 37 INTEL-1007, 2
[29A.1/Profile] said first memory location and a second memory location are respectively associated with a first device profile and a second device profile;	§§IV(B)(2)(b) (“device profile limitations”), IV(B)(1)(b)(1) (“device profile”): INTEL-1005, ¶¶59, 68, Fig. 9 INTEL-1006, ¶¶42, Fig. 3 INTEL-1007, 91-106, Table 38
[29A.2/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;	§§IV(B)(2)(b) (“device profile limitations”), IV(B)(1)(b)(4) (“device profile is optimal for a data type subject to the memory transaction”):

	INTEL-1005, ¶¶56-60, 57-68, Fig. 5, Fig. 9 INTEL-1006, ¶42 INTEL-1008, 23 INTEL-1009, 2 INTEL-1019, 4-5 INTEL-1020, 3-14
[29B] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;	§§IV(B)(2)(c) (“action steps”), IV(B)(1)(d)(2) (“identifying command details”): INTEL-1005, ¶¶56, 70-72 INTEL-1006, ¶¶34-38 INTEL-1007, 111-112, 117-118
[29B.1/Profile] said first device profile representing a first set of attributes of said first memory device, and said second device profile representing a second set of attributes of said second memory device, and a difference exists between said first and second device profiles;	§§IV(B)(2)(b) (“device profile limitations”), IV(B)(1)(b)(2) (“attributes”): INTEL-1005, ¶68 INTEL-1006, ¶¶42, 48 INTEL-1010, 4-5
[29C] after identifying the command details, obtaining the first set of attributes; and	§§IV(B)(2)(c) (“action steps”), IV(B)(1)(d)(3) (“obtaining the first set of attributes [and] using attributes from said first and second device profiles”): INTEL-1005, ¶¶17, 51, 56, 70, 73, 77

[29D] using attributes from said first and second device profiles; and	§§IV(B)(2)(c) (“action steps”), IV(B)(1)(d)(3) (“obtaining the first set of attributes [and] using attributes from said first and second device profiles”): INTEL-1005, ¶¶17, 51, 56, 70, 73, 77
[29E] selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes	§§IV(B)(2)(c) (“action steps”), §IV(B)(1)(d)(4) (“selecting a transfer size for the memory transaction”): INTEL-1005, ¶73, Fig. 2

3. Independent Claim 59⁵ (Memory Controller)

193. I have reproduced independent claim 59 below in its entirety with labels for each limitation reflecting the limitations place in the claim and functional grouping.

[59P] A memory controller comprising:

⁵ Because of the overlap in claim limitations, I address independent claim 45 having means plus function language after claim 59.

[59A] an interface controller coupled to a memory device interface and an input/output (IO) device interface;

[59B] a memory store; wherein the memory device interface is directly coupled to the memory store;

[59B.1/Action] said interface controller disposed to perform a memory transaction by addressing a first memory location in the memory store,

[59B.2/Profile] said first memory location and a second memory location respectively associated with a first device profile and a second device profile;

[59B.3/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;

[59B.4/Action] said interface controller identifies command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;

[59B.5/Profile] said device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles;

[59B.6/Action] said interface controller obtaining the first set of attributes after identifying the command details; and

[59B.7/Action] said addressing of said first memory location includes using said attributes from said first device profile; and

[59B.8/Action] said addressing of said first memory location includes selecting a transfer size for the memory transaction, wherein

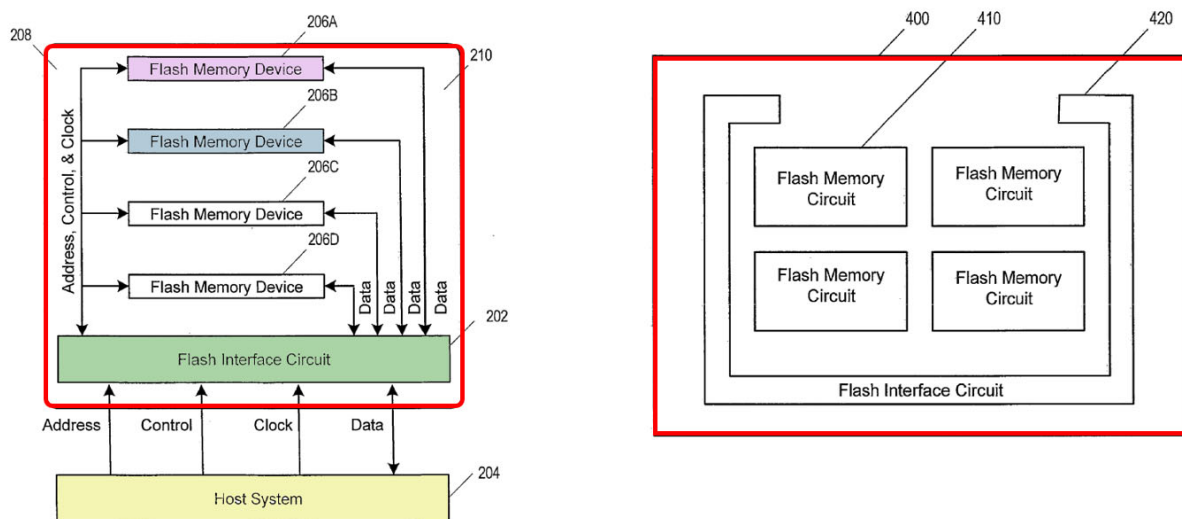
the transfer size is a function of a data size of the memory transaction and the first set of attributes.

a) Preamble [59P]

194. The flash interface circuit in the combination of Danilak and Cornwell is a “*memory controller*” [59P] as I discussed in §IV(B)(1)(a)(4).

195. The preamble of claim 59 requires the memory controller also include the “*memory store*.” The ’190 patent does not disclose any embodiments where the memory store is included within the memory controller. At most, the ’190 patent discloses the memory store “may be part of a computing device or equivalent device, such as a storage device 15, that includes multi-profile memory controller 2.” (INTEL-1001, 2:53-57.) Should an argument be made that the preamble is limiting and requires the controller and memory store to be part of a single device, Danilak discloses such an implementation.

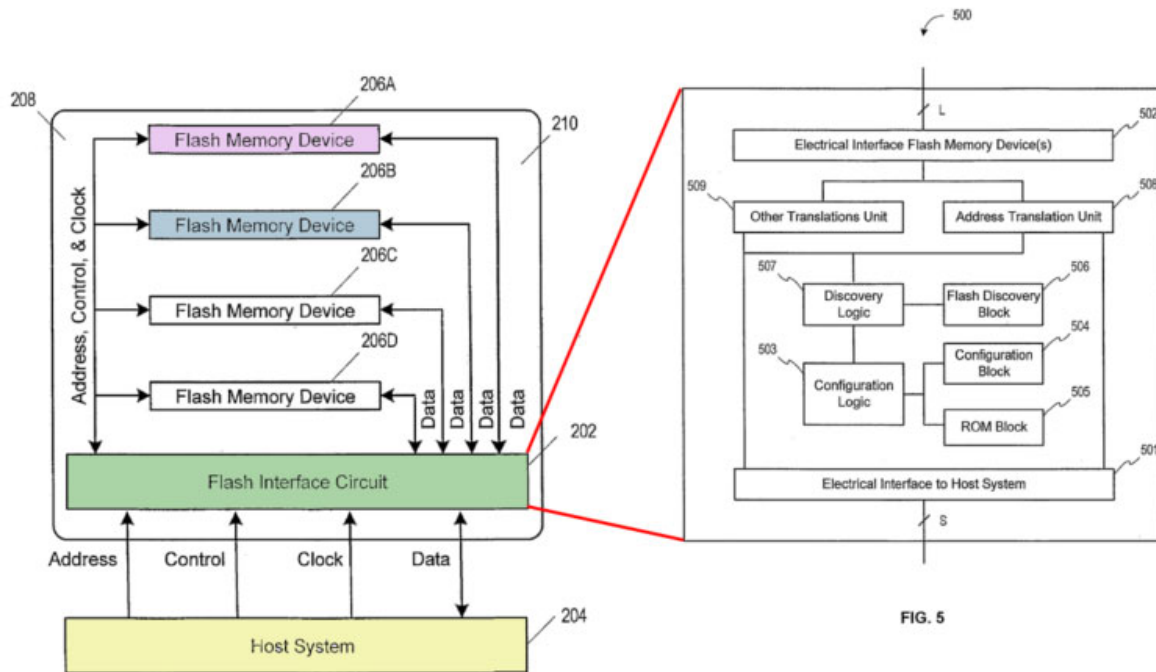
196. In Danilak, “the flash interface circuit may be integrated with one or more flash devices onto a single monolithic semiconductor die.” (INTEL-1005, ¶54.) Figure 4, reproduced below, illustrates “a die 400 including one or more flash memory circuits 410 and one or more flash interface circuits 420.” (INTEL-1005, ¶54.) This die (indicated by red outline) including the controller and memory store is therefore a “*memory controller*.”



Danilak, Figure 2 (left); Figure 4 (right)

b) “Interface Controller” [59A].

197. The combination of Danilak and Cornwell discloses “*an interface controller coupled to a memory device interface and an input/output (IO) device interface.*” Flash interface circuit 500, illustrated in Figure 5 below right, “includes an electrical interface to the host system 501, an electrical interface to the flash memory device(s) 502, configuration logic 503, a configuration block 504, a read-only memory (ROM) block 505, a flash discovery block 506, discovery logic 507, an address translation unit 508, and a unit for translations other than address translations 509.” (INTEL-1005, ¶55.)



Danilak, Annotated Figure 2 (left), Figure 5 (right)

198. The '190 patent discloses that the “[i]nterface controller 52 processes memory transaction request 6 by interpreting the memory transaction request and performing a memory transaction on a memory location.” ('190 patent, 7:21-24; *see also*, 6:53-55.) Translation units 508/509 are collectively the “*interface controller*” because they interpret received memory transaction requests from a host and perform the required memory operations on the physical memory devices. Translation units 508/509 “translate host flash memory access and configuration requests into requests to one or more flash memory devices” and “translate flash memory replies to host system replies if needed.” (INTEL-1005, ¶56.) As part of the request translation, address translation unit 508 “translate[s] addresses between the host and the flash memory devices.” (INTEL-1005, ¶69.) Translation units

508/509 may further “be configured to modify requests provided from the host system based on differences between the virtual configuration presented by the interface circuit 500 to the host system and the physical configuration of the flash memory devices.” (INTEL-1005, ¶56.)

199. Danilak’s “electrical interface 502 comprises the circuitry to drive and/or receive signals to/from the flash memory devices.” (INTEL-1005, ¶55.) The electrical interface 502 is the “*memory device interface*” because it is the interface to the flash memory devices. Electrical interface 502 is consistent with the ’190 patent which describes the memory device interface as “operat[ing] as the communication interface between memory store 4 and devices that form part of the multi-profile memory controller” and “provid[ing] the required physical signaling, such as by asserting electrical signals on memory bus 60, memory bus 62, or both.” (INTEL-1001, 6:35-37, 6:43-45.) As illustrated in Figure 5 above right, Danilak discloses the “*interface controller* [translation units 508 and 509] *coupled to a memory device interface* [interface 502].”

200. The “electrical interface 501 comprises the circuitry to drive and/or receive signals to/from the host system.” (INTEL-1005, ¶55.) The electrical interface to the host system 501 is the recited “*input/output (IO) device interface*” because it acts as an interface to the host. The interface to host system 501 is consistent with the ’190 patent which describes the IO device interface 54 as

“operat[ing] as the communication interface between an IO device, such as IO device 8, and devices that form part of multi-profile memory controller 2, such as memory device interface 50 and interface controller 52” and “provid[ing] the required physical signaling, such as by asserting electrical signals on communication path 68, that will enable IO device interface 54 to use path 68 when communicating with an IO device, such as by receiving memory transaction request 6.” (INTEL-1001, 6:65-7:2, 7:11-15.) As illustrated in Figure 5 above right, the translation units 508 and 509 are also coupled to the “*input/output (IO) device interface.*”

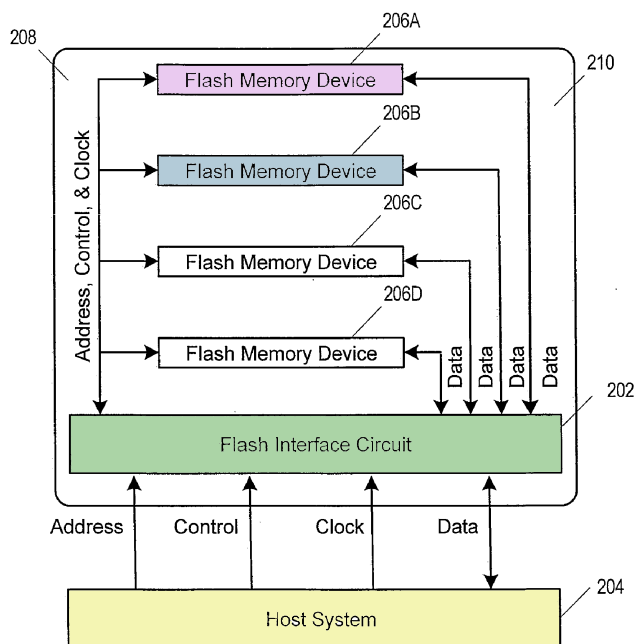
c) “*Memory Store*” [59B]

201. The combination of Danilak and Cornwell discloses “*a memory store; wherein the memory device interface is directly coupled to the memory store*” [59B]. In Danilak, the flash memory devices are exposed “as at least one flash memory device” to the rest of the system via flash interface circuit 102 (shaded green in Figure 2 above). (INTEL-1005, ¶17; *see also*, ¶¶51, 81.) The virtual flash memory device comprised of multiple physical flash memory devices is a “*memory store.*” (INTEL-1001, 2:63 (memory store “includes addressable memory locations”); 10:13-17 (memory store includes memory devices “grouped so that each of the memory devices are associated with only one memory bus controller”);

INTEL-1017, 89 (“3.2126 store. (1) A device into which data can be placed, in which they can be retained, and from which they can be retrieved”).)

202. Each flash memory device in Danilak’s “*memory store*” includes one or more addressable portions and therefore the “*memory store*” includes multiple “*memory locations*.” As I discussed in §IV(B)(1)(a)(2), a “flash memory is typically divided into sub-units, portions, or blocks.” (INTEL-1005, ¶24.) As shown in Figure 2 below, each flash memory device is connected to an address bus over which the address for the memory operation is provided. (INTEL-1005, Figure 2; INTEL-1008, 66-67.) Danilak also refers to addressable memory “locations” in its specification. (INTEL-1005, claim 7 (referring to “storage locations in the one or more flash memory devices”); ¶25 (describing accessing “physical memory location”).) Cornwell confirms a flash memory device includes multiple “*memory location[s]*”, teaching that flash memory “can be read or programmed a byte or word at a time in a random-access fashion” but “is usually erased a block at a time.” (INTEL-1006, ¶4.) Indeed, it is inherent that a memory device (and hence a “*memory store*” with multiple devices) includes “*memory locations*” because the sole purpose of a memory is to store data in an accessible location and provide the stored data upon request. (See INTEL-1008, 66-67; INTEL-1007, 2.) The ’190 patent also acknowledges this well-known fact stating

that NAND flash “includes a set of addressable non-volatile memory cells, named flash blocks.” (INTEL-1001, 4:44-45.)



Danilak, Figure 2

203. Danilak’s interface 502 “drive[s] and/or receive[s] signals to/from the flash memory devices.” (INTEL-1005, ¶55.) Because interface 502 is connected to each flash memory device in the “*memory store*”, interface 502 “*is directly coupled to the memory store.*”

d) “Device Profile” Limitations

204. The device profile limitations of claim 59 are substantially identical to the device profile limitations of claim 1. The only difference is that limitation 59B.7 recites only a “*device profile*” rather than a “*first device profile.*” However, I assume this omission of “*first*” in claim 59 is a typographical error. Thus, for the

reasons I discuss in §§IV(B)(1)(b)(1)-(2) and IV(B)(1)(b)(4) for limitations [1A.1/Profile], [1A.6/Profile], and [1B.1/profile], the combination of Danilak and Cornwell also discloses limitations [59B.2/Profile], [59B.3/Profile], and [59B.5/Profile].

Claim 1	Claim 59
[1A.1/Profile] said first memory location and a second memory location respectively associated with a first device profile and a second device profile,	[59B.2/Profile] said first memory location and a second memory location respectively associated with a first device profile and a second device profile,
[1A.6/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;	[59B.3/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;
[1B.1/Profile] said <u>first</u> device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles;	[59B.5/Profile] said device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles;

e) Action Limitations

205. Claim 59 recites a set of actions performed by the interface controller. These actions are substantially the same as the set of actions recited in claim 1. The primary difference between the claims is that claim 59 specifies the “*interface*

controller” performs the recited steps. As I discussed in §§IV(B)(2)(a) and IV(B)(3)(b), translation units 508/509 in the flash interface circuit perform the “*addressing*”, “*identifying*”, “*obtaining*”, “*using*” and “*selecting*” actions.

Therefore, for these reasons and the reasons I discussed in §§IV(B)(1)(d)(1)-(4) for limitations [1A]-[1E], the combination of Danilak and Cornwell also discloses limitations [59B.1], [59B.4], and [59B.6]-[59B.8].

Claim 1	Claim 59
[1A] <u>performing</u> a memory transaction comprising addressing a first memory location <u>in a first memory device</u> in a memory store:	[59B.1] said <u>interface controller</u> <u>disposed to perform</u> a memory transaction by addressing a first memory location in the memory store,
[1B] <u>said performing the memory transaction further comprising identifying</u> command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;	[59B.4] <u>said interface controller identifies</u> command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;
[1C] after identifying the command details, obtaining the first set of attributes;	[59B.6] <u>said interface controller</u> obtaining the first set of attributes after identifying the command details;
[1D] using attributes from said first and second device profiles; and	[59B.7] <u>said addressing of said first memory location includes</u> using said attributes from said first device profile;
[1E] selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.	[59B.8] <u>said addressing of said first memory location includes</u> selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory

	transaction and the first set of attributes.
--	--

f) Summary

206. In summary, following table presents the preamble and each limitation in order, the section or sections addressing the limitation, and the evidence cites.

Claim 59	
[59P] A memory controller comprising:	§IV(B)(3)(a) (“Preamble [59P]”): INTEL-1005, ¶54, Fig. 4 §IV(B)(1)(a)(4) (“controller”): INTEL-1005, ¶¶14, 22-42, 51, 56, 78, 80, Fig. 2, Fig. 10 INTEL-1006, ¶¶7, 19, 24, 53-55, Fig. 1 INTEL-1010, 27 INTEL-1008 at 58
[59A] an interface controller coupled to a memory device interface and an input/output (IO) device interface,	§IV(B)(3)(b) (“interface controller”): INTEL-1005, ¶¶55-56, 69, Fig. 5
[59B] a memory store; wherein the memory device interface is directly coupled to the memory store;	§IV(B)(3)(c) (“memory store”): INTEL-1005, ¶¶24-25, 51, 55, 81, Fig. 2, claim 7 INTEL-1006, ¶4 INTEL-1017, 89

	INTEL-1008, 66-67 INTEL-1007, 2
[59B.1/Action] said interface controller disposed to perform a memory transaction by addressing a first memory location in the memory store,	§IV(B)(3)(e) (“action limitations”), §IV(B)(1)(a)(1) (“memory device”): INTEL-1005, ¶¶17-18, 22, 22 §IV(B)(1)(a)(2) (“memory location”): INTEL-1005, ¶¶22, 24-25, 69, claim 7, Fig. 2 INTEL-1006, ¶¶3-4 INTEL-1008, 66-67, Fig. 3.2 INTEL-1007, 2 §IV(B)(1)(a)(3) (“memory store”): INTEL-1005, ¶¶51, 54, 80, Fig. 4 INTEL-1017, 89 §IV(B)(1)(d)(1) (“addressing a first memory location...”): INTEL-1005, ¶¶68, 72, 77 INTEL-1006, ¶¶25-27, 29, 35, 37 INTEL-1007, 2
[59B.2/Profile] said first memory location and a second memory location respectively associated with a first device profile and a second device profile;	§§IV(B)(3)(d) (“device profile limitations”),

	IV(B)(1)(b)(1) (“device profile”): INTEL-1005, ¶¶59, 68, Fig. 9 INTEL-1006, ¶¶42, Fig. 3 INTEL-1007, 91-106, Table 38
[59B.3/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;	§§IV(B)(3)(d) (“‘device profile’ limitations”), IV(B)(1)(b)(4) (“device profile is optimal for a data type subject to the memory transaction”): INTEL-1005, ¶¶56-60, 57-68, Fig. 5, Fig. 9 INTEL-1006, ¶42 INTEL-1008, 23 INTEL-1009, 2 INTEL-1019, 4-5 INTEL-1020, 3-14
[59B.4/Action] said interface controller identifies command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;	§§IV(B)(3)(e) (“action limitations”), IV(B)(1)(d)(2) (“identifying command details”): INTEL-1005, ¶¶56, 70-72 INTEL-1006, ¶¶34-38 INTEL-1007, 111-112, 117-118

[59B.5/Profile] said device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles;	§§IV(B)(3)(d) (“‘device profile’ limitations”), IV(B)(1)(b)(2) (“attributes”): INTEL-1005, ¶68 INTEL-1006, ¶¶42, 48 INTEL-1010, 4-5
[59B.6/Action] said interface controller obtaining the first set of attributes after identifying the command details; and	§§IV(B)(3)(e) (“action limitations”), IV(B)(1)(d)(3) (“obtaining the first set of attributes [and] using attributes from said first and second device profiles”): INTEL-1005, ¶¶51, 56, 70, 73, 77
[59B.7/Action] said addressing of said first memory location includes using said attributes from said first device profile; and	§§IV(B)(3)(e) (“action limitations”), IV(B)(1)(d)(3) (“obtaining the first set of attributes [and] using attributes from said first and second device profiles”): INTEL-1005, ¶¶51, 56, 70, 73, 77
[59B.8/Action] said addressing of said first memory location includes selecting a transfer size for the memory transaction, wherein the transfer size is a	§§IV(B)(3)(e) (“action limitations”),

function of a data size of the memory transaction and the first set of attributes.	IV(B)(1)(d)(4) (“selecting a transfer size for the memory transaction”): INTEL-1005, Fig. 2
--	---

4. Independent Claim 45 (Memory Controller)

207. I have reproduced independent claim 45 below in its entirety with labels for each limitation reflecting the limitations place in the claim and functional grouping.

[45P] A memory controller for performing memory transactions, said memory controller comprising:

[45A] means for performing a memory transaction comprising

[45A.1] a means for addressing a first memory location,

[45A.2/Profile] said first memory location and a second memory location are respectively associated with a first device profile and a second device profile;

[45A.3a] said means for addressing is directly coupled to the first memory location and second memory location;

[45A.3b] wherein said means for addressing includes an interface controller means for processing memory transaction requests;

[45A.3c] said means for addressing is directly coupled to a host as said means for addressing addresses the first memory location or second memory location;

[45A.4/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;

[45B] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;

[45C/Profile] said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles;

[45D] said means for addressing obtains the first set of attributes after identifying the command details; and

[45E] said means for addressing uses attributes from said first and second device profiles and selects a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.

a) Preamble [45P] and “Means for Performing” [45A]

208. The function of the “*means for performing a memory transaction*” is “performing a memory transaction” and the structure is a memory controller. (See §III(C)(1).) The combination of Danilak and Cornwell discloses this limitation.

209. Danilak’s flash interface circuit is a “memory controller”, meeting the structure for the “*means for performing.*” The flash interface circuit further

“perform[s] a memory transaction”, as I discussed in §§IV(B)(1)(a)(4), IV(B)(1)(c), IV(B)(1)(d)(1), and IV(B)(1)(e)(1). Specifically, translation units 508/509 in the flash interface translate a memory access request received from a host “into one or more physical flash memory device accesses”, “emulate attributes of the virtual configuration that differ from the physical flash memory devices”, and “return an appropriate response to the request to the host.” (INTEL-1005, ¶78; *see also* INTEL-1005, ¶56 (“translation units 508 and 509 may translate host flash memory access and configuration requests into requests to one or more flash memory devices, and may translate flash memory replies to host system replies if needed).)

210. The combination of Danilak and Cornwell discloses the “*means for performing a memory transaction*” for these reasons and the reasons I discussed in §§IV(B)(1)(a)(4), IV(B)(1)(c), IV(B)(1)(d)(1), and IV(B)(1)(e)(1).

211. The preamble of claim 45 recites a “*memory controller for performing memory transactions*” [45P]. The body of claim 45 includes a single means-plus-function claim (“*means for performing*”) which is further narrowed by limitations in the claims. The preamble is coextensive with the “*means for performing a memory transaction.*” Therefore, should an argument be made that the preamble is limiting, the combination of Danilak and Cornwell discloses the preamble for the same reasons as it discloses the “*means for performing a memory transaction.*”

b) “Means for Addressing” Limitations
[45A.1]/[45A.3a]/[45A.3c]/[45D]/[45E]

212. The function of the “*means for addressing*” is “addressing a first memory location”, “obtain[ing] the first set of attributes after identifying the command details”, “us[ing] attributes from said first and second device profiles”, and “select[ing] a transfer size for the memory transaction” and the structure is the interface controller, memory device interface, and IO interface, as I discussed in §III(C)(2) (claim construction).

213. The combination of Danilak and Cornwell discloses an interface controller (translation units 508/509), a memory device interface (interface 502), and an IO interface (interface 501) as I discussed in §§IV(B)(3)(b). These structures collectively meet the structure for the “*means for addressing*.”

214. The interface controller further performs the function of “addressing a first memory location” as I discussed in §§IV(B)(1)(d)(1), IV(B)(3)(b), and IV(B)(3)(e). The interface controller further performs the functions of “obtain[ing] the first set of attributes after identifying the command details” (discussed in §§IV(B)(1)(d)(3) and IV(B)(3)(e)), “us[ing] attributes from said first and second device profiles” (§§IV(B)(1)(d)(3) and IV(B)(3)(e)), and “select[ing] a transfer size for the memory transaction” (discussed in §§IV(B)(1)(d)(4) and IV(B)(3)(e)).

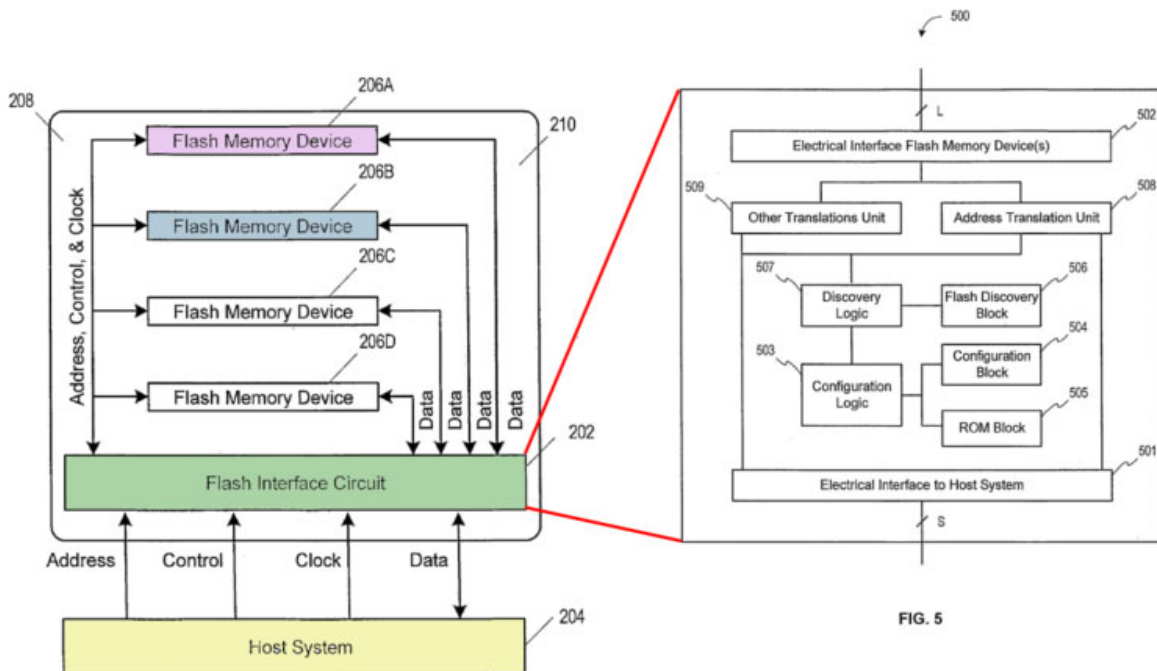
215. Accordingly, the combination of Danilak and Cornwell discloses “*means for addressing a first memory location*” [45A.1], “*said means for*

addressing obtains the first set of attributes after identifying the command details” [45D], and “*said means for addressing uses attributes from said first and second device profiles and selects a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes*” [45E].

216. Danilak’s interface 502 is connected to each memory device in the memory store and each memory device has a set of addressable locations, as I discussed in §§ IV(B)(1)(a)(2), IV(B)(1)(a)(5), and IV(B)(1)(a)(6). Accordingly, in the combination of Danilak and Cornwell, interface 502 is coupled to a first memory location in a first memory device and a second memory location in a second memory device. Because interface 502 is part of the “*means for addressing*”, the combination of Danilak and Cornwell discloses “*said means for addressing is directly coupled to the first memory location and second memory location*” [45A.3a].

217. Danilak’s interface 501 is connected to the host, as I discussed in §§ IV(B)(1)(a)(6) and IV(B)(1)(e)(1). As shown in Figure 2 below left, address, control, clock and data signals are communicated over the connection between host 204 and flash interface circuit 202. (*See also*, INTEL-1005, ¶22; *see also*, ¶18 (“Also shown are exemplary data bus & control signals between the flash interface circuit 102 and a host system 106”).) In Danilak, “all signals from the host system

are received by the flash interface circuit before presentation to the flash memory devices” and “all signals from the flash memory devices are received by the flash interface circuit before being presented to the host system 204.” (INTEL-1005, ¶22.) When the host requests a memory access, translation units 508/509 “return an appropriate response to the request to the host” using interface 501 after the transaction is performed. (INTEL-1005, ¶78.) Thus, the combination of Danilak and Cornwell discloses interface 501 “*is directly coupled to a host*” as the translation units 508/509 “*address[] the first memory location or the second memory location.*”



Danilak, Annotated Figure 2 (left), Figure 5 (right)

218. Because the translation units (interface controller) and interface 501 (IO interface) are part of the “*means for addressing*”, the combination of Danilak and

Cornwell disclose “*means for addressing is directly coupled to a host as said means for addressing addresses the first memory location or second memory location*” [45A.3c].

c) “Interface Controller Means”

219. The function of the “*interface controller means*” is “processing memory transaction requests” and the structure is an interface controller, as I discussed in §III(C)(3) (claim construction).

220. The combination of Danilak and Cornwell discloses an interface controller (translation units 508/509) as I discussed in § IV(B)(3)(b). Translation units 508/509 process memory transactions by, e.g., translating received requests to memory operations on physical memory devices, addressing the memory operations, and formulating memory operations to account for differences between the virtual configuration and the physical configuration. Thus, translation units 508/509 “*process[] transaction requests.*”

221. For these reasons and the reasons I discussed in § IV(B)(3)(b), the combination of Danilak and Cornwell discloses the “*means for addressing*” (memory device interface, interface controller, and IO interface) includes an “*interface controller means for processing memory transaction requests*” [45A.3b].

d) “Identifying Command Details” [45B]

222. Claim 45 further recites “*performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device*” [45B]. I note that claim 45 does not associate this action with a particular claimed structure.

223. Translation units 508/509 (interface controller) perform the “*identifying*” action, as I discussed in § IV(B)(1)(d)(2). Therefore, this action is performed by the “*interface controller means*”, the “*means for addressing*” which includes the “*interface controller means*” and the “*means for performing*” which includes the “*means for addressing.*”

224. For these reasons and the reasons I discussed in §§ IV(B)(1)(d)(2) and IV(B)(4)(a)-(b), the combination of Danilak and Cornwell discloses “*performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device*” [45B].

e) Device Profile Limitations

225. The device profile limitations of claim 45 are substantially identical to the device profile limitations of claim 1 and claim 59. The only difference is that limitation 59B.7 recites only a “*device profile*” rather than a “*first device profile*”

as recited in claims 1 and 45. Thus, for the reasons I discuss in §§IV(B)(1)(b)(1)-(2) and IV(B)(1)(b)(4) for limitations [1A.1/Profile], [1A.6/Profile], and [1B.1/profile], the combination of Danilak and Cornwell also discloses limitations [45A.2/Profile], [45A.4/Profile], and [4C/Profile].

Claim 1	Claim 45
[1A.1/Profile] said first memory location and a second memory location respectively associated with a first device profile and a second device profile,	[45A.2/Profile] said first memory location and a second memory location <u>are</u> respectively associated with a first device profile and a second device profile,
[1A.6/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;	[45A.4/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type
[1B.1/Profile] said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles;	[45C/Profile] said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles

f) Summary

226. In summary, following table presents the preamble and each limitation in order, the section or sections addressing the limitation, and the evidence cites.

Claim 45	
[45P] A memory controller for performing memory transactions, said memory controller comprising:	§IV(B)(4)(a) (“preamble and ‘means for performing’”): INTEL-1005, ¶¶78, 56 §IV(B)(1)(a)(4) (“controller”): INTEL-1005, ¶¶14, 22-42, 51, 56, 78, 80, Fig. 2, Fig. 10 INTEL-1006, ¶¶7, 19, 24, 53-55, Fig. 1 INTEL-1010, 27 INTEL-1008 at 58 §IV(B)(1)(e)(1) (“controller performs the memory read or memory write transaction on a memory location”): INTEL-1005, ¶¶29, 31, 33, 35 INTEL-1006, ¶24 §§IV(B)(1)(c) (“Preamble [1p]”): INTEL-1005, ¶¶33-40, 68, 72, 78, Fig. 2 INTEL-1006, ¶¶25-27, 31, 35, 37 INTEL-1018, 582 INTEL-1007, 2-3, 111-112, 117-118 §IV(B)(1)(d)(1) (“addressing a first

	memory location...”): INTEL-1005, ¶¶68, 72, 77 INTEL-1006, ¶¶25-27, 29, 35, 37 INTEL-1007, 2
[45A] means for performing a memory transaction comprising	§IV(B)(4)(a) (“preamble and ‘means for performing’”) INTEL-1005, ¶¶78, 56 §IV(B)(1)(a)(4) (“controller”): INTEL-1005, ¶¶14, 22-42, 51, 56, 78, 80, Fig. 2, Fig. 10 INTEL-1006, ¶¶7, 19, 24, 53-55, Fig. 1 INTEL-1010, 27 INTEL-1008, 58 §IV(B)(1)(e)(1) (“controller performs the memory read or memory write transaction on a memory location”): INTEL-1005, ¶¶29, 31, 33, 35 INTEL-1006, ¶24 §§IV(B)(1)(c) (“Preamble [1p]”): INTEL-1005, ¶¶33-40, 68, 72, 78, Fig. 2

	INTEL-1006, ¶¶25-27, 31, 35, 37 INTEL-1018, 582 INTEL-1007, 2-3, 111-112, 117-118 §IV(B)(1)(d)(1) (“addressing a first memory location...”): INTEL-1005, ¶¶68, 72, 77 INTEL-1006, ¶¶25-27, 29, 35, 37 INTEL-1007, 2
[45A.1] a means for addressing a first memory location,	§IV(B)(4)(b) (“‘means for addressing’ limitations”) INTEL-1005, ¶¶78, 56 §IV(B)(3)(b) (“interface controller”): INTEL-1005, ¶¶55-56, 69, Fig. 5 §IV(B)(1)(d)(1) (“addressing a first memory location...”): INTEL-1005, ¶¶68, 72, 77 INTEL-1006, ¶¶25-27, 29, 35, 37 INTEL-1007, 2 §IV(B)(1)(d)(3) (“obtaining the first set of attributes

	[and] using attributes from said first and second device profiles”): INTEL-1005, ¶¶51, 56, 70, 73, 77 §IV(B)(1)(d)(4) (“selecting a transfer size for the memory transaction”): INTEL-1005, ¶73, Fig. 2
[45A.2/Profile] said first memory location and a second memory location are respectively associated with a first device profile and a second device profile;	§IV(B)(4)(d) (“device profile limitations”) §IV(B)(1)(b)(1) (“device profile”): INTEL-1005, ¶¶59, 68, Fig. 9 INTEL-1006, ¶¶42, Fig. 3 INTEL-1007, 91-106, Table 38
[45A.3a] said means for addressing is directly coupled to the first memory location and second memory location;	§IV(B)(4)(b) (“‘means for addressing’ limitations”) §IV(B)(1)(a)(2) (“memory location”): INTEL-1005, ¶¶22, 24-25, 69, claim 7, Fig. 2 INTEL-1006, ¶¶3-4 INTEL-1008, 66-67, Fig. 3.2 INTEL-1007, 2

	§IV(B)(1)(a)(5) (“memory bus”): INTEL-1005, ¶¶18, 22, 41, 80, Fig. 2 INTEL-1006, ¶¶24, 34 INTEL-1017, 13, 57 INTEL-1008 at 67 §IV(B)(1)(a)(6) (“communication path”): INTEL-1005, ¶¶18, 22, Fig. 1, Fig. 2 INTEL-1006, ¶21
[45A.3b] wherein said means for addressing includes an interface controller means for processing memory transaction requests;	§IV(B)(4)(c) (“interface controller means”) §IV(B)(3)(b) (“interface controller”): INTEL-1005, ¶¶55-56, 69, Fig. 5
[45A.3c] said means for addressing is directly coupled to a host as said means for addressing addresses the first memory location or second memory location;	§IV(B)(4)(b) (“means for addressing’ limitations”): INTEL-1005, ¶¶ 18, 22, 78, Fig. 2 §IV(B)(1)(a)(6) (“communication path”): INTEL-1005, ¶¶18, 22, Fig. 1, Fig. 2 INTEL-1006, ¶21 §IV(B)(1)(e)(1) (“controller

	performs the memory read or memory write transaction on a memory location”): INTEL-1005, ¶¶22, 78, Fig. 2
[45A.4/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;	§IV(B)(4)(d) (“device profile limitations”) §IV(B)(1)(b)(4) (“device profile is optimal for a data type subject to the memory transaction”): INTEL-1005, ¶¶56-60, 57-68, Fig. 5, Fig. 9 INTEL-1006, ¶42 INTEL-1008, 23 INTEL-1009, 2 INTEL-1019, 4-5 INTEL-1020, 3-14
[45B] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;	§IV(B)(1)(d)(2) (“identifying command details”): INTEL-1005, ¶¶56, 70-72 INTEL-1006, ¶¶34-38 INTEL-1007, 111-112, 117-118
[45C/Profile] said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles;	§IV(B)(4)(d) (“device profile limitations”) §IV(B)(1)(b)(2) (“attributes”): INTEL-1005, ¶68 INTEL-1006, ¶¶42, 48

	INTEL-1010, 5 INTEL-1008, 26 INTEL-1011, 3
[45D] said means for addressing obtains the first set of attributes after identifying the command details; and	§IV(B)(4)(b) (“means for addressing’ limitations”) INTEL-1005, ¶¶78, 56 §IV(B)(3)(b) (“interface controller”): INTEL-1005, ¶¶55-56, 69, Fig. 5 §IV(B)(1)(d)(3) (“obtaining the first set of attributes [and] using attributes from said first and second device profiles”): INTEL-1005, ¶¶51, 56, 70, 73, 77
[45E] said means for addressing uses attributes from said first and second device profiles and selects a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.	§IV(B)(4)(b) (“means for addressing’ limitations”) INTEL-1005, ¶¶78, 56 §IV(B)(3)(b) (“interface controller”): INTEL-1005, ¶¶55-56, 69, Fig. 5 §IV(B)(1)(d)(3) (“obtaining the first set of attributes [and] using

	attributes from said first and second device profiles”): INTEL-1005, ¶¶51, 56, 70, 73, 77 §IV(B)(1)(d)(4) (“selecting a transfer size for the memory transaction”): INTEL-1005, ¶73, Fig. 2
--	--

5. Independent Claim 80 (Electronic Storage Device)

227. I have reproduced independent claim 80 below in its entirety with labels for each limitation reflecting the limitations place in the claim and functional grouping.

[80P] An electronic storage device, comprising:

[80A] a memory controller directly coupled to and disposed to **[action]** address a first memory location and a second memory location **[profile]** which are respectively associated with a first device profile and a second device profile;

[80A.1] said memory controller directly coupled to a host **[action]** as said memory controller addresses the first memory location or second memory location;

[80A.2/Profile] wherein said first device profile is optimal for a data type subject to a memory transaction, wherein said data type comprises one of a random data type or a sequential data type;

[80A.3/Action] said memory controller identifies command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;

[80A.4/Profile] said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles;

[80A.5/Action] said memory controller obtaining the first set of attributes after identifying the command details; and

[80A.6/Action] said memory controller using attributes from said first and second device profiles; and

[80A.7/Action] said memory controller selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.

a) Preamble [80P]

228. The combination of Danilak and Cornwell discloses an “*electronic storage device*” [80P]. Danilak is directed to a system and method for increasing capacity, performance, and flexibility of flash storage. (INTEL-1005, Title.)

Danilak discloses its system can be implemented as a “stacked package” with “the flash memory devices on top of a flash interface circuit.” (INTEL-1005, ¶52.)

Danilak further teaches flash memory devices are used in products such as USB

storage and flash hard drives. (INTEL-1005, ¶3.) The stacked package disclosed in Danilak is an “*electronic storage device*.”

b) “Memory Controller”

229. Claim 1 recites “[80A] *a memory controller directly coupled to and disposed to [action] address a first memory location and a second memory location [profile] which are respectively associated with a first device profile and a second device profile*” and “[80A.1] *said memory controller directly coupled to a host [action] as said memory controller addresses the first memory location or second memory location.*” I address the structural limitations in this section and the action and profile limitations in the sections that follow. The combination of Danilak and Cornwell discloses a “*memory controller*” as I discussed in §§IV(B)(1)(a)(4) and IV(B)(3)(a). Danilak’s flash interface circuit (“*memory controller*”) in the combined system is coupled to a memory store having multiple memory devices, each with a set of addressable “*memory locations.*” (§§IV(B)(1)(a)(2), §IV(B)(1)(a)(3), IV(B)(1)(a)(5).) Thus, the combination of Danilak and Cornwell discloses “*a memory controller directly coupled to ... a first memory location and a second memory location*” [80A].

230. Danilak’s flash interface circuit is coupled to the host, as illustrated in Figure 2, and discussed in §§IV(B)(1)(a)(6) and IV(B)(1)(e)(1). Thus, the

combination of Danilak and Cornwell also discloses “*the memory controller is directly coupled to the host ...*.”

c) “Device Profile” Limitations

231. The profile group of “*wherein*” clauses of claim 80 are substantially identical to the profile group of “*wherein*” clauses of claim 1. The only differences are non-material. Thus, for the reasons I discuss in §§IV(B)(1)(b)(1)-(2) and IV(B)(1)(b)(4) for limitations [1A.1/Profile], [1A.6/Profile], and [1B.1/profile], the combination of Danilak and Cornwell also discloses limitations [80A], [80A.2/Profile], and [80A.4/Profile].

Claim 1	Claim 80
[1A.1/Profile] said first memory location and a second memory location respectively associated with a first device profile and a second device profile,	[80A] a memory controller directly coupled to and disposed to address <u>a</u> first memory location and a second memory location <u>which are</u> respectively associated with a first device profile and a second device profile,
[1A.6/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;	[80A.2/Profile] wherein said first device profile is optimal for a data type subject to <u>a</u> memory transaction, wherein said data type comprises one of a random data type or a sequential data type;
[1B.1/Profile] said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference	[80A.4/Profile] said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory

exists between said first and second device profiles;	location, and a difference exists between said first and second device profiles;
---	--

d) Action Limitations

232. Claim 80 recites a set of actions performed by the interface controller. These actions are substantially the same as the set of actions recited in claim 1 with the following few differences. First, limitation [80A] requires the memory controller to address “*a second memory location*” in addition to the first memory location. This is disclosed by the combination of Danilak and Cornwell. As discussed in §IV(B)(1)(d)(3) Danilak describes an example of a translation in which “the configured page size for the virtual memory reported to the host is larger than the discovered page size[s]” for the physical memory devices. (*See* INTEL-1005, ¶73.) This difference results in a memory transaction request from a host specifying a data size consistent with the virtual page size but larger than can be handled by a single memory device. To account for this difference, Danilak’s translation units 508/509 are configured to translate a host write request with a large data size into multiple write operations of smaller sizes. As explained by Danilak, in this circumstance, the memory transaction requested by the host is “performed to **multiple flash memory devices** to form a page of the configured size.” (INTEL-1005, ¶73.) Thus, in this example, Danilak’s translation units issue a memory operation to a location in a first memory device, a different memory

operation to a location in a second memory device, and so on until the required page size is achieved. Danilak’s flash interface circuit (*memory controller*”) is therefore also disposed “*to address ... a second memory location.*” Second, limitation [80A.1] requires the host to be coupled to the memory controller as the addressing occurs. As I discussed in §IV(B)(5)(b) Danilak’s flash interface circuit is directly coupled to the host and communicates address, control, clock and data signals with the host over a connection, illustrated in Figure 2. When the host requests a memory access (which requires addressing by the memory controller), translation units 508/509 “return an appropriate response to the request to the host” over the connection. (INTEL-1005, ¶78.) The combination of Danilak and Cornwell discloses the “*memory controller [is] directly coupled to a host as [said]/[the] memory controller addresses the first memory location or [the] second memory location*” [80A.1]/[98A.1]. Thus, for the reasons I discuss in §§IV(B)(1)(d)(1)-(4) for limitations [1A]-[1E], the combination of Danilak and Cornwell also discloses limitations [80A], [80A.3/Action], [80A.5/Action], [80A.6/Action] and [80A.7/Action].

Claim 1	Claim 80
[1A] performing a memory transaction comprising addressing a first memory location in a first memory device in a memory store:	[80A] a memory controller directly coupled to and disposed to address a first memory location and a second memory location ...

	[80A.1] said memory controller directly coupled to a host [action] as said memory controller addresses the first memory location or second memory location
[1B] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;	[80A.3/Action] said memory controller identifies command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device
[1C] after identifying the command details, obtaining the first set of attributes;	[80A.5/Action] said memory controller obtaining the first set of attributes after identifying the command details
[1D] using attributes from said first and second device profiles; and	[80A.6/Action] said memory controller using attributes from said first and second device profiles
[1E] selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.	[80A.7/Action] said memory controller selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.

e) Summary

233. In summary, following table presents the preamble and each limitation in order, the section or sections addressing the limitation, and the evidence cites.

Claim 80	
-----------------	--

[80P] An electronic storage device, comprising:	§IV(B)(5)(a) (“Preamble [80p]”) INTEL-1005, Title, ¶¶ 3, 52
[80A] a memory controller directly coupled to and disposed to address a first memory location and a second memory location which are respectively associated with a first device profile and a second device profile;	§IV(B)(5)(b) (“memory controller”) §IV(B)(3)(a) (“Preamble [59P]”): INTEL-1005, ¶54, Fig. 4 §IV(B)(1)(a)(4) (“controller”): INTEL-1005, ¶¶14, 22-42, 51, 56, 78, 80, Fig. 2, Fig. 10 INTEL-1006, ¶¶7, 19, 24, 53-55, Fig. 1 INTEL-1010, 27 INTEL-1008, 58 §IV(B)(1)(a)(2) (“memory location”): INTEL-1005, ¶¶22, 24-25, 69, claim 7, Fig. 2 INTEL-1006, ¶¶3-4 INTEL-1008, 66-67, Fig. 3.2 INTEL-1007, 2 §IV(B)(5)(c) (“‘device profile’ limitations”) §IV(B)(1)(b)(1) (“device profile”):

	INTEL-1005, ¶¶59, 68, Fig. 9 INTEL-1006, ¶¶42, Fig. 3 INTEL-1007, 91-106, Table 38 §IV(B)(5)(d) (“action limitations”) §IV(B)(1)(d)(1) (“addressing a first memory location...”): INTEL-1005, ¶¶68, 72, 77 INTEL-1006, ¶¶25-27, 29, 35, 37 INTEL-1007, 2
[80A.1/Action] said memory controller directly coupled to a host as said memory controller addresses the first memory location or second memory location;	§IV(B)(5)(b) (“memory controller”): INTEL-1005, Fig. 2 §IV(B)(1)(a)(6) (“communication path”): INTEL-1005, ¶¶18, 22, Fig. 1, Fig. 2 INTEL-1006, ¶21 §IV(B)(1)(e)(1) (“controller performs the memory read or memory write transaction on a memory location”): INTEL-1005, ¶22, 78, Fig. 2

[80A.2/Profile] wherein said first device profile is optimal for a data type subject to a memory transaction, wherein said data type comprises one of a random data type or a sequential data type;	§IV(B)(5)(c) (“device profile’ limitations”) §IV(B)(1)(b)(4) (“device profile is optimal for a data type subject to the memory transaction”): INTEL-1005, ¶¶56-60, 57-68, Fig. 5, Fig. 9 INTEL-1006, ¶42 INTEL-1008, 23 INTEL-1009, 2 INTEL-1019, 4-5 INTEL-1020, 3-14
[80A.3/Action] said memory controller identifies command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;	§IV(B)(5)(d) (“action limitations”) §IV(B)(1)(d)(2) (“identifying command details”): INTEL-1005, ¶¶56, 70-72 INTEL-1006, ¶¶34-38 INTEL-1007, 111-112, 117-118
[80A.4/Profile] said first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, and a difference exists between said first and second device profiles;	§IV(B)(5)(c) (“device profile’ limitations”) §IV(B)(1)(b)(2) (“attributes”): INTEL-1005, ¶68 INTEL-1006, ¶¶42, 48 INTEL-1010, 5 INTEL-1008, 26

	INTEL-1011, 3
[80A.5/Action] said memory controller obtaining the first set of attributes after identifying the command details; and	§IV(B)(5)(d) (“action limitations”) §IV(B)(1)(d)(3) (“obtaining the first set of attributes [and] using attributes from said first and second device profiles”): INTEL-1005, ¶¶51, 56, 70, 73, 77
[80A.6/Action] said memory controller using attributes from said first and second device profiles; and	§IV(B)(5)(d) (“action limitations”) §IV(B)(1)(d)(3) (“obtaining the first set of attributes [and] using attributes from said first and second device profiles”): INTEL-1005, ¶¶51, 56, 70, 73, 77
[80A.7/Action] said memory controller selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.	§IV(B)(5)(d) (“action limitations”) §IV(B)(1)(d)(4) (“selecting a transfer size for the memory transaction”): INTEL-1005, ¶73, Fig. 2

6. Independent Claim 98

234. Independent claim 98 is substantially identical to independent claim 80.

I have reproduced claims 80 and 98 below with differences indicated with underline. The sole significant difference between the claims is in limitations [80A.4/Profile] and [98A.4/Profile]. Limitation [80A.4/Profile] includes the limitation “*a difference exists between said first and second device profiles*” which is not present in limitation [98A.4/Profile]. Thus, limitation [98A.4/Profile] is broader than limitation [80A.4/Profile] and is disclosed by the combination of Danilak and Cornwell for the same reason as limitation [80A.4/Profile].

Accordingly, for the reasons I discussed in §§IV(B)(5)(a)-(d) for [80P], [80A], and [80A.1]-[80A.7], the preamble [98P] and limitations [98A], and [98A.1]-[98A.7] are also disclosed by Danilak and Cornwell.

Claim 80	Claim 98
[80P] An electronic storage device, comprising:	[98P] An electronic storage device, comprising:
[80A/Action] a memory controller directly coupled to and disposed to address a first memory location and a second memory location which are respectively associated with a first device profile and a second device profile;	[98A/Action] a memory controller directly coupled to and disposed to address a first memory location and a second memory location which are respectively associated with a first device profile and a second device profile;
[80A.1/Action] <u>said</u> memory controller directly coupled to a host as <u>said</u> memory controller addresses the	[98A.1/Action] <u>wherein the</u> memory controller <u>is</u> directly coupled to a host as <u>the</u> memory controller addresses the

first memory location or second memory location;	first memory location or <u>the</u> second memory location;
[80A.2/Profile] wherein said first device profile is optimal for a data type subject to a memory transaction, wherein said data type comprises one of a random data type or a sequential data type;	[98A.2/Profile] wherein said first device profile is optimal for a data type subject to a memory transaction, wherein said data type comprises one of a random data type or a sequential data type;
[80A.3/Action] said memory controller identifies command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;	[98A.3/Action] said memory controller identifies command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;
[80A.4/Profile] <u>said</u> first device profile representing a first set of attributes of <u>said</u> first memory location, and <u>said</u> second device profile representing a second set of attributes of <u>said</u> second memory location, <u>and a difference exists between said first and second device profiles</u> ;	[98A.4/Profile] <u>the</u> first device profile representing a first set of attributes of <u>the</u> first memory location, and <u>the</u> second device profile representing a second set of attributes of <u>the</u> second memory location;
[80A.5/Action] said memory controller obtaining the first set of attributes after identifying the command details; and	[98A.5/Action] said memory controller obtaining the first set of attributes after identifying the command details; and
[80A.6/Action] said memory controller <u>using</u> attributes from <u>said</u> first and second device profiles; and	[98A.6/Action] <u>wherein</u> said memory controller <u>uses</u> attributes from <u>the</u> first and second device profiles;
[80A.7/Action] said memory controller selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size	[98A.7/Action] <u>wherein</u> said memory controller selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size

of the memory transaction and the first set of attributes	of the memory transaction and the first set of attributes.
---	--

7. Independent Claim 100 (Method)

235. Independent claim 100 has substantially overlapping subject matter with claim 1. I have reproduced the limitations of claim 100 below.

[100P] A method for performing memory transactions, the method comprising:

[100A] performing a memory transaction comprising addressing a first memory location **[structural]** in a first memory device in a memory store,

[100A.1/Profile] the first memory location and a second memory location respectively associated with a first device profile and a second device profile;

[100A.2/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;

[100B] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;

[100B.1/Profile] the first device profile representing a first set of attributes of the first memory location, and the second device profile representing a second set of attributes of the second memory location;

[100C] after identifying the command details, obtaining the first set of attributes;

[100C.1/Structural] wherein the memory store is directly coupled to a memory controller, and

[100C.2/Transaction] wherein the memory controller performs a memory transaction;

[100C.3/Structural] wherein a host is directly coupled to the memory controller [transaction] as the memory controller performs the memory transaction; and

[100D] using attributes from the first and second device profiles; and

[100E] selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.

a) Preamble

236. The preamble of claim 100 recites a “*method of performing memory transactions*” which is also recited in the preamble of claim 1 as shown in the table below. For the reasons I discussed in §IV(B)(1)(c), the combination of Danilak and Cornwell also discloses the preamble of claim 100.

Claim 1	Claim 100
<u>A method of performing memory transactions</u> on a set of memory locations that includes a first memory location and a second memory location, <u>the method comprising</u>	A method for performing memory transactions, the method comprising

b) Structural Limitations

237. Claim 100 recites overlapping structural limitations with claim 1. There are 4 differences, each of which makes claim 100 broader than claim 1. First, limitation [1C.2] recites the memory store is coupled to a memory bus which is coupled to the controller where limitation [100C.1] recites the memory store is coupled to the memory controller. Therefore, the combination of Danilak and Cornwell discloses [100C.1] for the same reason as limitation [1C.2]. Second, limitation [1C.4] recites that the host is coupled to the controller by a communication path and [100C.3] does not recite a communication path. Third, limitations [1C.3] and [1C.4] recite “memory read transaction [and/or] memory write transaction” whereas limitations [100C.2] and [100C.3] recite broadly “the memory transaction.” Limitations [100C.2] and [100C.3] are broader than limitation [1C.3] and [1C.4] and disclosed for the same reason as [1C.3] and [1C.4]. Finally, claim 100 recites a “memory controller” whereas claim 1 recites “a controller.” These are equivalent terms. Accordingly, for the reasons discussed in §§IV(B)(1)(a)(1)-(6), the combination of Danilak and Cornwell renders claim 100 obvious.

Claim 1	Claim 100
[1A] performing a memory transaction comprising addressing a first memory	[100A] performing a memory transaction comprising addressing a

location in a first memory device in a memory store,	first memory location in a first memory device in a memory store,
[1C.2/Structural] wherein the memory store is directly coupled <u>to at least one memory bus and wherein the at least one memory bus is directly coupled</u> to a controller, and	[100C.1/Structural] wherein the memory store is directly coupled to a <u>memory controller</u> , and
[1C.4/Structural] wherein the host is directly coupled to the controller <u>by a communication path</u> as the controller performs <u>the memory read transaction or memory write transaction</u>	[100C.3/Structural] wherein a host is directly coupled to the <u>memory controller</u> as the <u>memory controller</u> performs the <u>memory transaction</u> ;

c) “Device Profile” Group

238. The device profile limitations of claim 29 are substantially identical to the device profile limitations of claim 1. The primary difference is in limitations [1B.1] and [100B.1]. Limitation [1B.1] recites “*a difference exists between said first and second device profiles*” which is not present in limitation [100B.1]. Thus, limitation [100B.1] is broader than limitation [1B.1] and is met for the same reason as limitation [1B.1]. Thus, for the reasons I discuss in §§IV(B)(1)(b)(1)-(2) and IV(B)(1)(b)(4) for limitations [1A.1/Profile], [1A.6/Profile], and [1B.1/profile], the combination of Danilak and Cornwell also discloses limitation [100A.1/Profile], [100A.2/Profile], and [100B.1/Profile].

Claim 1	Claim 100
[1A.1/Profile] <u>said</u> first memory location and a second memory location respectively associated with a first	[100A.1/Profile] <u>the</u> first memory location and a second memory location respectively associated with a

device profile and a second device profile,	first device profile and a second device profile;
[1A.6/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;	[100A.2/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;
[1B.1/Profile] <u>said</u> first device profile representing a first set of attributes of said first memory location, and said second device profile representing a second set of attributes of said second memory location, <u>and a difference exists between said first and second device profiles;</u>	[100B.1/Profile] <u>the</u> first device profile representing a first set of attributes of the first memory location, and <u>the</u> second device profile representing a second set of attributes of the second memory location;

d) Method Steps and Transaction Limitations

239. The action limitations recited in independent claim 100 are identical to the action limitations of claim 1. Thus, for the reasons I discuss in §§IV(B)(1)(d)(1)-(4) for limitations [1A], [1B], [1C], [1D], and [1E], the combination of Danilak and Cornwell also discloses limitation [100A], [100B], [100C], [100D], and [100E].

240. Claim 100 also recites two transactional limitations: [100C.2] recites “wherein the memory controller performs a memory transaction and [100C.3] recites “the memory controller performs the memory transaction.” The transaction limitations of claim 100 are substantially identical to the read-write limitations of

claim 1. There are two primary differences. The first difference is that limitations [1C.3] and [1C.4] recites “*a controller*” while limitations [100C.2] and [100C/3] recite “a memory controller.” As discussed above in §IV(B)(7)(b), these are equivalent terms. Second, limitations [1C.3] and [1C.4] recites “the memory read transaction and[/or] memory write transaction” while limitations [100C.2] and [100C.3] recite “a memory transaction.” Thus, limitations [100C.2] and [100C.3] are broader than corresponding limitations [1C.3] and [1C.4] and met for the same reason as limitation limitations [1C.3] and [1C.4]. Thus, for the reasons I discuss in §IV(B)(1)(e)(1) for limitations [1C.3] and [1C.4], the combination of Danilak and Cornwell also discloses limitation [100C.2/Transaction] [and 100C.3/Transaction].

Claim 1	Claim 100
[1A] performing a memory transaction comprising addressing a first memory location in a first memory device in a memory store,	[100A] performing a memory transaction comprising addressing a first memory location in a first memory device in a memory store,
[1B] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;	[100B] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;
[1C] after identifying the command details, obtaining the first set of attributes;	[100C] after identifying the command details, obtaining the first set of attributes;

[1C.3/Read-Write] wherein the controller performs <u>the memory read transaction and memory write transaction on the memory store</u>	[100C.2/Transaction] wherein the <u>memory</u> controller performs <u>a</u> memory transaction;
[1C.4/Read-Write] wherein <u>the host is directly coupled to the controller by a communication path</u> as the controller performs the <u>memory read transaction or memory write transaction</u>	[100C.3/Transaction] wherein a host is directly coupled to the <u>memory controller</u> as the <u>memory controller</u> performs the <u>memory transaction</u> ;
[1D] using attributes from said first and second device profiles; and	[100D] using attributes from the first and second device profiles; and
[1E] selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.	[100E] selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.

e) Summary

Claim 100	
[100P] A method for performing memory transactions, the method comprising:	§IV(B)(7)(a) (“preamble”) §IV(B)(1)(a)(2) (“memory location”): INTEL-1005, ¶¶22, 24-25, 69, claim 7, Fig. 2 INTEL-1006, ¶¶3-4 INTEL-1008, 66-67, Fig. 3.2 INTEL-1007, 2 §IV(B)(1)(c) (“Preamble [1p]”):

	INTEL-1005, ¶¶33-40, 68, 72, 78, Fig. 2 INTEL-1006, ¶¶25-27 INTEL-1018, 582 INTEL-1007, 2-3, 111-112, 117-118
[100A] performing a memory transaction comprising addressing a first memory location [structural] in a first memory device in a memory store;	§IV(B)(7)(d) (“method steps and transaction limitations”) §IV(B)(1)(a)(1) (“memory device”): INTEL-1005, ¶¶17-18, 22, 22 §IV(B)(1)(a)(2) (“memory location”): INTEL-1005, ¶¶22, 24-25, 69, claim 7, Fig. 2 INTEL-1006, ¶¶3-4 INTEL-1008, 66-67, Fig. 3.2 INTEL-1007, 2 §IV(B)(1)(a)(3) (“memory store”): INTEL-1005, ¶¶51, 54, 80, Fig. 4 INTEL-1017, 89 §IV(B)(1)(d)(1) (“addressing a first memory location...”): INTEL-1005, ¶¶68, 72, 77 INTEL-1006, ¶¶25-27, 29, 35, 37 INTEL-1007, 2

[100A.1/Profile] the first memory location and a second memory location respectively associated with a first device profile and a second device profile;	§IV(B)(7)(c) (“device profile’ group”) §IV(B)(1)(b)(1) (“device profile”): INTEL-1005, ¶¶59, 68, Fig. 9 INTEL-1006, ¶¶42, Fig. 3 INTEL-1007, 91-106, Table 38
[100A.2/Profile] wherein said first device profile is optimal for a data type subject to the memory transaction, wherein said data type comprises one of a random data type or a sequential data type;	§IV(B)(7)(c) (“device profile’ group”) §IV(B)(1)(b)(4) (“device profile is optimal for a data type subject to the memory transaction”): INTEL-1005, ¶¶56-60, 57-68, Fig. 5, Fig. 9 INTEL-1006, ¶42 INTEL-1008, 23 INTEL-1009, 2 INTEL-1019, 4-5 INTEL-1020, 3-14
[100B] said performing the memory transaction further comprising identifying command details for causing the memory transaction to be performed, wherein said command details comprising the first memory device;	§IV(B)(7)(d) (“method steps and transaction limitations”) §IV(B)(1)(d)(2) (“identifying command details”): INTEL-1005, ¶¶56, 70-72

	INTEL-1006, ¶¶34-38 INTEL-1007, 111-112, 117-118
[100B.1/Profile] the first device profile representing a first set of attributes of the first memory location, and the second device profile representing a second set of attributes of the second memory location;	§IV(B)(7)(c) (“‘device profile’ group”) §IV(B)(1)(b)(2) (“attributes”): INTEL-1005, ¶68 INTEL-1006, ¶¶42, 48 INTEL-1010, 5 INTEL-1008, 26 INTEL-1011, 3
[100C] after identifying the command details, obtaining the first set of attributes;	§IV(B)(7)(d) (“method steps and transaction limitations”) §IV(B)(1)(d)(3) (“obtaining the first set of attributes [and] using attributes from said first and second device profiles”): INTEL-1005, ¶¶51, 56, 70, 73, 77
[100C.1/Structural] wherein the memory store is directly coupled to a memory controller, and	§IV(B)(7)(b) (“structural limitations”) §IV(B)(1)(a)(4) (“controller”): INTEL-1005, ¶¶14, 22-42, 51, 56, 78, 80, Fig. 2, Fig. 10 INTEL-1006, ¶¶7, 19, 24, 53-55, Fig. 1

	INTEL-1010, 27 INTEL-1008, 58 §IV(B)(1)(a)(5) (“memory bus”): INTEL-1005, ¶¶18, 22, 41, 80, Fig. 2 INTEL-1006, ¶¶24, 34 INTEL-1017, 13, 57 INTEL-1008 at 67
[100C.2/Transaction] wherein the memory controller performs a memory transaction;	§IV(B)(7)(d) (“method steps and transaction limitations”) §IV(B)(1)(e)(1) (“controller performs the memory read or memory write transaction on a memory location”): INTEL-1005, ¶¶29, 31, 33, 35 INTEL-1006, ¶24 §IV(B)(1)(c) (“Preamble [1p]”): INTEL-1005, ¶¶33- 40, 68, 72, 78, Fig. 2 INTEL-1006, ¶¶25- 27, 31, 35, 37 INTEL-1018, 582 INTEL-1007, 2-3, 111-112, 117-118 §IV(B)(1)(d)(1) (“addressing a first memory location...”):

	INTEL-1005, ¶¶68, 72, 77 INTEL-1006, ¶¶25-27, 29, 35, 37 INTEL-1007, 2
[100C.3/Structural] wherein a host is directly coupled to the memory controller as the memory controller performs the memory transaction; and	§IV(B)(7)(b) (“structural limitations”) §IV(B)(7)(d) (“method steps and transaction limitations”) §IV(B)(1)(a)(6) (“communication path”): INTEL-1005, ¶¶18, 22, Fig. 1, Fig. 2 INTEL-1006, ¶21 §IV(B)(1)(e)(1) (“controller performs the memory read or memory write transaction on a memory location”): INTEL-1005, ¶¶22, 78, Fig. 2 §IV(B)(1)(d)(4) (“selecting a transfer size for the memory transaction”): INTEL-1005, ¶73, Fig. 2
[100D] using attributes from the first and second device profiles; and	§IV(B)(7)(d) (“method steps and

	transaction limitations”) §IV(B)(1)(d)(3) (“obtaining the first set of attributes [and] using attributes from said first and second device profiles”): INTEL-1005, ¶¶51, 56, 70, 73, 77
[100E] selecting a transfer size for the memory transaction, wherein the transfer size is a function of a data size of the memory transaction and the first set of attributes.	§IV(B)(7)(d) (“method steps and transaction limitations”) §IV(B)(1)(d)(4) (“selecting a transfer size for the memory transaction”): INTEL-1005, ¶73, Fig. 2

C. Dependent Claims

1. Claims 2, 33, 46, and 82

241. Claims 2, 33, 46 and 82 share identical subject matter. I reproduce each claim below for ease of discussion.

2. The method of claim 1, wherein said first memory location is located in a first memory device, and said second memory location is located in a second memory device.

33. The computer readable medium of claim 29, wherein said first memory location is located in a first memory device and said second memory location is located in a second memory device.

46. The memory controller of claim 45, wherein said first memory location is located in a first memory device and said second memory location is located in a second memory device.

82. The electronic storage device of claim 80, wherein said first memory location is located in a first memory device and said second memory location is located in a second memory device.

242. I note that these claims merely restate and combine limitations from claim 1—“*first memory location in a first memory device*” [1A] and “*second memory location is in a second memory device*” [1A.2]. For the reasons I discussed in §§IV(B)(1)(a)(1)-(2), the combination of Danilak and Cornwell discloses the subject matter of claims 2, 33, 46, and 82.

2. Claims 5, 36, 51, 71, and 84

243. Claims 5, 36, 51, 71, and 84 share identical subject matter. I reproduce each claim below for ease of discussion.

5. The method of claim 1, wherein said difference includes different attribute qualifiers for at least one set of shared attributes from said first and second device profiles.

36. The computer readable medium of claim 29, wherein said difference includes different attribute qualifiers for at least one set of shared attributes from said first and second device profiles.

51. The memory controller of claim 45, wherein said difference includes different attribute qualifiers for at least one set of shared attributes from said first and second device profiles.

71. The memory controller of claim 59, wherein said difference includes different attribute qualifiers for at least one set of shared attributes from said first and second device profiles.

84. The electronic storage device of claim 80, wherein said difference includes different attribute qualifiers for at least one set of shared attributes from said first and second device profiles.

244. In the combination of Danilak and Cornwell, each memory device has a device profile which specifies the same set of attributes for each memory device, as I discussed in §§IV(B)(1)(b)(1). Cornwell teaches that the device data stored in the non-volatile memory includes identification of the memory device and its manufacturer, device type (e.g., NAND, NOR, etc.), device density (e.g., 512 Mb, 1 Gb, 2 Gb, etc.), device operating voltage (e.g., 3.3 volts), page size (1 k, 2K, etc.), spare area size (e.g., 8, 16 bytes, etc.), sequential access time (e.g., 30, 50 nanoseconds, etc.), block size (e.g., 64 k, 128 k, etc.), bus width (e.g., ×8, ×16, etc.), bad block identification, and any other information that is associated with attributes, properties or characteristics of the memory device 106.” (INTEL-1006,

¶42.) The device profile also includes configuration data disclosed by Danilak, and illustrated in Figure 9 below, including ONFI support, cell type, number of simultaneously programmed pages, cache program, etc.) (INTEL-1005, ¶59, Figure 9.) The attributes included in each data profile are “*shared attributes.*”

900

Byte	Bits	Mnemonic	Description
0	0	–	Reserved
	1	ONFI	ONFI standard support (No, Yes)
	3:2	CHIPS	Number of stacked chips (1, 2, 4, 8)
	7:4	–	Reserved
1	8	MKR	External maker code reported by flash interface circuit to host.
2	8	DEV	External device code reported by flash interface circuit to host.
3	1:0	CIPN	Internal chip number (1, 2, 4, 8)
	3:2	CELL	Cell type (2, 4, 8, 16 levels)
	5:4	SIMP	# of simultaneously programmed pages (1, 2, 4, 8)
	6	INTRL	Interleave between multiple chips (No, Yes)
	7	CACHE	Cache program (No, Yes)
4	1:0	PAGE	Page size (1, 2, 4, 8 KB)
	2	RSIZE	Redundant area size (8, 16 bytes / 512 bytes)
	7&3	SMIN	Serial access minimum (50/30, 25 ns)
	5:4	BSIZE	Block size (64, 128, 256, 512 KB)
	6	ORG	Organization (x8, x16)
5	1:0	–	Reserved
	3:2	PLANE	Plane number
	6:4	PLSIZE	Plane size (64, 128, 256, 512, 1Gb, 2Gb, 4Gb, 8Gb)
	7	–	Reserved
6	7:0	–	Reserved

FIG. 9
Danilak, Figure 9

245. Both Danilak and Cornwell teach that each “*attribute*” has an “*attribute qualifier*” (numerical value or non-numerical type). The “*attribute qualifiers*” vary

for memory devices in the combination of Danilak and Cornwell. (See INTEL-1005, ¶68 (“various attached flash memory devices may have differing attributes”).) The combination includes both NOR Flash devices and NAND Flash devices. The device profile for a NOR Flash device has different “*attribute qualifiers*” than the device profile for a NAND Flash device.

246. As I discussed in §IV(B)(1)(b)(1), a difference exists between device profiles of the memory devices in the combined system of Danilak and Cornwell. Brewer recognized that “Flash memory possesses its own set of attributes that define the product category and each specific product implements those attributes in a slightly different way.” (INTEL-1008, 26.) The device profiles for NAND and NOR devices differ because the devices have different values and types for specified attributes (e.g., type, page size, access time, etc.). For example, the non-numerical qualifiers for the device type attribute (NAND or NOR) and ONFI support differ between NAND and NOR devices. The numerical qualifiers for page size, block size, access time, and device density also differ between NOR and NAND devices. (See, e.g., INTEL-1011, 3 (noting differences between NAND and NOR devices); INTEL-1010, 5 (noting differences between NAND and NOR devices); INTEL-1023, 70, 73, 75 (device can be queried to return page and block size).) Cornwell also teaches, for example, that “[f]lash is available in a variety of block sizes.” (INTEL-1006, ¶48.) This teaching indicates that the attached memory

devices have varying qualifiers for the block size attribute. And, I note Toshiba indicates that the number of levels and density also varies within NAND devices. (See INTEL-1011, 3 (listing density for SLC and MLC NAND devices))

247. Thus, combination of Danilak and Cornwell discloses that “*said difference includes different attribute qualifiers for at least one set of shared attributes from said first and second device profiles*” [5]/[36]/[51]/[71]/[84].

3. Claims 6, 37, 52, 76, and 85

248. Claims 6, 37, 52, 76, and 85 share identical subject matter. I reproduce each claim below for ease of discussion.

6. The method of claim 5, wherein said different attribute qualifiers include different attribute values.

37. The computer readable medium of claim 36, wherein said different attribute qualifiers include different attribute values.

52. The memory controller of claim 51, wherein said different attribute qualifiers include different attribute values.

76. The memory controller of claim 71, wherein said different attribute qualifiers include different attribute values.

85. The electronic storage device of claim 84, wherein said different attribute qualifiers include different attribute values.

249. As I mentioned in the previous section, the combination includes both NOR Flash devices and NAND Flash devices. The device profile for a NOR Flash

device has different “*attribute qualifiers*” than the device profile for a NAND Flash device. The numerical qualifiers for page size, block size, access time, and device density also differ between NOR and NAND devices. (See, e.g., INTEL-1011, 3 (noting differences between NAND and NOR devices); INTEL-1010, 5 (Table 5 below noting differences between NAND and NOR devices).)

Characteristic	NAND Flash: MT29F2G08A	NOR Flash: MT28F128J3
Random access READ	25μs (first byte) 0.03μs each for remaining 2,111 bytes	0.12μs
Sustained READ speed (sector basis)	23 MB/s (x8) or 37 MB/s (x16)	20.5 MB/s (x8) or 41 MB/s (x16)
Random WRITE speed	≈300μs/2,112 bytes	180μs/32 bytes
Sustained WRITE speed (sector basis)	5 MB/s	0.178 MB/s
Erase block size	128KB	128KB
ERASE time per block (TYP)	2ms	750ms

250. Toshiba highlights the differences in density and access time between NAND and NOR devices in Figure 4 below. And I note Toshiba indicates that the number of levels and density also varies within NAND devices. (See INTEL-1011, 3 (Figure listing density for SLC and MLC NAND devices, density for NOR device as well as access times).)

	SLC NAND Flash (x8)	MLC NAND Flash (x8)	MLC NOR Flash (x16)
Density	512 Mbits ¹ – 4 Gbits ²	1Gbit to 16Gbit	16Mbit to 1Gbit
Read Speed	24 MB/s ³	18.6 MB/s	103MB/s
Write Speed	8.0 MB/s	2.4 MB/s	0.47 MB/s
Erase Time	2.0 mSec	2.0mSec	900mSec
Interface	I/O – indirect access	I/O – indirect access	Random access
Application	Program/Data mass storage	Program/Data mass storage	eXecuteInPlace

Figure 4: NAND and NOR Flash Operating Specifications

251. Thus, the combination of Danilak and Cornwell discloses “*said different attribute qualifiers* [e.g., page size, block size, access time, and density] *include different attribute values*” [6]/[37]/[52]/[76]/[85].

4. Claims 7, 38, 53, 77, and 87

252. Claims 7, 38, 53, 77, and 87 share identical subject matter. I reproduce each claim below for ease of discussion.

7. The method of claim 5, wherein said different attribute qualifiers include different attribute types.

38. The computer readable medium of claim 36, wherein said different attribute qualifiers include different attribute types.

53. The memory controller of claim 51, wherein said different attribute qualifiers include different attribute types.

77. The memory controller of claim 71, wherein said different attribute qualifiers include different attribute types.

87. The electronic storage device of claim 84, wherein said different attribute qualifiers include different attribute types.

253. As I mentioned in the previous section, the combination includes both NOR Flash devices and NAND Flash devices. The device profile for a NOR Flash device has different “*attribute qualifiers*” than the device profile for a NAND Flash device. For example, the non-numerical qualifiers for the device type attribute (NAND or NOR) and ONFI support (Yes or No) differ. Accordingly, the combination of Danilak and Cornwell discloses “*said different attribute qualifiers [e.g., device type and ONFI support] include different attribute types*” [7]/[38]/[53]/[77]/[87].

5. Claims 8, 39, 54, 72, and 86

254. Claims 8, 39, 54, 72, and 86 share identical subject matter. I reproduce each claim below for ease of discussion.

8. The method of claim 5, wherein said shared attributes include a memory device type attribute.

39. The computer readable medium of claim 36, wherein said shared attributes include a memory device type attribute.

54. The memory controller of claim 51, wherein said shared attributes include a memory device type attribute.

72. The memory controller of claim 71, wherein said shared attributes includes a memory device type attribute.

86. The electronic storage device of claim 84, wherein said shared attributes include a memory device type attribute.

255. Cornwell teaches that the device data stored in each non-volatile memory includes the device type (e.g., NAND, NOR, etc.). (See INTEL-1006, ¶42.) Because this attribute is present in each device profile, it is a “*shared attribute*.” The combination of Danilak and Cornwell therefore discloses that “*said shared attributes include a memory device type attribute*” [8]/[39]/[54]/[72]/[86].

6. Claims 11, 42, 57, and 92

256. Claims 11, 42, 57, and 92 share identical subject matter. I reproduce each claim below for ease of discussion.

11. The method of claim 5, wherein said shared attributes includes any one of the following memory device attributes: partial page size, page size, block size, memory protocol, and memory device type.

42. The computer readable medium of claim 36, wherein said shared attributes includes any one of the following memory device attributes: partial page size, page size, block size, memory protocol, and memory device type.

57. The memory controller of claim 51, wherein said shared attributes includes any one of the following memory device attributes: partial page size, page size, block size, memory protocol, and memory device type.

92. The electronic storage device of claim 84, wherein said shared attributes includes any one of the following memory device attributes: partial page size, page size, block size, memory protocol, and memory device type.

257. As I discussed in §§IV(B)(1)(b)(1)-(2), each device profile includes page size, block size, memory protocol (e.g., ONFI support), and memory device type. (See INTEL-1006, ¶42; INTEL-1005, ¶59, Figure 9.) Therefore, the shared attributes in the combination of Danilak and Cornwell include at least “*page size, block size, memory protocol, and memory device type*”, rendering this claim obvious.

7. Claim 19

258. Claim 19 is reproduced below for ease of discussion.

19. The method of claim 8, wherein said first memory device type attribute corresponds to a NAND flash memory device and said second memory device type attribute corresponds to a NOR flash memory device.

259. The memory of the combination of Danilak and Cornwell includes both NAND and NOR devices, as I discussed in §IV(A)(3). The device profile for a NOR Flash device differs from the device profile for a NAND Flash device as I discussed above in §§IV(C)(2)-(4). For example, the non-numerical qualifiers for the device type attribute (NAND or NOR) and ONFI support (Yes or No) differ.

(See, e.g., §IV(C)(4).) The numerical qualifiers for page size, block size, access time, and device density also differ between NOR and NAND devices. (See, e.g., §IV(C)(3).)

260. Thus, the combination of Danilak and Cornwell discloses “*wherein said first memory device type attribute corresponds to a NAND flash memory device and said second memory device type attribute corresponds to a NOR flash memory device.*”

8. Claim 30

261. Claim 30 is reproduced below for ease of discussion.

30. The computer readable medium of claim 29, wherein:

[30A] said first memory location is part of a memory space associated with a first memory device;

[30B] said second memory location is part of a memory space associated with a second memory device; and

[30C] said first and second memory devices comprise at least a portion of a memory store.

262. Each memory device has a set of addressable locations, as I discussed in §IV(B)(1)(a)(2). This set of addressable locations is the “*memory space*” for the device. The combination of Danilak and Cornwell discloses “*said first memory location is part of a memory space associated with a first memory device*” [30A]

and “*said second memory location is part of a memory space associated with a second memory device*” [30B].

263. The “*memory store*” includes multiple memory devices. (See §§IV(B)(1)(a)(3).) Accordingly, the combination of Danilak and Cornwell discloses “*said first and second memory devices comprise at least a portion of a memory store*” [30C].

9. Claims 94-97

264. Claims 94-97 share identical subject matter. I reproduce each claim below for ease of discussion.

94. The method of claim 1, wherein the first memory location and the second memory location are addressable memory locations.

95. The memory controller of claim 45, wherein the first memory location and the second memory location are addressable memory locations.

96. The memory controller of claim 59, wherein the first memory location and the second memory location are addressable memory locations.

97. The electronic device of claim 80, wherein the first memory location and the second memory location are addressable memory locations.

265. Each location in a memory device is addressable. (*See* §IV(B)(1)(a)(2).)

The combination of Danilak and Cornwell discloses “*the first memory location and the second memory location are addressable memory locations*”

V. Conclusion

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

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

Executed this 1st day of April in Los Altos, CA.

Respectfully submitted,



Donald Alpert

APPENDIX

Exhibit	Reference
1001	U.S. Patent No. 9,135,190 to Bruce et al. (“the ’190 patent”)
1002	File History of the ’190 patent
1005	U.S. Patent Publication 2008/0086588 A1 to Danilak et al. (“Danilak”)
1006	U.S. Patent Publication 2007/0180186 A1 to Cornwell et al. (“Cornwell”)
1007	Open NAND Flash Interface Specification, Revision 2.0 (February 27, 2008) (“ONFI specification”)
1008	Joe E. Brewer and Manzur Gill, “Nonvolatile Memory Technologies with Emphasis on Flash: A Comprehensive Guide to Understanding and Using NVM Devices” (IEEE Press, 2008) (“Brewer”)
1009	Peter DiPaolo, “NOR continues to battle NAND flash memory in the Handset” (Design & Reuse, July 12, 2005)
1010	Technical Note NAND Flash 101: An Introduction to NAND Flash and How to Design It In to Your Next Product, Micron Technology Technical Note 29-19 (Micron Technology Inc., November 2006)
1011	NAND vs. NOR Flash Memory Technology Overview (Toshiba America Electronics Components, Inc., April 25, 2006)
1012	G. Campardo, R. Micheloni, and D. Novosel, “VLSI Design of Non-Volatile Memories” (Springer-Verlag Berlin Heidelberg, 2005)
1013	“PoP, SiP, MCM, MCP or SoC? Assessing the Mobile/Embedded Design Tradeoffs” (Technical Article, Embedded, December 15, 2006)
1014	Chul Lee, Sung Hoon Baek, and Kyu Ho Park, “A Hybrid Flash File System Based on NOR and NAND Flash Memories for Embedded Devices (IEEE Transactions on Computers, Vol. 57, No. 7, July 2008)

Exhibit	Reference
1015	Rudolf F. Graf, Modern Dictionary of Electronics (Seventh Edition) (Newnes, 1999), pp. 93, 621, 680
1016	Kostas Magoutis, et al., “Structure and Performance of the Direct Access File System,” In Proceedings of the General Track of the annual conference on USENIX Annual Technical Conference (ATEC ‘02). USENIX Association, USA, 1–14
1017	IEEE 610.10-1994 IEEE Standard Glossary of Computer Hardware Terminology (Standards Coordinating Committee of the IEEE Computer Society, June 14, 1994)
1018	David A. Patterson and John L. Hennessy, Computer Organization and Design The Hardware/Software Interface (Third Edition, Morgan Kaufmann Publishers, 2004)
1019	MC68010/MC68012 16-/32-Bit Virtual Memory Microprocessors (Motorola Inc., May 1995)
1020	Pentium® Pro Family Developer’s Manual, Volume 1: Specifications (Intel Corporation, 1997)
1021	“Selecting and Implementing NAND Flash Storage in Embedded Applications” (Technical Article, Embedded, March 4, 2008)
1023	Numonyx™ StrataFlash® Embedded Memory (P30) Datasheet (Numonyx, B.V., August 2008)
1024	Defendants’ Opening Claim Construction Brief, Case Nos. 6:22-cv-00331-ADA and 6:22-cv-00335-ADA (W.D. Tex.) (Dkt. No. 33)
1025	Public version of Plaintiff BitMicro’s Responsive Claim Construction Brief, Case Nos. 6:22-cv-00331-ADA and 6:22-cv-00335-ADA (W.D. Tex.) (Dkt. No. 44)