(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 February 2014 (06.02.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/022159 A1

(51) **International Patent Classification**:
G06F 11/08 (2006.01) ***G06F 12/00*** (2006.01)

(21) **International Application Number**:
PCT/US2013/051720

(22) **International Filing Date**:
23 July 2013 (23.07.2013)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:
13/565,752 2 August 2012 (02.08.2012) US

(71) **Applicant**: **LSI CORPORATION** [US/US]; 1320 Ridder Park Dr, San Jose, California 95131 (US).

(72) **Inventors**: **CHEN, Zhengang**; 1859 Junewood Ave, San Jose, California 95132 (US). **WU, Yunxiang**; 851 Hyde Ave., Cupertino, California 95014 (US).

(74) **Agent**: **SMITH, Walstein**; P.O. Box 1668, Georgetown, Texas 78627-1668 (US).

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) **Title**: MIXED GRANULARITY HIGHER-LEVEL REDUNDANCY FOR NON-VOLATILE MEMORY

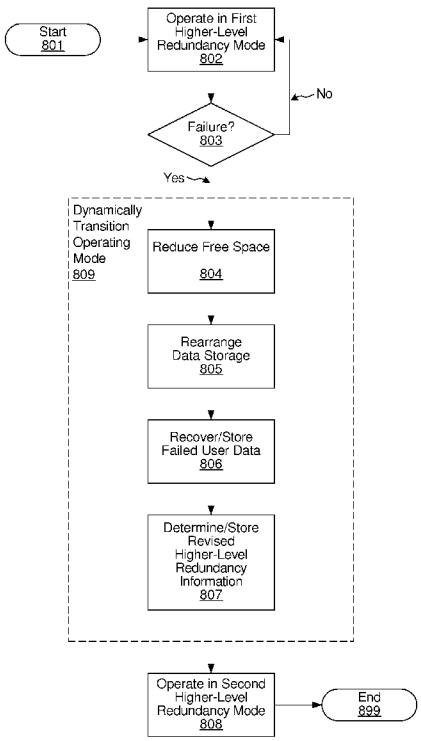


Fig. 8

(57) **Abstract**: Mixed-granularity higher-level redundancy for NVM provides improved higher-level redundancy operation with better error recovery and/or reduced redundancy information overhead. For example, pages of the NVM that are less reliable, such as relatively more prone to errors, are operated in higher-level redundancy modes having relatively more error protection, at a cost of relatively more redundancy information. Concurrently, blocks of the NVM that are more reliable are operated in higher-level redundancy modes having relatively less error protection, at a cost of relatively less redundancy information. Compared to techniques that operate the entirety of the NVM in the higher-level redundancy modes having relatively less error protection, techniques described herein provide better error recovery. Compared to techniques that operate the entirety of the NVM in the higher-level redundancy modes having relatively more error protection, the techniques described herein provide reduced redundancy information overhead.

WO 2014/022159 A1

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

MIXED GRANULARITY HIGHER-LEVEL REDUNDANCY FOR NON-VOLATILE MEMORY

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] Priority benefit claims for this application are made in the accompanying Application Data Sheet, Request, or Transmittal (as appropriate, if any). To the extent permitted by the type of the instant application, this application incorporates by reference for all purposes the following applications, all commonly owned with the instant application at the time the invention was made:

U.S. Non-Provisional Application (Docket No. L12-0674US1 and Serial No. 13/565,752), filed 08-02-2012, first named inventor Zhengang CHEN, and entitled **MIXED GRANULARITY HIGHER-LEVEL REDUNDANCY FOR NON-VOLATILE MEMORY**;

PCT Application (Docket No. SF-10-10PCT and Serial No. PCT/US11/062726), filed 11-30-2011, first named inventor Jeremy Isaac Nathaniel WERNER, and entitled **DYNAMIC HIGHER-LEVEL REDUNDANCY MODE MANAGEMENT WITH INDEPENDENT SILICON ELEMENTS**;

PCT Application (Docket No. SF-10-14PCT and Serial No. PCT/US12/21682), filed 01-18-2012, first named inventor Jeremy Isaac Nathaniel WERNER, and entitled **HIGHER-LEVEL REDUNDANCY INFORMATION COMPUTATION**;

U.S. Provisional Application (Docket No. SF-10-10 and Serial No. 61/418,846), filed 12-01-2010, first named inventor Jeremy Isaac Nathaniel WERNER, and entitled **DYNAMIC HIGHER-LEVEL REDUNDANCY MODE MANAGEMENT WITH INDEPENDENT SILICON ELEMENTS**;

U.S. Provisional Application (Docket No. SF-10-14 and Serial No. 61/433,918), filed 01-18-2011, first named inventor Jeremy Isaac Nathaniel WERNER, and entitled **HIGHER-LEVEL REDUNDANCY INFORMATION COMPUTATION**;

PCT Application (Docket No. SF-10-01PCTB and Serial No. PCT/US11/28244), filed 03-11-2011, first named inventor Hao ZHONG, and entitled **LDPC ERASURE DECODING FOR FLASH MEMORIES**; and

PCT Application (Docket No. SF-10-03PCT and Serial No. PCT/US11/57914), filed 10-26-2011, first named inventor Yan LI, and entitled **ADAPTIVE ECC TECHNIQUES FOR FLASH MEMORY BASED DATA STORAGE**.

BACKGROUND

[0002] Field: Advancements in storage technology are needed to provide improvements in performance, efficiency, and utility of use.

[0003] Related Art: Unless expressly identified as being publicly or well known, mention herein of techniques and concepts, including for context, definitions, or comparison purposes, should not be construed as an admission that such techniques and concepts are previously publicly known or otherwise part of the prior art. All references cited herein (if any), including patents, patent applications, and publications, are hereby incorporated by reference in their entireties, whether specifically incorporated or not, for all purposes.

SYNOPSIS

[0004] The invention may be implemented in numerous ways, e.g. as a process, an article of manufacture, an apparatus, a system, a composition of matter, and a computer readable medium such as a computer readable storage medium (e.g., media in an optical and/or magnetic mass storage device such as a disk, an integrated circuit having non-volatile storage such as flash storage), or a computer network wherein program instructions are sent over optical or electronic communication links. The Detailed Description provides an exposition of one or more embodiments of the invention that enable improvements in cost, profitability, performance, efficiency, and utility of use in the field identified above. The Detailed Description includes an Introduction to facilitate understanding of the remainder of the Detailed Description. The Introduction includes Example Embodiments of one or more of systems, systems, articles of manufacture, and computer readable media in accordance with concepts described herein. As is discussed in more detail in the Conclusions, the invention encompasses all possible modifications and variations within the scope of the issued claims.

Brief Description of Drawings

[0005] Fig. 1A illustrates selected details of an embodiment of a Solid-State Disk (SSD) including an SSD controller providing mixed-granularity higher-level redundancy for Non-Volatile Memories (NVMs).

[0006] Fig. 1B illustrates selected details of various embodiments of systems including one or more instances of the SSD of Fig. 1A.

[0007] Fig. 2 illustrates selected details of an embodiment of mapping a Logical Page Number (LPN) portion of a Logical Block Address (LBA).

[0008] Fig. 3 illustrates selected details of an embodiment of accessing a Non-Volatile Memory (NVM) at a read unit address to produce read data organized as various read units, collectively having a length measured in quanta of read units.

[0009] Fig. 4A illustrates selected details of an embodiment of a read unit.

[0010] Fig. 4B illustrates selected details of another embodiment of a read unit.

[0011] Fig. 5 illustrates selected details of an embodiment of a header having a number of fields.

[0012] Fig. 6 illustrates selected details of an embodiment of blocks, pages, and read units of multiple NVM devices (e.g. one or more flash die and/or flash chips) managed in logical slices and/or sections.

[0013] Fig. 7 illustrates selected details of various embodiments of higher-level redundancy techniques.

[0014] Fig. 8 illustrates selected details of an embodiment of dynamic higher-level redundancy mode management with a Redundant Array of Silicon Independent Elements (RASIE).

[0015] Fig. 9 illustrates an embodiment of read units having lower-level redundancy information of adaptive code rates protected by higher-level redundancy information stored in one or more of the read units.

[0016] Fig. 10 illustrates selected details of an embodiment of higher-level redundancy information result and data source correspondences.

[0017] Fig. 11 illustrates selected details of an embodiment of higher-level redundancy information computations.

[0018] Fig. 12 illustrates selected details of an embodiment of recovery from one (lower-level) failure (during a single operation).

[0019] Figs. 13A-13D illustrate selected details of an embodiment of recovery from two (lower-level) failures (during a single operation).

[0020] Figs. 14A and 14B illustrate selected details of an embodiment of computing higher-level redundancy information with respect to pages received from NVMs.

[0021] Figs. 15A-15C illustrate selected details of an embodiment of backing out of a computation of higher-level redundancy information with respect to a write provided to NVMs.

[0022] Fig. 16 illustrates selected details of an embodiment of mixed granularity higher-level redundancy.

[0023] Fig. 17 illustrates selected details of an embodiment of concurrent operation in a plurality of mixed granularity higher-level redundancy modes.

List of Reference Symbols in Drawings

[0024]

Ref. Symbol	Element Name
100	SSD Controller
101	SSD
102	Host
103	(optional) Switch / Fabric / Intermediate Controller
104	Intermediate Interfaces
105	OS
106	FirmWare (FW)
107	Driver
107D	dotted-arrow (Host Software ←→ I/O Device Communication)
109	Application
109D	dotted-arrow (Application ←→ I/O Device Communication via driver)
109V	dotted-arrow (Application ←→ I/O Device Communication via VF)
110	External Interfaces
111	Host Interfaces
112C	(optional) Card Memory
113	Tag Tracking
114	Multi-Device Management Software
115	Host Software
116	I/O Card
117	I/O & Storage Devices/Resources
118	Servers
119	LAN/WAN
121	Data Processing
123	Engines
131	Buffer
133	DMA
135	ECC-X
137	Memory
141	Map
143	Table

Ref. Symbol	Element Name
151	Recycler
161	ECC
171	CPU
172	CPU Core
173	Command Management
175	Buffer Management
177	Translation Management
179	Coherency Management
180	Memory Interface
181	Device Management
182	Identity Management
190	Device Interfaces
191	Device Interface Logic
192	Flash Device
193	Scheduling
194	Flash Die
199	NVM
211	LBA
213	LPN
215	Logical Offset
221	Map Info for LPN
223	Read Unit Address
225	Length in Read Units
311	Read Data
313	First Read Unit
315	Last Read Unit
401A	Read Unit
401B	Read Unit
410B	Header Marker (HM)
411A	Header 1
411B	Header 1
412B	Header 2
419A	Header N
419B	Header N

Ref. Symbol	Element Name
421A	Data Bytes
421B	Data Bytes
422B	Data Bytes
429B	Data Bytes
431A	Optional Padding Bytes
431B	Optional Padding Bytes
501	Header
511	Type
513	Last Indicator
515	Flags
517	LPN
519	Length
521	Offset
600	Striping Direction
610.0, **610.1**, **610.61**, **610.62**, **610.63**, **610.64**, **610.65**	Flash Die
610.0B0, **610.0B1**, **610.0B2**, **610.0BB**, **610.1B0**, **610.1B1**, **610.1B2**, **610.1BB**, **610.63B0**, **610.63B1**, **610.63B2**, **610.63BB**, **610.64B0**, **610.64B1**,	Blocks

Ref. Symbol	Element Name
610.64B2, **610.64BB**, **610.65B0**, **610.65B1**, **610.65B2**, **610.65BB**	
610.0P0, **610.0P1**, **610.0P2**, **610.0PP**, **610.63P0**, **610.63P1**, **610.63P2**, **610.63PP**, **610.64P0**, **610.64P1**, **610.64P2**, **610.64PP**, **610.65P0**, **610.65P1**, **610.65P2**, **610.65PP**	Pages
610.0R0, **610.0R1**, **610.0RR**, **610.1R0**, **610.1R1**, **610.1RR**, **610.65R0**, **610.65R1**, **610.65RR**	Read Units (RUs)
660.0, **660.1**, **660.2**, **660.R**	R-blocks
720	Flash Device(s)

Ref. Symbol	Element Name
730	Interface Channel(s)
740	Extra Flash Device(s)
750	Extra Interface Channel(s)
801	Start
802	Operate in First Higher-Level Redundancy Mode
803	Failure?
804	Reduce Free Space
805	Rearrange Data Storage
806	Recover/Store Failed User Data
807	Determine/Store Revised Higher-Level Redundancy Information
808	Operate in Second Higher-Level Redundancy Mode
809	Dynamically Transition Operating Mode
899	End
911, 931, 951, 971	Read Unit
911.E, 931.E, 951.E, 971.E	Lower-Level ECC
911.U, 931.U, 951.U, 971.U	User Data
1001	Result
1002	Result
1003	Result
1010	R0
1011	R1
1019	Data
1401A	Start
1401B	Start
1402A	Issue Read Operation
1402B	Page Ready?
1403A	All Issued?
1403B	Perform Page-Based Computations
1404B	Pages Finished?
1499A	End
1499B	End

Ref. Symbol	Element Name
1501A	Start
1501B	Start
1502A	Issue Write Operation
1502B	Write OK?
1503A	All Issued?
1503B	Backout Write from Redundancy
1504B	Writes Finished?
1599A	End
1599B	End
1600, **1601**, **1602**, **1609**	R-pages
1600.D, **1601.D**, **1602.D**, **1609.D**, **1610.D**, **1620.D**, **1690.D**	Data Information
1600.R, **1601.R**, **1602.R**, **1609.R**, **1610.R**, **1620.R**, **1690.R**	Redundancy Information
1700	Concurrently Operate in First and Second Granularity Higher-Level Redundancy Modes
1701	Start
1702	Operate in First Granularity Higher-Level Redundancy Mode
1703	Read/Write According to First Granularity Higher-Level Redundancy Mode
1704	Operate in Second Granularity Higher-Level Redundancy Mode
1705	Read/Write According to Second Granularity Higher-Level Redundancy Mode
1706	Granularity?
1706.1	First

Ref. Symbol	Element Name
1706.2	Second
1707	Read/Write Request

1

2

DETAILED DESCRIPTION

[0025] A detailed description of one or more embodiments of the invention is provided below along with accompanying figures illustrating selected details of the invention. The invention is described in connection with the embodiments. The embodiments herein are understood to be merely exemplary, the invention is expressly not limited to or by any or all of the embodiments herein, and the invention encompasses numerous alternatives, modifications, and equivalents. To avoid monotony in the exposition, a variety of word labels (including but not limited to: first, last, certain, various, further, other, particular, select, some, and notable) may be applied to separate sets of embodiments; as used herein such labels are expressly not meant to convey quality, or any form of preference or prejudice, but merely to conveniently distinguish among the separate sets. The order of some operations of disclosed processes is alterable within the scope of the invention. Wherever multiple embodiments serve to describe variations in process, system, and/or program instruction features, other embodiments are contemplated that in accordance with a predetermined or a dynamically determined criterion perform static and/or dynamic selection of one of a plurality of modes of operation corresponding respectively to a plurality of the multiple embodiments. Numerous specific details are set forth in the following description to provide a thorough understanding of the invention. The details are provided for the purpose of example and the invention may be practiced according to the claims without some or all of the details. For the purpose of clarity, technical material that is known in the technical fields related to the invention has not been described in detail so that the invention is not unnecessarily obscured.

INTRODUCTION

[0026] This introduction is included only to facilitate the more rapid understanding of the Detailed Description; the invention is not limited to the concepts presented in the introduction (including explicit examples, if any), as the paragraphs of any introduction are necessarily an abridged view of the entire subject and are not meant to be an exhaustive or restrictive description. For example, the introduction that follows provides overview information limited by space and organization to only certain embodiments. There are many other embodiments, including those to which claims will ultimately be drawn, discussed throughout the balance of the specification.

Acronyms

[0027] At least some of the various shorthand abbreviations (e.g. acronyms) defined here refer to certain elements used herein.

Acronym	Description
AHCI	Advanced Host Controller Interface
API	Application Program Interface
ATA	Advanced Technology Attachment (AT Attachment)
BCH	Bose Chaudhuri Hocquenghem
CD	Compact Disk
CF	Compact Flash
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DAS	Direct Attached Storage
DDR	Double-Data-Rate
DMA	Direct Memory Access
DNA	Direct NAND Access
DRAM	Dynamic Random Access Memory
DVD	Digital Versatile/Video Disk
DVR	Digital Video Recorder
ECC	Error-Correcting Code
eMMC	Embedded MultiMediaCard
eSATA	external Serial Advanced Technology Attachment
GPS	Global Positioning System
HDD	Hard Disk Drive
I/O	Input / Output
IC	Integrated Circuit
IDE	Integrated Drive Electronics
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
LBA	Logical Block Address
LDPC	Low-Density Parity-Check
LPN	Logical Page Number

Acronym	Description
MLC	Multi-Level Cell
MMC	MultiMediaCard
MPEG	Moving Picture Experts Group
NAS	Network Attached Storage
NCQ	Native Command Queuing
NVM	Non-Volatile Memory
ONA	Optimized NAND Access
ONFI	Open NAND Flash Interface
OS	Operating System
PC	Personal Computer
PCIe	Peripheral Component Interconnect express (PCI express)
PDA	Personal Digital Assistant
PHY	PHYsical interface
POS	Point Of Sale
RAID	Redundant Array of Inexpensive/Independent Disks
RASIE	Redundant Array of Silicon Independent Elements
ReRAM	Resistive Random Access Memory
RS	Reed-Solomon
SAN	Storage Attached Network
SAS	Serial Attached Small Computer System Interface (Serial SCSI)
SATA	Serial Advanced Technology Attachment (Serial ATA)
SCSI	Small Computer System Interface
SD	Secure Digital
SDR	Single-Data-Rate
SLC	Single-Level Cell
SMART	Self-Monitoring Analysis and Reporting Technology
SRAM	Static Random Access Memory
SSD	Solid-State Disk/Drive
UFS	Unified Flash Storage
USB	Universal Serial Bus
VF	Virtual Function
WAN	Wide Area Network

1

[0028] NAND flash memory uses an array of floating gate transistors to store information. In SLC technology, each bit cell (e.g. floating gate transistor) is enabled to store one bit of information. In MLC technology, each bit cell is enabled to store multiple bits of information. As manufacturing technology (e.g. CMOS technology) scales down, each floating gate stores fewer electrons. Further, as storage capacity and density increase, each bit cell stores more bits. Therefore, values stored in the bit cells are represented by smaller voltage ranges. Uncertainties in sensing and/or changes in amount of stored electrons over time increase a probability for data to be stored or read incorrectly. Use of one or more redundancy and/or ECC techniques (e.g. at a lower-level) enables correct retrieval of otherwise corrupted data from NAND flash memory, overcoming, in some usage scenarios, some of the aforementioned difficulties.

[0029] Some types of SSDs use flash memory to provide non-volatile storage (e.g., the flash memory retains information without application of power). Use of one or more ECC and/or redundancy techniques (e.g. at a higher-level) enables correct retrieval of otherwise corrupted data from flash memory, and/or enables proper system-level operation of an SSD even when one or more flash memory elements fail intermittently or permanently.

[0030] For example, an SSD controller enables dynamic higher-level redundancy mode management with independent silicon elements to provide graceful degradation as one or more NVM (e.g. flash) elements fail during operation of an SSD implemented in part by the controller. A portion of the NVM is read. If an error occurs that is not correctable using lower-level redundancy and/or error correction (such as in accordance with one or more ECC techniques), then higher-level redundancy and/or error correction (such as in accordance with one or more RASIE techniques and/or dynamic higher-level redundancy mode management techniques) is used to attempt to correct the error. If a failure of one of the NVM elements is detected by the lower-level and/or the higher-level redundancy and/or error correction, and/or by other techniques (such as a failing status reported by one or more of the NVM elements), then the higher-level redundancy and/or error correction is dynamically transitioned from operating in a current mode to operating in a new mode. The transition includes one or more of reducing free space available on the SSD, rearranging data storage of the SSD, recovering/storing failed user data (if possible), and determining/storing revised higher-level redundancy and/or error correction information. Operation then continues in the new mode. If another failure of another one of the NVM elements is detected with the higher-level redundancy and/or error correction now operating in the new mode, then another transition is made to another higher-level

redundancy and/or error correction mode. Writing the NVM is in accordance with a higher-level redundancy and/or error correction operating mode, including determining/storing higher-level redundancy and/or error correction information in accordance with the higher-level redundancy and/or error correction operating mode and write data.

[0031] If greater than a threshold number and/or rate of lower-level error corrections and/or one or more failures occur for an area of memory, then optionally the area of memory is dynamically transitioned to operating in a new higher-level redundancy and/or error correction operating mode that is enabled to recover from more errors than a current higher-level redundancy and/or error correction operating mode. For example, if the current higher-level redundancy and/or error correction operating mode is none (e.g. no higher-level redundancy information is computed and/or used to recover from a failure of lower-level redundancy to correct an error), then the new higher-level redundancy and/or error correction operating mode is one that enables recovery from a single failure that is uncorrectable by the lower-level redundancy. For another example, if the current higher-level redundancy and/or error correction operating mode is one that enables recovery from a single failure that is uncorrectable by the lower-level redundancy, then the new higher-level redundancy and/or error correction operating mode is one that enables recovery from two failures that are uncorrectable by the lower-level redundancy.

[0032] Determining/storing higher-level redundancy and/or error correction information is in accordance, in some embodiments and/or usage scenarios, with computation of higher-level redundancy information. Higher-level redundancy information computation enables an SSD controller to provide higher-level redundancy capabilities to maintain reliable operation in a context of failures of non-volatile (e.g. flash) memory elements during operation of an SSD implemented in part by the controller. A first portion of higher-level redundancy information is computed using parity coding via an XOR of all pages in a portion of data (e.g. a stripe) to be protected by the higher-level redundancy information. A second portion of the higher-level redundancy information is computed using a weighted-sum technique, each page in the portion being assigned a unique non-zero "index" as a weight when computing the weighted-sum. Arithmetic is performed over a finite field (such as a Galois Field, or such as the integers mod p where p is a prime).

[0033] The portions of the higher-level redundancy information are computable in any order, such as an order determined by an order of completion of one or more read operations performed on NVM elements, or such as an order based on order of data returned and/or

available from NVM elements, enabling, in various embodiments, reduced or eliminated buffering. The any order computability enables, in various embodiments, computing recovery data values and/or backing out of a write using relatively little temporary and/or intermediate buffering and/or state. The portions of the higher-level redundancy information are computable with any degree of parallelism, such as determined by available dedicated hardware elements, enabling, in various embodiments, reduced latency processing and/or reduced memory (e.g. NVM) bandwidth usage.

[0034] In some situations with a number and/or rate of lower-level error events (e.g. correctable and/or uncorrectable errors) and/or failures occurring differentially in various areas of NVM, mixed-granularity higher-level redundancy for the NVM enables improved higher-level redundancy operation with better error recovery and/or reduced redundancy information overhead. In some scenarios, the better error recovery results in increased usable lifetime of the NVM and/or an SSD implemented with the NVM.

[0035] For an example, initially all pages of an NVM have approximately equal rates of lower-level error events. All blocks of the NVM are initially operated in a first higher-level redundancy mode that enables recovery from a single lower-level uncorrectable error that occurs during a single operation. Then there is a change in reliability of the NVM, such that some pages of the NVM become less reliable (e.g. begin having a greater amount of lower-level error events), while all other pages of the NVM remain reliable (e.g. continue having a same or nearly same amount of lower-level error events). The aforementioned amounts of lower-level events are, e.g., a number, a rate, and/or a severity of lower-level events.

[0036] Conceptually, after the change in reliability, the NVM has two regions: a first region that is the blocks having solely reliable pages, and a second region that is the blocks having at least one unreliable page. In response to the change in reliability, the second region is selectively dynamically transitioned to being partially operated in a second higher-level redundancy mode enabling recovery from two lower-level uncorrectable errors that occur during a single operation. The transition is selective and partial in that only the less reliable pages are transitioned to being operated in the second higher-level redundancy mode, while the reliable pages of the second region continue to be operated in the first higher-level redundancy mode. Meanwhile, all of the blocks of the first region continue to be operated in the first higher-level redundancy mode.

[0037] The second region is described as being operated with page granular higher-level redundancy, since the higher-level redundancy mode varies by page (e.g. the reliable pages are operated in the first higher-level redundancy mode and the less reliable pages are operated in the second higher-level redundancy mode). In contrast, the first region is described as being operated with block granular higher-level redundancy, since the higher-level redundancy mode is the same for all blocks of the region (e.g. all of the blocks are operated in the first higher-level redundancy mode). If the first region is an entire R-block (described elsewhere herein), then the first region is alternatively describable as being operated with R-bock granular higher-level redundancy, since the higher-level redundancy mode is the same for all R-blocks of the region (e.g. the entire R-block is operated in the first higher-level redundancy mode).

[0038] Concluding the example, operating the less reliable pages in the second higher-level redundancy mode enables better error recovery than leaving the less reliable pages operating in the first higher-level redundancy mode. Alternatively, operating only the less reliable pages in the second higher-level redundancy mode enables reducing redundancy information overhead compared to operating all of the blocks having any of the unreliable pages in the second higher-level redundancy mode.

EXAMPLE EMBODIMENTS

[0039] In concluding the introduction to the detailed description, what follows is a collection of example embodiments, including at least some explicitly enumerated as "ECs" (Example Combinations), providing additional description of a variety of embodiment types in accordance with the concepts described herein; these examples are not meant to be mutually exclusive, exhaustive, or restrictive; and the invention is not limited to these example embodiments but rather encompasses all possible modifications and variations within the scope of the issued claims and their equivalents.

[0040] EC1) A method comprising:

operating in a first higher-level redundancy mode comprising storing M information portions in M respective areas of memory, each of the M respective areas being a same first size;

operating in a second higher-level redundancy mode comprising storing N information portions in N respective areas of the memory, each of the N respective areas being a same second size;

wherein the M information portions comprise respective M-J data information portions and respective J redundant information portions computed to protect the respective M-J data information portions;

wherein the N information portions comprise respective N-K data information portions and respective K redundant information portions computed to protect the respective N-K data information portions;

wherein each of the M respective areas and the N respective areas are non-overlapping areas of the memory;

wherein the first size is different than the second size; and

wherein each of the M respective areas is in a respective one of M physical devices of the memory and wherein each of the N respective areas is in a respective one of N physical devices of the memory.

[0041] EC2) The method of EC1, wherein at least some of the M physical devices are at least some of the N physical devices.

[0042] EC3) The method of EC1, wherein the M physical devices are the N physical devices.

[0043] EC4) The method of EC1, wherein the operating in the first higher-level redundancy mode and the operating in the second higher-level redundancy mode are concurrently active, such that at least for a portion of time the memory has information corresponding to the storing of the M information portions concurrently with having information corresponding to the storing of the N information portions.

[0044] EC5) The method of EC4, wherein the operating in the second higher-level redundancy mode is in response to any one or more of a manufacturing defect, process variation, uneven wear, an observed change in reliability or error rate, and a failure.

[0045] EC6) The method of EC1, further comprising accessing one or more locations comprised entirely in a particular area that is any one of the M respective areas or any one of the N respective areas, the accessing comprising determining the size of the particular area.

[0046] EC7) The method of EC1, wherein the memory comprises a plurality of flash devices.

[0047] EC8) The method of EC7, wherein the M respective areas correspond to M respective blocks of one or more of the plurality of flash devices.

[0048] EC9) The method of EC8, wherein the one or more flash devices are first one or more flash devices and the N respective areas correspond to N respective pages of a second one or more of the plurality of flash devices.

[0049] EC10) The method of EC9, wherein at least one of the first one or more flash devices and at least one of the second one or more flash devices are a same flash device.

[0050] EC11) The method of EC10, wherein the same flash device comprises an integrated circuit having one or more flash die.

[0051] EC12) The method of EC10, wherein the same flash device comprises a single flash die.

[0052] EC13) A method comprising:

operating in a first higher-level redundancy mode comprising storing M information portions in M respective areas of memory, each of the M respective areas being a same first type;

operating in a second higher-level redundancy mode comprising storing N information portions in N respective areas of the memory, each of the N respective areas being a same second type;

wherein the M information portions comprise respective M-J data information portions and respective J redundant information portions computed to protect the respective M-J data information portions;

wherein the N information portions comprise respective N-K data information portions and respective K redundant information portions computed to protect the respective N-K data information portions;

wherein each of the M respective areas and the N respective areas are non-overlapping areas of the memory;

wherein the first type is different than the second type; and

wherein each of the M respective areas is in a respective one of M physical devices of the memory and wherein each of the N respective areas is in a respective one of N physical devices of the memory.

[0053] EC14) The method of EC13, wherein

the memory comprises a plurality of flash devices;

the M respective areas correspond to M respective blocks of a first one or more of the flash devices, the first type being a block type; and

the N respective areas correspond to N respective pages of a second one or more of the flash devices, the second type being a page type.

[0054] EC15) The method of EC14, further comprising detecting a failure of one of the M respective blocks, and in response, operating at least the failing block in the second higher-level redundancy mode.

[0055] EC16) The method of EC1 or EC13, wherein J is not equal to K.

[0056] EC17) The method of EC16, wherein M is equal to N.

[0057] EC18) The method of EC1 or EC13, wherein M is equal to N.

[0058] EC19) The method of EC1 or EC13, wherein M is not equal to N.

[0059] EC20) The method of EC19, wherein J is not equal to K.

[0060] EC21) The method of EC1 or EC13, wherein at least some of the areas of memory comprise flash memory, and one or more of the acts of storing are at least in part via a flash memory interface coupled to the flash memory and compatible with at least one flash memory interface standard.

[0061] EC22) The method of EC1 or EC13, wherein at least some of the areas of memory comprise flash memory, and one or more of the acts of storing are at least in part in response to a request received from a host via a host interface compatible with at least one storage interface standard.

[0062] EC23) The method of EC1 or EC13, wherein at least some of the areas of memory comprise flash memory, one or more of the acts of storing are at least in part via a flash memory interface coupled to the flash memory and compatible with at least one flash memory interface standard, and at least one of the acts of storing is at least in part in response to a request received from a host via a host interface compatible with at least one storage interface standard.

[0063] EC24) The method of EC23, wherein the flash memory interface and the host interface are collectively implemented in a single Integrated Circuit (IC).

[0064] EC25) The method of EC23, wherein the flash memory interface and the host interface are comprised in a Solid-State Disk (SSD).

[0065] EC26) The method of EC23, further comprising, in the flash memory, receiving one or more write and/or programming commands generated at least in part in response to the at least one of the acts of storing.

[0066] EC27) The method of EC23, further comprising, in the host, generating the request.

[0067] EC28) A system comprising:

means for managing memory according to respective first and second higher-level redundancy modes;

means for storing to the memory, responsive to the means for managing memory;

wherein the means for managing memory is enabled to direct the means for storing to store M information portions in M respective areas of memory, each of the M respective areas being a same first size;

wherein the means for managing memory is further enabled to direct the means for storing to store N information portions in N respective areas of the memory, each of the N respective areas being a same second size;

wherein the M information portions comprise respective M-J data information portions and respective J redundant information portions computed to protect the respective M-J data information portions;

wherein the N information portions comprise respective N-K data information portions and respective K redundant information portions computed to protect the respective N-K data information portions;

wherein each of the M respective areas and the N respective areas are non-overlapping areas of the memory;

wherein the first size is different than the second size; and

wherein each of the M respective areas is in a respective one of M physical devices of the memory and wherein each of the N respective areas is in a respective one of N physical devices of the memory.

[0068] EC29) The system of EC28, wherein at least some of the M physical devices are at least some of the N physical devices.

[0069] EC30) The system of EC28, wherein the M physical devices are the N physical devices.

[0070] EC31) The system of EC28, wherein the means for managing memory is enabled to manage the memory concurrently according the first higher-level redundancy mode and the second higher-level redundancy mode, such that at least for a portion of time the memory has information corresponding to the storing of the M information portions concurrently with having information corresponding to the storing of the N information portions.

[0071] EC32) The system of EC31, wherein the means for managing memory is enabled to transition a portion of the memory from operating in the first higher-level redundancy mode to operating in the second higher-level redundancy mode in response to any one or more of a manufacturing defect, process variation, uneven wear, an observed change in reliability or error rate, and a failure.

[0072] EC33) The system of EC28, further comprising means for accessing one or more locations comprised entirely in a particular area that is any one of the M respective areas or any one of the N respective areas, the means for accessing enabled to determine the size of the particular area.

[0073] EC34) The system of EC28, wherein the memory comprises a plurality of flash devices.

[0074] EC35) The system of EC34, wherein the M respective areas correspond to M respective blocks of one or more of the plurality of flash devices.

[0075] EC36) The system of EC35, wherein the one or more flash devices are first one or more flash devices and the N respective areas correspond to N respective pages of a second one or more of the plurality of flash devices.

[0076] EC37) The system of EC36, wherein at least one of the first one or more flash devices and at least one of the second one or more flash devices are a same flash device.

[0077] EC38) The system of EC37, wherein the same flash device comprises an integrated circuit having one or more flash die.

[0078] EC39) The system of EC37, wherein the same flash device comprises a single flash die.

[0079] EC40) A system comprising:

means for managing memory according to respective first and second higher-level redundancy modes;

means for storing to the memory, responsive to the means for managing memory;

wherein the means for managing memory is enabled to direct the means for storing to store M information portions in M respective areas of memory, each of the M respective areas being a same first type;

wherein the means for managing memory is further enabled to direct the means for storing to store N information portions in N respective areas of the memory, each of the N respective areas being a same second type;

wherein the M information portions comprise respective M-J data information portions and respective J redundant information portions computed to protect the respective M-J data information portions;

wherein the N information portions comprise respective N-K data information portions and respective K redundant information portions computed to protect the respective N-K data information portions;

wherein each of the M respective areas and the N respective areas are non-overlapping areas of the memory;

wherein the first type is different than the second type; and

wherein each of the M respective areas is in a respective one of M physical devices of the memory and wherein each of the N respective areas is in a respective one of N physical devices of the memory.

[0080] EC41) The system of EC40, wherein

the memory comprises a plurality of flash devices;

the M respective areas correspond to M respective blocks of a first one or more of the flash devices, the first type being a block type; and

the N respective areas correspond to N respective pages of a second one or more of the flash devices, the second type being a page type.

[0081] EC42) The system of EC41, wherein the means for managing memory is enabled to detect a failure of one of the M respective blocks and to operate at least the failing block in the second higher-level redundancy mode in response to the detecting.

[0082] EC43) The system of EC28 or EC40, wherein J is not equal to K.

[0083] EC44) The system of EC43, wherein M is equal to N.

[0084] EC45) The system of EC28 or EC40, wherein M is equal to N.

[0085] EC46) The system of EC28 or EC40, wherein M is not equal to N.

[0086] EC47) The system of EC46, wherein J is not equal to K.

[0087] EC48) The system of EC28 or EC40, wherein at least some of the areas of memory comprise flash memory, and the means for storing comprises a flash memory interface coupled to the flash memory and compatible with at least one flash memory interface standard.

[0088] EC49) The system of EC28 or EC40, wherein at least some of the areas of memory comprise flash memory, and at least some of the storing is at least in part in response to a request received from a host via a host interface compatible with at least one storage interface standard.

[0089] EC50) The system of EC28 or EC40, wherein at least some of the areas of memory comprise flash memory, the means for storing comprises a flash memory interface coupled to the flash memory and compatible with at least one flash memory interface standard, and at least some of the storing is at least in part in response to a request received from a host via a host interface compatible with at least one storage interface standard.

[0090] EC51) The system of EC50, wherein the flash memory interface and the host interface are collectively implemented in a single Integrated Circuit (IC).

[0091] EC52) The system of EC50, wherein the flash memory interface and the host interface are comprised in a Solid-State Disk (SSD).

[0092] EC53) The system of EC50, further comprising, in the flash memory, means for receiving one or more write and/or programming commands generated at least in part in response to the at least some of the storing.

[0093] EC54) The system of EC53, further comprising any portions of the flash memory.

[0094] EC55) The system of EC50, further comprising, in the host, means for generating the request.

[0095] EC56) The system of EC55, further comprising any portions of the host.

[0096] EC57) An apparatus comprising:

higher-level redundancy control circuitry enabled to control memory according to respective first and second higher-level redundancy modes;

storage circuitry, responsive to the higher-level redundancy control circuitry;

wherein the higher-level redundancy control circuitry is enabled to direct the storage circuitry to store M information portions in M respective areas of memory, each of the M respective areas being a same first size;

wherein the higher-level redundancy control circuitry is further enabled to direct the storage circuitry to store N information portions in N respective areas of the memory, each of the N respective areas being a same second size;

wherein the M information portions comprise respective M-J data information portions and respective J redundant information portions computed to protect the respective M-J data information portions;

wherein the N information portions comprise respective N-K data information portions and respective K redundant information portions computed to protect the respective N-K data information portions;

wherein each of the M respective areas and the N respective areas are non-overlapping areas of the memory;

wherein the first size is different than the second size; and

wherein each of the M respective areas is in a respective one of M physical devices of the memory and wherein each of the N respective areas is in a respective one of N physical devices of the memory.

[0097] EC58) The apparatus of EC57, wherein at least some of the M physical devices are at least some of the N physical devices.

[0098] EC59) The apparatus of EC57, wherein the M physical devices are the N physical devices.

[0099] EC60) The apparatus of EC57, wherein the higher-level redundancy control circuitry is further enabled to manage the memory concurrently according the first higher-level redundancy mode and the second higher-level redundancy mode, such that at least for a portion of time the memory has information corresponding to the storing of the M information portions concurrently with having information corresponding to the storing of the N information portions.

[0100] EC61) The apparatus of EC60, wherein the higher-level redundancy control circuitry is further enabled to transition a portion of the memory from operating in the first higher-level redundancy mode to operating in the second higher-level redundancy mode in response to any one or more of a manufacturing defect, process variation, uneven wear, an observed change in reliability or error rate, and a failure.

[0101] EC62) The apparatus of EC57, further comprising accessing circuitry enabled to access one or more locations comprised entirely in a particular area that is any one of the M respective areas or any one of the N respective areas, the accessing circuitry further enabled to determine the size of the particular area.

[0102] EC63) The apparatus of EC57, wherein the memory comprises a plurality of flash devices.

[0103] EC64) The apparatus of EC63, wherein the M respective areas correspond to M respective blocks of one or more of the plurality of flash devices.

[0104] EC65) The apparatus of EC64, wherein the one or more flash devices are first one or more flash devices and the N respective areas correspond to N respective pages of a second one or more of the plurality of flash devices.

[0105] EC66) The apparatus of EC65, wherein at least one of the first one or more flash devices and at least one of the second one or more flash devices are a same flash device.

[0106] EC67) The apparatus of EC66, wherein the same flash device comprises an integrated circuit having one or more flash die.

[0107] EC68) The apparatus of EC66, wherein the same flash device comprises a single flash die.

[0108] EC69) An apparatus comprising:

higher-level redundancy control circuitry enabled to control memory according to respective first and second higher-level redundancy modes;

storage circuitry, responsive to the higher-level redundancy control circuitry;

wherein the higher-level redundancy control circuitry is enabled to direct the storage circuitry to store M information portions in M respective areas of memory, each of the M respective areas being a same first type;

wherein the higher-level redundancy control circuitry is further enabled to direct the storage circuitry to store N information portions in N respective areas of the memory, each of the N respective areas being a same second type;

wherein the M information portions comprise respective M-J data information portions and respective J redundant information portions computed to protect the respective M-J data information portions;

wherein the N information portions comprise respective N-K data information portions and respective K redundant information portions computed to protect the respective N-K data information portions;

wherein each of the M respective areas and the N respective areas are non-overlapping areas of the memory;

wherein the first type is different than the second type; and

wherein each of the M respective areas is in a respective one of M physical devices of the memory and wherein each of the N respective areas is in a respective one of N physical devices of the memory.

[0109] EC70) The apparatus of EC69, wherein

the memory comprises a plurality of flash devices;

the M respective areas correspond to M respective blocks of a first one or more of the flash devices, the first type being a block type; and

the N respective areas correspond to N respective pages of a second one or more of the flash devices, the second type being a page type.

[0110] EC71) The apparatus of EC70, wherein the higher-level redundancy control circuitry is further enabled to detect a failure of one of the M respective blocks and to operate at least the failing block in the second higher-level redundancy mode in response to the detecting.

[0111] EC72) The apparatus of EC57 or EC69, wherein J is not equal to K.

[0112] EC73) The apparatus of EC72, wherein M is equal to N.

[0113] EC74) The apparatus of EC57 or EC69, wherein M is equal to N.

[0114] EC75) The apparatus of EC57 or EC69, wherein M is not equal to N.

[0115] EC76) The apparatus of EC75, wherein J is not equal to K.

[0116] EC77) The apparatus of EC57 or EC69, wherein at least some of the areas of memory comprise flash memory, and the storage circuitry comprises a flash memory interface coupled to the flash memory and compatible with at least one flash memory interface standard.

[0117] EC78) The apparatus of EC57 or EC69, wherein at least some of the areas of memory comprise flash memory, and at least some of the storing is at least in part in response to a request received from a host via a host interface compatible with at least one storage interface standard.

[0118] EC79) The apparatus of EC57 or EC69, wherein at least some of the areas of memory comprise flash memory, the storage circuitry comprises a flash memory interface coupled to the flash memory and compatible with at least one flash memory interface standard, and at least some of the storing is at least in part in response to a request received from a host via a host interface compatible with at least one storage interface standard.

[0119] EC80) The apparatus of EC79, wherein the flash memory interface and the host interface are collectively implemented in a single Integrated Circuit (IC).

[0120] EC81) The apparatus of EC79, wherein the flash memory interface and the host interface are comprised in a Solid-State Disk (SSD).

[0121] EC82) The apparatus of EC79, wherein the flash memory is enabled to receive one or more write and/or programming commands generated at least in part in response to the at least some of the storing.

[0122] EC83) The apparatus of EC82, further comprising any portions of the flash memory.

[0123] EC84) The apparatus of EC79, wherein the host is enabled to generate the request.

[0124] EC85) The apparatus of EC84, further comprising any portions of the host.

[0125] EC86) A tangible computer readable medium having a set of instructions stored therein that when executed by a processing element cause the processing element to perform and/or control operations comprising:

operating in a first higher-level redundancy mode comprising storing M information portions in M respective areas of memory, each of the M respective areas being a same first size;

operating in a second higher-level redundancy mode comprising storing N information portions in N respective areas of the memory, each of the N respective areas being a same second size;

wherein the M information portions comprise respective M-J data information portions and respective J redundant information portions computed to protect the respective M-J data information portions;

wherein the N information portions comprise respective N-K data information portions and respective K redundant information portions computed to protect the respective N-K data information portions;

wherein each of the M respective areas and the N respective areas are non-overlapping areas of the memory;

wherein the first size is different than the second size; and

wherein each of the M respective areas is in a respective one of M physical devices of the memory and wherein each of the N respective areas is in a respective one of N physical devices of the memory.

[0126] EC87) The tangible computer readable medium of EC86, wherein at least some of the M physical devices are at least some of the N physical devices.

[0127] EC88) The tangible computer readable medium of EC86, wherein the M physical devices are the N physical devices.

[0128] EC89) The tangible computer readable medium of EC86, wherein the operating in the first higher-level redundancy mode and the operating in the second higher-level redundancy mode are concurrently active, such that at least for a portion of time the memory has information corresponding to the storing of the M information portions concurrently with having information corresponding to the storing of the N information portions.

[0129] EC90) The tangible computer readable medium of EC89, wherein the operating in the second higher-level redundancy mode is in response to any one or more of a manufacturing defect, process variation, uneven wear, an observed change in reliability or error rate, and a failure.

[0130] EC91) The tangible computer readable medium of EC86, wherein the operations further comprise accessing one or more locations comprised entirely in a particular area that is any one of the M respective areas or any one of the N respective areas, the accessing comprising determining the size of the particular area.

[0131] EC92) The tangible computer readable medium of EC86, wherein the memory comprises a plurality of flash devices.

[0132] EC93) The tangible computer readable medium of EC92, wherein the M respective areas correspond to M respective blocks of one or more of the plurality of flash devices.

[0133] EC94) The tangible computer readable medium of EC93, wherein the one or more flash devices are first one or more flash devices and the N respective areas correspond to N respective pages of a second one or more of the plurality of flash devices.

[0134] EC95) The tangible computer readable medium of EC94, wherein at least one of the first one or more flash devices and at least one of the second one or more flash devices are a same flash device.

[0135] EC96) The tangible computer readable medium of EC95, wherein the same flash device comprises an integrated circuit having one or more flash die.

[0136] EC97) The tangible computer readable medium of EC95, wherein the same flash device comprises a single flash die.

[0137] EC98) A tangible computer readable medium having a set of instructions stored therein that when executed by a processing element cause the processing element to perform and/or control operations comprising:

operating in a first higher-level redundancy mode comprising storing M information portions in M respective areas of memory, each of the M respective areas being a same first type;

operating in a second higher-level redundancy mode comprising storing N information portions in N respective areas of the memory, each of the N respective areas being a same second type;

wherein the M information portions comprise respective M-J data information portions and respective J redundant information portions computed to protect the respective M-J data information portions;

wherein the N information portions comprise respective N-K data information portions and respective K redundant information portions computed to protect the respective N-K data information portions;

wherein each of the M respective areas and the N respective areas are non-overlapping areas of the memory;

wherein the first type is different than the second type; and

wherein each of the M respective areas is in a respective one of M physical devices of the memory and wherein each of the N respective areas is in a respective one of N physical devices of the memory.

[0138] EC99) The tangible computer readable medium of EC98, wherein

the memory comprises a plurality of flash devices;

the M respective areas correspond to M respective blocks of a first one or more of the flash devices, the first type being a block type; and

the N respective areas correspond to N respective pages of a second one or more of the flash devices, the second type being a page type.

[0139] EC100) The tangible computer readable medium of EC99, wherein the operations further comprise detecting a failure of one of the M respective blocks, and in response, operating at least the failing block in the second higher-level redundancy mode.

[0140] EC101) The tangible computer readable medium of EC86 or EC98, wherein J is not equal to K.

[0141] EC102) The tangible computer readable medium of EC101, wherein M is equal to N.

[0142] EC103) The tangible computer readable medium of EC86 or EC98, wherein M is equal to N.

[0143] EC104) The tangible computer readable medium of EC86 or EC98, wherein M is not equal to N.

[0144] EC105) The tangible computer readable medium of EC104, wherein J is not equal to K.

[0145] EC106) The tangible computer readable medium of EC86 or EC98, wherein at least some of the areas of memory comprise flash memory, and at least some of the storing is at least in part via a flash memory interface coupled to the flash memory and compatible with at least one flash memory interface standard.

[0146] EC107) The tangible computer readable medium of EC86 or EC98, wherein at least some of the areas of memory comprise flash memory, and at least some of the storing is at least in part in response to a request received from a host via a host interface compatible with at least one storage interface standard.

[0147] EC108) The tangible computer readable medium of EC86 or EC98, wherein at least some of the areas of memory comprise flash memory, at least some of the storing is at least in part via a flash memory interface coupled to the flash memory and compatible with at least one flash memory interface standard, and at least some of the storing is at least in part in response to a request received from a host via a host interface compatible with at least one storage interface standard.

[0148] EC109) The tangible computer readable medium of EC108, wherein the flash memory interface and the host interface are collectively implemented in a single Integrated Circuit (IC).

[0149] EC110) The tangible computer readable medium of EC108, wherein the flash memory interface and the host interface are comprised in a Solid-State Disk (SSD).

[0150] EC111) The tangible computer readable medium of EC108, wherein the flash memory is enabled to receive one or more write and/or programming commands generated at least in part in response to the at least some of the storing.

[0151] EC112) The tangible computer readable medium of EC108, wherein the host is enabled to generate the request.

[0152] EC113) Any of the foregoing ECs having or referring to a flash memory interface standard, wherein the flash memory interface standard is compatible with one or more of

an Open NAND Flash Interface (ONFI),
a Toggle-mode interface,
a Double-Data-Rate (DDR) synchronous interface,
a DDR2 synchronous interface;
a synchronous interface, and
an asynchronous interface.

[0153] EC114) Any of the foregoing ECs having or referring to a host, wherein the host comprises one or more of

a computer,
a workstation computer,
a server computer,
a storage server,
a Storage Attached Network (SAN),
a Network Attached Storage (NAS) device,
a Direct Attached Storage (DAS) device,
a storage appliance,
a Personal Computer (PC),
a laptop computer,
a notebook computer,
a netbook computer,
a tablet device or computer,
an ultrabook computer,
an electronic reading device (an e-reader),
a Personal Digital Assistant (PDA),
a navigation system,
a (handheld) Global Positioning System (GPS) device,
an automotive control system,
an automotive media control system or computer,
a printer, copier or fax machine or all-in-one device,
a Point Of Sale (POS0 device,
a cash-register,
a media player,
a television,
a media recorder,
a Digital Video Recorder (DVR),
a digital camera,
a cellular handset,
a cordless telephone handset, and
an electronic game.

[0154] EC115) Any of the foregoing ECs having or referring to a storage interface standard, wherein the storage interface standard comprises one or more of

a Universal Serial Bus (USB) interface standard,
a Compact Flash (CF) interface standard,
a MultiMediaCard (MMC) interface standard,
an embedded MMC (eMMC) interface standard,
a Thunderbolt interface standard,
a UFS interface standard,
a Secure Digital (SD) interface standard,
a Memory Stick interface standard,
an xD-picture card interface standard,
an Integrated Drive Electronics (IDE) interface standard,
a Serial Advanced Technology Attachment (SATA) interface standard,
an external SATA (eSATA) interface standard,
a Small Computer System Interface (SCSI) interface standard,
a Serial Attached Small Computer System Interface (SAS) interface standard,
a Fibre Channel interface standard,
an Ethernet interface standard, and
a Peripheral Component Interconnect express (PCIe) interface standard.

[0155] EC116) Any of the foregoing ECs having or referring to at least one flash device, wherein at least a portion of the at least one flash device comprises one or more of

NAND flash technology storage cells, and
NOR flash technology storage cells.

[0156] EC117) Any of the foregoing ECs having or referring to at least one flash device, wherein at least a portion of the at least one flash device comprises one or more of

Single-Level Cell (SLC) flash technology storage cells, and
Multi-Level Cell (MLC) flash technology storage cells.

[0157] EC118) Any of the foregoing ECs having or referring to at least one flash device, wherein at least a portion of the at least one flash device comprises one or more of

polysilicon technology-based charge storage cells, and
silicon nitride technology-based charge storage cells.

[0158] EC119) Any of the foregoing ECs having or referring to at least one flash device, wherein at least a portion of the at least one flash device comprises one or more of

two-dimensional technology-based flash memory technology, and

three-dimensional technology-based flash memory technology.

SYSTEM

[0159] In some embodiments, an I/O device, such as an SSD, includes an SSD controller. The SSD controller acts as a bridge between the host interface and NVM of the SSD, and executes commands of a host protocol sent from a computing host via a host interface of the SSD. At least some of the commands direct the SSD to write and read the NVM with data sent from and to the computing host, respectively. In further embodiments, the SSD controller is enabled to use a map to translate between LBAs of the host protocol and physical storage addresses in the NVM. In further embodiments, at least a portion of the map is used for private storage (not visible to the computing host) of the I/O device. For example, a portion of the LBAs not accessible by the computing host is used by the I/O device to manage access to logs, statistics, or other private data.

[0160] In some embodiments, accessing compressed data of varying-sized quanta in NVM provides improved storage efficiency in some usage scenarios. For example, an SSD controller receives (uncompressed) data from a computing host (e.g., relating to a disk write command), compresses the data, and stores the compressed data into flash memory. In response to a subsequent request from the computing host (e.g., relating to a disk read command), the SSD controller reads the compressed data from the flash memory, uncompresses the compressed data, and provides the uncompressed data to the computing host. The compressed data is stored in the flash memory according to varying-sized quanta, the quanta size varying due to, e.g., compression algorithm, operating mode, and compression effectiveness on various data. The SSD controller uncompresses the data in part by consulting an included map table to determine where header(s) are stored in the flash memory. The SSD controller parses the header(s) obtained from the flash memory to determine where appropriate (compressed) data is stored in the flash memory. The SSD controller uncompresses the appropriate data from the flash memory to produce the uncompressed data to provide to the computing host. In the instant application, uncompress (and variants thereof) is synonymous with decompress (and variants thereof).

[0161] In various embodiments, an SSD controller includes a host interface for interfacing with a computing host, an interface for interfacing with NVM such as flash memory, and circuitry for controlling the interfaces and performing (and/or controlling various aspects of the performing) compressing and uncompressing, as well as lower-level redundancy and/or error correction, higher-level redundancy and/or error correction, and dynamic higher-level redundancy mode management with independent silicon elements.

[0162] According to various embodiments, some host interfaces are compatible with one or more of a USB interface standard, a CF interface standard, an MMC interface standard, an eMMC interface standard, a Thunderbolt interface standard, a UFS interface standard, an SD interface standard, a Memory Stick interface standard, an xD-picture card interface standard, an IDE interface standard, a SATA interface standard, a SCSI interface standard, a SAS interface standard, and a PCIe interface standard. According to various embodiments, the computing host is all or any portions of a computer, a workstation computer, a server computer, a storage server, a SAN, a NAS device, a DAS device, a storage appliance, a PC, a laptop computer, a notebook computer, a netbook computer, a tablet device or computer, an ultrabook computer, an electronic reading device (such as an e-reader), a PDA, a navigation system, a (handheld) GPS device, an automotive control system, an automotive media control system or computer, a printer, copier or fax machine or all-in-one device, a POS device, a cash-register, a media player, a television, a media recorder, a DVR, a digital camera, a cellular handset, a cordless telephone handset, and an electronic game. In some embodiments, an interfacing host (such as a SAS/SATA bridge) operates as a computing host and/or as a bridge to a computing host.

[0163] In various embodiments, the SSD controller includes one or more processors. The processors execute firmware to control and/or perform operation of the SSD controller. The SSD controller communicates with the computing host to send and receive commands and/or status as well as data. The computing host executes one or more of an operating system, a driver, and an application. Communication by the computing host with the SSD controller is optionally and/or selectively via the driver and/or via the application. In a first example, all communication to the SSD controller is via the driver, and the application provides higher-level commands to the driver that the driver translates into specific commands for the SSD controller. In a second example, the driver implements a bypass mode and the application is enabled to send specific commands to the SSD controller via the driver. In a third example, a PCIe SSD controller supports one or more Virtual Functions (VFs), enabling an application, once configured, to communicate directly with the SSD controller, bypassing the driver.

[0164] According to various embodiments, some SSDs are compatible with form-factors, electrical interfaces, and/or protocols used by magnetic and/or optical non-volatile storage, such as HDDs, CD drives, and DVD drives. In various embodiments, SSDs use various combinations of zero or more parity codes, zero or more RS codes, zero or more BCH codes, zero or more Viterbi or other trellis codes, and zero or more LDPC codes.

[0165] Fig. 1A illustrates selected details of an embodiment of an SSD including an SSD controller providing mixed-granularity higher-level redundancy for NVMs. The SSD controller is for managing non-volatile storage, such as implemented via NVM elements (e.g., flash memories). SSD Controller **100** is communicatively coupled via one or more External Interfaces **110** to a host (not illustrated). According to various embodiments, External Interfaces **110** are one or more of: a SATA interface; a SAS interface; a PCIe interface; a Fibre Channel interface; an Ethernet Interface (such as 10 Gigabit Ethernet); a non-standard version of any of the preceding interfaces; a custom interface; or any other type of interface used to interconnect storage and/or communications and/or computing devices. For example, in some embodiments, SSD Controller **100** includes a SATA interface and a PCIe interface.

[0166] SSD Controller **100** is further communicatively coupled via one or more Device Interfaces **190** to NVM **199** including one or more storage devices, such as one or more instances of Flash Device **192**. According to various embodiments, Device Interfaces **190** are one or more of: an asynchronous interface; a synchronous interface; a single-data-rate (SDR) interface; a double-data-rate (DDR) interface; a DRAM-compatible DDR or DDR2 synchronous interface; an ONFI compatible interface, such as an ONFI 2.2 or ONFI 3.0 compatible interface; a Toggle-mode compatible flash interface; a non-standard version of any of the preceding interfaces; a custom interface; or any other type of interface used to connect to storage devices.

[0167] Each of Flash Device **192** has, in some embodiments, one or more individual Flash Die **194**. According to type of a particular one of Flash Device **192**, a plurality of Flash Die **194** in the particular Flash Device **192** is optionally and/or selectively accessible in parallel. Flash Device **192** is merely representative of one type of storage device enabled to communicatively couple to SSD Controller **100**. In various embodiments, any type of storage device is usable, such as an SLC NAND flash memory, MLC NAND flash memory, NOR flash memory, flash memory using polysilicon or silicon nitride technology-based charge storage cells, two- or three-dimensional technology-based flash memory, read-only memory, static random access memory, dynamic random access memory, ferromagnetic memory, phase-change memory, racetrack memory, ReRAM, or any other type of memory device or storage medium.

[0168] According to various embodiments, Device Interfaces **190** are organized as: one or more busses with one or more instances of Flash Device **192** per bus; one or more groups of busses with one or more instances of Flash Device **192** per bus, having busses in a group generally accessed in parallel; or any other organization of one or more instances of Flash Device **192** onto Device Interfaces **190**.

[0169] Continuing in Fig. 1A, SSD Controller **100** has one or more modules, such as Host Interfaces **111**, Data Processing **121**, Buffer **131**, Map **141**, Recycler **151**, ECC **161**, Device Interface Logic **191**, and CPU **171**. The specific modules and interconnections illustrated in Fig. 1A are merely representative of one embodiment, and many arrangements and interconnections of some or all of the modules, as well as additional modules not illustrated, are conceived. In a first example, in some embodiments, there are two or more Host Interfaces **111** to provide dual-porting. In a second example, in some embodiments, Data Processing **121** and/or ECC **161** are combined with Buffer **131**. In a third example, in some embodiments, Host Interfaces **111** is directly coupled to Buffer **131**, and Data Processing **121** optionally and/or selectively operates on data stored in Buffer **131**. In a fourth example, in some embodiments, Device Interface Logic **191** is directly coupled to Buffer **131**, and ECC **161** optionally and/or selectively operates on data stored in Buffer **131**.

[0170] Host Interfaces **111** sends and receives commands and/or data via External Interfaces **110**, and, in some embodiments, tracks progress of individual commands via Tag Tracking **113**. For example, the commands include a read command specifying an address (such as an LBA) and an amount of data (such as a number of LBA quanta, e.g., sectors) to read; in response the SSD provides read status and/or read data. For another example, the commands include a write command specifying an address (such as an LBA) and an amount of data (such as a number of LBA quanta, e.g., sectors) to write; in response the SSD provides write status and/or requests write data and optionally subsequently provides write status. For yet another example, the commands include a de-allocation command (e.g. a trim command) specifying one or more addresses (such as one or more LBAs) that no longer need be allocated; in response the SSD modifies the Map accordingly and optionally provides de-allocation status. In some contexts an ATA compatible TRIM command is an exemplary de-allocation command. For yet another example, the commands include a super capacitor test command or a data hardening success query; in response, the SSD provides appropriate status. In some embodiments, Host Interfaces **111** is compatible with a SATA protocol and, using NCQ commands, is enabled to have up to 32 pending commands, each with a unique tag represented as a number from 0 to 31. In some embodiments, Tag Tracking **113** is enabled to associate an external tag for a command received via External Interfaces **110** with an internal tag used to track the command during processing by SSD Controller **100**.

[0171] According to various embodiments, one or more of: Data Processing **121** optionally and/or selectively processes some or all data sent between Buffer **131** and External

Interfaces **110**; and Data Processing **121** optionally and/or selectively processes data stored in Buffer **131**. In some embodiments, Data Processing **121** uses one or more Engines **123** to perform one or more of: formatting; reformatting; transcoding; and any other data processing and/or manipulation task.

[0172] Buffer **131** stores data sent to/from External Interfaces **110** from/to Device Interfaces **190**. In some embodiments, Buffer **131** additionally stores system data, such as some or all map tables, used by SSD Controller **100** to manage one or more instances of Flash Device **192**. In various embodiments, Buffer **131** has one or more of: Memory **137** used for temporary storage of data; DMA **133** used to control movement of data to and/or from Buffer **131**; and ECC-X **135** used to provide higher-level error correction and/or redundancy functions; and other data movement and/or manipulation functions. An example of a higher-level redundancy function is a RAID-like capability (e.g. RASIE), with redundancy at a flash device level (e.g., multiple ones of Flash Device **192**) and/or a flash die (e.g., Flash Die **194**) level instead of at a disk level.

[0173] According to various embodiments, one or more of: ECC **161** optionally and/or selectively processes some or all data sent between Buffer **131** and Device Interfaces **190**; and ECC **161** optionally and/or selectively processes data stored in Buffer **131**. In some embodiments, ECC **161** is used to provide lower-level error correction and/or redundancy functions, such as in accordance with one or more ECC techniques. In some embodiments, ECC **161** implements one or more of: a CRC code; a Hamming code; an RS code; a BCH code; an LDPC code; a Viterbi code; a trellis code; a hard-decision code; a soft-decision code; an erasure-based code; any error detecting and/or correcting code; and any combination of the preceding. In some embodiments, ECC **161** includes one or more decoders (such as LDPC decoders).

[0174] Device Interface Logic **191** controls instances of Flash Device **192** via Device Interfaces **190**. Device Interface Logic **191** is enabled to send data to/from the instances of Flash Device **192** according to a protocol of Flash Device **192**. Device Interface Logic **191** includes Scheduling **193** to selectively sequence control of the instances of Flash Device **192** via Device Interfaces **190**. For example, in some embodiments, Scheduling **193** is enabled to queue operations to the instances of Flash Device **192**, and to selectively send the operations to individual ones of the instances of Flash Device **192** (or Flash Die **194**) as individual ones of the instances of Flash Device **192** (or Flash Die **194**) are available.

[0175] Map **141** converts between data addressing used on External Interfaces **110** and data addressing used on Device Interfaces **190**, using Table **143** to map external data addresses to locations in NVM **199**. For example, in some embodiments, Map **141** converts LBAs used on External Interfaces **110** to block and/or page addresses targeting one or more Flash Die **194**, via mapping provided by Table **143**. For LBAs that have never been written since drive manufacture or de-allocation, the Map points to a default value to return if the LBAs are read. For example, when processing a de-allocation command, the Map is modified so that entries corresponding to the de-allocated LBAs point to one of the default values. In various embodiments, there are various default values, each having a corresponding pointer. The plurality of default values enables reading some de-allocated LBAs (such as in a first range) as one default value, while reading other de-allocated LBAs (such as in a second range) as another default value. The default values, in various embodiments, are defined by flash memory, hardware, firmware, command and/or primitive arguments and/or parameters, programmable registers, or various combinations thereof.

[0176] In some embodiments, Map **141** uses Table **143** to perform and/or to look up translations between addresses used on External Interfaces **110** and data addressing used on Device Interfaces **190**. According to various embodiments, Table **143** is one or more of: a one-level map; a two-level map; a multi-level map; a map cache; a compressed map; any type of mapping from one address space to another; and any combination of the foregoing. According to various embodiments, Table **143** includes one or more of: static random access memory; dynamic random access memory; NVM (such as flash memory); cache memory; on-chip memory; off-chip memory; and any combination of the foregoing.

[0177] In some embodiments, Recycler **151** performs garbage collection. For example, in some embodiments, instances of Flash Device **192** contain blocks that must be erased before the blocks are re-writeable. Recycler **151** is enabled to determine which portions of the instances of Flash Device **192** are actively in use (e.g., allocated instead of de-allocated), such as by scanning a map maintained by Map **141**, and to make unused (e.g., de-allocated) portions of the instances of Flash Device **192** available for writing by erasing the unused portions. In further embodiments, Recycler **151** is enabled to move data stored within instances of Flash Device **192** to make larger contiguous portions of the instances of Flash Device **192** available for writing.

[0178] In some embodiments, instances of Flash Device **192** are selectively and/or dynamically configured, managed, and/or used to have one or more bands for storing data of

different types and/or properties. A number, arrangement, size, and type of the bands are dynamically changeable. For example, data from a computing host is written into a hot (active) band, while data from Recycler **151** is written into a cold (less active) band. In some usage scenarios, if the computing host writes a long, sequential stream, then a size of the hot band grows, whereas if the computing host does random writes or few writes, then a size of the cold band grows.

[0179] CPU **171** controls various portions of SSD Controller **100**. CPU **171** includes CPU Core **172**. CPU Core **172** is, according to various embodiments, one or more single-core or multi-core processors. The individual processors cores in CPU Core **172** are, in some embodiments, multi-threaded. CPU Core **172** includes instruction and/or data caches and/or memories. For example, the instruction memory contains instructions to enable CPU Core **172** to execute programs (e.g. software sometimes called firmware) to control SSD Controller **100**. In some embodiments, some or all of the firmware executed by CPU Core **172** is stored on instances of Flash Device **192** (as illustrated, e.g., as Firmware **106** of NVM **199** in Fig. 1B).

[0180] In various embodiments, CPU **171** further includes: Command Management **173** to track and control commands received via External Interfaces **110** while the commands are in progress; Buffer Management **175** to control allocation and use of Buffer **131**; Translation Management **177** to control Map **141**; Coherency Management **179** to control consistency of data addressing and to avoid conflicts such as between external data accesses and recycle data accesses; Device Management **181** to control Device Interface Logic **191**; Identity Management **182** to control modification and communication of identify information, and optionally other management units. None, any, or all of the management functions performed by CPU **171** are, according to various embodiments, controlled and/or managed by hardware, by software (such as firmware executing on CPU Core **172** or on a host connected via External Interfaces **110**), or any combination thereof.

[0181] In some embodiments, CPU **171** is enabled to perform other management tasks, such as one or more of: gathering and/or reporting performance statistics; implementing SMART; controlling power sequencing, controlling and/or monitoring and/or adjusting power consumption; responding to power failures; controlling and/or monitoring and/or adjusting clock rates; and other management tasks.

[0182] Various embodiments include a computing-host flash memory controller that is similar to SSD Controller **100** and is compatible with operation with various computing hosts,

such as via adaptation of Host Interfaces **111** and/or External Interfaces **110**. The various computing hosts include one or any combination of a computer, a workstation computer, a server computer, a storage server, a SAN, a NAS device, a DAS device, a storage appliance, a PC, a laptop computer, a notebook computer, a netbook computer, a tablet device or computer, an ultrabook computer, an electronic reading device (such as an e-reader), a PDA, a navigation system, a (handheld) GPS device, an automotive control system, an automotive media control system or computer, a printer, copier or fax machine or all-in-one device, a POS device, a cash-register, a media player, a television, a media recorder, a DVR, a digital camera, a cellular handset, a cordless telephone handset, and an electronic game.

[0183] In various embodiments, all or any portions of an SSD controller (or a computing-host flash memory controller) are implemented on a single IC, a single die of a multi-die IC, a plurality of dice of a multi-die IC, or a plurality of ICs. For example, Buffer **131** is implemented on a same die as other elements of SSD Controller **100**. For another example, Buffer **131** is implemented on a different die than other elements of SSD Controller **100**.

[0184] Fig. 1B illustrates selected details of various embodiments of systems including one or more instances of the SSD of Fig. 1A. SSD **101** includes SSD Controller **100** coupled to NVM **199** via Device Interfaces **190**. The figure illustrates various classes of embodiments: a single SSD coupled directly to a host, a plurality of SSDs each respectively coupled directly to a host via respective external interfaces, and one or more SSDs coupled indirectly to a host via various interconnection elements.

[0185] As an example embodiment of a single SSD coupled directly to a host, one instance of SSD **101** is coupled directly to Host **102** via External Interfaces **110** (e.g. Switch / Fabric / Intermediate Controller **103** is omitted, bypassed, or passed-through). As an example embodiment of a plurality of SSDs each coupled directly to a host via respective external interfaces, each of a plurality of instances of SSD **101** is respectively coupled directly to Host **102** via a respective instance of External Interfaces **110** (e.g. Switch / Fabric / Intermediate Controller **103** is omitted, bypassed, or passed-through). As an example embodiment of one or more SSDs coupled indirectly to a host via various interconnection elements, each of one or more instances of SSD **101** is respectively coupled indirectly to Host **102**. Each indirect coupling is via a respective instance of External Interfaces **110** coupled to Switch / Fabric / Intermediate Controller **103**, and Intermediate Interfaces **104** coupling to Host **102**.

[0186] Some of the embodiments including Switch / Fabric / Intermediate Controller **103** also include Card Memory **112C** coupled via Memory Interface **180** and accessible by the SSDs. In various embodiments, one or more of the SSDs, the Switch / Fabric / Intermediate Controller, and/or the Card Memory are included on a physically identifiable module, card, or pluggable element (e.g. I/O Card **116**). In some embodiments, SSD **101** (or variations thereof) corresponds to a SAS drive or a SATA drive that is coupled to an initiator operating as Host **102**.

[0187] Host **102** is enabled to execute various elements of Host Software **115**, such as various combinations of OS **105**, Driver **107**, Application **109**, and Multi-Device Management Software **114**. Dotted-arrow **107D** is representative of Host Software ←→ I/O Device Communication, e.g. data sent/received to/from one or more of the instances of SSD **101** and from/to any one or more of OS **105** via Driver **107**, Driver **107**, and Application **109**, either via Driver **107**, or directly as a VF.

[0188] OS **105** includes and/or is enabled to operate with drivers (illustrated conceptually by Driver **107**) for interfacing with the SSD. Various versions of Windows (e.g. 95, 98, ME, NT, XP, 2000, Server, Vista, and 7), various versions of Linux (e.g. Red Hat, Debian, and Ubuntu), and various versions of MacOS (e.g. 8, 9 and X) are examples of OS **105**. In various embodiments, the drivers are standard and/or generic drivers (sometimes termed "shrink-wrapped" or "pre-installed") operable with a standard interface and/or protocol such as SATA, AHCI, or NVM Express, or are optionally customized and/or vendor specific to enable use of commands specific to SSD **101**. Some drives and/or drivers have pass-through modes to enable application-level programs, such as Application **109** via Optimized NAND Access (sometimes termed ONA) or Direct NAND Access (sometimes termed DNA) techniques, to communicate commands directly to SSD **101**, enabling a customized application to use commands specific to SSD **101** even with a generic driver. ONA techniques include one or more of: use of non-standard modifiers (hints); use of vendor-specific commands; communication of non-standard statistics, such as actual NVM usage according to compressibility; and other techniques. DNA techniques include one or more of: use of non-standard commands or vendor-specific providing unmapped read, write, and/or erase access to the NVM; use of non-standard or vendor-specific commands providing more direct access to the NVM, such as by bypassing formatting of data that the I/O device would otherwise do; and other techniques. Examples of the driver are a driver without ONA or DNA support, an ONA-enabled driver, a DNA-enabled driver, and an ONA/DNA-enabled driver. Further examples of the driver are a vendor-provided, vendor-developed, and/or vendor-enhanced driver, and a client-provided, client-developed, and/or client-enhanced driver.

[0189] Examples of the application-level programs are an application without ONA or DNA support, an ONA-enabled application, a DNA-enabled application, and an ONA/DNA-enabled application. Dotted-arrow **109D** is representative of Application ←→ I/O Device Communication (e.g. bypass via a driver or bypass via a VF for an application), e.g. an ONA-enabled application and an ONA-enabled driver communicating with an SSD, such as without the application using the OS as an intermediary. Dotted-arrow **109V** is representative of Application ←→ I/O Device Communication (e.g. bypass via a VF for an application), e.g. a DNA-enabled application and a DNA-enabled driver communicating with an SSD, such as without the application using the OS or the driver as intermediaries.

[0190] One or more portions of NVM **199** are used, in some embodiments, for firmware storage, e.g. Firmware **106**. The firmware storage includes one or more firmware images (or portions thereof). A firmware image has, for example, one or more images of firmware, executed, e.g., by CPU Core **172** of SSD Controller **100**. A firmware image has, for another example, one or more images of constants, parameter values, and NVM device information, referenced, e.g. by the CPU core during the firmware execution. The images of firmware correspond, e.g., to a current firmware image and zero or more previous (with respect to firmware updates) firmware images. In various embodiments, the firmware provides for generic, standard, ONA, and/or DNA operating modes. In some embodiments, one or more of the firmware operating modes are enabled (e.g. one or more APIs are "unlocked") via keys or various software techniques, optionally communicated and/or provided by a driver.

[0191] In some embodiments lacking the Switch / Fabric / Intermediate Controller, the SSD is coupled to the Host directly via External Interfaces **110**. In various embodiments, SSD Controller **100** is coupled to the Host via one or more intermediate levels of other controllers, such as a RAID controller. In some embodiments, SSD **101** (or variations thereof) corresponds to a SAS drive or a SATA drive and Switch / Fabric / Intermediate Controller **103** corresponds to an expander that is in turn coupled to an initiator, or alternatively Switch / Fabric / Intermediate Controller **103** corresponds to a bridge that is indirectly coupled to an initiator via an expander. In some embodiments, Switch / Fabric / Intermediate Controller **103** includes one or more PCIe switches and/or fabrics.

[0192] In various embodiments, such as some of the embodiments with Host **102** as a computing host (e.g. a computer, a workstation computer, a server computer, a storage server, a SAN, a NAS device, a DAS device, a storage appliance, a PC, a laptop computer, a notebook

computer, and/or a netbook computer), the computing host is optionally enabled to communicate (e.g. via optional I/O & Storage Devices/Resources **117** and optional LAN/WAN **119**) with one or more local and/or remote servers (e.g. optional Servers **118**). The communication enables, for example, local and/or remote access, management, and/or usage of any one or more of SSD **101** elements. In some embodiments, the communication is wholly or partially via Ethernet. In some embodiments, the communication is wholly or partially via Fibre Channel. LAN/WAN **119** is representative, in various embodiments, of one or more Local and/or Wide Area Networks, such as any one or more of a network in a server farm, a network coupling server farms, a metro-area network, and the Internet.

[0193] In various embodiments, an SSD controller and/or a computing-host flash memory controller in combination with one or more NVMs are implemented as a non-volatile storage component, such as a USB storage component, a CF storage component, an MMC storage component, an eMMC storage component, a Thunderbolt storage component, a UFS storage component, an SD storage component, a Memory Stick storage component, and an xD-picture card storage component.

[0194] In various embodiments, all or any portions of an SSD controller (or a computing-host flash memory controller), or functions thereof, are implemented in a host that the controller is to be coupled with (e.g., Host **102** of Fig. 1B). In various embodiments, all or any portions of an SSD controller (or a computing-host flash memory controller), or functions thereof, are implemented via hardware (e.g., logic circuitry), software and/or firmware (e.g., driver software and/or SSD control firmware), or any combination thereof. For example, functionality of or associated with an ECC unit (such as similar to ECC **161** and/or ECC-X **135** of Fig. 1A) is implemented partially via software on a host and partially via a combination of firmware and hardware in an SSD controller. For another example, functionality of or associated with a recycler unit (such as similar to Recycler **151** of Fig. 1A) is implemented partially via software on a host and partially via hardware in a computing-host flash memory controller.

MAPPING OPERATION

[0195] Fig. 2 illustrates selected details of an embodiment of mapping an LPN portion of an LBA. In some embodiments, a read unit is a finest granularity of an NVM that is independently readable, such as a portion of a page of the NVM. In further embodiments, the

read unit corresponds to check bits (sometimes-termed redundancy) of a (lower-level) error-correcting code along with all data protected by the check bits. For example, ECC **161** of Fig. 1A implements error correction via check bits such as via an LDPC code, and a read unit corresponds to coding bits implementing the LDPC code in addition to data bits protected by the LDPC coding bits.

[0196] In some embodiments, Map **141** maps LPN **213** portion of LBA **211** to Map Info for LPN **221**, such as via Table **143** (as illustrated in Fig. 1A). Map info for an LPN (such as Map Info for LPN **221)** is sometimes termed a map entry. Map **141** is said to associate an LPN with a corresponding map entry. In various embodiments, mapping is via one or more associative look-ups, via one or more non-associative look-ups, and/or via one or more other techniques.

[0197] In some embodiments, SSD Controller **100** maintains one map entry for each LPN potentially and/or actively in use.

[0198] In some embodiments, Map Info for LPN **221** includes respective Read Unit Address **223** and Length in Read Units **225**. In some embodiments, a length and/or a span are stored encoded, such as by storing the length as an offset from the span, e.g. in all or any portions of Length in Read Units **225**. In further embodiments, a first LPN is associated with a first map entry, a second LPN (different from the first LPN, but referring to a logical page of a same size as a logical page referred to by the first LPN) is associated with a second map entry, and the respective length in read units of the first map entry is different from the respective length in read units of the second map entry.

[0199] In various embodiments, at a same point in time, a first LPN is associated with a first map entry, a second LPN (different from the first LPN) is associated with a second map entry, and the respective read unit address of the first map entry is the same as the respective read unit address of the second map entry. In further embodiments, data associated with the first LPN and data associated with the second LPN are both stored in a same physical page of a same device in NVM **199**.

[0200] According to various embodiments, Read Unit Address **223** is associated with one or more of: a starting address in the NVM; an ending address in the NVM; an offset of any of the preceding; and any other techniques for identifying a portion of the NVM associated with LPN **213**.

[0201] Fig. 3 illustrates selected details of an embodiment of accessing an NVM at a read unit address to produce read data organized as various read units, collectively having a length measured in quanta of read units. According to various embodiments, First Read Unit **313** is one or more of: a one of read units in Read Data **311** with a lowest address in an address space of the NVM; a fixed one of the read units; an arbitrary one of the read units; a variable one of the read units; and a one of the read units selected by any other technique. In various embodiments, SSD Controller **100** is enabled to access NVM **199** and produce Read Data **311** by reading no more than a number of read units specified by Length in Read Units **225**.

[0202] Fig. 4A illustrates selected details of an embodiment of a read unit (such as Read Units **313** or **315** of Fig. 3) as Read Unit **401A**. In various embodiments and/or usage scenarios, Header 1 **411A** through Header N **419A** are contiguous, and respective data regions identified (such as via respective offsets) by each of the headers are contiguous following a last one of the headers. The data regions collectively form Data Bytes **421A**. The data regions are stored in a location order that matches the location order the headers are stored. For example, consider a first header, at the beginning of a read unit, with a second header and a third header contiguously following the first header. A first data region (identified by a first offset in the first header) contiguously follows the third header. A second data region (identified by a second offset in the second header) contiguously follows the first data region. Similarly, a third data region (identified by the third header) contiguously follows the second data region.

[0203] Fig. 4B illustrates selected details of another embodiment of a read unit (such as Read Units **313** or **315** of Fig. 3) as Read Unit **401B**. In various embodiments and/or usage scenarios, Header Marker (HM) **410B** is an optional initial field (such as a one-byte field) indicating a number of following contiguous headers (Header 1 **411B**, Header 2 **412B** ... Header N **419B**). Data regions (Data Bytes **421B**, Data Bytes **422B** ... Data Bytes **429B**) are identified respectively by the headers (Header 1 **411B**, Header 2 **412B** ... Header N **419B**) and are stored in a location order that is opposite of the location order that the headers are stored. Headers start at the beginning of a read unit, while corresponding data regions start at the end of a read unit. In some embodiments, data bytes within a data region (e.g. Data Bytes **421B**, Data Bytes **422B** ... Data Bytes **429B**) are arranged in a forward order (byte order matching location order), while in other embodiments, the data bytes are arranged in a reverse order (byte order reversed with respect to location order). In some embodiments, a header marker is used in read units where headers and data bytes are stored in a same location order (e.g. as illustrated in Fig. 4A).

[0204] In some embodiments, Optional Padding Bytes **431A** (or **431B**) are according to granularity of data associated with a particular LPN. For example, in some embodiments, if Data Bytes **421A** (or collectively Data Bytes **421B**, Data Bytes **422B** ... Data Bytes **429B**) have less than a fixed amount of remaining space, such as 8 bytes, after storing data associated with all but a last one of Header 1 **411A** through Header N **419A** (or Header 1 **411B**, Header 2 **412B** ... Header N **419B**), then data for an LPN associated with the last header starts in a subsequent read unit. In further embodiments, a particular offset value (e.g. all ones) in the last header indicates that the data for the LPN associated with the last header starts in the subsequent read unit.

[0205] Fig. 5 illustrates selected details of an embodiment of a header (such as any of Header 1 **411A** through Header N **419A** of Fig. 4A or Header 1 **411B** through Header **419B** of Fig. 4B) having a number of fields. In some embodiments, headers are fixed-length (e.g. each header is a same number of bytes long). Header **501** includes fields Type **511**, Last Indicator **513**, Flags **515**, LPN **517**, Length **519**, and Offset **521**. The type field identifies a category of the data bytes. For example, the type field indicates the category of the data bytes is one of host data (e.g. logical page data) or system data (e.g. map information or checkpoint information). The last field indicates that the header is the last header before the data bytes. In some embodiments with a header marker, the last field is optionally omitted. The LPN field is the LPN that the header is associated with. The LPN field enables parsing of the headers to determine a particular one of the headers that is associated with a particular LPN by, for example, searching the headers for one with an LPN field matching the particular LPN. The length field is the length, in bytes, of the data bytes (e.g. how many bytes of data there are in Data Bytes **421A** associated with Header **501**). In some embodiments, an offset in the offset field is rounded according to a particular granularity (e.g. 8-byte granularity).

[0206] In various embodiments, some or all information associated with a particular LPN is stored in a map entry associated with the particular LPN, a header associated with the particular LPN, or both. For example, in some embodiments, some or all of Length **519** is stored in a map entry rather than in a header.

[0207] Fig. 6 illustrates selected details of an embodiment of blocks, pages, and read units of multiple NVM devices (e.g. one or more flash die and/or flash chips) managed in logical slices and/or sections. The management functions include any one or more of reading, recycling, erasing, programming/writing, and other management functions. The logical slices and/or sections are sometimes referred to as R-blocks. The figure illustrates an embodiment

with 66 flash die. Three of the flash die are explicitly illustrated (Flash Die **610.65**, **610.1**, and **610.0**) and 63 of the flash die are implicitly illustrated (**610.64** … **610.2**).

[0208] Each of the flash die (such as any one of Flash Die **610.65** … **610.1**, and **610.0**) provides storage organized as blocks (such as Blocks **610.65BB** … **610.65B1**, and **610.65B0** of Flash Die **610.65**; Blocks **610.0BB** … **610.0B1**, and **610.0B0** of Flash Die **610.0**; and so forth). The blocks in turn include pages (such as Pages **610.65PP** … **610.65P1**, and **610.65P0** of Block **610.65B0**; Pages **610.0PP** … **610.0P1**, and **610.0P0** of Block **610.0B0**; and so forth). The pages in turn include read units (such as Read Units **610.65RR** … **610.65R1**, and **610.65R0** of Page **610.65P0**; Read Units **610.0RR** … **610.0R1**, and **610.0R0** of Page **610.0P0**; and so forth).

[0209] In some embodiments, each flash die includes an integer number of blocks (e.g. N blocks) and a block is a smallest quantum of erasing. In some embodiments, each block includes an integer number of pages and a page is a smallest quantum of writing. According to various embodiments, one or more of: a read unit is a smallest quantum of reading and error correction; each page includes an integer number of read units; an associated group of two or more pages includes an integer number of read units; and read units optionally and/or selectively span page boundaries.

[0210] In various embodiments, various NVM management functions (e.g. reading, recycling, erasing, and/or programming/writing) are performed in units of R-blocks. An R-block is exemplified as a logical slice or section across all die of a flash memory. For example, in a flash memory having R flash die, each flash die having N blocks, each R-block is the i^{th} block from each of the flash die taken together, for a total of N R-blocks. For another example, in a flash memory having R flash die, each with N blocks, each R-block is the i^{th} block and $(i+1)^{th}$ block from each of the flash die, for a total of N/2 R-blocks. For yet another example, in a flash memory having a plurality of dual plane devices, each R-block is the i^{th} even block and the i^{th} odd block from each of the dual plane devices.

[0211] In various embodiments with blocks treated in pairs or other associated groups as part of forming an R-block, respective pages from each block of an associated group of the blocks are also treated as a unit, at least for writing, forming a larger multi-block page. For example, continuing the foregoing dual plane example, a first page of a particular one of the even blocks and a first page of an associated one of the odd blocks are treated as a unit for writing, and optionally and/or selectively as a unit for reading. Similarly, a second page of the particular even block and a second page of the associated odd block are treated as a unit.

According to various embodiments, a page of NVM as used herein refers to one or more of: a single page of NVM; a multi-block page of NVM; a multi-block page of NVM for writing that is optionally and/or selectively treated as one or more individual pages for reading; and any other grouping or association of pages of NVM.

[0212] The figure illustrates a plurality of illustrative R-blocks, three of them explicitly (**660.0**, **660.1**, and **660.R**). Each illustrative R-block is the i^{th} block from each of the flash die, taken together. E.g. R-block **660.0** is Block **610.65B0** from Flash Die **610.65**, block 0 from Flash Die **610.64** (not explicitly illustrated), and so forth to Block **610.1B0** of Flash Die **610.1**, and Block **610.0B0** of Flash Die **610.0**. As there are N blocks per flash die, there are thus a total of N R-blocks (R-block **660.R** … R-block **660.1**, and R-block **660.0**).

[0213] Another example of an R-block is the i^{th} block and $(i+1)^{th}$ block from each of the flash die, taken together (e.g. Blocks **610.65B0** and **610.65B1** from Flash Die **610.65**, blocks 0 and 1 from Flash Die **610.64** (not explicitly illustrated), and so forth to Blocks **610.1B0** and **610.1B1** from Flash Die **610.1**, and Blocks **610.0B0** and **610.0B1** from Flash Die **610.0**). There are thus N/2 R-blocks, if there are N blocks in each flash die. Yet another example of an R-block is the i^{th} even and odd blocks from each of a plurality of dual plane devices. Other arrangements of flash die blocks for management as R-blocks are contemplated, including mapping between virtual and physical block addresses to ensure that R-blocks have one block from each die, even if some blocks are inoperable. In various embodiments, some of the N blocks in each flash die are used as spares so that the mapping between virtual and physical block addresses has spare (otherwise unused) blocks to replace defective ones of the blocks in the R-blocks.

[0214] In various embodiments, reads and/or writes of information in flash die are performed according to an order, such as a 'read unit first' order or a 'page first' order. An example of a read unit first order for read units illustrated in the figure begins with Read Unit **610.0R0** followed by **610.1R0** … **610.65R0**, **610.0R1**, **610.1R1** … **610.65R1**, and so forth, ending with **610.65RR**. An example of a page first order for read units illustrated in the figure begins with Read Unit **610.0R0** followed by **610.0R1** … **610.0RR**, **610.1R0**, **610.1R1** … **610.1RR** … **610.65R0**, **610.65R1**, and so forth, ending with **610.65RR**.

[0215] In various embodiments, a writing and/or a striping order of data within an R-block is page (e.g. lowest to highest) first, across all devices (e.g. lowest to highest numbered devices, as suggested conceptually by Striping Direction **600**), then the next highest page (across

all devices), and so forth, continuing throughout the last page of the R-block. Specifically with respect to R-block **660.0**, an example order begins with Page **610.0P0** (the first page in the first block of Flash Die **610.0**), followed by Page **610.1P0** (the first page in the first block of Flash Die **610.1**), and so forth continuing to Page **610.65P0** (the first page in the first block of Flash Die **610.65**, and the last block of R-block **660.0**). The example order continues with Page **610.0P1** (the second page in the first block of Flash Die **610.0**), followed by Page **610.1P1** (the second page in the first block of Flash Die **610.1**), and so forth continuing to Page **610.65P1** (the second page in the first block of Flash Die **610.65**). The example continues in an identical order. The example order completes with Page **610.0PP** (the last page in the first block of Flash Die **610.0**), followed by Page **610.1PP** (the last page in the first block of Flash Die **610.1**), and so forth ending with Page **610.65PP** (the last page in the first block of Flash Die **610.65**, and the last page in the last block of R-block **660.0**).

[0216] In various embodiments, Flash Die **610.65** … **610.1**, and **610.0** correspond to respective ones of one or more individual Flash Die **194**, of Fig. 1A. In some embodiments, Flash Die **610.65** … **610.1**, and **610.0** are a portion less than all of NVM **199**. For example, in various embodiments, data is striped independently across multiple groups of flash die, and each of the groups of flash die is independently accessible.

HIGHER-LEVEL REDUNDANCY TECHNIQUES

[0217] Fig. 7 illustrates selected details of various embodiments of higher-level redundancy techniques. Flash Device(s) **720** includes 64 flash die (Flash Die **610.63**, **610.62**, **610.61** … **610.0** as explicitly and implicitly illustrated in Fig. 6) and communicates via Interface Channel(s) **730**. Extra Flash Device(s) **740** includes up to two flash die (Flash Die **610.65** and **610.64** as explicitly and implicitly illustrated in Fig. 6) and communicates via Extra Interface Channel(s) **750**. The Flash Die provide storage for higher-level redundancy information and data storage (e.g. user data and/or user free space) in a storage sub-system, such as NVM in an SSD. (Examples of 'user data' in contexts of redundancy information and data storage include all data other than the redundancy information stored on flash memory for later retrieval, such as operating system data, application data, SSD management data, and so forth.) Higher-level redundancy enables, e.g., recovering from intermittent or permanent failure of one or more portions of one or more flash die, such as a failure to provide error-corrected data (e.g. via lower-level ECC functions) for a read operation or failure to properly complete a write operation.

[0218] For example, each flash die (or alternatively each block or each page within each block) is operated in a context of a Redundant Array of Silicon Independent Elements (RASIE). If a failure is detected in a particular flash die (e.g. due to an ECC-uncorrectable read error of a portion of a block of the particular die), then in response, redundant information stored in others of the flash die is used to determine information that would have been provided by the particular die. In some embodiments and/or usage scenarios, sufficient redundant information is stored to enable recovery from one failure within one flash die (during a single operation). Operation in a mode that enables recovery from a single failure includes, in some embodiments, allocating and managing space equivalent to one flash die for higher-level redundancy information, and is termed 'RASIE-1'. Operation in a mode that enables recovery from two failures includes, in some embodiments, allocating and managing space equivalent to two flash die for higher-level redundancy information, and is termed 'RASIE-2'. Operation in a mode that enables recovery from three failures includes, in some embodiments, allocating and managing space equivalent to three flash die for higher-level redundancy information, and is termed 'RASIE-3'.

[0219] In some embodiments and/or usage scenarios, managing die-level failures is an objective, and spreading information amongst die is performed. For example, higher-level redundancy information is stored in one or more die specifically allocated solely to the higher-level redundancy information. In some embodiments and/or usage scenarios, managing block-level failures is an objective, and spreading information amongst blocks within a die is performed. For example, higher-level redundancy information is stored in one or more blocks allocated to the higher-level redundancy information, the allocation being without regard to which particular die the blocks were part of. In some embodiments and/or usage scenarios, managing particular-entity-level failures includes spreading information so that no more than N elements (e.g. 1 for RASIE-1 and 2 for RASIE-2) are in any one of the particular entities. Example of the entities include a (packaged) device, a die, an R-block, a block, an R-page (described elsewhere herein), a page, cells associated with a word line, and one or more pluralities of the foregoing.

[0220] The higher-level redundancy information is computed and written in accordance with (user) data written to the flash die, and is thus available to provide information when a failure is detected. In various embodiments, the higher-level redundancy information is written to the flash die before, after, or in no particular time order with respect to writing of (user) data the higher-level redundancy information is associated with.

[0221] The figure illustrates various embodiments of RASIE operating modes, as summarized in the following table.

Usage Mode		Usage of Extra Flash Device(s) 740	Usage of Flash Device(s) 720
RASIE-1			
	1-0	-none-	**610.63** (1 die) – redundancy **610.62** … **610.0** (63 die) – data storage
	1-1	**610.64** (1 die) – redundancy	**610.63** … **610.0** (64 die) – data storage
RASIE-2			
	2-0	-none-	**610.63**, **610.62** (2 die) – redundancy **610.61** … **610.0** (62 die) – data storage
	2-1	**610.64** (1 die) – redundancy	**610.63** (1 die) – redundancy **610.62** … **610.0** (63 die) – data storage
	2-2	**610.65**, **610.64** (2 die) – redundancy	**610.63** … **610.0** (64 die) – data storage

[0222] More specifically, in RASIE-1 modes, space equivalent to one die is allocated to higher-level redundancy information. In RASIE-1 mode 1-0, Extra Flash Device(s) **740** is not used, as higher-level redundancy information is stored in one die of Flash Device(s) **720** (e.g. Flash Die **610.63**), leaving 63 of the die (Flash Die **610.62** … **610.0**) available for data storage (e.g. user data and/or user free space). In RASIE-1 mode 1-1, one die of Extra Flash Device(s) **740** is used (e.g. Flash Die **610.64**), leaving all of Flash Device(s) **720** (64 die) available for data storage.

[0223] In RASIE-2 modes, space equivalent to two die is allocated to higher-level redundancy information. In RASIE-2 mode 2-0, Extra Flash Device(s) **740** is not used, as higher-level redundancy information is stored in two die of Flash Device(s) **720** (e.g. Flash Die **610.63** and Flash Die **610.62**), leaving 62 of the die (Flash Die **610.61** … **610.0**) available for data storage. In RASIE-2 mode 2-1, one die of Extra Flash Device(s) **740** is used (e.g., Flash Die **610.64**), as higher-level redundancy information is partially stored in one die of Flash Device(s) **720** (e.g. Flash Die **610.63**), leaving 63 of the die (Flash Die **610.62** … **610.0**) available for data storage. In RASIE-2 mode 2-2, two die of Extra Flash Device(s) **740** are used (e.g. Flash Die **610.65** and Flash Die **610.64**), leaving all of Flash Device(s) **720** (64 die) available for data storage.

[0224] In some embodiments, die that are unused in all usage scenarios are unpopulated. For example, in a system operable only in RASIE 2-0 mode and RASIE 1-0 mode (but not in other RASIE modes), Extra Flash Device(s) **740** is unpopulated.

[0225] In some embodiments, higher-level redundancy information is stored entirely in "dedicated" die (e.g. Flash Die **610.63** in RASIE-1 mode 1-0 or Flash Die **610.65** and Flash Die **610.64** in RASIE-2 mode 2-2). In other embodiments, higher-level redundancy information is stored in any of the die, so for example, in RASIE-1 mode 1-0 Flash Die **610.62** is used for higher-level redundancy information, while Flash Die **610.63** and Flash Die **610.61** ... **610.0** are used for data storage. In some embodiments and/or usage scenarios, higher-level redundancy information is stored in different die (and/or portions thereof) over time, so, for example, in a first time period a first flash die holds higher-level redundancy information while in a second time period a second flash die hold higher-level redundancy information.

[0226] In various embodiments, there are a plurality of RASIE 1-0 modes (and a plurality of RASIE 2-0 modes) depending on how many flash die are usable. For example, in a first RASIE 1-0 mode (as illustrated in the table above), Flash Die **610.63** stores higher-level redundancy information, and Flash Die **610.62** ... **610.0** are available for data storage. In a second RASIE 1-0 mode, Flash Die **610.63** is no longer usable, Flash Die **610.62** stores higher-level redundancy information, and Flash Die **610.61** ... **610.0** are available for data storage, decreasing an available amount of data storage by one die. RASIE modes where a die (or any portions thereof) previously available for data storage is no longer available for data storage due to use of the die (or the portions) for higher-level redundancy information are sometimes referred to as reduced-capacity RASIE modes.

[0227] In some embodiments, higher-level redundancy information is stored using a same and/or a similar lower-level redundancy and/or error correction coding scheme as user data protected by the higher-level redundancy information. Using a lower-level redundancy and/or error correction scheme to protect the higher-level redundancy information enables determining if there is an uncorrectable error in the higher-level redundancy information, in a same and/or a similar manner that an uncorrectable lower-level error in the user data is determined.

[0228] In some embodiments, higher-level redundancy information is stored in different die for different portions of data. For instance, in some embodiments where flash die are managed in R-blocks, higher-level redundancy information is stored in different flash die for different R-blocks. For example, higher-level redundancy information for an R-block including

block 0 of Flash Die **610.0** is stored in Flash Die **610.0**, while higher-level redundancy information for an R-block including block 1 of Flash Die **610.0** is stored in Flash Die **610.1**, and so forth. In some embodiments, such as some embodiments where flash die are managed in R-blocks, higher-level redundancy information is written after data the higher-level redundancy information depends on is known and/or is written.

[0229] In some usage scenarios, one or more portions of an NVM element (e.g. a block of a device, such as Block **610.0BB** of Flash Die **610.0** of Fig. 6) are, or become during operation, inoperable. In some embodiments, the inoperable portions are mapped out via virtual and physical block addresses (e.g. via processing performed via Map **141** and/or Table **143** of Fig. 1A.). Alternatively, the inoperable portions are skipped (rather than explicitly mapped out). In some embodiments based on R-blocks, the skipping results in some of the R-blocks having differing numbers of blocks. For example, if Block **610.0B0** is defective and unusable, then R-block **660.0** has one fewer block than R-block **660.1**. The higher-level redundancy information is written in a (per R-block) variable location that is, for example, the last block of each R-block.

[0230] In various embodiments, one or more elements of Fig. 7 correspond to one or more elements of Fig. 1A. For example, Flash Device(s) **720** and Extra Flash Device(s) **740** collectively correspond to NVM **199**, and Interface Channel(s) **730** and Extra Interface Channel(s) **750** collectively correspond to Device Interfaces **190**. For another example, Flash Die **610.65** ... **610.0** collectively correspond to the instances of Flash Die **194**. For yet another example, one or more of the flash devices of Flash Device(s) **720** and/or Extra Flash Device(s) **740** correspond to one or more of the instances of Flash Devices **192**. In various embodiments, one or more elements of Fig. 1A manage higher-level redundancy information and/or recover user data based at least in part on the higher-level redundancy information in accordance with the RASIE operating modes described herein. For example, a portion of software execution capabilities of CPU **171** is used to manage computation of higher-level redundancy information according to various RASIE operating modes. For another example, Data Processing **121** and/or ECC-X **135** includes hardware elements dedicated to and/or specialized for computation of higher-level redundancy information and/or recovery of user data according to various RASIE operating modes. For yet another example, ECC **161** detects an ECC-uncorrectable (lower-level) read error of a portion of a flash die, and ECC-X **135** detects a RASIE (higher-level) read error and/or enables correction thereof.

[0231] In various embodiments, Interface Channel(s) **730** variously has one, four, eight, or 16 channels, and Extra Interface Channel(s) **750** variously has one or two channels. In

various embodiments, Flash Device(s) **720** is implemented as one, two, four, eight, or 16 devices, each having respectively 64, 32, 16, eight, and four of the Flash Die. In various embodiments, Extra Flash Device(s) **740** is implemented as one device having one or two die or as two devices each having one die. In some embodiments, the Flash Die of Extra Flash Device(s) **740** are implemented in devices that also implement the Flash Die of Flash Device(s) **720**. For example, one flash device implements 66 flash die (Flash Die **610.65** ... **610.0**). For another example, two flash devices each implement 33 flash die, e.g. in a first flash device (Flash Die **610.65** ... **610.33**) and in a second flash device (Flash Die **610.32** ... **610.0**). Other arrangements of flash die and flash devices are contemplated. In some embodiments having Extra Flash Device(s) **740** implemented in devices that also implement flash die of Flash Device(s) **720**, the flash die communicate via shared interface channels, or alternatively via interface channels dedicated to particular ones (or sets) of the flash die. While Flash Device(s) **720** and Extra Flash Device(s) **740** are illustrated with specific numbers of flash die (2 and 64, respectively), other embodiments are contemplated, such as Flash Device(s) **720** having 2, 4, 8, 16, 32, or 128 flash die, and/or Extra Flash Device(s) **740** having 0, 1, or 4 flash die.

[0232] Fig. 8 illustrates selected details of an embodiment of dynamic higher-level redundancy mode management with RASIE, such as dynamically switching between RASIE modes enabled by the various embodiments illustrated by Fig. 6 and/or Fig. 7. In some embodiments and/or usage scenarios, a form of graceful degradation is provided where a storage sub-system (e.g. an SSD) is dynamically transitioned from operating in first higher-level redundancy mode to operating in a second higher-level redundancy mode. The transition is in response to detection of a failure, such as a permanent or intermittent malfunction of an entire flash die or one or more portions thereof, or an operation (such as a read or write operation) thereto. According to various embodiments, the transition is one or more of: global for the SSD; performed on one or more subsets of the SSD; and performed on one or more R-blocks, blocks, and/or pages of the SSD. For example, if a particular block of one of the NVM devices storing RASIE-2 information fails during programming, then subsequent operation of the R-block containing the particular (failed) block transitions to a different higher-level redundancy mode (e.g. a RASIE-1 mode), whereas other R-blocks in the SSD are unaffected and continue to operate in the RASIE-2 mode.

[0233] With respect to Fig. 8, processing begins with higher-level redundancy information and data storage (e.g. user data and/or user free space) arranged in flash die in accordance with a first higher-level redundancy mode (Operate in First Higher-Level Redundancy Mode **802**). Flow then proceeds to determine if a failure has been detected

(Failure? **803**), such as a lower-level uncorrectable read error or a write/program failure. If no failure has been detected, then flow proceeds back to continue operation in the first higher-level redundancy mode. If a failure has been detected, then flow proceeds to switch from operating in the first higher-level redundancy mode to operating in a second higher-level redundancy mode (Dynamically Transition Operating Mode **809**).

[0234] The switch begins by (optionally) decreasing space available for data storage (Reduce Free Space **804**) to account for the failure. If the second higher-level redundancy mode uses sufficiently less higher-level redundancy information than the first higher-level redundancy mode, then the decreasing of available space is omitted. The switch continues by reorganizing data storage in accordance with the second higher-level redundancy mode (Rearrange Data Storage **805**). The reorganizing includes optionally moving all user data and/or user free space from the flash die where the failure occurred to another one of the flash die (user free space movement is accomplished, in some embodiments, by manipulation of pointers and/or other data structure elements). The switch further continues by selectively restoring (if possible), via the higher-level redundancy information of the first higher-level redundancy mode, any user data that was stored in the flash die where the failure occurred, and writing the restored user data to another one of the flash die, in accordance with the second higher-level redundancy mode (Recover/Store Failed User Data **806**). The restoring is omitted if the failure is a write/program failure. The switch further continues by optionally computing and writing to the flash die higher-level redundancy information in accordance with the second higher-level redundancy mode (Determine/Store Revised Higher-Level Redundancy Information **807**). The computing and the writing are omitted if the second higher-level redundancy mode is operable with higher-level redundancy information that was previously in place due to operating in the first higher-level redundancy mode. Then operation begins in the second higher-level redundancy mode (Operate in Second Higher-Level Redundancy Mode **808**).

[0235] The failure detection (Failure? **803**) is via one or more of: lower-level redundancy and/or error correction (e.g. in accordance with one or more ECC techniques), higher-level redundancy and/or error correction (e.g. in accordance with one or more RASIE techniques), and failing status reported by one or more of the flash die or portions thereof. For example, more than a threshold number of lower-level error corrections of reads within a particular portion (e.g. R-block, block, R-page, page, read unit, or cells associated with a word line) of a particular flash die optionally and/or conditionally results in the particular flash die (or the particular portion) being treated as failed and a higher-level redundancy mode switch is performed so that the failed flash die (or portion) is no longer used. For another example, if a

higher-level error correction fails, then an appropriate one of the flash die (or portion thereof) is treated as failed and a higher-level redundancy mode switch is performed so that the failed flash die (or portion) is no longer used. For yet another example, if a flash die returns a program failure status (indicating that a write operation was unsuccessful), then an appropriate block of an appropriate one of the flash die is treated as failed, and optionally and/or conditionally a higher-level redundancy mode switch is performed so that the failed flash die (or alternatively a portion thereof) is no longer used.

[0236] In some embodiments, a failed block is replaced by remapping via virtual and physical block addresses (e.g. via processing performed via Map **141** and/or Table **143** of Fig. 1A). A spare block from a pool of spare blocks is mapped in place of the failed block. Any contents written in the failed block are copied to the replacement block, and writing proceeds in the spare block from where the failure occurred in the failed block.

[0237] In some embodiments, a failed block is skipped (rather than explicitly remapped), resulting in a "hole" that optionally and/or conditionally results in a higher-level redundancy mode switch when the R-block that the hole is in is next erased (in preparation for re-writing). If the hole is in a location for data storage, then no switch is made, and the hole remains. If the hole is in a location for higher-level redundancy information, then the higher-level redundancy information is stored in another location, and optionally the higher-level redundancy mode is switched.

[0238] In some embodiments and/or usage scenarios, restoration of user data stored in the flash die where the failure occurred is not possible. For example, if the failure is due to some types of failures detected via higher-level redundancy and/or error correction and/or some types of failing status reported by one or more of the flash die or portions thereof, then some user data is lost.

[0239] In some embodiments, processing of Fig. 8 is performed in a context of (e.g. an SSD controller) dynamically transitioning between higher-level redundancy modes in response to a plurality of failures. Specifically, the SSD controller begins operating in a first higher-level redundancy mode and dynamically transitions to a second higher-level redundancy mode in response to a first failure, and subsequently dynamically transitions from the second higher-level redundancy mode to a third higher-level redundancy mode in response to a second failure, and so forth. For instance, an SSD controller operates various flash die in accordance with a RASIE-2 mode 2-2 and dynamically transitions the operation to be in accordance with a RASIE-2 mode

2-1 in response to a first failure. Subsequently, the SSD controller dynamically transitions the operation to be in accordance with a RASIE-2 mode 2-0 in response to a second failure. Further subsequently, the SSD controller dynamically transitions the operation to be in accordance with a RASIE-1 reduced-capacity mode 1-0 in response to a third failure (the reduced-capacity mode 1-0 being similar to RASIE-1 mode 1-0 except with one flash die used for higher-level redundancy information and 62 flash die used for data storage).

[0240] As a specific example, consider an SSD controller (such as SSD Controller **100** of Fig. 1A) coupled to the elements of Fig. 7, initially operating in RASIE-2 mode 2-2 (e.g. higher-level redundancy information in Flash Die **610.65** and Flash Die **610.64**, and data storage in Flash Die **610.63** … **610.0**), corresponding to operating in the first higher-level redundancy mode. Then a read, or alternatively a write, of one or more of the Flash Die is performed. The read results in an uncorrectable (lower-level) ECC failure, or alternatively the write is unsuccessful, in a portion of a particular one of the Flash Die (e.g. a page of Flash Die **610.62** used for user data and/or user free space). In response, the SSD controller dynamically switches from operating in RASIE-2 mode 2-2 to RASIE-2 mode 2-1, no longer using any of Flash Die **610.62**. As operation in RASIE-2 mode 2-1 provides 63 die for data storage (versus 64 die in RASIE-2 mode 2-2), space available for data storage is decreased from 64 die to 63 die, and user data and/or user free space is moved accordingly. E.g., all user data from Flash Die **610.62** is moved to portions of Flash Die **610.63** and Flash Die **610.61** … **610.0** in accordance with user free space. Any user data in the page having the uncorrectable ECC failure is recovered based on the higher-level redundancy information in Flash Die **610.65** and/or Flash Die **610.64**. Higher-level redundancy information based on data storage in Flash Die **610.63** and Flash Die **610.61** … **610.0** and in accordance with RASIE-2 mode 2-1 is computed and stored in Flash Die **610.65** and/or Flash Die **610.64**. The SSD controller then operates in RASIE-2 mode 2-1 (higher-level redundancy information in Flash Die **610.65** and Flash Die **610.64**, and data storage in Flash Die **610.63** and Flash Die **610.61** … **610.0**).

[0241] While the foregoing described several embodiments of dynamic higher-level redundancy mode management with independent silicon elements of a quantum of an entire flash die, other embodiments implement dynamic higher-level redundancy mode management with independent silicon elements that are portions of a die, such as one or more read units, word line associated cells, pages, R-pages, blocks, or R-blocks. The foregoing described several embodiments of concurrently active higher-level redundancy modes using a single granularity of an R-block. E.g. higher-level redundancy mode varies on an R-block basis (each R-block is operated entirely in a single respective higher-level redundancy mode). However, other

embodiments implement concurrently active higher-level redundancy modes using a plurality (mixture) of granularities, such as R-block granular in combination with page granular. E.g. higher-level redundancy mode varies in one region on an R-block basis, and varies in another region on a page basis (each page is operated entirely in a single respective higher-level redundancy mode).

[0242] In various embodiments, processing of or relating to one or more elements of Fig. 8 is performed entirely or in part by one or more elements (or portions thereof) of Fig. 1A. For example, a portion of software execution capabilities of CPU **171** is used to manage dynamic transitioning between higher-level redundancy modes, such as by directing decreasing space available for data storage or directing reorganizing data storage. For another example, Data Processing **121** and/or ECC-X **135** includes hardware elements dedicated to and/or specialized for computation of higher-level redundancy information in accordance with a 'target' redundancy mode. For yet another example, ECC **161** implements lower-level (e.g. ECC) error correction and detection of uncorrectable errors, while ECC-X **135** implements higher-level (e.g. RASIE) error correction and detection of uncorrectable errors and/or memory element failures. For another example, all or any portions of functionality relating to dynamic transitioning between (higher-level) redundancy modes is performed by one or more portions of ECC-X **135**.

HIGHER-LEVEL REDUNDANCY AND ADAPTIVE LOWER-LEVEL CODE RATES

[0243] In some embodiments and/or usage scenarios, lower-level redundancy and/or error correction uses an adaptive code rate (e.g. an adaptive ECC technique using a variable code rate). For instance, a first read unit is managed with a first code rate that provides relatively more usable data bits than a second read unit that is managed with a second code rate. In some embodiments and/or usage scenarios with lower-level redundancy and/or error correction using a variable code rate, higher-level redundancy information is stored in portions of independent silicon elements (such as portions of flash die) that are managed with lower-level redundancy and/or error correction that provides relatively more usable data bits or the most usable data bits with respect to data storage protected by the higher-level redundancy information. The portion(s) where the higher-level redundancy information is stored varies, in various embodiments and/or usage scenarios, on a per R-block basis, on a per-die basis, dynamically over time, or any combination thereof. In various embodiments, one or more die, R-blocks, blocks, and/or pages are selected for storage of higher-level redundancy data based on having the most usable data with respect to lower-level error correction code rate.

[0244] For example, higher-level redundancy information is stored at varying locations (such as block locations) on a per R-block basis, the locations (such as blocks) being those having the most usable data based on lower-level error correction code rate. In an arbitrary example, consider a RASIE-2 operating mode scenario in the context of Fig. 7, wherein the collection of 66 flash die is treated as a logical "stack" of up to M R-blocks of one-block height each, where M is the number of blocks per flash die. (In the most conceptually straightforward case, each R-block in the stack is made up of the same physical block number from each die, but to accommodate failed blocks this constraint is lifted in at least some embodiments. In yet other embodiments, the constraint is maintained, but "holes" are accommodated corresponding to the failed blocks.) Thus, each R-block has up to 66 blocks, corresponding to one block from each of Flash Die **610.0** through Flash Die **610.65**. (In Fig. 7, while only some flash die in the range **610.0** through **610.65** are explicitly enumerated, it is understood by the use of ellipsis that all flash die in this range are implicitly illustrated.) The higher-level redundancy information is written into whichever blocks of each R-block have the most useable data based on lower-level error correction code rate. If for example in a first R-block, blocks corresponding to Flash Die **610.15** and **610.49** happen to have the most usable data based on lower-level error correction code rate, then higher-level redundancy information is written into those blocks (corresponding to Flash Die **610.15** and **610.49**). While if in a second R-block, blocks corresponding to Flash Die **610.9** and **610.35** happen to have the most usable data based on lower-level error correction code rate, then higher-level redundancy information is written into those blocks (corresponding to Flash Die **610.9** and **610.35**). In some embodiments, the higher-level redundancy data is written after all other data in an R-block is known.

[0245] Fig. 9 illustrates an embodiment of Read Units (**911**, **931** … **951**, **971**) having lower-level redundancy information of adaptive (e.g. differing and/or varying over time and/or by location) code rates protected by higher-level redundancy information stored in one or more of the read units. Each of the read units has a portion enabled and/or allocated to contain User Data (**911.U**, **931.U** … **951.U**, **971.U**), and a remaining portion enabled and/or allocated to contain lower-level redundancy information, such as check bits of an ECC code as Lower-Level ECC (**911.E**, **931.E** … **951.E**, **971.E**).

[0246] In the figure, vertical dimensions are relatively to scale and indicative of relative size. Thus Read Units **911** and **951** are of a same first size (in some embodiments, all read units are a same size for all blocks of all NVM devices), Lower-Level ECC portions **931.E** and **951.E** are of a same second size, and Lower-Level ECC portions **911.E** and **971.E** are of a same third

size. Read Unit **931** is larger than Read Units **911** and **951** that are in turn larger than Read Unit **971**. User Data portion **931.U** is larger than User Data portion **951.U**. User Data portion **951.U** is larger than User Data portion **911.U**. Lower-Level ECC portion **951.E** is smaller than Lower-Level ECC portion **911.E**.

[0247] As illustrated in the figure, respective read units have respective sizes, e.g. per respective block of NVM, enabling varying lower-level code rates as used in the respective blocks. More specifically, Read Units **931** and **951** have a same amount of lower-level redundancy information (**931.E** and **951.E** are a same size), but a lower-level code rate of Read Unit **931** is higher than a lower-level code rate of Read Unit **951**, as Read Unit **931** contains more User Data (**931.U**) than Read Unit **951** (containing User Data **951.U**).

[0248] As illustrated in the figure, respective read units have respective sizes of user data, enabling various sizes of user data in each of two or more read units of a same size. For example, a size of user data is varied to change a lower-level code rate used in a particular read unit. More specifically, Read Units **951** and **911** have a same size, but have different respective amounts of User Data (**951.U** and **911.U**), and thus different respective amounts of lower-level redundancy information (**951.E** and **911.E**), enabling Read Unit **951** to have a higher lower-level code rate than Read Unit **911**.

[0249] In some embodiments and/or usage scenarios, varying and/or changing a lower-level code rate advantageously enables providing a sufficient amount of lower-level ECC information to achieve lower-level error correction requirements while maximizing an amount of user data.

[0250] In some embodiments with a varying amount of user data in read units, higher-level redundancy information is stored in a one or more of the read units having a largest amount of user data. For example in Fig. 9, using a RASIE-1 mode, higher-level redundancy information is stored in User Data **931.U**, and using a RASIE-2 mode, higher-level redundancy information is stored in User Data **931.U** and User Data **951.U**. Storing the higher-level redundancy information in read units with the largest amount of user data (among the read units protected by the higher-level redundancy information) ensures that there is sufficient higher-level redundancy information to protect the user data in all of the other read units.

[0251] According to various embodiments, one or more techniques are used to determine which of one or more read units among a number of read units protected by higher-

level redundancy information are used to store higher-level redundancy information. In a first example, the latest-written one (for RASIE-1) or two (for RASIE-2) read units that have a largest amount of user data are used. In a second example, the earliest-written one (for RASIE-1) or two (for RASIE-2) read units that have a largest amount of user data are used. Similarly, any technique to deterministically select one or more read units having a largest amount of user data so as to protect all of the remaining user data in the other read units is within the scope of the techniques considered herein.

HIGHER-LEVEL REDUNDANCY INFORMATION COMPUTATION TECHNIQUES

[0252] In various embodiments and/or usage scenarios, higher-level redundancy information is computed with a variety of techniques, such as via parity, RS, and/or weighted-sum techniques. For example, in some higher-level redundancy modes enabling recovery from one (lower-level) failure (e.g. RASIE-1), higher-level redundancy information is computed via parity techniques. For another example, in some higher-level redundancy modes enabling recovery from two (lower-level) failures (e.g. RASIE-2), higher-level redundancy information is computed via a combination of parity and RS techniques. A first portion of the higher-level redundancy information is computed using parity coding and a second portion is computing using RS coding. For yet another example, in some higher-level redundancy modes enabling recovery from two failures (e.g. RASIE-2), higher-level redundancy information is computed via a combination of parity and weighted-sum techniques. A first portion of the higher-level redundancy information is computed using parity coding and a second portion is computing using weighted-sum coding. The higher-level redundancy information is managed, e.g. via reads and writes of pages of NVM, using lower-level failure detection techniques (such as ECC) identical to or similar to lower-level failure detection techniques used for pages of the NVM available for storing data protected by the higher-level redundancy information.

[0253] As a specific example for RASIE-2, a first page of higher-level redundancy information is computed using parity coding via an XOR of all first pages in a stripe across an R-block. More specifically, an XOR is performed of all of the first bytes of all of the first pages in the stripe across the R-block, resulting in a first byte of the first page of higher-level redundancy information. Similarly, a second byte of higher-level redundancy information is formed by XORing all of the second bytes of all of the first pages in the stripe across the R-block, and so forth for all of the bytes of all of the first pages in the stripe. A second page of higher-level redundancy information is computed using a weighted-sum technique as follows.

[0254] Arithmetic is performed over a finite field, such as a Galois Field (used as an example). Examples assume data being operated on is byte-wide, and thus a field such as GF(256) is used. In various embodiments, data is operated on in any units.

[0255] Each page in a stripe is assigned a unique non-zero "index". The values of the indices are chosen to simplify implementation complexity, and are not related to any form of generator polynomial. For example, the pages are labeled (e.g. by software) by die location in a stripe from 0 to N-1, and a suitable value for the indices is the ones-complement of the die number (ensured to be non-zero provided N < 255). Another selection of index values is the lowest-weight (fewest number of set bits or fewest number of clear bits) non-zero integers, e.g. to reduce and/or minimize hardware costs. Selecting gray-coded values for the indices, in some embodiments and/or usage scenarios, minimizes transitions and/or reduces power as pages are processed.

[0256] The index values are not selected according to finite field arithmetic, but are selected according to other principles. Notwithstanding this, each index value corresponds to a non-zero element in the finite field. Assume that page i has index value Ki (and page j has index value Kj). The weighted-sum redundancy is the (GF field) sum (over corresponding bytes Pi from each page i) of Ki*Pi, each byte multiplied (over the GF field) by its index value.

[0257] Thus weighted-sum redundancy information is computed for each byte as:

R0 = sum over all corresponding bytes Pi;

R1 = sum over all corresponding bytes Ki*Pi;

R0 is the XOR of all the corresponding bytes; and

R1 is a weighted sum of the bytes, where the weights are selected as the index values.

The foregoing computation is iterated for each of the corresponding bytes in a page, producing corresponding pages of bytes for each of R0 and R1. In the following discussion, R0 and R1 are described in some contexts respectively as single elements (e.g. each being a single byte) for clarity of exposition, but as in the foregoing computation, each are representative of a respective page of elements (e.g. each being a page of bytes).

[0258] Pi represents a byte in page i, and Pj represents a byte in page j. Processing is described with respect to one stripe of corresponding bytes from each page, and iteration is performed over all corresponding bytes. If some pages are "shorter" than others, due to for example to having a different (lower-level) redundancy code rate, then the shorter pages are

zero-padded (or padded by any known value used the same way on encode and decode) so that all pages that are processed effectively have the same size.

[0259] Summations to compute R0 and R1 are performable in any order, via various serial and/or parallel computations, according to various embodiments. Pages do not have to be processed in any specific order as whether Ki*Pi is added in prior to or subsequent to Kj*Pj has no effect on the result in R1. Computation of R0 and R1 values corresponding to various bytes of a page of redundancy information are independent of each other and are computable in any order, via various serial and/or parallel computations, according to various embodiments. Further, subtracting Ki*Pi from R1 (and subtracting Pi from R0) enables "backing out" of computation effects on pages. Since over a GF field, addition and subtraction are both XOR (thus subtracting is equivalent to simply adding in a second time), in some embodiments and/or usage scenarios, no special hardware is needed for GF field implementations (e.g. a logical XOR capability is sufficient) to "back out" a page.

[0260] In the event of an uncorrectable lower-level error, higher-level error correction begins, in some embodiments, by re-computing R0 and R1, but by omitting page(s) (sometimes referred to as column(s)) that have uncorrectable lower-level errors. Correction proceeds by subtracting the recomputed R0 from the original R0 to produce ΔR0, and subtracting the recomputed R1 from the original R1 to produce ΔR1.

[0261] If there are no uncorrectable lower-level errors, then the recomputed R0 and R1 are both zero. If there are uncorrectable lower-level errors, then the recomputed R0 and R1 (after the subtraction) reflect the "missing" data (that was not added in the second time, but was present in the original values).

[0262] If there is one uncorrectable lower-level error, then the recomputed R0 is used to correct the error (and the recomputed R1 is not needed).

[0263] If there are two uncorrectable lower-level errors, then the recomputed R0 and R1 are used to correct the errors. If both the pages of R0 and R1 values are the pages with uncorrectable lower-level errors, then no correction of data storage pages is needed. If the page of R1 values is one of the pages with uncorrectable lower-level errors, then correction is via R0 (the recomputed R0 value is the value of the data storage page with uncorrectable lower-level errors).

[0264] If there are two uncorrectable lower-level errors in data storage pages, or if the R0 page is one of the pages with uncorrectable lower-level errors, then correction starts by computing ΔR0 and ΔR1 as above. If the R0 page is one of the pages with uncorrectable lower-level errors, computation of the ΔR0 page is optionally omitted. If page i and page j are pages with uncorrectable lower-level errors, then the recomputed ΔR0 = Pi + Pj, and the recomputed ΔR1 = Ki*Pi + Kj*Pj. Equation solving produces:

Pi = (ΔR1 – Kj*ΔR0)/(Ki – Kj)

Pj = ΔR0 - Pi

If R0 is one of the pages with uncorrectable lower-level errors, then (because R0 is not included in R1), ΔR1 = Ki*Pi, or Pi = ΔR1/Ki; the same result obtained by setting Kj = 0 in the formulas above (to ignore ΔR0).

[0265] In an alternative embodiment, a finite field defined by integers mod p, where p is a prime, is used instead of a Galois Field. The computations are identical to those described above, with addition being integer addition mod p, and multiplication being integer multiplication mod p. For example, if pages are pages of bytes, a finite field of integers mod 257 is used. All user data bytes are in a range 0 – 255 and are stored in one byte each. R1 results, however, have values ranging from 0 – 256, requiring more than one byte for representation. There are many ways to encode values from 0 – 256 to minimize storage space and enable the R1 page to be stored with reduced overhead. For example, values 0 and 256 are stored as nine-bit sequences 000000000 and 000000001 respectively, and all other values are stored in eight bits. Given a random distribution of R1 values, storage overhead is < 0.1%. As described with respect to Fig. 9, the R1 page is selected to have a largest amount of user data, enabling the storage overhead to be hidden in some usage scenarios.

[0266] Fig. 10 illustrates selected details of an embodiment of higher-level redundancy information result and data source correspondences, for instance as used by RASIE-2 mode 2-2 in a context such as Fig. 7 and as further illustrated in Fig. 6. Fig. 10 explicitly illustrates Flash Die **610.0**, **610.63**, **610.64** and, **610.65**, and by ellipsis (…) implicitly illustrates Flash Die **610.1** … **610.62**. Selected details of blocks, pages, and bytes within pages of the Flash Die are illustrated. A first portion of higher-level redundancy information is illustrated as R0 **1010** (stored in Flash Die **610.64**), and is computed using parity coding. A second portion of higher-level redundancy information is illustrated as R1 **1011** (stored in Flash Die **610.65**), and is computed using weighted-sum coding. Storage for data information is illustrated as Data **1019** (stored in Flash Die **610.0** … **610.63**).

[0267] Dashed-arrow **1001** conceptually indicates a two-byte redundancy computation result (one byte for each of R0 **1010** and R1 **1011**) based on corresponding first bytes of all first pages (across all flash die) of Data **1019**. As illustrated, the two-byte result is the first byte on each of the first R0 and R1 pages. Dashed arrow **1002** conceptually indicates a two-byte redundancy computation result (one byte for each of R0 **1010** and R1 **1011**) based on corresponding last bytes of all first pages of Data **1019**. As illustrated, the two-byte result is the last byte on each of the first R0 and R1 pages. Dashed-arrow **1003** conceptually indicates a two-page redundancy computation result (one page for each of R0 **1010** and R1 **1011**) based on corresponding last pages of the first blocks (across all flash die) of Data **1019**.

[0268] Note that as described elsewhere herein, in some embodiments higher-level redundancy information is stored in different die for different portions of data. Thus, R0 and R1, in some embodiments, are stored across various die, rather than two "dedicated" die, as illustrated in Fig. 10.

[0269] Fig. 11 illustrates selected details of an embodiment of higher-level redundancy information computations, for instance as used by RASIE-2 mode 2-2 in a context such as Fig. 7 and as further illustrated in Fig. 6 and Fig. 10, with various operating conditions according to various embodiments. More specifically, Fig. 11 illustrates parity coding computation for a byte of R0 and weighted-sum coding computation for a byte of R1, in accordance with, e.g., the two-byte redundancy computation result illustrated conceptually by dashed-arrow **1001** of Fig. 10. The operating conditions include one or more of: performing arithmetic over a finite field (such as a Galois Field), indices (corresponding to dummy summation variable "i" values in Fig. 11) being ones-complement of flash die number (or any other unique and non-zero numbering), and indices corresponding to non-zero elements in the finite field. The computation as illustrated in Fig. 11 is repeated for all of the bytes of R0 and R1, based on the corresponding data bytes. Note that non-zero indices enable R1 values to include a contribution from every element of Pi.

[0270] Thus there are no dependencies on computing any of the bytes of R0 on each other or between any of the bytes of R1. Therefore various embodiments are contemplated where R0 and R1 values are computed entirely (massively) in parallel, partially in parallel, or entirely in serial. For example, all of the R0 and/or R1 bytes of a page are computed in parallel. For another example, all of the R0 bytes of a page are computed in parallel followed (or preceded) by computing in parallel all of the R1 bytes of a page.

[0271] Further, there are no ordering requirements on computing any of the bytes of R0 with respect to each other or with respect to computing any of the bytes of R1. Therefore various embodiments are contemplated where R0 and R1 values are computed entirely in-order with respect to each other, in-order with respect to each other separately (e.g. R0 computations are in-order with respect to each other but with no ordering with respect to R1, and vice-versa), or with no particular ordering (e.g. entirely out-of-order, unordered, or randomly ordered). For example, all of the R0 bytes of a page are computed in a particular order (e.g. from lowest byte to highest byte), followed by all of the R1 bytes of the page in the particular order. For another example, all of the R0 bytes of a page are computed in the particular order, and independently all of the R1 bytes of the page are computed in the particular order. For yet another example, all of the R0 bytes of a page as well as all of the R1 bytes of a page are computed in no particular order with respect to each other (e.g. as data operands become available).

[0272] For yet another example, all of the R0 and R1 bytes of one or more pages are computed in an order determined by an order of completion of one or more read operations performed on one or more NVMs (each having, e.g., one or more flash die), the read operations for reading the data bytes referenced by the summation and weighted-summation computations (Pi) illustrated in Fig. 11. Performing the computations in the order determined by the completion of the read operations enables, in some embodiments and/or usage scenarios, reduction or elimination of buffering between the NVMs and elements enabled to perform the computations. Performing the computations in the order determined by the completion of the read operations enables, in some embodiments and/or usage scenarios, reduction in memory bandwidth used to perform the computations, such as memory bandwidth of the NVMs.

[0273] For yet another example, all of the R0 and R1 bytes of one or more pages are computed in an order determined by an order of data returned and/or data available in response to read operations performed on one or more NVMs (each having, e.g., one or more flash die), the data returned and/or data available corresponding to the data bytes referenced by the summation and weighted-summation computations (Pi) illustrated in Fig. 11. In some embodiments and/or usage scenarios, performing the computations in the order determined by the data returned and/or data available reduces or eliminates buffering between the NVMs and elements enabled to perform the computations. In some embodiments and/or usage scenarios, performing the computations in the order determined by the data returned and/or data available reduces memory bandwidth used to perform the computations, such as memory bandwidth of the NVMs. In some embodiments, the read operations are performed in a particular order (e.g. from

lowest byte to highest byte of Pi), while in other embodiments, the read operations are performed in no particular order.

[0274] Fig. 12 illustrates selected details of an embodiment of recovery from one (lower-level) failure (during a single operation), for instance in a context such as associated with Fig. 11, and where the lower-level failure has occurred on page j. Note that if the lower-level failure is on an R0 or an R1 page, then R0 (or R1) is re-determined as described by Fig. 11. Fig. 12 illustrates a computation for one byte of a recovered value for page j (note that the summation omits page j where the failure occurred). The computation as illustrated in Fig. 12 is repeated to determine recovered values for all of the bytes of page j, based on corresponding bytes of R0 and corresponding bytes from all of the data pages except for page j.

[0275] Thus there are no dependencies on computing any of the bytes of the recovered values of page j on each other. Therefore various embodiments are contemplated where Pj recovery values are computed ranging from highly parallel to highly serial, similarly to the aforementioned computations for R0 and R1 values. Further, there are no order requirements on computing any of the recovery values of page j with respect to each other. Therefore various embodiments are contemplated where Pj recovery values are computed in varying orderings with respect to each other, similarly to the aforementioned computations for R0 and R1 values.

[0276] Some exemplary embodiments perform computations related to determining recovery values in orderings at least in part determined by ordering of completion of one or more read operations performed on one or more NVMs (each having, e.g., one or more flash die), the read operations for reading the NVMs to obtain R0 and/or Pi values as illustrated by Fig. 12. Performing the computations in the order of the completion of the read operations enables, in some embodiments and/or usage scenarios, reduction or elimination of buffering between the NVMs and elements enabled to perform the computations. Performing the computations in the order of the completion of the read operations enables, in some embodiments and/or usage scenarios, reduction in memory bandwidth used to perform the computations, such as memory bandwidth of the NVMs.

[0277] Some exemplary embodiments perform computations related to determining recovery values in orderings at least in part determined by ordering of data returned and/or data available from one or more NVMs (each having, e.g., one or more flash die), the data returned and/or data available being in response to read operations performed on the NVMs to obtain R0 and/or Pi values as illustrated by Fig. 12. In some embodiments and/or usage scenarios,

performing the computations in the order of the data returned and/or data available from the read operations reduces or eliminates buffering between the NVMs and elements enabled to perform the computations. In some embodiments and/or usage scenarios, performing the computations in the order of the data returned and/or data available from the read operations reduces memory bandwidth used to perform the computations, such as memory bandwidth of the NVMs. In some embodiments, the read operations are performed in a particular order (e.g. from lowest byte to highest byte of Pi), while in other embodiments, the read operations are performed in no particular order.

[0278] Figs. 13A-13D illustrate selected details of an embodiment of recovery from two (lower-level) failures (during a single operation), for instance in a context such as associated with Fig. 11, and where the lower-level failures have occurred on pages m and n. Note that if the lower-level failures are on R0 and R1 pages, then R0 and R1 are unneeded for data recovery, and there is no processing to recover data. Otherwise, note that if one of the lower-level failures is an R1 page, then data recovery proceeds as described by Fig. 12. If neither of the lower-level failures are on R0 or R1 pages, then recovery of Pm and Pn values proceeds as follows. Fig. 13A illustrates computations for one byte of a revised R0 as R0' and for one byte of a revised R1 as R1' (note that the summations omit pages m and n where the failures occurred). Fig. 13B illustrates equalities relating one byte of the original redundancy information (R0, R1) to the revised redundancy information (R0', R1'), and the terms omitted from the summations used to form the revised R0 and R1 (Pm + Pn and Km*Pm + Kn*Pn). Fig. 13C illustrates an algebraic rearrangement of Fig. 13B, with introduced terms delta R0 (ΔR0) and delta R1 (ΔR1). Fig. 13D illustrates a solution of Fig. 13C for Pn and Pm, and thus represents computations for one byte of a recovered value for page n and one byte of a recovered value for page m, based on the corresponding bytes of R1, R0, and corresponding bytes from all of the data pages except for pages m and n. Note that unique indices enable a non-zero denominator, as Kn is unique with respect to Km.

[0279] In various embodiments, computations as represented by Figs. 13A-13D are performed to determine one byte of a recovered value for page m and one byte of a recovered value for page n. The computations are repeated to determine recovered values for all of the bytes of pages m and n, based on corresponding bytes of R0, R1, and corresponding bytes from all of the data pages except for pages m and n.

[0280] Thus there are no dependencies on computing any of the bytes of the recovered values of pages m or n on each other. Therefore various embodiments are contemplated where

Pm and/or Pn recovery values are computed ranging from highly parallel to highly serial, similarly to the aforementioned computations for Pj recovery values. Further, there are no order requirements on computing any of the recovery values of page m and/or page n with respect to each other. Therefore various embodiments are contemplated where Pm and/or Pn recovery values are computed in varying orderings with respect to each other, similarly to the aforementioned computations for Pj recovery values.

[0281] Some exemplary embodiments perform computations related to determining recovery values (such as computations for R0' and R1') in orderings at least in part determined by ordering of completion of one or more read operations performed on one or more NVMs (each having, e.g., one or more flash die), the read operations to obtain any one or more of values illustrated as sources for computations in any of Figs. 13A-13D. Performing the computations in the order of the completion of the read operations enables, in some embodiments and/or usage scenarios, reduction or elimination of buffering between the NVMs and elements enabled to perform the computations. Performing the computations in the order of the completion of the read operations enables, in some embodiments and/or usage scenarios, reduction in memory bandwidth used to perform the computations, such as memory bandwidth of the NVMs.

[0282] Some other exemplary embodiments perform computations related to determining recovery values (such as computations for R0' and R1') in orderings at least in part determined by ordering of data returned and/or data available from one or more NVMs (each having, e.g., one or more flash die), the data returned and/or data available being in response to read operations performed on the NVMs to obtain any one or more of values illustrated as sources for computations in any of Figs. 13A-13D. In some embodiments and/or usage scenarios, performing the computations in the order of the data returned and/or data available from the read operations reduces or eliminates buffering between the NVMs and elements enabled to perform the computations. In some embodiments and/or usage scenarios, performing the computations in the order of the data returned and/or data available from the read operations reduces memory bandwidth used to perform the computations, such as memory bandwidth of the NVMs. In some embodiments, the read operations are performed in a particular order (e.g. from lowest byte to highest byte of Pi), while in other embodiments, the read operations are performed in no particular order.

[0283] Figs. 14A and 14B illustrate selected details of an embodiment of computing higher-level redundancy information with respect to pages received from NVMs. Fig. 14A

illustrates selected details of an embodiment of sending a plurality of read commands to one or more NVMs via issuing (Issue Read Operation **1402A**), checking if all of the commands have been sent (All Issued? **1403A**), and if not, then looping back to send another of the commands. Note that other embodiments are contemplated where a plurality of commands are issued at a time, rather than one at a time.

[0284] Fig. 14B illustrates selected details of an embodiment of processing pages received from the NVMs in response to the read commands sent as illustrated in Fig. 14A. A check is made to determine if a page is available (Page Ready? **1402B**). If not, then processing loops back to perform the check again. If a page is available, then higher-level redundancy information processing relating to the page is carried out (Perform Page-Based Computations **1403B**). Then a check is made to determine if all pages have been processed (Pages Finished? **1404B**). If so, then processing is complete (End **1499B**), otherwise flow loops back to determine if another page is available.

[0285] Other than reception of pages in response to the commands sent as illustrated in Fig. 14A, the processing illustrated in Fig. 14B is independent of processing illustrated in Fig. 14A. In various scenarios, arrival order of the pages varies according to the NVM type, state, operating environment, and other factors, and in some circumstances is different than sending order or arrival order of the read commands corresponding to the arriving pages. As processing of Fig. 14A is independent of Fig. 14B (other than page data arrival being dependent on a corresponding read command being sent), in some embodiments and/or usage scenarios, read commands are being sent (Fig. 14A) while read data is being received/processed (Fig. 14B). In some embodiments and/or usage scenarios, some of the pages are provided from one or more buffers rather than being requested via read commands directed to the NVMs, for example if a particular page is present in a particular buffer before a read command for the particular page is to be sent. In some embodiments and/or usage scenarios, pages other than in response to the commands sent are provided by the NVMs intermixed with the pages provided in response to the commands sent, e.g. pages provided in response to read commands sent for other activities.

[0286] In various embodiments, computations for R0 and R1, as illustrated by Fig. 11, are performed at least in part as illustrated by Figs. 14A and 14B. For a first example, read commands for all data pages necessary to compute corresponding R0 and R1 pages are sent to one or more NVMs as illustrated by Fig. 14A. The pages of data received in response to the read commands are processed as the pages are received to compute the R0 and the R1 pages, as illustrated by Fig. 11. For a second example, read commands for a pair (corresponding, e.g., to

two planes of a dual-plane NVM) of R0 and R1 pages are sent to one or more NVMs as illustrated by Fig. 14A. The pages of data received in response to the read commands are processed as the pages are received to compute the R0 and the R1 pages, as illustrated by Fig. 11.

[0287] In various embodiments, computations for Pj, as illustrated by Fig. 12, are performed at least in part as illustrated by Figs. 14A and 14B. For a first example, read commands for all data pages necessary to compute a particular Pj page are sent to one or more NVMs as illustrated by Fig. 14A. The pages of data received in response to the read commands are processed as the pages are received to compute the Pj page, as illustrated by Fig. 12. For a second example, read commands for a pair of Pj pages (corresponding, e.g., to two planes of a dual-plane NVM) are sent to one or more NVMs as illustrated by Fig. 14A, and the pages of data received are processed as received, as illustrated by Fig. 14B, to compute the pair of Pj pages.

[0288] In various embodiments, computations related to determining recovery values (such as computations for R0' and R1'), as illustrated by any of Figs. 13A-13D, are performed at least in part as illustrated by Figs. 14A and 14B. For a first example, read commands for all data pages necessary to compute a particular R0' page and a particular R1' are sent to one or more NVMs as illustrated by Fig. 14A. The pages of data received in response to the read commands are processed as the pages are received to compute the R0' and R1' pages, as illustrated by Fig. 13A. For a second example, read commands for a pair of R0' and R1' pages (corresponding, e.g., to two planes of a dual-plane NVM) are sent to one or more NVMs as illustrated by Fig. 14A, and the pages of data received are processed as received, as illustrated by Fig. 14B, to compute the pair of R0' and R1' pages.

[0289] Figs. 15A-15C illustrate selected details of an embodiment of backing out of a computation of higher-level redundancy information with respect to a write provided to NVMs, for instance in a context such as associated with Fig. 11. Fig. 15A illustrates selected details of an embodiment of sending a plurality of write commands to one or more NVMs via issuing (Issue Write Operation **1502A**), checking if all of the commands have been sent (All Issued? **1503A**), and if not, then looping back to send another of the commands. Note that other embodiments are contemplated where a plurality of commands are issued at a time, rather than one at a time.

[0290] Fig. 15B illustrates selected details of an embodiment of processing write completion and status information received from the NVMs in response to the write commands sent as illustrated in Fig. 15A. A check is made to determine if a write has completed without errors (Write OK? **1502B**). If so, then a check is made to determine if all writes have been completed (Writes Finished? **1504B**). If so, then processing is complete (End **1599B**). If a write has been completed but with a (lower-level) error such as a program failure, then the flow proceeds to "undo" the effect of the write with respect to higher-level redundancy information computation (Backout Write from Redundancy **1503B**). More specifically, data for the write with the lower-level error is de-computed from any corresponding higher-level redundancy information computations (assuming that the data for the write had already been included in the corresponding higher-level redundancy computations under a presumption that the write would succeed). For example, a lower-level write failure is detected on a particular page j. In response, revised R0 and R1 pages are computed such that page j data (Pj) is set to zero. Fig. 15C illustrates selected details of an embodiment of a computation for a single byte of a revised R0 (nR0) and a single byte of a revised R1 (nR1), where j is the page of the lower-level write failure. Note that in contexts of Fig. 12 such as associated with Fig. 11, if the finite field is a Galois Field the subtraction operation ("-") illustrated in Fig. 15C is equivalent to a logical XOR operation. Other processing (not illustrated) is performed, in various embodiments, to store the page with the lower-level write failure (Pj), as well as the revised higher-level redundancy pages (nR0 and nR1).

[0291] Other than reception of write completion and status information in response to the commands sent as illustrated in Fig. 15A, the processing illustrated in Fig. 15B is independent of processing illustrated in Fig. 15A. In various scenarios, arrival order of the write completion and status information varies according to the NVM type, state, operating environment, and other factors, and in some circumstances is different than sending order or arrival order of the write commands corresponding to the arriving write completion and status information. As processing of Fig. 15A is independent of Fig. 15B (other than write completion and status information arrival being dependent on a corresponding write command being sent), in some embodiments and/or usage scenarios, write commands are being sent (Fig. 15A) while write completion and status information is being received/processed (Fig. 15B).

[0292] In some embodiments and/or usage scenarios, a significant latency transpires between write commands being sent to the NVMs and reception of write completion and status information from the NVMs in response to the write commands. In some embodiments and/or usage scenarios, write completion and status information other than in response to the

commands sent as illustrated in Fig. 15A are provided by the NVMs intermixed with the write completion and status information provided in response to the commands sent as illustrated in Fig. 15A, e.g. write completion and status information provided in response to write commands sent for other activities.

[0293] In various embodiments and/or usage scenarios, one or more pages that would otherwise be used for data information are unused. In various scenarios an unused page is a first, middle, or last page of a block, R-block, stripe, or sequence of pages any kind. In some circumstances, unused pages are unused a priori ("left out"), and in some circumstances unused pages are unused after some use ("removed from service"). An example of an unused page that is left out is a page that is manufactured incorrectly. An example of an unused page that is removed from service is a page that fails to write properly (e.g. as described as a lower-level write error with respect to Fig. 15B). Processing, e.g. relating to Figs. 10-12, 13A-13D, 14A-14B, and 15A-15B, skips over any unused pages (whether left out or removed from service), such as by omitting the unused pages entirely from computations, or by performing computations as if all data on the unused pages were zero.

MIXED GRANULARITY HIGHER-LEVEL REDUNDANCY

[0294] In various embodiments and/or usage scenarios, respective regions of NVM are operated in respective higher-level redundancy modes (e.g. RASIE-1 and RASIE-2), such as illustrated by and described in conjunction with Figs. 7, 8, and 10. In various embodiments, the respective higher-level redundancy modes are of a same granularity, for instance, the respective regions are each integer multiples of a same (e.g. size or type) element. Exemplary elements include one or more R-blocks and/or blocks, one or more R-pages and/or pages, and one or more cells associated with one or more word lines. For example, a first region of an NVM is operated in RASIE-1 mode at a granularity of a single block (e.g. the first region is an integer number of blocks, such as the first region is one or more R-blocks) and a second region of the NVM is operated in RASIE-2 mode at a granularity of a single block (e.g. the second region is also an integer number of blocks, such as the second region is one or more R-blocks). Thus the first and the second regions are operated in respective higher-level redundancy modes that are both of a same granularity (one block).

[0295] In various other embodiments, the respective higher-level redundancy modes are at various granularities, e.g. at least some of the respective regions are integer multiples of

respective different elements. For example, a first region of an NVM is operated in RASIE-1 mode at a granularity of a single page (e.g. the first region is an integer number of pages, such as corresponding to a page from each die of the NVM, an exemplary R-page, described elsewhere herein) and a second region of the NVM is operated in RASIE-2 mode at a granularity of a single block (e.g. the second region is an integer number of blocks, such as the first region is one or more R-blocks). Thus the first and the second regions are operated in respective higher-level redundancy modes that are of distinct respective granularities. For another example, respective regions of an NVM are operated in a same higher-level redundancy mode and at respective distinct granularities (e.g. a first region and a second region are operated in RASIE-1, the first region is at a single page granularity, and the second region is at a single R-block granularity).

[0296] In various embodiments and/or usage scenarios, granularity of a higher-level redundancy mode is based on integer multiples of elements having a same size, a same type, or both. For example, elements of a same size (and a same type) include pages in contexts where due, e.g., to identical lower-level redundancy coding rates, each of the pages is a same size (and each of the pages is a same type, a 'page'). For another example, elements of a same type (but different sizes) include pages in contexts where due, e.g., to differing lower-level redundancy coding rates, some of the pages are of different sizes than others of the pages, but all of the pages are of a same type, a 'page'.

[0297] Fig. 16 illustrates selected details of an embodiment of mixed granularity higher-level redundancy, in a context of an arrangement of flash die as illustrated in Fig. 6. Fig. 16 explicitly illustrates Flash Die **610.0**, **610.63**, **610.64** and, **610.65**, and by ellipsis (…) implicitly illustrates Flash Die **610.1** … **610.62**. Selected details of blocks and pages within the blocks of the Flash Die are illustrated. A plurality of R-blocks is illustrated as R-blocks **660.0**, **660.1**, **660.2** … and **660.R**. A plurality of R-pages is illustrated as R-pages **1600**, **1601**, **1602** … and **1609** (that collectively are R-block **660.0**).

[0298] An R-page conceptually corresponds to a logical slice of pages from a plurality of NVM devices (as an R-block conceptually corresponds to a logical slice of blocks from a plurality of NVM devices,). An exemplary R-page is the i^{th} page from the j^{th} block of each of flash die, taken together. There are other exemplary R-pages, such as conceptually corresponding to the various examples of R-blocks, such as the i^{th} and $(i+1)^{th}$ pages from the j^{th} block from each of flash die, taken together.

[0299] As illustrated, R-page **1600** is the first page from the first block of each of the Flash Die, taken together (Pages **610.0P0** … **610.63P0**, **610.64P0**, and **610.65P0**). R-page **1601** is the second page from the first block of each of the Flash Die, taken together (Pages **610.0P1** … **610.63P1**, **610.64P1**, and **610.65P1**). R-page **1602** is the third page from the first block of each of the Flash Die, taken together (Pages **610.0P2** … **610.63P2**, **610.64P2**, and **610.65P2**). R-page **1609** is the last page from the first block of each of the Flash Die, taken together (Pages **610.0PP** … **610.63PP**, **610.64PP**, and **610.65PP**).

[0300] Various regions of the Flash Die are illustrated as being concurrently operated in corresponding higher-level redundancy modes at various granularities, as follows. R-page **1600** is operated in a RASIE-2 mode at a granularity of a page, and includes Data Information **1600.D** stored in Pages **610.0P0** … **610.63P0** respectively of Flash Die **610.0** … **610.63**. and corresponding Redundancy Information **1600.R** stored in Pages **610.64P0** and **610.65P0** respectively of Flash Die **610.64** and **610.65**. R-page **1601** is also operated in a RASIE-2 mode at a granularity of a page, and includes Data Information **1601.D** stored in Pages **610.0P1** … **610.63P1** respectively of Flash Die **610.0** … **610.63**, and corresponding Redundancy Information **1601.R** stored in Pages **610.64P1** and **610.65P1** respectively of Flash Die **610.64** and **610.65**. R-page **1602** is operated in a RASIE-1 mode at a granularity of a page, and includes Data Information **1602.D** stored in Pages **610.0P2** … **610.64P2** respectively of Flash Die **610.0** … **610.64**, and corresponding Redundancy Information **1602.R** stored in Page **610.65P2** of Flash Die **610.65**. R-page **1609** is operated in a RASIE-3 mode at a granularity of a page, and includes Data Information **1609.D** stored in Pages **610.0PP** … **610.62PP** respectively of Flash Die **610.0** … **610.62**, and corresponding Redundancy Information **1609.R** stored in Pages **610.63PP** … **610.65PP** respectively of Flash Die **610.63** … **610.65**. Note that Flash Die **610.62** (and the Pages therein such as Page **610.62PP**) are implicitly illustrated rather than explicitly illustrated.

[0301] In contrast to R-pages **1600**, **1601**, **1602**, and **1609** that are operated in respective higher-level redundancy modes at a same granularity of a page, R-blocks **660.1**, **660.2**, and **660.R** are operated in respective higher-level redundancy modes at a same granularity of a block. Specifically, R-block **660.1** is operated in a RASIE-2 mode at a granularity of a block, and includes Data Information **1610.D** stored in Blocks **610.0B1** … **610.63B1** respectively of Flash Die **610.0** … **610.63**, and corresponding Redundancy Information **1610.R** stored in Blocks **610.64B1** and **610.65B1** respectively of Flash Die **610.64** and **610.65**. R-block **660.2** is operated in a RASIE-1 mode at a granularity of a block, and includes Data Information **1620.D** stored in Blocks **610.0B2** … **610.64B2** respectively of Flash

Die **610.0** … **610.64**, and corresponding Redundancy Information **1620.R** stored in Block **610.65B2** of Flash Die **610.65**. R-block **660.R** is operated in a RASIE-3 mode at a granularity of a block, and includes Data Information **16109.D** stored in Blocks **610.0BB** … **610.62BB** respectively of Flash Die **610.0** … **610.62**, and corresponding Redundancy Information **1690.R** stored in Blocks **610.63BB** … **610.65BB** respectively of Flash Die **610.63** … **610.65**.

[0302] Thus R-pages **1600**, **1601**, **1602**, and **1609** are all concurrently operated in various higher-level redundancy modes at a same granularity of one page (such that each region is a corresponding page from each of Flash Die **610.0** … **610.65**). Further, R-blocks **660.0**, **660.1**, **660.2**, and **660.R** are all concurrently operated in various higher-level redundancy modes at a same granularity of one block (such that each region is a corresponding block from each of Flash Die **610.0** … **610.65**). Lastly, the concurrent operation of R-pages **1600**, **1601**, **1602**, and **1609** is concurrent with the concurrent operation of R-blocks **660.0**, **660.1**, **660.2**, and **660.R**. Therefore the Flash Die are collectively and concurrently operated in various higher-level redundancy modes at various (mixed) granularities.

[0303] As a specific example of a benefit of concurrently operating in higher-level redundancy modes at a mixture of granularities, consider the following. As illustrated in Fig. 16, R-pages **1600**, **1601**, and **1602** are each operated entirely in a respective higher-level redundancy mode (specifically a RASIE-2 mode, a RASIE-2 mode, and a RASIE-1 mode). R-page **1609** is operated entirely in a RASIE-3 mode. Although not explicitly illustrated, assume that the remaining R-pages of R-block **660.0** are operated entirely in a RASIE-1 mode. Thus all of R-block **660.0** is operated in higher-level redundancy modes with page granularity. As illustrated, R-blocks **660.1**, **660.2**, and **660.R** are each operated entirely in respective higher-level redundancy modes. Although not explicitly illustrated, assume the remaining R-blocks are also each operated entirely in respective higher-level redundancy modes. Thus all R-blocks except for R-block **660.0** are operated in higher-level redundancy modes with block (or alternatively R-block) granularity, and R-block **660.0** is operated in higher-level redundancy modes with page granularity.

[0304] Incurring an incremental redundancy overhead cost of a single page (e.g. Page **610.63PP**), enables protection, via RASIE-3, of R-page **1609** from three failures during a single operation, compared to protection, via RASIE-2, from two failures during a single operation. In implementations without mixed-granularity of higher-level redundancy modes, protection of R-page **1609** incurs an incremental redundancy overhead cost of an entire block. Thus the mixed-granularity higher-level redundancy modes enabled reduced redundancy information overhead

for a same protection of a same entity, such as a (packaged) device, a die, an R-block, a block, an R-page, a page, cells associated with a word line, and one or more pluralities of the foregoing.

[0305] Fig. 17 illustrates selected details of an embodiment of concurrent operation in a plurality of mixed granularity higher-level redundancy modes (Concurrently Operate in First and Second Granularity Higher-Level Redundancy Modes **1700**). Flow begins (Start **1701**), e.g., in response to an event. Flow then continues along two concurrent paths, one for each respective granularity of higher-level redundancy mode operation, corresponding, e.g., to respective regions of NVM. Operation in a first granularity (Operate in First Granularity Higher-Level Redundancy Mode **1702**) is illustrated in the left portion of the figure, and operation in a second granularity (Operate in Second Granularity Higher-Level Redundancy Mode **1704**) is illustrated in the right portion of the figure. As illustrated, operation continues concurrently (e.g. wholly or partially in parallel) for the two granularities. Although not illustrated, in some embodiments and/or usage scenarios, in some situations flow exits from the concurrent operation of the two granularities, such as, reverting to operation in a single granularity (e.g. in response to a reboot), or proceeding to concurrent operation in more than two granularities (e.g. in response to another event). Further, while the figure illustrates concurrent operation in two granularities of higher-level redundancy modes, various other embodiments implement concurrent operation in various granularities of higher-level redundancy modes.

[0306] After initiation of the concurrent operation, in response to a request (Read/Write Request **1707**), such as from a host (e.g. Host **102** of Fig. 1B), or from an agent internal to an SSD controller (e.g. Recycler **151** of Fig. 1A), flow continues to determine which of the operating granularities the request corresponds to (Granularity? **1706**). If the request is of the first granularity (First **1706.1**), then flow proceeds to process the request according to the first granularity (Read/Write According to First Granularity Higher-Level Redundancy Mode **1703**). If the request is of the second granularity (Second **1706.2**), then flow proceeds to process the request according to the second granularity (Read/Write According to Second Granularity Higher-Level Redundancy Mode **1705**).

[0307] An example of the event that initiates the illustrated flow (e.g. via Start **1701**) and/or the event that initiates concurrent operation in more than two granularities (not explicitly illustrated) includes detection of a failure (e.g. a lower-level uncorrectable read error or a write/program failure). In various embodiments, a transition implied by initiation of the illustrated flow (e.g. via Start **1701**) and/or the initiation of operation in more than two

granularities (not explicitly illustrated), corresponds to a dynamic transition of operating mode (e.g. Dynamically Transition Operating Mode **809** of Fig. 8).

[0308] Note that here, as well as elsewhere herein, description relating to concurrent operation of higher-level redundancy modes at a plurality of granularities refers, e.g., to a plurality of entities being simultaneously used to store information and/or being simultaneously available to write/read information to/from according to the plurality of granularities. Examples of the entities include a (packaged) device, a die, an R-block, a block, an R-page, a page, cells associated with a word line, and one or more pluralities of the foregoing. At various points in time, while concurrently operating at the plurality of granularities, any one or more of: only a single one of the entities is performing (or alternatively is being directed, e.g., by an SSD controller, to perform) a single operation (e.g. a write or a read), various ones of the entities are concurrently performing various respective operations, and various sub-entities of one or more of the entities are concurrently performing various respective operations.

[0309] For example, consider an SSD controller managing several flash die. The SSD controller concurrently operates various regions of the flash die in various higher-level redundancy modes at various granularities as follows. The SSD controller operates a first of the regions in a RASIE-1 mode at a granularity of an R-block, and a second of the regions in a RASIE-2 mode at a granularity of an R-block. The SSD controller operates a third of the regions in the RASIE-2 mode at a granularity of an R-page, and a fourth of the regions in a RASIE-3 mode at a granularity of an R-page. During the concurrent operation, each of the regions simultaneously stores information and is simultaneously available to write/read information according to an R-block or an R-page granularity.

[0310] Continuing with the example, over a period of time during the concurrent operation, the SSD controller sequentially provides a series of write commands to each of the flash die. The series of write commands corresponds to writing the entirety of the first region and is in accordance with the RASIE-1 mode at a granularity of an R-block. In some scenarios, each of the write commands is emitted by the SSD controller one at a time to the flash die. Thus the SSD controller is performing a single operation at a time while concurrently operating the regions in various higher-level redundancy modes at various granularities. In some scenarios, the write commands are received by the flash die in a relatively short time span, enabling at least two or more of the flash die to proceed with respective page programming operations at least partially concurrently. Thus the at least two or more flash die are concurrently performing a

plurality of operations while being concurrently operated in various higher-level redundancy modes at various granularities.

[0311] In some embodiments and/or usage scenarios, concurrently operating in higher-level redundancy modes at a plurality of granularities is used in combination with adaptive lower-level code rates. For example, a plurality of R-blocks of an NVM is initially operated in a RASIE-1 mode at an R-block granularity and using a first adaptive lower-level code rate. Then one of the pages of a particular one of the R-blocks becomes less reliable. In response, one or more of the read units of the less reliable page are operated using a second adaptive lower-level code rate, having better error correction than the first adaptive lower-level code rate. Subsequently, the less reliable page becomes further less reliable. In response, the R-page including the less reliable page is operated in a second higher-level redundancy mode (having better error correction than the first higher-level redundancy mode) at a granularity of an R-page. The other R-pages of the particular R-block continue to be operated in the first higher-level redundancy mode but now at a granularity of an R-page (rather than an R-block). The R-blocks other than the particular R-block continue to be operated in the first high-level redundancy mode and at a granularity of an R-block. Therefore, even though the less reliable page becomes less and less reliable, the less reliable page is still usable (albeit at reduced usable storage capacity). Thus, concurrently operating in higher-level redundancy modes at a plurality of granularities in conjunction with adaptive lower-level code rates enables, in some scenarios, improved NVM and/or SSD lifetime.

OTHER EMBODIMENT INFORMATION

[0312] In various embodiments, processing of or relating to one or more elements of Figs. 8, 10-12, 13A-13D, 14A-14B, 15A-15B, and 17 is performed entirely or in part by one or more elements (or portions thereof) of Fig. 1A. For example, a portion of software execution capabilities of CPU **171** is used to manage higher-level redundancy processing, such as by directing dynamic transition of operating modes as illustrated by Fig. 8, by directing data recovery operations as illustrated by Fig. 12 and Figs. 13A-13D, and/or by directing concurrent operation in various granularities of higher-level redundancy modes as illustrated by Fig. 17. For another example, Data Processing **121** includes hardware elements dedicated to and/or specialized for computation of higher-level redundancy information in accordance with computations such as illustrated by any one or more of Figs. 11, 12, 13A-13D, and 15C.

EXAMPLE IMPLEMENTATION TECHNIQUES

[0313] In some embodiments, various combinations of all or any portions of operations performed by a system implementing mixed-granularity higher-level redundancy for NVMs (e.g. flash memories), a computing-host flash memory controller, and/or an SSD controller (such as SSD Controller **100** of Fig. 1A), and portions of a processor, microprocessor, system-on-a-chip, application-specific-integrated-circuit, hardware accelerator, or other circuitry providing all or portions of the aforementioned operations, are specified by a specification compatible with processing by a computer system. The specification is in accordance with various descriptions, such as hardware description languages, circuit descriptions, netlist descriptions, mask descriptions, or layout descriptions. Example descriptions include: Verilog, VHDL, SPICE, SPICE variants such as PSpice, IBIS, LEF, DEF, GDS-II, OASIS, or other descriptions. In various embodiments, the processing includes any combination of interpretation, compilation, simulation, and synthesis to produce, to verify, or to specify logic and/or circuitry suitable for inclusion on one or more integrated circuits. Each integrated circuit, according to various embodiments, is designable and/or manufacturable according to a variety of techniques. The techniques include a programmable technique (such as a field or mask programmable gate array integrated circuit), a semi-custom technique (such as a wholly or partially cell-based integrated circuit), and a full-custom technique (such as an integrated circuit that is substantially specialized), any combination thereof, or any other technique compatible with design and/or manufacturing of integrated circuits.

[0314] In some embodiments, various combinations of all or portions of operations as described by a computer readable medium having a set of instructions stored therein, are performed by execution and/or interpretation of one or more program instructions, by interpretation and/or compiling of one or more source and/or script language statements, or by execution of binary instructions produced by compiling, translating, and/or interpreting information expressed in programming and/or scripting language statements. The statements are compatible with any standard programming or scripting language (such as C, C++, Fortran, Pascal, Ada, Java, VBscript, and Shell). One or more of the program instructions, the language statements, or the binary instructions, are optionally stored on one or more computer readable storage medium elements. In various embodiments, some, all, or various portions of the program instructions are realized as one or more functions, routines, sub-routines, in-line routines, procedures, macros, or portions thereof.

CONCLUSION

[0315] Certain choices have been made in the description merely for convenience in preparing the text and drawings, and unless there is an indication to the contrary, the choices should not be construed per se as conveying additional information regarding structure or operation of the embodiments described. Examples of the choices include: the particular organization or assignment of the designations used for the figure numbering and the particular organization or assignment of the element identifiers (the callouts or numerical designators, e.g.) used to identify and reference the features and elements of the embodiments.

[0316] The words "includes" or "including" are specifically intended to be construed as abstractions describing logical sets of open-ended scope and are not meant to convey physical containment unless explicitly followed by the word "within."

[0317] Although the foregoing embodiments have been described in some detail for purposes of clarity of description and understanding, the invention is not limited to the details provided. There are many embodiments of the invention. The disclosed embodiments are exemplary and not restrictive.

[0318] It will be understood that many variations in construction, arrangement, and use are possible consistent with the description, and are within the scope of the claims of the issued patent. For example, interconnect and function-unit bit-widths, clock speeds, and the type of technology used are variable according to various embodiments in each component block. The names given to interconnect and logic are merely exemplary, and should not be construed as limiting the concepts described. The order and arrangement of flowchart and flow diagram process, action, and function elements are variable according to various embodiments. Also, unless specifically stated to the contrary, value ranges specified, maximum and minimum values used, or other particular specifications (such as flash memory technology types; and the number of entries or stages in registers and buffers), are merely those of the described embodiments, are expected to track improvements and changes in implementation technology, and should not be construed as limitations.

[0319] Functionally equivalent techniques known in the art are employable instead of those described to implement various components, sub-systems, operations, functions, routines, sub-routines, in-line routines, procedures, macros, or portions thereof. It is also understood that many functional aspects of embodiments are realizable selectively in either hardware (e.g.,

generally dedicated circuitry) or software (e.g., via some manner of programmed controller or processor), as a function of embodiment dependent design constraints and technology trends of faster processing (facilitating migration of functions previously in hardware into software) and higher integration density (facilitating migration of functions previously in software into hardware). Specific variations in various embodiments include, but are not limited to: differences in partitioning; different form factors and configurations; use of different operating systems and other system software; use of different interface standards, network protocols, or communication links; and other variations to be expected when implementing the concepts described herein in accordance with the unique engineering and business constraints of a particular application.

[0320] The embodiments have been described with detail and environmental context well beyond that required for a minimal implementation of many aspects of the embodiments described. Those of ordinary skill in the art will recognize that some embodiments omit disclosed components or features without altering the basic cooperation among the remaining elements. It is thus understood that much of the details disclosed are not required to implement various aspects of the embodiments described. To the extent that the remaining elements are distinguishable from the prior art, components and features that are omitted are not limiting on the concepts described herein.

[0321] All such variations in design are insubstantial changes over the teachings conveyed by the described embodiments. It is also understood that the embodiments described herein have broad applicability to other computing and networking applications, and are not limited to the particular application or industry of the described embodiments. The invention is thus to be construed as including all possible modifications and variations encompassed within the scope of the claims of the issued patent.

WHAT IS CLAIMED IS:

1. A method comprising:
operating in a first higher-level redundancy mode comprising storing M information portions in M respective areas of memory, each of the M respective areas being a same first size;
operating in a second higher-level redundancy mode comprising storing N information portions in N respective areas of the memory, each of the N respective areas being a same second size;
wherein the M information portions comprise respective M-J data information portions and respective J redundant information portions computed to protect the respective M-J data information portions;
wherein the N information portions comprise respective N-K data information portions and respective K redundant information portions computed to protect the respective N-K data information portions;
wherein each of the M respective areas and the N respective areas are non-overlapping areas of the memory;
wherein the first size is different than the second size; and
wherein each of the M respective areas is in a respective one of M physical devices of the memory and wherein each of the N respective areas is in a respective one of N physical devices of the memory.

2. The method of claim 1, wherein at least some of the M physical devices are at least some of the N physical devices.

3. The method of claim 1, further comprising accessing one or more locations comprised entirely in a particular area that is any one of the M respective areas or any one of the N respective areas, the accessing comprising determining the size of the particular area.

4. The method of claim 1, wherein.
the memory comprises a plurality of flash devices;
the M respective areas correspond to M respective blocks of a first one or more of the flash devices and the first size corresponds to a block; and
the N respective areas correspond to N respective pages of a second one or more of the flash devices and the second size corresponds to a page.

5. The method of claim 4, further comprising detecting a failure of one of the M respective blocks, and in response, operating at least the failing block in the second higher-level redundancy mode.

6. The method of claim 5, wherein J is not equal to K.

7. A system comprising:

means for managing memory according to respective first and second higher-level redundancy modes;

means for storing to the memory, responsive to the means for managing memory;

wherein the means for managing memory is enabled to direct the means for storing to store M information portions in M respective areas of memory, each of the M respective areas being a same first size;

wherein the means for managing memory is further enabled to direct the means for storing to store N information portions in N respective areas of the memory, each of the N respective areas being a same second size;

wherein the M information portions comprise respective M-J data information portions and respective J redundant information portions computed to protect the respective M-J data information portions;

wherein the N information portions comprise respective N-K data information portions and respective K redundant information portions computed to protect the respective N-K data information portions;

wherein each of the M respective areas and the N respective areas are non-overlapping areas of the memory;

wherein the first size is different than the second size; and

wherein each of the M respective areas is in a respective one of M physical devices of the memory and wherein each of the N respective areas is in a respective one of N physical devices of the memory.

8. The system of claim 7, wherein at least some of the M physical devices are at least some of the N physical devices.

9. The system of claim 7, further comprising means for accessing one or more locations comprised entirely in a particular area that is any one of the M respective areas or any one of the N respective areas, the means for accessing enabled to determine the size of the particular area.

10. The system of claim 7, wherein.
the memory comprises a plurality of flash devices;
the M respective areas correspond to M respective blocks of a first one or more of the flash devices and the first size corresponds to a block; and
the N respective areas correspond to N respective pages of a second one or more of the flash devices and the second size corresponds to a page.

11. The system of claim 10, wherein the means for managing memory is enabled to detect a failure of one of the M respective blocks and to operate at least the failing block in the second higher-level redundancy mode in response to the detecting.

12. The system of claim 11, wherein J is not equal to K.

13. An apparatus comprising:
higher-level redundancy control circuitry enabled to control memory according to respective first and second higher-level redundancy modes;
storage circuitry, responsive to the higher-level redundancy control circuitry;
wherein the higher-level redundancy control circuitry is enabled to direct the storage circuitry to store M information portions in M respective areas of memory, each of the M respective areas being a same first size;
wherein the higher-level redundancy control circuitry is further enabled to direct the storage circuitry to store N information portions in N respective areas of the memory, each of the N respective areas being a same second size;
wherein the M information portions comprise respective M-J data information portions and respective J redundant information portions computed to protect the respective M-J data information portions;
wherein the N information portions comprise respective N-K data information portions and respective K redundant information portions computed to protect the respective N-K data information portions;
wherein each of the M respective areas and the N respective areas are non-overlapping areas of the memory;
wherein the first size is different than the second size; and
wherein each of the M respective areas is in a respective one of M physical devices of the memory and wherein each of the N respective areas is in a respective one of N physical devices of the memory.

14. The apparatus of claim 13, wherein at least some of the M physical devices are at least some of the N physical devices.

15. The apparatus of claim 13, further comprising accessing circuitry enabled to access one or more locations comprised entirely in a particular area that is any one of the M respective areas or any one of the N respective areas, the accessing circuitry further enabled to determine the size of the particular area.

16. The apparatus of claim 13, wherein.

the memory comprises a plurality of flash devices;

the M respective areas correspond to M respective blocks of a first one or more of the flash devices and the first size corresponds to a block; and

the N respective areas correspond to N respective pages of a second one or more of the flash devices and the second size corresponds to a page.

17. The apparatus of claim 16, wherein the higher-level redundancy control circuitry is further enabled to detect a failure of one of the M respective blocks and to operate at least the failing block in the second higher-level redundancy mode in response to the detecting.

18. The apparatus of claim 17, wherein J is not equal to K.

19. A tangible computer readable medium having a set of instructions stored therein that when executed by a processing element cause the processing element to perform and/or control operations comprising:

operating in a first higher-level redundancy mode comprising storing M information portions in M respective areas of memory, each of the M respective areas being a same first size;

operating in a second higher-level redundancy mode comprising storing N information portions in N respective areas of the memory, each of the N respective areas being a same second size;

wherein the M information portions comprise respective M-J data information portions and respective J redundant information portions computed to protect the respective M-J data information portions;

wherein the N information portions comprise respective N-K data information portions and respective K redundant information portions computed to protect the respective N-K data information portions;

wherein each of the M respective areas and the N respective areas are non-overlapping areas of the memory;

wherein the first size is different than the second size; and

wherein each of the M respective areas is in a respective one of M physical devices of the memory and wherein each of the N respective areas is in a respective one of N physical devices of the memory.

20. The tangible computer readable medium of claim 19, wherein at least some of the M physical devices are at least some of the N physical devices.

21. The tangible computer readable medium of claim 19, wherein the operations further comprise accessing one or more locations comprised entirely in a particular area that is any one of the M respective areas or any one of the N respective areas, the accessing comprising determining the size of the particular area.

22. The tangible computer readable medium of claim 19, wherein.

the memory comprises a plurality of flash devices;

the M respective areas correspond to M respective blocks of a first one or more of the flash devices and the first size corresponds to a block; and

the N respective areas correspond to N respective pages of a second one or more of the flash devices and the second size corresponds to a page.

23. The tangible computer readable medium of claim 22, wherein the operations further comprise detecting a failure of one of the M respective blocks, and in response, operating at least the failing block in the second higher-level redundancy mode.

24. The tangible computer readable medium of claim 23, wherein J is not equal to K.

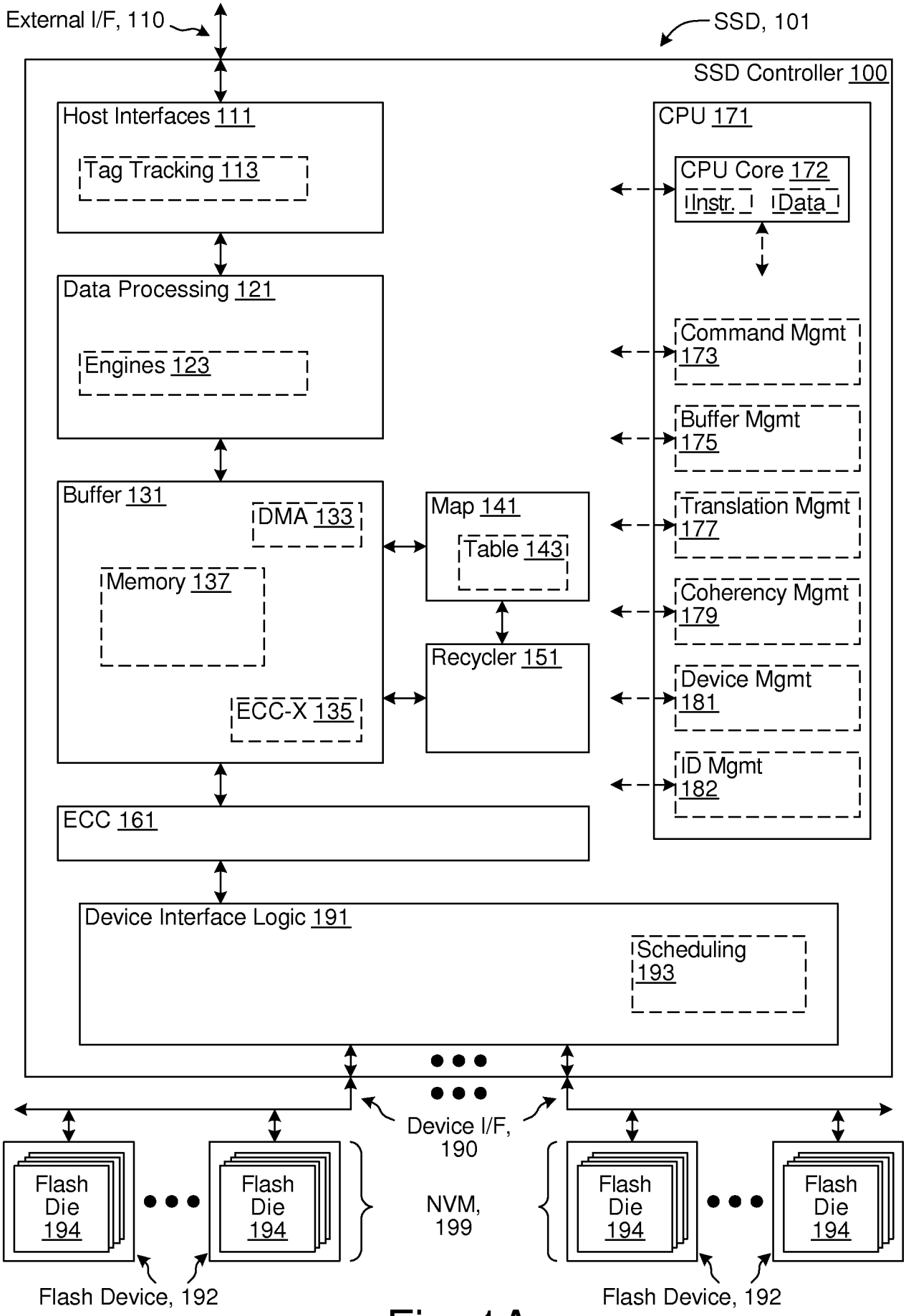

External I/F, 110
SSD, 101
SSD Controller 100
Host Interfaces 111
Tag Tracking 113
CPU 171
CPU Core 172
Instr.
Data
Data Processing 121
Engines 123
Command Mgmt 173
Buffer Mgmt 175
Buffer 131
DMA 133
Memory 137
ECC-X 135
Map 141
Table 143
Recycler 151
Translation Mgmt 177
Coherency Mgmt 179
Device Mgmt 181
ID Mgmt 182
ECC 161
Device Interface Logic 191
Scheduling 193
Device I/F, 190
Flash Die 194
NVM, 199
Flash Device, 192
Flash Device, 192

Fig. 1A

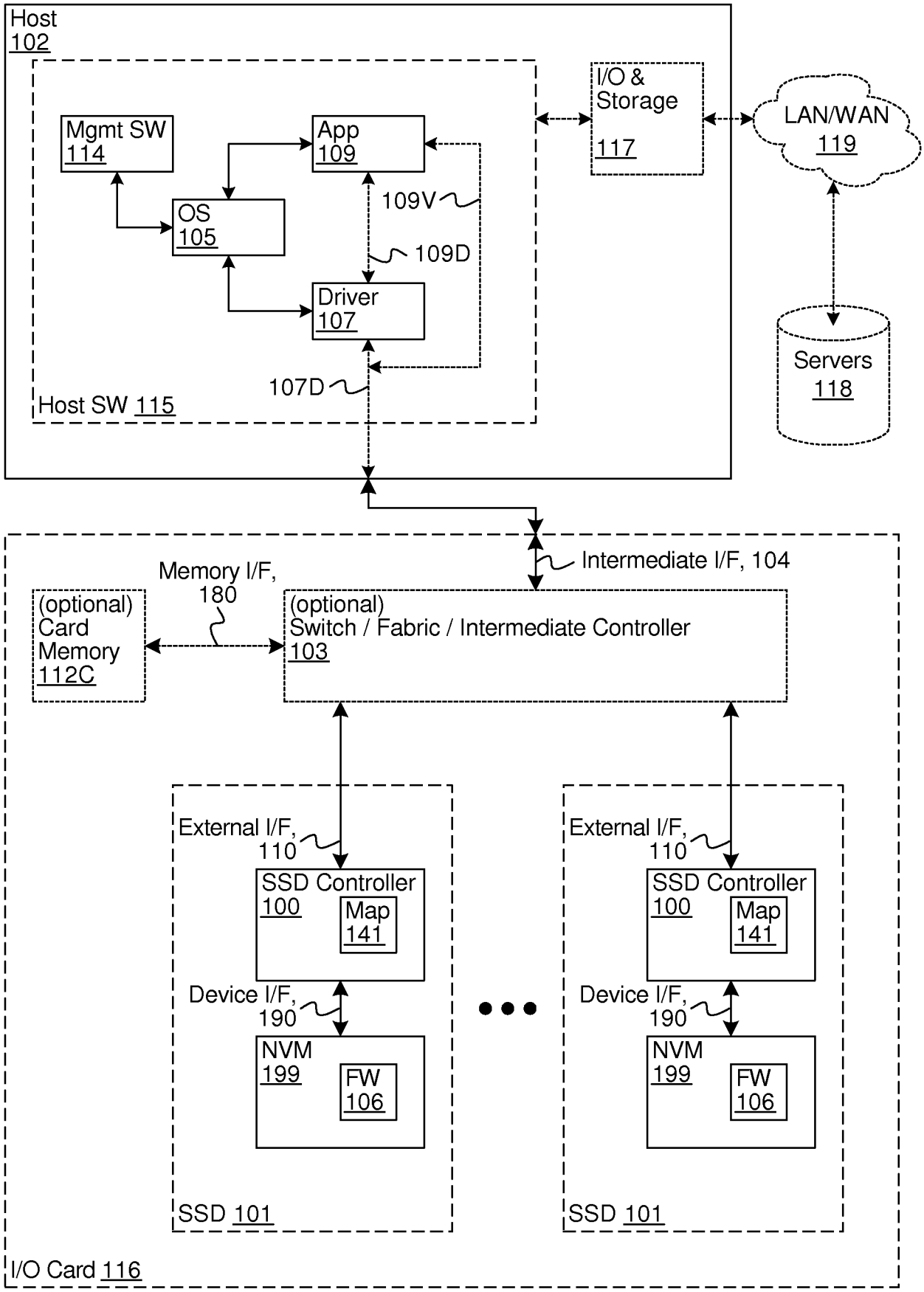

Host
102
Mgmt SW
114
App
109
OS
105
109V
109D
Driver
107
107D
Host SW 115
I/O &
Storage
117
LAN/WAN
119
Servers
118
Intermediate I/F, 104
Memory I/F,
180
(optional)
Card
Memory
112C
(optional)
Switch / Fabric / Intermediate Controller
103
External I/F,
110
SSD Controller
100
Map
141
Device I/F,
190
NVM
199
FW
106
SSD 101
External I/F,
110
SSD Controller
100
Map
141
Device I/F,
190
NVM
199
FW
106
SSD 101
I/O Card 116

Fig. 1B

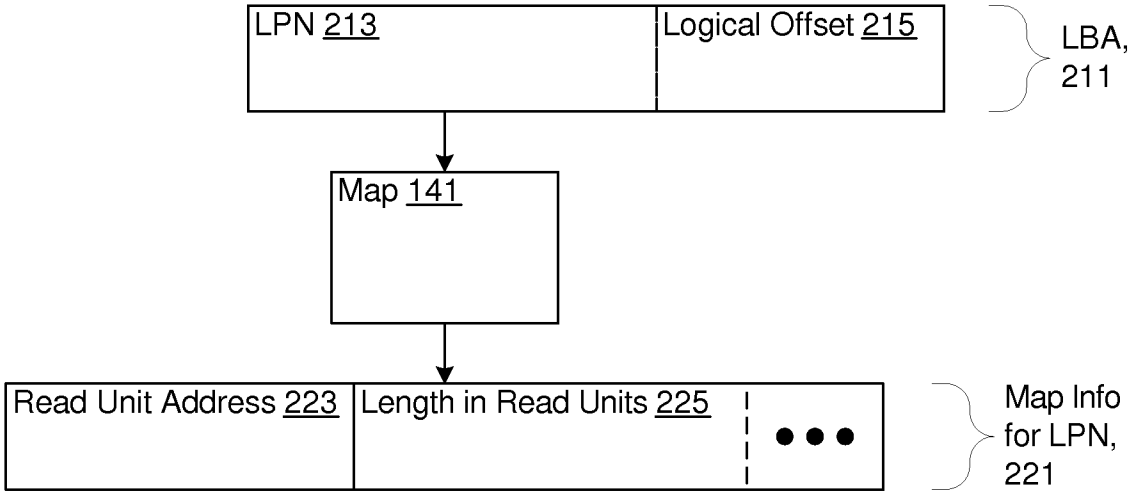

LPN 213
Logical Offset 215
LBA, 211
Map 141
Read Unit Address 223
Length in Read Units 225
Map Info for LPN, 221

Fig. 2

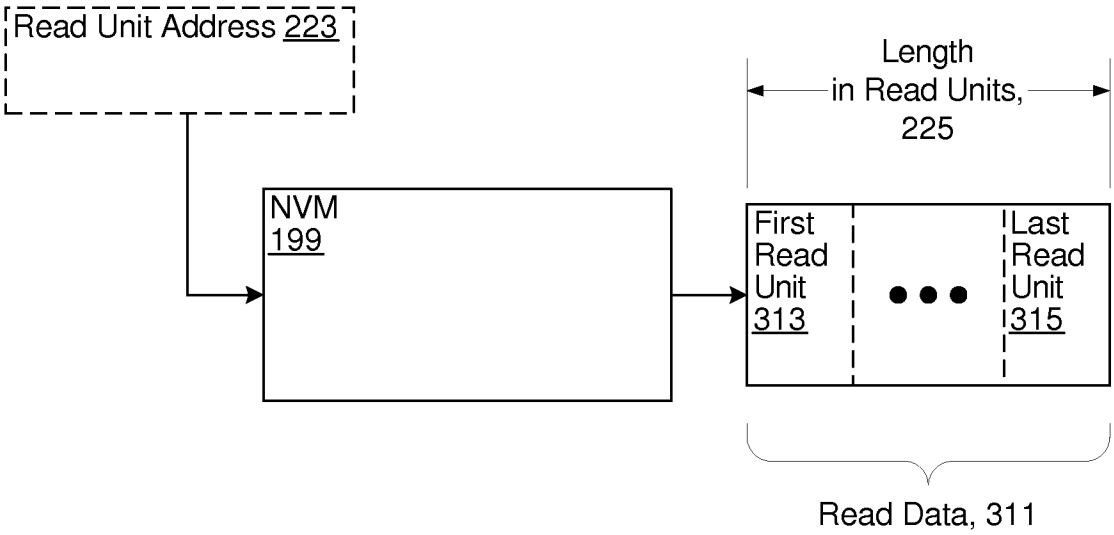

Read Unit Address 223
Length in Read Units, 225
NVM 199
First Read Unit 313
Last Read Unit 315
Read Data, 311

Fig. 3

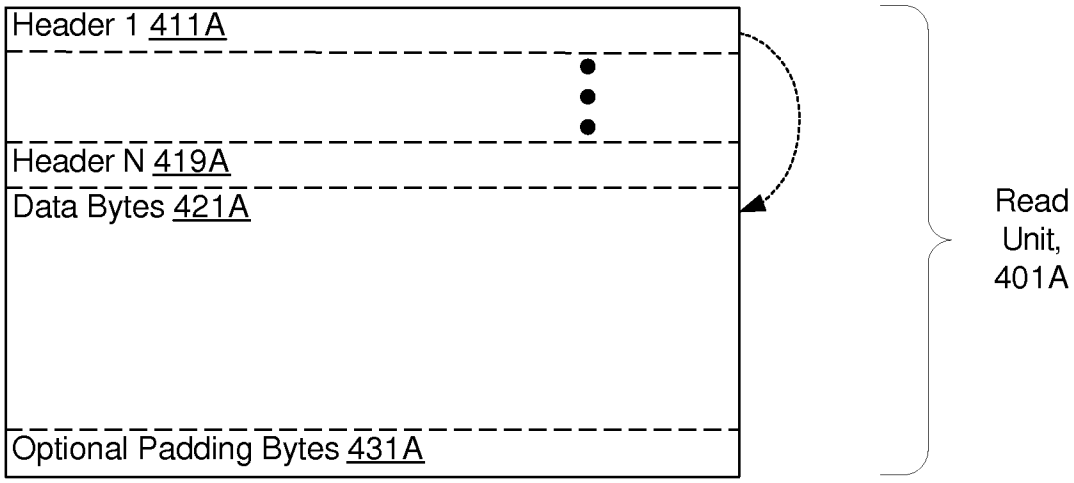

Header 1 411A
Header N 419A
Data Bytes 421A
Optional Padding Bytes 431A
Read Unit, 401A

Fig. 4A

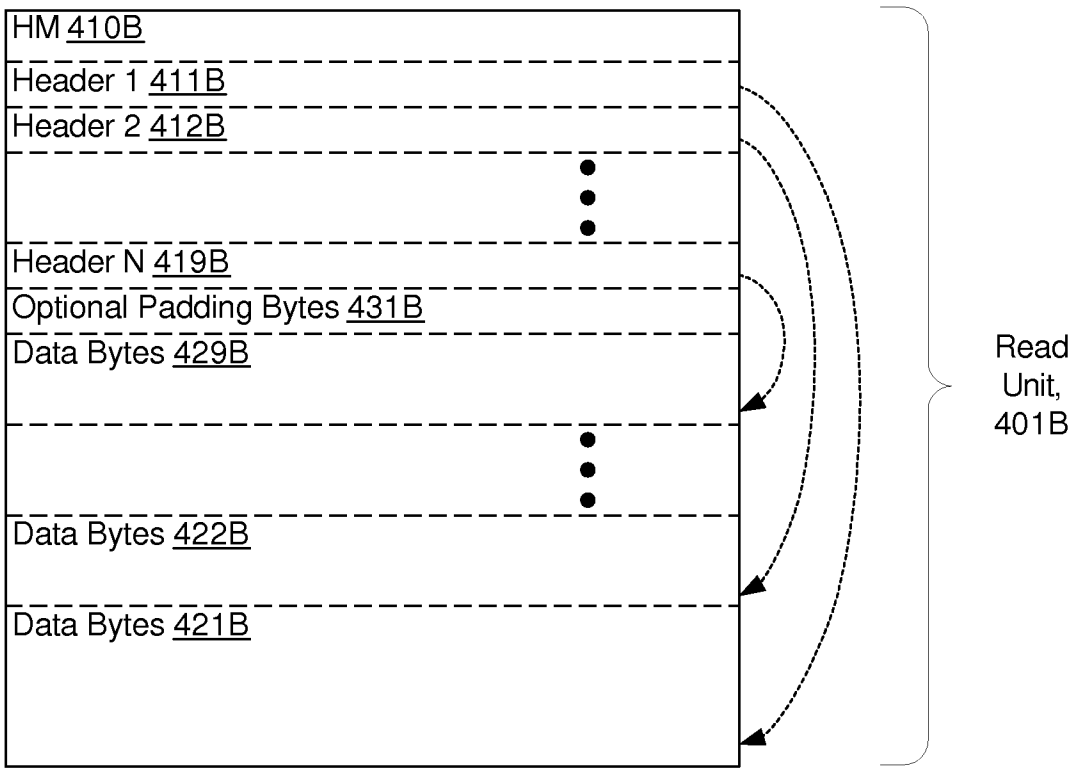

HM 410B
Header 1 411B
Header 2 412B
Header N 419B
Optional Padding Bytes 431B
Data Bytes 429B
Data Bytes 422B
Data Bytes 421B
Read Unit, 401B

Fig. 4B

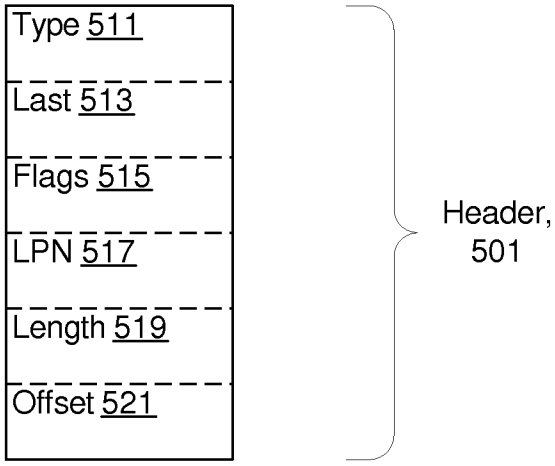

Type 511
Last 513
Flags 515
LPN 517
Length 519
Offset 521
Header, 501

Fig. 5

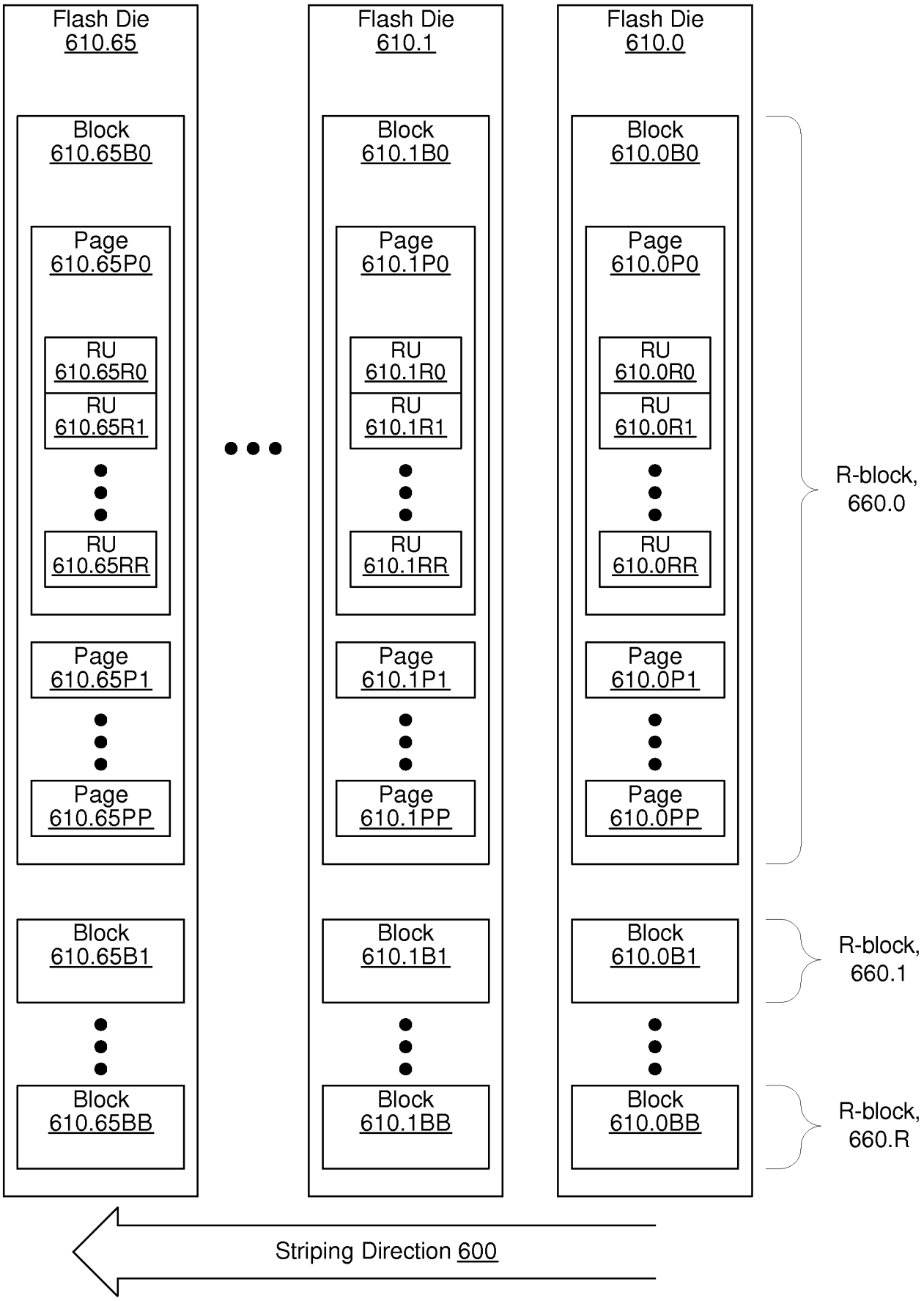

Flash Die 610.65
Flash Die 610.1
Flash Die 610.0
Block 610.65B0
Block 610.1B0
Block 610.0B0
Page 610.65P0
Page 610.1P0
Page 610.0P0
RU 610.65R0
RU 610.65R1
RU 610.65RR
RU 610.1R0
RU 610.1R1
RU 610.1RR
RU 610.0R0
RU 610.0R1
RU 610.0RR
Page 610.65P1
Page 610.1P1
Page 610.0P1
Page 610.65PP
Page 610.1PP
Page 610.0PP
Block 610.65B1
Block 610.1B1
Block 610.0B1
Block 610.65BB
Block 610.1BB
Block 610.0BB
R-block, 660.0
R-block, 660.1
R-block, 660.R
Striping Direction 600

Fig. 6

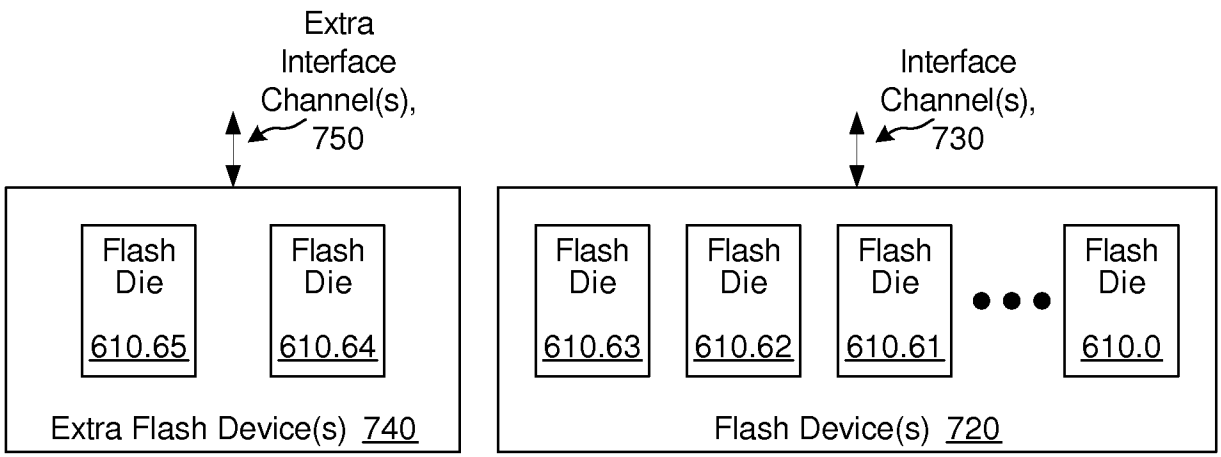

Extra
Interface
Channel(s),
750
Interface
Channel(s),
730
Flash
Die
610.65
Flash
Die
610.64
Extra Flash Device(s) 740
Flash
Die
610.63
Flash
Die
610.62
Flash
Die
610.61
Flash
Die
610.0
Flash Device(s) 720

Fig. 7

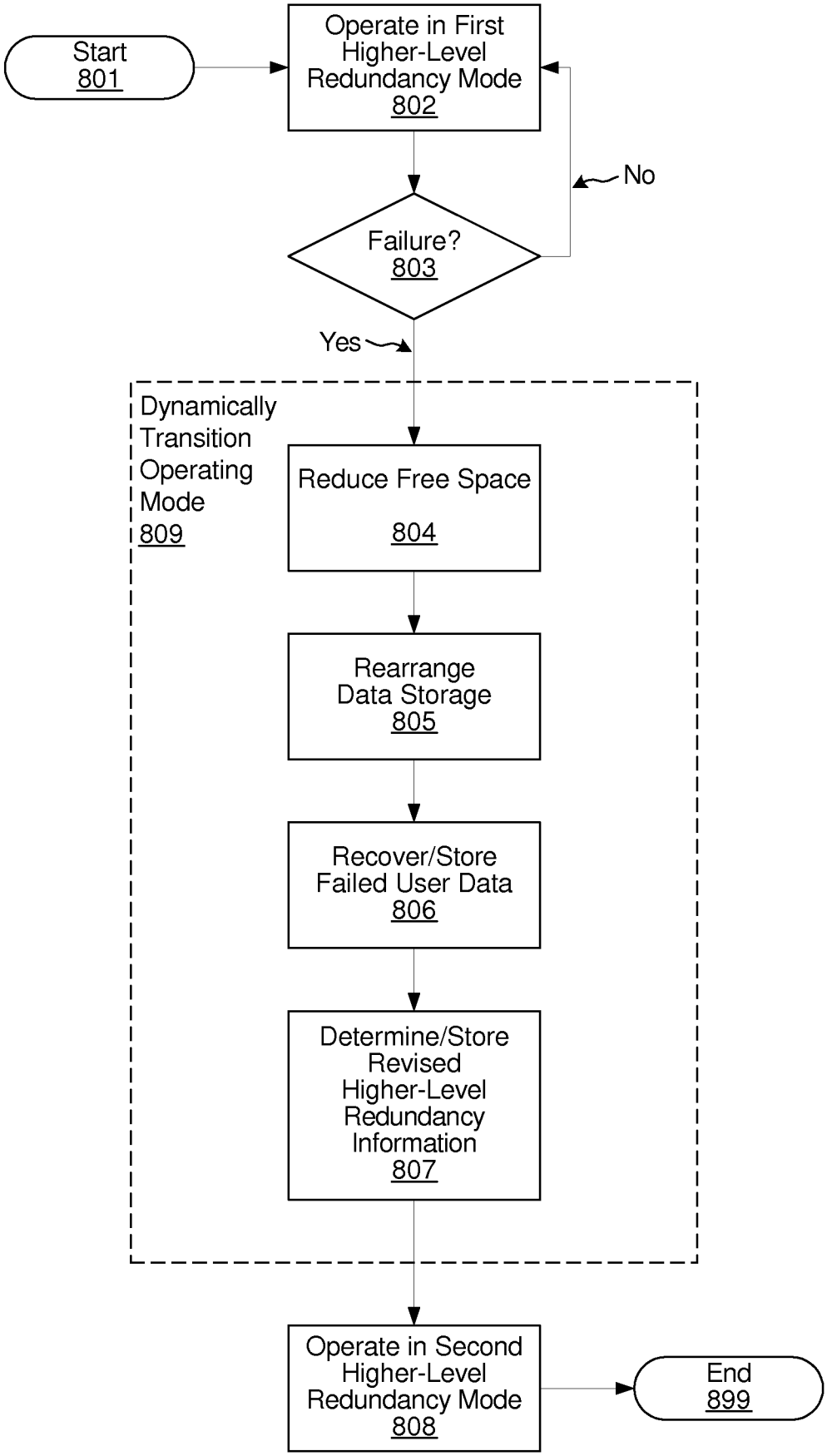

Start
801
Operate in First Higher-Level Redundancy Mode
802
Failure?
803
No
Yes
Dynamically Transition Operating Mode
809
Reduce Free Space
804
Rearrange Data Storage
805
Recover/Store Failed User Data
806
Determine/Store Revised Higher-Level Redundancy Information
807
Operate in Second Higher-Level Redundancy Mode
808
End
899

Fig. 8

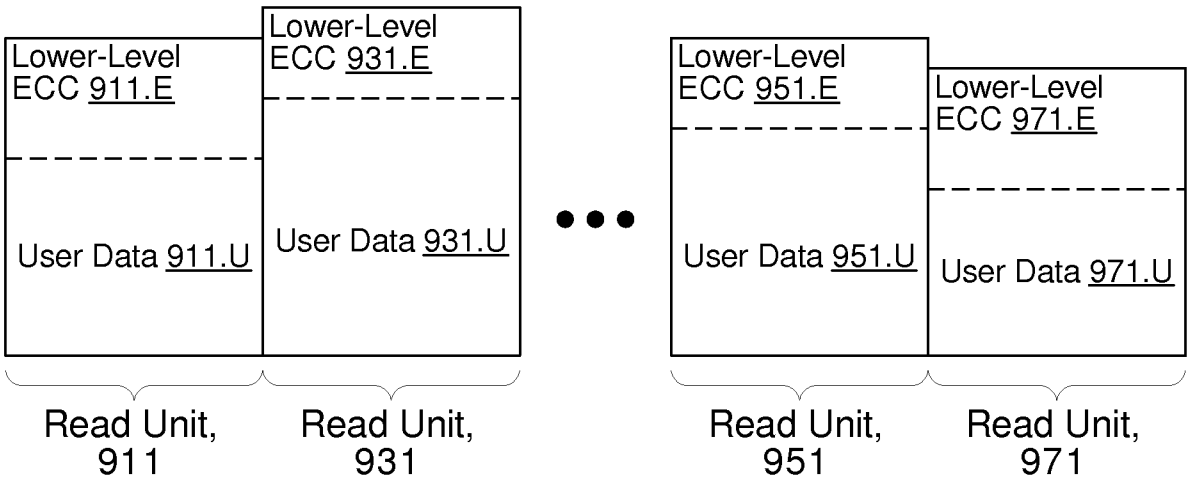

Lower-Level ECC 911.E
User Data 911.U
Lower-Level ECC 931.E
User Data 931.U
Lower-Level ECC 951.E
User Data 951.U
Lower-Level ECC 971.E
User Data 971.U
Read Unit, 911
Read Unit, 931
Read Unit, 951
Read Unit, 971

Fig. 9

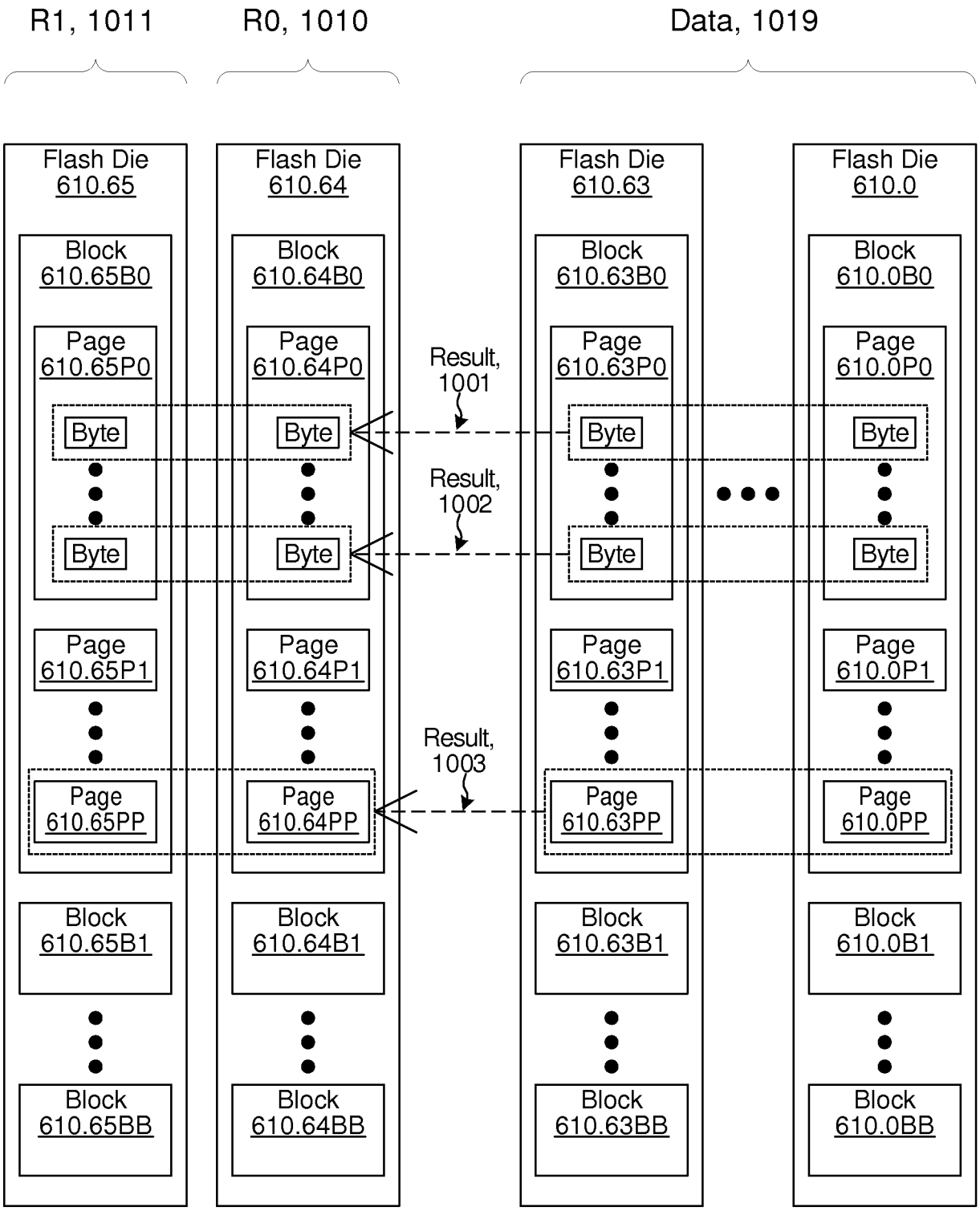

R1, 1011
R0, 1010
Data, 1019
Flash Die 610.65
Flash Die 610.64
Flash Die 610.63
Flash Die 610.0
Block 610.65B0
Block 610.64B0
Block 610.63B0
Block 610.0B0
Page 610.65P0
Page 610.64P0
Page 610.63P0
Page 610.0P0
Byte
Result, 1001
Result, 1002
Result, 1003
Page 610.65P1
Page 610.64P1
Page 610.63P1
Page 610.0P1
Page 610.65PP
Page 610.64PP
Page 610.63PP
Page 610.0PP
Block 610.65B1
Block 610.64B1
Block 610.63B1
Block 610.0B1
Block 610.65BB
Block 610.64BB
Block 610.63BB
Block 610.0BB

Fig. 10

$$R0 = \sum_i Pi \qquad R1 = \sum_i Ki * Pi$$

Fig. 11

$$Pj = R0 - \sum_{i!=j} Pi$$

Fig. 12

$$R0' = \sum_{i!=m,n} Pi \qquad R1' = \sum_{i!=m,n} Ki * Pi$$

Fig. 13A

$$R0 = R0' + Pm + Pn$$

$$R1 = R1' + Km * Pm + Kn * Pn$$

Fig. 13B

$$\Delta R0 = R0 - R0' = Pm + Pn$$

$$\Delta R1 = R1 - R1' = Km * Pm + Kn * Pn$$

Fig. 13C

$$Pn = (\Delta R1 - \Delta R0 * Km) / (Kn - Km)$$

$$Pm = \Delta R0 - Pn$$

Fig. 13D

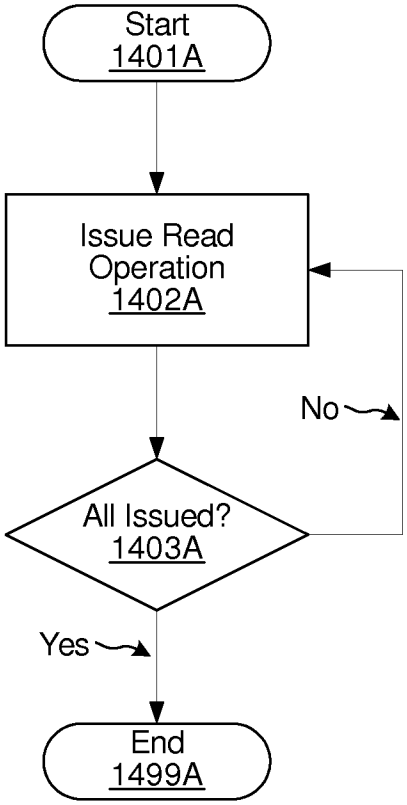


Fig. 14A

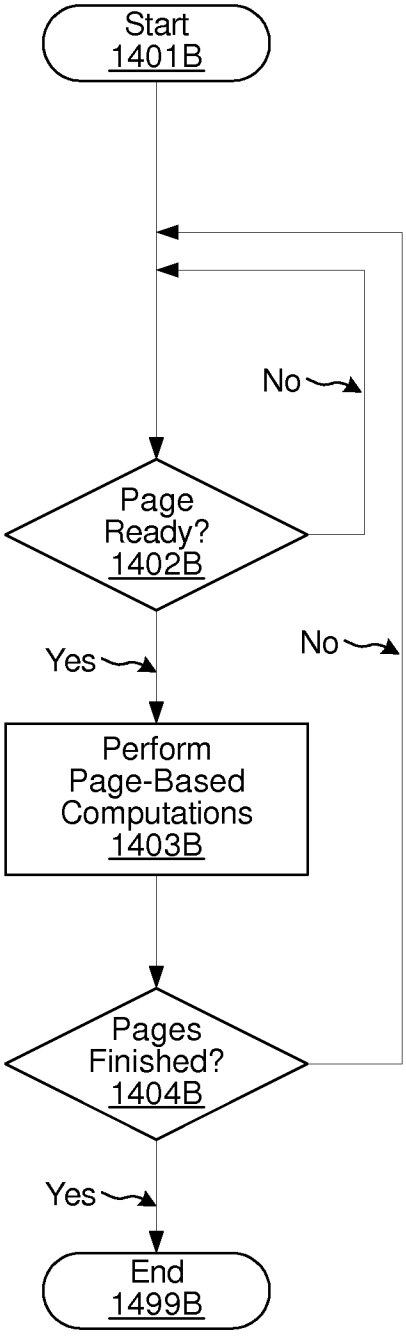


Fig. 14B

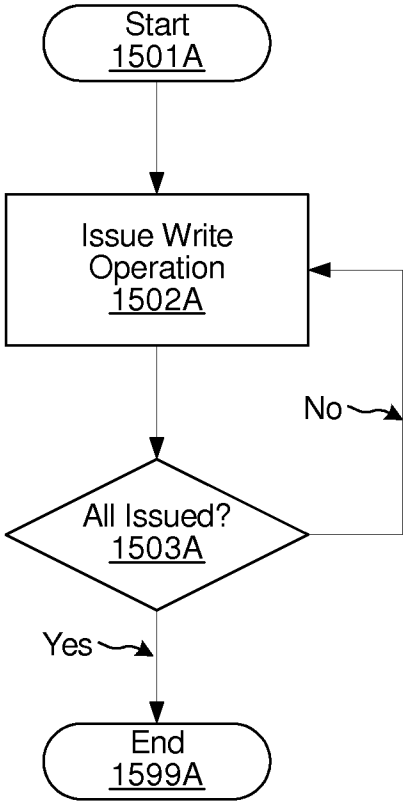


Fig. 15A

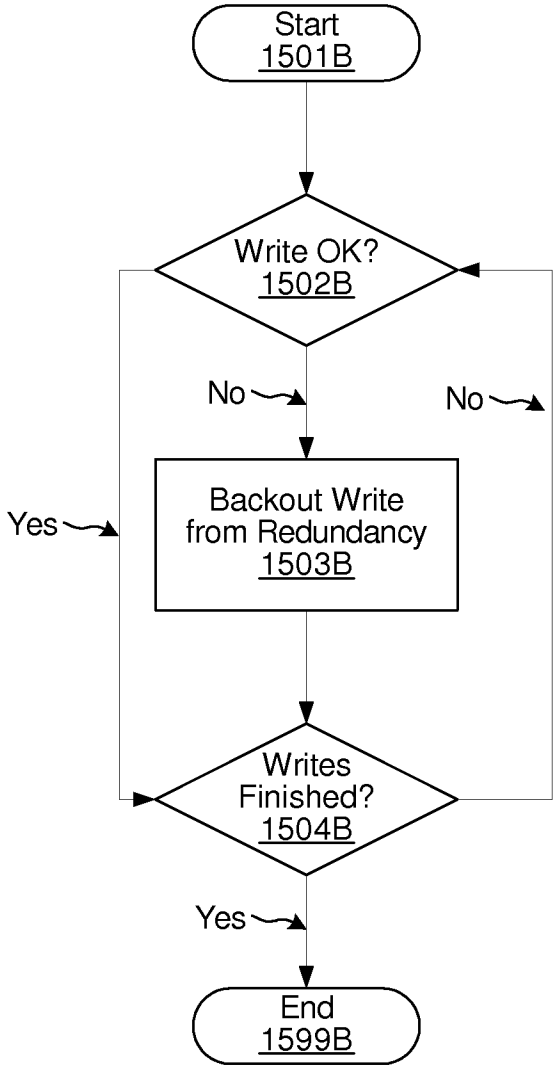


Fig. 15B

$$nR0 = R0 - Pj$$

$$nR1 = R1 - Kj * Pj$$

Fig. 15C

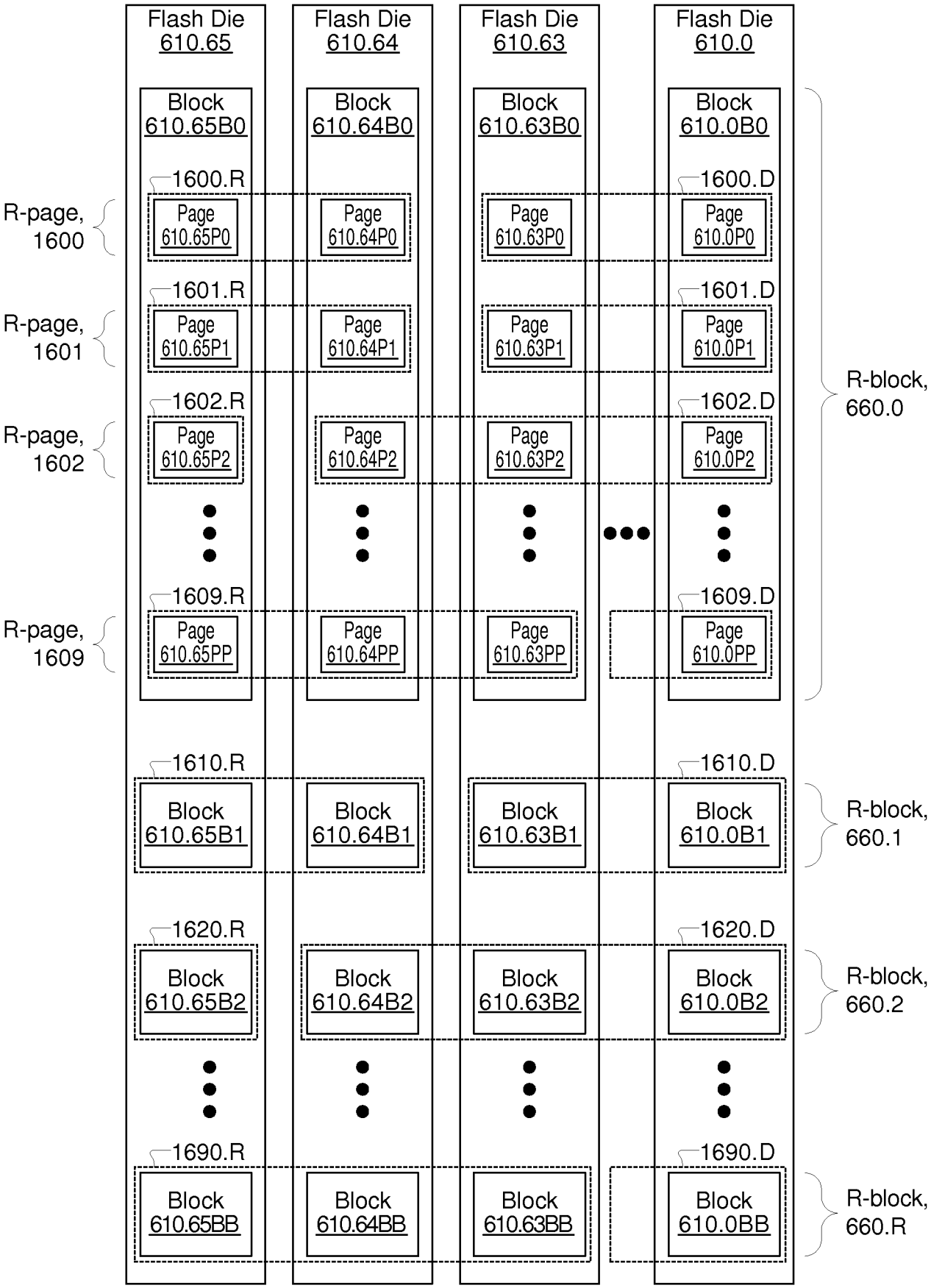

Flash Die 610.65
Flash Die 610.64
Flash Die 610.63
Flash Die 610.0
Block 610.65B0
Block 610.64B0
Block 610.63B0
Block 610.0B0
1600.R
1600.D
R-page, 1600
Page 610.65P0
Page 610.64P0
Page 610.63P0
Page 610.0P0
1601.R
1601.D
R-page, 1601
Page 610.65P1
Page 610.64P1
Page 610.63P1
Page 610.0P1
1602.R
1602.D
R-page, 1602
Page 610.65P2
Page 610.64P2
Page 610.63P2
Page 610.0P2
1609.R
1609.D
R-page, 1609
Page 610.65PP
Page 610.64PP
Page 610.63PP
Page 610.0PP
R-block, 660.0
1610.R
1610.D
Block 610.65B1
Block 610.64B1
Block 610.63B1
Block 610.0B1
R-block, 660.1
1620.R
1620.D
Block 610.65B2
Block 610.64B2
Block 610.63B2
Block 610.0B2
R-block, 660.2
1690.R
1690.D
Block 610.65BB
Block 610.64BB
Block 610.63BB
Block 610.0BB
R-block, 660.R

Fig. 16

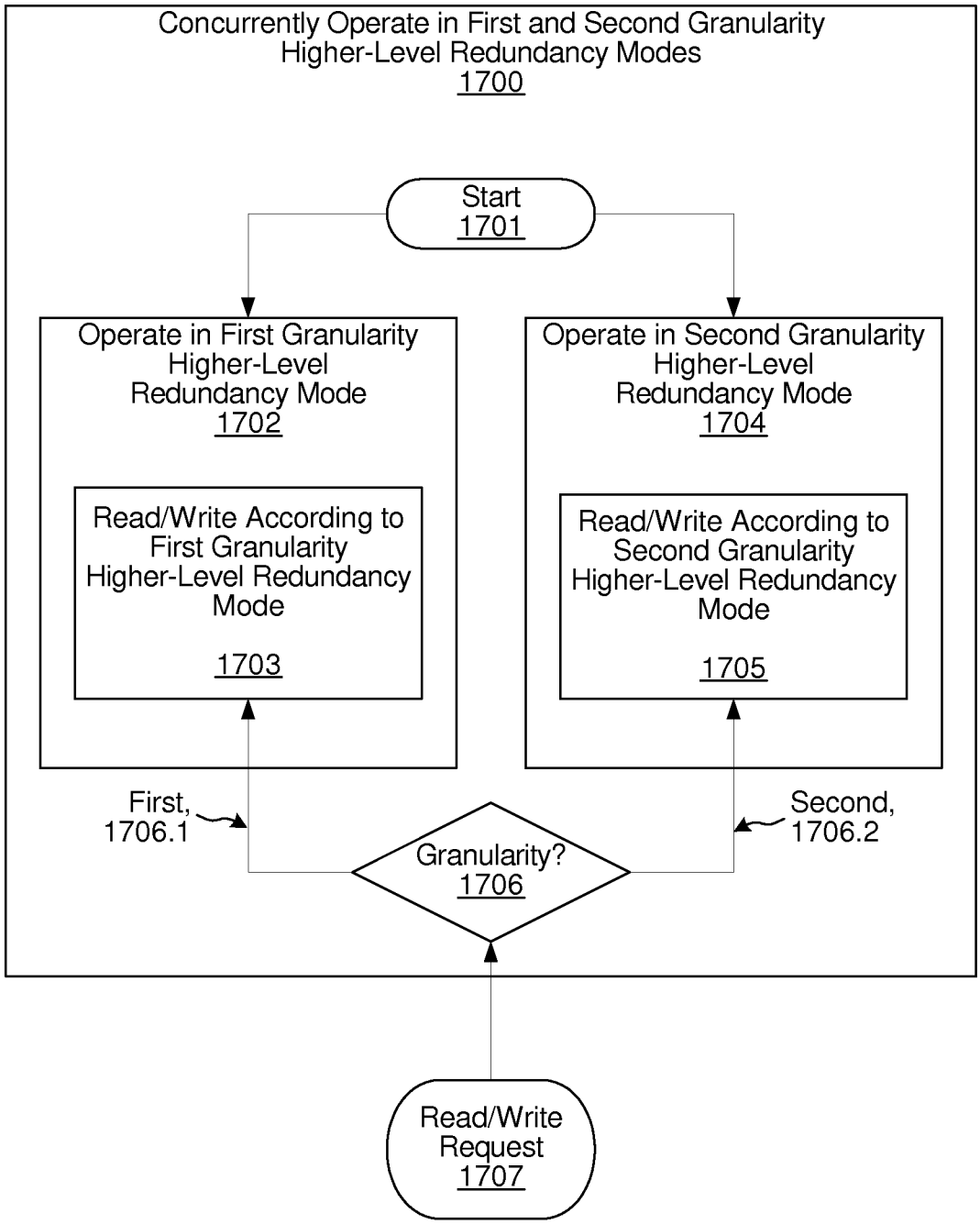

Concurrently Operate in First and Second Granularity Higher-Level Redundancy Modes
1700
Start
1701
Operate in First Granularity Higher-Level Redundancy Mode
1702
Read/Write According to First Granularity Higher-Level Redundancy Mode
1703
Operate in Second Granularity Higher-Level Redundancy Mode
1704
Read/Write According to Second Granularity Higher-Level Redundancy Mode
1705
First,
1706.1
Second,
1706.2
Granularity?
1706
Read/Write Request
1707

Fig. 17

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/051720

A. CLASSIFICATION OF SUBJECT MATTER

G06F 11/08(2006.01)i, G06F 12/00(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 11/08; G06F 12/16; G06F 11/10; G11C 29/00; G06F 11/16; G06F 12/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & keywords: memory, redundancy, protect, block, portion, and similar terms

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2012-075200 A2 (SANDFORCE INC.) 07 June 2012 See paragraphs [0030], [0031]; and claim 12.	1-24
A	US 2009-0172498 A1 (MARK SHLICK et al.) 02 July 2009 See paragraphs [0054]-[0056]; figure 1; and claim 1.	1-24
A	US 2008-0184064 A1 (JUI-JEN WU et al.) 31 July 2008 See paragraphs [0020]-[0022]; and figure 3.	1-24
A	US 5862158 A (SANDRA JOHNSON BAYLOR et al.) 19 January 1999 See column 4, line 18 - column 10, line 45; and claim 1.	1-24
A	US 2002-0176287 A1 (YOUNG DO HUR et al.) 28 November 2002 See paragraphs [0067]-[0074]; figure 9; and claim 1.	1-24

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 November 2013 (27.11.2013)	**27 November 2013 (27.11.2013)**

Name and mailing address of the ISA/KR
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
HWANG, Yun Koo
Telephone No. +82-42-481-5715

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2013/051720

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2012-075200 A2	07/06/2012	KR 10-2013-0114681 A	17/10/2013
		KR 10-2013-0118370 A	29/10/2013
		TW 201232264 A	01/08/2012
		US 2013-0246839 A1	19/09/2013
		WO 2012-075200 A3	16/08/2012
		WO 2012-099937 A2	26/07/2012
		WO 2012-099937 A3	13/09/2012
US 2009-0172498 A1	02/07/2009	TW 200935432 A	16/08/2009
		US 8443260 B2	14/05/2013
		WO 2009-083954 A1	09/07/2009
US 2008-0184064 A1	31/07/2008	CN 101236791 A	06/08/2008
		CN 101236791 B	06/04/2011
		CN 101236791 C0	06/08/2008
		US 7505319 B2	17/03/2009
US 05862158 A	19/01/1999	JP 03396136 B2	14/04/2003
		JP 09-146850 A	06/06/1997
		KR 10-0247534 B1	15/03/2000
US 2002-0176287 A1	28/11/2002	JP 2002-352595 A	06/12/2002
		KR 10-0388208 B1	19/06/2003
		KR 10-2002-0089988 A	30/11/2002
		US 6639854 B2	28/10/2003