

**IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
AUSTIN DIVISION**

CONCURRENT VENTURES, LLC AND
XTREAMEDGE, INC.,

Plaintiffs,

v.

ADVANCED MICRO DEVICES, INC. AND
PENSANDO SYSTEMS, INC.,

Defendants.

Civil Action No. 1:24-cv-335-ADA

**Defendants Advanced Micro Devices, Inc. and Pensando Systems, Inc.'s
Opening Claim Construction Brief**

Table of Contents

	<u>Page</u>
Table of Authorities	ii
Table of Exhibits.....	iii
I. Level of Ordinary Skill in the Art.....	1
II. U.S. Patent No. 8,924,596 (“the ’596 patent”)	2
A. “queue implemented in hardware” (Claims 1, 7, and 13).....	2
B. “register” (Claims 1, 7, 13).....	5
III. U.S. Patent No. 9,529,767 (“the ’767 patent”)	10
A. “abstraction protocol” (Claims 1, 2, 4, 6 and 16)	10
B. “storage media device interface” (Claims 1 and 16)	13
IV. U.S. Patent No. 10,873,753 (“the ’753 patent”)	15
A. “interleaved memory” (Claims 4, 7, 15).....	15
B. “swapped memory over a specified address range” (Claims 4, 7, 15)	16
C. “one or more swappable pods or cards” / “the plurality of swappable pods or cards” (Claims 1-6).....	18
V. U.S. Patent No. 10,985,943 (“the ’943 patent”)	19
A. “programmable logic device” (Claims 1, 10)	19
VI. U.S. Patent No. 10,944,634 (“the ’634 patent”)	26
A. “geographical area” (Claims 1, 8, and 15).....	26
B. “network connected device” (Claims 16 and 20)	29
VII. Conclusion	30

Table of Authorities

	Page(s)
<u>Cases</u>	
<i>Bicon, Inc. v. Straumann Co.</i> 441 F.3d 945 (Fed. Cir. 2006).....	3, 16
<i>Egenera v. Cisco Sys., Inc.</i> 972 F.3d 1367 (Fed. Cir. 2020).....	10
<i>Eon Corp. IP Holdings v. Silver Spring Networks</i> 815 F.3d 1314 (Fed. Cir. 2016).....	4, 18
<i>Intel Corp. v. Qualcomm Inc.</i> 21 F.4th 801 (Fed. Cir. 2021)	3
<i>Mosaic Brands, Inc. v. Ridge Wallet LLC</i> 55 F.4th 1354 (Fed. Cir. 2022)	7
<i>Nautilus, Inc. v. Biosig Instruments, Inc.</i> 572 U.S. 898 (2014).....	5, 10, 12
<i>Thorner v. Sony Comput. Ent. Am. L.L.C.</i> 669 F.3d 1362 (Fed. Cir. 2012).....	5
<i>Verizon Servs. Corp. v. Vonage Holdings Corp.</i> 503 F.3d 1295, 1308 (Fed. Cir. 2007).....	7

Table of Exhibits

No.	Description	Production Number
1	Annotated U.S. Patent No. 8,924,596	
2	Declaration of R. Jacob Baker Regarding the '596 Patent ("Baker Decl.")	
3	Annotated excerpts from Plaintiffs' Suppl. Infringement Contentions for the '596 Patent (Dec. 20, 2024)	
4	Annotated excerpts from <i>Comprehensive Dictionary of Electrical Engineering</i> (2d ed. 2005)	AMD-CV-0008603-27
5	Annotated excerpts from Steven Schwarz and William Oldham, <i>Electrical Engineering: An Introduction</i> (2d ed. 1993)	AMD-CV-0008640-51
6	Annotated excerpts from Daniel Gajski, <i>Principles of Digital Design</i> (1997)	AMD-CV-0008678-85
7	Annotated excerpts from <i>Modern Dictionary of Electronics</i> (7th ed. 1999)	CV-AMD0017608-23
8	Annotated U.S. Patent No. 9,529,767	
9	Declaration of Darrell Long Regarding the '767 Patent ("Long Decl.")	
10	Annotated excerpts from <i>Dictionary of Computer Networking</i> (1st ed. 2011)	AMD-CV-0008714-16 AMD-CV-0008719-20
11	Annotated excerpts from <i>Dictionary of Computing</i> (6th ed. 2010)	AMD-CV-0008721-23
12	Annotated Jeff Kramer, <i>Is Abstraction the Key to Computing?</i> (2007)	AMD-CV-0008990-95
13	Annotated excerpts from <i>Dictionary of Computer and Internet Terms</i> (11th ed. 2013)	AMD-CV-0008711-13
14	Annotated excerpts from Minoli-Cordovana's <i>Authoritative Computer & Network Security Dictionary</i> (2006)	AMD-CV-0008731-33
15	Annotated excerpt from <i>Webster's New World Telecom Dictionary</i> (2008)	AMD-CV-0008741-43
16	Annotated U.S. Patent No. 10,873,753	
17	Declaration of Bill Lin Regarding the '753 and '943 Patents ("Lin Decl.")	

No.	Description	Production Number
18	Annotated excerpts from <i>Microsoft Computer Dictionary</i> (5th ed. 2002)	AMD-CV-0008706-08
19	Annotated excerpts from <i>Chambers Dictionary</i> (13th ed. 2014)	AMD-CV-0008702-04
20	Annotated U.S. Patent No. 10,985,943	
21	Annotated excerpts from the File History of U.S. Patent 10,985,943	CV-AMD-0000184, 275-289, 302-324
22	Annotated excerpts from Richard F. Tinder, <i>Engineering Digital Design</i> (2d ed. 2000)	AMD-CV-0008872, 888-89, 904, 921, 928-55, 960
23	Annotated excerpts from <i>JEDEC Dictionary of Terms for Solid-State Technology</i> (7th ed. Feb. 2018)	
24	Annotated excerpts from M. Morris Mano, <i>Digital Design</i> (2013)	AMD-CV-0008795, 801-809, 834-846
25	Annotated excerpts from Plaintiffs' Infringement Contentions for the '943 Patent (Oct. 11, 2024)	
26	Annotated U.S. Patent No. 10,944,634	
27	Declaration of Kevin Jeffay Regarding the '634 Patent ("Jeffay Decl.")	
28	Annotated excerpts from the File History of U.S. Patent No. 10,944,634	CV-AMD0000064-078, 091-100, 102-123, 136-147
29	Annotated U.S. Patent Application Publication No. 2008/0080458 to Cole (Apr. 3, 2008)	AMD-CV-0024255-82
30	Annotated excerpts from <i>Merriam-Webster's Collegiate Dictionary</i> (11th ed. 2007)	AMD-CV-0008686-89
31	Annotated excerpts from <i>New Oxford American Dictionary</i> (3d ed. 2010)	AMD-CV-0008690-93
32	Annotated excerpts from <i>The American Heritage Dictionary of the English Language</i> (5th ed. 2016)	AMD-CV-0008694-97

Defendants Advanced Micro Devices, Inc. and Pensando Systems, Inc. request construction of the five asserted patents. Defendants' constructions are supported by the intrinsic and extrinsic evidence, which includes expert opinions provided by Dr. Baker ('596 patent), Dr. Long ('767 patent), Dr. Lin ('753 and '943 patents), and Dr. Jeffay ('634 patent).

I. Level of Ordinary Skill in the Art

A person of ordinary skill in the art at the time of the alleged invention of the asserted patents would have at least (1) a bachelor's degree in the relevant major with two years of post-graduate experience in the field of the patent; or (2) a graduate degree in the relevant major with specialization in the field of the patent. A higher level of education or skill can make up for less experience, and vice-versa.

Patent	Field of Education	Field of the Patent
'596 Patent Baker Decl. ¶¶6-9	Electrical Engineering (EE), Computer Engineering (CE), or similar	Computer architecture, including the use of queues and multiple processing elements
'767 Patent Long Decl. ¶¶3-5	EE, CE, Computer Science (CS) or similar	Computer storage technology
'753 Patent Lin Decl. ¶¶40-41	EE, CE, or similar	Data processing and streaming architectures including user-defined hardware and software modules
'943 Patent Lin Decl. ¶¶4-5	EE, CE, or similar	PLDs and data processing systems
'634 Patent Jeffay Decl. ¶3	EE, CE, CS, or similar	Networking and telecommunications

II. U.S. Patent No. 8,924,596 (“the ’596 patent”)

A. “queue implemented in hardware” (Claims 1, 7, and 13)

Defendants’ Construction: “queue implemented in physical circuits without software intervention”; Alternatively, indefinite under Plaintiffs’ infringement theories	Plaintiffs’ Construction: Plain and ordinary meaning
--	---

The parties dispute whether a “queue implemented in hardware” includes queues implemented using software. Computers have hardware (e.g., physical circuitry) and software (e.g., instructions executed by hardware). Baker Decl. ¶14. When a queue is implemented in hardware, there are dedicated physical circuits for queue functions and management, including, e.g., a write pointer that indicates where the next entry should be written, a read pointer that indicates from where the next entry should be read, and storage elements for data in the queue. *Id.*, ¶15. On the other hand, when a queue is implemented in software, software maintains and uses a data structure in memory (e.g., RAM) as a queue, for example by tracking the start (write pointer) and end of the queue (read pointer), the size of the queue, etc. *Id.*, ¶16. Since both situations involve hardware to some extent—after all, software is executed on hardware processors—the distinguishing characteristic between a queue implemented in software, versus a queue implemented in hardware, is whether software is involved in that implementation.

Defendants’ construction gives meaning to the requirement that queues are “implemented in hardware,” rather than software, and is supported by the intrinsic record. Because this requirement excludes Plaintiffs’ infringement theories, Plaintiffs seek to avoid construction of this claim term, so they are free to accuse queues implemented by software in RAM.

Claim Language. The independent claims (1, 7, and 13) recite “an input queue implemented in hardware” and “an output queue implemented in hardware.” Therefore, the claims are clear that the input and output queues are implemented in hardware, rather than

software, which supports Defendants’ construction that these queues are implemented in physical circuits without software intervention.

Plaintiffs’ “plain and ordinary meaning” is anything but. As revealed by their infringement contentions, Plaintiffs accuse queues implemented using software in memory. *See* Ex. 3 at 28 (accusing a queue in DRAM); Baker Decl. ¶¶34. Therefore, Plaintiffs’ construction vitiates claim language requiring the queues be “implemented in hardware.” *See Bicon, Inc. v. Straumann Co.*, 441 F.3d 945, 950 (Fed. Cir. 2006) (“[C]laims are interpreted with an eye toward giving effect to all terms in the claim.”); *Intel Corp. v. Qualcomm Inc.*, 21 F.4th 801, 809-10 (Fed. Cir. 2021) (“It is highly disfavored to construe terms in a way that renders them void, meaningless, or superfluous.”) (citation omitted). Under Plaintiffs’ supposed “plain and ordinary meaning,” there would be no distinction between a “queue implemented in hardware” versus, simply, a “queue.”

Specification. Figures 3-5 differentiate between components implemented in hardware, such as the input and output queues, and software running on processing elements.

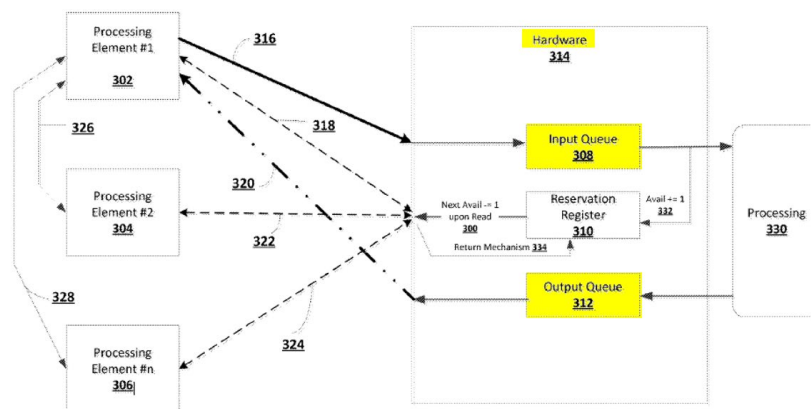


FIG. 3

’596 pat., Fig. 3 (annotated). The patent describes “an input queue 308 implemented in hardware 314” and “an output queue 312 also implemented in hardware” while simultaneously describing software executed on the processing elements. *Id.*, 5:3-45 (describing “computer

storage storing instructions” “executed by one of the additional processing elements”); *see also id.*, 3:11-12 (“The invention considers the case where software must issue commands to a hardware module.”), 4:55-57.

While discussing various features, the ’596 patent distinguishes between hardware and software. The specification emphasizes the advantage of functionality “handled in hardware” because “[t]here is no need for the software....” *Id.*, 4:14-19¹. The specification also describes hardware performing various functions automatically, “without software intervention.” *Id.*, 4:43-46 (“The hardware will then automatically, without software intervention, update the reserved free space when the command is actually removed from the queue for consumption.”).

Therefore, based on the intrinsic evidence, a queue implemented in hardware is implemented in physical circuits without software intervention.

Plaintiffs’ Construction. Plaintiffs’ construction provides no guidance as to what is meant by “implemented in hardware.” Baker Decl. ¶33. Rather, Plaintiffs’ construction only highlights the uncertainty—particularly in view of Plaintiffs’ infringement theories—as to whether a “queue implemented in hardware” includes queues implemented using software. The parties have a clear dispute, and Plaintiffs’ “plain and ordinary” meaning does not resolve that dispute. Rejection of Defendants’ proposed construction would therefore improperly “leave [the] question of claim scope unanswered.” *Eon Corp. IP Holdings v. Silver Spring Networks*, 815 F.3d 1314, 1318-19 (Fed. Cir. 2016) (reversing judgment premised in part on the district court’s erroneous decision “not to construe the terms” at issue).

Plaintiffs’ broad interpretation is at odds with the specification. “The words of a claim are generally given their ordinary and customary meanings as understood by a person of ordinary

¹ All emphasis is added in this brief unless stated otherwise.

skill in the art when read in the context of the specification and prosecution history.” *Thorner v. Sony Comput. Ent. Am. L.L.C.*, 669 F.3d 1362, 1365 (Fed. Cir. 2012). Nowhere does the specification describe queues managed or controlled by software. Nor would that make sense because the claim states that the queues are “implemented in hardware.”

Moreover, Plaintiffs’ supposed “plain and ordinary meaning” renders the claims indefinite when viewed in the context of Plaintiffs’ infringement theories. By alleging that a “queue implemented in hardware” can include queues implemented using software, Plaintiffs remove the boundaries created by this claim language, and a POSITA would not be able to discern, with reasonable certainty, when a queue is implemented in hardware, and when it is not. *See Nautilus, Inc. v. Biosig Instruments, Inc.*, 572 U.S. 898, 910 (2014); Baker Decl. ¶¶37-38.

Defendants’ construction is consistent with the intrinsic record and well-understood meaning of a “queue implemented in hardware” as a queue implemented in physical circuits without software intervention and should be adopted given the dispute between the parties.

B. “register” (Claims 1, 7, 13)

Defendants’ Construction: “set of flip-flops that store a set of related bits”	Plaintiffs’ Construction: Plain and ordinary meaning, that is “a memory location within a microprocessor used to store values and external memory addresses”
---	---

The ’596 patent aims to synchronize task processing across multiple processing elements. ’596 pat., 1:9-13. Because command queues are used to distribute commands but can only hold a finite number of commands, synchronization is used to prevent overfilling the queues. *Id.*, 1:22-32. “In the prior art, such synchronization is performed using software techniques” that stored a count of available space “in memory.” *Id.*, 1:25-27. The patent describes such techniques as “non-optimal” (’596 pat., 1:27-29) and instead “provides a shared counter resource (e.g., a register) in the hardware representing how much free space there is in the command

queue.” *Id.*, 3:30-34. The patent explains that a hardware register reduces “software overhead” compared with software approaches that stored a count of available space in memory. *Id.*, 4:11-19 (avoiding overhead for “a shared available space count variable in shared memory...”); Baker Decl. ¶¶20-22.

The parties dispute whether the claimed register must be a hardware register, or whether it can be satisfied by a memory location used for storing a variable, e.g., in RAM. The patent contrasts hardware registers and mere memory locations used for storing variables. The intrinsic record is clear: a register must be a register, implemented in hardware. Hardware registers were well known in the art to be basic circuit components implemented using flip-flops. The claimed register is a fixed hardware structure, in contrast to a memory location, which can be mapped to different portions of physical memory (RAM) at different times, depending on page table configurations and other factors. Baker Decl. ¶42. Plaintiffs’ construction is incorrect because it expands “register” to encompass a variable at a memory location—e.g., the prior art software techniques distinguished by the ’596 patent.

Claim Language. The claims establish two key points. First, the independent claims (1, 7, and 13) require that the register be implemented in hardware: “a reservation register implemented in hardware storing a value indicative of available space in said input queue[.]” ’596 pat., claims 1, 7, and 13. Second, some dependent claims (5, 11, and 17) recite auto-decrement functionality that is performed by the register, in contrast to memory locations that simply store data.

Specification. The specification confirms that the register is implemented in hardware. Regarding Figure 1, the specification states that “the present invention provides a shared counter resource (e.g. a register) in the hardware representing how much free space there is in the

command queue, accessible to one or more processing elements.” *Id.*, 3:30-34, Abstract. The ’596 patent consistently refers to the present invention as having a register implemented in hardware. *See Mosaic Brands, Inc. v. Ridge Wallet LLC*, 55 F.4th 1354, 1361-62 (Fed. Cir. 2022) (quoting *Verizon Servs. Corp. v. Vonage Holdings Corp.*, 503 F.3d 1295, 1308 (Fed. Cir. 2007) (“When a patent ... describes the features of the ‘present invention’ as a whole, this description functions as a disclaimer that limits the scope of the invention.”) (emphasis original)).

Consistent with claims 5, 11, and 17, the specification explains how the register’s auto-decrement functionality is performed by hardware: when software reads the reservation register, the “hardware automatically decrements the available free space by a preconfigured amount (e.g., 1) and returns the value of the free space immediately prior to the read/reservation.” ’596 pat., 3:43-47, Abstract. Hardware automatically performs this functionality when the register is read, in contrast to data in memory that does not change when read.

The specification emphasizes that its solution uses hardware to avoid the software overhead of sharing a variable (with the count of available space) in memory: “All arbitration and mutual exclusion tasks between multiple processing elements are handled in hardware. There is no need for the software overhead of multiple processing elements handling the synchronization of a shared available space count variable in shared memory or otherwise.” ’596 pat., 4:11-19.

The specification directly contradicts Plaintiffs’ proposed construction. The register must be implemented in hardware and cannot simply be a memory location used to store a value.

Extrinsic Evidence. Registers were basic, ubiquitous computer components that were known since the advent of computers to be a group of flip-flops that store a set of related bits. *See Ex. 4* at 583 (Comp. Dict. of Elec. Eng’g) (“register[:] a circuit formed from identical flip-

flops or latches and capable of storing several bits of data”); Baker Decl. ¶49. A POSITA would have recognized this as a term of art because undergraduate electrical and computer engineering textbooks teach implementing registers using flip-flops. Baker Decl. ¶50; Ex. 5 at 435 (“... byte containing N bits is stored in a set of N flip-flops. This set of flip-flops is then referred to as a register.”); Ex. 6 at 268 (“The simplest of the storage components is a register which can be thought of as a bitwise extension of a flip-flop. Each register consists of n flip-flops driving by a common clock[.]”).

Plaintiffs’ Construction. Plaintiffs ask for plain and ordinary meaning but propose a construction, tacitly admitting construction is needed. Plaintiffs’ construction is improper for numerous reasons and should be rejected. First, Plaintiffs’ construction broadens the claim to encompass the memory-based prior art that the ’596 patent distinguishes, by converting a register into simply “a memory location” “used to store values.” This is contradicted by the claims, which require a register “implemented in hardware[,]” rather than simply being a memory location. The specification also contradicts Plaintiffs’ construction, explaining how the count of available space for a queue is stored in a hardware register to avoid the software overhead of sharing a variable in memory. ’596 pat., 3:30-34, 4:11-19 (“All arbitration and mutual exclusion tasks ... handled in hardware” so “[t]here is no need for the software overhead of ... synchronization of a shared available space count variable in shared memory or otherwise.”). Plaintiffs’ construction transforms the claimed “register implemented in hardware” into what the specification distinguished: a variable, at a memory location, for storing a data value.

Second, Plaintiffs’ construction morphs a register into a mere memory location by over-emphasizing the perspective of a single microprocessor, which excludes the embodiments of the ’596 patent. Baker Decl. ¶¶52-53. For example, Plaintiffs’ construction requires a register to be

“a memory location within a microprocessor,” but Figure 2 depicts the reservation register outside of the processors. Likewise, Figures 1 and 3-5 depict an external reservation register outside the processing elements, which can be “disparate processors[.]” ’596 pat., 3:36-39.

Plaintiffs’ dictionary definitions likewise focus on registers within a microprocessor, which excludes embodiments of the ’596 patent. Webster’s Computer Dictionary refers to registers “within a microprocessor” that are used “while the microprocessor performs logical and arithmetic operations on them.” Baker Decl. ¶55. Likewise, the Microsoft Computer Dictionary definition is focused on registers “within a microprocessor.” Baker Decl. ¶56. Plaintiffs’ construction appears to misuse language from Microsoft’s definition that refers to a register as “[a] set of bits of high-speed memory....” *Id.* However, while a register might be considered a type of memory in a broader sense of storing data, that does not mean that all memory is a register. *Id.* A variable in RAM is not a register, yet it would be encompassed by Plaintiffs’ construction. *Id.* Microsoft’s definition never states that a register can be just any type of memory. Instead, it describes specific characteristics of a register: it is a set of bits, comprising high-speed memory—a reference to a register being a set of flip-flops—that is referred to in assembly language by a specific name. Plaintiffs’ construction omits these characteristics. *Id.*

Plaintiffs offer The Modern Dictionary of Electronics, but it supports Defendants’ construction. The first entry (“A short-term, fast-access circuit used to store bits or words in a CPU”) reinforces that “register” refers to a circuitry, rather than just any type of memory as Plaintiffs contend. Ex. 7 at 636; Baker Decl. ¶54. The seventh entry is also consistent with Defendants’ construction, stating that register refers to “Logic elements (gates, flip-flops, shift registers) that, taken together, store 4-, 8-, or 16-bit numbers.” Ex. 7 at 636; Baker Decl. ¶54.

Plaintiffs’ construction should be rejected because it contradicts the claims, the specification, and is not supported by their extrinsic evidence. Defendants’ construction is supported by intrinsic evidence and is consistent with the known meaning of “register.” “Register” should be construed to mean a “set of flip-flops that store a set of related bits.”

III. U.S. Patent No. 9,529,767 (“the ’767 patent”)

The ’767 patent relates to computer storage technology. A host (e.g., computer) is connected to storage media devices by a routable packet-switched network that includes master controllers and edge controllers. At least the master controllers and edge controllers communicate via an abstraction protocol.

A. “abstraction protocol” (Claims 1, 2, 4, 6 and 16)

Defendants’ Construction: Indefinite; Subject to § 112(f), claimed function: abstracting communication of commands between master controllers and edge controllers, corresponding structure: none	Plaintiffs’ Construction: Plain and ordinary meaning; Not subject to § 112(f)
--	--

“Abstraction protocol” does not connote structure and § 112(f) applies. Because the specification does not disclose a corresponding structure, “abstraction protocol” is indefinite. But even if § 112(f) does not apply, “abstraction protocol” is indefinite for failing to inform, with reasonable certainty, a POSITA of the scope of the claimed invention under *Nautilus*.

§ 112(f) applies. The presumption that a claim term not reciting “means” does not invoke § 112(f) is rebutted if the “claim term either fails to ‘recite sufficiently definite structure’ or recites ‘function without reciting sufficient structure for performing that function.’” *Egenera v. Cisco Sys., Inc.*, 972 F.3d 1367, 1373 (Fed. Cir. 2020) (citations omitted). The presumption is not strong, and it is unnecessary to find that the term is devoid of structure, but merely that there is insufficient structure. *Id.* Here, the presumption is rebutted because “abstraction protocol” does not connote known structure in the field of computer storage, and the claim does not recite

sufficient structure to perform the abstraction protocol's function. Long Decl. ¶¶10-20.

The '767 patent says the abstraction protocol is the software method part of its invention. '767 pat., 3:65-4:8 ("The invention is comprised of both hardware ... and software (method—abstraction protocol)."). But the specification fails to describe its structure—there are no algorithms, no flow charts, nor pseudocode. Long Decl. ¶¶16-17, 22. At most, the specification recites features or properties that the abstraction protocol should possess, without explaining how they are implemented. *See, e.g., id.*; '767 pat., 3:65-4:8 (reciting, *inter alia*, providing a full duplex protocol, supporting full command queuing, providing high-level access commands to storage media devices), 4:30-32 (supporting flow-control), 4:35-45 (being performance centric, supporting common high-level read and write access, and being link-agnostic), 4:55-56 (packet-switched routable fabric and error recovery), 8:47-48 (enabling communications between the master controller and the edge controller).

Nor was "abstraction protocol" a term known and used in the field. A "protocol" is a set of rules that define a procedure to follow when transmitting and receiving data between devices. Long Decl. ¶11; Ex. 10 at 343 (Dict. of Comp. Networking) (defining protocol as "the formal specification that defines the procedures to follow when transmitting and receiving data"); Ex. 11 at 270 (Dict. of Computing) (defining protocol as "the pre-agreed signals, codes, and rules to be used for data exchange between systems"). "Abstraction" is a well-known technique that involves simplifying, generalizing, or hiding underlying details in a design or system. Long Decl. ¶13. For example, transit maps are often an abstraction of a geographic map depicting the precise path of the subway. Long Decl. ¶¶13-14; Ex. 12 at 39-40. Plaintiffs' dictionary evidence is consistent. *See, e.g.,* Ex. 7 at 3 (Mod. Dict. of Elec.) ("[a] simplified description or

specification of a system that emphasizes some of the system's details or properties while suppressing others."); Long Decl. ¶15.

Combining "abstraction" with "protocol" suggests that an "abstraction protocol" is a simplified or generalized protocol. Long Decl. ¶16. But that meaning does not connote sufficient structure to a POSITA because abstraction is a highly-subjective design process. *Id.*, ¶23, ¶¶16-18. Even if a POSITA is told to create an abstraction protocol by abstracting an existing protocol, a POSITA would not immediately know what that structure would look like until they go through significant design efforts. *Id.*, ¶23. Because "abstraction protocol" does not connote sufficient structure, the presumption against applying § 112(f) is rebutted.

Claimed Function. Claim 1 recites "the master controller and the edge controller communicate via the abstraction protocol, the abstraction protocol comprising a full-duplex protocol supporting full command queuing for the at least one storage media device." '767 pat., 12:45-50. Claim 16 similarly recites that the master controllers and edge controllers communicate "via the abstraction protocol." *Id.*, 14:27-32. Thus, the claimed function is enabling communication of commands between master controllers and edge controllers for a storage media device. Long Decl. ¶¶18-20.

Corresponding Structure. As discussed above, the specification does not disclose any structure for abstraction protocol, let alone a structure that performs the claimed function for the abstraction protocol. Long Decl. ¶¶16-17, 21-23. Therefore, "abstraction protocol" is indefinite.

Indefinite under *Nautilus*. If § 112(f) does not apply, an abstraction protocol is a simplified or generalized storage protocol. Long Decl. ¶16. Even with that meaning, the term fails to inform, with reasonable certainty, a POSITA of the scope of the claimed invention because the claims and intrinsic evidence do not provide any objective criteria to determine

whether a storage protocol has been sufficiently simplified or generalized to constitute the claimed abstraction protocol. Long Decl. ¶24.

B. “storage media device interface” (Claims 1 and 16)

Defendants’ Construction: “a physical interface for transmitting and receiving commands and data between the edge controller and the storage media device”	Plaintiffs’ Construction: Plain and ordinary meaning, that is “interface that allows communication with at least one storage media device”
---	---

The claims and specification are clear that a storage media device interface is a particular type of interface – an interface to a storage media device. Plaintiffs’ construction is overbroad because it equally applies to the other claimed interfaces that do not recite storage media device.

Claim Language. Claims 1 and 16 distinguish several different interfaces. For example, a master controller interface is used to communicate with a host (’767 pat., 12:37-38) and a master controller control link interface is used to communicate with an edge controller (*id.*, 12:41-44). But a storage media device interface is the only claimed interface used to communicate with a storage media device. *Id.*, 12:44-45. This follows the plain and ordinary meaning of the term because “storage media device” is recited as a modifier to “interface.” Long Decl. ¶¶25-27.

Specification. The storage media device interface is the only type of interface used to connect an edge controller with a storage media device in the specification. Long Decl. ¶28. In Figure 4A, storage media device interfaces 418 and 420 directly connect to storage media devices 422 and 424 using SAS or SATA links 426 and 428. *Id.*, Fig. 4A and 5:38-40. This arrangement is also in Figure 4C. *Id.*, Fig. 4C and 6:33-35 (“one or more SMD interfaces 418’ & 420’ which communicate with one or more SMDs 422’ & 424’ via SATA and/or SAS links 426’ and 428’’”).

Extrinsic Evidence. Relevant dictionaries support Defendants’ construction. Numerous dictionaries define an “interface” as a structure that exists “between” two objects. Long Decl. ¶29; Ex. 13 at 261 (“interface[:] the connection between two systems through which information is exchange...”); Ex. 10 at 199-200 (“Interface[:] The point at which a connection is made between two hardware devices...”); Ex. 14 at 179 (“[a]n architectural concept that provides interconnections between physical blocks at reference points...”); Ex. 15 at 245 (“[t]he device or component that serves to physically and logically interconnect two other devices or systems and that enables their interoperation”). Each of these dictionary definitions reinforces that an “interface” is a structure connecting two relevant objects. Accordingly, any communication made through the interface should be understood as occurring between the two connected objects. Therefore, in the context of the ’767 patent, if the edge controller contains a “storage media device interface,” it should be interpreted as a structure that connects the edge controller and the storage media device, enabling communication between them.

Plaintiffs’ Construction. Plaintiffs’ construction eliminates the distinction between the storage media device interface and the other recited interfaces. In the claimed arrangement, the host’s storage commands pass through at least one master controller and edge controller through recited interfaces such as the master controller interface and master controller control link interfaces. Long Decl. ¶¶27-28; ’767 pat., claim 1. Even though those other interfaces are not the storage media device interface, they would satisfy Plaintiffs’ construction because each is an “interface that allows communication with at least one storage media device.” Moreover, Plaintiffs’ construction does nothing to define the claimed structure, which must be the interface that is used to connect the edge controller and storage media device.

IV. U.S. Patent No. 10,873,753 (“the ’753 patent”)

A. “interleaved memory” (Claims 4, 7, 15)

Defendants’ Construction: “an arrangement of different types of memory that are alternately accessed to achieve an average access bandwidth”	Plaintiffs’ Construction: “blended different types of memory”
---	--

The ’753 patent recites “interleaved memory” as one of several selectable arrangements for a plurality of types of memory. By alternating access to these different types of memory, interleaved memory achieves an access bandwidth that is the average of the different types of memory. ’753 pat., 8:35-40. The intrinsic evidence supports Defendants’ construction.

Claim Language. The claims recite a memory superposition with a plurality of types of memory; “interleaved memory” is an arrangement of memory superposition. Claims 4, 7, and 15 recite “memory superposition with selectable arrangement of the plurality of types of memory as ... interleaved memory.”

Specification. Similarly, the specification states that, “for memory superposition 140, two or more types of memory ... are made available for a data flow or other data processing operation.” ’753 pat., 7:1-9. The patent describes several arrangements for memory superposition, including “interleaved memory” which “achieve[s] an average access bandwidth” (*id.*, 8:35-40) because access to the memories are alternated (interleaved) between those different memories. Lin Decl. ¶47. Therefore, the specification confirms that “interleaved memory” is an arrangement of different types of memory that are alternately accessed to achieve an average access bandwidth.

The specification explains how interleaved memory results in bandwidth and power characteristics being blended among the different types of memory. ’753 pat., 8:35-40. When access is interleaved between different types of memory, access bandwidth becomes a blend (or

average) of the bandwidth for each type of memory. *Id.* Therefore, the bandwidth characteristic becomes blended. The specification uses “blend” to describe a blend of characteristics that results from interleaved memory, but it does not refer to a “blend” of different types of memory.

Plaintiffs’ Construction. Plaintiffs’ construction misreads the specification. Plaintiffs lift the word “blend” from the specification; but, as explained above, that was a discussion of blended characteristics—such as blended bandwidth—that result from interleaved memory, rather than a blend of memories themselves.

By redefining “interleaved memory” to be a blend of different types of memory, Plaintiffs render that term superfluous, effectively removing “interleaved memory” as one of the selective arrangements required by the claim. The claims separately recite a “memory superposition” with “a plurality of types of memory” selectively arranged in various ways, including as interleaved memory. Therefore, the claims already require different types of memory to be used together in a memory superposition. Lin Decl. ¶¶49-50. Plaintiffs’ construction redefines “interleaved memory” to be duplicative of other claim language, which is improper because “[c]laims are interpreted with an eye toward giving effect to all terms in the claim.” *Bicon, Inc. v. Straumann Co.*, 441 F.3d 945, 950-51 (Fed. Cir. 2006) (rejecting construction under which a portion of the claim language “has no role in the claim and thus is entirely superfluous”).

B. “swapped memory over a specified address range” (Claims 4, 7, 15)

Defendants’ Construction: “memory with a specified address range where information is moved between the memory and non-volatile storage”	Plaintiffs’ Construction: Plain and ordinary meaning
---	---

The claims and specification provide little guidance as to the meaning of “swapped memory,” instead relying on a POSITA’s knowledge of this term. Defendants’ construction

tracks the well-known meaning of this term of art. Claim construction is needed because this is a technical term with a specific meaning in the art, rather than a term that a lay juror would understand without construction, particularly since the patent provides no explanation.

Intrinsic Evidence. Claims 4, 7, and 15 list this term as one of the selectable arrangements of “memory superposition.” Similarly, the specification simply states that “swapped memory over a specified address range” is “another arrangement of memory superposition.” ’753 pat., 8:41-42, 14:25-31.

Extrinsic Evidence. “Swapped memory” is a term of art that a POSITA would have recognized because it has been commonly used, since Windows 95 or before, to refer to a particular memory arrangement. Lin Decl. ¶55. For example, the Microsoft Computing Dictionary explains that swapping “involves moving blocks of information in units known as pages between memory and disk [.]” Ex. 18 at 504. This technique expanded a computer’s memory capacity by saving the contents of memory pages into a “swap file” on non-volatile storage, such as a hard disk. Lin Decl. ¶56; Ex. 18 at 504 (definition of swap file). Similarly, the Chambers Dictionary explains that non-volatile storage is used for swapping memory, defining a swap file as “an area of the hard disk of a computer into which any information not immediately required may be moved.” Ex. 19 at 1570.

Given the ’753 patent’s absence of further explanation, a POSITA would have understood “swapped memory” based on its customary meaning in the art. Because the claims recite “swapped memory over a specified address range,” the customary meaning was a memory with a specified address range where information is moved between the memory and non-volatile storage. Lin Decl. ¶57.

Plaintiffs’ Construction. Plaintiffs’ assertion that the plain and ordinary meaning should control is belied by the fact that they have not identified any supposed plain and ordinary meaning. Rather, the ordinary and customary meaning in the art is reflected in Defendants’ construction. Rejection of Defendants’ construction would therefore leave the question of claim scope unanswered, as the parties would lack guidance on what “swapped memory” means, if not the ordinary and customary meaning in the art that is captured by Defendants’ construction. *See Eon Corp.*, 815 F.3d at 1319.

C. “one or more swappable pods or cards” / “the plurality of swappable pods or cards” (Claims 1-6)

Defendants’ Construction: Indefinite	Plaintiffs’ Construction: [one or more / plurality of] pods or cards that are removable, insertable, or replaceable
---	--

Claim 1 is indefinite because its internal inconsistencies make it unclear whether the claim can be satisfied by one swappable pod or card. Claim 1 recites “one or more swappable pods or cards.” Subsequently, claim 1 refers back to this term as “the plurality of swappable pods or cards.” Some dependent claims (2 and 4-6) recite “one or more,” while claim 3 recites both “the one or more” and “the plurality” of swappable pods or cards. The inconsistent use of plural and singular forms renders the claims indefinite because it is uncertain whether the claims can be satisfied by just one swappable pod or card.

The specification does not resolve the confusion, referring to “one or more swappable pods or cards,” but referring to the one or more of “the plurality of swappable pods or cards” in the discussion that immediately follows. ’753 pat., 1:25-35, Abstract.

V. U.S. Patent No. 10,985,943 (“the ’943 patent”)

A. “programmable logic device” (Claims 1, 10)

Defendants’ Construction: “an integrated circuit where the user can alter the connection pattern of the circuits”	Plaintiffs’ Construction: Plain and ordinary meaning, that is “a device with electronic circuitry capable of being configured or reconfigured”
--	---

The ’943 patent aims to overcome the limitations of software-based systems and fixed-function hardware. Dedicated hardware functions are faster than software, but developing new hardware for each application is expensive. ’943 pat., 1:8-21. The patent seeks to provide the speed of dedicated hardware while avoiding the need to design and manufacture a new hardware chip for each application (*id.*) by using a type of integrated circuit called a “programmable logic device (PLD)” (*id.*, 1:36-44). Unlike a conventional IC with hardware circuits that are fixed at manufacturing, PLDs have programmable circuits that users can alter to create custom functions in hardware. Lin Decl. ¶16, ¶¶7-9. As stated by the Applicant in the file history, “PLD is a term of art referring to a specific type of integrated circuit that has user programmable electronic circuits.” Ex. 21, CV-AMD000308.

The parties disagree whether “PLD” is a term of art that refers to a specific type of IC with programmable electronic circuits. All evidence—intrinsic and extrinsic—confirms Defendants’ construction, which makes clear that PLD refers to an IC with circuits that are programmable. In contrast, Plaintiffs’ construction broadens PLD to encompass devices where the software is programmable but the circuits are not, by redefining PLD to encompass “a device with electronic circuitry” (which would be any IC) where the device is “capable of being configured or reconfigured” to run different software.

Claim Language. The claims require a PLD to include “sandboxes” where the user can define the programmable circuits within. Independent claims 1 and 10 recite “a programmable

logic device (PLD)” with “one or more sandboxes with user-definable programmable electronic circuits of the PLD.” Therefore, the claims confirm Defendants’ position—that the PLD must include circuits that users can alter—while refuting Plaintiffs’ construction, which would expand “PLD” to include an IC where the user cannot define a single circuit inside. In other words, Plaintiffs’ construction is overly broad because it would expand PLD to encompass a device whose operation can be configured or reconfigured through software but whose circuits cannot be changed.

File History. During prosecution, the Applicant stated that “[p]rogrammable logic device, or PLD, is a term of art referring to a specific type of integrated circuit that has user programmable electronic circuits.” Ex. 21, CV-AMD000308-310. Claim 1 was amended to overcome prior art rejections by reciting that the sandboxes consisted of user-definable programmable electronic circuits inside a PLD. Ex. 21, CV-AMD000303.

Thus, the file history confirms two things. First, “PLD” is a term of art, not a generic term that could cover nearly any IC as Plaintiffs contend. Second, PLD refers to an IC with programmable circuits, as made clear in Defendants’ construction, rather than a fixed device whose circuits cannot be altered by the user, as encompassed by Plaintiffs’ construction.

Specification. The specification refers to “a PLD ... or other programmable device that can configure electronic circuitry.” ’943 pat., 4:30-47, 1:36-44, 3:8-13, 5:19-25. Therefore, the specification further confirms that users can configure the electronic circuits inside a PLD.²

² The specification ends with generic paragraphs about computer-readable code and other boilerplate. ’943 pat., 16:11-17:37. One paragraph has boilerplate regarding “components [that] may be described or claimed as ‘configured to’ or ‘configurable to’ perform a task[.]” *Id.*, 16:57-17:22. However, this is not relevant because the dispute is not about generic patent language for performing a task; it is about whether PLDs have circuits the user can configure.

Moreover, the specification explains how sandboxes inside an FPGA (a type of PLD) address the patent’s goal of providing the speed of dedicated hardware (’943 pat., 1:8-21): “[t]he use of FPGAs 122, and especially with the introduction of sandboxes 132, support the ‘hardware defined everything’ concept[.]” *Id.*, 13:14-17. Therefore, the patent confirms that user-definable programmable circuits inside the sandboxes allow the user to define “hardware” functionality inside the PLD (*id.*), rather than simply software as Plaintiffs’ construction would allow.

The specification discusses hardware modules and software modules. ’943 pat., Fig. 2. The ’943 patent claims user-definable hardware modules (PLDs), while the also-asserted ’753 patent—which shares an identical specification—claims both “user-definable hardware modules” and “user-definable software modules.” *See* ’753 pat., claim 1. The shared specification also states that “processors 136 are available for software modules 112.” ’943 pat., 6:46-6:64. Therefore, the patentee knew how to describe and claim software modules, but specifically claimed PLDs with user-definable programmable electronic circuits in the ’943 patent.

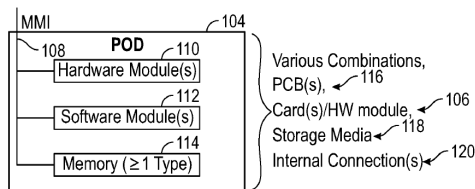


FIG. 2

The key requirement—reiterated in the claims, file history, and specification—is that a PLD must include circuits that the user can alter. Whether the PLD also runs software or not is irrelevant. For example, because the circuits of the PLD can be configured to perform any hardware function, those circuits can be configured into a processor. ’943 pat., 7:15-20 (suggesting possibility of a “softcore” processor); Lin Decl. ¶26. As the specification notes, “in some versions” a PLD can “includ[e] one or more processors or controllers.” ’943 pat., 4:27-39, 3:8-13 (PLD may “optionally includ[e] processors”). Alternatively, PLDs can include

processors in addition to programmable circuits. Lin Decl. ¶26. But regardless of whether software is executed or not, the defining feature of a PLD is the presence of circuits that the user can alter. If the user turns those circuits into a processor for running software, it does not change the fact that the PLD included circuits that the user altered.

Extrinsic Evidence. Both parties' extrinsic evidence demonstrate that "PLD" has been used for decades to refer to devices where the user can alter the connection pattern of the circuits. First, PLD is defined by JEDEC, a prominent IC standards organization, as "[a]n integrated circuit consisting of an array of logic elements whose interconnection pattern can be programmed (either mask-programmed or user-programmed) to perform specific logic functions." Ex. 23 at 171 (entry for "programmable logic device (PLD)"). JEDEC further explains that programming or configuring in the context of PLDs refers to "modify[ing] the logic connections inside a programmable device, causing its functions to change." *Id.*, (entry for "program; configure").

A POSITA would have recognized "PLD" as a term of art because it is taught as a basic concept in digital circuit design, as illustrated by several textbooks. For example, Mano's Digital Design is an undergraduate textbook (Ex. 24 at ix; Lin Decl. ¶34) that uses exemplary PLDs to show how users altered the connection pattern of the circuits:

A PLD is an integrated circuit with internal logic gates connected through electronic paths that behave similarly to fuses. In the original state of the device, all the fuses are intact. Programming the device involves blowing those fuses along the paths that must be removed in order to obtain the particular configuration of the desired logic function.

Ex. 24 at 299-230. Figure 7.17 shows how a user blows fuses in a grid of wires to break electrical connections, similar to a home fuse box, thereby altering the connection pattern of circuits (highlighted below for output *x*) to form custom logic functions in hardware.

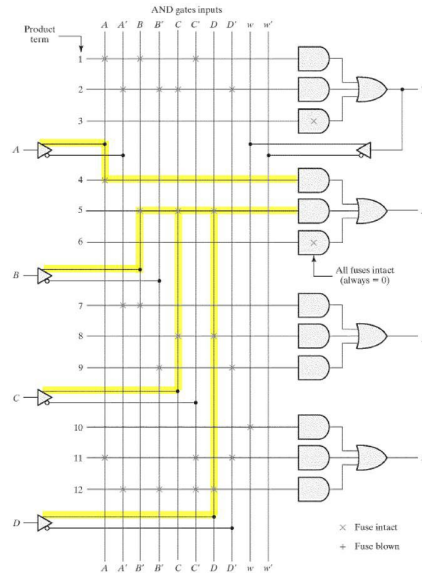


FIGURE 7.17
Fuse map for PAL as specified in Table 7.6

Ex. 24 at 328; Lin Decl. ¶13. Mano further explains how a type of PLD, called an FPGA, was “configured by downloading a stream of bits into the device to configure transmission gates to establish the internal connectivity required by a specified logic function[.]” *Id.* at 230. This FPGA configuration process altered the connection pattern of the circuits inside the FPGA by using “programmable interconnect points” that connect a “grid of two layers of metal segments.” *Id.* at 336, Fig. 7.23; *see also* Lin Decl. ¶36 (“FPGA ... An array of logic cells placed in an infrastructure of interconnections....”) (quoting Penguin Dict. of Elecs.).

Tinder’s Engineering Digital Design, another first-year textbook on the “basics and background fundamentals” of digital circuit design, explains how PLDs include combinational array logic and encompass “ROMs, PLAs, PALs or FPGAs, etc.” Ex. 22 at 24, xix-xx (“first course of study”); Lin Decl. ¶33. Tinder illustrates combinational array logic circuits using an FPGA with “a matrix of programmable interconnections” (Ex. 22 at 320), where the user alters the connection pattern of these circuits by blowing “Cross Fuse” points at the intersections between wires, which fuses the wires together and creates a connection in the circuit. *Id.* at 321:

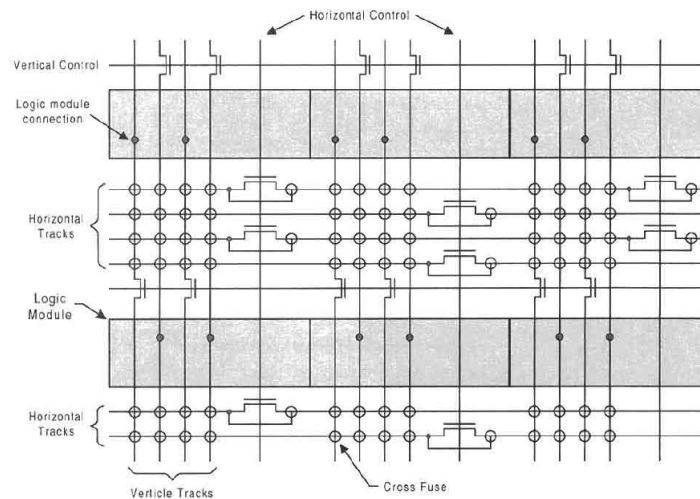
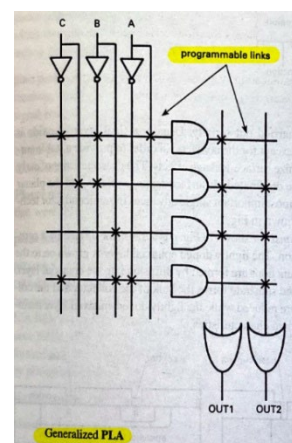
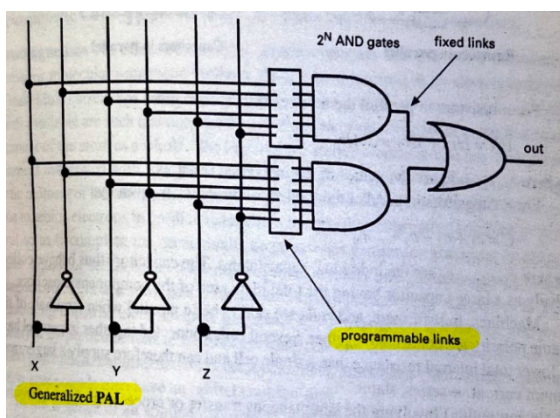


FIGURE 7.25
Interconnect architecture for ACT-1 family of FPGAs. Source: *ACT Family Field Programmable Gate Array Databook*; Actel Corp., Sunnyvale, CA, 1991.

In a chapter dedicated to PLDs, Tinder explains how electronic circuits, rather than software, are programmed by the user with examples showing how the logic connections are changed in various types of PLDs. *See, e.g., id.* at 299 (Fig. 7.3), 303-304 (Figs. 7.6-7.7), 306 (Fig. 7.10).

Tellingly, Plaintiffs’ evidence supports Defendants’ construction. Plaintiffs cite the Penguin Dictionary of Electronics, which confirms that “the designer must program the function required” into the PLD, with examples including PALs and PLAs. CV-AMD0017593. Penguin includes diagrams showing how, in those PLDs, the user alters the connection pattern of the circuits using “programmable links.” Lin Decl. ¶35:



Plaintiffs also cite the Electrical Engineering Dictionary. But this definition confirms that PLDs are ICs where the user (also referred to as the “designer”) programmed circuits. Lin Decl. ¶36 (“programmable logic device (PLD) an integrated circuit able to implement ... digital functions defined by the designer and programmed into this circuit.”). Similarly, Plaintiffs’ other evidence confirms that PLDs were programmed by configuring the connection patterns of their circuits. *Id.*, (“You ... program the chip to make the necessary connections.... ... The logic cells and interconnection scheme, in turn, determine the design of the input and output circuits as well as the programming scheme.”) (quoting EDACafe, *PROGRAMMABLE ASICS*); (“PLDs ... are used when ... programmable interconnect is required.”) (quoting Adsantec, *All About ASIC*). Both parties’ evidence confirms that “PLD” was a term of art that referred to a specific type of IC where the user can alter the connection pattern of the circuits.

Plaintiffs’ Construction. Contradicting the claims and file history, Plaintiffs expand “PLD” from a term of art referring to a specific type of IC with user-programmable electronic circuits into a general term that encompasses ICs with fixed circuits that users cannot change. In Plaintiffs’ construction, a device can be a PLD, even if its circuits are fixed and cannot be changed by the user, so long as it has electronic circuitry (as all ICs do) and is “capable of being configured or reconfigured” to run software.³ This would encompass a broad range of ICs, including conventional PC processors going back decades. Lin Decl. ¶37-38. Plaintiffs’ construction breaks the explicit connection in the ’943 patent between PLDs and configurable circuits.

However, the ’943 patent specification is clear: electronic circuits must be configurable in a PLD. ’943 pat., 4:30-47 (“a PLD ... or other programmable device that can configure

³ See Ex. 25 at 2-9, 15, 27 (Pls.’ ’943 Patent Infringement Contentions) (accusing software on P4 and ARM CPU cores in a network-processing chip).

electronic circuitry”). The claims and file history also require circuits to be configurable, consistent with the known meaning of PLD, which refers to a family of ICs where the user can alter the connection pattern of the circuits. It is also the point of the patent, which uses “sandboxes” in the PLD, where the user can alter circuits to perform custom functions (’943 pat., 5:19-29, Claim 1), thereby accomplishing the patent’s stated goal of providing the speed of dedicated hardware without having to design and manufacture a new IC. *Id.*, 1:8-21.

Plaintiffs’ construction renders the claim indefinite because it redefines “PLD” to encompass nearly any IC. Therefore, a POSITA would not be able to discern the boundaries of a PLD, under Plaintiffs’ construction, with reasonable certainty.

VI. U.S. Patent No. 10,944,634 (“the ’634 patent”)

The ’634 patent is titled “Optimization for network connections.” Parameter values are collected for determination of bandwidth of network connections. A tuner server determines whether a next network connection matches a geographical area of a past network connection. The next network connection is initiated based on the determination at the tuner server, with a transmission bandwidth based on the parameter values for the past network connection.

A. “geographical area” (Claims 1, 8, and 15)

Defendants’ Construction: physical place such as an apartment complex, neighborhood, city, or state	Plaintiffs’ Construction: Plain and ordinary meaning
--	---

The proper construction of “geographical area” is a “physical place such as an apartment complex, neighborhood, city, or state.” Plaintiffs contend that an IP address is equivalent to a geographical area, but as explained below, this is incorrect because an IP address is not a physical place. Jeffay Decl. ¶6.

The term “geographical area” is recited in all three independent claims, which require a determination that a “next network connection from the first endpoint device to a second endpoint device matches a geographical area of a past network connection.” ’634 pat., 14:6-8 & 60-62, 15:48-16:2. As explained below, the “geographical area” of a network connection between two endpoints is a physical place, and supports Defendants’ construction.

The specification states that a “network connection” can be in a “geographic area (such as an apartment complex, neighborhood, or city to city or state to state network connection),” where the network connection spans two endpoints. *Id.*, 2:52-57. This means a “network connection” can span a “geographic area” within an “apartment complex” (e.g., both endpoints are within the same complex), a “neighborhood” (e.g., both endpoints are within the same neighborhood), from “city to city” (e.g., from San Francisco to Boston), or from “state to state” (e.g., from California to Massachusetts). These examples of a “geographic area” are all physical places, e.g., an apartment complex, neighborhood, area between cities, and area between states, which supports Defendants’ construction. *See id.*⁴ These examples also support the examples of a “geographical area” identified in Defendants’ construction (i.e., “apartment complex, neighborhood, city, or state”).

The patent’s prosecution history also supports the construction. In the Patent Office’s September 3, 2019 office action, the examiner found that the “geographical area” limitation was disclosed by the Cole prior art. Ex. 28 at CV-AMD0000066-68. Reading Cole, a POSITA would have recognized that Cole’s connection manager is operable to manage connections in Internet Protocol (IP) networks. Jeffay Decl. ¶12. This is because Cole explicitly discloses

⁴ The patent uses the term “geographic area” at 2:54 and “geographical area” everywhere else but draws no distinction between the two.

examples that involve IP network connections. First, Cole discloses placing a voice-over-IP (VOIP) call, which clearly requires an IP connection between IP addresses. Ex. 29 at ¶9, ¶104; Jeffay Decl. ¶12. Second, Cole discloses performing connectivity tests such as an HTTP test for web browsing. Ex. 29 at ¶107. It was well known that HTTP connections are created using the IP protocol. Jeffay Decl. ¶12. Yet, the examiner did not cite to Cole’s disclosure of IP addresses as teaching geographical areas. *See* Ex. 28 at CV-AMD0000066-68. In the remarks to its response, the Applicant stated, “Cole is silent as to geographical area.” Ex. 28 at CV-AMD0000098. The Applicant could not have made this statement (while satisfying the Applicant’s duty of candor to the patent office) had it actually thought that an IP address is equivalent to the geographical area.

Then, in the Patent Office’s March 6, 2020 final rejection, the examiner explained the “geographical area” was disclosed by the “Location” parameter identified in Cole, where the “Exemplar Parameter Values” in Cole were all physical places, i.e., a “home, office, or public environment.” Ex. 28 at CV-AMD0000106. In Cole’s specification, a “public environment” is referred to as a physical place, e.g., “a coffee shop, a library, a home airport, various destination airports, nearby homes or businesses, etc.” *See* Ex. 29 at ¶51 (“During use, the mobile device 110 may be employed in many different public environments where wireless networks 130, 140, 150 are detectable, such as a coffee shop, a library, a home airport, various destination airports, nearby homes or businesses, etc.”) (underlining added). Elsewhere in Cole, the “location” is referred to as a physical place as well, e.g., a “university or home airport,” an “airplane,” or a “country.” *See* Ex. 29 at ¶52 (“For example, the user may specify a general location category of public, with a more detailed subcategory designation of university or home airport for the location.”), ¶63 (“Yet another situation where a fast connect is useful is upon exiting an airplane

or other location that bans the use of RF networks.”) (underlining added); ¶141 (“For example, the user may be traveling to a different location or *country* outside the reach of the currently subscribed services.”); *see also id.* at ¶142. In the July 31, 2020 response to the rejection, the Applicant did not contest the examiner’s finding that Cole disclosed the “geographical area” limitation of the claims. Ex. 28 at CV-AMD0000143-44.

The construction of “geographical area” as a physical place is also supported by dictionary definitions. Merriam-Webster defines “area” as “a level piece of ground” or a “geographic region”; “geographical” as “of or relating to geography” or “belonging to or characteristic of a particular region”; and “geography” as a “science that deals with the description, distribution, and interaction of the diverse physical, biological, and cultural features of the earth’s surface.” Ex. 30 at 65, 523. New Oxford American defines “area” as a “region or part of a town, a country, or the world” or “the extent or measurement of a surface or piece of land”; “geographical” as “of or relating to geography”; and “geography” as “the nature and relative arrangement of places and physical features.” Ex. 31 at 83, 725. American Heritage defines “area” as a “roughly bounded part of the space on a surface; a region” or a “surface, especially an open, unoccupied piece of ground”; “geographical” as “[o]f or relating to geography” or “[c]oncerning the topography of a specific region”; and “geography” as the “physical characteristics, especially the surface features, of an area.” Ex. 32 at 94, 735.

All these dictionary definitions—with their repeated references to terms like “physical,” “surface,” “region,” “ground,” and “land,” among other things—support the construction of “geographical area” as a physical place.

B. “network connected device” (Claims 16 and 20)

The “network connected device” of claims 16 and 20 is indefinite because it is circular. In particular, claims 16 and 20 disclose the “network connected device of claim 15.” Claim 15,

meanwhile, recites that the network connected device is “acting as a tuner server.” ’634 pat., 15:46-47. Claim 16 requires the “network connected device” to “send the parameter values for determination of bandwidth, from the first endpoint device, to the tuner server,” i.e., to send parameter values to itself, which is circular. *Id.*, 16:11. The claim also requires that the network connected device “send, from the first endpoint device to the tuner server, connection information for the next network connection” and “receive, at the first endpoint device, from the tuner server, the one or more parameter values for the transmission bandwidth,” which is also circular. *Id.*, 16:15, 20. Similarly, claim 20 requires the “network connected device” to “transmit data associated with the network traffic from the first endpoint to the network connected device” and to “receive data for adjusting the transmission bandwidth from the network connected device,” i.e., transmitting data to and receiving data from itself, which is likewise circular. *Id.*, 16:41-42, 45-46.

Requiring a network connected device to function circularly makes no sense and a POSITA would not have understood its meaning. *See* Jeffay Decl. ¶¶13-18.

VII. Conclusion

For the reasons set forth above, the Court should adopt Defendants’ constructions.

Respectfully submitted,

Dated: April 23, 2025

By /s/ Harper S. Batts

Harper S. Batts (CA #242603)
hbatts@sheppardmullin.com
Christopher Ponder (TX #24065916)
cponder@sheppardmullin.com
Jeffrey Liang (CA #281429)
jliang@sheppardmullin.com
Lai L. Yip (CA #258029)
lyip@sheppardmullin.com
SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP
1540 El Camino Real Suite 120
Menlo Park, California 94025-4111
Telephone: 650.815.2600
Fax: 650.815.2601

*Attorneys for Defendants
Advanced Micro Devices, Inc.
and Pensando Systems, Inc.*

CERTIFICATE OF SERVICE

I hereby certify that on this 23rd day of April, 2025, I electronically filed the foregoing **Defendants' Opening Claim Construction Brief (including exhibits and declarations)** with the Clerk of the Court using the CM/ECF system which will send notification of such filing to the following:

Michael Matulewicz-Crowley
(mmatulewicz-crowley@reichmanjorgensen.com)
Christine E. Lehman
(clehman@reichmanjorgensen.com)
Scott L. Cole
(scole@reichmanjorgensen.com)
Ariane S. Mann
(amann@reichmanjorgensen.com)
Laura Carwile
(lcarwile@reichmanjorgensen.com)
RJ_XtreamEdge-
AMD@reichmanjorgensen.com
REICHMAN JORGENSEN
LEHMAN & FELDBERG LLP
Counsel for Plaintiffs

/s/ Harper S. Batts

Harper S. Batts
*Attorneys for Defendants
Advanced Micro Devices, Inc.
and Pensando Systems, Inc.*