

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

META PLATFORMS, INC. AND SAMSUNG ELECTRONICS CO., LTD.,
Petitioners,

v.

GAEA, LLC,
Patent Owner.

Case No. IPR2026-00427
U.S. Patent No. 12,265,715

**DECLARATION OF HEINER LITZ, PH.D.
IN SUPPORT OF PETITION FOR *INTER PARTES* REVIEW OF
U.S. PATENT NO. 12,265,715**

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	BACKGROUND	2
A.	Experience, Qualifications, and Status as an Independent Expert.....	2
B.	Applicable Legal Standards	4
1.	Anticipation	5
2.	Obviousness.....	5
III.	MATERIALS REVIEWED AND CONSIDERED	7
IV.	OVERVIEW OF THE TECHNOLOGY	11
A.	Continued Scaling of Storage Capacity and Performance.....	11
B.	Scaling Introduced New Device-Level Constraints.....	13
C.	Traditional Interfaces Hid Device Complexity.....	14
D.	Hyperscale Systems Drove Software-Programmable Policies	15
E.	Host-Programmable SSD Policies Were Known Before November 2016.....	16
F.	More-Complex Policies Used Request Attributes and Data- Processing Rules	18
G.	Policy-Programmable Storage at Multiple Layers.....	19
V.	THE '715 PATENT.....	21
A.	Overview.....	21
B.	Prosecution History.....	24

Declaration of Heiner Litz, Ph.D. in Support of
Petition for *Inter Partes* Review USP No. 12,265,715

1.	Prosecution of the '023 Patent.....	24
2.	Prosecution of the children of the '023 patent, including the '715 patent	25
C.	Priority Date.....	26
D.	Person of Ordinary Skill in the Art.....	26
VI.	CLAIM CONSTRUCTION	27
VII.	THE PRIOR ART	27
A.	Meir (Ex. 1010).....	27
B.	Salessi (Ex. 1011)	30
C.	Chou (Ex. 1012).....	32
D.	Lovinger (Ex. 1015).....	33
VIII.	OPINIONS ON GROUND 1: MEIR IN VIEW OF SALESSI WOULD HAVE RENDERED OBVIOUS CLAIMS 1-2, 5-8, 10-11, 14-18, AND 20.....	33
A.	Motivation to Combine	34
B.	Reasonable Expectation of Success	40
C.	Independent Claim 1	44
1.	Preamble 1[pre]: “A solid-state drive comprising:”	44
2.	Limitation 1[a]: “a solid-state memory;”	45
3.	Limitation 1[b]: “a device memory;”	46
4.	Limitation 1[c]: “a programmable processor operatively coupled to the device memory;”	47

Declaration of Heiner Litz, Ph.D. in Support of
Petition for *Inter Partes* Review USP No. 12,265,715

5.	Limitation 1[d]: “a device controller operatively coupled with the solid-state memory and the programmable processor; and”	49
6.	Limitation 1[e]: “an application stored on the device memory and comprising instructions that, when executed by the programmable processor, direct the programmable processor to:”	50
7.	Limitation 1[e.i]: “process content for storage based on a storage policy;”	52
8.	Limitation 1[e.ii]: “transfer the content to the device controller; and”	58
9.	Limitation 1[f]: “the device controller to:”	61
10.	Limitation 1[f.i]: “receive the content from the programmable processor; and”	61
11.	Limitation 1[f.ii]: “store the content on the solid-state memory.”	62
D.	Dependent Claims 2, 5-8, 10.....	65
1.	Dependent Claim 2	65
2.	Claim 5	76
3.	Claim 6	82
4.	Claim 7	90
5.	Claim 8	97
6.	Claim 10: “The solid-state drive of claim 1 wherein the content comprises one or more of video data, image data, on-line temporal video data, cached data, financial data, online transaction data, credit card data, or medical data.”	104
E.	Independent Claim 11	105

Declaration of Heiner Litz, Ph.D. in Support of
Petition for *Inter Partes* Review USP No. 12,265,715

1.	Preamble 11[pre]: “A method of operating a solid-state drive, the method comprising:”	105
2.	Limitation 11[a]: “in response to a storage request comprising content, processing the content based on a storage policy;”	107
3.	Limitation 11[b]: “storing the processed content on the solid-state memory;”	109
4.	Limitation 11[c]: “in response to a read request, reading requested content from the solid-state memory;”	112
5.	Limitation 11[d]: “processing the requested content for transfer based on the storage policy; and”	114
6.	Limitation 11[e]: “transferring the requested content.”	116
F.	Dependent Claims 14-18.....	116
1.	Claim 14	116
2.	Claim 15	119
3.	Claim 16: “The method of claim 11 wherein the storage policy indicates how to process content for storage on the solid-state drive and how to process the requested content for transfer.”	122
4.	Claim 17: “The method of claim 16 wherein the storage policy comprises one or more of an encryption policy, a decryption policy, a compression policy, or a decompression policy.”	124
5.	Claim 18: “The method of claim 17 further comprising retrieving the storage policy from memory.”	125
G.	Independent Claim 20:.....	126
1.	Preamble 20[pre]: “A solid-state drive comprising:”	126

2.	Limitation: 20[a]: “a solid-state memory;”	127
3.	Limitation 20[b]: “a device memory;”	128
4.	Limitation 20[c]: “a programmable processor operatively coupled to the device memory;”	129
5.	Limitation 20[d]: “a device controller operatively coupled with the solid-state memory and the programmable processor; and”	131
6.	Limitation 20[e]: “instructions stored on the device memory that, when executed by the programmable processor, direct the programmable processor to:”	132
7.	Limitation 20[e.i]: “process content for storage on the solid-state memory based on a storage policy;”	133
8.	Limitation 20[e.ii]: “transfer the content to the device controller to store on the solid-state memory;”	136
9.	Limitation 20[e.iii]: “in response to a content request, receive requested content read from the solid-state memory by the device controller;”	139
10.	Limitation 20[e.iv]: “process the requested content for transfer based on the storage policy; and”	140
11.	Limitation 20[e.v]: “transfer the requested content.”	142

IX. OPINIONS ON GROUND 2: MEIR IN VIEW OF SALESSI AND CHOU WOULD HAVE RENDERED OBVIOUS CLAIM 9 142

A.	Motivation to Combine	142
B.	Reasonable Expectation of Success	144
C.	Dependent Claim 9	145
1.	Preamble 9[pre]: “The solid-state drive of claim 1 further comprising:”	145

2. Limitation 9[a]: “a communication interface to communicatively couple the solid-state drive to a host device.” 145
3. Limitation 9[b]: “a data link to communicatively couple the programmable processor and the device controller;” 147
4. Limitation 9[c]: “a printed circuit board to mount the solid-state memory, the device memory, the device controller, the programmable processor, and the data link; and” 149
5. Limitation 9[d]: “a case to mount the printed circuit board and the communication interface.” 150

X. OPINIONS ON GROUND 3: MEIR IN VIEW OF SALESSI AND LOVINGER WOULD HAVE RENDERED OBVIOUS CLAIM 19 ...153

- A. Motivation to Combine 153
- B. Reasonable Expectation of Success 154
- C. Claim 19 155
 1. Claim 19: The method of claim 17 wherein the storage request comprises the storage policy 155

XI. CONCLUSION.....157

I. INTRODUCTION

1. I have been retained by Meta Platforms, Inc. (“Meta”) to analyze the scope and content of U.S. Patent No. 12,265,715 (“the ’715 patent”) relative to the state of the art at the time of the earliest application to which the ’715 patent claims priority. My analysis relates to claims 1, 2, 5-11, and 14-20. I have also been retained to provide analysis regarding what a person of ordinary skill in the art would have understood at the time of the earliest application underlying the ’715 patent.

2. This declaration summarizes the opinions I have formed to date. I reserve the right to modify or amend my opinions, if necessary, based on further review and analysis of information that I receive subsequent to the filing of this declaration, including in response to positions taken by Gaea, LLC (“Gaea”) or its experts that I have not yet seen, including any secondary considerations evidence that Gaea or its expert may consider and present.

3. It is my opinion that claims 1-2, 5-11, and 14-20 of the ’715 patent are invalid as obvious based on the following grounds.

Ground 1	Claims 1-2, 5-8, 10-11, 14-18, and 20 are obvious in view of Meir and Salessi
Ground 2	Claim 9 is obvious in view of Meir, Salessi, and Chou
Ground 3	Claim 19 is obvious in view of Meir, Salessi, and Lovinger

II. BACKGROUND

A. Experience, Qualifications, and Status as an Independent Expert

4. My curriculum vitae (“CV”) is Exhibit 1004.
5. A detailed list of my professional activities, memberships, and speaking engagements is included in my CV (Ex. 1004).
6. I am being compensated at the rate of \$600 per hour for my work on this declaration. My fee is not contingent on the outcome of any matter or on any of the technical positions I explain in this declaration.
7. I have no financial interest in Meta. I have been informed that Gaea, LLC (“Gaea”) claims ownership of the ’715 patent. I have no financial interest in Gaea.
8. I earned a degree in Computer Engineering from the University of Mannheim, Germany, in 2006. In 2012, I completed a Ph.D. in Computer Engineering from the same institution, graduating Summa Cum Laude. Between 2006 and 2012, my graduate research focused on computer architecture, hardware chip design, scalable storage systems, and high-performance computing networks.
9. In 2013, I began my academic career as a Postdoctoral Researcher at Stanford University, working in the research groups of Professors David Cheriton and Christos Kozyrakis. From 2012 to 2016, my research centered on programmable “Open-Channel” solid-state drives (SSDs). I introduced techniques to reduce access

latency by shifting performance-critical tasks such as data placement from the SSD to the host processor. This work led to “RAIL: Predictable, Low Tail Latency for NVMe Flash”, which achieved a 7× reduction in read access latency through host-orchestrated SSD data placement.

10. In 2017, I served as Visiting Faculty at Google, working on programmable storage devices within the Platforms Infrastructure Group.

11. In 2018, I joined the Computer Science Department at the University of California, Santa Cruz (UCSC), as an Assistant Professor. I was promoted to Associate Professor with tenure in 2023. In 2024, I was named the Kumar Malavalli Endowed Professor for Storage Systems Research, a position supported by a \$1 million gift from Kumar Malavalli.

12. Since 2023, I have served as Director of the Center for Research in Storage Systems (CRSS) at UCSC, a center comprised of five faculty and 15 Ph.D. students. CRSS advances research in storage systems and is supported by the National Science Foundation (NSF) and industry sponsors. Our work spans a range of topics, including programmable storage and emerging technologies such as Open-Channel and Zoned Namespace (ZNS) SSDs. In the context of ZNS, my lab proposed the group append feature to optimize ZNS SSD integration with file systems (*e.g.*, “Append is Near: Log-based Data Management on ZNS SSDs”).

13. In 2024, I was a Visiting Professor at the Massachusetts Institute of Technology (MIT) in Cambridge.

14. Over the course of my academic career, I have published more than 80 peer-reviewed research papers, journal articles, and patents in the areas of storage systems and computer hardware. I have advised over 15 Ph.D. students and secured more than \$10 million in research funding. My honors include the NSF CAREER Award, Intel Outstanding Researcher Award, and multiple Faculty Awards from Google and Facebook. I received the Best Paper Award at MICRO 2023 and SoCC 2025 and several IEEE MICRO Top Pick awards. I have also served as Program Chair for HotChips 2023 and NVMW 2024, and as Associate Editor for IEEE Computer Architecture Letters (CAL).

B. Applicable Legal Standards

15. I understand that statutory and judicially created standards must be considered to determine the validity of a patent claim. I am not an attorney and, consequently, will offer no opinion on the law itself. My understanding of the pertinent law is described in this section and is the result of explanations provided by counsel. I have applied this understanding in my analysis.

16. I understand that a patent claim is unpatentable if it is anticipated by, or obvious in view of, the prior art. I further understand that the time frame of reference

for determining whether a claim is unpatentable is from the perspective of a person of ordinary skill in the art at the time of invention.

1. Anticipation

17. I have been informed that for a patent claim to be anticipated by the prior art, each and every limitation of the claim must be found, expressly or inherently, in a single prior art reference as recited in the claim. I have been informed that a claim limitation not expressly found in a prior art reference is inherent if the prior art necessarily functions in accordance with, or includes, the claim limitation. Mere probability that a limitation is included is not sufficient to establish inherency.

2. Obviousness

18. In analyzing obviousness in light of the prior art I have been informed that it is important to understand the scope of the claims, the level of skill in the relevant art, the scope and content of the prior art, the differences between the prior art and the claims, and any objective indicia of non-obviousness (also called secondary considerations).

19. I have been informed that a patent claim is unpatentable for obviousness if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person of ordinary skill in the art to which said subject

matter pertains. I have been informed that obviousness may be based on one reference or a combination of references. I have been informed that the combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results.

20. I have been informed that when a patented invention is a combination of known elements, the Patent Trial and Appeal Board (“Board”) must determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue by considering the teachings of prior art references, the effects of demands known to people working in a field or present in the marketplace, and the background knowledge possessed by a person of ordinary skill in the art.

21. I have been informed that the Supreme Court has recognized several rationales for combining references or modifying a reference to show obviousness of the claimed subject matter. I understand that several of these rationales are: (1) combining prior art elements according to known methods to yield predictable results; (2) simple substitution of one known element for another to obtain predictable results; (3) use of a known technique to improve a similar device (method or product) in the same way; (4) applying a known technique to a known device (method or product) ready for improvement to yield predictable results; (5) choosing from a finite number of identified, predictable solutions, with a reasonable

expectation of success; (6) and some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

22. I have been informed that in order to prove that a claimed invention is not patentable for obviousness, a petitioner must (1) identify the differences between the claim and particular disclosures in the prior art references, singly or in combination, (2) specifically explain how the prior art references could have been combined in order to arrive at the subject matter of the claimed invention, and (3) specifically explain why a person of ordinary skill in the art would have had reasons to so combine the prior art references.

III. MATERIALS REVIEWED AND CONSIDERED

23. My opinions in this declaration are based on working in the field of data storage systems, including my teaching, research, and consulting. I have an established understanding of the relevant field at the relevant time frame and I have an understanding of the skill set, capabilities, and knowledge of a person of ordinary skill in the art as of 2016.

24. My opinions are also based on investigation and study of the patent at issue, its file history, and the prior art. In the course of forming my opinions, I have reviewed the exhibits submitted with the Petition and any other materials cited in

Declaration of Heiner Litz, Ph.D. in Support of
Petition for *Inter Partes* Review USP No. 12,265,715

this declaration; I have reviewed the following exhibits, which I understand are attached to the Petition for *Inter Partes* Review of the '715 patent.

Exhibit No.	Description
1001	U.S. Patent No. 12,265,715
1002	File History of U.S. Patent '715 Patent
1005	U.S. Patent Application Publication No. 2015/0052167 to Olster et al. ("Olster")
1006	U.S. Patent Publication No. 2016/0313943 to Hashimoto et al. ("Hashimoto")
1007	U.S. Patent No. 9,075,851 to Kilian et al. ("Kilian")
1008	U.S. Patent No. 8,874,835 to Davis et al. ("Davis")
1009	U.S. Patent Application Publication No. 2012/0166749 to Eleftheriou et al. ("Eleftheriou")
1010	U.S. Patent Application Publication No. 2012/0246435 to Meir ("Meir")
1011	U.S. Patent Application Publication No. 2015/0032938 to Salessi ("Salessi")
1012	U.S. Patent Application Publication No. 2015/0146364 to Chou ("Chou")
1015	U.S. Patent Application Publication No. 2014/0089278 to Lovinger et al. ("Lovinger")
1016	File History of U.S. Patent No. 10,776,023
1017	Chip Walter, 'Kryder's Law' (2005)' Scientific American, Aug. 2005.
1018	Timothy Feldman and Garth A. Gibson, 'Shingled Magnetic Recording' ACM Queue / CMU, 2013.

Declaration of Heiner Litz, Ph.D. in Support of
Petition for *Inter Partes* Review USP No. 12,265,715

Exhibit No.	Description
1019	Albert Chen and Jim Malina, 'Addressing Shingled Magnetic Recording Drives with Linear Tape File System' SNIA Storage Developer Conference, 2013
1020	Jorge Campello, 'SMR: The Next Generation of Storage Technology' SNIA Storage Developer Conference, 2015.
1021	Samarth Bhargava, 'Lowering HAMR Near-Field Transducer Temperature via Inverse Electromagnetic Design' IEEE Transactions on Magnetics, Apr. 2015.
1022	Yu Cai et al., 'Read Disturb Errors in MLC Nand Flash Memory: Characterization, Mitigation, and Recovery' DSN 2015.
1023	Nitin Agrawal et al., 'Design Tradeoffs for SSD Performance' USENIX Annual Technical Conference, 2008
1024	Feng Chen, David A. Koufaty, and Xiaodong Zhang; 'Understanding Intrinsic Characteristics and System Implications of Flash Memory Based Solid State Drive' SIGMETRICS 2009
1025	Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, 'The Google File System' SOSP 2003
1026	Kirk McKusick and Sean Quinlan, 'GFS: Evolution on Fast-Forward,' ACM Queue, Aug. 2009 / Communications of the ACM, Mar. 2010
1027	Konstantin Shvachko et al., 'The Hadoop Distributed File System,' IEEE MSST 2010
1028	Sage A. Weil et al., 'Ceph: A Scalable, High-Performance Distributed File System' OSDI 2006
1029	Sage A. Weil et al., 'CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data' SC 2006

Declaration of Heiner Litz, Ph.D. in Support of
Petition for *Inter Partes* Review USP No. 12,265,715

Exhibit No.	Description
1030	Leslie Ashton et al., ‘Two Decades of Policy-Based Storage Management for the IBM Mainframe Computer’ IBM Systems Journal, 2003
1031	IBM, ‘DFSMSHsm Primer’; IBM Redbooks, Sept. 2015
1032	IBM, ‘GPFS: Advanced Administration Guide’ including policy-based data management for GPFS, pre-2016 IBM documentation
1033	IBM, ‘IBM Spectrum Scale (formerly GPFS)’ IBM Redbooks, May 2015
1034	NVM Express, ‘NVM Express Revision 1.0’ Mar. 1, 2011
1035	NVM Express, ‘NVM Express Revision 1.2’ Nov. 3, 2014
1036	Open-Channel Solid State Drives, ‘NVMe Specification Revision 1.2’ Apr. 2016
1037	Trusted Computing Group, ‘Storage Security Subsystem Class: Opal Specification Version 2.00’ Feb. 24, 2012
1038	T10, “SCSI Block Commands - 3 (SBC-3),” Working Draft Revision 25, Oct. 27, 2010
1039	Fred Knight and David L. Black, ‘SBC-3 Thin Provisioning Commands’ T10 proposal 08-356r5, Jan. 14, 2009
1040	T10, ‘SCSI Block Commands - 4 (SBC-4)’ draft index, including pre-priority revisions in 2014-2016
1041	David A. Patterson, Garth Gibson, and Randy H. Katz, ‘A Case for Redundant Arrays of Inexpensive Disks (RAID)’ SIGMOD 1988
1042	Peter M. Chen et al., ‘RAID: High-Performance, Reliable Secondary Storage’ ACM Computing Surveys, 1994

Exhibit No.	Description
1043	John Colgrove et al., 'Purity: Building Fast, Highly-Available Enterprise Flash from Commodity Components,' SIGMOD 2015
1044	EMC Corporation, 'XtremIO Data Protection (XDP): Flash-Specific Data Protection,' White Paper, July 2014
1045	U.S. Patent Application Publication No. 2004/0230795 to Armitano ("Armitano")
1046	U.S. Patent Application Publication Nos. 2007/0244939 and 2007/0244940 to Devarakonda ("Devarakonda")
1047	U.S. Patent Application Publication No. 2011/0022642 to deMilo ("deMilo")
1048	U.S. Patent Application Publication No. 2010/0145917 to Bone ("Bone")
1049	U.S. Patent Application Publication No. 2010/0332401 to Prahlad ("Prahlad")
1050	U.S. Patent Application Publication No. 2014/0297695 to Dejana ("Dejana")
1051	IBM, 'Understanding Hard Drive Media Defects White Paper – Servers' IBM Product Documentation, Dec. 1997

IV. OVERVIEW OF THE TECHNOLOGY

A. Continued Scaling of Storage Capacity and Performance

25. Over the last several decades, storage capacity and performance have improved rapidly under the same broad scaling forces commonly associated with Moore's law. In magnetic storage, this trend is often described as Kryder's law: hard-

disk areal density historically increased exponentially, moving commercial systems from megabytes and gigabytes to terabytes, petabytes, and eventually hyperscale exabyte deployments. That progress was remarkable, but it was not achieved by simply making the same devices larger. Each generation required new device physics, firmware, and storage-management algorithms to maintain reliability, latency, endurance, and cost. Ex. 1017 (‘Kryder’s Law’).

26. For hard disk drives, manufacturers introduced technologies such as shingled magnetic recording (SMR), which increases areal density by overlapping adjacent tracks, and heat-assisted magnetic recording (HAMR), which uses localized heating to write smaller, more stable magnetic bits. For flash storage, manufacturers increased bits per cell from SLC to MLC, TLC, and QLC, and moved from planar NAND to 3D NAND. These techniques increased capacity and lowered cost per bit, but they also made device behavior more complex and workload dependent. Ex. 1018 (‘Shingled Magnetic Recording’); Ex. 1019 (‘Addressing Shingled Magnetic Recording Drives with Linear Tape File System’); Ex. 1020 (‘SMR: The Next Generation of Storage Technology’); Ex. 1021 (‘Lowering HAMR Near-Field Transducer Temperature via Inverse Electromagnetic Design’).

27. Flash storage or memory refers to persistent memory with no moving parts invented in the 1980s based on EEPROM technology. There are two primary types of flash memory, NOR Flash and NAND Flash. NOR uses parallel

connections for fast random access, while NAND uses series connections for high density. This requires NAND memory data to be written in large blocks rather than individually which lead to cheaper, higher-capacity storage, making solid-state drives which included NAND memory more popular in the 1990s through 2000s.

B. Scaling Introduced New Device-Level Constraints

28. SMR illustrates the tradeoff. By writing overlapping tracks, SMR increases density but limits fine-grained in-place updates. In host-managed or host-aware SMR, zones are subject to sequential-write constraints; modifying data within a zone may require rewriting a larger band or zone. Drive-managed SMR can hide this behavior behind a conventional block interface, but doing so requires internal buffering, mapping, caching, and cleaning policies that affect performance, especially under random-write workloads. Ex. 1018 (‘Shingled Magnetic Recording’); Ex. 1022 (‘Read Disturb Errors in MLC Nand Flash Memory: Characterization, Mitigation, and Recovery’).

29. NAND flash scaling created different constraints. Multi-level NAND stores multiple bits per cell by assigning values to threshold-voltage ranges; as the number of programmed states increases and process geometry shrinks, voltage margins narrow. By the mid-2010s, MLC and TLC NAND were understood to present endurance, retention, read-disturb, and read-threshold tradeoffs that SSD controllers had to manage. The tighter voltage margins generally reduce

program/erase endurance, increase the need for error correction and read-retry mechanisms, and can increase latency. Controllers respond with wear leveling, garbage collection, bad-block management, read-retry calibration, SLC caching, and stronger error correction. Ex. 1020 (‘SMR: The Next Generation of Storage Technology’); Ex. 1021 (‘Lowering HAMR Near-Field Transducer Temperature via Inverse Electromagnetic Design’).

30. These technologies involve explicit tradeoffs. SLC, MLC, and TLC differ in capacity, cost, latency, and endurance. Over-provisioning provides another example: reserving more physical flash for controller-managed spare area can improve garbage collection, reduce write amplification, and increase endurance and sustained write performance, but it reduces user-visible capacity. Lower over-provisioning increases usable capacity and lowers cost per usable gigabyte, but can degrade performance and endurance as the drive fills. Ex. 1023 (‘Design Tradeoffs for SSD Performance’).

C. Traditional Interfaces Hid Device Complexity

31. Traditionally, device manufacturers hid this complexity behind a simple logical block interface. A host issued reads and writes; the device firmware decided how to translate logical addresses to physical media, buffer small writes, perform garbage collection, distribute wear, avoid defective blocks, and recover from media errors. If a hardware technology favored large contiguous writes, the

device could collect many small writes and destage them as a larger internal operation. If flash required block erasure before page rewriting, the flash translation layer handled the copying and indirection. This abstraction preserved compatibility and simplified software, but it prevented sophisticated users from directly exploiting knowledge of device behavior.

D. Hyperscale Systems Drove Software-Programmable Policies

32. A primary drawback of NAND memory is that each memory cell can only endure a finite number of write and erase cycles before it wears out. Manufacturers and designers address this problem with different approaches. For example, wear-leveling could be used to evenly distribute write and erase cycles across all NAND memory cells. As another example, systems could employ a defect management or bad block management policy that accounted for defective memory cells.

33. The rise of hyperscale operators changed the economics. Companies such as Google and Meta operate storage at such scale that small inefficiencies in capacity utilization, write amplification, power, or tail latency can have large total-cost-of-ownership effects. They also control much of the stack, including applications, distributed storage software, operating systems, and hardware procurement. That made hardware/software co-optimization practical: rather than

hiding all device characteristics, storage systems could expose selected characteristics to software and let software choose policies suited to the workload.

34. This design philosophy was established well before November 2016. Google's File System was designed around Google's application workloads and technological environment, rather than traditional file-system assumptions, and later discussions emphasized that Google could adjust both the file system and the applications above it. HDFS similarly targets large files, streaming access, high aggregate bandwidth, and fault tolerance on commodity hardware. Ceph uses policy-driven placement through CRUSH so clients and storage daemons can compute where objects and replicas should be placed without centralized lookup tables. Ex. 1024 ('Understanding Intrinsic Characteristics and System Implications of Flash Memory Based Solid State Drive'); Ex. 1025 ('The Google File System'); Ex. 1026 ('GFS: Evolution on Fast-Forward'); Ex. 1027 ('The Hadoop Distributed File System'); Ex. 1028 ('Ceph: A Scalable, High-Performance Distributed File System').

E. Host-Programmable SSD Policies Were Known Before November 2016

35. Policy-based storage management long predated the '715 patent. IBM implemented hierarchical storage management for MVS in 1978 and later introduced DFSMS, a policy-based framework using administrator-defined classes and groups for allocation, performance, availability, migration, backup, retention, and storage

groups. IBM Spectrum Scale likewise documented pre-2016 information lifecycle management policies for file placement, migration, deletion, listing, and external storage pools. Ex. 1029 ('CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data'); Ex. 1030 ('Two Decades of Policy-Based Storage Management for the IBM Mainframe Computer'); Ex. 1031 ('DFSMSHsm Primer'); Ex. 1032 ('GPFS: Advanced Administration Guide'); Ex. 1033 ('IBM Spectrum Scale (formerly GPFS)').

36. Hierarchical or tiered storage systems apply policies that migrate and store data based on certain characteristics of the data. For example, a policy could store data in different tiers, storing hot data (frequently used data) in the more expensive disk drives in one tier, while cold data (less frequently used or LFU data) in cheaper, long-term tape archives in another tier.

37. Device-level interfaces evolved in the same direction. NVMe 1.0 and 1.1 provided a high-performance host-controller interface with host-configurable features that affected device behavior. Open-Channel SSDs went further: the April 2016 Open-Channel SSD NVMe Specification exposed physical media geometry and internal parallelism so host software could implement flash-management policies such as data placement, bad-block handling, garbage collection, and wear leveling. Ex. 1034 ('NVM Express Revision 1.0'); Ex. 1035 (NVM Express

Revision 1.2’); Ex. 1036 (‘NVMe Specification Revision 1.2’); Ex. 1037 (‘Storage Security Subsystem Class: Opal Specification Version 2.00’).

38. SCSI/SBC provides the same point in a narrower form. Before the priority date, SBC-3 WRITE SAME and UNMAP commands let the host send command fields or directives that told the device to write a pattern, apply protection-related treatment, or deallocate/unmap logical blocks, with device firmware carrying out the resulting media operations. Ex. 1038 (‘SCSI Block Commands - 3 (SBC-3)’); Ex. 1039 (‘SBC-3 Thin Provisioning Commands’); Ex. 1040 (SCSI Block Commands - 4 (SBC-4’).

F. More-Complex Policies Used Request Attributes and Data-Processing Rules

39. RAID distributes content across multiple disks, commonly by striping data and storing parity information so that, if a disk fails, the missing content can be reconstructed from the surviving data and parity. Ex. 1051 (‘Understanding Hard Drive Media Defects White Paper – Servers’). The original RAID paper and later survey described how disk arrays used striping and redundancy to improve performance and reliability. Ex. 1041 (‘A Case for Redundant Arrays of Inexpensive Disks (RAID)’); Ex. 1042 (‘RAID: High-Performance, Reliable Secondary Storage’). Commercial storage products made RAID policy choices concrete in controller software. Pure Storage's FlashArray implemented RAID policies in the storage controller to protect against device failures, bit errors, and

performance variability while EMC's XtremIO used row-and-diagonal parity, and distributed free space rebuilding policies to improve data reliability. Ex. 1043 ('Purity: Building Fast, Highly-Available Enterprise Flash from Commodity Components'); Ex. 1044 ('XtremIO Data Protection (XDP): Flash-Specific Data Protection').

40. Pure Storage RAID-3D and EMC XtremIO XDP are examples of complex configurable storage policies implemented by storage systems: controller software selected and implemented policies for redundancy level, parity layout, spare-space use, rebuild behavior, fault tolerance, performance, and flash endurance. Other pre-priority publications also disclosed storage requests carrying policy-like criteria, metadata, or attributes for replication, placement, searchability, storage-server policy engines, cloud-storage routing, and rule-based file operations. Ex. 1005 (Olster); Ex. 1045 (Armitano); Ex. 1046 (Devarakonda); Ex. 1047 (deMilo); Ex. 1048 (Bone); Ex. 1049 (Prahlad); Ex. 1050 (Dejana).

G. Policy-Programmable Storage at Multiple Layers

41. By November 2016, storage systems were policy-programmable at virtually all layers. At the cluster level, distributed storage software could choose replication, erasure coding, placement, failure domains, rebalancing, and migration. At the server or storage-head level, file systems, object stores, and key-value stores could route requests to particular pools or devices. At the device level, controllers

could implement software-configurable policies for caching, error recovery, wear leveling, garbage collection, over-provisioning, bad-block avoidance, shingled-zone management, and data placement.

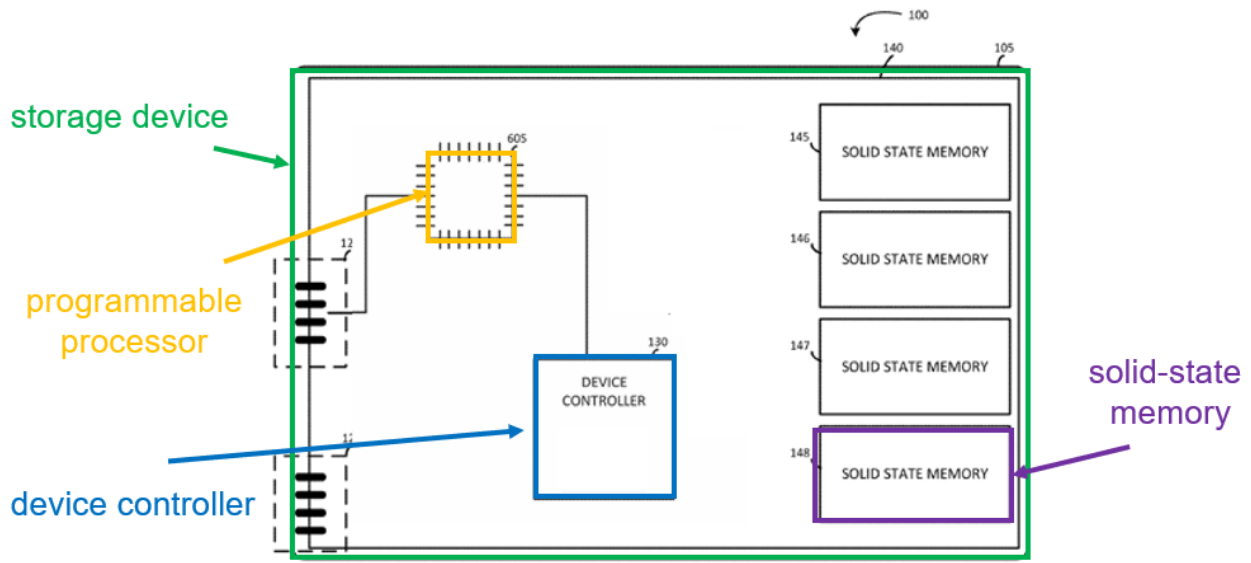
42. This layered policy architecture enabled unprecedented scale while preserving fault tolerance and performance. A system could apply one policy to write-heavy data and another to read-mostly data; one policy to temporary data and another to durable content; one policy to latency-sensitive data and another to archival data; or one policy to fast SLC-like regions and another to TLC/QLC bulk storage. The engineering overhead increased, but software-programmable policies enabled hardware/software co-optimization and total-cost-of-ownership improvements that were especially important at hyperscale. Against that background, storage device policies that are received, updated, and applied by a device controller were part of a well-developed progression from policy-based storage management to distributed placement policies, host-configurable device features, and software-defined flash. *See, e.g.*, Ex. 1005 (Olster); Ex. 1006 (Hashimoto); Ex. 1007 (Kilian); Ex. 1008 (Davis); Ex. 1009 (Eleftheriou); Ex. 1010 (Meir); Ex. 1011 (Salessi); Ex. 1015 (Lovinger).

V. THE '715 PATENT

A. Overview

43. The '715 patent generally relates to “data storage devices with configurable policy-based read and write behavior.” Ex. 1001, 1:26-28. The '715 patent explains that “data storage devices come in a wide variety of technologies and configurations such as solid-state drives, hard disk drives, floppy disk drives, and the like” but that “[e]ach of these devices (DRAM, NAND, magnetic disks) has imperfections.” *Id.*, 1:32-41. “[T]o deal with these imperfections, and still have a functional and performant device, all of these devices have trade-off decisions made that affect storage space, reliability, speed, etc.” *Id.*, 1:41-44.

44. But the '715 patent identifies the problem at the time as this: “These trade-off decisions have largely been made by the manufacturer at the time of design and/or manufacture” and provide users “very little to no influence” on how data is stored. *Id.*, 1:45-49. The '715 patent attempts to address this problem using a storage device policy that allows “a user to customize the operation of the storage device.” *Id.*, 3:16-17.



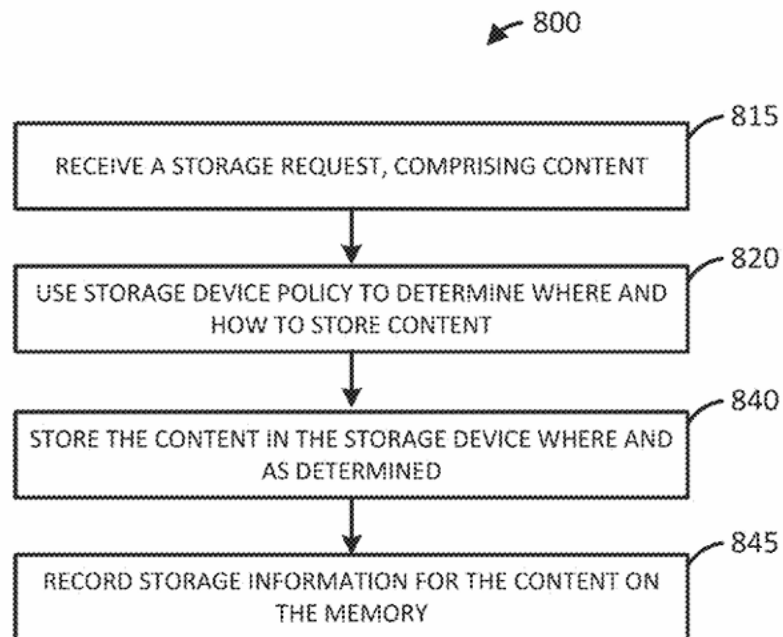
'715 patent, Ex. 1001, Figs. 1B and 6 (modified and with annotations in color)

45. The '715 Patent describes a **solid-state drive** including a **programmable processor**, **device controller**, and **solid-state memory**. *Id.*, FIGS. 1B, 6, 1:53-55. The '715 patent describes that the solid-state drive “uses a storage device policy...to determine where to store the content.” *Id.*, 14:41-43. “The determination can relate partially to the content being stored, and partially to the storage media.” *Id.*, 14:43-45.

46. According to the '715 patent, the **programmable processor**, which includes for example programmable logic device 605, “can be programmed by user of the drive 600 for any operation need that is desired.” *Id.*, 11:21-23. This means that the **programmable processor** can implement the storage policy, but also other functions such as “custom encryption through hardware” or “compression.” *Id.*,

11:23-41. The claims recite that a **device memory** is coupled to the **programmable processor** and stores an application (comprising code or program instructions) that directs the programmable processor on how to operate.

47. Moreover, the **programmable processor** transfers data to and receives data from the **solid-state memory** via the **device controller**. Specifically, the “programmable logic device 605 is shown connected between interface connector 125 and device controller 130.” *Id.*, 11:42-44.



Ex. 1001, Fig. 8A

48. The '715 patent depicts one implementation of writing content according to a storage device policy in Fig. 8A (which I have reproduced above). *Id.*, 14:31-15:5. As shown in the figure, at step 815, “[a] user sends a request to store

content.” *Id.*, 14:33-34. The ’715 patent explains that “this request comprises content to be stored.” *Id.*, 14:37-38. At step 820, the device controller 130 “uses a storage device policy...to determine where to store the content.” *Id.*, 14:41-42. The programmable processor will also process the content based on the storage policy, such as compressing or encrypting the content. *Id.*, 11:21-36. At step 840, “the device controller can write the content to the storage media 110 as determined in step 820.” *Id.*, 14:59-60.

49. The solid-state drive is also capable of reading content according to a storage device policy. Specifically, the device controller reads the requested content from the solid-state memory and transfers the content to the programmable processor, which then processes the content for transfer according to the storage policy, such as decompressing or decrypting the content.

50. The claimed solid-state drive can further include additional components, such as a communication interface, a data link, a printed circuit board, and a case to mount the printed circuit board and the communication interface, such as “case 105, which physically supports” the components of the drive (*id.*, 3:33-36).

B. Prosecution History

1. Prosecution of the ’023 Patent

51. I understand that the ’553 patent is the second continuation of the ’023 patent. During the prosecution of the ’023 patent, the pending claims of the ’023

patent were rejected as anticipated and obvious over U.S. Patent App. Pub. No. 2015/0052167 to Olster et al. (“Olster”). On May 3, 2019, the Examiner issued a non-final rejection of all pending claims as anticipated and obvious. On October 3, 2019, the Applicant amended the claims to recite “at least one storage device policy [that] indicates at least a storage density.” Ex. 1016, 163-164. On October 5, 2019, the Examiner issued a final rejection of all claims as obvious over Olster in view of US 2002/0036852 to Leow et al. (“Leow”).

52. The Examiner eventually allowed the claims after the Applicant removed references to storage density and amended them to require that the device controller receives “from a host” a storage policy that “indicates at least a recording method for writing content to the storage media.” Ex. 1016, 224.

2. Prosecution of the children of the ’023 patent, including the ’715 patent

53. I have reviewed the prosecution histories of the children of the ’023 patent, including the ’715 patent. While the Examiner only allowed the ’023 patent claims after the Applicant added the requirement that the device controller receives “from a host” a storage policy that “indicates at least a recording method for writing content to the storage media,” none of the claims of the ’023 patent’s children include this requirement. Instead, during the prosecution of the ’715 patent (which does not require the storage policy to come “from a host”), the Examiner only made double-patenting rejections, which the Applicant overcame with terminal

disclaimers. Based on my review of the prosecution histories, it appears that the challenged claims of the '715 patent were allowed to issue with little to no consideration on the substance of the claims.

C. Priority Date

54. I have been asked to assume that the priority date of the '715 patent is November 7, 2016. I have not been asked to determine whether the '715 patent is entitled to that priority date.

D. Person of Ordinary Skill in the Art

55. A person of ordinary skill in the art in 2016 would have had at least a bachelor's degree in electrical engineering, computer science, computer engineering, or a similar technical field, with at least two years of experience in the field of data storage systems. Additional experience could substitute for less education, and additional education could likewise substitute for less experience.

56. My opinions do not turn on this precise definition, and the challenged claim would be unpatentable from the perspective of any reasonable person of ordinary skill in the art at the relevant time.

57. I believe that I would have qualified as a person of at least ordinary skill in the art as of the earliest alleged November 7, 2016 priority date of the '715 patent. I believe that I have sufficient level of knowledge, experience, and education to provide an expert opinion in the field of the '715 patent.

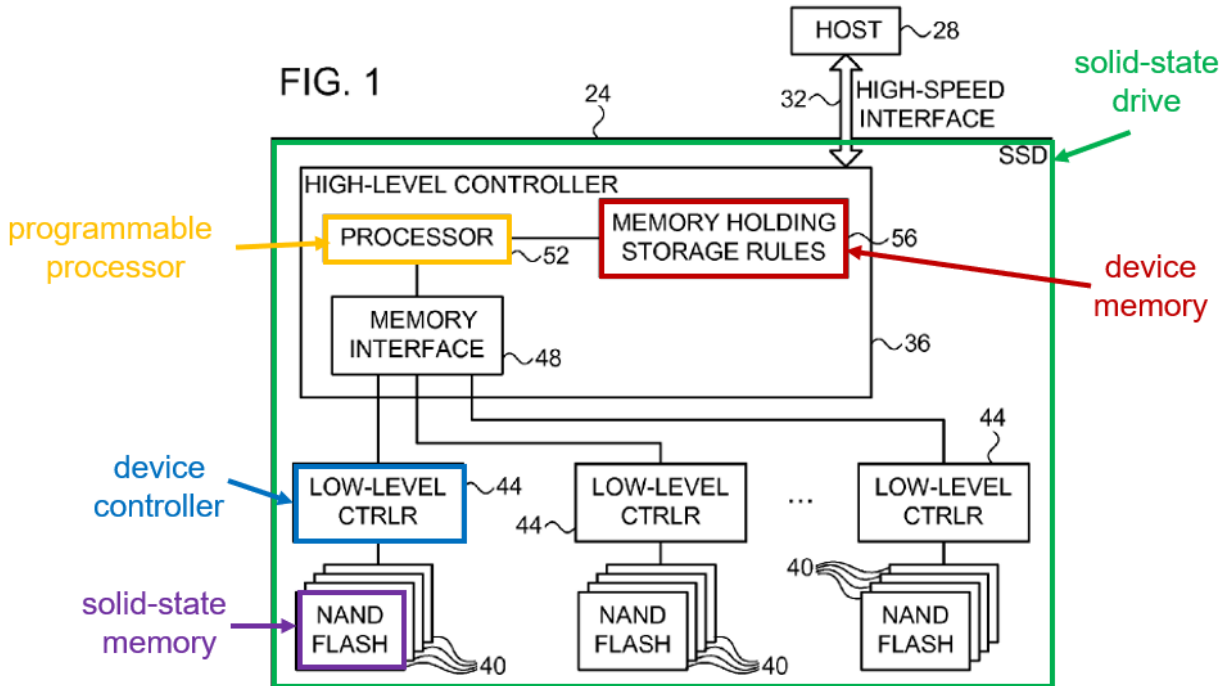
VI. CLAIM CONSTRUCTION

58. I have applied the plain meaning of the terms and phrases in the claims in my analysis, as a person of ordinary skill in the art would have understood those terms and phrases at the relevant time. I understand that the Board will construe the claims using the same claim construction standard that would be used in district courts. My declaration will establish that the prior art meets each of the claim limitations under its plain meaning. I have not been asked to consider any express constructions.

VII. THE PRIOR ART

A. Meir (Ex. 1010)

59. Meir was published in 2012 and describes data storage methods and systems that apply a policy. Ex. 1010, Abstract. In Meir's Fig. 1 (reproduced below), Meir depicts a **solid-state drive** ("SSD") 24 having "a high-level controller 36" as well as a "**low-level controller[]** 44." Each **low-level controller** is coupled with **memory units** 40. Ex. 1010, FIG. 1, [0031] ("each low-level controller [is] assigned to a group of one or more memory units 40" each of which "comprises processing circuitry (not shown in the figure.)").



Meir, Ex. 1010, Fig. 1 (with my color annotations)

60. The high-level controller 36 includes “a **processor 52** that carries out the various functions of the high-level controller” as well as a “**memory 56** [that] holds a definition of a policy that the memory controller applies in storing data in memory units 40.” Ex. 1010, FIG. 1, [0030] (“SSD 24 comprises a high-level controller 36, which stores data in one or more non-volatile memory units 40”), [0033] (“[h]igh-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example via low-level controllers 44), and a processor 52 that carries out the various functions of the high-level controller. High-level controller 36 comprises a memory 56...memory 56

holds a definition of a policy that the memory controller applies in storing data in memory units 40”).

61. Meir’s solid-state drive has configurable policy-based read and write behavior. Ex. 1010, [0014] (“a data storage method including receiving from a memory controller a communication...reporting a definition of a policy to be applied by the memory controller in accessing one or more memory units”). Indeed, Meir explains that “one or more of the rules in the memory controller policy may be configurable.” *Id.*, [0055].

62. Meir’s processor 52 is “programmed in software to carry out the functions described.” *Id.*, [0034]. Accordingly, Meir’s processor 52 is a programmable processor that can implement the storage policy including rules applied to the content such as “applying a redundant storage scheme such as Redundant Array of Inexpensive Disks (RAID) among the memory units.” *Id.*, [0025]. According to Meir, “redundancy rules may specify whether such a scheme is applied or not, as well as various parameters of the redundant storage scheme.” *Id.*, [0048]. “Such parameters may comprise, for example, the formulas or rules used by the memory controller in calculating parity information over the data.” *Id.* Software (comprising program instructions) can be stored along with the storage policy in memory 56, which is coupled to processor 52 and can direct processor 52 on how to operate. *Id.*, [0034] (“The software may be downloaded to the

processors...and/or stored on non-transitory tangible media, such as magnetic, optical, or electronic memory”).

B. Salessi (Ex. 1011)

63. Salessi describes “providing enhanced data processing and analysis” in an SSD. Ex. 1011, Abstract. Such “enhanced data processing” includes *process[ing] of content*, as Salessi describes that data includes content, such as social media posts and pictures and video content, among other types of content. *Id.* [0002] (“Every day, several quintillion bytes of data may be created around the world. This data comes from everywhere: posts to social media sites, digital pictures and videos, purchase transaction records, bank transactions, sensors used to gather data and intelligence, like climate information, cell phone GPS signal, and many others.”). In Salessi, *process[ing] the content* includes compressing or encrypting the content for storage and decompressing or decrypting the content upon retrieval. *Id.*, Abstract.

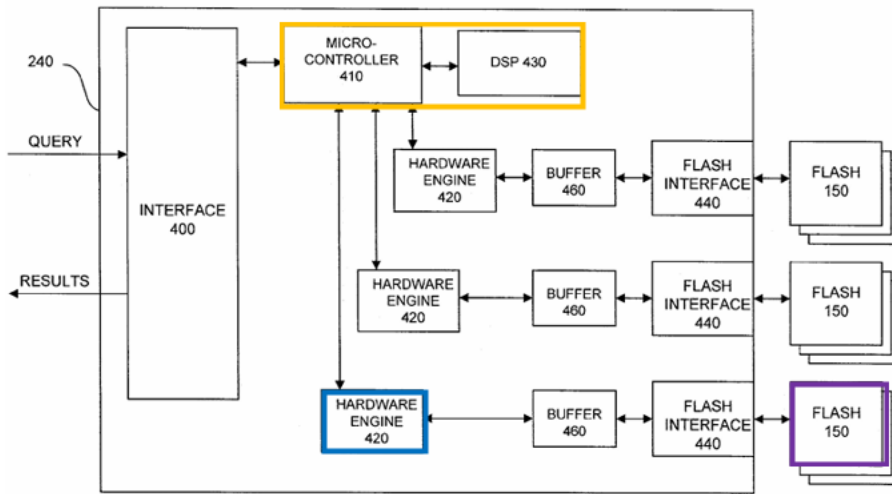


FIG. 7

Salessi, Ex. 1011, Fig. 7 (with my annotations in color)

64. Salessi describes an SSD that “includes flash memory and an SSD processing unit.” Ex. 1011, FIG. 7, Abstract, [0006]. Salessi teaches that its processing unit is “capable of processing the data...being stored or retrieved from the flash memory.” *Id.*, [0006]. “Processing capabilities incorporated into the SSD may include encryption and decryptions [and] compression and decompression.” *Id.* Salessi further teaches that the processing unit “us[es] a processing algorithm[] to generate processed data.” *Id.*, [0023].

65. By incorporating encryption, decryption, compression, and decompression capabilities into the SSD, Salessi teaches cloud data storage systems can perform “efficient processing of data stored in storage devices configured for cloud storage applications.” *Id.*, [0001]. Rather than “requir[ing] comparatively

slow reading and re-writing of the data,” Salessi teaches that performing processing of data “within the SSD” improves the system’s efficiency. *Id.*, [0036]. In particular, when used together with RAID, Salessi’s teachings of encryption, decryption, compression, and decompression “achieve similar advantages in optimizing performance and resource utilization, while taking advantage of efficiencies in RAID parity calculations and the number of physical inputs and outputs (I/Os) performed,” thereby “mak[ing] RAID controllers and subsystems more efficient.” *Id.*, [0057].

C. Chou (Ex. 1012)

66. Chou generally describes an SSD assembly and an assembly method for SSDs that does not require using screws, by “aligning a printed circuit board with a first cover and a second cover.” Ex. 1012, Abstract. In Chou, the SSD therefore includes printed circuit board mounting components within a case, similar to the requirements in claim 9 of the ’715 patent.

67. Specifically, Chou describes an SSD that includes a printed circuit board assembly (“PCBA”) and a housing that supports the PCBA. Ex. 1012, [0006] (“A SSD generally includes a printed circuit board assembly (PCBA) within a metallic housing.”), [0034] (“To assemble the SSD 100, the PCBA 112 is placed inside the upper and lower covers 114a and 114b”). As for the components mounted on the PCBA, Chou also describes “[o]ne or more memory modules and other

electronic components 122,” which can be the solid-state memory, device memory, programmable processor, device controller and other components. *Id.*, [0033], FIG. 2.

D. Lovinger (Ex. 1015)

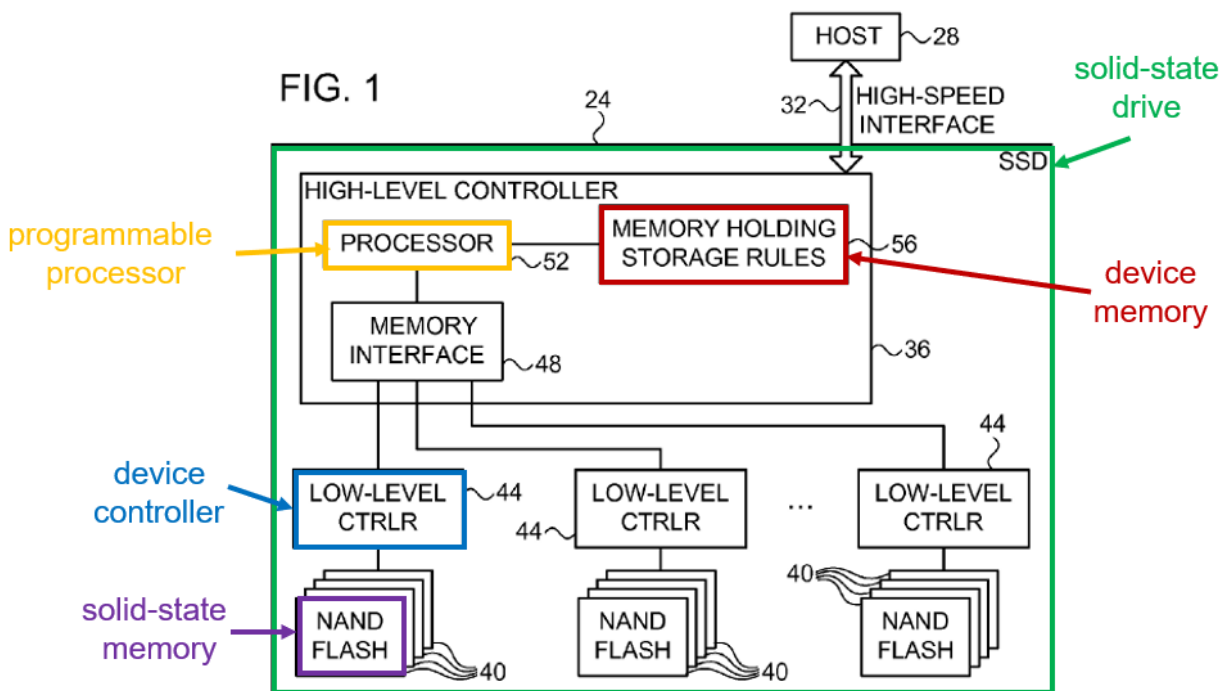
68. Lovinger describes a storage request that includes a policy, as required by claim 19. Lovinger describes a method “for storing content based on one or more attributes associated with the content.” Ex. 1015, [0003]. The client “may transmit content...with information corresponding to the retention policy.” *Id.*, [0040]. “[W]hen the content and retention policy have been received, the server 120 writes the content 112 to one or more storage devices associated with the server 120 according to the retention policy.” *Id.* This enables a client “to store content according to the retention policy without reprogramming device firmware.” *Id.*, [0041].

69. Lovinger also teaches a data storage system that includes “one or more storage devices that may be used to store and access various kinds of content 112 including, videos, music, documents, photos and the like.” Ex. 1015, [0017].

VIII. OPINIONS ON GROUND 1: MEIR IN VIEW OF SALESSI WOULD HAVE RENDERED OBVIOUS CLAIMS 1-2, 5-8, 10-11, 14-18, AND 20

70. In my opinion, the combination of Meir and Salessi renders obvious claims 1-2, 5-8, 10-11, 14-18, and 20. A person of ordinary skill in the art would

have found it obvious to combine the teachings of Meir and Salessi. Specifically, the data storage system resulting from the combination of the teachings of Meir and Salessi is a SSD including a processor that executes access commands from a host according to a policy that includes algorithms and rules for *process[ing] content*, including compression, decompression, encryption, and decryption.



Meir, Ex. 1010, Fig. 1 (annotated)

A. Motivation to Combine

71. In my opinion, a person of ordinary skill in the art would have been motivated to modify the software (which is the claimed *application*) and policy (which is the claimed *policy*) stored in memory 56 (which is the claimed *device*

memory) of Meir to direct the processor 52 (which is the claimed *programmable processor*) to *process content for storage based on a storage policy*, such as by compressing and encrypting the content to be stored according to algorithms and rules in the policy as taught by Salessi.

72. Salessi already expresses that compression and encryption rules may be used in the implementation of redundancy schemes like the one taught in Meir. Specifically, Meir teaches that redundancy schemes including RAID may be used. Ex. 1010, [0025] (“[T]he memory controller policy involves storing the data while applying a redundant storage scheme such as Redundant Array of Inexpensive Disks (RAID) among the memory units.”).

73. A person of ordinary skill in the art implementing the system of Meir would have naturally looked at teachings improving the performance of solid-state drives, particularly, those implementing RAID schemes, which were well-known by the relevant time frame. RAID was already a widely adopted storage architecture that combines multiple physical storage devices to improve performance, reliability, fault tolerance, and storage capacity. For example, when RAID is employed, data is distributed across multiple drives to increase read and write throughput, and duplicated to provide redundancy in the event of drive failure, or supplemented with parity information that enables recovery of lost data. RAID was therefore a

routinely-used technique for improving the performance and reliability of storage systems.

74. As Salessi expressly states, compression and encryption “may be used in a redundant array of independent disks (RAID) to achieve similar advantages in optimizing performance and resource utilization.” Ex. 1011, [0057] (“these embodiments can be used to make RAID controllers and subsystems more efficient”).

75. It is my opinion that it would have been well-understood by a person of ordinary skill in the art that modifying policies (like the policies in Meir) to include compression and encryption (like the functions in Salessi) would improve the overall performance and security of the data storage system. Ex. 1010, [0036] (“execut[ing] the memory access commands received from host 28 in accordance with a predefined policy . . . optimize[s] performance”); Ex. 1011, [0050] (“functions such as encryption of the data written to the flash memory 150 . . . , or similarly for compression . . . at write . . . increase[s] the effective storage capacity of the SSD 130”).

76. Compressing data reduces the size of data before it is stored or transmitted, which means that a storage system can use its available capacity more efficiently. This also means a reduction in the bandwidth needed to transmit the data across a network, resulting in faster load times, fewer bottlenecks, and improved

transfer performance. Moreover, because compression reduces the amount of storage required, it reduces the amount of physical data written to the storage media, which reduces wear on flash memory, reduces storage costs and also allows servers and databases to house more information on existing hardware. In this manner, compression was a well-known technique to improve both storage efficiency and endurance.

77. Encryption protects data by converting it to an unreadable form that can be accessed only by authorized users or systems possessing the appropriate decryption key. In this manner, encryption protects sensitive information (including confidential data) from unauthorized access even if the physical media is lost, stolen, intercepted during transmission, or otherwise compromised. Thus, encrypting data strengthens security by protecting against data theft or unauthorized access.

78. In my opinion, a person of ordinary skill in the art would have known that modifying Meir's policy to include Salessi's compression and encryption algorithm and rules would result in cost and resource savings, for the reasons explained above. Specifically, because compression allows the data storage system to store more data, compression can be used to expand the storage capacity of the system without expending resources on additional drives. Additionally, compression also reduces wear on the flash memory, which saves on hardware replacement cost.

79. Moreover, encryption comes with its own benefits. For example, when there is a drive failure, it may be necessary to expend resources in order to ensure that unencrypted sensitive data is protected. Using encryption conserves these resources that are otherwise expended if the data is unencrypted. In this manner, using encryption allows systems to bypass certain resource-intensive protocols designed to ensure that sensitive data on a failed drive is not accessible. Instead of needing to implement these resource-intensive protocols, a failed drive containing encrypted data can simply be discarded because the encryption already ensures the data on the drive is not accessible.

80. As I explained above, storing less data through compression reduces wear on SSDs. NAND memory (the primary chip inside SSDs) has a significant disadvantage, in that it has limited lifespan. In other words, each memory cell can only endure a finite number of write and erase cycles before it wears out. This is because each physical write contributes to the gradual wear of the storage media. By compressing data before it is written, the SSD stores fewer bytes for compressed content, which reduces the number of physical write operations and associated erase cycles. Thus, in this manner, compressing data will extend the life of the physical drives, improve storage efficiency, and reduce the frequency at which NAND blocks are written and rewritten, thereby reducing drive failure.

81. In my opinion, a person of ordinary skill in the art would have found it obvious to modify Meir's policy to include Salessi's encryption, which would have been driven by regulatory compliance requirements. At the relevant time, privacy mandates including GDPR had just been enacted. Taking GDPR as an example, GDPR is a European Union data protection law that establishes requirements for processing and protecting personal data. GDPR requires organizations to implement a level of security appropriate to the risk, and specifically identifies encryption as an example of a technical measure that can help protect personal data from unauthorized access. Encryption would have rendered certain personal data requiring protection to be unreadable.

82. Moreover, by implementing encryption through a policy provides flexibility to administrators and users. Rather than requiring that all stored data be encrypted, policy-based encryption allows the administrators and users to selectively apply encryption to designated data. For example, administrators may specify which data should be encrypted based on certain criteria, such as file type, storage location, user, or data classification. Administrators may also be able to designate particular files or directories for encryption. In this manner, a policy-based approach to encryption enables the encryption to be tailored to the sensitivity of the data, while saving on the overhead associated with encrypting information that does not actually

require protection. Thus, a policy-based encryption approach reduces the encryption necessary, which in turn reduces the processing time for reads and writes.

B. Reasonable Expectation of Success

83. In my opinion, a person of ordinary skill in the art would have had a reasonable expectation of success in modifying Meir's processor 52 to perform compression, decompression, encryption, and decryption (as taught by Salessi) according to the storage policy.

84. The structures and components of the SSDs in Meir and Salessi are similar. Ex. 1010, FIG.1; Ex. 1011, FIGS. 2, 4, 7. Specifically, the SSDs in both references include a memory controller or processing unit that interfaces with the solid-state memory. For example, in Meir, "SSD 24 comprises multiple low-level controllers 44, each low-level controller assigned to a group of one or more memory units 40." Ex. 1010, [0031]. Moreover, Meir's "[h]igh-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example via low-level controllers 44)." *Id.*, [0033]. Similarly, in Salessi, "[t]he SSD processing unit 240 includes . . . a hardware engine 420 [and] there may be a separate hardware engine 420 for each flash channel," and "the microcontroller 410 may be connected to a plurality of hardware engines 420." Ex. 1011 [0047], [0055]. Thus, both references describe a memory controller or processing unit that communicates with multiple units of memory.

85. Further, the memory controller (Meir) and processing unit (Salessi) can optimize performance through parallelism and pipeline rules. In Meir, “[b]eing aware of the parallelism rule, the host is able to assign LBAs to memory access commands in a way that increases the parallelism among memory units 40 and/or low-level controllers 44.” Ex. 1010, [0042] (emphasis added). Similarly, Salessi describes that “in one embodiment of an SSD processing unit 240, the microcontroller 410 may be connected to a plurality of hardware engines 420, each of which may access a plurality of flash memories 150 This structure allows searches to be performed in parallel, resulting in shorter processing times” Ex. 1011, [0055]. By enabling multiple flash devices to operate simultaneously, both Meir’s and Salessi’s SSDs are able to take advantage of parallel pipelines, which distribute data and I/O requests across multiple independent flash resources, increasing throughput. Thus, both structures in Meir and Salessi allow for parallelism that reduces I/O bottlenecks and improves responsiveness.

86. Thus, as I depict below in Fig. 1 of Meir and Fig. 7 of Salessi, both with my colored annotations, the Meir and Salessi structures are similar in that both teach a *programmable processor* that interfaces with multiple *device controllers*. Specifically, Meir teaches a high-controller 36 including processor 52 (*programmable processor*) that interfaces with low-level controllers 44 (*device*

controller), and Salessi teaches a microcontroller 410 (*programmable processor*) that interfaces with hardware engines 420 (*device controller*).

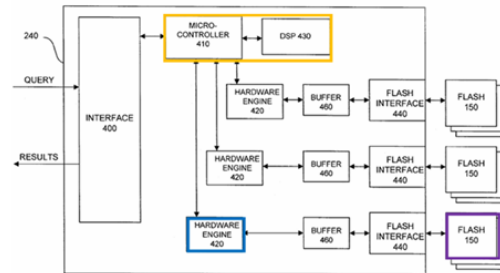
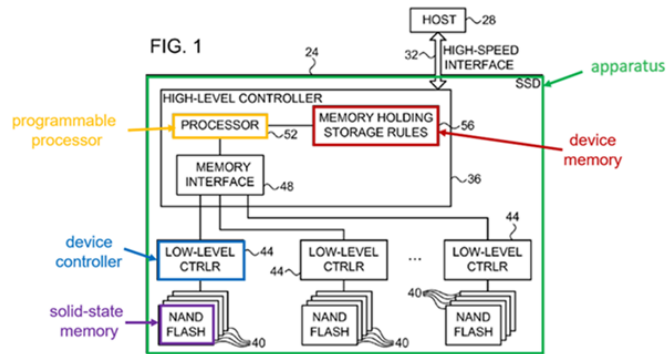


FIG. 7

Meir Ex. 1010, Fig. 1; Salessi Ex. 1011, Fig. 7 (both annotations in color)

87. Meir and Salessi both reference programmable processors that execute various operations including processing data. Specifically, in Meir, “[h]igh-level controller 36 comprises . . . a processor 52 that carries out the various functions of the high-level controller.” Ex. 1010, [0033] (emphasis added). And in Salessi, “the SSD 130 includes an SSD processing unit 240 that is designed for data operations or analytics, such as search and analysis of a large volume of unstructured data.” Ex. 1011, [0042] (emphasis added). “The SSD processing unit 240 includes . . . a microcontroller 410 [and] a digital signal processor (DSP) or DSP core 430”. *Id.*, [0047]

88. Another similarity between Meir and Salessi is that the SSDs of both references also include *device controllers* that interface with the *solid-state memory*.

Specifically, in Meir, “a high-level controller 36, which stores data in one or more non-volatile memory units 40 [that] comprise NAND Flash devices or dies.” Ex. 1010, [0030], *see also* [0033] (“High-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example via low-level controllers 44), and a processor 52 that carries out the various functions of the high-level controller.”). And in Salessi, “[t]he SSD 340 may include . . . flash memory 150” and “the microcontroller 410 may be connected to a plurality of hardware engines 420, each of which may access a plurality of flash memories 150”. Ex. 1011, [0042], [0055] (emphases added).

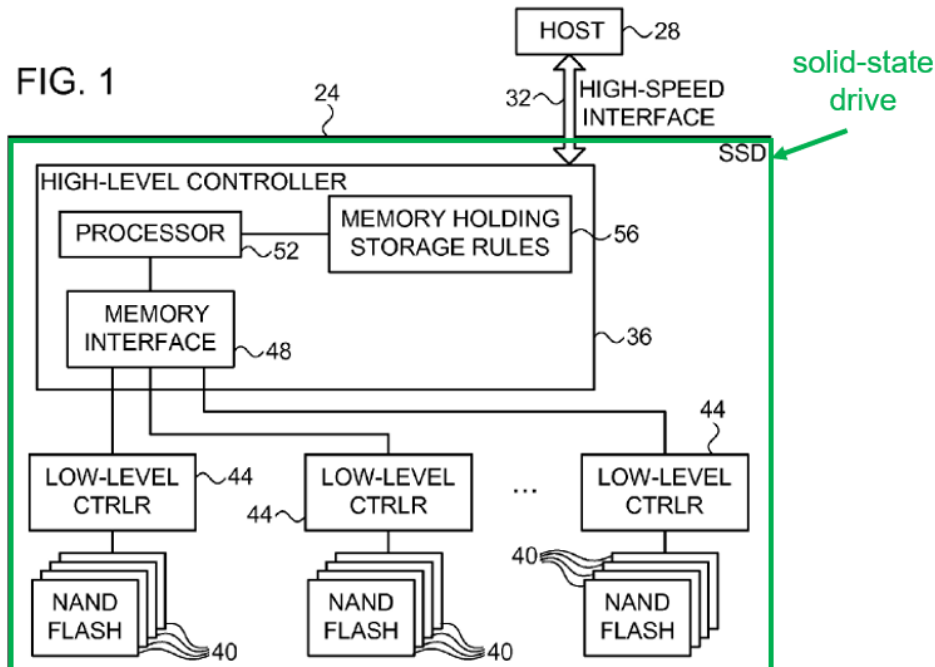
89. Given the similarities in structure, function, and hardware components between Meir and Salessi, the proposed modification to use Meir’s policy to include Salessi’s algorithms and rules for compression, decompression, encryption, and decryption would have simply involved a change of programming. Indeed, Meir already teaches that “a given rule in the policy is configurable.” Ex. 1010, [0011]. Further, Salessi teaches that “processing algorithms” are used to implement compression, decompression, encryption, and decryption. Ex. 1011, [0007]-[0011]. In my opinion, these teachings taken together with the knowledge of a person of ordinary skill in the art, configuring Meir’s rules in the storage policy to include Salessi’s processing algorithms regarding compression, decompression, encryption,

and decryption would have been well within the skill set of a person of ordinary skill in the art at the relevant time.

C. Independent Claim 1

1. Preamble 1[pre]: “A solid-state drive comprising:”

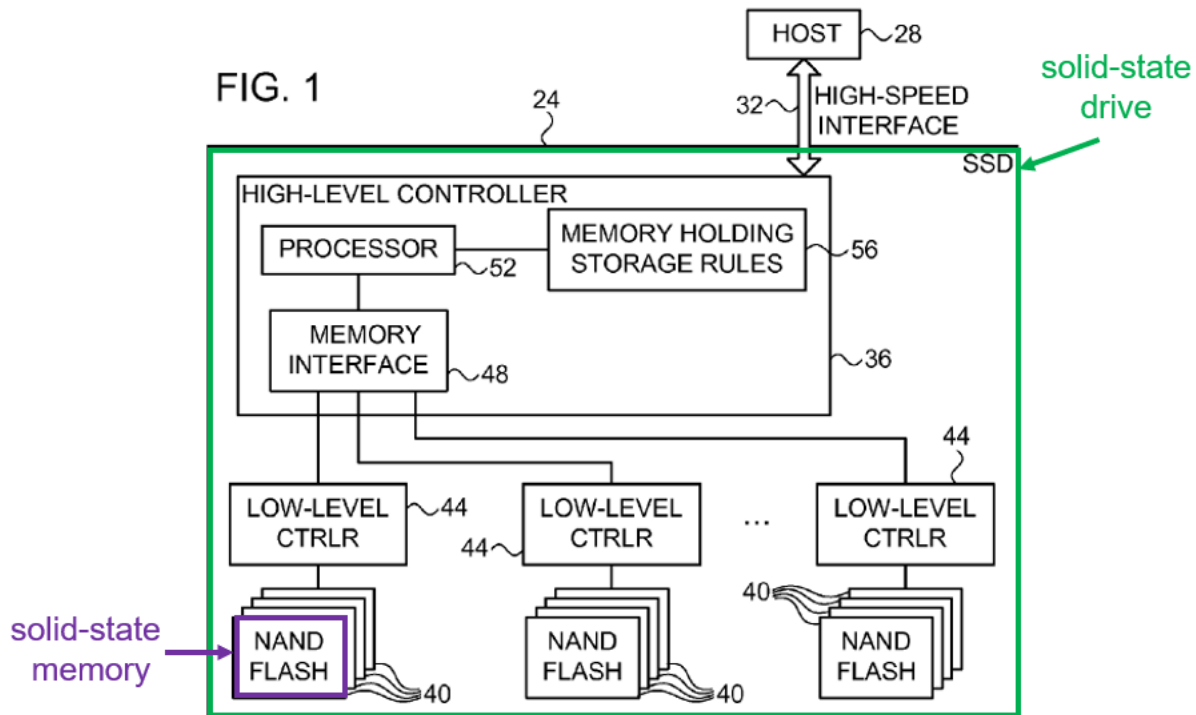
90. I have been asked to assume the preamble of claim 1 is limiting. Under that assumption, it is my opinion that Meir discloses the preamble of claim 1, which recites “[a] solid-state drive.” Meir relates to a solid-state drive with configurable policy-based memory access operations. See my discussion above in Section VII.A. Meir describes a solid-state drive (“SSD”) 24 (*solid-state drive*). Ex. 1010, FIG. 1, [0029].



Ex. 1010, Fig. 1 (annotations in color)

2. Limitation 1[a]: “a solid-state memory;”

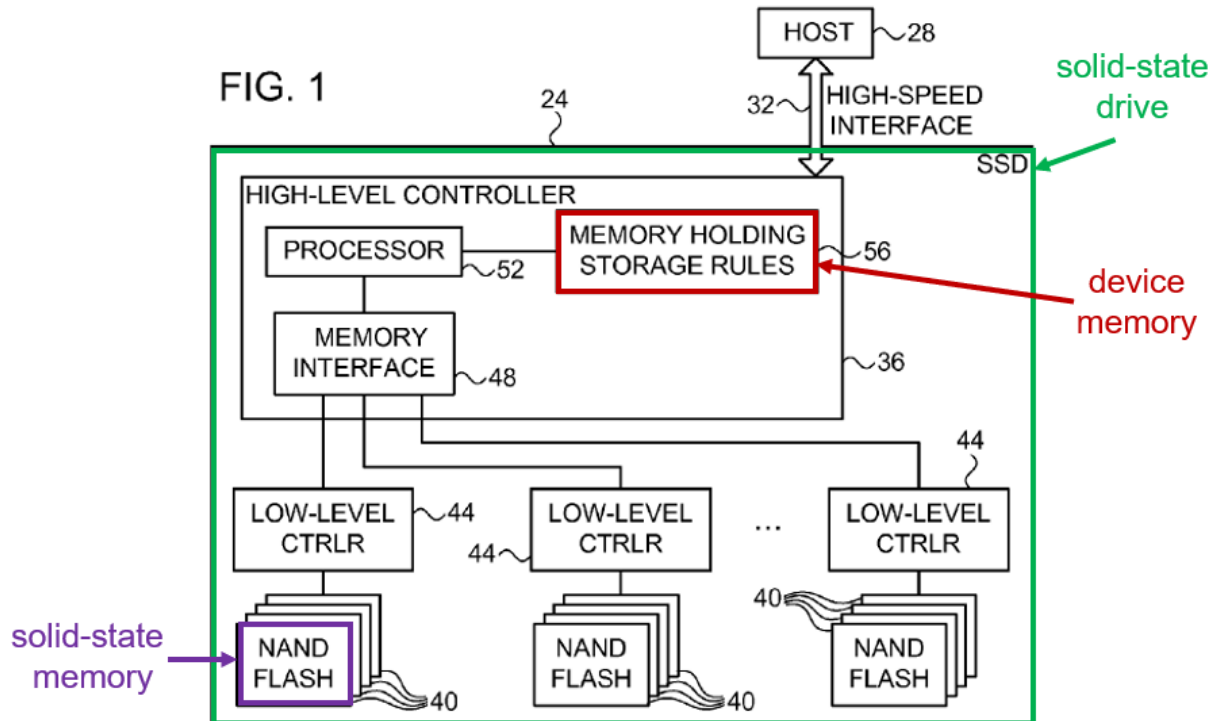
91. In my opinion, Meir discloses 1[a]: “a solid-state memory.” Meir’s SSD 24 comprises NAND Flash memory units 40 (*solid-state memory*). Ex. 1010, FIG. 1, [0030] (“SSD 24 comprises a high-level controller 36, which stores data in one or more non-volatile memory units 40.”). “[M]emory units 40 comprise NAND Flash devices or dies” or “any other suitable type of non-volatile memory units.” *Id.* NAND Flash and other non-volatile memories are types of solid-state memories. Flash devices (i.e., flash memory) are ubiquitous in solid-state devices. A person of ordinary skill in the art would have understood that flash memory is a type of solid-state memory because it has no moving parts and stores data electronically using microchips.



Ex. 1010, Fig. 1 (annotations in color)

3. Limitation 1[b]: “a device memory;”

92. In my opinion, Meir discloses 1[b]: “a device memory.” Meir describes memory 56 (*device memory*). Ex. 1010, FIG. 1, [0033] (“memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40”), cl. 18 (“[a] memory controller, comprising: a memory, which is configured to hold a definition of a policy to be applied by the memory controller in accessing one or more memory units”).

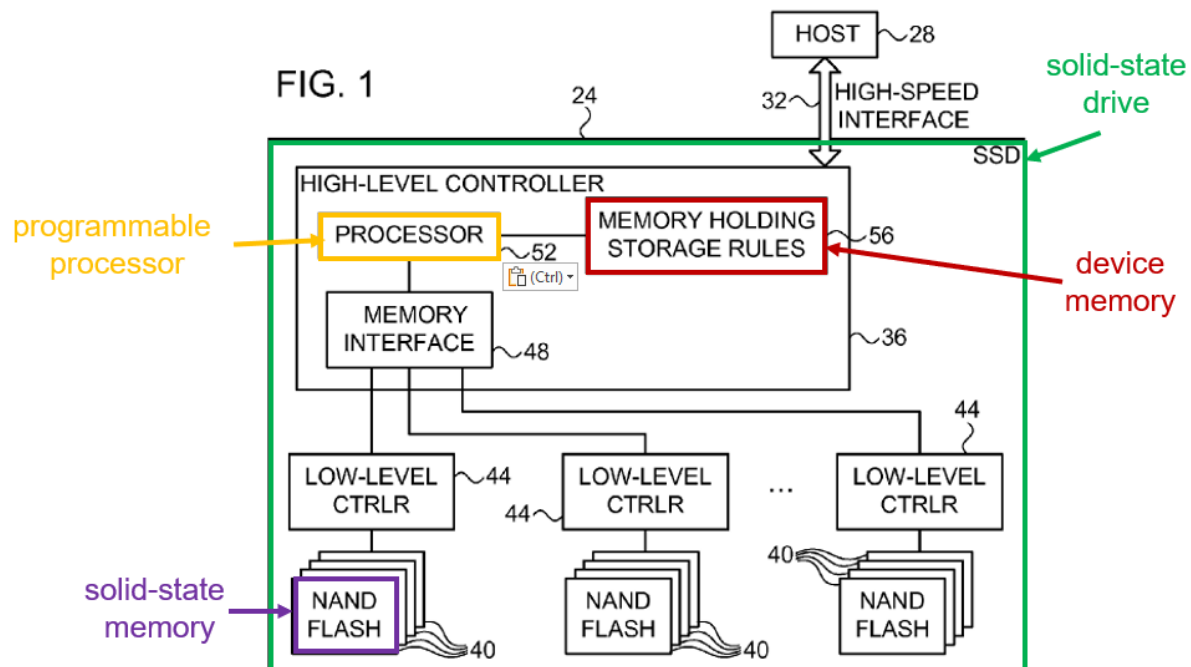


Ex. 1010, Fig. 1 (annotations in color)

4. **Limitation 1[c]: “a programmable processor operatively coupled to the device memory;”**

93. In my opinion, Meir discloses 1[c]: “a programmable processor operatively coupled to the device memory.” Meir describes a processor 52 (*programmable processor*). Ex. 1010, FIG. 1, [0033] (“a processor 52 that carries out the various functions of the high-level controller”), [0034] (“the memory controller comprises one or more general-purpose processors”), cl. 18 (“[a] memory controller, comprising . . . a processor, which is configured to . . . execute the received memory access commands in the memory units”). Programmable processors are chips or devices that run software instructions to perform different tasks. Types of programmable processors include, but are not limited to, central

processing units (CPUs), digital signal processors (DSPs), microcontrollers, and programmable logic controllers (PLCs), and field-programmable gate arrays (FPGAs). A person of ordinary skill in the art would have known that a processor like processor 52 that carries out the functions of a controller would be programmable, because the processor would have to be programmed to carry out those functions. Processor 52 (*programmable processor*) is *operatively coupled* to memory 56 (*device memory*) as illustrated by the connection between these two blocks in Figure 1. *Id.*

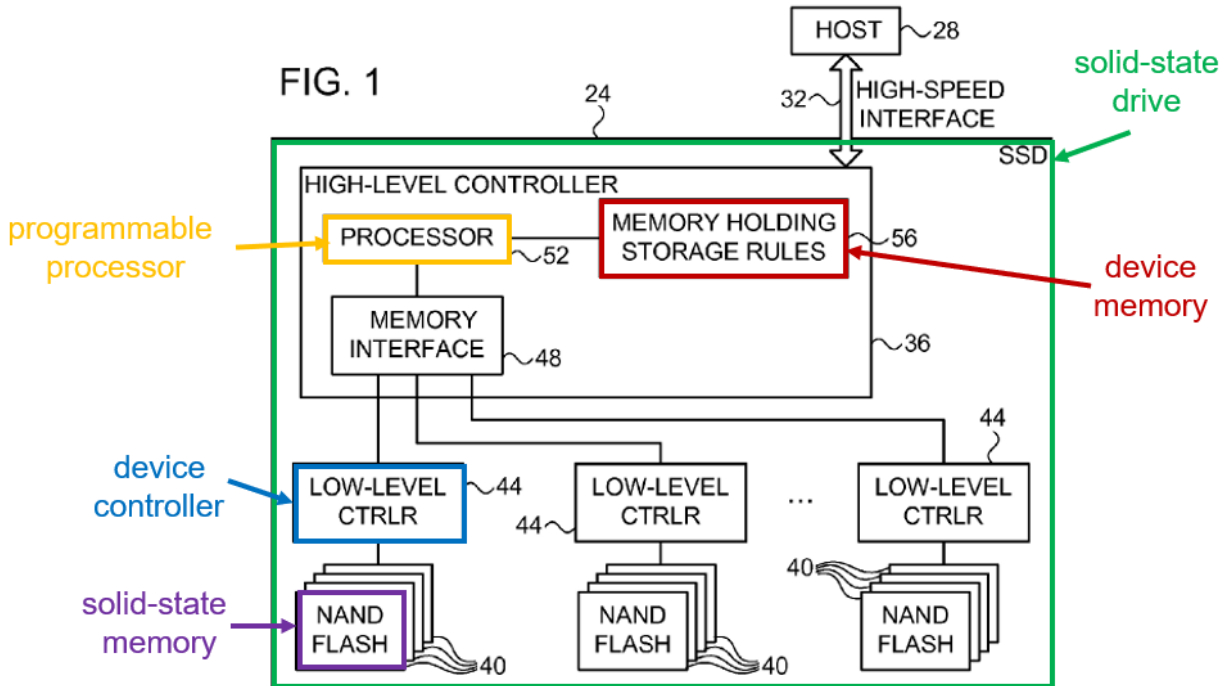


Ex. 1010, Fig. 1 (annotations in color)

5. Limitation 1[d]: “a device controller operatively coupled with the solid-state memory and the programmable processor; and”

94. In my opinion, Meir discloses 1[d]: “a device controller operatively coupled with the solid-state memory and the programmable processor.” Meir describes a low-level controller 44 (*device controller*). *Id.*, FIG. 1, [0031] (“SSD 24 comprises multiple low-level controllers 44”). As illustrated in Figure 1 by the connection between low-level controller 44 and memory unit 40, low-level controller 44 (*device controller*) is *operatively coupled with* memory unit 40 (*solid-state memory*). *Id.*

95. In addition, the low-level controller 44 (*device controller*) is also *operatively coupled to* processor 52 (*programmable processor*) through memory interface 48 which permits processor 52 to communicate with memory units 40 via the low-level controller 44. *Id.*; *see also* [0033] (“High-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example via low-level controllers 44), and a processor 52 that carries out the various functions of the high-level controller.”).



Ex. 1010, Fig. 1 (annotations in color)

6. **Limitation 1[e]: “an application stored on the device memory and comprising instructions that, when executed by the programmable processor, direct the programmable processor to:”**

96. In my opinion, Meir in view of Salessi renders obvious 1[e]: “an application stored on the device memory and comprising instructions that, when executed by the programmable processor, direct the programmable processor to.” Meir describes “the memory controller may . . . run suitable software” (*an application . . . when executed by the programmable processor, direct the programmable processor*). Ex. 1010, [0034]. Meir describes in further detail:

Some or all of the functions of the memory controller may be implemented in hardware. Alternatively, the memory controller may comprise one or more microprocessors that run suitable software, or a combination of hardware and software elements. In some

embodiments, the memory controller comprises one or more general-purpose processors, which are programmed in software to carry out the functions described herein. The software may be downloaded to the processors in electronic form, over a network, for example, or it may, alternatively or additionally, be provided and/or stored on non-transitory tangible media, such as magnetic, optical, or electronic memory.

Id.; *see also id.*, [0032] (“the high-level controller, the low-level controllers and the processing circuitry in the memory units are referred to collectively as a memory controller”).

97. Meir explains that the software can be *stored* in the memory 56 (*device memory*) for processor 52 to execute. Specifically, Meir explains that “[t]he software may be . . . stored on non-transitory tangible media.” *Id.*, [0034]. Meir also describes that the software *comprises program instructions* that cause the processor to perform functions when the instructions are executed. *Id.* (the software is programmed “to carry out the functions” performed by the processor). As Meir teaches, for a controller, like the processor 52, to perform processes when the instructions in the software are executed, that software must be available to the processor 52. Thus, the software would be available to processor 52 through a memory coupled to it, memory 56.

98. Software comprises a collection of program instructions stored in a computer-readable medium that, when executed by one or more processors, cause a computing device to perform one or more processes or functions. These instructions

define the sequence of operations carried out by the processor, including receiving inputs, processing data, performing calculations or logical operations, accessing memory or storage, communicating with other hardware or software components, and generating outputs. By executing the programmed instructions, the processor carries out the intended functionality of the software, enabling the computing device to perform tasks ranging from basic data processing to complex system operations.

99. As I further explain below for 1[e.i], a person of ordinary skill in the art would have been motivated to modify Meir's software (*application*) to direct its processor 52 to perform Salessi's teachings of compression/decompression and encryption/decryption in accordance to algorithms and rules defined by the policy. Specifically, Meir already explicitly states that "[t]he host may...modify the memory controller's policy and thus improve performance" (Ex. 1010, [0027]), and Salessi improves "RAID controllers and subsystems" by making them "more efficient" (Ex. 1011, [0057]).

7. Limitation 1[e.i]: "process content for storage based on a storage policy;"

100. In my opinion, Meir in view of Salessi renders obvious 1[e.i]: "process content for storage based on a storage policy." Meir teaches that "SSD 24 comprises a high-level controller 36, which stores data in one or more non-volatile memory units 40." Ex. 1010, [0030]. A person of ordinary skill in the art would have readily understood that Meir's data is *content for storage* at least because it is being stored

on the non-volatile memory units. Meir explains that the data is transmitted and stored via memory access commands: “[h]ost 28 typically sends memory access commands (e.g., read and write commands) to SSD 24 over interface 32” and “[t]he memory system comprises a memory controller which receives from the host memory access commands (e.g., read and write commands) and executes the commands in the memory units.” Ex. 1010, [0030], [0020].

101. It is also my opinion that Meir describes that the memory controller will *process* commands *based on a storage policy*. For example, Meir describes that “[t]he memory controller typically executes the memory access commands received from the host in accordance with a certain policy.” Ex. 1010, [0021], [0033] (“memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40”). Meir elaborates that “[t]he policy comprises a set of rules, for example parallelization rules, pipeline rules, prioritization rules, redundancy rules, power consumption rules and/or memory unit characteristics rules.” *Id.*, [0037].

102. Parallelization rules, pipeline rules, and prioritization rules are storage policies because they dictate how data storage is implemented including how the storage tasks are delegated among the storage components and in what order the data is stored. As an example of policy-based storage discussed by Meir, when applying the redundancy rules from the policy, the software (*application*) can direct the

processor 52 to *process* the data (*content for storage*) by implementing Redundant Array of Inexpensive Disks (RAID), e.g., striping the data dividing it into blocks, mirroring the data (making a copy of the data to store a redundant copy in other memory units), calculating parity information on the data, and/or storing the parity information together with the data it relates to. *Id.*, [0025] (“[t]he memory controller policy involves storing the data while applying a redundant storage scheme such as Redundant Array of Inexpensive Disks (RAID) among the memory units”) *See* my discussion above in Section IV.F. In my opinion, a person of ordinary skill in the art would have understood that applying RAID (e.g., calculating parity or similar redundancy coding information) constitutes processing the data content at least because the data is divided into blocks, copied, and used to calculate parity information..

103. To the extent that it is determined that Meir does not disclose that the software (*instructions*) executed directs the processor 52 (*programmable processor*) to *process* the data (*content for storage*) included with the commands *based on a storage policy*, it is my opinion that this is rendered obvious by the system resulting from the combination of the teachings of Meir and Salessi, as I discuss in further detail below.

104. Similar to Meir’s SSD (which includes memory units 40 and a memory controller), Salessi’ teaches an SSD that “includes flash memory and an SSD

processing unit.” Ex. 1011, FIG. 7, [0006]; *see also* Ex. 1010, FIG. 1. Salessi teaches that its processing unit can *process content for storage* because it is “capable of processing the data . . . being stored or retrieved from the flash memory.” Ex. 1011, [0006].

105. As I explained for the Overview of Salessi in Section VII.B, Salessi’s “enhanced data processing” includes *process[ing]* of *content*, as Salessi’s data includes *content*, such as social media posts and pictures and video content, among other types of content. *Id.* [0002] (“Every day, several quintillion bytes of data may be created around the world. This data comes from everywhere: posts to social media sites, digital pictures and videos, purchase transaction records, bank transactions, sensors used to gather data and intelligence, like climate information, cell phone GPS signal, and many others.”).

106. Salessi gives specific examples of processing: “Processing capabilities incorporated into the SSD may include encryption and decryptions [and] compression and decompression.” *Id.*; *see also id.*, [0042] (“the SSD 130 includes an SSD processing unit 240 that is designed for data operations or analytics. . . . and other operations such as compression, decompression, encryption, and decryption”). Salessi explains that the processing unit “process[es], using a processing algorithm, to generate processed data” (*content*) and “stor[es] . . . the processed data in the flash memory.” *Id.*, [0023].

107. In the combination of Meir and Salessi, the policy stored in memory 56 as taught by Meir controls compression, decompression, decryption, and encryption algorithms and instructions as taught by Salessi. Meir already teaches that the “one or more of the rules in the memory controller policy may be configurable” and “the host may issue to the memory controller a command that configures or modifies the memory controller, so as to improve performance.” Ex. 1010, [0045] (emphasis added). A person of ordinary skill in the art would take advantage of this configurability to modify the memory controller policy of Meir to incorporate the processing algorithms and instructions of Salessi, in such a manner so that Meir’s policy would define which data to apply Salessi’s processing algorithms to and where the data would be transferred and stored. Ex. 1011, [0006]. Such a skilled person would have understood that incorporating compression, decompression, encryption, and decryption in storage device policies maximizes storage efficiency and secures sensitive data.

108. As I discussed above in Section VIII.A, a person of ordinary skill in the art would have been motivated to modify the software (*application*) and *policy* stored in memory 56 (*device memory*) of Meir to direct the processor 52 (*programmable processor*) to compress and encrypt the data to be stored according to algorithms and rules in the policy because it would optimize the performance and improve the security of the system. *See* Section VIII.A; Ex. 1010, [0036]

(“execut[ing] the memory access commands received from host 28 in accordance with a predefined policy . . . optimize[s] performance”); Ex. 1011, [0050] (“functions such as encryption of the data written to the flash memory 150 . . . , or similarly for compression . . . at write . . . increase[s] the effective storage capacity of the SSD 130”). Meir describes configurability of its policies, so a person of ordinary skill in the art would know that those policies could be modified to incorporate the benefits of Salessi’s algorithms to further bolster the performance and security of the system.

109. Further, as I discussed above in Section VIII.B, such a person of ordinary skill would have had a reasonable expectation of success since the SSDs of Meir and Salessi have similar structures and components, and Salessi expressly teaches that the use of its processing algorithms in such similar systems. *See* Section VIII.B; Ex. 1010, FIG. 1, [0033] (“High-level controller 36 comprises a memory interface 48. In particular, memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40. The policy comprises a set of rules, for example one or more parallelization rules, pipeline rules and/or prioritization rules.”); Ex. 1011, FIG. 7, [0006] (“In one embodiment, the SSD includes flash memory and an SSD processing unit capable of processing the data in the flash memory, or being stored or retrieved from the flash memory, without returning all of the data to be searched to a host CPU outside of the SSD.”).

8. Limitation 1[e.ii]: “transfer the content to the device controller; and”

110. In my opinion, Meir discloses 1[e.ii]: “transfer the content to the device controller.”

111. Meir describes that the software (*application*) directs the processor 52 to send the data (*transfer the content*) to the low-level controller 44 (*device controller*) to store. Meir also teaches that the storage policy sets parallelism rules. Ex. 1010, [0021] (“[t]he policy may specify, for example, parallelism rules”), [0037] (“Parallelism rules define how the memory controller distributes the memory access commands among the memory units.”), [0041], (“a parallelism rule may define ranges of LBAs that can be accessed in parallel by the memory controller”).

112. To be specific, once a write command is received, the processor 52 processes the command according to the policy and transfers the data to the assigned low-level controller 44. *Id.*, [0005]. In particular, the parallelism rules defined by the policy will “specify distribution of the memory access commands among the memory units.” *Id.* Meir describes that “a mapping, defined in the memory controller, between logical addresses associated with the memory access commands and respective physical storage location in the memory units” will be used by processor 52 via its software to direct the access command to the appropriate low-level controller 44 (the low-level controller assigned to those memory addresses). *Id.*, [0007], [0031] (“each low-level controller assigned to a group of one or more

memory units 40”); *see also id.*, [0009], [0023], [0031] (“logical-to-physical address translation is performed by the high-level controller”). In particular, because “a parallelism rule may define . . . ranges of LBAs that are mapped to respective low-level controllers,” processor 52 will use the parallelism rule that is part of the policy to identify the assigned low-level controller 44 to send the data to be stored.” *Id.*, [0041]; *see also* [0041] (“a certain subset of bits in the logical address may indicate the memory unit or low-level controller in which this address is assigned”).

113. In other words, a storage policy that sets parallelism rules defines how data and I/O operations should be distributed across the available storage resources, so that multiple operations can occur concurrently across multiple storage devices. The policy dictates how the storage system uses parallel hardware resources, typically to achieve certain performance, efficiency, and/or reliability goals. In this manner, the policy enables the processor to optimize performance, resource utilization, endurance, or other system objectives.

114. In Meir, because translation is performed by the high-level controller (*id.*, [0031]), and because the logical addresses are mapped to low-level controllers (*id.*, [0041]), a person of ordinary skill in the art would have understood that the data to be stored is transferred from the host to high-level controller 56, and the low-level controller 44 couples the processor 52 of the high-level controller 56 to the memory units 40. Given that the high-level controller includes processor 52 (*id.*, [0033]) and

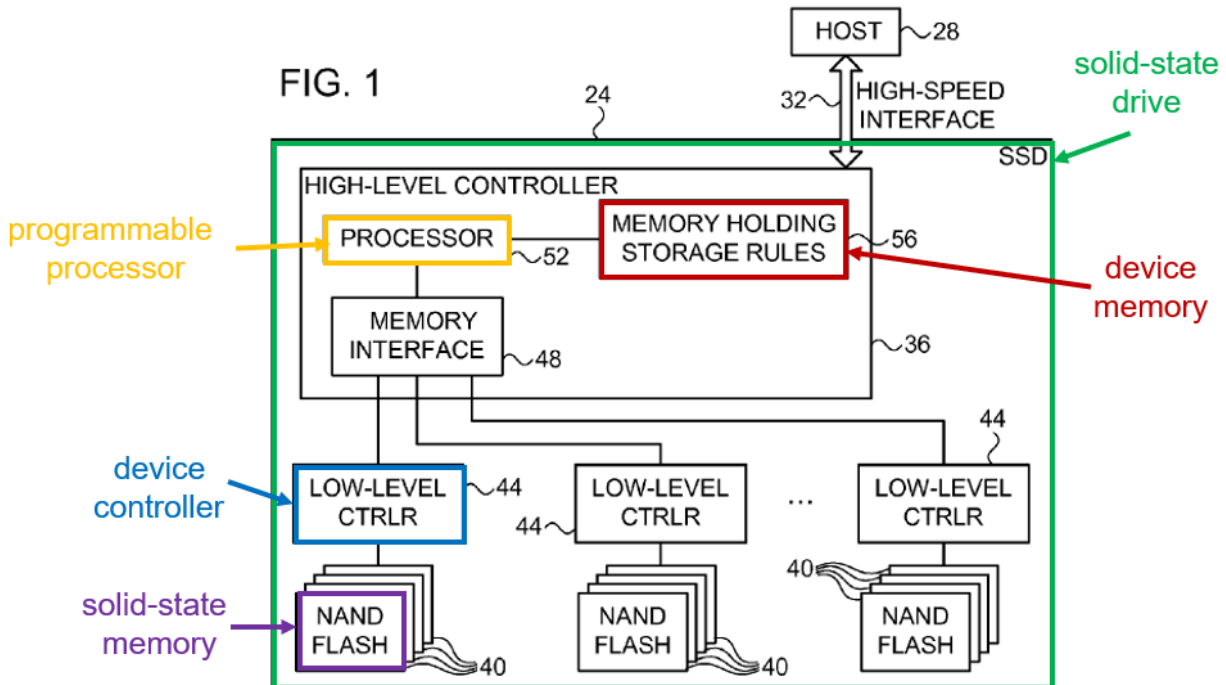
that the software carries out the function of the processor (*id.*, [0034]), it follows that the software (*application*) of processor 52 directs the processor 52 (*programmable processor*) to transfer the data (*transfers the content*) to the low-level controller 44 (*device controller*). As Meir explains, the transferring of the data from high-level controller to the low-level controller is supported by the connection between the two: “[h]igh-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example *via low-level controllers 44*).” *Id.*, [0033] (emphasis added), FIG. 1.

115. Additionally, the pipeline rules break down “memory access commands” into “parts for execution by different pipeline stages” (i.e., the high-level controller, the low-level controller, and the memory units). *Id.*, [0044], [0045] (“In an embodiment, each read command is broken into three pipeline stages—fetching the data (e.g., by the processing circuitry in the memory units), decoding the ECC (e.g., by the low-level controllers) and sending the data to the host (e.g., by the high-level controller).”). A person of ordinary skill in the art would have understood that that the pipeline rules break down and assign write commands as well because memory access commands include write commands. *Id.*, [0030] (“memory access commands (e.g., read and write commands)”). Such a skilled person would have understood that the delegation of tasks among the high-level

controller and the low-level controller implies that, for write commands, the data is transferred from the high-level controller to the low-level controller.

9. Limitation 1[f]: “the device controller to:”

116. In my opinion, Meir discloses 1[f]: “the device controller to.” As explained for 1[d], the solid-state drive 24 includes low-level controller 44, which is a *device controller*. See Section VIII.C.5; Ex. 1010, FIG. 1. As discussed below, the low-level controller 44 performs the steps recited in 1[f.i]-1[f.ii].



Ex. 1010, Fig. 1 (annotations in color)

10. Limitation 1[f.i]: “receive the content from the programmable processor; and”

117. In my opinion, Meir discloses 1[f.i]: “receive the content from the programmable processor.” As I explained for 1[e.ii], in view of the data path

between the host and storage described by Meir, a person of ordinary skill in the art would have recognized that the software (*application*) of processor 52 (*programmable processor*) necessarily sends the data to the assigned low-level controller 44 (*device controller*). See Section VIII.C.8; Ex. 1010, FIG. 1. In doing so, a person of such skill would have also understood that the low-level controller 44 (*device controller*) necessarily receives the data for storage (*receive the content*) from processor 52 (*programmable processor*). *Id.*, FIG. 1.

11. Limitation 1[f.ii]: “store the content on the solid-state memory.”

118. In my opinion, Meir discloses 1[f.ii]: “store the content on the solid-state memory.” Meir explains that the “memory controller . . . executes the commands in the memory units.” Ex. 1010, [0020]; *see also id.*, [0032] (“the high-level controller, the low-level controllers and the processing circuitry in the memory units are referred to collectively as a memory controller”). In Meir’s embodiment where the memory controller includes high-level controller 36 and low-level controller 44 (*device controller*), the high-level controller 36 communicates with memory units 40 (*solid-state memory*) through its connection with low-level controller 44 (*device controller*). *Id.*, [0033] (“High-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example via low-level controllers 44), and a processor 52 that carries out the various functions of the high-level controller.”). As such, the low-level controller 44 (*device*

controller) implements the execution of a write command by writing the data (*stores the content*) to the memory unit 40 (*on the solid-state memory*) according to the logical-to-physical address mapping defined by the parallelism rule that is part of the policy. *Id.*, [0040] (“According to the policy, the memory controller translates the logical addresses to respective physical storage locations in memory units 40, and executes the commands in the memory units according to the logical-to-physical address mapping.”).

119. According to Meir, there are advantages to partitioning the execution of a command between the high-level controller 36 (as executed by processor 52 or *programmable processor*) and low-level controller 44. For example, the processor 52 (*programmable processor*) in high-level controller 36 can distribute the execution of various commands among various memory units 40 (*solid-state memory*) via the low-level controller 44 (*device controller*). Ex. 1010, [0037] (“Parallelism rules define how the memory controller distributes the memory access commands among the memory units. Pipeline rules define properties of a pipeline in the memory controller (not shown in the figure) that executes the commands.”).

120. Meir explains that “[b]eing aware of the parallelism rule, the host is able to assign LBAs to memory access commands in a way that increases the parallelism among memory units 40 and/or low-level controllers 44.” *Id.*, [0042]. For example, while one low-level controller 44 is storing data, processor 52 of the

high-level controller 36 can execute another command, sending data to another low-level controller 44 to store. Meir goes on to explain that “LBA assignment or ordering of this sort” in accordance with the storage policy “may avoid temporary congestion in a certain memory unit or low-level controller, and thus reduce latency.” Ex. 1010, [0042]. Indeed, a person of ordinary skill in the art would have known that parallelism in storage systems provide higher throughput, scalability, and more efficient resource utilization by distributing data operations across multiple low-level controllers and memory units that can operate concurrently. Modern storage systems including NAND flash-based SSDs contain multiple levels of parallel resources, such as channels, packages, and dies. A storage controller can exploit these resources by dividing data into portions and scheduling read and write operations across multiple independent components, thereby allowing multiple operations to proceed simultaneously rather than requiring each operation to be performed sequentially. This parallel execution reduces bottlenecks associated with latency of individual storage devices and increases the overall bandwidth that is available to applications. Therefore, a person of ordinary skill in the art would have used the implementation of Meir from Figure 1, because Meir discusses those very advantages. Ex. 1010, Fig. 1.

D. Dependent Claims 2, 5-8, 10

1. Dependent Claim 2

a. Preamble 2[pre]: “The solid-state drive of claim 1 wherein:”

121. In my opinion, Meir discloses 2[pre]: “The solid-state drive of claim 1”. As I discussed, Meir describes SSD 24 (*solid-state drive*) as recited in claim 1. *See* Section VIII.C.1.

b. Limitation 2[a]: “the device controller is to:”

122. In my opinion, Meir discloses 2[a]: “the device controller is to.” As I discussed for 1[d], Meir describes low-level controller 44 (*device controller*) “that executes the memory access commands issued by host 28.” *See* Section VIII.C.5.

c. Limitation 2[a.i]: “read requested content from the solid-state memory;”

123. In my opinion, Meir discloses 2[a.i]: “read requested content from the solid-state memory.”

124. Meir explains that the “memory controller . . . executes the commands in the memory units.” Ex. 1010, [0020]; *see also id.*, [0032] (“the high-level controller, the low-level controllers and the processing circuitry in the memory units are referred to collectively as a memory controller”). In Meir’s embodiment where the memory controller includes high-level controller 36 and low-level controller 44 (*device controller*), the high-level controller 36 communicates with memory units 40 (*solid-state memory*) via low-level controller 44 (*device controller*). *Id.*, [0033]

(“High-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example via low-level controllers 44), and a processor 52 that carries out the various functions of the high-level controller.”). As such, as part of executing a read command, the high-level controller 36 “translates the logical addresses to respective physical storage locations in memory units 40, and executes the commands in the memory units according to the logical-to-physical address mapping.” *Id.* [0040]. For example, “a certain subset of bits in the logical address may indicate the . . . low-level controller in which this address is assigned” and after translation, the high-level controller 36 will direct that assigned low-level controller 44 (*device controller*), to execute the access command, here the read command (*read requested content from the solid-state memory*). *Id.*, [0044].

125. Meir also explains that the SSD 24 may use error correction code (“ECC”) to encode the data (*content*) to prevent errors:

For example, a read command typically involves fetching ECC-encoded data from memory units 40, decoding the ECC and sending the decoded data to the host. In an embodiment, each read command is broken into three pipeline stages—fetching the data (e.g., by the processing circuitry in the memory units), decoding the ECC (e.g., by the low-level controllers) and sending the data to the host (e.g., by the high-level controller).

Id., [0045]. Thus, Meir teaches that when a read command is received, the low-level controller 44 will use the processing circuitry in memory units 40 to fetch the ECC-

encoded data (*read requested content from the solid-state memory*) and the low-level controller 44 will then decode the data. *Id.*

d. Limitation 2[a.ii]: “transfer the requested content to the programmable processor; and”

126. In my opinion, Meir discloses 2[a.ii]: “transfer the requested content to the programmable processor.” Meir describes that once the low-level controller 44 (*device controller*) decodes the requested data (*requested content*) received from the memory units 40, the low-level controller 44 sends the decoded data (*transfer the requested content*) to the processor 52 (*programmable processor*) of high-level controller 36. *Id.*, [0045].

e. Limitation 2[b]: “the instructions further direct the programmable processor to:”

127. In my opinion, Meir in view of Salessi renders obvious 2[b]: “the instructions further direct the programmable processor to.” As I discussed in 1[e], Meir describes software comprising *instructions stored on* memory 56 (*device memory*), *that when executed by* processor 52 (*programmable processor*), *direct the programmable processor to* execute the access commands according to a policy stored on the memory 56. *See* Section VIII.C.6; Ex. 1010, FIG. 1, [0034] (“In some embodiments, the memory controller comprises one or more general-purpose processors, which are programmed in software to carry out the functions described

herein. The software may be . . . provided and/or stored on non-transitory tangible media, such as magnetic, optical, or electronic memory.”).

f. Limitation 2[b.i]: “receive the requested content from the device controller;”

128. In my opinion, Meir discloses 2[b.i]: “receive the requested content from the device controller.”

129. As I discussed above, Meir describes that the software comprising *instructions directs* processor 52 (*programmable processor*). See Section VIII.C.6. For example, Meir describes that the software (*instructions*) *directs* processor 52 (*programmable processor*) to *receive* the decoded data (*requested content*) from low-level controller 44 (*device controller*). Ex. 1010, [0044] (“the policy defines that the memory controller executes the memory access commands in a pipelined manner [where] the different pipeline stages may be executed by different controllers in the memory controller.”), [0045] (“a read command typically involves fetching ECC-encoded data from memory units 40, decoding the ECC and sending the decoded data to the host”). Meir describes that when a “read command is broken into three pipeline stages,” “decoding the ECC” can be performed by the low-level controller 44, which then sends the decoded data to the processor 52 (*programmable processor*) of the high-level controller 36. *Id.*, [0045]. Processor 52 (*programmable processor*) performs the step of “sending the data to the host” as directed by the software *instructions* as I explained above. *Id.*

g. Limitation 2[b.ii]: “process the requested content for transfer based on the storage policy; and”

130. In my opinion, Meir in view of Salessi renders obvious 2[b.ii]: “process the requested content for transfer based on the storage policy.” Meir teaches that “SSD 24 comprises a high-level controller 36, which stores data in one or more non-volatile memory units 40.” Ex. 1010, [0030]. This data is *requested content* because it was requested by the host through a read command. Meir explains that “[h]ost 28 typically sends memory access commands (e.g., read and write commands) to SSD 24 over interface 32” and “[t]he memory system comprises a memory controller which receives from the host memory access commands (e.g., read and write commands) and executes the commands in the memory units.” Ex. 1010, [0030], [0020]. Therefore, Meir describes a data access system comprising a memory controller that receives and executes a read command in order to retrieve the data requested in the read command.

131. Meir describes that the memory controller will *process* commands *based on a storage policy*. For example, Meir describes that “[t]he memory controller typically executes the memory access commands received from the host in accordance with a certain policy.” Ex. 1010, [0021], [0033] (“memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40”). As I discussed above, parallelization rules, pipeline rules, prioritization rules, redundancy rules, power consumption rules and/or memory unit

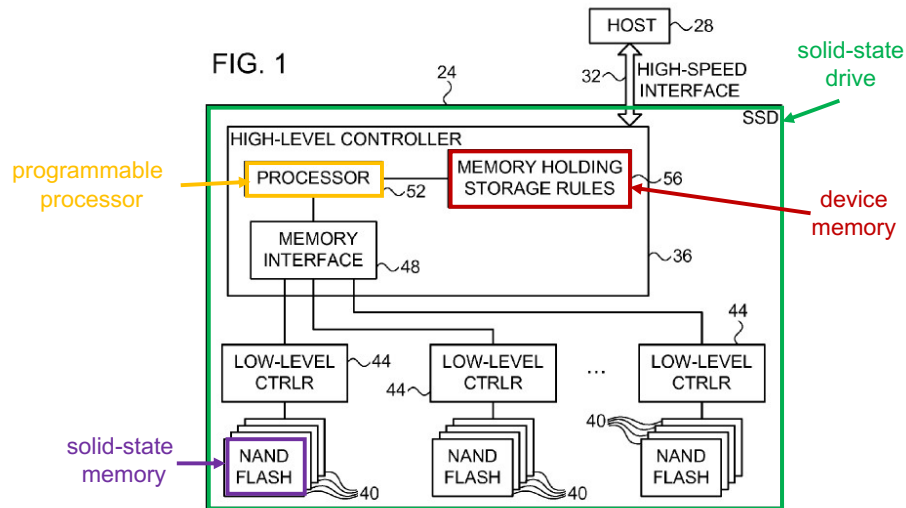
characteristics are storage policies. *Id.*, [0037] (“The policy comprises a set of rules, for example parallelization rules, pipeline rules, prioritization rules, redundancy rules, power consumption rules and/or memory unit characteristics rules.”); *see also* Section VIII.C.7. As an example of policy-based storage discussed by Meir, when applying the redundancy rules from the policy, the software directs the processor 52 to process data by implementing RAID. Ex. 1010, [0025] (“[t]he memory controller policy involves storing the data while applying a redundant storage scheme such as Redundant Array of Inexpensive Disks (RAID) among the memory units”). *See* my discussion above in Sections VIII.C.7 and IV.F. In addition, a person of ordinary skill in the art would have understood that a RAID implementation would have involved processing when reading data, such as recomputing data from redundancy information.

132. The software (*application*) can direct the processor 52 to *process* the received data (*requested content for transfer*) based on the redundant storage scheme as I described above. *Id.*, [0025]. For example, when the data has been striped, the processor 52 splits the read request and pulls different parts of the requested data from multiple drives. *Id.*, [0038]. Processor 52 (*programmable processor*) will reassemble the different parts to transfer to the host. Similarly, when a drive failure has been identified, a person of ordinary skill in the art would understand that processor 52 will use the parity data to rebuild the requested data.

133. To the extent that it is determined that Meir does not disclose that the software executed by processor 52 (*instructions*) *process* the data (*requested content*) included with the commands *based on a storage policy*, it is my opinion that this is rendered obvious by the system resulting from the combination of the teachings of Meir and Salessi, as I discuss further below.

134. Meir's SSD is similar to Salessi's SSD. For example, like Meir's SSD (which includes memory units 40 and a memory controller), Salessi teaches an SSD that "includes flash memory and an SSD processing unit." Ex. 1011, FIG. 7, [0006]; *see also* Ex. 1010, FIG. 1 (*see below*). Salessi teaches that its processing unit can *process requested content* because it is "capable of processing the data . . . being stored or retrieved from the flash memory." Ex. 1011, [0006]. Salessi gives examples of processing: "Processing capabilities incorporated into the SSD may include encryption and decryption [and] compression and decompression." *Id.* Salessi further teaches that the processing unit "process[es], using a processing algorithm, data received from the flash memory to form processed data." *Id.*, [0007]; *see also id.*, [0042] ("the SSD 130 includes an SSD processing unit 240 that is designed for data operations or analytics. . . . and other operations such as compression, decompression, encryption, and decryption"), [0016] ("According to an embodiment of the present invention there is provided a method of . . . retrieving data from, a solid state drive including a host interface and a flash memory, the

method including: . . . processing, by the solid state drive, the retrieved data, using a processing algorithm, to generate processed data.”).



Ex. 1010, Fig. 1 (annotations in color)

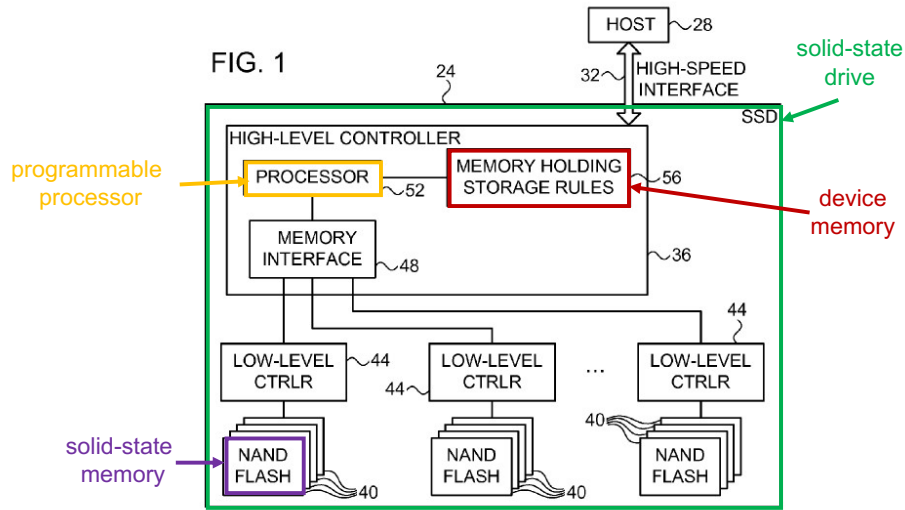
135. In the combination of Meir and Salessi, the policy stored in memory 56 as taught by Meir includes compression, decompression, decryption, and encryption algorithms and instructions as taught by Salessi. Specifically, Meir teaches that the “one or more of the rules in the memory controller policy may be configurable” and “the host may issue to the memory controller a command that configures or modifies the memory controller, so as to improve performance.” Ex. 1010, [0045]. In my opinion, a person of ordinary skill in the art would take advantage of this configurability to modify the memory controller policy of Meir to incorporate the

processing algorithms and instructions of Salessi, wherein Meir's policy would define which data to apply Salessi's processing algorithms to and where the data would be transferred and stored. Ex. 1011, [0006] ("Processing capabilities incorporated into the SSD may include encryption and decryption, compression and decompression, and in-line indexing of data."). Such a skilled person would have understood that incorporating compression, decompression, encryption, and decryption in storage device policies maximizes storage efficiency and secures sensitive data.

136. As I discussed, a person of ordinary skill in the art would have been motivated to modify the software (*application*) and *policy* stored in memory 56 (*device memory*) of Meir to direct the processor 52 (*programmable processor*) to decompress and decrypt the data read from storage according to algorithms and rules in the policy because it would optimize the performance and improve the security of the system. See Section VIII.A; Ex. 1010, [0036] ("execut[ing] the memory access commands received from host 28 in accordance with a predefined policy . . . optimize[s] performance"); Ex. 1011, [0050] ("functions such as . . . decryption of the data read from the flash memory 150 . . . , or similarly . . . decompression . . . at . . . read . . . increase[s] the effective storage capacity of the SSD 130"). Meir describes configurability of its policies, so a person of ordinary skill in the art would

know that those policies could be modified to incorporate the benefits of Salessi's algorithms to further bolster the performance and security of the system.

137. Further, as I discussed, such a person of ordinary skill would have had a reasonable expectation of success since the SSDs of Meir and Salessi have similar structures and components, and Salessi expressly teaches the use of its processing algorithms in such similar systems. *See* Section VIII.B; Ex. 1010, FIG. 1, [0033] (“High-level controller 36 comprises a memory interface 56. In particular, memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40. The policy comprises a set of rules, for example one or more parallelization rules, pipeline rules and/or prioritization rules.”); Ex. 1011, FIG. 7, [0006] (“In one embodiment, the SSD includes flash memory and an SSD processing unit capable of processing the data in the flash memory, or being stored or retrieved from the flash memory, without returning all of the data to be searched to a host CPU outside of the SSD.”).



Ex. 1010, Fig. 1 (annotations in color)

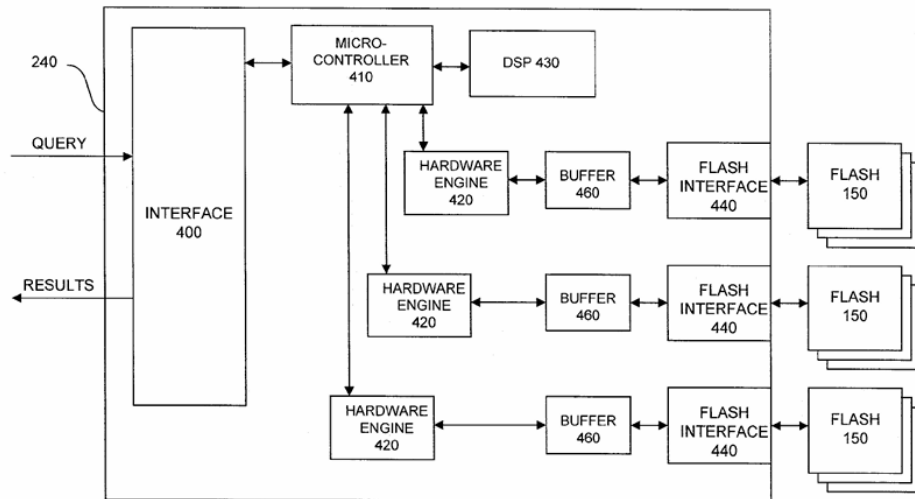


FIG. 7

Ex. 1011, Fig. 7

h. Limitation 2[b.iii]: “transfer the requested content.”

138. In my opinion, Meir discloses 2[b.iii]: “transfer the requested content.” Meir describes the low-level controller 44 (*device controller*) will send the ECC-decoded data (*transfer the requested content*) to processor 52 of the high-level controller, which will then “send[] the data” (*transfer the requested content*) “to the host (e.g., by the high-level controller).” Ex. 1010, [0045].

2. Claim 5

a. Preamble 5[pre]: “The solid-state drive of claim 1 wherein:”

139. I have been asked to assume the preamble of claim 5 is limiting. Under that assumption, it is my opinion that Meir discloses the preamble of claim 5, which recites “The solid-state drive of claim 1 wherein.” As I discussed, Meir describes SSD 24 (*solid-state drive*) as recited in claim 1. See Section VIII.C.1.

b. Limitation 5[a]: “the instructions further direct the programmable processor to:”

140. In my opinion, Meir in view of Salessi renders obvious 5[a]: “the instructions further direct the programmable processor to.” As I discussed in 1[e], Meir describes software comprising *instructions stored on* memory 56 (*device memory*), *that when executed by* processor 52 (*programmable processor*), *direct the programmable processor to* execute the access commands according to a policy stored on the memory 56. See Section VIII.C.6; Ex. 1010, FIG. 1, [0034] (“In some

embodiments, the memory controller comprises one or more general-purpose processors, which are programmed in software to carry out the functions described herein.”).

c. Limitation 5[a.i]: “compress the content based on the storage policy;”

141. In my opinion, Meir in view of Salessi renders obvious 5[a.i]: “compress the content based on the storage policy.” As I discussed, Meir discloses, and Meir in view of Salessi renders obvious, that the memory controller will *process* commands *based on a storage policy*. See Section VIII.C.7. Although Meir does not explicitly disclose that the software executed by processor 52 (*instructions*) *compress* the data (*requested content*) included with the commands *based on a storage policy*, in my opinion this is rendered obvious by the system combining the teachings of Meir and Salessi.

142. Salessi teaches that its processing unit can *compress content for storage* because Salessi’s SSD processing unit is “capable of processing data” and “[p]rocessing capabilities incorporated into the SSD may include encryption and decryption [and] compression and decompression.” Ex. 1011, [0006], [0042] (“the SSD 130 includes an SSD processing unit 240 that is designed for data operations or analytics. . . . and other operations such as compression, decompression, encryption, and decryption”), [0023] (“Salessi further teaches that the processing

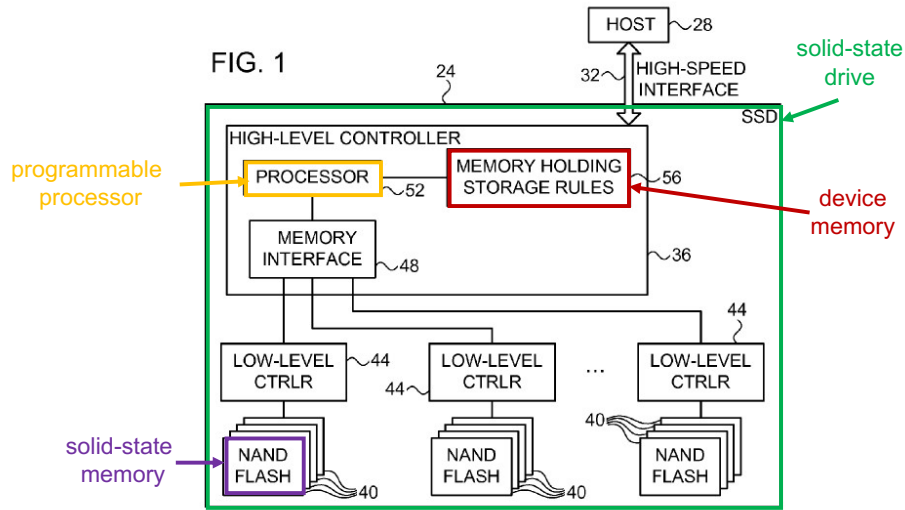
unit “us[es] a processing algorithm[] to generate processed data.”); *see* Section VIII.C.7.

143. In the combination of Meir and Salessi, the policy stored in memory 56, as taught by Meir, includes compression, as taught by Salessi. A person of ordinary skill in the art would be motivated to combine Meir and Salessi in this way. Namely, Meir teaches that the “one or more of the rules in the memory controller policy may be configurable” and “the host may issue to the memory controller a command that configures or modifies the memory controller, so as to improve performance.” Ex. 1010, [0045]. In my opinion, a such a skilled person would rely on this configurability to modify the memory controller policy of Meir to incorporate the compression algorithms and instructions of Salessi and define which data to apply Salessi’s compression algorithms to. Ex. 1011, [0006].

144. As I discussed, a person of ordinary skill in the art would have been motivated to modify the software (*application*) and policy store in memory 56 (*policy*) of Meir to direct the processor 52 to compress the data to be stored according to algorithms and rules in the policy because it would optimize the performance and improve the security of the system. *See* Section VIII.A; Ex. 1010, [0036] (“execut[ing] the memory access commands received from host 28 in accordance with a predefined policy . . . optimize[s] performance”); Ex. 1011, [0050] (“functions such as encryption of the data written to the flash memory 150 . . . , or

similarly for compression . . . at write . . . increase[s] the effective storage capacity of the SSD 130”).

145. Further, as I discussed, a person of ordinary skill in the art would have had a reasonable expectation of success since the SSDs of Meir and Salessi have similar structures and components, and Salessi expressly teaches that the use of its processing algorithms in such similar systems. *See* Section VIII.B; Ex. 1010, FIG. 1, [0033] (“High-level controller 36 comprises a memory interface 56. In particular, memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40. The policy comprises a set of rules, for example one or more parallelization rules, pipeline rules and/or prioritization rules.”); Ex. 1011, FIG. 7, [0006] (“In one embodiment, the SSD includes flash memory and an SSD processing unit capable of processing the data in the flash memory, or being stored or retrieved from the flash memory, without returning all of the data to be searched to a host CPU outside of the SSD.”).



Ex. 1010, Fig. 1 (annotations in color)

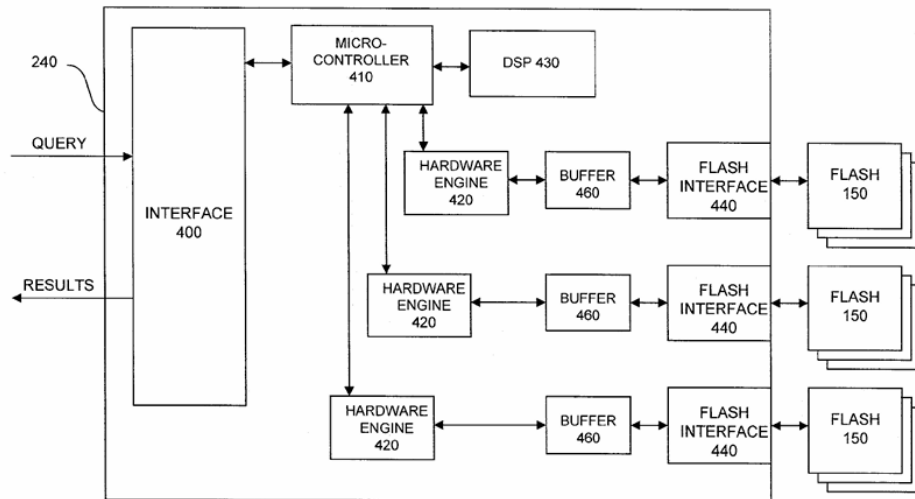


FIG. 7

Ex. 1011, Fig. 7

d. Limitation 5[a.ii]: “transfer the compressed content to the device controller; and”

146. In my opinion, Meir in view of Salessi renders obvious 5[a.ii]: “transfer the compressed content to the device controller.” Meir discloses *transferring the content to the device controller*. See Section VIII.C.8. Salessi discloses that the *content is compressed*. See Section VIII.D.2.c. In my opinion, a person of ordinary skill in the art would have been motivated to combine Meir with Salessi for the reasons I discussed above. See Sections VIII.A-B, VIII.D.2.c.

e. Limitation 5[b]: “the device controller is to:”

147. In my opinion, Meir discloses 5[b]: “the device controller is to.” As I discussed for 1[d], Meir describes low-level controller 44 (*device controller*) “that executes the memory access commands issued by host 28.” See Section VIII.C.5.

f. Limitation 5[b.i]: “receive the compressed content from the programmable processor; and”

148. In my opinion, Meir in view of Salessi renders obvious 5[b.i]: “receive the compressed content from the programmable processor.” Meir discloses *receiving the content from the programmable processor*. See Section VIII.C.10. Salessi discloses that the *content is compressed*. See Section VIII.D.2.c. In my opinion, a person of ordinary skill in the art would have been motivated to combine Meir with Salessi for the reasons I discussed above. See Sections VIII.A-B, VIII.D.2.c.

g. Limitation 5[b.ii]: “store the compressed content on the solid-state memory.”

149. In my opinion, Meir in view of Salessi renders obvious 5[b.ii]: “store the compressed content on the solid-state memory.” Meir discloses *storing the content on the solid-state memory*. See Section VIII.C.11. Salessi discloses that the *content is compressed*. See Section VIII.D.2.c. Further, Salessi expressly teaches that the processed data is stored: “processing, by the processing unit, the data, using a processing algorithm, to generate processed data; and storing, by the processing unit, the processed data in the flash memory.” Ex. 1011, [0023]. In my opinion, a person of ordinary skill in the art would have been motivated to combine Meir with Salessi for the reasons I discussed above. See Sections VIII.A-B, VIII.D.2.c.

3. Claim 6

a. Preamble 6[pre]: “The solid-state drive of claim 5 wherein:”

150. I have been asked to assume the preamble of claim 6 is limiting. Under that assumption, it is my opinion that Meir discloses the preamble of claim 6, which recites “The solid-state drive of claim 5 wherein.”. As I discussed above, Meir describes SSD 24 (*solid-state drive*) as recited in claim 1[pre]. See Section VIII.C.1.

b. Limitation 6[a]: “The device controller is to:”

151. In my opinion, Meir discloses 6[a]: “The device controller is to.” As I discussed for 1[d], Meir describes low-level controller 44 (*device controller*) “that executes the memory access commands issued by host 28.” *See* Section VIII.C.5.

c. Limitation 6[a.i]: “read the compressed content from the solid-state memory; and”

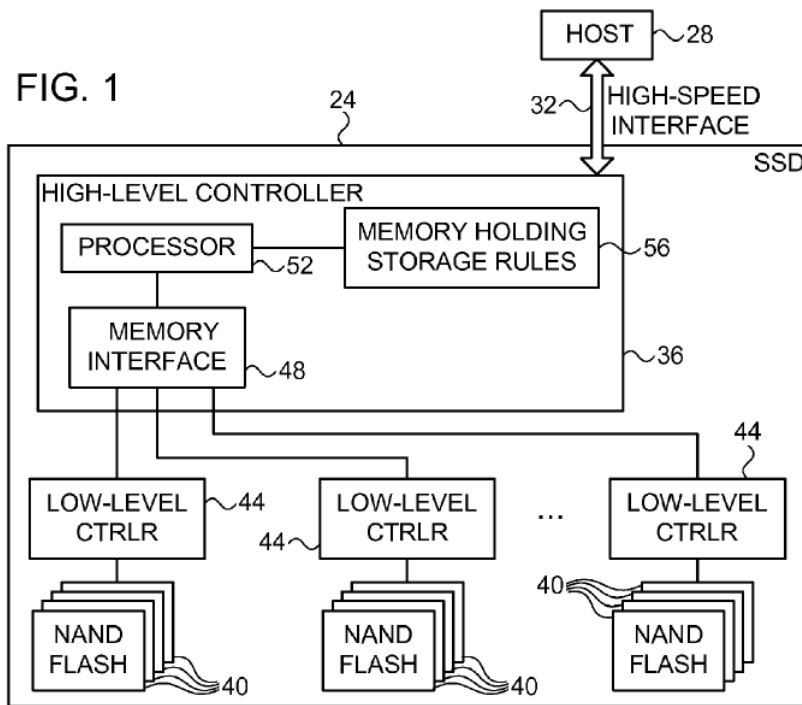
152. In my opinion, Meir in view of Salessi renders obvious 6[a.i]: “read the compressed content from the solid-state memory.” Meir discloses *reading the content from the solid-state memory*. *See* Section VIII.D.1.I.D.1.cc. Salessi discloses that the *content* is *compressed*. *See* Section VIII.D.2.c. In my opinion, a person of ordinary skill in the art would have been motivated to combine Meir with Salessi for the reasons I discussed above. *See* Sections VIII.A-B, VIII.D.2.c.

d. Limitation 6[a.ii]: “transfer the compressed content to the programmable processor; and”

153. In my opinion, Meir in view of Salessi renders obvious 6[a.ii]: “transfer the compressed content to the programmable processor.” Meir discloses *transferring the content to the programmable processor*. *See* Section VIII.D.1.d. Salessi discloses that the *content* is *compressed*. *See* Section VIII.D.2.c. In my opinion, a person of ordinary skill in the art would have been motivated to combine Meir with Salessi for the reasons I discussed above. *See* Sections VIII.A-B, VIII.D.2.c.

e. Limitation 6[b]: “the instructions further direct the programmable processor to:”

154. In my opinion, Meir discloses 6[b]: “the instructions further direct the programmable processor to.” As I discussed in 1[e] above, Meir describes software comprising *instructions stored on memory 56 (device memory), that when executed by processor 52 (programmable processor), direct the programmable processor to execute the access commands according to a policy stored on the memory 56. See Section VIII.C.6; Ex. 1010, FIG. 1, [0034]* (“In some embodiments, the memory controller comprises one or more general-purpose processors, which are programmed in software to carry out the functions described herein. The software may be . . . provided and/or stored on non-transitory tangible media, such as magnetic, optical, or electronic memory.”).



Ex. 1010, Fig. 1

f. Limitation 6[b.i]: “receive the compressed content from the device controller;”

155. In my opinion, Meir in view of Salessi renders obvious 6[b.i]: “receive the compressed content from the device controller.” Meir discloses *receiving the content from the device controller*. See Section VIII.D.1.f. Salessi discloses that the *content is compressed*. See Section VIII.D.2.c. In my opinion, a person of ordinary skill in the art would have been motivated to combine Meir with Salessi for the reasons I discussed above. See Sections VIII.A-B, VIII.D.2.c.

g. Limitation 6[b.ii]: “decompress the content based on the storage policy; and”

156. In my opinion, Meir in view of Salessi renders obvious 6[b.ii]: “decompress the content based on the storage policy.” As I discussed, Meir discloses, and Meir in view of Salessi renders obvious, that the memory controller processes commands *based on a storage policy*. See Section VIII.C.7. Although Meir does not explicitly disclose that the software executed by processor 52 (*instructions*) *decompress* the data (*requested content*) included with the commands *based on a storage policy*, in my opinion this is rendered obvious by the system combining the teachings of Meir and Salessi.

157. Salessi teaches that its processing unit can *decompress content* because Salessi’s SSD processing unit is “capable of processing data” and “[p]rocessing capabilities incorporated into the SSD may include encryption and decryptions [and] compression and decompression.” Ex. 1011, [0006], [0042] (“the SSD 130 includes an SSD processing unit 240 that is designed for data operations or analytics. . . . and other operations such as compression, decompression, encryption, and decryption”), [0023] (“Salessi further teaches that the processing unit “us[es] a processing algorithm[] to generate processed data.”); see Section VIII.C.7.

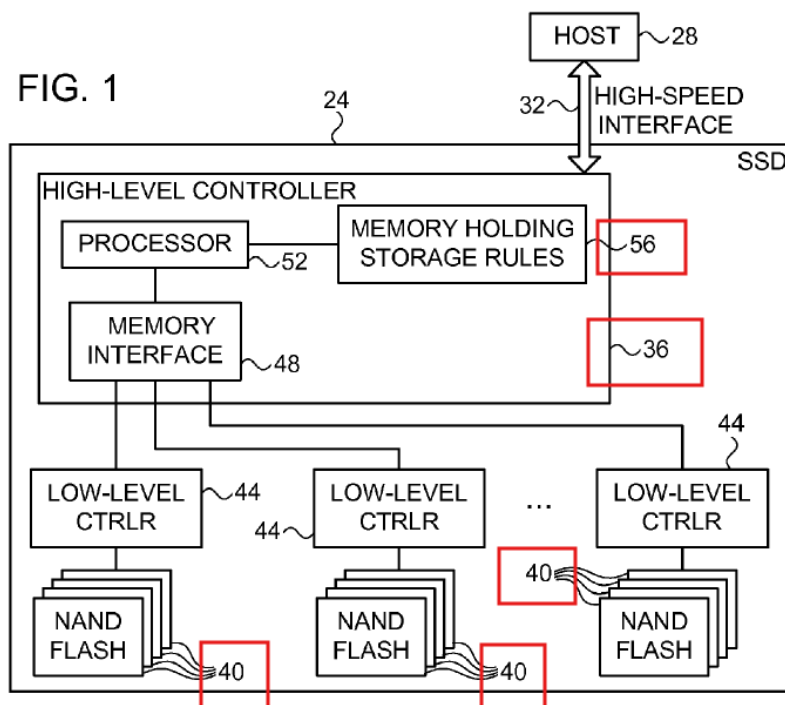
158. In the combination, the policy stored in memory 56 as taught by Meir includes decompression as taught by Salessi. A person of ordinary skill in the art would be motivated to combine Meir and Salessi in this way. Namely, Meir teaches

that the “one or more of the rules in the memory controller policy may be configurable” and “the host may issue to the memory controller a command that configures or modifies the memory controller, so as to improve performance.” Ex. 1010, [0045]. In my opinion, such a skilled person would rely on this configurability to modify the memory controller policy of Meir to incorporate the decompression algorithms and instructions of Salessi and define which data to apply Salessi’s decompression algorithms to. Ex. 1011, [0006].

159. As I discussed, a person of ordinary skill in the art would have been motivated to modify the software (*application*) and policy store in memory 56 (*policy*) of Meir to direct the processor 52 to decompress the data to be stored according to algorithms and rules in the policy because it would optimize the performance and improve the security of the system. *See* Section VIII.A; Ex. 1010, [0036] (“execut[ing] the memory access commands received from host 28 in accordance with a predefined policy . . . optimize[s] performance”); Ex. 1011, [0050] (“functions such as . . . decryption of the data read from the flash memory 150 . . . , or similarly . . . decompression . . . at . . . read . . . increase[s] the effective storage capacity of the SSD 130”).

160. Further, as I discussed, a person of ordinary skill in the art would have had a reasonable expectation of success since the SSDs of Meir and Salessi have similar structures and components, and Salessi expressly teaches that the use of its

processing algorithms in such similar systems. *See* Section VIII.B; Ex. 1010, FIG. 1, [0033] (“High-level controller 36 comprises a memory interface 56. In particular, memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40. The policy comprises a set of rules, for example one or more parallelization rules, pipeline rules and/or prioritization rules.”); Ex. 1011, FIG. 7, [0006] (“In one embodiment, the SSD includes flash memory and an SSD processing unit capable of processing the data in the flash memory, or being stored or retrieved from the flash memory, without returning all of the data to be searched to a host CPU outside of the SSD.”).



Ex. 1010, Fig. 1 (annotations in color)

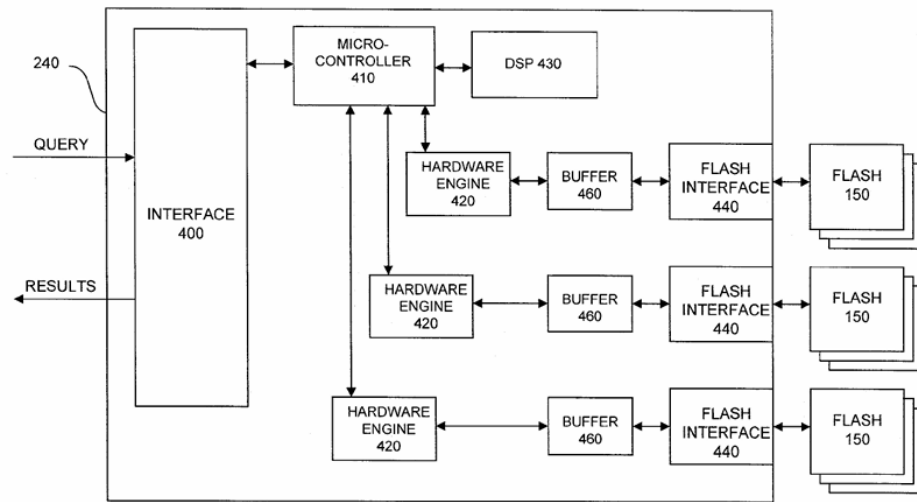


FIG. 7

Ex. 1011, Fig. 7

h. Limitation 6[b.iii]: “transfer the decompressed content.”

161. In my opinion, Meir in view of Salessi renders obvious 6[b.iii]: “transfer the decompressed content.” Meir discloses *transferring the content*. See Section VIII.D.1.h. Salessi discloses that the *content* is *decompressed*. See Section VIII.D.3.g. In my opinion, a person of ordinary skill in the art would have been motivated to combine Meir with Salessi for the reasons I discussed above. See Sections VIII.A-B, VIII.D.3.g.

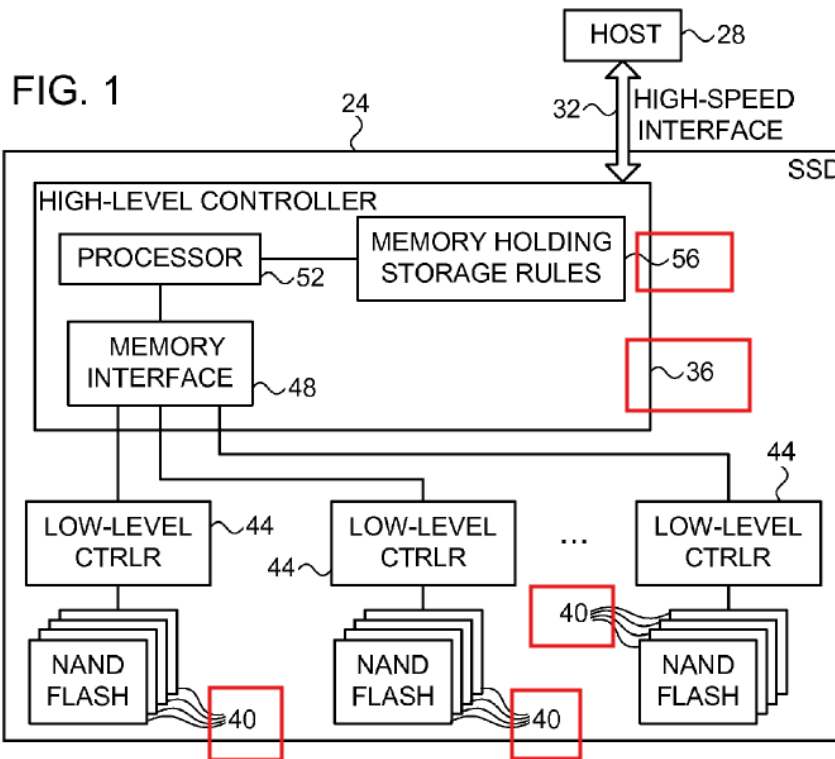
4. Claim 7

a. Preamble 7[pre]: “The solid-state drive of claim 1 wherein:”

162. I have been asked to assume the preamble of claim 7 is limiting. Under that assumption, it is my opinion that Meir discloses the preamble of claim 7, which recites “The solid-state drive of claim 1 wherein.” As I discussed above, Meir describes SSD 24 (*solid-state drive*) as recited in claim 1. *See* Section VIII.C.1.

b. Limitation 7[a]: “the instructions further direct the programmable processor to:”

163. In my opinion, Meir in view of Salessi renders obvious 7[a]: “the instructions further direct the programmable processor to.” As I discussed above in 1[e], Meir describes software comprising *instructions stored on* memory 56 (*device memory*), *that when executed by* processor 52 (*programmable processor*), *direct the programmable processor to* execute the access commands according to a policy stored on the memory 56. *See* Section VIII.C.6; Ex. 1010, FIG. 1, [0034] (“In some embodiments, the memory controller comprises one or more general-purpose processors, which are programmed in software to carry out the functions described herein. The software may be . . . provided and/or stored on non-transitory tangible media, such as magnetic, optical, or electronic memory.”).



Ex. 1010, Fig. 1 (annotations in color)

c. Limitation 7[a.i]: “encrypt the content based on the storage policy;”

164. In my opinion, Meir in view of Salessi renders obvious 7[a.i]: “encrypt the content based on the storage policy.” As I discussed, Meir discloses, and Meir in view of Salessi renders obvious, that the memory controller will *process* commands *based on a storage policy*. See Section VIII.C.7. Although Meir does not explicitly disclose that the software executed by processor 52 (*instructions*) *encrypt* the data (*requested content*) included with the commands *based on a storage*

policy, it is my opinion that this is rendered obvious by the system combining the teachings of Meir and Salessi.

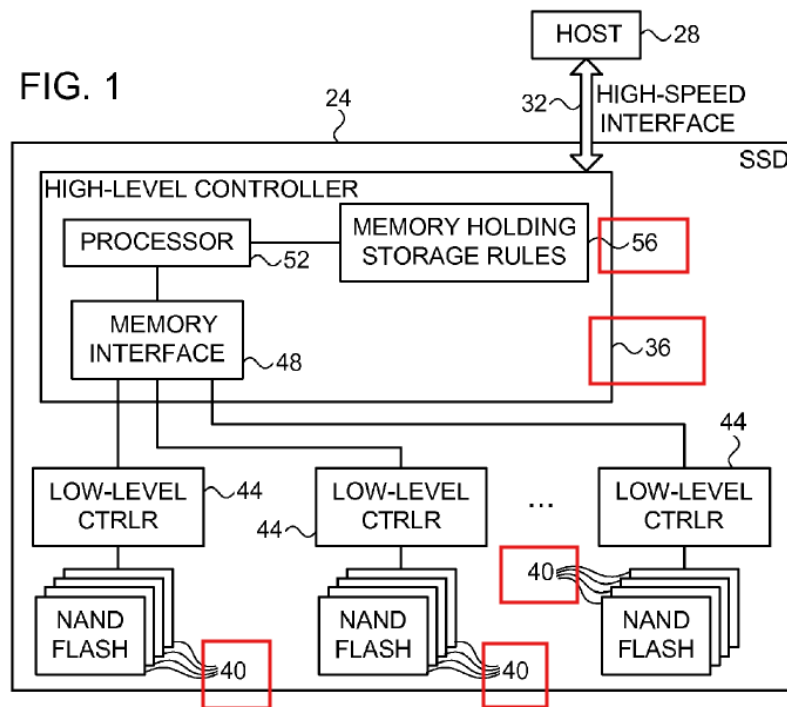
165. Salessi teaches that its processing unit can *encrypt content for storage* because Salessi's SSD processing unit is "capable of processing data" and "[p]rocessing capabilities incorporated into the SSD may include encryption and decryptions [and] compression and decompression." Ex. 1011, [0006], [0042] ("the SSD 130 includes an SSD processing unit 240 that is designed for data operations or analytics. . . . and other operations such as compression, decompression, encryption, and decryption"), [0023] ("Salessi further teaches that the processing unit "us[es] a processing algorithm[] to generate processed data."); *see* Section VIII.C.7.

166. In the combination, the policy stored in memory 56 as taught by Meir includes encryption as taught by Salessi. A person of ordinary skill in the art would be motivated to combine Meir and Salessi in this way. Namely, Meir teaches that the "one or more of the rules in the memory controller policy may be configurable" and "the host may issue to the memory controller a command that configures or modifies the memory controller, so as to improve performance." Ex. 1010, [0045]. In my opinion, such a skilled person would rely on this configurability to modify the memory controller policy of Meir to incorporate the encryption algorithms and instructions of Salessi and define which data to apply Salessi's encryption

algorithms to. Ex. 1011, [0006] (“Processing capabilities incorporated into the SSD may include encryption and decryption, compression and decompression, and in-line indexing of data.”).

167. As I discussed above, a person of ordinary skill in the art would have been motivated to modify the software (*application*) and policy store in memory 56 (*policy*) of Meir to direct the processor 52 to encrypt the data to be stored according to algorithms and rules in the policy because it would optimize the performance and improve the security of the system. *See* Section VIII.A; Ex. 1010, [0036] (“execut[ing] the memory access commands received from host 28 in accordance with a predefined policy . . . optimize[s] performance”); Ex. 1011, [0050] (“functions such as encryption of the data written to the flash memory 150 . . . , or similarly for compression . . . at write . . . increase[s] the effective storage capacity of the SSD 130”). Further, as I discussed, a person of ordinary skill in the art would have had a reasonable expectation of success since the SSDs of Meir and Salessi have similar structures and components, and Salessi expressly teaches that the use of its processing algorithms in such similar systems. *See* Section VIII.B; Ex. 1010, FIG. 1, [0033] (“High-level controller 36 comprises a memory interface 56. In particular, memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40. The policy comprises a set of rules, for example one or more parallelization rules, pipeline rules and/or prioritization

rules.”); Ex. 1011, FIG. 7, [0006] (“In one embodiment, the SSD includes flash memory and an SSD processing unit capable of processing the data in the flash memory, or being stored or retrieved from the flash memory, without returning all of the data to be searched to a host CPU outside of the SSD.”).



Ex. 1010, Fig. 1 (annotations in color)

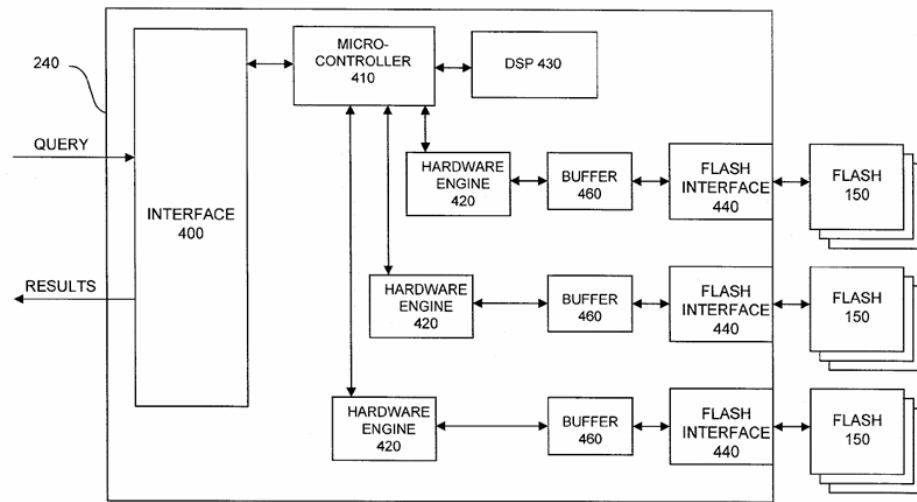


FIG. 7

Ex. 1011, Fig. 7

d. Limitation 7[a.ii]: “transfer the encrypted content to the device controller; and”

168. In my opinion, Meir in view of Salessi renders obvious 7[a.ii]: “transfer the encrypted content to the device controller.” As I discussed above, Meir discloses *transferring the content to the device controller*. See Section VIII.C.8. Salessi discloses that the *content is encrypted*. See Section VIII.D.4.c. In my opinion, a person of ordinary skill in the art would have been motivated to combine Meir with Salessi for the reasons I discussed above. See Sections VIII.A-B, VIII.D.4.c.

e. Limitation 7[b]: “the device controller is to:”

169. In my opinion, Meir discloses 7[b]: “the device controller is to.” As I discussed for 1[d], Meir describes low-level controller 44 (*device controller*) “that executes the memory access commands issued by host 28.” *See* Section VIII.C.5.

f. Limitation 7[b.i]: “receive the encrypted content from the programmable processor; and”

170. In my opinion, Meir in view of Salessi renders obvious 7[b.i]: “receive the encrypted content from the programmable processor.” As I discussed above, Meir discloses *receiving the content from the programmable processor*. *See* Section VIII.C.10. Salessi discloses that the *content* is *encrypted*. *See* Section VIII.D.4.c. In my opinion, a person of ordinary skill in the art would have been motivated to combine Meir with Salessi for the reasons discussed above. *See* Sections VIII.A-B, VIII.D.4.c.

g. Limitation 7[b.ii]: “store the encrypted content on the solid-state memory.”

171. In my opinion, Meir in view of Salessi renders obvious 7[b.ii]: “store the encrypted content on the solid-state memory.” As I discussed above, Meir discloses *storing the content on the solid-state memory*. *See* Section VIII.C.11. Salessi discloses that the *content* is *encrypted*. *See* Section VIII.D.4.c. In my opinion, a person of ordinary skill in the art would have been motivated to combine

Meir with Salessi for the reasons I discussed above. *See* Sections VIII.A-B, VIII.D.4.c.

5. Claim 8

a. Preamble 8[pre]: “The solid-state drive of claim 5 wherein:”

172. I have been asked to assume the preamble of claim 8 is limiting. Under that assumption, it is my opinion that Meir discloses the preamble of claim 8, which recites “The solid-state drive of claim 5 wherein.”. As I discussed above, Meir describes SSD 24 (*solid-state drive*) as recited in claim 1. *See* Section VIII.C.1.

b. Limitation 8[a]: “The device controller is to:”

173. In my opinion, Meir discloses 8[a]: “The device controller is to.” As I discussed for 1[d] above, Meir describes low-level controller 44 (*device controller*) “that executes the memory access commands issued by host 28.” *See* Section VIII.C.5.

c. Limitation 8[a.i]: “read the encrypted content from the solid-state memory; and”

174. In my opinion, Meir in view of Salessi renders obvious 8[a.i]: “read the encrypted content from the solid-state memory.” As I discussed above, Meir discloses *reading the content from the solid-state memory*. *See* Section VIII.D.1.c. Salessi discloses that the *content* is *encrypted*. *See* Section VIII.D.4.4.c. In my opinion, a person of ordinary skill in the art would have been motivated to combine

Meir with Salessi for the reasons discussed above. *See* Sections VIII.A-B, VIII.D.4.4.c.

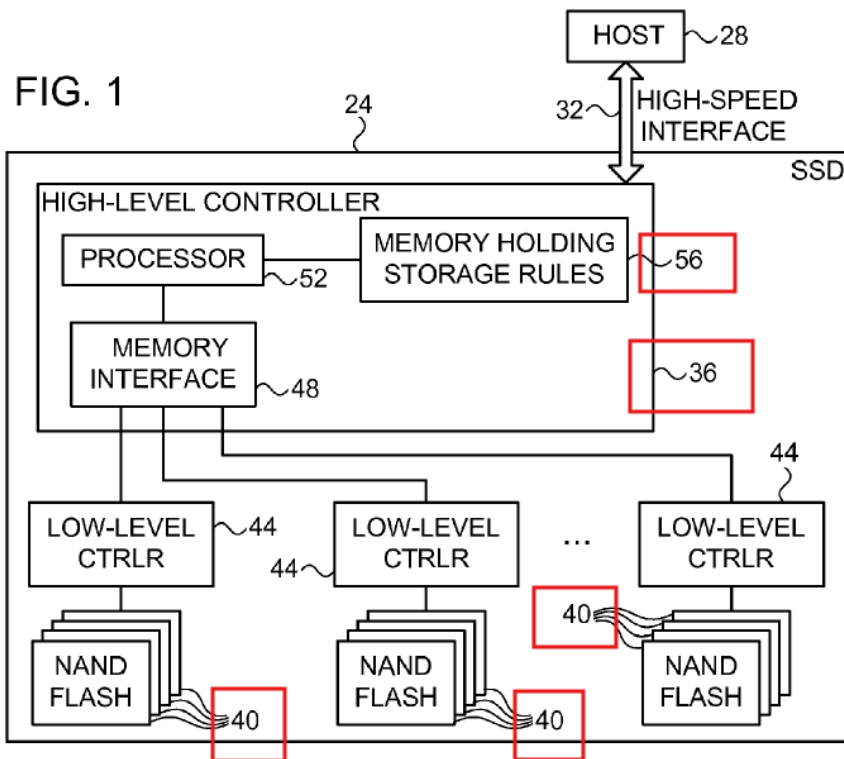
d. Limitation 8[a.ii]: “transfer the encrypted content to the programmable processor; and”

175. In my opinion, Meir in view of Salessi renders obvious 8[a.ii]: “transfer the encrypted content to the programmable processor.” As I discussed above, Meir discloses *transferring the content to the programmable processor*. *See* Section VIII.D.1.d. Salessi discloses that the *content* is *encrypted*. *See* Section VIII.D.4.4.c. In my opinion, a person of ordinary skill in the art would have been motivated to combine Meir with Salessi for the reasons I discussed above. *See* Sections VIII.A-B, VIII.D.4.4.c.

e. Limitation 8[b]: “the instructions further direct the programmable processor to:”

176. In my opinion, Meir discloses 8[b]: “the instructions further direct the programmable processor to.” As I discussed in 1[e], Meir describes software comprising *instructions stored on memory 56 (device memory), that when executed by processor 52 (programmable processor), direct the programmable processor to execute the access commands according to a policy stored on the memory 56*. *See* Section VIII.C.6; Ex. 1010, FIG. 1, [0034] (“In some embodiments, the memory controller comprises one or more general-purpose processors, which are programmed in software to carry out the functions described herein. The software

may be . . . provided and/or stored on non-transitory tangible media, such as magnetic, optical, or electronic memory.”).



Ex. 1010, Fig. 1 (annotations in color).

f. Limitation 8[b.i]: “receive the encrypted content from the device controller;”

177. In my opinion, Meir in view of Salessi renders obvious 8[b.i]: “receive the encrypted content from the device controller.” As I discussed above, Meir discloses *receiving the content from the device controller*. See Section VIII.D.1.f. Salessi discloses that the *content* is *encrypted*. See Section VIII.D.4.4.c. In my opinion, a person of ordinary skill in the art would have been motivated to combine

Meir with Salessi for the reasons I discussed above. *See* Sections VIII.A-B, VIII.D.4.4.c.

g. Limitation 8[b.ii]: “decrypt the content; and”

178. In my opinion, Meir in view of Salessi renders obvious this limitation. As I discussed, Meir discloses, and Meir in view of Salessi renders obvious, that the memory controller will *process* commands *based on a storage policy*. *See* Section VIII.C.7. Although Meir does not explicitly disclose that the software executed by processor 52 (*instructions*) *decrypt* the data (*requested content*) included with the commands based on a storage policy, in my opinion this is rendered obvious by the system combining the teachings of Meir and Salessi.

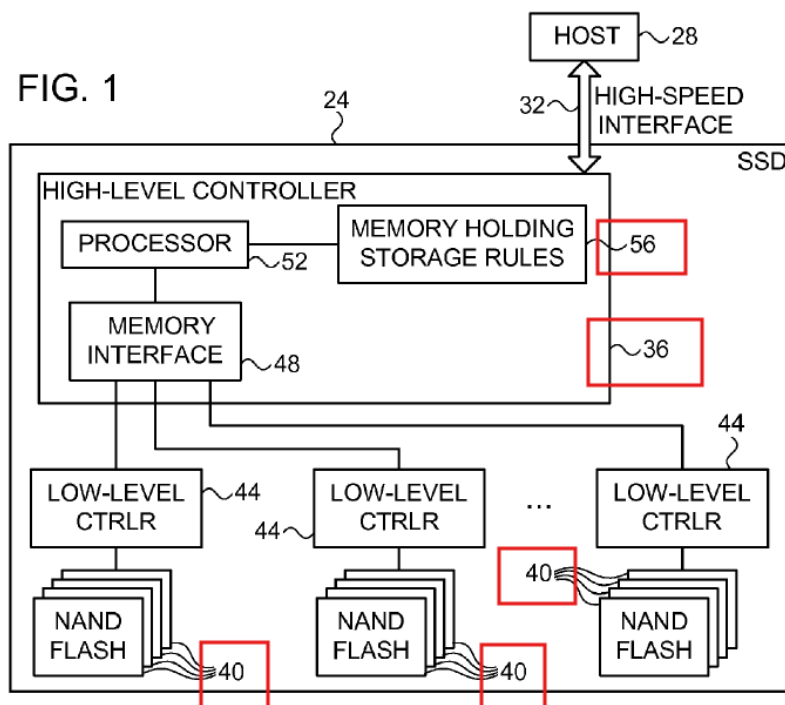
179. Salessi teaches that its processing unit can *decrypt content* because Salessi’s SSD processing unit is “capable of processing data” and “[p]rocessing capabilities incorporated into the SSD may include encryption and decryptions [and] compression and decompression.” Ex. 1011, [0006], [0042] (“the SSD 130 includes an SSD processing unit 240 that is designed for data operations or analytics. . . . and other operations such as compression, decompression, encryption, and decryption”), [0023] (“Salessi further teaches that the processing unit “us[es] a processing algorithm[] to generate processed data.”); *see* Section VIII.C.7.

180. In the combination, the policy stored in memory 56 as taught by Meir includes decryption as taught by Salessi. A person of ordinary skill in the art would

be motivated to combine Meir and Salessi in this way. Namely, Meir teaches that the “one or more of the rules in the memory controller policy may be configurable” and “the host may issue to the memory controller a command that configures or modifies the memory controller, so as to improve performance.” Ex. 1010, [0045]. In my opinion, such a skilled person would rely on this configurability to modify the memory controller policy of Meir to incorporate the decryption algorithms and instructions of Salessi and define which data to apply Salessi’s decryption algorithms to. Ex. 1011, [0006].

181. As I discussed, a person of ordinary skill in the art would have been motivated to modify the software (*application*) and policy store in memory 56 (*policy*) of Meir to direct the processor 52 to decrypt the data to be stored according to algorithms and rules in the policy because it would optimize the performance and improve the security of the system. *See* Section VIII.A; Ex. 1010, [0036] (“execut[ing] the memory access commands received from host 28 in accordance with a predefined policy . . . optimize[s] performance”); Ex. 1011, [0050] (“functions such as . . . decryption of the data read from the flash memory 150 . . . , or similarly . . . decompression . . . at . . . read . . . increase[s] the effective storage capacity of the SSD 130”). Further, as I discussed, a person of ordinary skill in the art would have had a reasonable expectation of success since the SSDs of Meir and Salessi have similar structures and components, and Salessi expressly teaches that

the use of its processing algorithms in such similar systems. *See* Section VIII.B; Ex. 1010, FIG. 1, [0033] (“High-level controller 36 comprises a memory interface 56. In particular, memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40. The policy comprises a set of rules, for example one or more parallelization rules, pipeline rules and/or prioritization rules.”); Ex. 1011, FIG. 7, [0006] (“In one embodiment, the SSD includes flash memory and an SSD processing unit capable of processing the data in the flash memory, or being stored or retrieved from the flash memory, without returning all of the data to be searched to a host CPU outside of the SSD.”).



Ex. 1010, Fig. 1 (annotations in color)

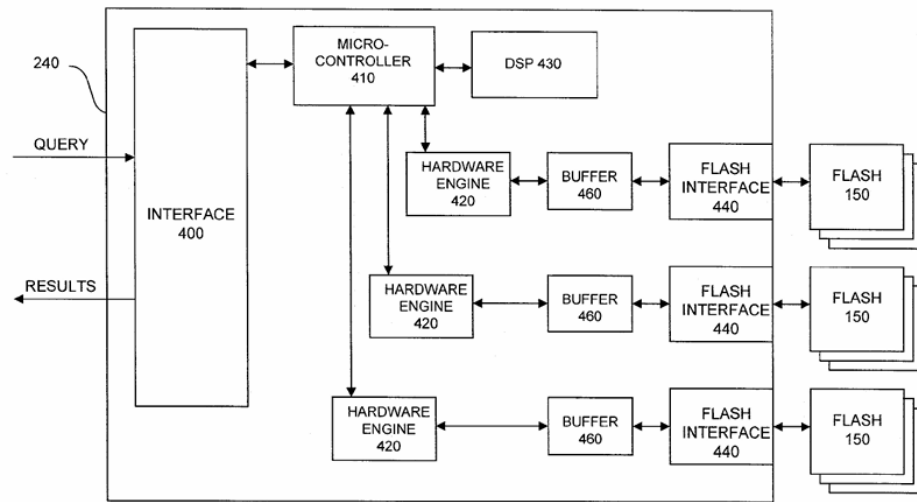


FIG. 7

Ex. 1011, Fig. 7

h. Limitation 8[b.iii]: “transfer the decrypted content.”

182. In my opinion, Meir in view of Salessi renders 8[b.iii]: “transfer the decrypted content.” As I discussed above, Meir discloses *transferring the content*. See Section VIII.D.1.h. Salessi discloses that the *content* is *decrypted*. See Section VIII.D.5.g. In my opinion, a person of ordinary skill in the art would have been motivated to combine Meir with Salessi for the reasons I discussed above. See Sections VIII.A-B, VIII.D.5.g.

6. Claim 10: “The solid-state drive of claim 1 wherein the content comprises one or more of video data, image data, on-line temporal video data, cached data, financial data, online transaction data, credit card data, or medical data.”

183. In my opinion, Meir in view of Salessi renders obvious 10: “The solid-state drive of claim 1 wherein the content comprises one or more of video data, image data, on-line temporal video data, cached data, financial data, online transaction data, credit card data, or medical data.”

184. Meir discloses data (*content*). See Section VIII.C.7. To the extent that Meir does not disclose one of the recited types of data, it is my opinion that this is rendered obvious by the system resulting from the combination of the teachings of Meir and Salessi.

185. Salessi explains that the data (*content*) is *video data, image data, on-line temporal video data, financial data, or online transaction data*. For example, Salessi teaches a data storage system for “data [that] comes from everywhere: posts to social media sites, digital pictures and video, purchase transaction records, bank transactions, sensors used to gather data and intelligence, like climate information, cell phone GPS signal, and many others.” Ex. 1011, [0002]. A person of ordinary skill in the art would have understood that posts to social media sites include video data and image data, digital pictures are image data, digital video is video data, purchase transaction records and bank transactions are financial data and/or online

transaction data. Such a skilled person would have understood that “many others” would include medical data as well. “This vast amount of data eventually is stored and maintained in storage nodes, such like . . . solid-state storage drives (SSD), and the like.”). *Id.*

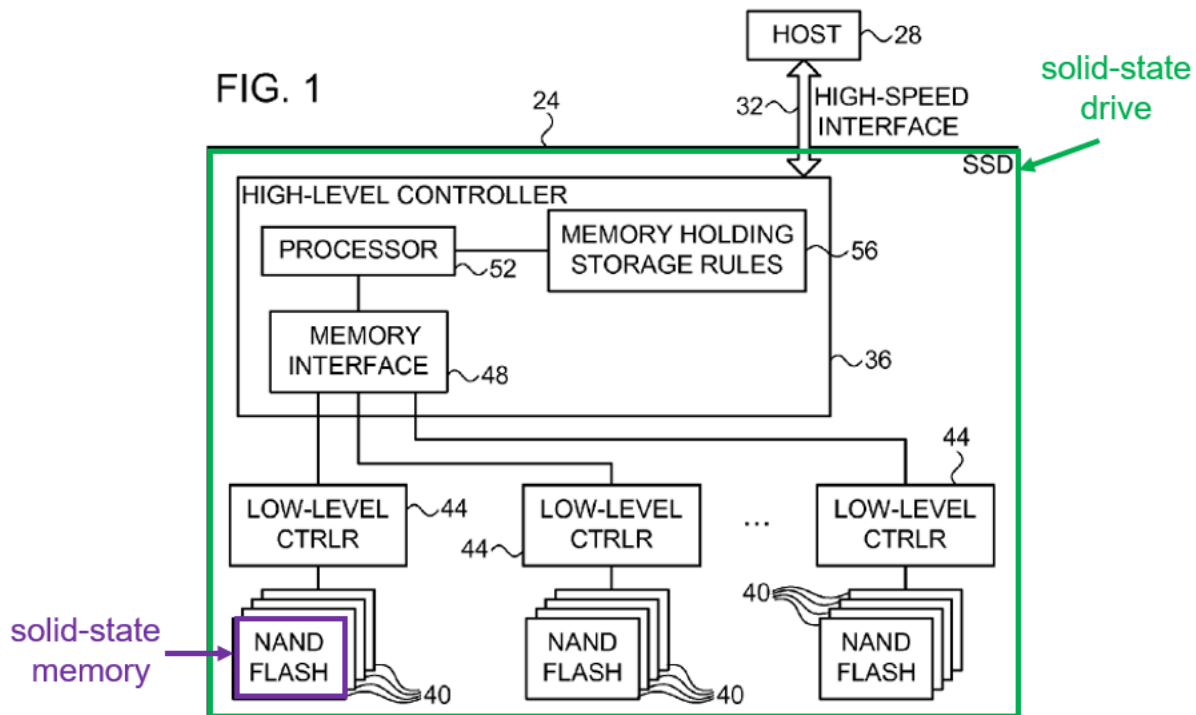
186. Salessi further explains that the data (*content*) is *cached data*. Ex. 1011, [0006] (“capable of processing the data . . . being stored or retrieved from the flash memory.”), [0023] (the processing unit “us[es] a processing algorithm[] to generate processed data.” Salessi describes that “[o]ne or more computer processors operating in accordance with instructions may implement the functions associated with managing use of cache devices in accordance with the present disclosure as described above.” Ex. 1011, [0058]. In my opinion, a person of ordinary skill in the art would have understood that the data being stored in a cache device is *cached data*. Such a skilled person would have been motivated to combine Meir with Salessi for the reasons I discussed above. *See* Sections VIII.A-B.

E. Independent Claim 11

1. Preamble 11[pre]: “A method of operating a solid-state drive, the method comprising:”

187. I have been asked to assume the preamble of claim 11 is limiting. Under that assumption, it is my opinion that Meir discloses the preamble of claim 11, which recites “[a] method of operating a solid-state drive.” As I discussed for 1[pre], Meir describes a solid-state drive (“SSD”) 24 (*solid-state drive*). *See* Section

VIII.C.1; Ex. 1010, FIG. 1, [0029] (“FIG. 1 is a block diagram that schematically illustrates a Solid State Disk (SSD) 24, in accordance with an embodiment of the present invention.”). Meir explains that “SSD 24 comprises a high-level controller 36, which stores data in one or more non-volatile memory units 40.” *Id.*, [0030].

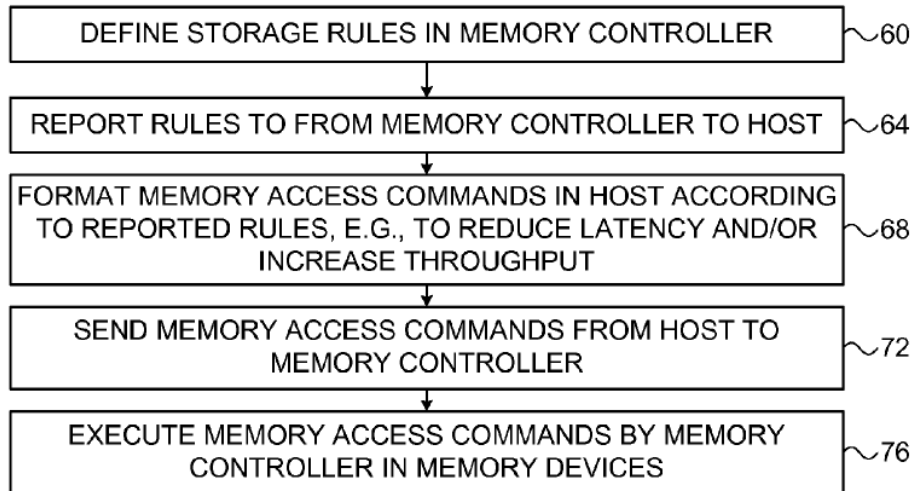


Ex. 1010, Fig. 1 (annotations in color)

188. Meir teaches *a method of operating SSD 24 (solid-state drive)*. For example, Meir describes “[a] data storage method” using SSD 24 that “accepts memory access commands from a host for execution in one or more memory units.” *Id.*, Abstract. Meir also explains that “FIG. 2 is a flow chart that schematically

illustrates a method for data storage.” *Id.*, [0019]. This is, therefore, a method of operating the solid-state drive.

FIG. 2



Ex. 1010, Fig. 2

2. Limitation 11[a]: “in response to a storage request comprising content, processing the content based on a storage policy;”

189. In my opinion, Meir in view of Salessi renders obvious 11[a]: “in response to a storage request comprising content, processing the content based on a storage policy,” as I discussed above for claim 1[e.i]. *See* Section VIII.C.7.

190. Meir describes that, *in response to a* write command including the data to be stored (*storage request comprising content*), the memory controller will *process* the data (*content*) *based on a storage policy*. For example, Meir describes that “[t]he memory controller typically executes the memory access commands received from the host in accordance with a certain policy.” Ex. 1010, [0021],

[0033] (“memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40”).

191. Meir goes on to explain that “[t]he policy comprises a set of rules, for example parallelization rules, pipeline rules prioritization rules, redundancy rules, power consumption rules and/or memory unit characteristics rules.” *Id.*, [0037]. As an example of policy-based storage discussed by Meir, when applying the redundancy rules from the policy, the software (*application*) can direct the processor 52 to *process* the data (*content for storage*) by implementing RAID, e.g., striping the data dividing it into blocks, mirroring the data (making a copy of the data to store a redundant copy in other memory units), calculating parity information on the data, and/or storing the parity information together with the data it relates to. *Id.*, [0025] (“[t]he memory controller policy involves storing the data while applying a redundant storage scheme such as Redundant Array of Inexpensive Disks (RAID) among the memory units”). See my discussion above in Section IV.F. In my opinion, a person of ordinary skill in the art would have understood that applying RAID (e.g., calculating parity or similar redundancy coding information) constitutes processing the data content.

192. To the extent that Meir does not disclose that the software (*instructions*) directs the processor 52 (*programmable processor*) to *process* the data (*content for storage*) included with the commands *based on a storage policy*, it is my opinion

that this is rendered obvious by the system resulting from the combination of the teachings of Meir and Salessi.

193. Salessi teaches that its processing unit can *process content for storage* because it is “capable of processing the data . . . being stored or retrieved from the flash memory.” Ex. 1011, [0006]. Salessi gives examples of processing: “Processing capabilities incorporated into the SSD may include encryption and decryptions [and] compression and decompression.” *Id.* Salessi further teaches that the processing unit “us[es] a processing algorithm[] to generate processed data” and “stor[es] . . . the processed data in the flash memory.” *Id.*, [0023].

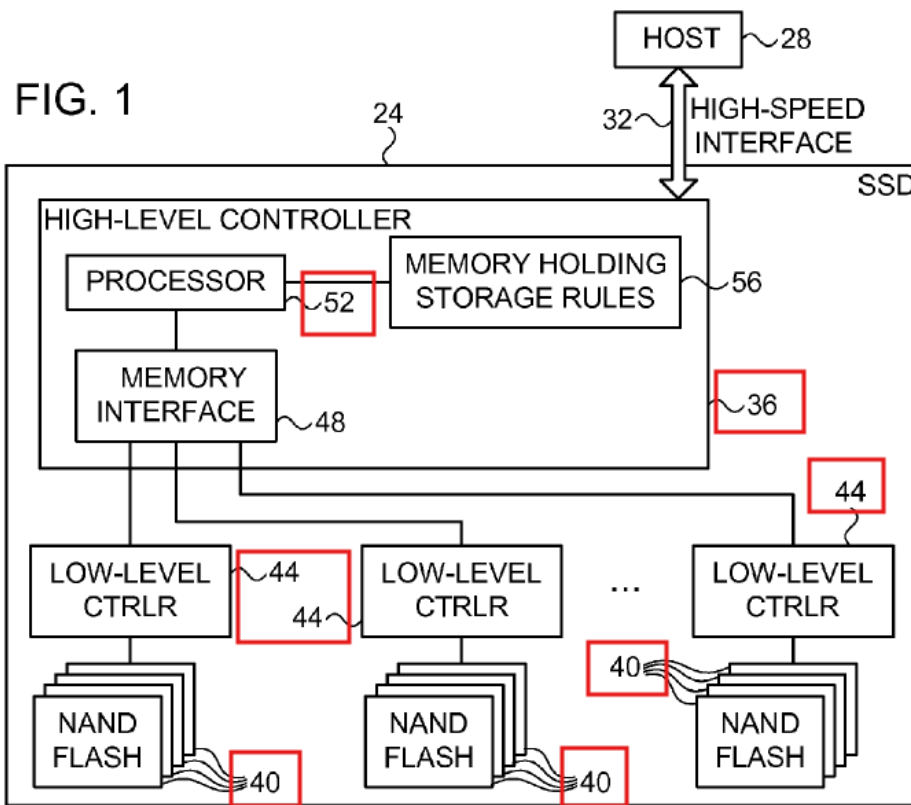
194. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi with a reasonable expectation of success for the reasons I discussed above. *See* Sections VIII.A-B, VIII.C.7.

3. Limitation 11[b]: “storing the processed content on the solid-state memory;”

195. In my opinion, Meir discloses 11[b]: “storing the processed content on the solid-state memory,” as I discussed for claim 1[f.ii]. *See* Section VIII.C.11.

196. Meir explains that the “memory controller . . . executes the commands in the memory units.” Ex. 1010, [0020]; *see also id.*, [0032] (“the high-level controller, the low-level controllers and the processing circuitry in the memory units are referred to collectively as a memory controller”). In the embodiment where the memory controller includes high-level controller 36 and low-level controller 44, the

high-level controller 36 communicates with memory units 40 (*solid-state memory*) through its connection with low-level controller 44 (*device controller*). *Id.*, [0033] (“High-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example via low-level controllers 44), and a processor 52 that carries out the various functions of the high-level controller.”). As such, the low-level controller 44 implements the execution of a write command by writing the data (*storing the content*) to the memory unit 40 (*on the solid-state memory*) according to the logical-to-physical address mapping defined by the parallelism rule that is part of the policy. *Id.*, [0040] (“According to the policy, the memory controller translates the logical addresses to respective physical storage locations in memory units 40, and executes the commands in the memory units according to the logical-to-physical address mapping.”).



Ex. 1010, Fig. 1 (annotations in color)

197. Meir explains that advantages of partitioning the execution of a command between the high-level controller 36 (as executed by processor 52) and low-level controller 44. Namely, the processor 52 in high-level controller 36 can distribute the execution of various commands among various memory units 40 (*solid-state memory*) via the low-level controller 44. Ex. 1010, [0037] (“Parallelism rules define how the memory controller distributes the memory access commands among the memory units. Pipeline rules define properties of a pipeline in the memory controller (not shown in the figure) that executes the commands.”).

198. Meir explains that “[b]eing aware of the parallelism rule, the host is able to assign LBAs to memory access commands in a way that increases the parallelism among memory units 40 and/or low-level controllers 44.” *Id.*, [0042]. For example, while one low-level controller 44 is storing data, processor 52 of the high-level controller 36 can execute another command, sending data to another low-level controller 44 to store. Meir goes on to explain that “LBA assignment or ordering of this sort” in accordance with the storage policy “may avoid temporary congestion in a certain memory unit or low-level controller, and thus reduce latency.” Ex. 1010, [0042].

4. Limitation 11[c]: “in response to a read request, reading requested content from the solid-state memory;”

199. In my opinion, Meir discloses 11[c]: “in response to a read request, reading requested content from the solid-state memory,” as I discussed above in 2[a.i]. *See* Section VIII.D.1.c.

200. Meir explains that the “memory controller . . . executes the commands in the memory units.” Ex. 1010, [0020] ; *see also id.*, [0032] (“the high-level controller, the low-level controllers and the processing circuitry in the memory units are referred to collectively as a memory controller”). In the embodiment where the memory controller includes high-level controller 36 and low-level controller 44 (*device controller*), the high-level controller 36 communicates with memory units

40 (*solid-state memory*) via low-level controller 44 (*device controller*). *Id.*, [0033] (“High-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example via low-level controllers 44), and a processor 52 that carries out the various functions of the high-level controller.”). As such, as a part of executing a read command, the high-level controller 36 “translates the logical addresses to respective physical storage locations in memory units 40, and executes the commands in the memory units according to the logical-to-physical address mapping.” *Id.* [0040].

201. For example, “a certain subset of bits in the logical address may indicate the . . . low-level controller in which this address is assigned.” *Id.*, [0041]. After translation and *in response to the* read command (*read request*), the high-level controller 36 will direct that assigned low-level controller 44 (*device controller*), to execute the access command, here the read command (*read requested content from the solid-state memory*). *Id.*, [0044] (“In a pipelined execution the memory controller breaks the commands into parts for execution by different pipeline stages, and executes one part of one command concurrently with execution of another part of another command. The different pipeline stages may be executed by different controllers in the memory controller.”).

202. As Meir further explains, in certain embodiments, the SSD 24 may use error correction code (“ECC”) to encode the data (*content*) to prevent errors. *Id.*,

[0045]. Meir goes on to explain that, in this embodiment, the memory unit fetches the data, the low-level controller decodes the ECC, and the high-level controller sends the data to the host. *Id.* Thus, Meir teaches that when a read command is received, the low-level controller 44 will use the processing circuitry in the memory units 40 to fetch the ECC-encoded data (*read requested content from the solid-state memory*) and the low-level controller 44 will then decode the data. *Id.*

5. Limitation 11[d]: “processing the requested content for transfer based on the storage policy; and”

203. In my opinion, Meir in view of Salessi renders obvious 11[d]: “processing the requested content for transfer based on the storage policy,” as I discussed above for claim 2[b.ii]. *See* Section VIII.D.1.g.

204. Meir teaches that “SSD 24 comprises a high-level controller 36, which stores data in one or more non-volatile memory units 40.” Ex. 1010, [0030]. This data is *requested content* because it was requested by the host through a read command.

205. Meir describes the memory controller *processing* memory access commands *based on a storage policy*. For example, Meir describes that “[t]he memory controller typically executes the memory access commands received from the host in accordance with a certain policy.” Ex. 1010, [0021], [0033] (“memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40”). As I discussed above, parallelization rules,

pipeline rules, and prioritization rules are storage policies. *Id.*, [0037] (“The policy comprises a set of rules, for example parallelization rules, pipeline rules, prioritization rules, redundancy rules, power consumption rules and/or memory unit characteristics rules.”); *see also* Section VIII.C.7. As an example of policy-based storage discussed by Meir, when applying the redundancy rules from the policy, the software directs the processor 52 to process data by implementing RAID. Ex. 1010, [0025] (“[t]he memory controller policy involves storing the data while applying a redundant storage scheme such as Redundant Array of Inexpensive Disks (RAID) among the memory units”).

206. To the extent that Meir does not explicitly disclose that the software (*instructions*) executed directs the processor 52 (*programmable processor*) to be *processing* the data (*requested content*) included with the commands *based on a storage policy*, it is my opinion that this is rendered obvious by the system resulting from the combination of Meir and Salessi.

207. Salessi teaches that its processing unit can *process requested content* because it is “capable of processing the data . . . being stored or retrieved from the flash memory.” Ex. 1011, [0006]. Salessi gives examples of processing: “Processing capabilities incorporated into the SSD may include encryption and decryptions [and] compression and decompression.” *Id.* Salessi further teaches that the processing unit “process[es], using a processing algorithm, data received from

the flash memory to form processed data.” *Id.*, [0007]; *see also id.*, [0016] (“According to an embodiment of the present invention there is provided a method of . . . processing, by the solid state drive, the retrieved data, using a processing algorithm, to generate processed data.”).

208. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi with a reasonable expectation of success for the reasons I discussed above. *See* Sections VIII.A-B, VIII.D.1.g.

6. Limitation 11[e]: “transferring the requested content.”

209. In my opinion, Meir discloses 11[e]: “transferring the requested content,” as I discussed for 2[b.iii]. *See* Section VIII.D.1.h. Meir describes “sending the data” (*transfer the requested content*) “to the host (e.g., by the high-level controller).” Ex. 1010, [0045].

F. Dependent Claims 14-18

1. Claim 14

a. Preamble 14[pre]: “The method of claim 11 wherein:”

210. In my opinion, to the extent 14[pre]: “The method of claim 11 wherein” is limiting, Meir discloses it. As I discussed above, Meir describes a *method operating* an SSD 24 (*solid-state drive*) as recited in claim 11. *See* Section VIII.E.1.

b. Limitation 14[a]: “processing the content based on the storage policy comprises compressing the content based on the storage policy;”

211. In my opinion, Meir in view of Salessi renders obvious 14[a]: “processing the content based on the storage policy comprises compressing the content based on the storage policy.” As I discussed above, Meir in view of Salessi teaches *processing the content based on the storage policy*. See Section VIII.E.2.

212. Meir in view of Salessi further teaches *compressing the content based on the storage policy*. See Section VIII.D.2.c

213. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi for the reasons I discussed previously. *Id.*

c. Limitation 14[b]: “storing the processed content on the solid-state memory comprises storing the compressed content on the solid-state memory;”

214. In my opinion, Meir in view of Salessi renders obvious 14[b]: “storing the processed content on the solid-state memory comprises storing the compressed content on the solid-state memory.” As I discussed, Meir in view of Salessi teaches *storing the processed content on the solid-state memory*. See Section VIII.E.3.

215. Meir in view of Salessi further teaches *storing the compressed content on the solid-state memory*. See Section VIII.D.2.g.

216. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi for the reasons I discussed above. *Id.*

d. Limitation 14[c]: “reading requested content from the solid-state memory comprises reading the compressed content from the solid-state memory;”

217. In my opinion, Meir in view of Salessi renders obvious 14[c]: “reading requested content from the solid-state memory comprises reading the compressed content from the solid-state memory.” As I discussed above, Meir in view of Salessi teaches *reading requested content from the solid-state memory*. See Section VIII.E.4.

218. Meir in view of Salessi also teaches *reading the compressed content from the solid-state memory*. See Section VIII.D.3.c.

219. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi for the reasons I discussed above. *Id.*

e. Limitation 14[d]: “processing the requested content for transfer based on the storage policy comprises decompressing the compressed content based on the storage policy; and”

220. In my opinion, Meir in view of Salessi renders obvious 14[d]: “processing the requested content for transfer based on the storage policy comprises decompressing the compressed content based on the storage policy.” As I discussed previously, Meir in view of Salessi teaches *processing the requested content for transfer based on the storage policy*. See Section VIII.E.5.

221. Meir in view of Salessi further teaches *decompressing the compressed content based on the storage policy*. See Section VIII.D.3.g.

222. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi for the reasons I discussed above. *Id.*

f. Limitation 14[e]: “transferring the requested content comprises transferring the decompressed content.”

223. In my opinion, Meir in view of Salessi renders obvious 14[e]: “transferring the requested content comprises transferring the decompressed content.” As I discussed, Meir in view of Salessi teaches *transferring the requested content*. See Section VIII.E.6.

224. Meir in view of Salessi also teaches *transferring the decompressed content*. See Section VIII.D.3.h.

225. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi for the reasons I discussed above. *Id.*

2. Claim 15

a. Preamble 15[pre]: “The method of claim 11 wherein:”

226. In my opinion, to the extent 15[pre]: “The method of claim 11 wherein” is limiting, Meir discloses it. As I discussed, Meir explains a method by a SSD 24 (*solid-state drive*) as recited in claim 11. See Section VIII.E.1.

b. Limitation 15[a]: “processing the content based on the storage policy comprises encrypting the content based on the storage policy;”

227. In my opinion, Meir in view of Salessi renders obvious 15[a]: “processing the content based on the storage policy comprises encrypting the content based on the storage policy.” As I discussed, Meir in view of Salessi teaches *processing the content based on the storage policy*. See Section VIII.E.2.

228. Meir in view of Salessi further teaches *encrypting the content based on the storage policy*. See Section VIII.D.4.c.

229. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi for the reasons above. *Id.*

c. Limitation 15[b]: “storing the processed content on the solid-state memory comprises storing the encrypted content on the solid-state memory;”

230. In my opinion, Meir in view of Salessi renders obvious 15[b]: “storing the processed content on the solid-state memory comprises storing the encrypted content on the solid-state memory.” As I discussed above, Meir in view of Salessi teaches *storing the processed content on the solid-state memory*. See Section VIII.E.3.

231. Meir in view of Salessi further teaches *storing the encrypted content on the solid-state memory*. See Section VIII.D.4.g.

232. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi for the reasons I discussed above. *Id.*

d. Limitation 15[c]: “reading requested content from the solid-state memory comprises reading the encrypted content from the solid-state memory;”

233. In my opinion, Meir in view of Salessi renders obvious 15[c]: “reading requested content from the solid-state memory comprises reading the encrypted content from the solid-state memory.” As I discussed previously, Meir in view of Salessi teaches *reading requested content from the solid-state memory*. See Section VIII.E.4.

234. Meir in view of Salessi also teaches *reading the encrypted content from the solid-state memory*. See Section VIII.D.5.c.

235. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi for the reasons I discussed above. *Id.*

e. Limitation 15[d]: “processing the requested content for transfer based on the storage policy comprises decrypting the encrypted content based on the storage policy; and”

236. In my opinion, Meir in view of Salessi renders obvious 15[d]: “processing the requested content for transfer based on the storage policy comprises decrypting the encrypted content based on the storage policy.” As I discussed, Meir in view of Salessi teaches *processing the requested content for transfer based on the storage policy*. See Section VIII.E.5.

237. Meir in view of Salessi also teaches *decrypting the encrypted content based on the storage policy*. See Section VIII.D.5.g.

238. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi for the reasons I discussed above. *Id.*

f. Limitation 15[e]: “transferring the requested content comprises transferring the decrypted content.”

239. In my opinion, Meir in view of Salessi renders obvious 15[e]: “transferring the requested content comprises transferring the decrypted content.” As I discussed above, Meir in view of Salessi teaches *transferring the requested content*. See Section VIII.E.6.

240. Meir in view of Salessi further teaches *transferring the decrypted content*. See Section VIII.D.5.h.

241. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi for the reasons I discussed above. *Id.*

3. Claim 16: “The method of claim 11 wherein the storage policy indicates how to process content for storage on the solid-state drive and how to process the requested content for transfer.”

242. In my opinion, Meir in view of Salessi renders obvious 16: “The method of claim 11 wherein the storage policy indicates how to process content for storage on the solid-state drive and how to process the requested content for transfer.”

243. Meir describes that the memory controller will *process* commands *based on a storage policy* because “[t]he memory controller typically executes the memory access commands received from the host in accordance with a certain policy.” Ex. 1010, [0021], [0033] (“memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40”). “The policy comprises a set of rules, for example one or more parallelization rules, pipeline rules and/or prioritization rules.” *Id.*, [0033]; *see also* [0025] (“[t]he memory controller policy involves storing the data while applying a redundant storage scheme such as Redundant Array of Inexpensive Disks (RAID) among the memory units”).

244. To the extent that Meir does not disclose wherein the storage policy indicates how to process content for storage on the solid-state drive and how to process the requested content for transfer, it is my opinion that this is rendered obvious by the combination system of Meir and Salessi.

245. Salessi teaches *how to process content for storage on the solid-state drive and how to process the requested content for transfer* because its processing unit is “capable of processing the data . . . being stored or retrieved from the flash memory.” Ex. 1011, [0006]. “Processing capabilities incorporated into the SSD may include encryption and decryptions [and] compression and decompression.” *Id.* Salessi further teaches that the processing unit “us[es] a processing algorithm[] to generate processed data.” *Id.*, [0023].

246. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi for the reasons above. *See* Section VIII.C.7.

4. Claim 17: “The method of claim 16 wherein the storage policy comprises one or more of an encryption policy, a decryption policy, a compression policy, or a decompression policy.”

247. In my opinion, Meir in view of Salessi renders obvious 17: “The method of claim 16 wherein the storage policy comprises one or more of an encryption policy, a decryption policy, a compression policy, or a decompression policy.”

248. Meir describes a *storage policy* because “[t]he memory controller typically executes the memory access commands received from the host in accordance with a certain policy.” Ex. 1010, [0021], [0033] (“memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40”). “The policy comprises a set of rules, for example one or more parallelization rules, pipeline rules and/or prioritization rules.” *Id.*, [0033]; *see also* [0025] (“[t]he memory controller policy involves storing the data while applying a redundant storage scheme such as Redundant Array of Inexpensive Disks (RAID) among the memory units”).

249. To the extent that Meir does not disclose wherein the storage policy comprises one or more of an encryption policy, a decryption policy, a compression

policy, or a decompression policy, it is my opinion that this is rendered obvious by the system resulting from the combination of the teachings of Meir and Salessi.

250. Salessi teaches the storage policy comprises one or more of an encryption policy, a decryption policy, a compression policy, or a decompression policy because “[p]rocessing capabilities incorporated into the SSD may include encryption and decryptions [and] compression and decompression.” Ex. 1011, [0006]. Salessi further teaches that the processing unit “us[es] a processing algorithm[] to generate processed data.” *Id.*, [0023].

251. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi for the reasons above. *See* Section VIII.C.7.

5. Claim 18: “The method of claim 17 further comprising retrieving the storage policy from memory.”

252. In my opinion, Meir discloses claim 18: “The method of claim 17 further comprising retrieving the storage policy from memory.”

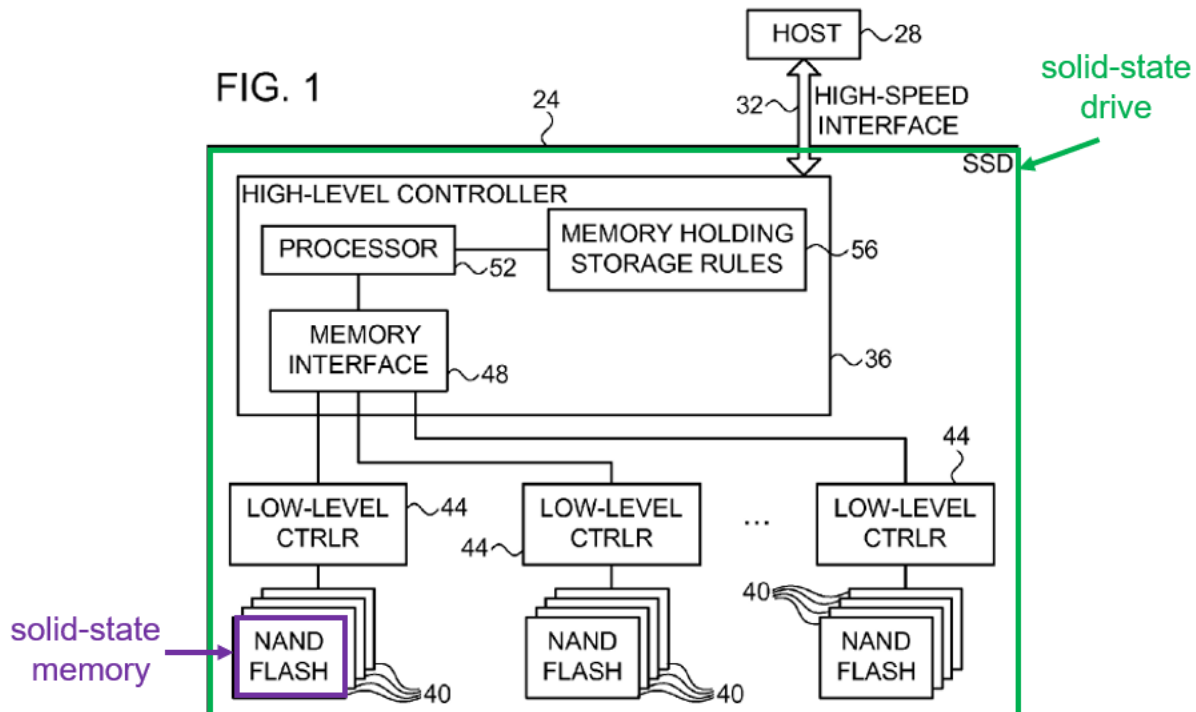
253. Meir describes that the *storage policy* is stored in *memory* 56. Ex. 1010, FIG. 1, [0033] (“memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40”). Meir further explains that “[t]he memory controller typically executes the memory access commands received from the host in accordance with a certain policy.” Ex. 1010, [0021]. In my opinion, a person of ordinary skill in the art would have understood that the memory controller

would need to *retrieve the storage policy from memory* in order to execute memory access commands in accordance with that storage policy.

G. Independent Claim 20:

1. Preamble 20[pre]: “A solid-state drive comprising:”

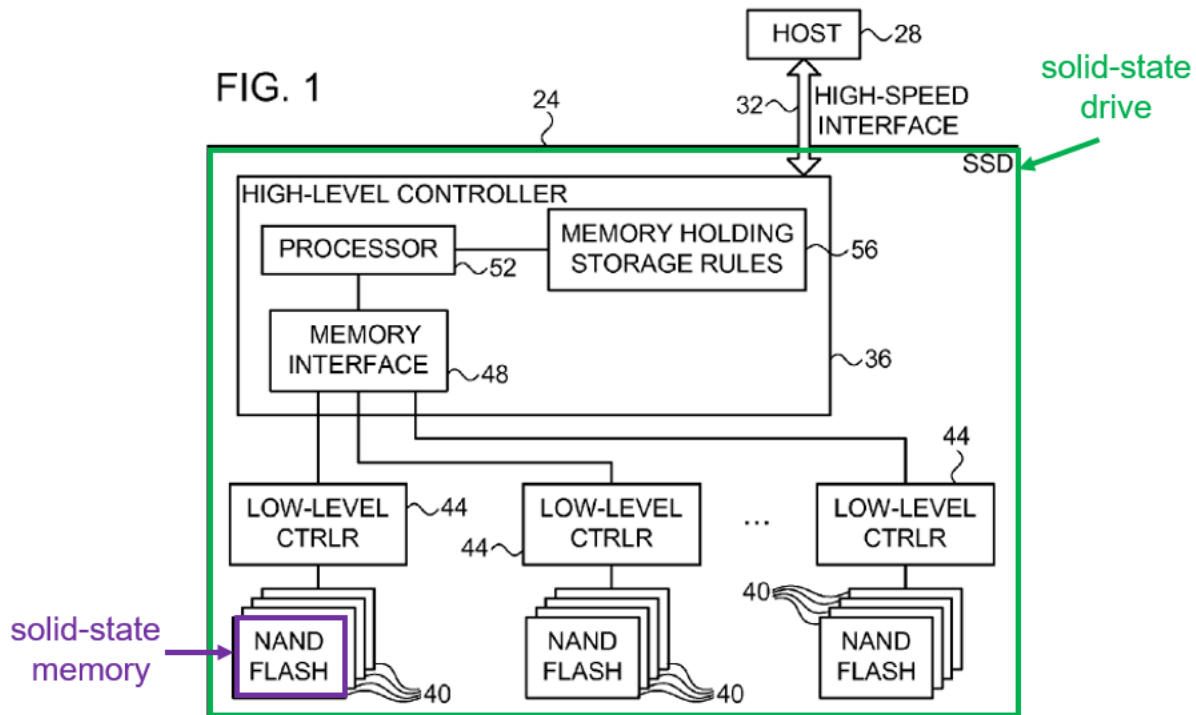
254. I have been asked to assume the preamble of claim 20 is limiting. Under that assumption, it is my opinion that Meir discloses the preamble of claim 20, which recites “[a] solid-state drive.” As I discussed for 1[pre], Meir describes a solid-state drive (“SSD”) 24 (*solid-state drive*). See Section VIII.C.1; Ex. 1010, FIG. 1, [0029] (“FIG. 1 is a block diagram that schematically illustrates a Solid State Disk (SSD) 24, in accordance with an embodiment of the present invention.”).



Ex. 1010, Fig. 1 (annotations in color)

2. Limitation: 20[a]: “a solid-state memory;”

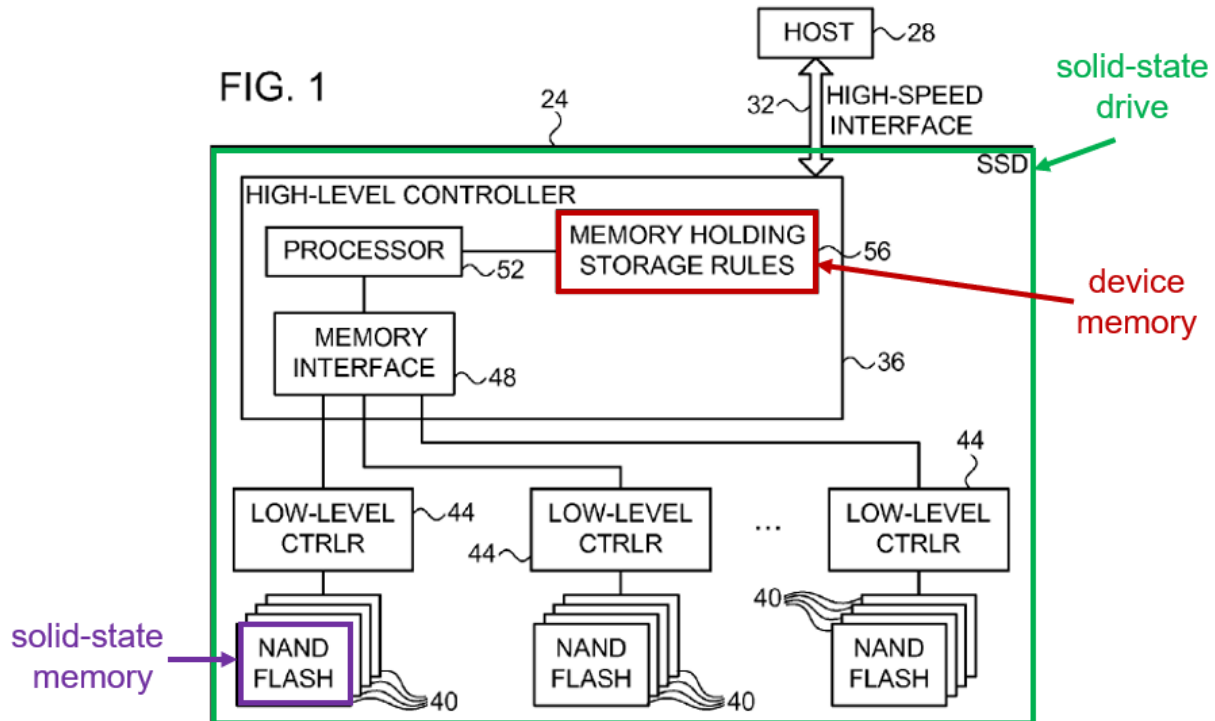
255. In my opinion, as I discussed for 1[a], Meir discloses 20[a]: “a solid-state memory.” *See* Section VIII.C.2. Meir describes memory unit 40 (*solid-state memory*). *See* Section VIII.C.2; Ex. 1010, FIG. 1, [0030] (“SSD 24 comprises a high-level controller 36, which stores data in one or more non-volatile memory units 40.”). “[M]emory units 40 comprise NAND Flash devices or dies” or “any other suitable type of non-volatile memory units.” *Id.* NAND Flash and other non-volatile memories are types of solid-state memories. Flash devices (i.e., flash memory) are ubiquitous in solid-state devices. A person of ordinary skill in the art would have understood that flash memory is a type of solid-state memory because it has no moving parts and stores data electronically using microchips.



Ex. 1010, Fig. 1 (annotations in color)

3. Limitation 20[b]: “a device memory;”

256. In my opinion, as I discussed for 1[b], Meir discloses 20[b]: “a device memory.” See Section VIII.C.3. Meir describes memory 56 (*device memory*). See Section VIII.C.3; Ex. 1010, FIG. 1, [0033] (“memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40”), cl. 18 (“[a] memory controller, comprising: a memory, which is configured to hold a definition of a policy to be applied by the memory controller in accessing one or more memory units”).

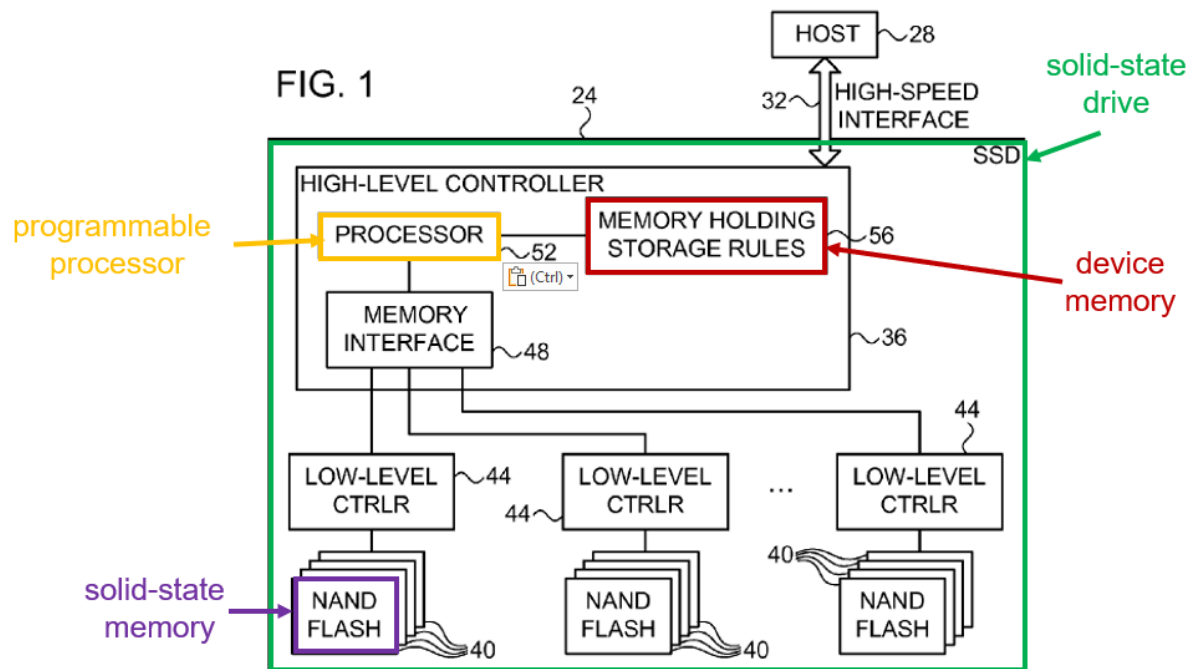


Ex. 1010, Fig. 1 (annotations in color)

4. **Limitation 20[c]: “a programmable processor operatively coupled to the device memory;”**

257. In my opinion, as I discussed for 1[c], Meir discloses 20[c]: “a programmable processor operatively coupled to the device memory.” Meir describes processor 52 (*programmable processor*) *operatively coupled to* memory 56 (*device memory*). See Section VIII.C.4; Ex. 1010, FIG. 1, [0033] (“a processor 52 that carries out the various functions of the high-level controller”), [0034] (“the memory controller comprises one or more general-purpose processors”), cl. 18 (“[a] memory controller, comprising . . . a processor, which is configured to . . . execute the received memory access commands in the memory units”). Programmable processors are chips or devices that run software instructions to perform different

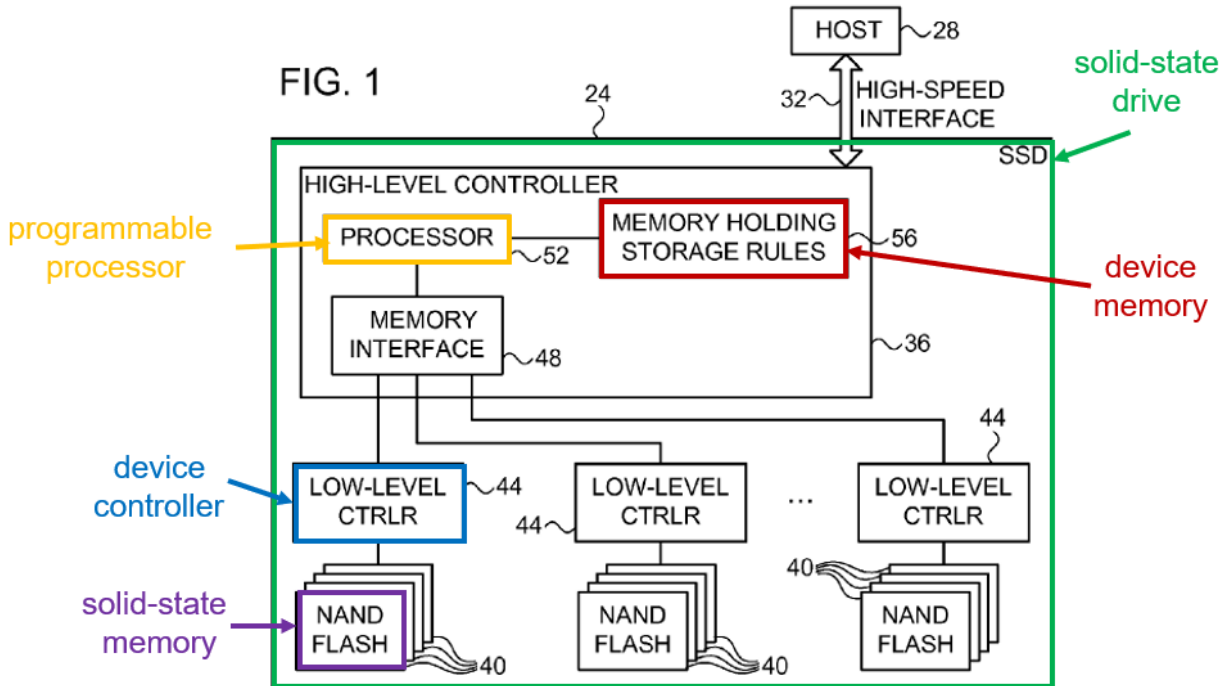
tasks. Types of programmable processors include, but are not limited to, central processing units (CPUs), digital signal processors (DSPs), microcontrollers, and programmable logic controllers (PLCs), and field-programmable gate arrays (FPGAs). A person of ordinary skill in the art would have known that a processor like processor 52 that carries out the functions of a controller would be programmable, because the processor would have to be programmed to carry out those functions. Processor 52 (*programmable processor*) is *operatively coupled to* memory 56 (*device memory*) as illustrated by the connection between these two blocks in Figure 1. *Id.*



Ex. 1010, Fig. 1 (annotations in color)

5. Limitation 20[d]: “a device controller operatively coupled with the solid-state memory and the programmable processor; and”

258. In my opinion, as I discussed for 1[d], Meir discloses 20[d]: “a device controller operatively coupled with the solid-state memory and the programmable processor.” *See* Section VIII.C.5. Meir describes low-level controller 44 (*device controller*). *See* Section VIII.C.5; Ex. 1010, FIG. 1, [0031] (“SSD 24 comprises multiple low-level controllers 44”). As illustrated in Figure 1 below by the connection between low-level controller 44 and memory unit 40, low-level controller 44 (*device controller*) is *operatively coupled with the* memory unit 40 (*solid-state memory*). *Id.* Low-level controller 44 (*device controller*) is also *operatively coupled to* processor 52 (*programmable processor*) through interface 48 which permits processor 52 to communicate memory units 40 via the low-level controller 44. *See* Section VIII.C.5; Ex. 1010, FIG. 1, [0031], [0033] (“High-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example via low-level controllers 44), and a processor 52 that carries out the various functions of the high-level controller.”).



Ex. 1010, Fig. 1 (annotations in color)

6. **Limitation 20[e]: “instructions stored on the device memory that, when executed by the programmable processor, direct the programmable processor to:”**

259. In my opinion, as I discussed for 1[e], Meir discloses 20[e]: “instructions stored on the device memory that, when executed by the programmable processor, direct the programmable processor.” See Section VIII.C.6. Meir describes software comprising *instructions stored on memory 56 (device memory), that when executed by processor 52 (programmable processor), direct the programmable processor to execute the access commands according to a policy stored on the memory 56.* See Section VIII.C.6; Ex. 1010, FIG. 1, [0034]. Meir describes in further detail:

Some or all of the functions of the memory controller may be implemented in hardware. Alternatively, the memory controller may comprise one or more microprocessors that run suitable software, or a combination of hardware and software elements. In some embodiments, the memory controller comprises one or more general-purpose processors, which are programmed in software to carry out the functions described herein. The software may be downloaded to the processors in electronic form, over a network, for example, or it may, alternatively or additionally, be provided and/or stored on non-transitory tangible media, such as magnetic, optical, or electronic memory.

Id.; *see also id.*, [0032] (“the high-level controller, the low-level controllers and the processing circuitry in the memory units are referred to collectively as a memory controller”).

260. Meir explains that the software can be *stored* in the memory 56 (*device memory*) for processor 52 (*programmable processor*) to execute. *Id.*, [0034] (“[t]he software may be . . . stored on non-transitory tangible media.”). Meir also describes that the software comprises program *instructions* that cause the processor to perform functions when the instructions are executed. *Id.* (the software is programmed “to carry out the functions” performed by the processor).

7. Limitation 20[e.i]: “process content for storage on the solid-state memory based on a storage policy;”

261. In my opinion, as I discussed for 1[e.i], Meir in view of Salessi renders obvious 20[e.i]: “process content for storage on the solid-state memory based on a storage policy.” *See* Section VIII.C.7.

262. Meir describes that the memory controller will *process* the data (*content*) based on a storage policy. For example, Meir describes that “[t]he memory controller typically executes the memory access commands received from the host in accordance with a certain policy.” Ex. 1010, [0021], [0033] (“memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40”). Further, the data (*content*) is stored in memory units 40 (*on the solid-state memory*). See Section VIII.C.2.

263. Meir goes on to explain that “[t]he policy comprises a set of rules, for example parallelization rules, pipeline rules, prioritization rules, redundancy rules, power consumption rules and/or memory unit characteristics rules.” *Id.*, [0037]. As an example of policy-based storage discussed by Meir, when applying the redundancy rules from the policy, the software (*application*) can direct the processor 52 to *process* the data (*content for storage*) by implementing RAID, e.g., striping the data dividing it into blocks, mirroring the data (making a copy of the data to store a redundant copy in other memory units), calculating parity information on the data, and/or storing the parity information together with the data it relates to. *Id.*, [0025] (“[t]he memory controller policy involves storing the data while applying a redundant storage scheme such as Redundant Array of Inexpensive Disks (RAID) among the memory units”). See my discussion above in Section IV.F. In my opinion, a person of ordinary skill in the art would have understood that applying

RAID (e.g., calculating parity or similar redundancy coding information) constitutes *processing* the data *content*.

264. To the extent that Meir does not disclose that the software (*instructions*) directs the processor 52 (*programmable processor*) to *process* the data (*content for storage*) included with the commands *based on a storage policy*, it is my opinion that this is rendered obvious by the system resulting from the combination of the teachings of Meir and Salessi.

265. Salessi teaches that its processing unit can *process content for storage* because it is “capable of processing the data . . . being stored or retrieved from the flash memory.” Ex. 1011, [0006]. Salessi gives examples of processing: “Processing capabilities incorporated into the SSD may include encryption and decryptions [and] compression and decompression.” *Id.* Salessi further teaches that the processing unit “us[es] a processing algorithm[] to generate processed data” and “stor[es] . . . the processed data in the flash memory.” *Id.*, [0023].

266. In my opinion, a person of ordinary skill in the art would combine the teachings of Meir with that of Salessi with a reasonable expectation of success for the reasons I discussed above. *See* Sections VIII.A-B, VIII.C.7.

8. Limitation 20[e.ii]: “transfer the content to the device controller to store on the solid-state memory;”

267. In my opinion, as I discussed for 1[e.ii], Meir discloses 20[e.ii]: “transfer the content to the device controller to store on the solid-state memory.” *See* Section VIII.C.8, VIII.C.11.

268. Once a write command is received, the processor 52 processes the command according to the policy and transfers the data to the assigned low-level controller 44. *Id.*, [0005]. In particular, the parallelism rules defined by the policy will “specify distribution of the memory access commands among the memory units.” *Id.* “[A] mapping, defined in the memory controller, between logical addresses associated with the memory access commands and respective physical storage location in the memory units” will be used by processor 52 via its software to direct the access command to the appropriate low-level controller 44 (the low-level controller assigned to those memory addresses). *Id.*, [0007], [0031] (“each low-level controller assigned to a group of one or more memory units 40”); *see also id.*, [0009], [0023], [0031] (“logical-to-physical address translation is performed by the high-level controller”).

269. In particular, because “a parallelism rule may define . . . ranges of LBAs that are mapped to respective low-level controllers,” processor 52 will use the parallelism rule that is part of the policy to identify the assigned low-level controller 44 to send the data to be stored.” *Id.*, [0041]; *see also* [0041] (“a certain subset of

bits in the logical address may indicate the memory unit or low-level controller in which this address is assigned”).

270. Because the data to be stored is transferred from the host to high-level controller 36, and the low-level controller 44 couples the processor 52 of the high-level controller 36 to the memory units 40, it follows that the software (*application*) of processor 52 sends the data (*transfers the content*) to the low-level controller 44 (*device controller*). *Id.*, FIG. 1, [0033]. As Meir explains, “[h]igh-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example **via low-level controllers 44**).” *Id.*, [0033] (emphasis added).

271. Further, Meir describes *storing the content on the solid-state memory*. See Section VIII.C.11. Meir explains that the “memory controller . . . executes the commands in the memory units.” Ex. 1010, [0020]; *see also id.*, [0032] (“the high-level controller, the low-level controllers and the processing circuitry in the memory units are referred to collectively as a memory controller”). In the embodiment where the memory controller includes high-level controller 36 and low-level controller 44 (*device controller*), the high-level controller 36 communicates with memory units 40 (*solid-state memory*) through its connection with low-level controller 44 (*device controller*). *Id.*, [0033] (“High-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example via low-level controllers 44), and a processor 52 that carries out the various functions of

the high-level controller.”). As such, the low-level controller 44 (*device controller*) implements the execution of a write command by writing the data (*stores the content*) to the memory unit 40 (*on the solid-state memory*) according to the logical-to-physical address mapping defined by the parallelism rule that is part of the policy. *Id.*, [0040] (“According to the policy, the memory controller translates the logical addresses to respective physical storage locations in memory units 40, and executes the commands in the memory units according to the logical-to-physical address mapping.”).

272. As I explained above, Meir explains that advantages of partitioning the execution of a command between the high-level controller 36 (as executed by processor 52 or *programmable processor*) and low-level controller 44. Namely, the processor 52 (*programmable processor*) in high-level controller 36 can distribute the execution of various commands among various memory units 40 (*solid-state memory*) via the low-level controller 44 (*device controller*). Ex. 1010, [0037] (“Parallelism rules define how the memory controller distributes the memory access commands among the memory units. Pipeline rules define properties of a pipeline in the memory controller (not shown in the figure) that executes the commands.”). *See* Section VIII.C.11. A person of ordinary skill in the art would have known that parallelism in storage systems provide higher throughput, scalability, and improved

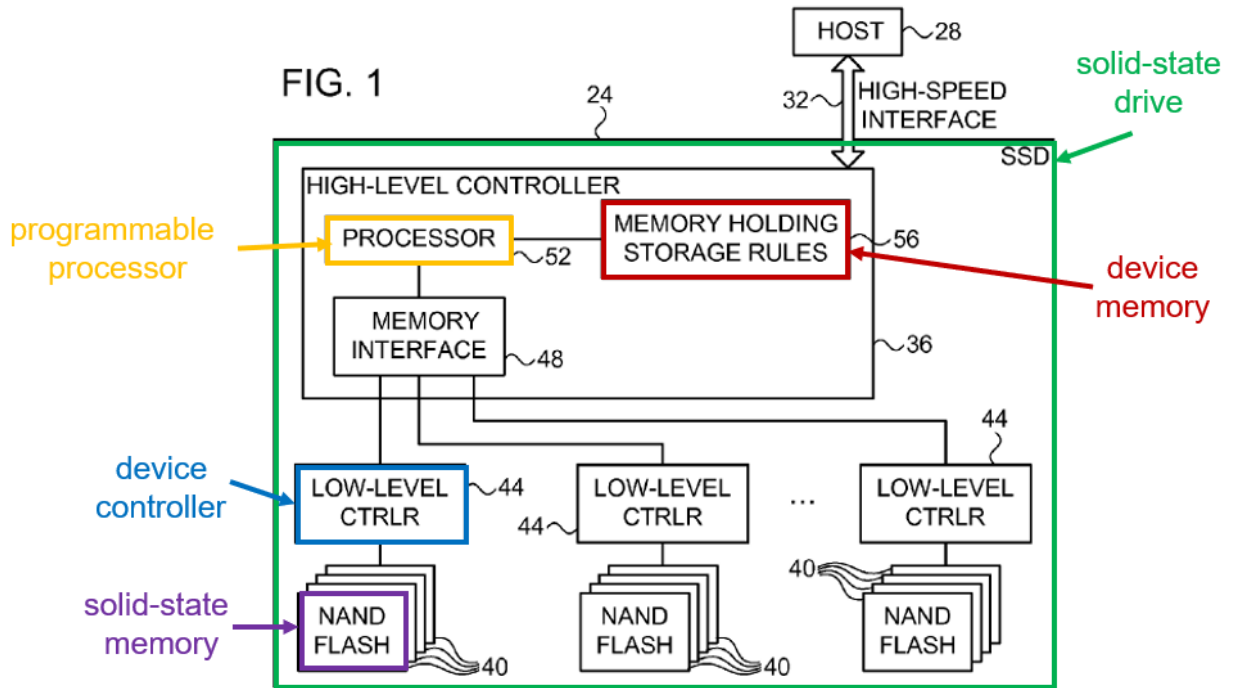
resource use by spreading data tasks across many low-level controllers and memory units at the same time.

9. Limitation 20[e.iii]: “in response to a content request, receive requested content read from the solid-state memory by the device controller;”

273. In my opinion, Meir discloses 20[e.iii]: “in response to a content request, receive requested content read from the solid-state memory by the device controller.”

274. Meir describes software comprising *instructions* stored on memory 56 (*device memory*), that when executed by processor 52 (*programmable processor*), *direct the programmable processor (see Section VIII.C.6) to execute receive data (requested content) from the memory unit 40 (solid-state memory) according to a policy stored on the memory 56 (based on a storage policy), including applying parallelism rules. See Section VIII.C.7; Ex. 1010, FIG. 1, [0020]-[0021] (“[t]he memory controller typically executes the memory access commands received from the host in accordance with a certain policy.”), [0025] (“[t]he memory controller policy involves storing the data while applying a redundant storage scheme such as Redundant Array of Inexpensive Disks (RAID) among the memory units”), [0030] (“SSD 24 comprises a high-level controller 36, which stores data in one or more non-volatile memory units 40.”), [0033] (“High-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present*

example via low-level controllers 44), and a processor 52 that carries out the various functions of the high-level controller.”).



Ex. 1010, Fig. 1 (annotations in color)

10. Limitation 20[e.iv]: “process the requested content for transfer based on the storage policy; and”

275. In my opinion, as I discussed for 11[d], Meir discloses 20[e.iv]: “process the requested content for transfer based on the storage policy.” See Section VIII.E.5.

276. Meir teaches that “SSD 24 comprises a high-level controller 36, which stores data in one or more non-volatile memory units 40.” Ex. 1010, [0030]. This data is *requested content* because it was requested by the host through a read command.

277. Meir describes the memory controller *processing* memory access commands *based on a storage policy*. For example, Meir describes that “[t]he memory controller typically executes the memory access commands received from the host in accordance with a certain policy.” Ex. 1010, [0021], [0033] (“memory 56 holds a definition of a policy that the memory controller applies in storing data in memory units 40”). As I discussed above, parallelization rules, pipeline rules, and prioritization rules are storage policies. *Id.*, [0037] (“The policy comprises a set of rules, for example parallelization rules, pipeline rules, prioritization rules, redundancy rules, power consumption rules and/or memory unit characteristics rules.”); *see also* Section VIII.C.7. As an example of policy-based storage discussed by Meir, when applying the redundancy rules from the policy, the software directs the processor 52 to process data by implementing RAID. Ex. 1010, [0025] (“[t]he memory controller policy involves storing the data while applying a redundant storage scheme such as Redundant Array of Inexpensive Disks (RAID) among the memory units”).

278. The software (*instructions*) can direct the processor 52 to *process* the received data (*requested content for transfer*) based on the redundant storage scheme as I described above. *Id.*, [0025]. For example, when the data has been striped, the processor 52 splits the read request and pulls different parts of the requested data from multiple drives. *Id.*, [0038]. Processor 52 (*programmable processor*) will

reassemble the different parts to transfer to the host. Similarly, when a drive failure has been identified, processor 52 will use the parity data to rebuild the requested data.

11. Limitation 20[e.v]: “transfer the requested content.”

279. In my opinion, as I discussed for 11[e], Meir discloses 20[e.v]: “transfer the requested content.” *See* Section VIII.E.6. Meir describes “sending the data” (transfer the requested content) “to the host (e.g., by the high-level controller).” Ex. 1010, [0045].

IX. OPINIONS ON GROUND 2: MEIR IN VIEW OF SALESSI AND CHOU WOULD HAVE RENDERED OBVIOUS CLAIM 9

A. Motivation to Combine

280. In my opinion, a person of ordinary skill in the art would have been motivated to modify the Meir/Salessi storage drive, including the memory unit 40 (*solid-state memory*), the processor 52 (*programmable processor*), the memory 56 (*device memory*), the low-level controller 44 (*device controller*), the high-speed interface 32 (*communication interface*), and the memory interface 48 (*data link*), to be mounted on Chou’s printed circuit board that is mounted by a case.

281. Mounting the SSD components on a printed circuit board would optimize system performance by providing a compact and reliable platform for electrically interconnecting the various components. A person of ordinary skill in the art would have understood that a PCB provides short, controlled signal paths

between components rather than longer cable-based or external interconnects. This would improve signal integrity, reduce communication latency between components, and reduce the likelihood of communication errors and associated retries or retransmissions. Ex. 1010, [0042] (“maximiz[ing] a throughput of the execution [and] minimiz[ing] a latency of the execution”).

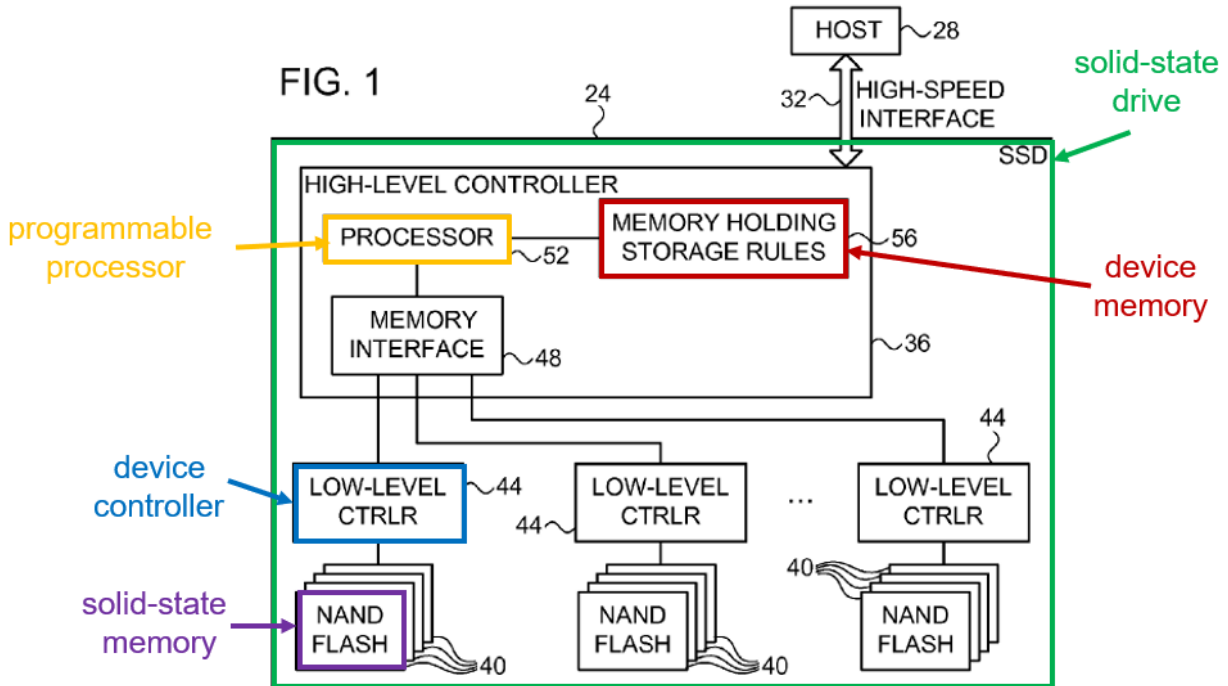
282. Mounting the printed circuit board in a case and enclosing the SSD in the case would improve the resiliency and lifetime of the components. The protective case enhances the durability and reliability of the SSD by shielding the electronic components from physical damage, dust, moisture, electrostatic discharge, and electromagnetic interference. The case also assists with thermal management by dissipating heat generated during operation. A case also provides mechanical support and protects the internal SSD components from external environmental conditions, as well as providing a standardized form factor that facilitates installation into computer and other electronic devices.

283. Mounting the SSD components on a PCB would reduce manufacturing cost and complexity. Doing so would have allowed the PCB to route power and high speed data signals amongst the processors and memory devices while providing mechanical support for the components. A person of ordinary skill in the art would have understood that using a PCB was a conventional way to assemble, align, interconnect, and mechanically support electronic components in a repeatable

manufacturing process, reducing the need for custom wiring or less standardized assembly techniques.

B. Reasonable Expectation of Success

284. In my opinion, a person of ordinary skill in the art would have been able to modify the Meir/Salessi storage drive to incorporate Chou's printed circuit board and case with a reasonable expectation of success. The storage systems of Meir, Salessi, and Chou are SSDs that include controllers and solid-state memory. Ex. 1010, FIG.1; Ex. 1011 [0006] ("In one embodiment, the SSD includes flash memory and an SSD processing unit capable of processing the data in the flash memory"); Ex. 1012, [0002] ("The present invention relates generally to a solid state drive assembly and an assembly method for solid state drives..."). Using printed circuit boards and cases as taught in Chou was a well-known and common design option that was routinely used to provide better stability and protection for hardware.



Ex. 1010, Fig. 1 (annotations in color)

C. Dependent Claim 9

1. Preamble 9[pre]: “The solid-state drive of claim 1 further comprising:”

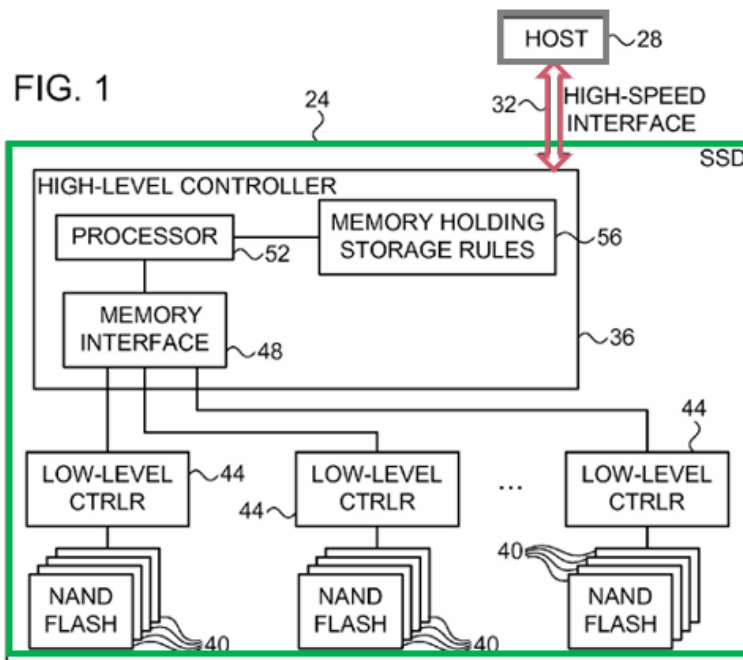
285. In my opinion, Meir discloses 9[pre]: “[t]he solid-state drive of claim 1 further comprising.” Meir describes SSD 24 (*solid-state drive*) as recited in claim

1. See Section VIII.A.1.

2. Limitation 9[a]: “a communication interface to communicatively couple the solid-state drive to a host device.”

286. In my opinion, Meir discloses 9[a]: “a communication interface to communicatively couple the solid-state drive to a host device.”.

287. A communication interface is necessary to allow different components in a system such as Meir's storage system drive (including the SSD and host) to communicate. Meir describes a *high-speed interface 32 (communication interface)* that connects (*communicatively couples*) SSD 24 (*solid-state device*) to host 28 (*host device*) as shown by the connection between these two components in Figure 1; Ex. 1010, FIG. 1; *see also id.*, [0029] ("Host 28 and SSD 24 communicate over a high-speed interface 32, for example a SATA, SAS or NVM express interface."), FIG. 1. This interface allows data (content) to move between SSD 24 (*solid-state device*) and host 28 (*host device*), as seen below. This communication would have been routine and expected, otherwise, host 28 would have no way of interacting with the SSD 24.



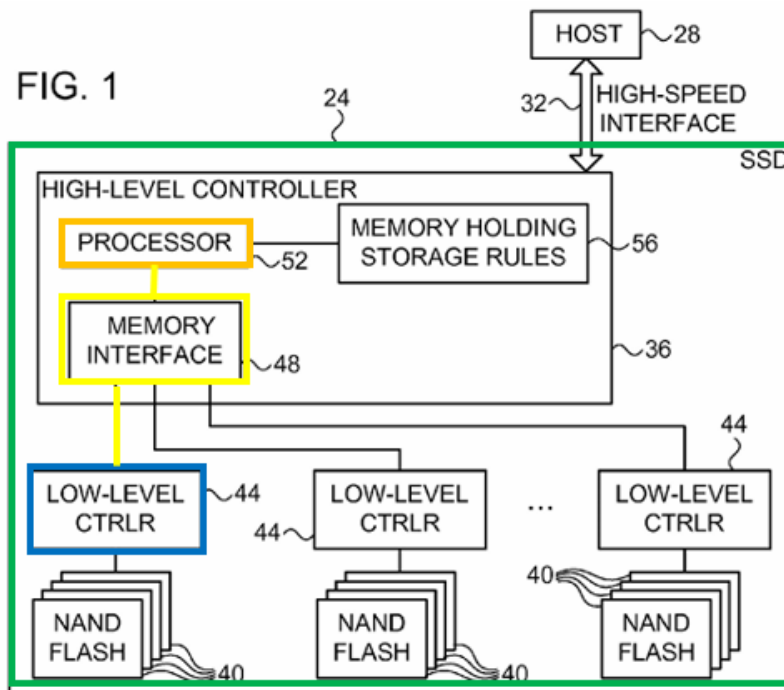
Ex. 1010, Fig. 1 (annotations in color)

3. **Limitation 9[b]: “a data link to communicatively couple the programmable processor and the device controller;”**

288. In my opinion, Meir discloses 9[b]: “a data link to communicatively couple the programmable processor and the device controller.”

289. A data link is necessary to allow different components in a system such as Meir’s storage system drive to communicate between themselves by transferring data or content. Meir describes a memory interface 48 (*data link*) to *communicatively couple* processor 52 (*the programmable processor*) and low-level controller 44 (*the device controller*) as shown by the connection between these two components in Figure 1; Ex. 1010, FIG. 1; *see also id.*, [0033] (“High-level

controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example via low-level controllers 44), and a processor 52 that carries out the various functions of the high-level controller.”). Memory interface 48 (*data link*) allows data or content to be transferred between processor 52 (*programmable processor*) and low-level controller (*the device controller*), which would be critical to the functions of the storage system drive.



Ex. 1010, Fig. 1 (annotations in color)

4. Limitation 9[c]: “a printed circuit board to mount the solid-state memory, the device memory, the device controller, the programmable processor, and the data link; and”

290. In my opinion, Meir in view of Salessi and Chou renders obvious 9[c]: “a printed circuit board to mount the solid-state memory, the device memory, the device controller, the programmable processor, and the data link.”

291. Meir discloses *solid-state memory* (see Section VIII.C.2), *device memory* (see Section VIII.C.3), a *device controller* (see Section VIII.C.5), and a *programmable processor* (see Section VIII.C.4), and a *data link* (see Section IX.C.3).

292. To the extent that neither Meir nor Salessi explicitly discloses a *printed circuit board to mount* those components, in my opinion, this would have been obvious in view of Chou.

293. Chou describes a printed circuit board assembly (PCBA) (*printed circuit board*). Ex. 1012, [0006]. A person of ordinary skill in the art would regularly use PCBAs. Chou also describes that “[o]ne or more memory modules and other electronic components 122 are on the PCBA 112.” *Id.*, [0033]. In my opinion, a person of ordinary skill in the art would then understand that the *solid-state memories*, *device memories*, *device controllers*, and *programmable processors* are all either “memory modules” or “other electronic components” that would be on

the PCBA. Further, this would align with the common practice of a person of ordinary skill in the art.

294. In my opinion, such a person of ordinary skill in the art also would have combined the Meir/Salessi data storage drive with Chou's PCBA because this would have furthered Meir's goal of "maximiz[ing] a throughput of the execution [and] minimiz[ing] a latency of the execution" because having the different components on a single PCB would reduce the amount of time and distance that data has to travel from one component to another. Ex. 1010, [0042]; *see* Sections IX.A-B. In order to have a data storage system (like Meir/Salessi) that is composed of a solid-state memory, a device memory, a device controller, and a programmable processor, it is my opinion that a person of ordinary skill in the art would have understood that a physical means was required to connect these electronic components, and a printed circuit board would have been the most conventional and routine means used for mounting the components.

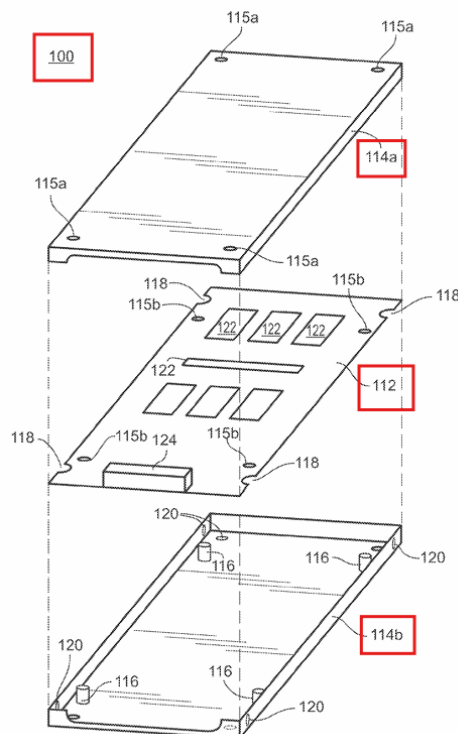
5. Limitation 9[d]: "a case to mount the printed circuit board and the communication interface."

295. In my opinion, Meir in view of Salessi and Chou renders obvious 9[d]: "a case to mount the printed circuit board and the communication interface."

296. Meir describes a *communication interface* (*see* Section IX.C.2). To the extent that neither Meir nor Salessi explicitly discloses *a case to mount the printed*

circuit board and the communication interface, in my opinion this would have been obvious in view of Chou.

297. Like Meir, Chou describes a solid-state drive (SSD) that includes a printed circuit board assembly (“PCBA”) (*printed circuit board*) and a “housing” (*case*). Ex. 1012, [0025]. The “housing” (*case*) supports the “PCBA” (*printed circuit board*) because “[t]o assemble the SSD 100, the PCBA 112 is placed inside the upper and lower covers 114a and 114b.” *Id.*, [0034], FIG. 2. Such components would be well known to a person of ordinary skill in the art.



Chou, Fig. 2

298. In my opinion, a person of ordinary skill in the art would have combined the Meir/Salessi data storage system with Chou's housing because a housing was a conventional structure for mounting and protecting SSD components. Chou teaches that the PCBA is placed inside upper and lower covers, which a person of ordinary skill in the art would have understood to mechanically support the PCB and help maintain the position of the mounted components and communication interface. The housing would also protect the PCB and internal SSD components from physical stress, movement, external environmental conditions, and external signal interference. By shielding against external signal interference, the housing would help improve signal integrity and reduce the likelihood of communication errors, retries, or retransmissions, thereby supporting reliable and performant operation. Using such a housing would therefore have been a routine and predictable way to improve the mechanical stability, durability, reliability, and performance of Meir's SSD. Ex. 1010, [0042]; *see* Sections IX.A-B. Such a combination would align with the common practices of such a person of ordinary skill in the art, because improving mechanical stability and reducing the likelihood of errors and retransmission is a widely applicable goal.

X. OPINIONS ON GROUND 3: MEIR IN VIEW OF SALESSI AND LOVINGER WOULD HAVE RENDERED OBVIOUS CLAIM 19

A. Motivation to Combine

299. In my opinion, a person of ordinary skill in the art would have been motivated to modify Meir-Salessi's SSD to receive a write command that includes the policy as taught by Lovinger.

300. Such a modification would enhance the goals of Meir. One of Meir's goals is to make policies configurable. Ex. 1010, [0011]. Lovinger teaches transmitting a storage request that include information associated with the storage policy. Ex. 1015, [0040], [0043]. A person of ordinary skill in the art would understand that configuring Meir and Salessi's storage system to include a storage policy in the storage request would increase the configurability of the policy because the host can configure the policy before sending it as part of the request.

301. Lovinger's approach is a natural extension of Meir's configurable policy. Lovinger teaches that including the policy with the storage request enables a client "to store content according to the retention policy without reprogramming device firmware." *Id.*, [0041]. A person of ordinary skill in the art improving on the system of Meir would have looked at ways to configure the policy without reprogramming device firmware.

302. Such a modification would provide more flexibility, particularly as to the compression or encryption algorithm used. Indeed, Salessi teaches that "[t]he

algorithm used by the application layer may be communicated to the DSP by the host CPU.” Ex. 1011, [0050]. A person of ordinary skill in the art would have looked at known ways to communicate the Meir-Salessi compression and encryption policy from the host to the storage system as taught in Lovinger.

303. Including the policy with the storage request as taught by Lovinger was a well-known design option. By 2016, this approach was a common option among multiple specifications. For example, the Open-Channel SSD Interface Rev. 1.2 specifies a write command that includes the content to be stored as well as a per-write policy word that instructs the device on the recording method that should be performed. *Id.*; Ex. 1036.

B. Reasonable Expectation of Success

304. In my opinion, a person of ordinary skill in the art would have been able to modify the Meir-Salessi’s system to incorporate the policy in a storage request as taught by Lovinger with a reasonable expectation of success.

305. The storage devices of Meir, Salessi, and Lovinger have similar structures and components. Ex. 1010, FIG.1; Ex. 1011, Figs. 2, 4, 7, Ex. 1015, [0029], [0037]-[0040], [0069], Figs. 1, 6. Meir, Salessi, and Lovinger’s storage devices each include memory controller or processing unit that interfaces with the solid-state memory. Ex. 1010, [0031] (“SSD 24 comprises multiple low-level controllers 44, each low-level controller assigned to a group of one or more memory

units 40.”), [0033] (“[h]igh-level controller 36 comprises a memory interface 48 for communicating with memory units 40 (in the present example via low-level controllers 44)”; Ex. 1011 [0047] (“[t]he SSD processing unit 240 includes...a hardware engine 420 [and] there may be a separate hardware engine 420 for each flash channel”), [0055] (“the microcontroller 410 may be connected to a plurality of hardware engines 420.”); Ex. 1015, [0069] (“the computing device 600 may include at least one processing unit 602 [and] data storage devices”).

306. Given the similar structures, the modification would simply be a change of the storage request flow. Indeed, Meir includes “memory access commands (e.g., read and write commands)” and storage policies including “redundancy rules.” Ex. 1010, [0030], [0033], [0038]. Further, Lovinger teaches sending the policy as part of the memory access command. Ex. 1015, [0040]. Updating the storage request flow would have been well within the skill set of a person of ordinary skill in the art at the relevant time.

C. Claim 19

1. Claim 19: The method of claim 17 wherein the storage request comprises the storage policy

307. In my opinion, Meir in view of Salessi and Lovinger renders obvious this limitation.

308. Meir discloses *a storage request*. See Section VIII.E.2. Meir and Salessi disclose *a storage policy*. Section VIII.C.7. Although Meir and Salessi do

not explicitly disclose a *storage request* that *comprises a storage policy*, this would have been obvious in view of Lovinger.

309. Lovinger describes that “the client 110 may store content 112 on a server 120.” Ex. 1015, [0029]. A person of ordinary skill in the art would have understood that storing content 112 requires the client to send a *storage request* that the storage device can execute to store the content 112. Lovinger further teaches a retention policy which is a *storage policy* because “the content is written to the one or more storage devices based on the retention policy.” Ex. 1015, [0003]; *see also id.* [0029] (“the storage attributes and capabilities of the one or more storage devices utilized by the server 120 may include retention policies”).

310. Lovinger discloses *storage request* that *comprises a storage policy*: “the client 110 may transmit the content 112 to the server 120 with information corresponding to the retention policy 130.” *Id.*, [0040]; *see also id.*, FIG. 1. Specifically, “when the content and retention policy have been received, the server 120 writes the content 112 to one or more storage devices associated with the server 120 according to the retention policy 130.” *Id.*, [0040].

311. As discussed, a person of ordinary skill in the art would have been motivated to modify the Meir-Salessi system to include the storage policy in the storage request as taught by Lovinger with a reasonable expectation of success. *See* Section X.A-B.

XI. CONCLUSION

312. For all the reasons stated above, it is my opinion that the challenged claims of the '715 patent are unpatentable.

313. This declaration and my opinions are made to the best of my knowledge and understanding, and based on the material available to me at the time of signing this declaration. All statements made herein of my own knowledge are true, and all statements made on information and belief are believed to be true. Further, I am aware that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. § 1001. I declare under penalty of perjury that the foregoing is true and correct.

Declaration of Heiner Litz, Ph.D. in Support of
Petition for *Inter Partes* Review USP No. 12,265,715

Executed this 05 day of 08 2026 in Heidelberg, Germany.