

Paper No. \_\_

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

GOOGLE LLC,  
Petitioner,

v.

VIRTAMOVE, CORP.,  
Patent Owner.

---

Case No. IPR2025-00489  
Patent No. 7,784,058

---

**PETITION FOR INTER PARTES REVIEW  
UNDER 35 U.S.C. §§311-319 AND 37 C.F.R. §42.1 et seq**

**TABLE OF CONTENTS**

MANDATORY NOTICES.....x

- A. Real Party-In-Interest .....x
- B. Related Matters.....x
  - 1. United States Patent & Trademark Office .....x
  - 2. USPTO Patent Trial and Appeal Board ..... xi
  - 3. U.S. District Court for the Eastern District of Texas ..... xi
  - 4. U.S. District Court for the Western District of Texas..... xi
  - 5. U.S. District Court for the Northern District of California..... xii
- C. Counsel and Service Information - §42.8(b)(3) and (4)..... xii

I. STANDING ..... 1

II. GROUNDS FOR UNPATENTABILITY ..... 1

III. THE '058 PATENT..... 1

- A. Background and Specification..... 1
- B. Person of Ordinary Skill in the Art (“POSA”)..... 6
- C. Prosecution History ..... 7

IV. CLAIM INTERPRETATION ..... 8

V. CALLENDER RENDERS CLAIMS 1-18 OBVIOUS..... 8

- A. Callender (EX1005)..... 8
- B. Claim-by-Claim Analysis..... 15
  - 1. Claim 1 ..... 15
    - a. [1PRE]: “A computing system for executing a plurality of software applications comprising:”..... 15
    - b. [1A] “a) a processor;” ..... 18
    - c. [1B] ..... 18
      - i. [1B.1] “b) an operating system having an operating system kernel...” ..... 18
      - ii. [1B.2] “[OS kernel] having OS critical system elements (OSCSEs)...” ..... 20

iii.	[1B.3] “[OSCSEs] for running in kernel mode using said processor” .....	25
d.	[1C] .....	27
i.	[1C.1] “c) a shared library having shared library critical system elements (SLCSEs) stored therein...” .....	27
ii.	[1C.2] “for use by the plurality of software applications in user mode...” .....	30
e.	[1D] .....	34
i.	[1D.1] “i) wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and...” .....	34
ii.	[1D.2] “are accessible to some of the plurality of software applications and...” .....	39
iii.	[1D.3] “when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications” .....	41
f.	[1E] .....	42
i.	[1E.1] “ii) wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications and...” .....	42
ii.	[1E.2] “where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function, and...” .....	46
g.	[1F] “iii) wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously.” .....	47

2. Claim 2 .....	49
3. Claim 3 .....	51
4. Claim 4 .....	52
a. [4A] .....	52
b. [4B] .....	53
i. Queue Setup .....	54
ii. Interrupt Setup and Event Notification.....	58
5. Claim 5 .....	59
a. [5A] .....	59
b. [5B] .....	61
6. Claim 6 .....	64
7. Claim 7 .....	65
8. Claim 8 .....	66
9. Claim 9 .....	67
10. Claim 10 .....	68
11. Claim 11 .....	69
a. Device access .....	69
b. Interrupt Delivery.....	69
c. Virtual memory mapping .....	70
12. Claim 12 .....	70
13. Claim 13 .....	71
14. Claim 14 .....	71
15. Claim 15 .....	72
16. Claim 16 .....	74
17. Claim 17 .....	74
18. Claim 18 .....	75
VI. DISCRETIONARY DENIAL IS UNWARRANTED .....	76
A. §314(a).....	76
1. <i>Fintiv</i> .....	76

a. Stay Potential .....	76
b. Trial Timing .....	77
c. Litigation Investment .....	77
d. Issue Overlap.....	77
e. Litigation Defendants.....	77
f. Other Circumstances .....	77
2. Unified Reexamination .....	78
3. Amazon IPR .....	78
B. §325(d).....	79
VII. CONCLUSION.....	79
VIII. CLAIM LISTING.....	83

## TABLE OF AUTHORITIES

### CASES

<i>BMW of North America, LLC v. Michigan Motor Techs., LLC</i> , IPR2023-01224, Paper 15 (Feb. 15, 2024).....	77
<i>Facebook, Inc. v. Express Mobile Inc.</i> , IPR2021-01457, Paper 10 (Mar. 18, 2022).....	78
<i>Ford Motor Co. v. Neo Wireless LLC</i> , IPR2023-00763, Paper 28 (Mar. 22, 2024).....	78
<i>Google LLC v. Security First Innovations, LLC</i> , IPR2024-00215, Paper 15 (May 23, 2024) .....	8
<i>KSR Int’l v. Teleflex</i> , 550 U.S. 398 (2007) .....	22
<i>Markforged Inc. v. Continuous Composites</i> , IPR2022-00679, Paper 7 (Oct. 25, 2022).....	77
<i>PLR Worldwide Sales Ltd. v. Flip Phone Games, Inc.</i> , IPR2024-00209, Paper 9 (May 10, 2024) .....	21, 35
<i>Protect Animals With Satellites LLC v. OnPoint Sys., LLC</i> , IPR2021-01483, Paper 11 (Mar. 4, 2022).....	77
<i>Sand Revolution II, LLC, v. Cont’l Intermodal Group–Trucking</i> , IPR2019-01393, Paper 24 (June 16, 2020) .....	76
<i>Tesla, Inc. v. Graphite Charging Company LLC</i> , IPR2024-00388, Paper 15 (Aug. 27, 2024).....	78
<i>Videndum Production Sols., Inc. v. Rotolight Ltd.</i> , IPR2023-01218, Paper 12 (Apr. 19, 2024) .....	79
<i>Volkswagen Grp. of America, Inc. v. Carucel Investments, L.P.</i> , IPR2019-01105, Paper 8 (Dec. 2, 2019) .....	79

### REGULATIONS

37 C.F.R. §42.100(b) .....	8
----------------------------	---

37 C.F.R. §42.104(a).....1  
37 C.F.R. §42.104(b)(3).....8

**STATUTES**

35 U.S.C. §102(e) .....1  
35 U.S.C. §103 .....1  
35 U.S.C. §282(b) .....8  
35 U.S.C. §314(a) ..... 76, 78  
35 U.S.C. §315(d) .....78  
35 U.S.C. §325(d) .....79

**OTHER AUTHORITIES**

Director’s Interim Procedure for Discretionary Denials (June 21, 2022) ..... 76, 77

## APPENDIX LISTING OF EXHIBITS

Exhibit	Description
1001	U.S. Patent No. 7,784,058
1002	Prosecution History of U.S. Patent No. 7,784,058
1003	Declaration of Samrat Bhattacharjee, Ph.D. (“Bhattacharjee”)
1004	Curriculum Vitae of Dr. Samrat Bhattacharjee
1005	U.S. Patent No. 7,024,672 (“Callender”)
1006	RESERVED
1007	U.S. Patent No. 6,263,376 (“Hatch”)
1008	U.S. Patent No. 6,260,075 (“Cabrerero”)
1009	U.S. Patent No. 6,212,574 (“O’Rourke”)
1010	U.S. Patent No. 5,481,706 (“Peek”)
1011	U.S. Patent No. 7,499,966 (“Elnozahy”)
1012	U.S. Patent App. Pub. No. 2004/0216145 (“Wong”)
1013	U.S. Patent App. Pub. No. 2003/0065856 (“Kagan”)
1014	U.S. Patent No. 7,583,681 (“Green”)
1015	Excerpts from Charles Petzold, <i>Programming Windows 95</i> (Microsoft Press 1996) (“Petzold”)
1016	U.S. Patent App. Pub. No. 2002/0066086 (“Linden”)
1017	U.S. Patent No. 6,349,355 (“Draves”)
1018	Excerpts from Silberschatz et. al, <i>Operating System Concepts</i> (Wiley 6 <sup>th</sup> ed. 2002) (“Silberschatz”)
1019	U.S. Patent No. 7,451,456 (“Andjelic”)
1020	Chart re: ’058 Patent accompanying Plaintiff VirtaMove Corp.’s Supplemental Preliminary Disclosure of Asserted Claims and Infringement Contentions, in <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Sep. 26, 2024)
1021	U.S. Patent App. Pub. No. 2002/0095224 (“Braun”)
1022	U.S. Patent No. 6,173,336 (“Stoeckl”)
1023	U.S. Patent No. 6,016,515 (“Shaw”)
1024	U.S. Patent No. 7,080,172 (“Schmalz”)
1025	Excerpts from <i>Webster’s New World Computer Dictionary</i> (10 <sup>th</sup> ed. 2003)
1026	Excerpts from <i>Barron’s Dictionary of Computer and Internet Terms</i> (8 <sup>th</sup> ed. 2003)
1027	U.S. Patent No. 7,324,450 (“Oliver”)
1028	U.S. Patent App. Pub. No. 2003/0103455 (“Pinto”)
1029	U.S. Patent No. 6,526,567 (“Cobbett”)

<b>Exhibit</b>	<b>Description</b>
1030	U.S. Patent App. Pub. No. 2004/0142563 (“Fontarensky”)
1031	U.S. Patent No. 7,284,246 (“Kemp”)
1032	U.S. Patent No. 5,375,241 (“Walsh”)
1033	U.S. Patent No. 6,698,015 (“Moberg”)
1034	Excerpts from Collin, <i>Dictionary of Computing</i> (Collin 4 <sup>th</sup> Ed. 2002) (“Collin”)
1035	Microsoft Computer Dictionary (Microsoft 5 <sup>th</sup> ed. 2002) (“Microsoft”)
1036	U.S. Patent No. 6,526,159 (“Nickerson”)
1037	U.S. Patent App. Pub. No. 2003/0154320 (“Calusinski”)
1038	U.S. Patent No. 7,437,483 (“Goossen”)
1039	U.S. Patent No. 7,100,162 (“Green-162”)
1040	U.S. Patent No. 7,453,852 (“Buddhikot”)
1041	U.S. Patent No. 5,584,023 (“Hsu”)
1042	U.S. Patent No. 7,144,031 (“Lin”)
1043	U.S. Patent No. 6,792,492 (“Griffin”)
1044	U.S. Patent App. Pub. No. 2002/0116563 (“Lever”)
1045	U.S. Patent No. 6,594,698 (“Chow”)
1046	U.S. Patent No. 7,216,164 (“Whitmore”)
1047	U.S. Patent No. 6,988,271 (“Hunt”)
1048	Liss et. al, <i>Efficient Exploitation of Kernel Access to Infiniband: a Software DSM Example</i> , 11th Symposium on High Performance Interconnects, 2003 (“Liss”)
1049	Patel et. al., <i>A Model of Completion Queue Mechanisms Using the Virtual Interface API</i> , Proc. IEEE Int’l Conf. on Cluster Computing, 281-282 (IEEE 2002) (“Patel”)
1050	Scheduling Order in VirtaMove, Corp. v. Google LLC, 7:24-cv-00033-DC-DTG (W.D. Tex. June 17, 2024) (ECF 34)
1051	Federal Court Management Statistics–Profiles, U.S. District Courts–Combined Civil and Criminal (September 2024)
1052	Google LLC’s Proposed Claim Terms for Construction, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 1, 2024)
1053	Plaintiff’s Disclosure of Proposed Claim Constructions, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 1, 2024)
1054	U.S. Patent No. 7,124,260 (“LaBerge”)
1055	U.S. Patent No. 7,406,677 (“Colling”)
1056	U.S. Patent No. 5,278,969 (“Pashan”)

<b>Exhibit</b>	<b>Description</b>
1057	Excerpts of <i>Webster’s New World Dictionary</i> (4 <sup>th</sup> Ed. 2003)
1058	Joint Claim Construction Statement, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Dec. 18, 2024)
1059	Google’s Opening Claim Construction Brief, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 22, 2024)
1060	Request for <i>Ex Parte</i> Reexamination of U.S. Patent No. 7,784,058 (“Unified Reexam”)
1061	U.S. Patent No. 7,210,124 (“Chan”)
1062	U.S. Patent No. 7,055,152 (“Ganapathy”)
1063	U.S. Patent No. 6,419,502 (“Trammel”)
1064	U.S. Patent App. Pub. No. 2003/0140051 (“Fujiwara”)
1065	U.S. Patent No. 6,442,752 (“Jennings”)
1066	U.S. Patent App. Pub. No. 2003/0225864 (“Gardiner”)
1067	VirtaMove’s Responsive Claim Construction Brief, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Nov. 12, 2024)
1068	VirtaMove’s Surreply Claim Construction Brief, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Dec. 13, 2024)
1069	U.S. Patent No. 7,213,247 (“Wilner”)
1070	U.S. Patent App. Pub. No. 2004/0078600 (“Nilsen”)
1071	U.S. Patent App. Pub. No. 2002/0138675 (“Mann”)
1072-1088	RESERVED
1089	Order Granting Defendant Google LLC’s Motion to Transfer Venue, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Jan. 22, 2025)
1090	Order Cancelling Markman Hearing, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Jan. 17, 2025)
1091-1096	RESERVED
1097	U.S. Patent No. 7,424,710 (“Nelson”)
1098-1106	RESERVED
1107	Plaintiff VirtaMove Corp.’s Supplemental Preliminary Disclosure of Asserted Claims and Infringement Contentions, in <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Sept. 06, 2024)

## MANDATORY NOTICES

### A. Real Party-In-Interest

Petitioner is the Real Party-in-Interest.<sup>1</sup>

### B. Related Matters

#### 1. United States Patent & Trademark Office

The application from which U.S. Patent No. 7,784,058 issued claims priority to provisional application No. 60/504,213, filed September 22, 2003.

The following U.S. patent applications claim the benefit of priority to U.S. Patent 7,784,058:

(i) U.S. Patent Application 11/432,843 (U.S. Patent No. 7,757,291), filed May 12, 2006; (ii) U.S. Patent Application 11/380,285 (U.S. Patent No. 7,774,762), filed April 26, 2006; (iii) U.S. Patent Application 12/075,842 filed March 13, 2008.

U.S. Patent No. 7,784,058 is the subject of *Ex Parte* Reexamination No. 90/019,676, requested by Unified Patents, LLC, filed on September 23, 2024.

---

<sup>1</sup> Google LLC is a subsidiary of XXVI Holdings Inc., which is a subsidiary of Alphabet Inc. XXVI Holdings Inc. and Alphabet Inc. are not real parties-in-interest to this proceeding.

## **2. USPTO Patent Trial and Appeal Board**

Concurrently with the present petition, Petitioner is filing IPR2025-00490, also challenging U.S. Patent No. 7,784,058.

Petitioner is also filing IPR2025-00487 and IPR2025-00488 challenging U.S. Patent No. 7,519,814, which is also asserted in *VirtaMove, Corp. v. Google LLC*, Case No. 7:24-cv-00033, listed below.

U.S. Patent No. 7,784,058 is the subject of *Amazon.com, Inc. v. VirtaMove, Corp.*, IPR2025-00561, filed on January 30, 2025, by different petitioner Amazon.com, Inc.

## **3. U.S. District Court for the Eastern District of Texas**

(i) *VirtaMove, Corp. v. Hewlett Packard Enterprise Company*, Case No. 2:24-cv-00093;

(ii) *VirtaMove, Corp. v. International Business Machines Corporation*, Case No. 2:24-cv-00064.

## **4. U.S. District Court for the Western District of Texas**

(i) *VirtaMove, Corp. v. Google LLC*, Case No. 7:24-cv-00033 (pending transfer to Northern District of California per Order dated January 22, 2025, *see* EX1089);

(ii) *VirtaMove, Corp. v. Amazon.com, Inc. et al*, Case No. 7:24-cv-00030 (pending transfer to Northern District of California per Order dated January 22, 2025, *see* Docket Entry No. 87);

(iii) *VirtaMove, Corp. v. Microsoft Corp.*, Case No. 7:24-cv-00338;

(iv) *VirtaMove, Corp. v. Oracle Corp.*, Case No. 7:24-cv-00339.

**5. U.S. District Court for the Northern District of California**

(i) *Red Hat, Inc. v. VirtaMove, Corp.*, No. 5:23-cv-04740.

**C. Counsel and Service Information - §42.8(b)(3) and (4)**

Lead Counsel	Elisabeth H. Hunt, Reg. No. 67,336
Backup Counsel	Anant K. Saraswat, Reg. No. 76,050 Gregory S. Nieberg, Reg. No. 57,063
Service Information	<u>E-mail</u> : EHunt-PTAB@wolfgreenfield.com ASaraswat-PTAB@wolfgreenfield.com GNieberg-PTAB@wolfgreenfield.com  <u>Post and hand delivery</u> : Wolf, Greenfield & Sacks, P.C. 600 Atlantic Avenue Boston, MA 02210-2206  <u>Telephone</u> : 617-646-8000 <u>Facsimile</u> : 617-646-8646

A power of attorney is submitted with the Petition. Counsel for Petitioner consents to service of all documents via electronic mail.

Petitioner requests *inter partes* review and cancellation of claims 1-18 of the '058 patent (EX1001).

## **I. STANDING**

Petitioner certifies that the '058 patent is available for *inter partes* review and that Petitioner is not barred or estopped from requesting *inter partes* review of the challenged claims. 37 C.F.R. §42.104(a).

## **II. GROUNDS FOR UNPATENTABILITY**

Claims 1-18 are unpatentable under pre-AIA 35 U.S.C. §103 over Callender (EX1005). Callender is prior art under (at least) pre-AIA §102(e) to the '058 patent's earliest claimed priority date and was not of record during prosecution.

## **III. THE '058 PATENT**

### **A. Background and Specification**

A typical computer system may run multiple “application[s]” (or “application programs”), *e.g.*, email or word-processing, that must share the computer system's resources, like hardware or files. EX1001, 1:21-31; EX1035, 378; EX1003 (“Bhattacharjee”), ¶31. The computer's “operating system” (“OS”) “traditionally...provides mechanisms to...control [the applications'] access to shared resources.” EX1001, 1:20-24. Additionally, the '058 patent states that the OS “‘normally’ supplie[s]” “service[s] or part[s] of a service” that are “critical to the operation of a software application”; the patent calls such services, or parts thereof, “critical system elements” (“CSEs”). EX1001, 2:1, 6:6-9; Bhattacharjee,

¶32. Examples of CSEs include services such as networking, file access, “memory allocation” and “device access.” EX1001, 5:38-45, 6:12-28, 9:33-35;

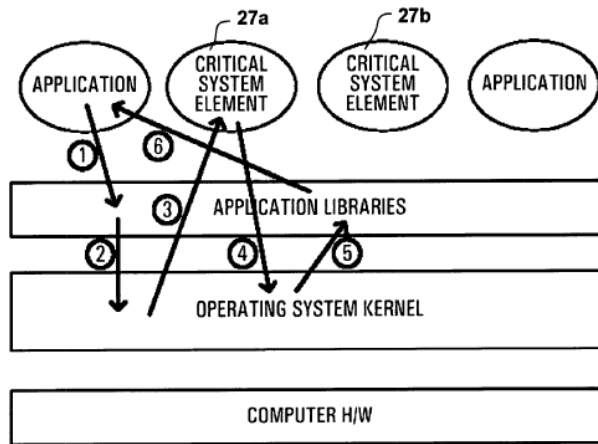
Bhattacharjee, ¶32.

CSEs are “normally” found in the OS’s “kernel.” EX1001, 5:21-23. The “kernel” is the “core” of an OS and performs tasks like “manag[ing] memory, files, and peripheral devices” (*e.g.*, disks, network interfaces), and “allocat[ing] system resources.” EX1035, 300, 398; Callender, 1:40-43; Bhattacharjee, ¶33.

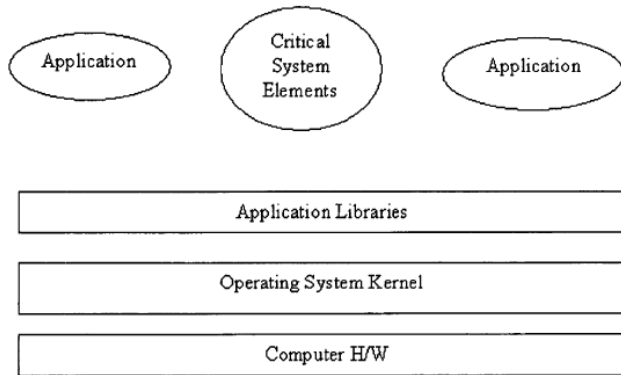
Conventional computer processors typically run in a special “kernel mode” when executing the kernel; running in this mode gives the kernel unrestricted access to the computer’s resources, enabling the kernel to perform its tasks. EX1001, 6:32-36; Callender, 1:37-48; EX1003, ¶34. In contrast, when executing applications, processors typically run in “user mode,” where access to certain resources is restricted to prevent applications from interfering with each other or damaging the system—application code cannot run in kernel mode. EX1001, 6:31-36;

Callender, 1:32-48; Bhattacharjee, ¶35. However, applications often need to use CSEs that are typically provided by the kernel; to access kernel services, applications typically make “system calls” to request that the OS kernel perform a needed service (like a CSE) for the application, using the processor executing in kernel mode. EX1001, 5:38-58, 6:66-7:16; Bhattacharjee, ¶¶36-37.

The above-described system-call-based technique has known disadvantages, including that (i) switching processor modes takes time and thus may impact performance; and (ii) all applications must use the same version of a CSE, *i.e.*, whatever version the computer system’s OS kernel provides. Callender, 1:50-56; EX1001, 5:54-6:3; Bhattacharjee, ¶38. Thus, it was known to implement some CSEs in user mode instead. EX1001, 7:23-25 (’058 patent admitting this was known); Bhattacharjee, ¶39. In some known user-mode CSE implementations, the CSE was provided by a “process[]” that executed in user mode but was separate from any application that used the CSE. EX1001, 7:23-30. In other words, when a user-mode application needed a CSE, it would request that a different **user-mode** process, rather than the kernel, perform the CSE. Different user-mode processes provided different user-mode CSEs, or a single user-mode process provided all user-mode CSEs. EX1001, 7:23-61; Figs. 2a-2b; Bhattacharjee, ¶¶40-41. As illustrated in Fig. 2a (below), however, communication between a user-mode application needing a CSE and a user-mode process providing that CSE still passed through the kernel—that is, the application issued a request to the kernel, which then invoked the user-mode CSE process. EX1001, 7:31-52; Bhattacharjee, ¶41.



**FIG. 2a**  
**Prior Art**



**FIG. 2b**  
**Prior Art**

In contrast to the above-described scheme requiring communication with a separate user-mode CSE process via the kernel, the '058 patent discloses “replicat[ing] [CSEs] in user mode” by “placing CSEs similar to those in the OS in shared libraries.” EX1001, 5:22-34; Bhattacharjee, ¶42. “The CSE library includes replicas or substantial functional equivalents or replacements of kernel functions.” EX1001, 8:27-28. POSAs understood that a “function” is a predefined

set of instructions that carry out a specific action (e.g., a CSE). Bhattacharjee, ¶43; EX1007, 1:22-33. Placing a CSE in a shared library “provides a means of attaching or linking a CSE service to an application having access to the shared library.” EX1001, 5:29-31, 9:15-27; Bhattacharjee, ¶44. The “functions” in the shared CSE library “**run in the context** of a software application.” EX1001, 5:22-25;<sup>2</sup> *see also id.*, 9:41-42 (“FIG. 4 shows that the invention allows for [CSEs] to exist in the same context as an application.”), FIG. 4. In other words, the user-mode application itself performs the CSE (by calling a function from the library), rather than asking another user-mode process to perform the CSE. Bhattacharjee, ¶45.

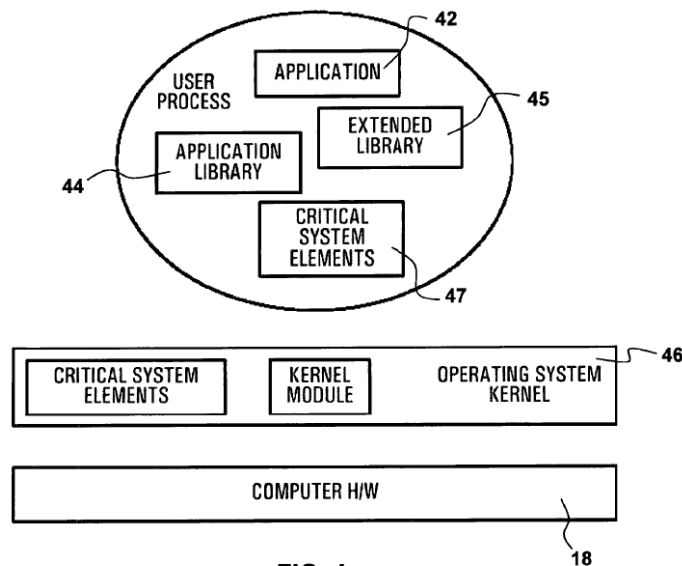


FIG. 4

<sup>2</sup> Emphases added throughout unless otherwise indicated.

The patent refers to user-mode CSEs in a shared library as “SLCSEs,” and to CSEs in the kernel as “OS [operating system] critical system elements (OSCSEs).” EX1001, 2:7-12. The patent mentions the well-known “dynamic linked library (DLL)” as an example of a “shared library.” EX1001, 2:45-51; Bhattacharjee, ¶46.

The patent contends that CSEs that “execute” “in the same context as an application” “contrast” with prior-art user-mode CSEs that were allegedly only implemented as “shared service[s].” EX1001, 1:46-54; Bhattacharjee, ¶47. However, the patent admits that “sharing of code” between applications via a user-mode shared library was “**common practice.**” EX1001, 3:30-33; *see also* EX1001, 7:3-5; EX1003, ¶47. The patent also admits that it was “typical” to provide “functions” in “libraries which applications link with.” EX1001, 6:37-45; Bhattacharjee, ¶47. Nowhere does the patent explain what is purportedly novel about using linking and shared libraries to make CSEs available to applications. Bhattacharjee, ¶48. In fact, as demonstrated below, Callender disclosed doing just that, well before the ’058 patent’s filing.

#### **B. Person of Ordinary Skill in the Art (“POSA”)**

A POSA as of the ’058 patent’s September 22, 2003 earliest claimed priority date would have had at least a bachelor’s degree in computer science, computer engineering, or a related field, with three years of academic and/or industry experience in the areas of “computing system[s]” and “application libraries.”

EX1001, 1:15-17. More education may substitute for less experience.

Bhattacharjee, ¶¶49-51.

### **C. Prosecution History**

As detailed where relevant *infra* §V.B, the examiner rejected the originally-filed claims over Cabrero (EX1008), after which the applicants amended claim 1 to require SLCSEs to be “*functional replicas of OSCSEs.*” EX1002, 216-222, 235, 265-266, 273, 278; Bhattacharjee, ¶¶52-53. The examiner then rejected the claims over O’Rourke (EX1009) and Peek (EX1010). EX1002, 296-297. The applicants traversed the rejection, as also detailed *infra* §V.B. EX1002, 323-325.

Subsequently, for reasons not recorded, the examiner discussed two different references, Elnozahy (EX1011) and Wong (EX1012), in an examiner interview, and then allowed the claims without further discussing O’Rourke or Peek.

EX1002, 348-351. The allowed claims contain an examiner’s amendment to Element [1E] (*see* claim listing *infra* §VIII) to recite “*at least a first*” and “*at least a second*” of the *plurality of software applications*. EX1002, 350-351. The examiner also incorporated into claim 1 originally-filed dependent claim 6, which is essentially limitation [1F]’s “wherein” clause. EX1002, 350-352; *infra* §VIII; Bhattacharjee, ¶¶54-56.

The examiner said neither Elnozahy nor Wong disclosed the amended language of claim 1. EX1002, 352-353. But neither reference explicitly discloses

shared libraries or an SLCSE in a shared library, and there is no indication that the examiner considered whether these features would have been obvious.

Bhattacharjee, ¶¶57-61. Furthermore, Callender was not of record.

#### **IV. CLAIM INTERPRETATION**

Claim terms are construed herein using the standard used in civil actions under 35 U.S.C. §282(b), in accordance with the ordinary and customary meaning as understood by POSAs and the patent’s prosecution history. 37 C.F.R. §42.100(b). In the concurrent district-court proceeding, the parties (Petitioner “Google” and Patent Owner “VirtaMove”) have proposed constructions of various claim terms, including competing constructions for some terms and agreed constructions for others. EX1052-EX1053. As discussed further within the Ground, this Petition presents alternative mappings of the prior art under both parties’ proposed constructions of disputed terms, thus demonstrating unpatentability regardless of which proposed construction is correct. *Google LLC v. Security First Innovations, LLC*, IPR2024-00215, Paper 15, 6 (May 23, 2024) (finding this approach “complies with” Rule 42.104(b)(3)).

#### **V. CALLENDER RENDERS CLAIMS 1-18 OBVIOUS**

##### **A. Callender (EX1005)**

Callender discloses techniques for “a single implementation of operations that are common to both kernel mode processing and user mode processing, relative to a hardware adapter.” Callender, 2:28-31; Bhattacharjee, ¶62. An

example “adapter” is a “host channel adapter (‘HCA Adapter’),” which enables communication using the well-known “InfiniBand” networking protocol.

Callender, 4:51-57; Bhattacharjee, [0007]-[0008]; EX1014, 1:50-60;

Bhattacharjee, ¶63. Examples of “operations” that are “common” to both user- and kernel-mode processing include “operations” for “sending and receiving”

information through the adapter. Callender, Abstract, 2:41-50, 3:40-4:16, 4:51-5:5, claims 4, 12, 22; Bhattacharjee, ¶64.

Callender’s techniques are performed in a “computer” with a conventional “operating system.” Callender, 8:19-9:5, Fig. 4; Bhattacharjee, ¶65. Callender’s “computer” also has various hardware components, including network “adapter[s]” such as an HCA adapter. Callender, 9:22-24; Bhattacharjee, ¶66. Callender’s Figure 1 illustrates “a hardware driver model that allows for a single implementation of common user mode and kernel mode operations.” Callender, 3:39-44. A “driver” is software by which a computer interacts with a hardware device such as an adapter. EX1035, 177; Bhattacharjee, ¶67. In Callender’s model, a “user mode implementation” of an operation provides applications with “direct access to [the] adapter...without switching to kernel mode.” Callender, 3:45-49. The model also comprises a “kernel mode implementation” of “all operations possible for [the] adapter,” including “operations unique to kernel mode” and operations common to both modes. Callender, 3:52-4:12. “[F]requent

operations like sending and receiving information” are operations for the adapter that are “likely candidates to be included” in both user- and kernel-mode implementations. Callender, 4:4-12; Bhattacharjee, ¶¶68.

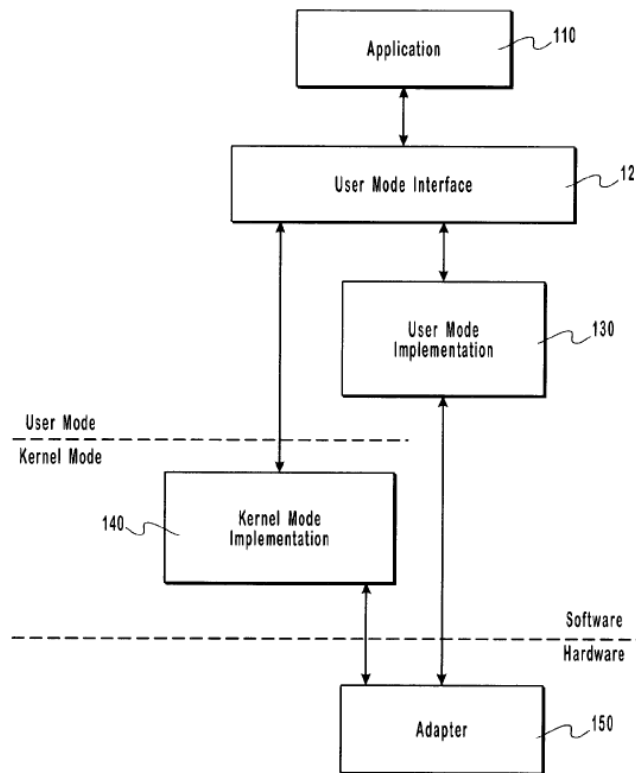


Fig. 1

Figure 2A (annotated below) shows a more specific use case for the general model of Figure 1—an “example hardware driver model that allows for a single implementation of common user mode and kernel mode operations” specifically for an HCA adapter. Callender, 4:51-57. The example model includes a “kernel mode miniport driver” in kernel mode and a “user mode library” in user mode. Callender, 4:55-59; Bhattacharjee, ¶¶69-70. An “InfiniBand application 210A,”

which is a user-mode application using the InfiniBand networking protocol, “accesses [HCA adapter] 280A through **user mode library** 230A and through **kernel mode miniport** driver 240A.” Callender, 4:54-57; Bhattacharjee, ¶71. The “user mode library” is “implemented as a dynamic link library (‘DLL’)” (an alternate spelling for what the ’058 patent calls a “dynamic linked library (DLL)”). Callender, 5:13-15; EX1001, 2:48; Bhattacharjee, ¶72. As explained below, POSAs understood that Callender’s “**user mode library**” includes “operations” for “sending and receiving” data via the HCA adapter that are also implemented in the “**kernel mode miniport driver.**” Callender, 4:4-12, 4:55-59; Bhattacharjee, ¶73. In other words, just as the ’058 patent describes (*supra* §III.A), the same service is provided by both a user-mode DLL and the kernel, and applications configured to call the DLL can use the user-mode service, while other applications can use the kernel-mode service. Bhattacharjee, ¶73.

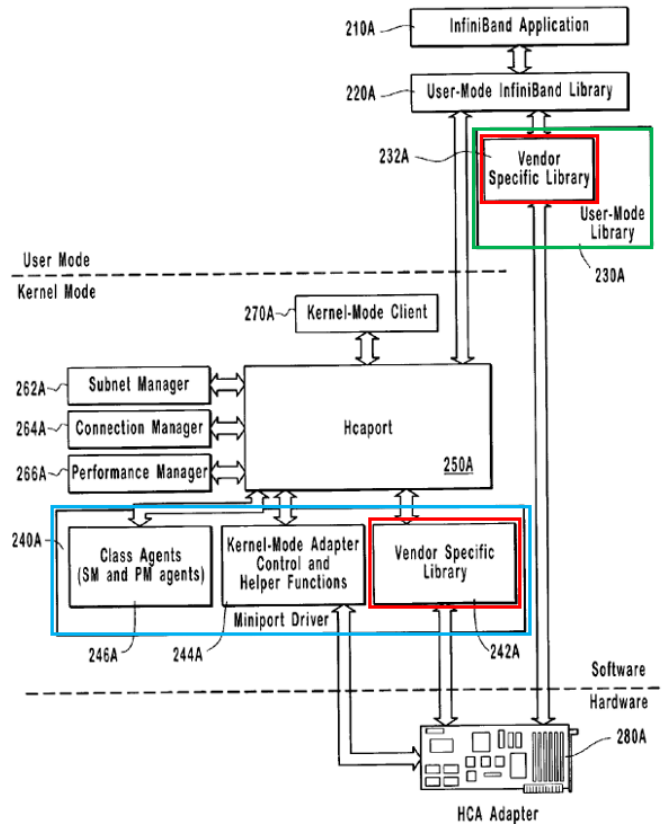


Fig. 2A

Callender explains that “hardware-specific operations” for the HCA adapter are implemented in a “**vendor specific library**,” also called a “**process mode independent (‘PMI’) HCA IO [input/output] access library**.” Callender, 5:20-28, Fig. 2A; Bhattacharjee, ¶74. The “**PMI HCA IO access library**” is incorporated into both the **user mode library** and the **kernel-mode miniport driver**. Callender, 5:19-42, 6:45-56, Fig. 2A; Bhattacharjee, ¶75. POSAs understood that “operations” for “sending and receiving” information through the adapter are examples of “IO [input/output] access,” because those operations require sending data into and reading data out of, respectively, the adapter. Callender, Abstract,

2:41-52, 3:53-4:16, 4:51-5:5, 5:26-28; Bhattacharjee, ¶76. Thus, POSAs understood that the “IO access” features of the “**vendor specific library**”/“**PMI HCA IO access library**” are used by both the user-mode and kernel-mode versions of the “operations for sending and receiving” information through the HCA adapter. Callender, 2:40-52, 3:40-4:16, 4:51-5:27; Bhattacharjee, ¶76.

Additionally, Callender refers to “kernel-bypass IO [input/output] features offered by the user mode library 230A.” Callender, 5:15-18, Fig. 2A (annotated below, showing arrow from “User-Mode Library 230” to “HCA Adapter 280A” bypassing the “Kernel Mode”); Bhattacharjee, ¶77.

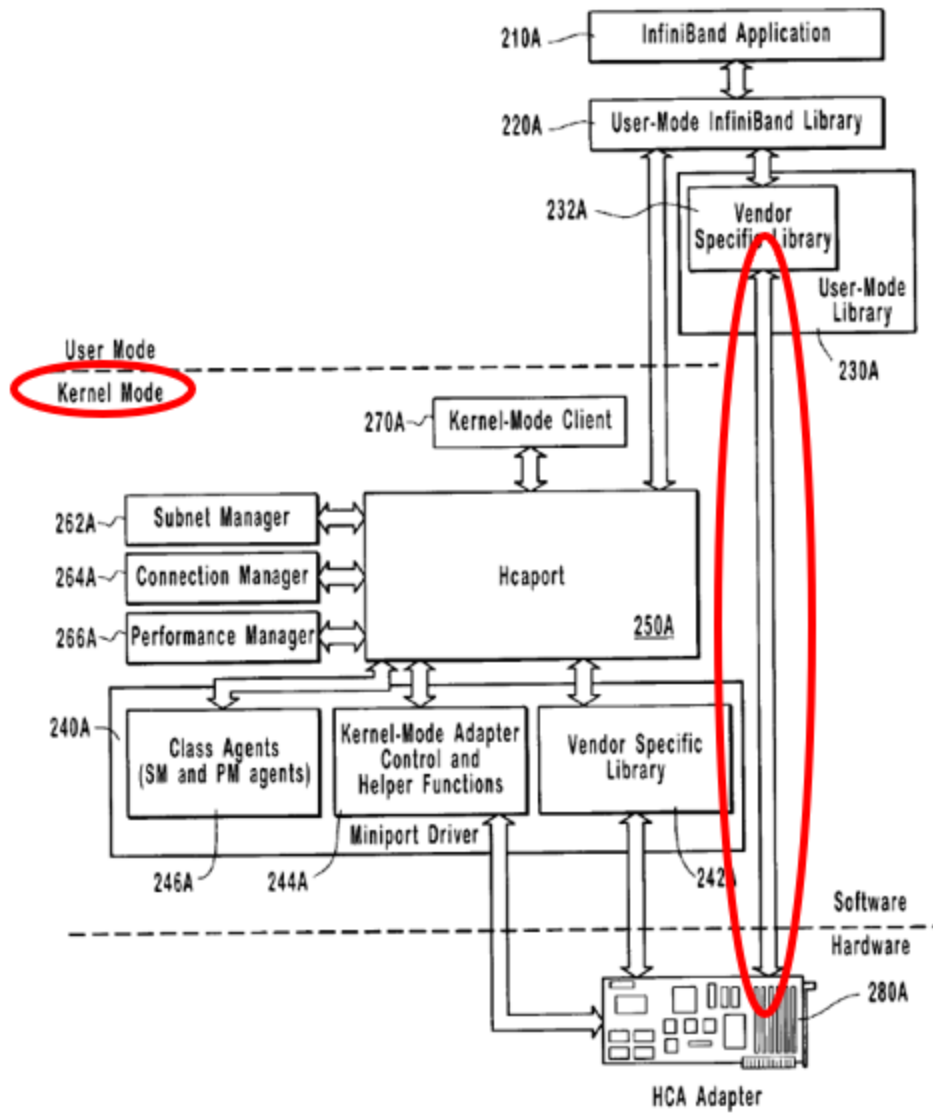


Fig. 2A

As noted above, POSAs understood that Callender’s “operations” for “sending and receiving” information through the adapter are examples of “IO access” operations. Additionally, Callender notes that “some software drivers bypass kernel mode for certain operations.” Callender, 1:59-61. Therefore, POSAs understood that the user mode implementations of the “sending” and “receiving” “operations” are

contained in the “user mode library 230A” and utilize the “kernel-bypass IO features.” Bhattacharjee, ¶77.

Finally, POSAs also understood that the sending and receiving “operations” in the “user mode library” are implemented as functions. As noted *supra* §III.A, POSAs understood that a function is a predefined set of instructions that carry out a specific action. Furthermore, Callender discloses “functions” in a library. Callender, 5:19-28, 6:45-56 (describing “functions” in the “PMI library” that is part of the user mode library). To the extent Callender does not disclose implementing the sending and receiving operations as functions, this would have been obvious, as the use of functions in a DLL was a well-known and customary software technique that was within a POSA’s skill and would beneficially provide a predefined set of instructions for performing a repeated operation like sending or receiving. Bhattacharjee, ¶78; EX1015, 959-960 (“...one purpose of dynamic link libraries is to provide functions and resources that can be used by many different programs...”); EX1007, 1:22-29.

## **B. Claim-by-Claim Analysis**

### **1. Claim 1**

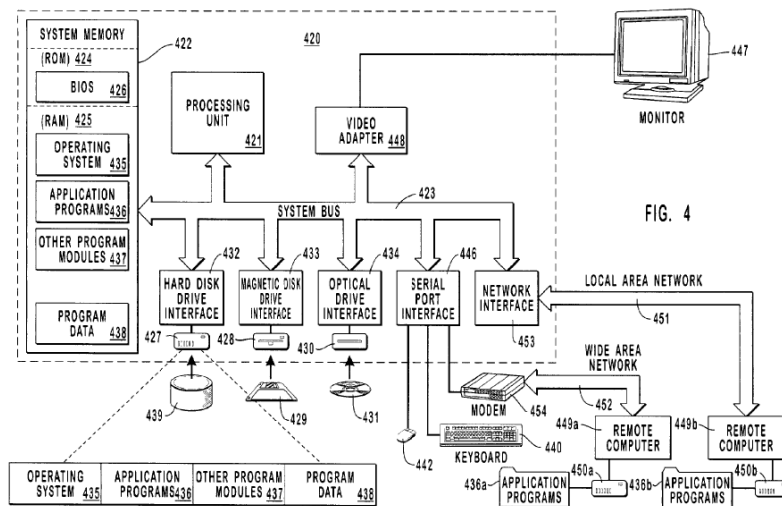
#### **a. [1PRE]: “A computing system for executing a plurality of software applications comprising:”**

Callender’s independent claims 1 and 9 recite a “*computer system*.”

Additionally, Callender’s Figure 4 depicts an “exemplary *system* for implementing

[its] invention.” Callender, 8:19-20. This “system” includes a “**computer**” having a “*system* memory” and “*system* bus.” Callender, 8:19-24, Fig. 4. Thus, POSAs would understand that Callender’s “computer” is a *computing system*.

Bhattacharjee, ¶79.



Callender, Fig. 4

Callender’s computer includes a “processing unit” connected to “system memory” storing “application **programs**.” Callender, 8:19-58, Fig. 4; Bhattacharjee, ¶80. Additionally, when discussing Fig. 1, which is a “block diagram of [Callender’s] hardware driver model,” Callender says the “model” lets “*applications*” (plural) “interact with adapter 150.” Callender, 3:10-13, 3:40-59; Bhattacharjee, ¶81. Thus, Callender’s system includes a *plurality* of “*applications*,” which Callender also calls “application programs,” that interact

with the adapter. Callender, 3:57, 8:57; Bhattacharjee, ¶82. POSAs understood that these applications are *software applications* at least because Callender uses “application” to refer to a “*software*” application. Callender, 1:18-29, Fig. 1 (showing “Application” 110 running in “user mode” “layer” of “*Software*”); Bhattacharjee, ¶83.

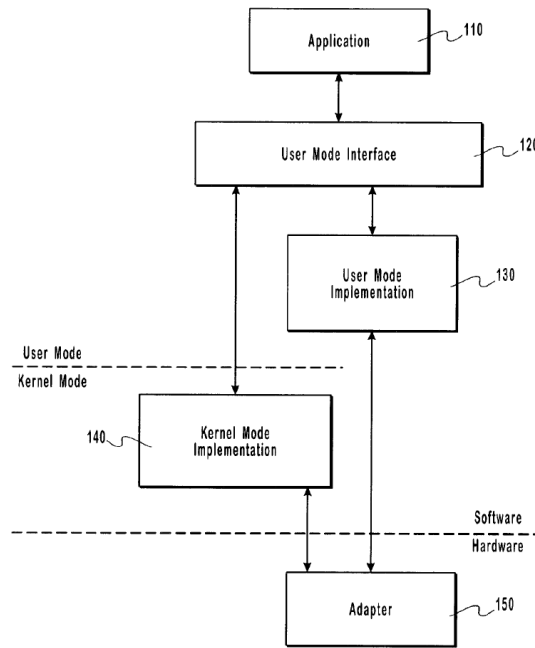


Fig. 1

POSAs further understood that the “applications” comprise “computer-*executable* instructions.” Callender, 9:59-62 (Callender’s invention is “described in the general context of computer-*executable* instructions”); Bhattacharjee, ¶84. Finally, POSAs understood that the “processing unit” in Callender’s “system” *executes* the application programs’ instructions because executing instructions is

the well-known purpose of a processing unit. Callender, 8:19-58, Fig. 4; Bhattacharjee, ¶85; EX1016, [0005].

Thus, Callender’s “computer” (8:21) is a *computing system for executing a plurality of software applications*. Bhattacharjee, ¶86.

**b. [1A] “a) a processor;”**

Callender’s “computer” includes a “*processing* unit,” which POSAs understood is a “*processor*.” Callender, 8:19-24, Fig. 4; Bhattacharjee, ¶87; *see also* EX1020, 3-4 (VirtaMove infringement contentions accusing “CPU,” *i.e.*, “central processing unit”); Microsoft, 92. POSAs understood that this processing unit is a physical computer processor. Bhattacharjee, ¶88; Callender, 8:19-9:5, Fig. 4 (describing hardware including the “processing unit”). Thus, Callender’s “processing unit” meets the agreed litigation construction of *processor*. *See* EX1058, 6.

**c. [1B]**

**i. [1B.1] “b) an operating system having an operating system kernel...”**

Callender’s “computer” *comprises* an “*operating system*.” Callender, Fig. 4, 5:55-65, 8:54-58; Bhattacharjee, ¶89. POSAs understood that Callender’s “operating system” is “[t]he software that controls the allocation and usage of hardware resources such as memory, central processing unit (CPU) time, disk space, and peripheral devices” (EX1035, 378), consistent with Callender’s

“Background” discussion of “operating systems” having an “operating system kernel” that “acts as a *gatekeeper* to computer *resources*” (Callender, 1:41-42). Bhattacharjee, ¶¶90-91. Thus, Callender’s “operating system” meets Google’s litigation construction. EX1059, 4.<sup>3</sup>

Callender’s “Background” references an “*operating system kernel*.” Callender, 1:39-42. Callender also describes its inventive “hardware driver model” (3:40-43) as having “kernel mode” components. *E.g.*, Callender, Abstract, 4:41-5:5; Bhattacharjee, ¶92. Additionally, Callender says the “user mode library” provides “kernel-bypass IO features.” Callender, 5:15-18. POSAs thus understood that the “operating system” in Callender’s computer *has an operating system kernel*. Bhattacharjee, ¶93. At minimum, POSAs would have found it obvious to include a kernel in the OS; it was customary for OSs to have kernels, as the ’058 patent admits, and POSAs would reasonably have expected success in this customary implementation. EX1001, 6:62-7:16 (referring to the “operating system kernel” in the “conventional architecture” illustrated in Figure 1); EX1017, 1:38-47; Bhattacharjee, ¶94. POSAs understood that Callender’s “kernel” (3:42) is a conventional kernel, *i.e.*, the “core...portion” of an operating system “that manages memory, files, and peripheral devices; maintains the time and date; launches

---

<sup>3</sup> VirtaMove asserted no construction necessary.

applications; and allocates system resources” (EX1035, 257), consistent with Callender’s “Background” discussion of the “operating system kernel” noted above (Callender, 1:40-43). Bhattacharjee, ¶¶95-96. Thus, Callender’s “kernel” meets Google’s litigation construction. EX1059, 4.<sup>4</sup>

Thus, Callender’s computer *comprises an operating system having an operating system kernel*. Bhattacharjee, ¶97.

**ii. [1B.2] “[OS kernel] having OS critical system elements (OSCSEs)...”**

The ’058 patent says a *critical system element* (CSE) is “[a]ny service or part of a service, ‘normally’ supplied by an operating system, that is critical to the operation of a software application.” EX1001, 6:6-8. Examples of CSEs include “[n]etwork services” including “message passing protocols.” EX1001, 6:12-13; Bhattacharjee, ¶98. The patent also notes that a “kernel may provide a TCP/IP service,” which POSAs understood was a type of network service, and describes this as an example of a CSE. EX1001, 3:22-30; Bhattacharjee, ¶99; EX1066, [0004] (referring to “TCP/IP” as a “network protocol”).

Callender discloses “operations” for “sending” information to and “receiving” information from the HCA adapter. *E.g.*, Callender, Abstract, 2:40-52, 4:51-5:5; Bhattacharjee, ¶100; *supra* §V.A. HCA adapters are used for

---

<sup>4</sup> VirtaMove asserted no construction necessary.

communication using the InfiniBand networking protocol. *Supra* §V.A; Bhattacharjee, ¶101; EX1013, [0007]-[0008]; EX1014, 1:50-60. Callender’s “operations for sending and receiving information from [an HCA] adapter” (Callender, 4:4-7) are therefore “[n]etwork services” (EX1001, 6:12-13). Bhattacharjee, ¶102. Thus, while the outer boundaries of what is “normally supplied” and “critical” in the ’058 patent’s definition of *CSE* (EX1001, 6:6-8) are unspecified (EX1059, 14-16), POSAs understood that at least Callender’s “operations” for “sending” information to and “receiving” information from the HCA adapter are within the scope of *CSEs*. Bhattacharjee, ¶103; *PLR Worldwide Sales Ltd. v. Flip Phone Games, Inc.*, IPR2024-00209, Paper 9, 39 (May 10, 2024) (“even if full scope of” a term “is indefinite because the specification does not provide a sufficient boundary...that does not prevent us from making a determination that a teaching is well within that boundary”).

Furthermore, Callender’s operations for sending and receiving are *CSEs* even if VirtaMove’s litigation construction were accepted (*see* EX1059, 14), since these operations match examples of *CSEs* provided in the ’058 patent. *See* EX1067, 17 (VirtaMove claim-construction brief citing “TCP/IP” and “network services” as examples of *CSEs* in patent); Bhattacharjee, ¶104.

The ’058 patent also says “[a] *CSE* is a dynamic object providing some function that is executing instructions used by the applications.” EX1001, 6:8-10.

Neither party has asserted that this language imposes a claim requirement— nevertheless, this would have been an obvious implementation of Callender. A “dynamic object” is an “object,” *i.e.* a set of instructions and/or data, that can be created by a program and then erased. Bhattacharjee, ¶¶105-106; EX1021, [0053]. Callender’s “operations” for “sending and receiving” (4:8-9) are part of a “driver.” Callender, 1:11-18, 3:10-15; Bhattacharjee, ¶107. Implementing drivers as sets of instructions was well-known. Bhattacharjee, ¶107; EX1103, 1:29-33. POSAs would have implemented Callender’s drivers as dynamic objects, and reasonably expected success, because (1) it was known to load drivers into memory and erase them as needed; and (2) dynamically loading objects was one of two known options, *i.e.* static and dynamic. Bhattacharjee, ¶108; EX1024, 1:7-11; EX1032, 1:36-59; EX1104, 1:11-22; *KSR Int’l v. Teleflex*, 550 U.S. 398, 416-17 (2007).

Callender discloses “kernel mode” implementations of the above-discussed sending and receiving “operations.” *Supra* §V.A; Callender, 2:40-52; Bhattacharjee, ¶109. POSAs understood, or would have found obvious, that operations implemented in “kernel mode” were part of the *operating system* because “kernel mode” operations were customarily part of the *OS* (which includes the kernel). Bhattacharjee, ¶109; EX1017, 1:38-47 (“Critical **operating system components** are implemented in kernel mode[.]”). Thus, Callender’s “kernel mode implementation[s]” (2:40-45) of the “operations for sending and receiving

information from adapter 150” (4:5-7) are *OS critical system elements (OSCSEs)*.

Bhattacharjee, ¶110.

Furthermore, as explained below, POSAs understood that the *OS kernel has* these *OSCSEs*. As discussed above, Callender’s OSCSEs are part of Callender’s “kernel mode implementation.” Callender, 4:4. POSAs understood that Callender’s “kernel mode implementation” is part of Callender’s “kernel,” *i.e.*, the *[OS] kernel*. *Supra* §V.B.1.c.i; Bhattacharjee, ¶¶111-112. Callender discusses “kernel-bypass IO features offered by the user mode library 230A,” and Figure 2A (annotated below) shows an arrow going from “User-Mode Library 230A” directly to “HCA Adapter” 280A, bypassing the components in “Kernel Mode.” Callender, 5:15-18. Furthermore, referencing Figure 2B, Callender states that “[k]ernel specific routines are implemented in kernel mode adapter control and helper functions 244B.” Callender, 5:43-54; EX1003, ¶¶113-114. Thus, POSAs understood that in Callender, “kernel mode” operations or functions reside in the “kernel,” which is “bypassed” by certain “IO features” in the user-mode library. Bhattacharjee, ¶115.

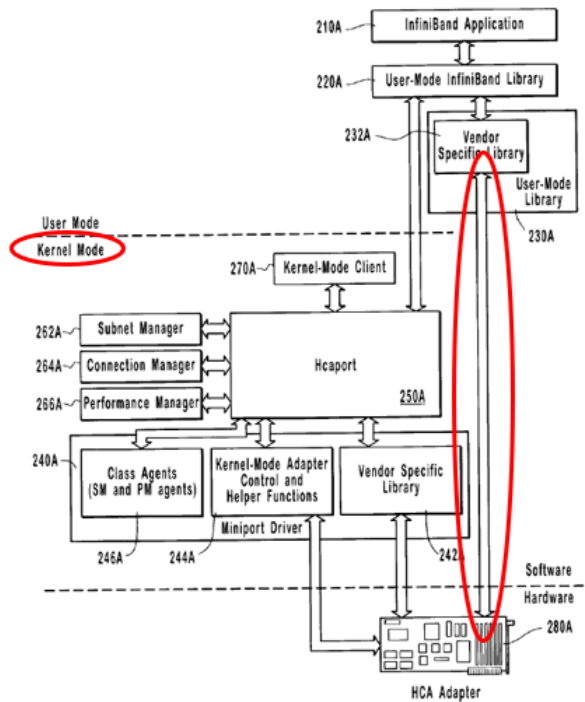


Fig. 2A

To the extent Callender is not considered to disclose that the “kernel mode” “operations for sending and receiving” (4:4-7) are in the OS kernel, this would have been obvious. POSAs understood that it was customary to implement critical functions, including communicating with hardware, in the kernel. Bhattacharjee, ¶116; EX1035, 300; EX1017, 1:38-47; EX1025, 205; EX1026, 275. Furthermore, Callender explains that the kernel-mode “operations for sending and receiving information” are part of a “driver.” *E.g.*, Callender, 1:11-18, 3:10-15. It was customary for drivers to be part of an OS kernel. Bhattacharjee, ¶¶117-118; EX1027, 8:64-9:5; EX1028, [0031]; EX1029, 5:45-49; EX1070, [0065]. POSAs

would reasonably have expected success in such an implementation because it was a customary kernel implementation, as mentioned above. Bhattacharjee, ¶119.

Thus, Callender's OS has an OS kernel *having OSCSEs*. Bhattacharjee, ¶120.

**iii. [1B.3] “[OSCSEs] for running in kernel mode using said processor”**

Callender's “kernel mode” versions of the “operations for sending and receiving” (4:4-7) are *OSCSEs*. *Supra* §V.B.1.c.ii. Callender says the “operations for sending and receiving may be common to both user mode processing and **kernel mode processing**.” Callender, 2:47-50; Bhattacharjee, ¶¶121-122.

Callender also discloses that “processes” “**run[]**” in either user or kernel mode. Callender, 1:33-39 (“application processes **run** within user mode...Processes **running** in kernel mode are privileged” and are “without the restrictions that apply to user mode processes.”). Thus, POSAs understood, or at minimum would have found obvious, that Callender's kernel-mode operations for sending and receiving are used by “processes” that “run” in “*kernel mode*,” *i.e.*, these operations are *for running in* Callender's “*kernel mode*.” Bhattacharjee, ¶¶123-124. POSAs also understood that Callender's “kernel mode” is *run using* Callender's “processing unit” (8:22), *i.e.*, *processor* (*supra* §V.B.1.b ([1A])), because a computer's processor runs kernel-mode processes. Bhattacharjee, ¶125; Callender, 1:44-48

(“many microprocessors have processing modes to support the distinctions between user mode processes and kernel mode processes.”).

The '058 patent's specification defines “kernel mode” as “[t]he context in which the kernel portion of an operating system executes.” EX1001, 6:32-33. The parties' agreed litigation construction adopts this definition and adds the specification's statement that “[a]pplicaton code cannot run in kernel mode.” EX1001, 6:34-36; EX1058, 6; Bhattacharjee, ¶126.

Callender describes “kernel mode” as a “processing *context*” that the “vendor specific library” (or “PMI library”) runs in. Callender, 6:45-56; Bhattacharjee, ¶127. Furthermore, POSAs understood that the “kernel” of Callender's OS executes in Callender's “kernel mode.” Callender, 1:40-42 (“Because the [OS] kernel acts as a gatekeeper to computer resources, direct access to resources is generally limited to kernel mode processes.”); Bhattacharjee, ¶128. Callender also notes that “microprocessors” have “processing modes to support the distinctions between user mode processes and kernel mode processes.” Callender, 1:44-47. Callender further notes that “application processes run within user mode.” Callender, 1:32-34. Thus, POSAs understood Callender's applications (*see supra* §V.B.1.a) run in user mode, not kernel mode. Bhattacharjee, ¶128. Alternatively, implementing Callender such that applications did not run in kernel mode would have been obvious, because it was conventional for applications not to

run in kernel mode, as the '058 patent admits. Callender, 1:32-35; EX1001, 6:33-36; Bhattacharjee, ¶129. Thus, Callender's "*kernel mode*" meets the parties' litigation construction. See EX1058, 6; Bhattacharjee, ¶130.

Therefore, Callender's OSCSEs are *for running in kernel mode using said processor*. Bhattacharjee, ¶131.

**d. [1C]**

**i. [1C.1] "c) a shared library having shared library critical system elements (SLCSEs) stored therein..."**

The '058 patent states that a DLL is an example of a "shared library," as POSAs understood. EX1001, 2:45-50 ("A shared library or dynamic linked library (DLL) refers to an approach..."); EX1017, 3:53-61 (referring to prior-art "shareable program module such as a DLL (dynamic link library)"); EX1097, 18:31-46 (explaining that with a DLL, "the same block of library code can be *shared* between several tasks"); EX1015, 959-966; Bhattacharjee, ¶132.

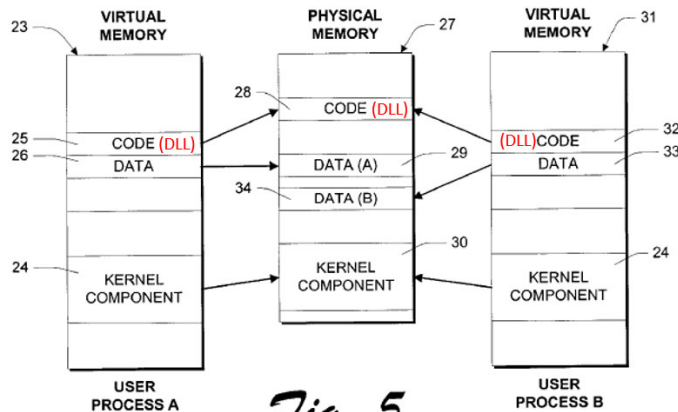
Callender's "example hardware driver model" includes a "user mode library" "implemented as a dynamic link library ('DLL')"; thus, Callender's "user mode library" is a *shared library*. Callender, 4:51-5:18, Fig. 2A; Bhattacharjee, ¶¶133-134.

Furthermore, the '058 patent's specification defines a "*shared library*" as:

An application library code space shared among all user mode applications. The code space is different than that occupied by the

kernel and its associated files. The shared library files are placed in an address space that is accessible to multiple applications.

EX1001, 6:49-53. “Code space” refers to physical memory space. EX1001, 3:39-43 (“same physical memory space, that is, shared code space”); Bhattacharjee, ¶¶135. POSAs understood that a DLL satisfies the ’058 patent’s specification’s above-quoted definition of “shared library,” which Google’s litigation construction adopts (EX1059, 16). *See, e.g.*, EX1017, 2:28-31, 3:52-4:32, Fig. 5 (describing prior art “shareable” “DLL” located in a physical memory space, *i.e.*, code space, that is separate from that occupied by the kernel and that is mapped to a “code portion” in the “virtual address space” of two different applications); EX1097, 18:31-46; EX1015, 959-966; Bhattacharjee, ¶¶136-138.



*Fig. 5*  
*Prior Art*

**EX1017, Fig. 5**

Callender’s “user mode library” also meets VirtaMove’s litigation construction, which requires “an application library whose code space is shared

among all user mode applications” (EX1068, 11), for the following reasons.

Bhattacharjee, ¶139. First, POSAs understood that the shared physical memory space into which the DLL is loaded is the DLL’s (library’s) code space.

Bhattacharjee, ¶140; EX1017, 2:28-31, 3:52-4:32; EX1001, 3:39-43; EX1067, 19 (VirtaMove arguing that “‘code space’ is a space occupied by code”). Second, POSAs understood that this code space is shared among all user mode applications in the same manner described in the ’058 patent. The ’058 patent says the “manner” in which “the code” for an SLCSE “is shared by all applications on the same compute platform” is that “all applications may physically execute the same set of instructions” representing an SLCSE that reside in “the same physical memory space, that is, shared code space.” EX1001, 3:30-44; Bhattacharjee, ¶141. Likewise, as discussed above, POSAs understood that the code representing a DLL is stored in a memory location that all user-mode applications may access to execute the same code. Bhattacharjee, ¶141; EX1017, 2:28-31, 3:52-4:32, Fig. 5; EX1097, 18:31-46; EX1015, 959-966.

Callender discloses both user-mode and kernel-mode implementations of “operations” for “sending” and “receiving” information to/from the HCA adapter. *Supra* §V.A. These “operations” are *CSEs* for the reasons explained *supra* §V.B.1.c.ii ([1B.2]). The user-mode “operations” for sending and receiving are also *CSEs* for an additional, independent reason. As noted *supra* §V.A, the

“sending” and “receiving” “operations” in Callender’s “user mode library” utilize the “kernel-bypass IO [input/output] features.” Callender, 5:15-18, Fig. 2A; Bhattacharjee, ¶¶142-143. The ’058 patent identifies a “kernel bypass” “network optimization[.]” feature as an example of a CSE. EX1001, 6:11-26; Bhattacharjee, ¶143.

The user-mode implementations of the “sending” and “receiving” “operations,” which are CSEs as discussed above, are implemented as functions contained in Callender’s “user mode library.” *Supra* §V.A; Bhattacharjee, ¶144; EX1065, 1:46-50. Because the functions implementing the sending and receiving operations are in a shared library, *i.e.*, the “user mode library,” these functions are SLCSEs; thus, Callender’s “user mode library” is a *shared library having SLCSEs stored therein*. EX1001, 2:45-47 (“In accordance with this invention, SLCSEs are placed in shared libraries[.]”); Bhattacharjee, ¶¶144-145.

**ii. [1C.2] “for use by the plurality of software applications in user mode...”**

Callender’s “applications” (also called “application programs”) that interact with the HCA adapter are *the plurality of software applications*. *Supra* §V.B.1.a ([1Pre]). POSAs understood that Callender’s “user mode library” is *for use by* these *applications*, for two independent reasons. Bhattacharjee, ¶146.

*First*, as noted *supra* §V.B.1.d.i, Callender’s “user mode library” is a DLL. POSAs understood that DLLs are shared by multiple applications. EX1097, 18:31-

46 (“well known” that “[a] DLL is a library...that is dynamically linked to application programs.... This means that the same block of library code can be shared between several tasks.”); EX1015, 959-960 (“...one purpose of dynamic link libraries is to provide functions and resources that can be used by many different programs...”), 965-66; Bhattacharjee, ¶147.

*Second*, POSAs understood that Callender discloses multiple “applications” using the user mode library. Callender’s “user mode library” provides “access[]” to the “HCA Adapter.” Callender, 4:54-63. Callender’s “Background” section notes that “some hardware *adapters* support enforcement of security measures...so that user mode *applications* may access the hardware directly.” Callender, 1:53-61. Additionally, Callender discloses that the user mode library enables user-mode access to the HCA adapter by bypassing the kernel. *See* Callender, 5:15-18 (referring to “kernel-bypass IO features offered by the user mode library”), FIG. 2A (annotated below, showing arrow going directly from “Vendor Specific Library” in “User-Mode Library” to “HCA Adapter”). Thus, POSAs understood that the HCA adapter is a “hardware adapter” that “*applications*” (plural) “may access...directly” using the “user mode library.” Callender, 1:53-61; *see also* Callender, 4:54-63; Bhattacharjee, ¶148; EX1062, 1:11-39 (“Background” noting that “Infini[B]and...allows *applications* running on *a* computer...to send messages”).

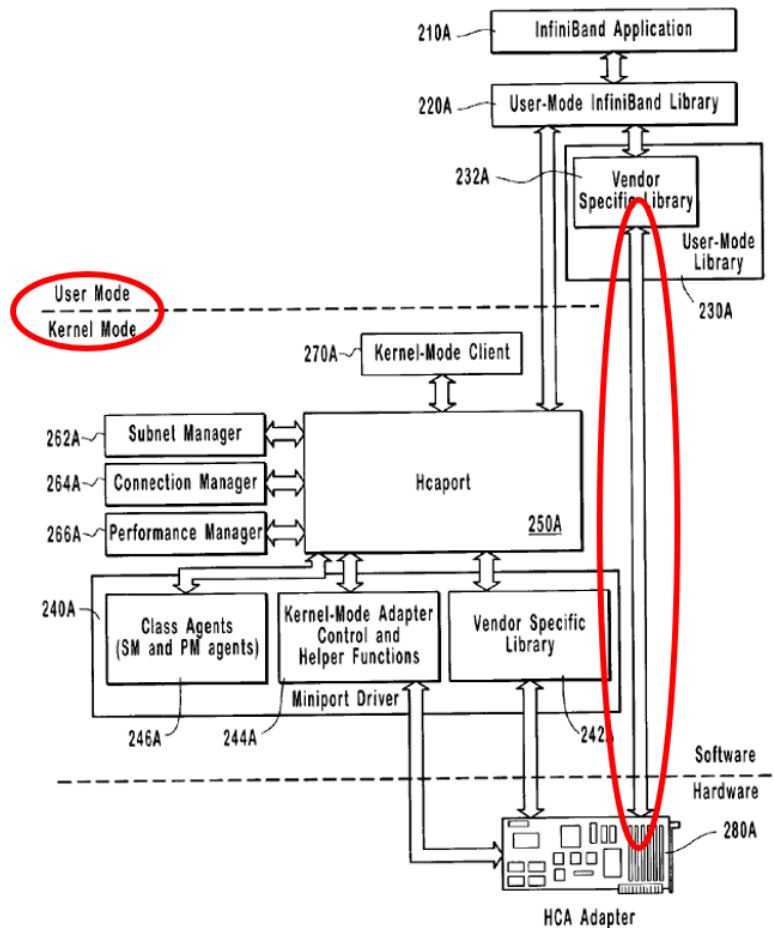


Fig. 2A

The functions in the “user mode library” that implement the “sending” and “receiving” “operations” are claimed *SLCSEs*. *Supra* §V.B.1.d.i. Because the “user mode library” is *for use by the plurality of software applications* as discussed above, POSAs understood that the sending and receiving functions in the library are likewise *for use by the plurality of software applications* that interact with the HCA adapter, because sending and receiving data are operations that multiple applications that interact with the HCA adapter perform. Bhattacharjee, ¶149.

Callender's Figures 2A and 2C show an arrow going between the "InfiniBand Application" and the "User-Mode InfiniBand Library," and an arrow going between the "User-Mode InfiniBand Library" and the "User-Mode Library." This visually depicts an example "InfiniBand application 210A access[ing] host channel adapter ('HCA adapter') 280A through user mode library 230A" (Callender, 4:54-57); thus, the "sending" and "receiving" functions in the "user-mode library" are for use by the plurality of applications. Bhattacharjee, ¶150. POSAs understood that the application uses an SLCSE in the "user-mode library" by calling through the "User-Mode InfiniBand Library" to access the user-mode library and the SLCSE. Bhattacharjee, ¶151. Indeed, Callender describes the "user-mode library" as an "extension" of the "user-mode InfiniBand library." Callender, 5:9-11. Moreover, it was well-known for applications to use functionality in a library via another library. Bhattacharjee, ¶151; EX1030, [0034]; EX1031, 1:53-60; EX1061, 3:27-35, 22:27-29. The '058 patent too describes an embodiment where user-mode CSE functionality is implemented via different libraries that work together. EX1001, 8:4-13, 8:16-26, 9:43-46, Figs. 3-4; Bhattacharjee, ¶151.

The SLCSEs are in Callender's "*user mode* library" and are used by applications that "run within *user mode*." Callender, 4:54-57, 1:33-35; Bhattacharjee, ¶152. Callender's "user mode" meets the '058 patent's definition,

and the parties' litigation definition, of *user mode*, which is “[t]he context in which applications execute,” for two reasons. EX1001, 6:31; EX1058, 6; Bhattacharjee, ¶153. First, Callender describes “user mode” as a “processing *context*” that the “vendor specific library” (or “PMI library”) runs in. Callender, 6:45-56; EX1003, ¶154. Second, POSAs understood that Callender’s applications execute in “user mode.” Callender, 1:33-35 (“Generally, application processes run within user mode...”); Bhattacharjee, ¶155.

Therefore, POSAs understood that the SLCSEs are *for use by the plurality of software applications in user mode*. Bhattacharjee, ¶156.

e. [1D]

i. [1D.1] “i) wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and...”

The SLCSEs are *stored in the shared library, i.e.*, the “user mode library,” as discussed *supra* §V.B.1.d.i ([1C.1]).

The outer boundary of the scope of “*functional replicas*” is unclear to POSAs. EX1059, 18-20. However, the ’058 patent says, “the term replica used herein is meant to denote a CSE having similar attributes to, but not necessarily and preferably not an exact copy of a CSE in the [OS].” EX1001, 1:65-2:1.

Additionally, the patent says, “[t]he CSE library includes replicas or substantial functional equivalents or replacements of kernel functions. The term replica, shall

encompass any of these meanings, and although not a preferred embodiment, may even be a copy of a CSE that is part of the OS.” EX1001, 8:27-32. Thus, at a minimum, a *functional replica* of an OSCSE encompasses a copy of an OSCSE. Bhattacharjee, ¶¶157-158.

In Callender, the kernel-mode sending and receiving operations are *OSCSEs* (*supra* §V.B.1.c.ii ([1B.2])), while the functions in the user-mode library implementing the sending and receiving operations are *SLCSEs* (*supra* §V.B.1.d.i ([1C.1])). Callender says, “***the same source code*** may be written for a kernel mode communication operation and a user mode communication operation.” Callender, 4:30-32; Bhattacharjee, ¶159. Furthermore, Callender says this source code can be in a “static library derived from a common source.” Callender, 4:33-34. POSAs understood that if the source code for a given operation is in a static library, then the executable code for the operation is copied into both the user-mode and kernel-mode implementations of the operation. EX1001, 6:54-55; EX1015, 959-961; EX1003, ¶159. Thus, Callender discloses that each *SLCSE* includes a copy of the executable code for the operation that is the corresponding OSCSE, and hence is a *functional replica* of the corresponding OSCSE. Bhattacharjee, ¶160; *PLR*, IPR2024-00209, Paper 9, 39 (Board may determine that teaching is within claim boundaries even if boundaries are unclear).

Furthermore, even if the SLCSEs were not copies of the corresponding OSCSEs (*see infra* §V.B.18 (claim 18)), the relationship between Callender’s user-mode InfiniBand sending/receiving operations and the corresponding kernel-mode operations parallels the example of “replication” of OSCSEs by SLCSEs provided in the ’058 patent. *See* EX1001, 6:22-55. Specifically, the specification says “[e]mbodiments of the invention enable the replication of critical system elements normally found in an operating system kernel,” and describes an “example” in which the “CSE that is part of a shared library” is a “TCP/IP stack,” and where the “TCP/IP services in the CSE are the same as those provided in the Linux [operating system] kernel.” EX1001, 6:22-55; Bhattacharjee, ¶161. POSAs understood that InfiniBand, like TCP/IP, is a networking protocol. Additionally, the user-mode InfiniBand sending/receiving operations provide “services” to the user-mode applications that are the “same as those provided in the [operating system] kernel” (EX1001, 6:22-55) by the kernel-mode operations; indeed, Callender discloses that both the user-mode and kernel-mode InfiniBand sending/receiving operations are invoked via a “common interface.” Callender, Abstract, 2:32-39, 3:29-34, 4:23-40, 7:5-26; Bhattacharjee, ¶161. Thus, although the full scope of “similar attributes” in the ’058 patent’s definition of “replica” is unclear to POSAs (EX1059, 19-20), POSAs would have understood that Callender’s user-mode functions for performing the InfiniBand sending/receiving

operations, which are claimed *SLCSEs* (*supra* §V.B.1.d.i ([1C.1])), have the same relevant attributes (*i.e.*, services provided) as their kernel-mode counterpart operations, which are claimed *OSCSEs* (*supra* §V.B.1.c.ii ([1B.2])), in the same manner described in the '058 patent, and are thus *functional replicas of OSCSEs* even though the outer boundaries of this language are unclear.

The user-mode sending/receiving functions are also *functional replicas of OSCSEs* under VirtaMove's litigation construction, "substantial functional equivalents or replacements of kernel functions." EX1067, 22. As noted above, Callender discloses that the user-mode functions include copies of the executable code for the corresponding kernel-mode operations, and VirtaMove has argued in litigation that "the term replica' specifically in the context of *functional replicas*" includes "copies of OSCSEs." EX1067, 23 (emphasis original). Furthermore, each Callender SLCSE replicates the function of the corresponding OSCSE: the user-mode sending function performs the same "operation" (sending data via the HCA adapter) as the kernel-mode counterpart operation, and the user-mode receiving function likewise performs the same "operation" (receiving data via the HCA adapter) as the kernel-mode counterpart operation. Bhattacharjee, ¶162. Additionally, the user-mode sending and receiving functions replace the corresponding kernel-mode operations for applications using the user-mode library. *Supra* §V.B.1.d.ii ([1C.2]); Bhattacharjee, ¶162.

During prosecution, the '058 patent's applicants distinguished O'Rourke (EX1009) by arguing that a "functional replica" cannot be a mere "intermediary" to the kernel. EX1002, 323; Bhattacharjee, ¶163. POSAs understood that Callender's user-mode sending and receiving functions (the *SLCSEs*) are not mere intermediaries. Callender's user-mode sending and receiving functions implement complete operations by executing their own source code, which may be the same as the source code of their kernel-mode counterpart operations. Callender, 3:53-55 ("Having *corresponding operations implemented in both* user mode implementation 130 and kernel mode implementation 140 is desirable"), 4:29-36 ("the same source code may be written" for both kernel- and user-mode operations), 7:11-26 (describing benefits of Callender's "single implementation" of common operations), claim 1; Bhattacharjee, ¶164. Moreover, the user-mode functions perform operations that "bypass" the kernel, and as such are not intermediaries to the kernel. Callender, 5:15-18; *supra* §V.A. In contrast, O'Rourke's user-mode "proxies," which the '058 patent's applicants distinguished, invoke kernel-level code. EX1002, 323 (applicants citing EX1009, 6:12-19 and 10:12-33); EX1009, 6:12-19 ("primary object" is "user mode proxies for kernel mode filters"), 10:12-33 (describing "prox[ies]" as invoking "kernel mode" components that perform operations); Bhattacharjee, ¶164.

Thus, Callender's SLCSEs are *functional replicas of the OSCSEs*.

Bhattacharjee, ¶165.

**ii. [1D.2] “are accessible to some of the plurality of software applications and...”**

The *plurality of software applications* that access Callender's HCA adapter utilize the “sending” and “receiving” functions in the “user mode library.” *Supra* §V.B.1.d.ii ([1C.2]). These functions are *SLCSEs*. *Supra* §V.B.1.d.i ([1C.1]). POSAs understood that these *SLCSEs* are *accessible to* the applications, for multiple independent reasons, as explained below. Bhattacharjee, ¶166.

First, Callender's “user mode library” containing the SLCSEs is a “DLL.” Callender, 5:14-15; *supra* §V.B.1.d.i ([1C.1]). POSAs understood that a DLL's purpose is to serve as a “library” accessible to multiple applications. Bhattacharjee, ¶167; EX1015, 959-960, 965-966; EX1097, 18:31-46; EX1032, 1:28-31. As discussed *supra* §V.B.1.d.ii ([1C.2]), POSAs understood that in Callender, the plurality of applications (such as the exemplary “InfiniBand application 210A” shown in Figure 2A) use SLCSEs in the “user-mode” library by calling through the “User-Mode InfiniBand Library” to access the user-mode library and the SLCSEs. Thus, because the applications access the SLCSEs by calling through the “User-Mode InfiniBand Library,” the SLCSEs are *accessible to* the applications. Bhattacharjee, ¶168.

POSAs understood that a function is a predefined set of instructions that carry out a specific action. Bhattacharjee, ¶169; EX1007, 1:22-29; *supra* §V.A. POSAs also understood that executing the instructions that make up a function involves the processor reading those instructions from memory. Bhattacharjee, ¶168; EX1054, 1:14-17 (“A conventional computer system includes a processor that retrieves or reads program instructions...and executes the program instructions...”); EX1055, 1:22-27; EX1056, 4:18-21; Bhattacharjee, ¶169. Therefore, POSAs understood that when the applications use the SLCSEs, which are functions as discussed above, the processor that is executing the application reads the executable instructions that make up the SLCSEs; thus, Callender’s SLCSEs meet Google’s litigation construction of “*accessible*,” which requires that “two or more of the plurality of the software applications can read SLCSEs stored in the shared library.” EX1059, 17; Bhattacharjee, ¶170.

Callender’s SLCSEs also meet VirtaMove’s litigation construction, requiring that “some of the plurality” of the applications can “use” SLCSEs. EX1059, 17-18. POSAs understood that by reading and executing the instructions making up the SCLSEs as discussed above, the applications use the SCLSEs. Bhattacharjee, ¶171; *see also* EX1067, 21 (VirtaMove conceding “access” in claim encompasses reading). Furthermore, because the *plurality* of applications

collectively use each SLCSE, at least *some* of the applications use SLCSEs.

Bhattacharjee, ¶171.

Therefore, the SLCSEs are *accessible to* at least *some of the plurality of software applications*. Bhattacharjee, ¶172.

- iii. [1D.3] **“when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications”**

The '058 patent says SLCSEs “*form a part* of at least some of the software applications **by being linked** thereto.” EX1001, 2:10-15; *see also* claim 16 (reciting “SLCSEs form a part of at least some of the plurality of software applications, by being linked thereto”); Bhattacharjee, ¶173. The parties’ agreed litigation construction of “*forms a part of...*” adopts the “*linked to*” requirement. EX1058, 6.

Callender’s SLCSEs are in a “dynamic *link[ed]* library” (“DLL”). Callender, 5:13-15; *supra* §V.B.1.d.i ([1C.1]). POSAs understood that a “DLL is a library...that is dynamically linked to application programs.” EX1097, 18:31-34; EX1033, 4:47-49; Bhattacharjee, ¶¶174-175. POSAs also understood that a DLL exists on a computer in the form of “a file that gets mapped into a process’s address space such that any functions in the DLL appear to be *part of* the process.” EX1018, 745; EX1015, 965-966; EX1017, 3:53-61, Fig. 5; Bhattacharjee, ¶175. POSAs therefore understood that when an application *accesses* a Callender SLCSE

(see *supra* §V.B.1.e.ii), the SLCSE *forms a part of* that application because the DLL is linked to the application and gets mapped into its address space.

Bhattacharjee, ¶175. Thus, *when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications.* EX1003, ¶176.

f. [1E]

- i. [1E.1] **“ii) wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications and...”**

“*Instance*” is not a term of art specific to CSEs, but rather customarily refers to an object or a copy of an object. Bhattacharjee, ¶177; EX1034, 186; EX1035, 276. The’058 patent’s specification does not define an *instance* of a SLCSE; however, the specification says the “intent” of having applications “link” to the SLCSE library “is to *provide an application* with a unique *instance of a CSE.*” EX1001, 3:20-29; Bhattacharjee, ¶178.

The specification also does not explain what it means for an *instance* of an SLCSE to *run in a context of an application*. However, the specification says the “ability to allow a CSE to execute in the same context as an application...allows...an ability to deploy multiple instances of a CSE.” EX1001,

1:46-54; Bhattacharjee, ¶179. The specification also says SLCSEs *run in a context* of an application because the SLCSEs reside in a “shared library” that is linked to the application. EX1001, 5:22-26 (“Embodiments of the invention enable the **replication** of critical system elements.... These **replicated CSEs** are then able to **run in the context of a software application**. Critical system elements are **replicated through the use of shared libraries.**”), 8:27-33 (“The **CSE library** includes replicas or substantial functional equivalents or replacements of kernel functions.... These functions can be directly called by the applications 42 and **as such can be run in the same context as the applications.**”), 9:15-20 (“...some system elements that are critical to the operation of a software application are replicated...into user mode **in the same context** as that of the application. These system elements are **contained in a shared library. As such they are linked** to a software application....”); Bhattacharjee, ¶179.

Callender discloses applications accessing SLCSEs using the same technique as in the '058 patent, which POSAs understood would yield the same result, as explained below. Bhattacharjee, ¶180. Callender's functions in the “user mode library” for performing the “sending and receiving” “operations” are claimed *SLCSEs*. *Supra* §V.B.1.d.i ([1C.1]). As also discussed *supra* §V.B.1.d.i, the “user mode library” is a “dynamic **link** library,” which is a shared library “**link[ed]**” to applications, just as the '058 patent's SLCSE library is “linked” to applications.

Callender, 5:13-15; EX1001, 3:20-29; Bhattacharjee, ¶181. As the '058 patent itself admits, as discussed above (*see* EX1001, 5:22-25, 8:27-33, 9:15-20), the well-known consequence of linking a DLL to an application is that *an instance of each function is provided to each application from the shared library and is run in the context of that application.* Bhattacharjee, ¶181.

The '058 patent does not expressly explain how SLCSE *instances* are provided to applications *without being shared* between applications. However, the patent states that although there is one copy of the computer “instructions” (or “code”) representing an SLCSE that is “shared by all applications,” each application “uses [its own] separate data area[]” when executing the instructions representing the SLCSE. EX1001, 3:30-42; Bhattacharjee, ¶182. “In this manner,” “CSEs are not shared among applications even though the code is shared.” EX1001, 3:42-44; Bhattacharjee, ¶182.

POSAs understood that in Callender, applications access SLCSEs in the same manner described in the '058 patent. SLCSEs are provided to Callender’s applications in a DLL, as discussed *supra* §V.B.1.d.i ([1C.1]). POSAs understood that the different applications share the code representing the “sending” and “receiving” functions but use their own data areas in memory when executing the code, since this was the known way for DLLs to be used. Bhattacharjee, ¶183; EX1015, 965-966 (“Each process has its own address space for any data the DLL

uses.”); EX1018, 745 (“A DLL...gets mapped into a process’s address space such that any functions in the DLL appear to be part of the process.”).

Alternatively, POSAs would have found it obvious to implement Callender such that applications share the code for functions in the DLL containing the SLCSEs (*i.e.*, the user-mode library) but use application-specific data areas when executing the code. The ’058 patent admits that such an implementation was “common practice,” and POSAs understood that using this technique would beneficially allow multiple applications to invoke the same function; POSAs would also reasonably have expected success in implementing this common practice. EX1001, 3:30-44 (“This sharing of code is common practice. All applications currently share code.... Each application has its own unique data space. This indivisible data space ensures that CSEs are unique to an application[.]”); EX1015, 965-966; EX1018, 745; Bhattacharjee, ¶184.

Thus, in Callender, *an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications.* Bhattacharjee, ¶185.

- ii. [1E.2] “where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function, and...”

POSAs understood that the *plurality of software applications run under the operating system*, or at least would have found it obvious to implement Callender this way, because this was the standard relationship between applications and the OS. Bhattacharjee, ¶186; EX1036, 1:13-14 (“A computer system often concurrently runs several applications under an operating system.”); EX1037, [0003].

Callender’s user-mode “sending” and “receiving” functions, *i.e.*, the *SLCSEs*, are in the “user mode library,” which is a DLL. *Supra* §V.B.1.d.i ([1C.1]). Because DLLs are shared libraries, POSAs understood that multiple applications running under the computer’s OS can call the same function in a DLL, *i.e.*, that multiple applications *have use of* the same function *for performing same function* in a DLL. Bhattacharjee, ¶187; EX1015, 959-960, 965-966; EX1097, 18:31-46.

The ’058 patent says an application obtains use of a “*unique instance*” of an SLCSE by “linking” to the “library containing the [SLCSE].” EX1001, 3:25-29; Bhattacharjee, ¶188. The patent also states that because “[e]ach application has its own unique data space...[SLCSEs] are unique to an application.” EX1001, 3:36-

39; Bhattacharjee, ¶189. In Callender, each application is linked to the “user mode library” containing the SLCSEs. *Supra* §V.B.1.e.iii ([1D.3]). Furthermore, each application has its own unique data space in memory, as discussed *supra* §V.B.1.f.i. Thus, in Callender, *at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function.*

- g. [1F] “iii) wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously.”**

The functions for “sending and receiving information” in Callender’s “user mode library 230A” are claimed *SLCSEs*. *Supra* §V.B.1.d.i ([1C.1]). POSAs understood, or would at minimum have found obvious, that functions are invoked by reference to some *predetermined* identifier, *e.g.*, the *function* name, as this was the typical way to invoke functions contained in shared libraries and was within a POSA’s skill. Bhattacharjee, ¶190; EX1069, 10:55-58 (“Code module 126a includes executable code that includes instructions that reference functions called ‘foo()’ and ‘goo(),’ neither of which are part of code module 126a.”). Thus, each SLCSE is, and thus *relates to*, a *predetermined function*. Bhattacharjee, ¶190. Additionally, these SLCSEs are *provided to the plurality of the software*

*applications*, including the *first* and *second* of the plurality, because they are in a DLL accessed by the applications. *Supra* §V.B.1.d.ii ([1C.2]); Bhattacharjee, ¶190.

Furthermore, the SLCSEs are provided to the applications *for running instances of* a SLCSE; the SLCSEs are in a DLL, and when an application invokes an SLCSE from the DLL, an *instance* of the SLCSE *runs* in that application’s context. *Supra* §V.B.1.f.i ([1E.1]); Bhattacharjee, ¶191.

Finally, the SLCSEs are provided to the *first* and *second* of the *plurality of applications* for running the *first* and *second* instances *simultaneously*.

“*[S]imultaneous[]*” appears nowhere in the ’058 patent’s specification. Element [1F] was originally filed as dependent claim 6. EX1002, 22. During prosecution, the applicants distinguished as-filed claim 6 over Cabrero by arguing that CSEs are “replicated and **can be accessed *simultaneously*** by...an application in user mode **by way of** the application using an instance of the CSE within a shared library,” which was in “contradistinction” to Cabrero, where CSEs are “independent applications that neither reside in the OS nor in shared libraries.” EX1002, 241-242, 244; Bhattacharjee, ¶192. The applicants also said “claim 6 **allows** different instances of an SLCSE...to be executed simultaneously by another one or more applications.” EX1002, 244. Thus, according to the applicants, Element [1F]’s “*for running...simultaneously*” is met by allowing two different applications to

have simultaneous access to an SLCSE in a shared library. Bhattacharjee, ¶192. As discussed above, Callender’s SLCSEs are operations provided by a DLL, and POSAs understood that one purpose of DLLs was to allow multiple applications to access the same code in the DLL “*simultaneously*.” EX1015, 959-960, 965-966 (“Multiple applications can use the same DLL *simultaneously*”); EX1097, 18:31-46. Bhattacharjee, ¶¶192-193.

## 2. Claim 2

In Callender, *first* and *second* applications can *simultaneously* each run an *instance* of an SLCSE. *Supra* §V.B.1.g ([1F]). The SLCSE is *stored in a shared library*. *Supra* §V.B.1.d.i ([1C.1]). The *first* and *second* instances together comprise *multiple instances*. Bhattacharjee, ¶194.

If VirtaMove argues that “*multiple*” requires more than two instances, this would have been obvious, because POSAs understood that three or more applications running on a single computer can all use the same InfiniBand hardware. EX1062, 1:11-39; Bhattacharjee, ¶195; *supra* §V.B.1.a ([1PRE]) (Callender discloses plural applications using the HCA adapter in a computer system). POSAs also understood that there were at least three different types of applications for which InfiniBand could be used. *E.g.*, EX1063, 1:15-21 (“InfiniBand products are ideally suited for clustering, I/O extension, and native attachment”); EX1071, [0004] (“InfiniBand based networks are designed to satisfy

bandwidth-hungry network applications, such as those combining voice, data, and video on the Internet.”). POSAs would have been motivated to run at least three different InfiniBand applications on the same computer to achieve the efficiency benefit of using the same hardware for multiple tasks, and would reasonably have expected success given that Callender itself discloses running multiple applications, as noted above. Bhattacharjee, ¶196.

POSAs understood that when the SLCSE instances are run (whether simultaneously or not), *i.e., in operation*, they run *within the operating system*. The '058 patent does not explain what it means for an SLCSE instance to run “*within*” the OS. However, the patent refers to “**applications** [that] run[] under the [OS]” and that have “use of a unique **instance** of a corresponding critical system element.” EX1001, Abstract, 2:18-21, 2:35-40; Bhattacharjee, ¶197. Furthermore, POSAs understood that the prior art sometimes referred to typical application execution as applications running “within” an OS. EX1038, 1:18-20 (referring to “conventional computer systems having...software applications running within the [OS]”); EX1039, 1:6-15; Bhattacharjee, ¶197. Thus, POSAs understood that claim 2’s reference to “*instances of an SLCSE...run[ning]...within the [OS]*” encompasses applications running within the OS and using their own instances of SLCSEs. Bhattacharjee, ¶198.

In Callender, the first and second applications are provided with their own unique instances of an SLCSE. *Supra* §V.B.1.f.i-V.B.1.f.ii ([1E]); Bhattacharjee, ¶199. POSAs understood that in Callender, the applications *run within* the OS because the application’s processes are “*run within* [the OS’s] user mode.” Callender, 1:29-35 (“Many **operating systems provide** at least two process modes,” namely “user mode” and “kernel mode,” and [g]enerally, **application processes run within user mode**”), claim 1; Bhattacharjee, ¶199; EX1018, 3 (“An **operating system**...provides a basis for application programs[.]”), 5 (“An operating system...manages the execution of user programs[.]”).

Thus, in Callender’s computer system, *in operation, multiple instances of an SLCSE stored in the shared library run simultaneously within the operating system*. Bhattacharjee, ¶200. Additionally, Callender’s *SLCSEs* replicate the functionality of CSEs in the OS (*i.e.*, *OSCSEs*) (*supra* §V.B.1.e.i [1D.1]) and are in shared libraries from where they can be run simultaneously (*supra* §V.B.1.g [1F]), in contrast to the ’058 patent applicants’ characterization of Cabrero when distinguishing claim 2 during prosecution. EX1002, 242; Bhattacharjee, ¶201.

### 3. Claim 3

Callender’s “operations for sending and receiving information from adapter 150” in the “*kernel* mode implementation” are claimed *OSCSEs*. Callender, 3:40-4:16; *supra* §V.B.1.c.ii ([1B.2]). These *OSCSEs* *correspond to* and are *capable of*

*performing the same function as the SLCSEs*, which are the functions in the “user mode library” for performing the “operations” for “sending and receiving information.” *Supra* §V.B.1.d.i ([1C.1]). *See also* Callender, 3:53-4:3 (explaining desirability of Callender’s techniques with “**corresponding operations** implemented in **both** user mode...and kernel mode”); Bhattacharjee, ¶202. The OSCSEs are part of the OS kernel (*supra* §V.B.1.c.ii ([1B.2])); thus, the OSCSEs *remain in the [OS] kernel*. Bhattacharjee, ¶203; EX1002, 242 (applicants saying claim 3 “reinforces the fact that OSCSEs that correspond to SLCSEs remain within the kernel”).

#### 4. Claim 4

##### a. [4A]

“*[E]xclusive*” appears nowhere in the ’058 patent’s specification. However, the specification refers to “an application” being “provided” a “unique instance” of a SLCSE via “link[ing]” to the SLCSE library. EX1001, 3:25-29; Bhattacharjee, ¶204. In Callender, the *SLCSEs* are in a “DLL” that the *applications* “link[]” to. *Supra* §V.B.1.e.iii ([1D.3]). An instance of an SLCSE is *provided to one of the plurality of software applications* when the application calls the SLCSE from the linked library. *Supra* §V.B.1.f.i ([1E.1]). Additionally, when Callender’s *SLCSEs* are called by an application, the SLCSEs run in the context of the application, and an executing SLCSE is not shared. *Supra* §V.B.1.f.i ([1E.1]). That is, while the

DLL containing the source code for SLCSEs is shared among applications, an instance of an SLCSE that is executed from that source code runs in the exclusive context of an individual application in Callender, just as in the '058 patent.

Bhattacharjee, ¶205; EX1001, 3:30-44. Thus, the application *has exclusive use of* the SLCSE instance that is *provided* to the application by virtue of the application calling the SLCSE. Bhattacharjee, ¶205.

If VirtaMove argues that claim 4 requires an SLCSE to only be usable by one application, this would be incorrect. Claim 4 depends from claim 1, which requires that the executable SLCSEs (as opposed to an executing SLCSE) are in a *shared* library and are usable by the *plurality* of applications (*see* Elements [1C.1]-[1C.2]). Bhattacharjee, ¶¶206-207 (citing EX1032, 472, 355).

**b. [4B]**

POSAs understood a “system call” is a mechanism for “an application to invoke a kernel service.” EX1018, 463; Bhattacharjee, ¶208.

Callender’s “user-mode library” has “kernel-bypass IO features.” Callender, 5:15-18. The user-mode sending and receiving functions in the user-mode library that perform the sending and receiving operations use these kernel-bypass IO features. *Supra* §V.A. However, although the library has kernel-bypass features, and although Callender’s Figures 2A and 2C show an arrow going between the “HCA Adapter” and the “user-mode library,” POSAs understood that operations in

the user-mode library would still access kernel services. Callender notes that “some software drivers bypass kernel mode for **certain operations**,” but Callender never refers to drivers bypassing the kernel for all operations. Callender, 1:59-61; Bhattacharjee, ¶209. As explained below, POSAs understood that the user-mode sending and receiving functions, which are *SLCSEs* (*supra* §V.B.1.d.i ([1C.1])), use kernel-bypass features (*e.g.*, within the sending/receiving operation specifically) but also involve related operations for which the kernel is *not* bypassed—*e.g.*, queue setup, interrupt setup, and event notification.

#### **i. Queue Setup**

Callender’s “**user-mode library**” includes the “**Vendor Specific Library**,” also called the “PMI” or “process mode independent” library. Callender, 5:20-28, 6:45-48, Fig. 2A (annotated below); Bhattacharjee, ¶210; *supra* §V.A.

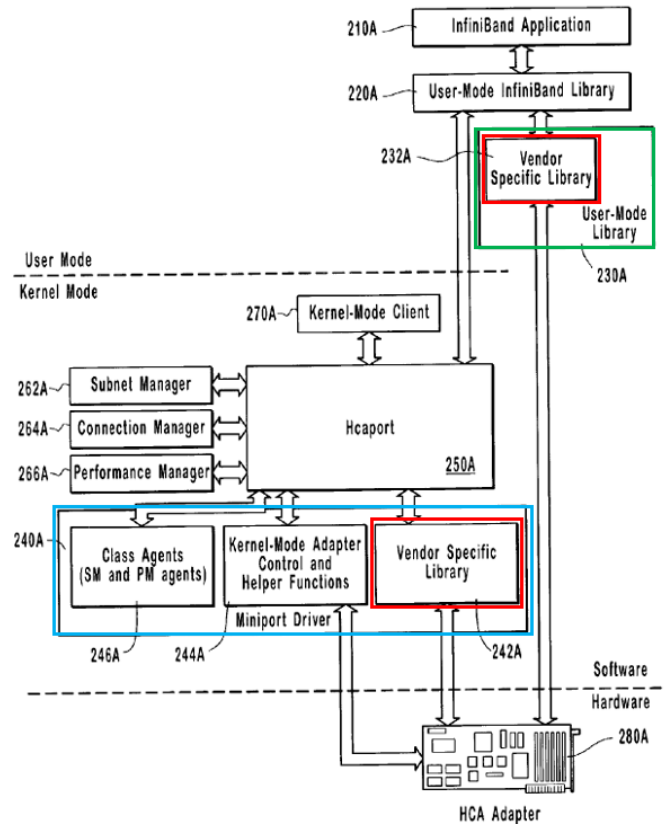


Fig. 2A

The functionality in the PMI library includes “the **user mode side** of protection domain / completion queue / work queue (‘PD/CQ/WQ’) creation and destruction.” Callender, 6:56-60. POSAs understood that the “completion” and “work” queues are used by the user-mode library (“user mode side”) to send data to and receive data from the adapter. Bhattacharjee, ¶211; Callender, 6:56-7:1 (“IO requests directly manipulate an allocated queue pair”); EX1013, [0008]-[0010] (explaining use of completion queues and work queues in InfiniBand).

Callender explains that “[t]he PMI routines for PD/CQ/WQ creation...operate in a two step process. The first step acquires needed resources for allocation of a queue pair by the miniport in kernel mode. The second step checks the results and if a request fails, frees resources allocated to the process.” Callender, 6:57-67. POSAs understood from this explanation that when the PMI library tries to create a queue, the following operations happen (Bhattacharjee, ¶212):

1. The library acquires resources (*i.e.* the “first step” of the “user mode side” (Callender, 6:56-66));
2. The “miniport in kernel mode” (Callender, 6:64-67) attempts to allocate the queue pair; and
3. The library checks the results (*i.e.*, the “second step” of the “user mode side” (Callender, 6:56-67)).

POSAs also understood that a “miniport” is a low-level driver that “directly manages a network interface card (NIC) and provides an interface to higher-level drivers.” EX1040, 15:4-12; Bhattacharjee, ¶213.

In Callender, an element “in kernel mode” is in the “kernel,” *i.e.*, the *OS kernel*. *Supra* §V.B.1.c.ii ([1B.2]). Thus, POSAs understood that Callender’s “miniport in kernel mode” (6:66) is in the *OS kernel*. At minimum, POSAs would have found it obvious to implement Callender such that the miniport is part of the

*OS kernel*, because it was customary to implement components like the miniport, which operates in kernel mode and provide interfaces to hardware, as part of the OS kernel. Bhattacharjee, ¶214; Callender, 1:40-43 (noting it was typical that “the [OS] kernel acts as a gatekeeper to computer resources”); EX1018, 456; 696-697; EX1019, 1:16-34; EX1027, 8:64-9:5; EX1041, 8:56-64.

Where the miniport is part of the *OS kernel*, the miniport’s services for allocating queues in the queue pair are *services in the OS kernel*. Furthermore, POSAs would have found it obvious to implement *accessing* these services *using a system call*, because system calls were a well-known and customary way for user-mode processes to access kernel services. Bhattacharjee, ¶215; Callender, 1:36-38 (noting in “Background”: “User processes switch to kernel mode when making *system calls*, generating an exception or fault, when an interrupt occurs, etc.”).

Finally, POSAs would have found it obvious to implement the user-mode sending and receiving functions, which are *SLCSEs* (*supra* §V.B.1.d.i ([1C.1])), to check to see if a communication queue is available and create one if needed; such an implementation would be within a POSA’s skill because it involves known, simple programming techniques, like conditional instructions. Bhattacharjee, ¶216; EX1013, [0008] (“When a consumer needs to open communications with some other entity via the IB fabric, it asks the HCA to provide the necessary transport resources by allocating a QP for its use.”). Furthermore, such an implementation

would beneficially ensure that queues needed for communication are available when needed but not allocated before they are needed. Bhattacharjee, ¶216. When the sending and receiving functions are implemented this way, they each *use a system call to access services in the OS kernel*, and thus the SLCSEs collectively *use system calls to access services in the OS kernel*. Bhattacharjee, ¶217.

## ii. Interrupt Setup and Event Notification

The '058 patent mentions initializing “interrupt handling” as an example of an SLCSE using a system call to access kernel services. EX1001, 2:64-65, 8:53-59; Bhattacharjee, ¶218.

Callender’s HCA adapter uses the InfiniBand standard. *Supra* §V.A. POSAs understood that communication under the InfiniBand standard involved generating “interrupts” that result in “events” that signify the “completion” of an operation. EX1013, [0007]-[0010]; Bhattacharjee, ¶219.

Callender’s “miniport” provides “adapter control and helper functions.” Callender, 5:31-33. These functions include “management” of “interrupts” and also include “kernel mode helper functions” including “event request management, such as the creation and destruction of the event queue and the processing of events from the event queue.” Callender, 6:13-33; Bhattacharjee, ¶220. POSAs understood, or would at least have found obvious, that the user-mode sending and receiving functions, *i.e.*, *SLCSEs* (*supra* §V.B.1.d.i (Limitation [1C.1])), rely on

the interrupt and event-request management functions in the miniport, because these functions are a normal part of sending and receiving data in the InfiniBand standard, as discussed above. The services provided by the miniport, including the interrupt and event-request management services, are *services in the OS kernel* for the same reasons discussed *supra* §V.B.4.b.i with respect to the queue setup operation. Furthermore, POSAs would have found it obvious to implement *accessing these services using a system call*, as also discussed *supra* §V.B.4.b.i. When the sending and receiving functions are implemented this way, they each *use a system call to access services in the OS kernel*, and thus the SLCSEs collectively *use system calls to access services in the [OS] kernel*. Bhattacharjee, ¶221.

## **5. Claim 5**

### **a. [5A]**

A “*kernel module*” is “[a] set of functions that reside and execute in kernel mode as extensions to the [OS] kernel.” EX1001, 6:56-59. The parties’ agreed litigation construction adopts this definition. EX1058, 6; Bhattacharjee, ¶222.

Callender discloses an “Hcaport” that operates in “kernel mode” and performs tasks including “establish[ing] an operating environment that provides the miniport with operating system functionality.” Callender, 5:29-58, Fig. 2A; Bhattacharjee, ¶223. Such an element operating in “kernel mode” is in the “kernel,” *i.e.*, the *operating system kernel*. *Supra* §V.B.1.c.ii ([1B.2]). The

Hcaport is therefore a “set of functions that resides and executes in the kernel.” EX1001, 6:56-59; Bhattacharjee, ¶223. Additionally, POSAs understood that the Hcaport is an “extension” to the OS kernel (EX1001, 6:56-59) because it “adds functionality to” the kernel by “provid[ing] the miniport” in the kernel “with operating system functionality.” Callender, 5:29-65; EX1035, 203 (“extension” is a “module that adds functionality to or extends the effectiveness of a program”); Bhattacharjee, ¶224.

Having the Hcaport reside and execute in the kernel as an extension of the kernel would have also been obvious. The Hcaport operates in kernel mode and provides interfaces to hardware, as noted above. POSAs understood that it was customary to implement components with these properties as extensions of the kernel. Bhattacharjee, ¶225; Callender, 1:40-43 (noting that it was typical that “the [OS] kernel acts as a gatekeeper to computer resources”); EX1001, 6:57-59 (“It is common in most systems to include kernel modules which provide extensions to the existing operating system kernel.”); EX1018, 456, 696-697; EX1019, 1:16-34; EX1027, 8:64-9:5; EX1041, 8:56-64.

Thus, Callender’s *[OS] system kernel comprises a kernel module, i.e., the Hcaport*. Bhattacharjee, ¶226.

b. [5B]

The '058 patent does not define “*interface*,” but describes an embodiment where a “kernel module” acts as a “device interface” enabling “data exchange between a user mode CSE and a device driver in kernel mode.” EX1001, 9:60-10:20; Bhattacharjee, ¶227. The patent also describes an exemplary “kernel module” serving as an “interface” to a device driver by notifying a “service in the context of an application” of an “interrupt” from the device. EX1001, 8:46-52, Fig. 5; Bhattacharjee, ¶227. These examples are consistent with how POSAs understood “interface” in the context of applications and device drivers. EX1035, 279 (“interface” is “[s]oftware that enables a program to work with...another program...or with the computer’s hardware”); Bhattacharjee, ¶227.

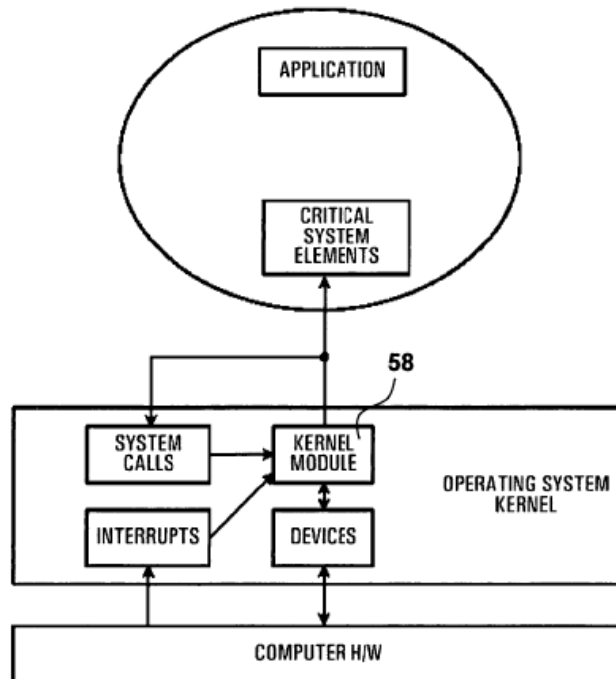


FIG. 5

The sending and receiving functions in Callender’s “user mode library” are *SLCSEs*. *Supra* §V.B.1.d.i ([1C.1]). POSAs understood that these functions access the services in the “miniport” for queue creation, interrupt setup, and event notification. *Supra* §V.B.4.b ([4B]). Callender states that the Hcaport “provides an abstracted general HCA device *interface*.” Callender, 5:43-44, Fig. 2A (annotated below); Bhattacharjee, ¶228.

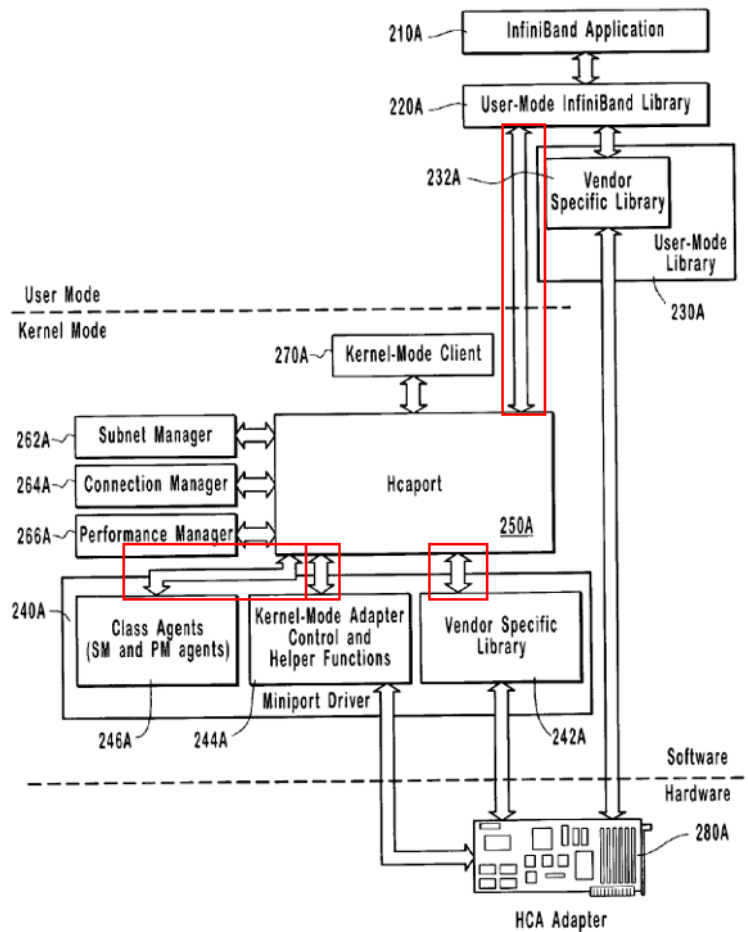


Fig. 2A

For multiple reasons, POSAs understood, or would at least have found obvious, that the user-mode library accesses the Miniport through the Hcport.

Bhattacharjee, ¶229. As explained *supra* §V.B.4.b ([4B]), the user-mode library has access to components (like the Hcport) that run in kernel mode. Moreover, Callender’s figures show arrows going from the “User-mode InfiniBand Library” to the Hcport and then from the Hcport to the miniport (which indicates to POSAs that user-mode processes can access the miniport through the Hcport) and a bidirectional arrow between the user-mode library and user-mode InfiniBand library (which indicates to POSAs that the user-mode library can access user-mode InfiniBand library services as well as vice-versa). Callender, Figs. 2A, 2C; Bhattacharjee, ¶229.

In such an implementation, the Hcport is *adapted to serve as an interface between* the sending or receiving function and a *device driver, i.e.,* the miniport (*see supra* §V.B.4.b.i ([4B])). The sending/receiving functions are *SLCSEs* running *in the context of an application*. *Supra* §V.B.1.f.i ([1E.1]); Bhattacharjee, ¶230.

Thus, Callender discloses or renders obvious that the Hcport, which is a *kernel module (supra* §V.B.5.a), is *adapted to serve as an interface between an SLCSE in the context of an application program and a device driver* (the miniport). Bhattacharjee, ¶231.

## 6. Claim 6

POSAs understood that the InfiniBand HCA adapter (*supra* §V.A) receives data in “packet” form. EX1013, [0007]; EX1042, 1:44-49. POSAs also understood that receiving data in InfiniBand involves the HCA adapter generating an “interrupt” that results in an “*event*” that signals to an application that incoming packets have been processed. EX1013, [0007]-[0010]; Bhattacharjee, ¶232.

POSAs understood that packet arrival is an *asynchronous event*, *i.e.*, one that may occur at any time. Bhattacharjee, ¶233; EX1043, 1:12-17; EX1035, 38. POSAs also understood that this event *requires information to be passed to* the user-mode receive function, which is *an SLCSE that runs in the context of an application program (supra* §V.B.1.f.i ([1E.1])), *from outside the application*, for two reasons: (1) the event requires the SLCSE to be notified of the packet’s arrival by the miniport, which is *outside* the application; and (2) in response to the event, the SLCSE gets incoming data from the HCA, which is also outside the application. EX1013, [0009]-[0010]; Callender, Fig. 2A; Bhattacharjee, ¶234.

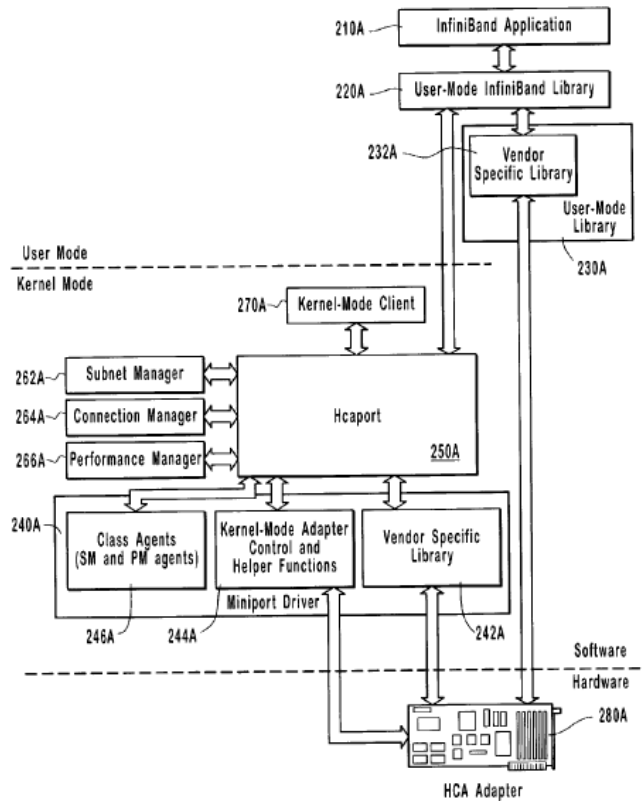


Fig. 2A

While Callender’s miniport provides “interrupt” and “event request” “management” (*supra* §V.B.4.b.ii ([4B])), Callender’s Hcaport, *i.e.*, the *kernel module*, is the interface between the user-mode receive function and the miniport (*supra* §V.B.5.b ([5B])); thus, POSAs understood that the Hcaport *is adapted to provide a notification* of the above-discussed *event* to the receive function by passing the notification from the miniport. Bhattacharjee, ¶235.

## 7. Claim 7

“[N]otifying the SLCSE” includes notifying about claim 6’s *event*, which is caused by an interrupt. *Supra* §V.B.6 (claim 6); EX1013, [0007]-[0010];

Bhattacharjee, ¶236. Callender’s miniport, *i.e.*, the *kernel module* (*supra* §V.B.5.a ([5A])), performs “helper” functions including “management” of “interrupts.” Callender, 6:11-20. POSAs understood that it was customary to manage interrupts using an interrupt “*handler*.” EX1013, [0003]; Bhattacharjee, ¶237. POSAs would also have found it obvious to implement the interrupt handler to notify the user-mode receive function, *i.e.* the *SLCSE*, of the *event* that an incoming packet has been processed via an *upcall mechanism*, which is “[a] means by which a service in kernel mode executes a function in a user mode application context” (EX1001, 6:60-61), because using an upcall to inform user-mode processes about interrupts was a known technique within a POSA’s skill that would have beneficially signaled the user-mode receive function to read incoming data. Bhattacharjee, ¶¶238-239; EX1044, [0042]; EX10145, 26:36-53; EX1046, 14:55-59.

## 8. Claim 8

Callender’s kernel-mode processes have access to “all available memory,” including user-mode memory. Callender, 1:38-41; Bhattacharjee, ¶240. Thus, POSAs understood that *in operation*, the *up call mechanism* discussed *supra* §V.B.7 (claim 7) that is used by the interrupt handler to invoke the user-mode receive function, which is an *SLCSE*, can *execute instructions from* that *SLCSE*, which is *resident in user mode space*, while remaining in *kernel mode*. POSAs

would have been motivated to implement the interrupt handler this way to beneficially avoid the “impact [on] performance” of switching to user mode, and would reasonably have expected success because Callender’s kernel-mode processes can access all memory. Callender, 1:53-54; Bhattacharjee, ¶241.

### 9. Claim 9

“A function overlay occurs when the implementation of a function that would normally be called, is replaced such that an extension or replacement function is called instead.” EX1001, 8:62-64; Bhattacharjee, ¶242. POSAs understood that Callender’s user-mode sending and receiving functions are “replace[ments]” for “function[s]” in the OS “that would normally be called” (EX1001, 8:62-64) because the sending and receiving operations these functions perform were normally provided by the OS. *Supra* §V.B.1.c.ii ([1B.2]); Bhattacharjee, ¶243.

Thus, in Callender, *a function overlay is used to provide one of the plurality of software applications access to operating system services* because the sending and receiving services normally supplied by the OS to applications are now supplied by user-mode functions. *Supra* §V.B.1.d.i ([1C.1]); Bhattacharjee, ¶244.

## 10. Claim 10

During prosecution, the applicants asserted that then-pending claim 11, which became claim 10, “distinctly claims that user mode CSEs are part of an application” rather than “independent applications.” EX1002, 247.

Callender’s *SLCSEs* form a part of applications by virtue of applications linking to the DLL containing the *SLCSEs*, and are therefore not “independent applications.” *Supra* §V.B.1.e.iii ([1D.3]); EX1002, 247; Bhattacharjee, ¶¶245-246. Furthermore, because Callender’s *SLCSEs* are part of a dynamic *link* library, they are *linked to particular software applications of the plurality of software applications*, namely the applications using the DLL, and each application has *a link that provides unique access to a unique instance of a CSE* for the reasons discussed *supra* §V.B.1.f.ii ([1E.2]). Bhattacharjee, ¶246.

POSAs would have found it obvious to implement Callender to link the DLL to applications *as the applications are loaded*, because this was a standard way of linking shared libraries (as the ’058 patent itself admits). Bhattacharjee, ¶247; EX1001, 9:18-22 (CSEs “are contained in a shared library. *As such* they are *linked* to a software application *as the application is loaded*.”); EX1097, 18:31-46 (“As is well known, [a] DLL...is dynamically linked” to applications “*when they are loaded*”); EX1047, 42:66-43:10; Bhattacharjee, ¶247.

## 11. Claim 11

Callender's *SLCSEs* (user-mode sending and receiving functions, *supra* §V.B.1.d.i ([1C.1])), both *utilize kernel services supplied by the [OS] kernel* for the purposes recited in claim 11, as explained below. Bhattacharjee, ¶248.

### a. Device access

Callender's miniport provides "access to [the] HCA adapter," *i.e.*, *device access*. Callender, 4:57-61; Bhattacharjee, ¶249. Furthermore, *SLCSEs utilize* the "queues" the miniport creates for "IO," *i.e.* input/output, *i.e.*, *access*, to the HCA adapter. Callender, 6:56-64; *supra* §V.B.4.b.i ([4B]); Bhattacharjee, ¶250. The miniport also performs security functions necessary to enable the *SLCSEs* to access the adapter. Callender, 4:41-5:5; Bhattacharjee, ¶251. Callender's "miniport" is part of the OS kernel *supra* §V.B.4.b.i ([4B]); thus, *the SLCSEs utilize kernel services supplied by the operating system kernel, i.e.* the miniport's services, *for device access*. Bhattacharjee, ¶252.

### b. Interrupt Delivery

Callender's user-mode receiving function is notified by a miniport interrupt handler when the HCA adapter has processed an incoming packet. *Supra* §V.B.7 (claim 7). Additionally, the InfiniBand sending operation involves generating an interrupt to provide notification that the HCA adapter has completed a transmission. EX1013, [0009]-[0010]; Bhattacharjee, ¶253. Thus, the receiving

and sending functions *utilize kernel services* as claimed for *interrupt delivery*.

Bhattacharjee, ¶253.

### c. Virtual memory mapping

POSAs understood that “*virtual-to-physical*” memory “address *mapping*” is a standard *kernel service supplied by the operating system kernel*. Bhattacharjee, ¶254; EX1017, 1:48-60. Additionally, POSAs understood that applications linking to the “user mode library”—a DLL (Callender, 5:13-15)—would have the library in their “virtual address space.” EX1017, 3:52-4:32, Fig. 5; Bhattacharjee, ¶254. Thus, POSAs would have found it obvious to implement Callender’s user-mode library’s sending and receiving functions to use the kernel’s services *for virtual memory mapping* to achieve known benefits of virtual memory, *e.g.*, “isolating processes from each other,” and would reasonably have expected success because such use of the kernel was well-known. EX1017, 1:50-56; Bhattacharjee, ¶255.

## 12. Claim 12

Callender’s *SLCSEs* send/receive data via the HCA adapter, which uses the “InfiniBand” networking protocol (*supra* §V.A); therefore, the *SLCSEs include services related to at least networking protocol processes*. Bhattacharjee, ¶256.

If claim 12 also requires the *SLCSEs* to include services related to *management of files*, this would have been obvious. The ’058 patent’s specification never mentions “management,” but mentions “access[ing] files that

reside in different locations” as an example of “File System services” that are CSEs. EX1001, 6:10-17. It was well-known to use InfiniBand to connect to “network attached storage (NAS)” providing “file access”; thus, in an obvious use of Callender’s SLCSEs to access files on NAS, the SLCSEs include services *related to the management of files*. EX1064, [0002]-[0003]; Bhattacharjee, ¶257.

### **13. Claim 13**

POSAs understood that a known alternative to using interrupts to signal completion of an InfiniBand operation (*see supra* §V.B.4.b.ii ([4B])) was to poll to see if the operation is complete, and understood that using interrupts vs. polling involved a tradeoff between reducing latency (the overall time between the start and end of an operation) via polling or reducing processor resource usage via interrupts. Bhattacharjee, ¶258; EX1048, 2; EX1049, 281-282. Thus, POSAs would have been motivated to *modify* the user-mode sending and receiving functions, *i.e.* the SLCSEs, to use polling *for a particular one of the software applications* that requires minimizing latency, and would reasonably have expected success because polling was a known technique. EX1003, ¶259.

### **14. Claim 14**

The interrupt-based and polling-based versions of Callender’s SLCSEs are used by different applications (*supra* §V.B.13 (claim 13)); these different SLCSE versions are “respective versions” that different applications are “provided with,”

*i.e., SLCSEs that are application specific.* EX1001, 3:57-60 (“software **applications are provided with respective versions** of [CSEs].... the system elements which are ***application specific*** reside in user mode[.]”); Bhattacharjee, ¶260. These SLCSEs are in the user-mode library, *i.e., reside in user mode.* *Supra* §V.B.1.d.i ([1C.1]); Bhattacharjee, ¶260.

Callender’s OS *critical system elements* (OSCSEs), are in, and therefore *reside in, the operating system kernel.* *Supra* §V.B.1.c.ii ([1B.2]); Bhattacharjee, ¶261. The ’058 patent defines a “compute ***platform***” as [t]he combination of computer hardware and a single instance of an operating system.” EX1001, 6:29-30. Callender’s OSCSEs are hardware-specific because they use a “vendor specific library” for the HCA adapter, which is “hardware,” and they are OS instance-specific because they are part of the operating system kernel; thus, the OSCSEs are *platform specific.* Callender, 4:51-5:28; *supra* §V.B.1.c.ii ([1B.2]); Bhattacharjee, ¶261.

## 15. Claim 15

Callender’s Hcaport is a *kernel module*, and Callender’s “***kernel mode*** miniport ***driver***” (Callender, 4:56, Fig. 2A)—a *device driver* (*Supra* §§V.B.5.a-V.B.5.b (claim 5))—is *in kernel mode.* Bhattacharjee, ¶262. Callender’s *SLCSEs* are in the “***user mode*** library” and thus *in user mode.* *Supra* §V.B.1.d.i ([1C.1]); Callender, 4:51-5:18; Bhattacharjee, ¶262.

The '058 patent says that “[a]s a **device interface**, the kernel module enables *data exchange* between a user mode CSE and a device driver in kernel mode” using “virtual memory such that data is transferred in both directions without a copy.” EX1001, 10:60-66. Immediately following this statement, the patent states that “[s]ervices exported **for device interface** typically include” a list of possibilities including “[i]nitialization,” “[e]stablish[ing] a channel between a CSE in user mode and a specific device,” and “[i]nform[ing] the interrupt service that this CSE requires notification.” EX1001, 9:66-10:20. Callender’s Hcport serves as an interface between the miniport and the SLCSEs for purposes including (1) SLCSEs sending queue-creation requests to the miniport; and (2) the miniport sending event notifications to the SLCES. *Supra* §§V.B.5.a-V.B.5.b (claim 5); Bhattacharjee, ¶263. Therefore, the Hcport is *adapted to enable data exchange between the SLCSEs in user mode and a device driver in kernel mode*, and *data is transferred both* from the SLCSE to the driver and vice-versa as claim 15 recites. Bhattacharjee, ¶263.

One well-known way to exchange data between two entities was via data structures in memory, *e.g.*, queues, accessible to both entities. Bhattacharjee, ¶264; EX1019, 2:22-39 (describing well-known “virtual interface” standard using “queues” “mapped directly to user address space” for data exchange); Bhattacharjee, ¶264. POSAs would have found it obvious to *use mapping of*

*virtual memory* to implement these data structures to achieve the well-known benefits of virtual memory, and would reasonably have expected success because virtual memory mapping was well-known. EX1017, 1:48-60; Bhattacharjee, ¶264.

#### **16. Claim 16**

Callender's SLCSEs *form a part of* software applications *by being* in a DLL *linked* to the applications. *Supra* §V.B.1.f.i ([1E.1]); Bhattacharjee, ¶265.

#### **17. Claim 17**

Callender says “access to [the] HCA adapter...through [the] user mode library...is practical” because in InfiniBand, “[i]nitiating and terminating access are controlled through kernel mode,” but “[a]fter initiation...extremely fast user mode access” is enabled. Callender, 4:57-67. Callender also says the miniport “is responsible for creating and destroying communication queues,” whereas the user mode library “permits user mode data transfer to the communication queues at [the] HCA adapter.” Callender, 4:57-5:5. Thus, POSAs understood that after the miniport creates queues and initiates access, the user mode sending/receiving functions do not need the kernel for anything else except interrupts and any virtual memory access that may occur. Bhattacharjee, ¶266; *see also* EX1013, [0009]-[0010] (describing well-known InfiniBand sending/receiving methods, not utilizing kernel services).

Thus, other than *utilizing kernel services supplied by the operating system kernel for device access, interrupt delivery, and virtual memory mapping*, which the user-mode sending and receiving functions, *i.e. the SLCSEs*, do (*supra* §V.B.11 (claim 11)), *the SLCSEs otherwise execute without interaction from the operating system kernel*. Bhattacharjee, ¶267.

### 18. Claim 18

POSAs understood that Callender discloses implementing kernel-mode sending and receiving operations and their corresponding user-mode operations with different code. Bhattacharjee, ¶268. Callender teaches defining a common “interface” for user-mode and kernel mode operations. *E.g.*, Callender, 7:5-20, Fig. 3; claims 1, 9, 17; Bhattacharjee, ¶268. While the same code **can** be used for the underlying implementations of operations (4:30-32), Callender does not **require** this; rather, Callender says “user mode implementations of a given operation **tend to differ** from kernel mode implementations” and that as an alternative, “same source code **may** be written.” Callender, 4:17-32; Bhattacharjee, ¶268. While Callender acknowledges the benefits of using the same code (4:33-40), Callender does not teach away from different code. Bhattacharjee, ¶268. Furthermore, POSAs understood that using different code provided benefits such as allowing the user-mode library to be updated periodically without requiring

modifications to the OS kernel, thus avoiding the risk of damaging the kernel.

Bhattacharjee, ¶268.

Thus, in one obvious implementation of Callender, the user-mode functions for the sending and receiving operations, *i.e.* *SLCSEs*, do not use the same code as, and therefore *are not copies of*, the kernel-mode implementations of those operations, *i.e.* *OSCSEs*. Bhattacharjee, ¶269.

## **VI. DISCRETIONARY DENIAL IS UNWARRANTED**

### **A. §314(a)**

#### **1. *Fintiv***

“[T]he PTAB will not deny institution based on *Fintiv* if,” as here, “there is compelling evidence of unpatentability.” Director’s Interim Procedure for Discretionary Denials, 5 (June 21, 2022) (“Interim Procedure”) (discussing *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (Mar. 20, 2020) (precedential)). That should conclude the *Fintiv* analysis.

If considered, the *Fintiv* factors weigh against denial.

#### **a. Stay Potential**

This factor is currently neutral or favors institution. *Sand Revolution II, LLC, v. Cont’l Intermodal Group–Trucking*, IPR2019-01393, Paper 24, 7 (June 16, 2020) (informative). The district-court proceeding is currently temporarily stayed pending transfer. EX1089, 4-5.

**b. Trial Timing**

Petitioner’s motion to transfer the district-court proceeding to the Northern District of California was recently granted. EX1089. That district’s median time-to-trial is 47.9 months (EX1051 at 66), and no post-transfer trial schedule has yet been set. Factor 2 therefore weighs against denial. *BMW of North America, LLC v. Michigan Motor Techs., LLC*, IPR2023-01224, Paper 15, 11 (Feb. 15, 2024).

**c. Litigation Investment**

There has been little investment in invalidity-related issues in the district court litigation. A *Markman* hearing was canceled before the transfer decision EX1090, and “a significant portion of work remains to be done” including “fact and expert discovery and dispositive motions.” *Protect Animals With Satellites LLC v. OnPoint Sys., LLC*, IPR2021-01483, Paper 11, 14-15 (Mar. 4, 2022).

**d. Issue Overlap**

This Petition challenges 18 claims, whereas only seven are asserted in district court. *See* EX1020. This weighs against denial. *Markforged Inc. v. Continuous Composites*, IPR2022-00679, Paper 7, 32-33 (Oct. 25, 2022).

**e. Litigation Defendants**

This factor is at worst neutral. *Id.* at 33.

**f. Other Circumstances**

The Petition’s “strong showing on the merits” weighs against denial. Interim Procedure, 3-5.

## 2. Unified Reexamination

The '058 patent is the subject of *Ex Parte* Reexamination No. 90/019,676 filed by Unified Patents, LLC (“Unified”) (the “Unified Reexam”). EX1060. Multiple Board panels have held that the discretionary-denial factors applicable when there has been a previous **IPR petition** on the same patent do not apply where, as here, the previous challenge was an **ex-parte reexam** by another party. *See, e.g., Tesla, Inc. v. Graphite Charging Company LLC*, IPR2024-00388, Paper 15, 6-8 (Aug. 27, 2024) (rejecting patent owner’s argument that petition should be discretionarily denied under §314(a) (or under §315(d)) because of prior reexam, collecting cases). Moreover, the Unified Reexam challenges only claims 1 and 12, while this Petition challenges all claims. EX1060, 2. Discretionary denial based on the Unified Reexam is therefore unwarranted.

## 3. Amazon IPR

The day before this Petition’s filing, Amazon.com, Inc. (“Amazon”) filed IPR2025-00561 challenging the '058 patent over different prior art than this Petition. Google and Amazon have different allegedly infringing products, had no involvement in preparing each other’s IPRs, and thus have no “significant relationship.” *Ford Motor Co. v. Neo Wireless LLC*, IPR2023-00763, Paper 28, 7, 10 (Mar. 22, 2024) (director review); *Facebook, Inc. v. Express Mobile Inc.*, IPR2021-01457, Paper 10, 11-12 (Mar. 18, 2022) (no “significant relationship”

where litigation defendants in “joint defense relationship” did not coordinate on IPRs); *Videndum Production Sols., Inc. v. Rotolight Ltd.*, IPR2023-01218, Paper 12, 7 (Apr. 19, 2024) (director review) (discretionary denial “not justified” where petitioners “do not have a significant relationship”). Moreover, by filing at essentially the same time as Amazon, Google obtains no road-mapping benefit from Amazon’s IPR, and is only exercising its own right to challenge the patent Google is accused of infringing. *Volkswagen Grp. of America, Inc. v. Carucel Investments, L.P.*, IPR2019-01105, Paper 8, 16-17 (noting also that “[t]he Trial Practice Guide[’s]” provisions on parallel petitions “*by a petitioner*” do not apply to petitions “filed by different petitioners”) (emphasis original). Discretionary denial based on Amazon’s IPR is therefore unwarranted.

**B. §325(d)**

Callender and the background references cited herein were not before the Office during prosecution, nor are they at issue in the Unified Reexam, so denial under §325(d) is unwarranted.

**VII. CONCLUSION**

The Board should institute review and cancel claims 1-18.

Dated: January 31, 2025

Respectfully submitted,  
*Google LLC*

/Elisabeth Hunt/

Elisabeth H. Hunt, Reg. No. 67,336  
WOLF, GREENFIELD & SACKS, P.C.

**CERTIFICATE OF SERVICE UNDER 37 C.F.R. § 42.6 (E)(4)**

I certify that on January 31, 2025, I will cause a copy of the foregoing document, including any exhibits or appendices filed therewith, to be served via USPS Priority Mail Express at the following correspondence address of record for the patent:

Allen, Dyer, Doppelt + Gilchrist, PA  
1135 East State Road 434  
Suite 3001  
Winter Springs, FL 32708

Date: January 31, 2025

/Dara Del Rosario/  
Dara Del Rosario  
Paralegal  
WOLF, GREENFIELD & SACKS, P.C.

## **CERTIFICATE OF WORD COUNT**

Pursuant to 37 C.F.R. § 42.24, the undersigned certifies that the foregoing Petition for *Inter Partes* Review contains 13,994 words excluding a table of contents, a table of authorities, Mandatory Notices under § 42.8, a certificate of service or word count, or appendix of exhibits or claim listing. Petitioner has relied on the word count feature of the word processing system used to create this paper in making this certification.

Date: January 31, 2025

/Dara Del Rosario/  
Dara Del Rosario  
Paralegal  
WOLF, GREENFIELD & SACKS, P.C.

## VIII. CLAIM LISTING

The following claim listing assigns element labels (e.g., [1A1], [1A2], etc.) to certain claims for clarity.

<b>Claim 1</b>
[1PRE] A computing system for executing a plurality of software applications comprising:
[1A] a) a processor;
[1B.1] b) an operating system having an operating system kernel having...
[1B.2] OS critical system elements (OSCSEs)...
[1B.3] for running in kernel mode using said processor; and,
[1C.1] c) a shared library having shared library critical system elements (SLCSEs) stored therein...
[1C.2] for use by the plurality of software applications in user mode and
[1D.1] i) wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and...
[1D.2] are accessible to some of the plurality of software applications and...
[1D.3] when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications,
[1E.1] ii) wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications and...
[1E.2] where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function, and
[1F] iii) wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously.
<b>Claim 2</b>
A computing system as defined in claim 1, wherein in operation, multiple instances of an SLCSE stored in the shared library run simultaneously within the operating system.

**Claim 3**

A computing system according to claim 1 wherein OSCSEs corresponding to and capable of performing the same function as SLCSEs remain in the operating system kernel.

**Claim 4**

[4A] A computing system according to claim 1 wherein the one or more SLCSEs provided to one of the plurality of software applications having exclusive use thereof,

[4B] use system calls to access services in the operating system kernel.

**Claim 5**

[5A] A computing system according to claim 1 wherein the operating system kernel comprises a kernel module

[5B] adapted to serve as an interface between an SLCSE in the context of an application program and a device driver.

**Claim 6**

A computing system according to claim 5 wherein the kernel module is adapted to provide a notification of an event to an SLCSE running in the context of an application program, wherein the event is an asynchronous event and requires information to be passed to the SLCSE from outside the application.

**Claim 7**

A computing system according to claim 6 wherein a handler is provided for notifying the SLCSE in the context of one of the plurality of software applications through the use of an up call mechanism.

**Claim 8**

A computing system according to claim 7 wherein the up call mechanism in operation, executes instructions from an SLCSE resident in user mode space, in kernel mode.

**Claim 9**

A computing system according to claim 2, wherein a function overlay is used to provide one of the plurality of software applications access to operating system services.

**Claim 10**

A computing system according to claim 2 wherein SLCSEs stored in the shared library are linked to particular software applications of the plurality of software applications as the particular software applications are loaded such that the particular software applications have a link that provides unique access to a unique instance of a CSE.

**Claim 11**

A computing system according to claim 2 wherein the SLCSEs utilize kernel services supplied by the operating system kernel for device access, interrupt delivery, and virtual memory mapping.

**Claim 12**

A computing system according to claim 1, wherein SLCSEs include services related to at least one of, network protocol processes, and the management of files.

**Claim 13**

A computing system according to claim 10 wherein some SLCSEs are modified for a particular one of the plurality of software applications.

**Claim 14**

A computing system according to claim 13 wherein the SLCSEs that are application specific, reside in user mode, while critical system elements, which are platform specific, reside in the operating system kernel.

**Claim 15**

[15A] A computing system according to claim 5 wherein the kernel module is adapted to enable data exchange between the SLCSEs in user mode and a device driver in kernel mode,

[15B] and wherein the data exchange uses mapping of virtual memory such that data is transferred both from the SLCSEs in user mode to the device driver in kernel mode and from the device driver in kernel mode to the SLCSEs in user mode.

**Claim 16**

A computing system according to claim 1 wherein SLCSEs form a part of at least some of the plurality of software applications, by being linked thereto.

**Claim 17**

A computing system according to claim 2 wherein the SLCSEs utilize kernel services supplied by the operating system kernel for device access, interrupt delivery, and virtual memory mapping and otherwise execute without interaction from the operating system kernel.

**Claim 18**

A computer system as defined in claim 2 wherein SLCSEs are not copies of OSCSEs.