

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AMERICAN AIRLINES, INC.,

and

SOUTHWEST AIRLINES, CO.,

Petitioners,

v.

THE REGENTS OF THE UNIVERSITY OF CALIFORNIA
(INTELLECTUAL VENTURES I LLC),

Patent Owner

Case No. IPR2025-01511
U.S. Patent No. 7,712,080

PATENT OWNER'S PRELIMINARY RESPONSE

TABLE OF CONTENTS

PATENT OWNER’S EXHIBIT LIST	iv
I. INTRODUCTION	1
II. THE ‘080 PATENT	2
A. Summary.....	2
B. Prosecution History	5
1. Fukuda ‘99 (EX2006).....	6
2. Fukuda ‘01 (EX1009).....	9
III. CLAIM CONSTRUCTION.....	11
A. “wherein the at least one intermediate condition comprising one intermediate result” (Claims 1 and 9)	12
B. “wherein the distributed parallel computing program is configured to operate across a distributed share[d] memory system” (Claims 17 and 18).....	13
C. Recent Director decisions confirm that Petitioners’ failure to explain inconsistent claim scope positions is fatal	14
IV. REFERENCES RELIED ON IN THE PETITION	16
A. Declaration of Michael Ian Shamos (EX1003)	16
B. Fukuda ‘98 (EX1008).....	18
V. PETITIONERS HAVE NOT MET THEIR BURDEN TO ESTABLISH ANTICIPATION UNDER 35 U.S.C §102.....	20
A. The Petition Fails To Explain How Fukuda ‘98 Discloses Establishing At Least Distributed Shared Variable That is A Single Variable That Includes Several Variables	22
B. The Mappings In The Petition Impermissibly Pick And Choose Various Disclosures From Fukuda ‘98 And Fail To Demonstrate That The Reference Discloses All Of The Elements Of Claim 1 As Arranged In The Claim	24

C.	The Petition Fails To Present A Coherent Or Consistent Mapping Of The At Least One Intermediate Condition Recited Throughout Claim 1	27
D.	The Petition Fails To Explain How Fukuda ‘98 Performs The Parallel Processing And/Or Operations Recited In Claim 1	30
E.	The Petition’s Erroneous Allegation That Fukuda ‘98 Is Not Cumulative Of The Art In Front Of The Examiner Is Predicated On A False Factual Assertion	33
F.	The Petition Fails To Present A Reasonable Likelihood Of Success With Respect To Any Of The Challenged Claims	37
VI.	CONCLUSION	38

PATENT OWNER'S EXHIBIT LIST

EX2001	<i>American Airlines, Inc. and v. Intellectual Ventures II LLC</i> , IPR2025-00785 and IPR2025-00786, Paper 11 (PTAB August 29, 2025) (declining to exercise discretion to deny IPRs).
EX2002	<i>American Airlines, Inc. and v. Intellectual Ventures II LLC</i> , IPR2025-00782 and IPR2025-00931, Paper 12 (PTAB September 19, 2025) (declining to exercise discretion to deny IPRs).
EX2003	<i>American Airlines, Inc. and v. Intellectual Ventures II LLC</i> , IPR2025-00987 and IPR2025-01055, Paper 9 (declining to exercise discretion to deny IPRs).
EX2004	Defendant Southwest Airline Co.'s Preliminary Invalidity Contentions
EX2005	IAM 2014 - What's Inside IVs Patent Portfolio
EX2006	Munehiro Fukuda, Lubomir F. Bic, Michael B. Dillencourt, and Jason M. Cahill, "Messages versus Messengers in Distributed Programming," <i>Journal of Parallel and Distributed Computing</i> , vol. 57, Issue 2, May 1999, pp. 1-8 (Fukuda '99)

I. INTRODUCTION

The Petition challenges claims 1-18 (the Challenged Claims) of U.S. Patent No. 7,712,080 (the ‘080 Patent) on a single statutory ground: that all of the Challenged Claims are anticipated by Fukuda ‘98. Institution should be denied for at least the following reasons.

First, the Petition is predicated on inconsistent claim construction positions. For example, in the district court, Petitioner Southwest contends that the limitation that recites “wherein the at least one intermediate condition comprising one intermediate result” in independent claims 1 and 9 is indefinite, while in this IPR, Petitioners purport to apply the *Phillips* standard and interpret the same language as having a workable meaning. The Petition never acknowledges these inconsistent claim construction positions, nor explains why inconsistent positions are warranted in this case.

Second, the Petition attempts to fill gaps in the prior art with expert testimony that lacks grounding in the evidence. The expert declaration is not prior art and cannot substitute for a missing disclosure in the asserted reference. That is especially the case here, where the aspects of expert declaration relied on to bridge gaps in the prior art largely restate word for word the Petition’s conclusory assertions without providing additional substantive explanation.

Third, the purported mappings of the asserted reference to the Challenged Claims are seriously deficient. In some cases, the Petition either completely fails to address the claim language or fails to explain how the asserted reference meets the claim language. In other cases, the Petition relies on disparate aspects of the asserted reference for limitations that are connected by the claim language and, when doing so, fails to explain how the asserted reference discloses the claimed elements arranged as in the claim, as required to establish anticipation. In still other cases, the mappings in the Petition are inconsistent in that when the limitation first appears in the claim, the Petition maps one aspect of the asserted reference to the claim language, but when the same limitation is referenced later in the claim the Petition maps a different aspect of the asserted reference to the limitation.

For at least these reasons, and as explained more fully below, the Petition fails to demonstrate a reasonable likelihood of success on the merits.

II. THE ‘080 PATENT

A. Summary

The ‘080 Patent is directed to systems and methods for developing parallel distributed programs, including programs configured to operate across multiple processors or nodes and multiple memories. EX1001, 2:7-18. The ‘080 Patent identifies that conventional approaches such as message passing often require programmers to restructure code and data structures to handle inter-processor

communication, which can be complicated, tedious, and error-prone, and may cause the resulting program to lose algorithmic integrity and data structure integrity. EX1001, 1:30-52.

The '080 Patent describes an alternative programming model, referred to as “navigational programming,” that uses self-migrating threads or tasks capable of moving from one processor to another. EX1001, 3:12-26. The patent explains that such self-migrating threads can be programmed explicitly, including using a hop() command, and can migrate to another processor when needed to access memory belonging to that other processor, then resume computation. EX1001, 3:20-32.

The '080 Patent describes a foundational concept referred to as “distributed sequential computing” (“DSC”), which is computing using a single locus of computation over distributed data, where a self-migrating thread may perform DSC by migrating between processors to access distributed portions of the DSV. EX1001, 3:50-58.

The specification also emphasizes the “pivot-computes” principle, which it defines as the principle under which computation takes place on the node that owns the large-sized data, rather than moving the large portion of data to a different node for computation. EX1001, 4:1-13. In this way, navigational programming and DSC are presented as preserving original code structure and data structure while allowing the locus of computation to move to the data, reducing the need for the low-level

message-handling constructs that “pollute” source code in distributed memory programming. EX1001, 3:59-66 and 6:1-15.

The ‘080 Patent further explains that distributed parallel computing (“DPC”) can be achieved by computing using multiple concurrent DSC programs, where the “intersection” between composing DSC programs may be minimal and may consist only of local synchronizations, thereby minimizing code pollution among the composing DSC programs. EX1001, 5:55-66. The patent discloses a distributed environment where a computing thread injects, or spawns, a separate DSC program to perform such that the two DSC threads of the DPC program run concurrently and operations are pipelined. EX1001, 6:25-64.

Consistent with this disclosure, independent claim 1 recites establishing at least one distributed shared variable, developing at least one DSC program to access the distributed shared variable, and transforming the DSC program into at least one DPC program by spawning at least one child DSC program when at least one intermediate condition occurs within the DSC program, where the DPC program concurrently uses the DSC program and the spawned child DSC program to perform parallel processing and/or operations. EX1001, 11:1-25. Claim 1 further requires that the intermediate condition comprises one intermediate result that will be required by the spawned child DSC program to continue computation. EX1001, 11:22-28. Independent claim 9 recites a corresponding system architecture,

including the same intermediate condition and intermediate result requirements. EX1001, 12:11-27.

B. Prosecution History

The prosecution history for U.S. Appl. No. 10/850,842 reflects multiple Office Actions in which the Examiner repeatedly maintained rejections over prior art directed to mobile agents or threads and distributed programming. The first Office Action (dated May 21, 2007) rejected all pending claims, and included a rejection of claims 1–7, 9–17, and 20–23 as anticipated by Pan, a rejection of claim 8 as obvious over Pan in view of Fukuda ‘99, and a rejection of claims 18, 19, 24, and 25 as obvious over Pan in view of Pan2. EX1002, 161–162, 164 and 169–170. In a second non-final Office Action (dated October 18, 2007), pending claims 1–25 were rejected again. EX1002, 124.

In a final Office Action (dated July 22, 2008), the Examiner rejected the pending claims as obvious over Pan in view of Suzuki and further in view of Fukuda ‘99. EX1002, 97–100. After the Applicant filed a Request for Continued Examination, the Examiner issued a further Office Action (dated April 2, 2009) maintaining the rejection of claims 1–6, 9, 10, and 15–22 as obvious over Pan, Suzuki, and Fukuda ‘99. EX1002, 72–76. Prosecution continued thereafter and ultimately culminated in a Notice of Allowability allowing claims 1–6, 9, 10, 15–22, 26, and 27 (which were renumbered as 1–18). EX1002, 36.

1. Fukuda ‘99 (EX2006)

“Messages versus Messengers in Distributed Programming” (Fukuda ‘99) was cited by the Examiner as the basis for a 35 U.S.C. § 103 rejection in multiple Office Actions, including those dated 4/12/2009 (EX1002, 72–81), 7/22/2008 (EX1002, 97–106), 10/18/2007 (EX1002, 123–133), and 5/21/2007 (EX1002, 161–172).

Fukuda ‘99 lists Munehiro Fukuda, Lubomir F. Bic, Michael B. Dillencourt, and Jason M. Cahill as authors. EX2006, 188. The ‘080 Patent identifies inventors Lei Pan, Lubomir R. Bic, and Michael B. Dillencourt. EX1001, cover page. Thus, two of the authors of Fukuda ‘99 -- Lubomir Bic and Michael Dillencourt -- are also named inventors of the ‘080 Patent. EX2006, 188; EX1001, cover page.

Substantively, Fukuda ‘99 contrasts conventional message-passing distributed programming (send/receive primitives, RPC/rendezvous, passive data messages) with a “Messengers” approach based on autonomous “self-migrating” Messengers that have identity and behavior, actively navigate the computational network to perform tasks at visited nodes, and coordinate with other autonomous Messengers. EX2006, 188–189. Fukuda ‘99 explains that Messengers applications are written using navigational commands rather than send/receive primitives. The navigational style reduces the semantic gap between abstract algorithm descriptions

and their implementations and can simplify the construction, understanding, and maintenance of distributed programs. EX2006, 188.

Fukuda '99 describes a three-layer network model, including a physical network (LAN/WAN), a daemon network in which each daemon is a UNIX process running a MESSENGERS interpreter, and an application-specific logical network created on top of the daemon network. EX2006, 190–191. Fukuda '99 explains that, at system startup, an init logical node is created on each daemon node, and a Messenger may be injected into an init node and then create additional logical nodes and links on the current or other daemons. EX2006, 190–191. Fukuda '99 further describes Messenger programs (Messenger scripts) as written in a subset of C and compiled, and explains that each script is carried by the Messenger as it propagates through the network and is replicated when the Messenger needs to follow more than one logical link. EX2006, 190–191. Fukuda '99 emphasizes that the application consists of the logical network and the corresponding Messengers navigating this network, rather than user-supplied node programs typical of message passing. EX2006, 190–191.

Fukuda '99 also describes a scheduling policy for Messenger execution. It explains that a daemon will interrupt a Messenger only when the Messenger issues a navigational command or begins executing a native-mode function, and will not

interrupt a Messenger in the middle of a native-mode function or other computational command. EX2006, 191.

Fukuda '99 describes the variable model used by Messengers, distinguishing among messenger variables, node variables, and network variables. Messenger variables are described as private to, and carried by, each Messenger as it propagates through the logical network. Node variables are described as resident in nodes of the logical network and shared by all Messengers currently visiting the same logical node. Network variables are predefined at each logical node and provide access to local network information, including the current host address and the last traversed link. EX2006, 191–192. Fukuda '99 explains that this distinction provides software engineering and performance advantages, including avoiding the copying of data into and out of message buffers that occurs in message passing systems. EX2006, 191–192.

Finally, Fukuda '99 introduces the navigational statements central to Messengers-based programming, focusing on hop, create, and delete. EX2006, 192. Fukuda '99 explains that hop permits a Messenger to move around the logical network using a destination specification and that hop semantics replicate the Messenger to nodes matching the destination criteria, after which the Messenger executing the hop ceases to exist. EX2006, 192.

2. Fukuda '01 (EX1009)

As shown below, the Examiner expressly identified “MESSENGERS: Distributed Programming Using Mobile Agents” (Fukuda '01) as “considered pertinent to the applicant’s disclosure” in the May 21, 2007 Office Action. EX1002, 171.

Conclusion

12. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.
- Fukuda et al., (Messengers: Distributed Programming Using Mobile Agents, Transactions of the SDPS, December 2001, Vol.5, No.4, page 95-112) discloses Messengers system using mobile agents.
 - Stanislaw Chrobot, (Sharing Variables in Distributed Memory, Fundamenta Informaticae 44 (2000) 63-81, IOS Press, pages 63-81) discloses shared variables in distributed system.
 - Itzkovitz et al., (Thread migration and its applications in distributed shared memory, The journal of systems and software 42 (1998) 71-97) discloses thread migration.
 - Wicke et al., (Compiling for Fast State Capture of Mobile Agents, Parallel Computing: Fundamentals & Applications, Proceedings of the International Conference ParCo'99, 17-20 August 1999, pages 1-8) discloses compiler for compiling mobile agent.
 - Pan et al., (Distributed Sequential computing using mobile code: Moving Computation to Data, IEEE 2001, pages 77-84) discloses distributed sequential computing using mobile code for moving computation to data.

Fukuda '01 describes the MESSENGERS distributed programming system, in which “Messengers” operate as mobile agents that carry executable behavior as they move through the system. Fukuda '01 explains that Messengers navigate among nodes, perform tasks at the nodes they visit, and coordinate with other Messengers. EX1009, 96. Fukuda '01 presents a three-layer network model that includes a physical network (e.g., LAN/WAN), a daemon network comprising UNIX processes

running MESSENGERS daemons on physical nodes, and an application-defined logical network constructed on top of the daemon network. EX1009, 96–97. Fukuda ‘01 further explains that daemons receive arriving Messengers, schedule and supervise their execution, and forward them to subsequent destinations based on the navigational operations invoked by the Messengers. EX1009, 97.

Fukuda ‘01 further describes Messenger scripts as written in a subset of C and compiled into native code, and categorizes script statements into computational statements, navigational statements, and task-control or coordination statements. EX1009, 97–99. As examples of navigational statements, Fukuda ‘01 describes hop as selecting one or more destination nodes and then moving the Messenger, with separate copies propagated when multiple destinations are selected, and it also describes create and delete operations for creating and deleting logical nodes or links in connection with Messenger movement. EX1009, 98–99. Fukuda ‘01 also explains how Messengers maintain and share state, distinguishing messenger variables carried with each Messenger from node variables that are resident at a logical node and shared among Messengers currently visiting that node. EX1009, 98. Finally, Fukuda ‘01 describes task-control mechanisms including inject, which initiates execution of one or more instances of another Messenger with specified initial parameters and optional starting destinations, as well as signal/wait operations for event-based synchronization among Messengers. EX1009, 99.

The authorship of Fukuda '01 also overlaps with the inventorship of the '080 Patent. Fukuda '01 lists four authors: Munehiro Fukuda, Lubomir F. Bic, Michael B. Dillencourt, and Fehmina Merchant. EX1009, 95. The '080 Patent identifies three inventors: Lei Pan, Lubomir R. Bic, and Michael B. Dillencourt. EX1001, cover page. Thus, two authors of Fukuda '01 -- Lubomir Bic and Michael Dillencourt -- are also named inventors of the '080 Patent. EX1009, 95; EX1001, cover page.

III. CLAIM CONSTRUCTION

In this IPR, Petitioners state that they “apply the *Phillips* . . . claim construction standard,” and contend that the Petition “establishes that the challenged claims are obvious under any reasonable interpretation under the *Phillips* standard.” Petition, 12. The claim construction positions advanced by Petitioner Southwest in this forum are inconsistent with the claim construction positions Petitioner Southwest has advanced in the district court.

In the district court, Southwest has argued that key claim terms of the '080 Patent are “indefinite under 35 U.S.C. § 112(b)” because, according to Southwest, such terms are “open to multiple interpretations, making the claim scope uncertain.” EX2004, 302-303. Yet, in this IPR, Southwest abandons those indefiniteness positions in favor of purportedly workable interpretations intended to support invalidity arguments in this forum, without even acknowledging, let alone justifying, the inconsistency. While a petitioner is not strictly prohibited from advancing

inconsistent positions, when it does so it must explain why the inconsistent positions are warranted. *Cambridge Mobile Telematics, Inc. v. Sfara, Inc.*, IPR2024-00952, Paper 12 at 7-8 (PTAB Dec. 13, 2024) (informative) (requiring, “[a]t a minimum,” an explanation “why the inconsistent positions are warranted”). Petitioners provide no such explanation here.

As detailed below, Southwest’s district court contention that certain claim terms are indefinite because they are “open to multiple interpretations, making the claim scope uncertain,” cannot be reconciled with Petitioners’ positions in this IPR that the same terms can be construed and mapped to Fukuda ‘98 under *Phillips*. EX2004, 302-303; Petition, 12. A claim term cannot simultaneously be indefinite under § 112(b) due to multiple competing interpretations that render scope uncertain and also have a plain and ordinary meaning that Petitioners can confidently apply to build their anticipation theory, absent a reasoned explanation for why the positions differ. Petitioners’ failure to present a consistent, legally sound position on claim scope undermines the integrity of the proceeding and independently demonstrates that Petitioners have not met their burden on claim construction and related showings.

A. “wherein the at least one intermediate condition comprising one intermediate result” (Claims 1 and 9)

Southwest’s inconsistency is clear from its treatment of the phrase “wherein the at least one intermediate condition comprising one intermediate result” in

independent claims 1 and 9. In district court, Southwest contends this phrase is indefinite and asserts:

“What is meant by that at least one intermediate condition ‘comprises’ one intermediate result? Is the result the condition or vice versa? How can a condition comprise a result? Without a defined meaning, the phrase is open to multiple interpretations, making the claim scope uncertain.” EX2004, 296.

In this IPR, however, Petitioners take the opposite position. Petitioners purport to apply the *Phillips* standard and interpret the same language as having a workable meaning, stating that “the intermediate condition is the receipt of the intermediate result from part of the computation on the current node,” and then attempt to map that interpretation onto Fukuda ‘98. Petition, 35-38. The Petition never acknowledges these inconsistent claim construction positions, and the Petition never explains why the indefiniteness position advanced by Petitioner Southwest in the district court should be disregarded here. EX2004, 296; Petition, 35-38.

B. “wherein the distributed parallel computing program is configured to operate across a distributed share[d] memory system” (Claims 17 and 18)

Southwest’s inconsistent claim construction positions are also shown by its treatment of the phrase “wherein the distributed parallel computing program is configured to operate across a distributed share[d] memory system” in claims 17 and 18. In district court, Southwest alleges this limitation is indefinite and asserts:

“What is meant by that the distributed parallel computing program operates across a shared memory system? The specification

only makes passing mention to ‘shared memory systems’ as existing in prior systems, such as clusters and networks of workstations. However, it is unclear how the distributed parallel computing program that is the subject of the ‘080 Patent claims [is] configured to operate across a shared memory system. Without a defined meaning, the phrase is open to multiple interpretations, making the claim scope uncertain.” EX2004, 296-297.

In this IPR, by contrast, Southwest again abandons that position. Petitioners do not contend, in this forum, that the term is indefinite or “open to multiple interpretations.” Instead, Petitioners proceed as though the phrase has a sufficiently definite meaning under *Phillips* and purport to map claims 17 and 18 to Fukuda ‘98. Petition, 51-52. Petitioners again provide no explanation why these opposing positions are justified.

C. Recent Director decisions confirm that Petitioners’ failure to explain inconsistent claim scope positions is fatal

Recent Director decisions underscore that Petitioners cannot freely switch claim scope positions between the district court and the Board, particularly where Petitioners do so without explanation. In *Revvo Technologies, Inc. v. Cerebrum Sensor Technologies, Inc.*, IPR2025-00632, Paper 20 (Dir. Rev. Nov. 3, 2025) (precedential), the Director explained that the Board’s claim construction rules are designed to minimize inconsistency between forums and discourage petitioners from seeking different constructions in different proceedings. The Director emphasized that, after the Office aligned PTAB claim construction with litigation, petitioners have “less flexibility to advance inconsistent arguments about claim scope,” and

“will instead be required to choose a single claim construction that best captures the meaning of the patent claim.” *Revvo*, Paper 20 at 4-5. Allowing petitioners to continue the practice of proposing different constructions across forums does not further the Office’s goal of “providing greater predictability and certainty in the patent system.” *Revvo*, at 4.

Similarly, in *Tesla, Inc. v. Intellectual Ventures II LLC*, IPR2025-00340, Paper 18 (Dir. Rev. Nov. 5, 2025) (informative), the Director vacated an institution decision where the petitioner argued in district court that certain terms were indefinite while simultaneously maintaining in the IPR that those same terms should be given their “plain and ordinary meaning.” *Tesla*, Paper 18 at 2-3. The same defect exists here. Southwest has argued in district court that the terms above are indefinite because they are open to multiple interpretations, and that the claim scope is therefore uncertain. Yet Petitioners now proceed in this IPR as if those same terms have a plain and ordinary meaning that can be applied to support Petitioners’ anticipation theory, without acknowledging or justifying the inconsistency. EX2004, 296-297, 302-303; Petition, 12, 35-38, 51-52.

Under *Cambridge Mobile Telematics*, Petitioners at a minimum were required to explain why inconsistent claim construction positions are warranted. Petitioners do not do so. That failure is an independent and sufficient basis to deny institution

because the Petitioners have not carried their burden to proffer coherent and legally sound claim construction positions.

IV. REFERENCES RELIED ON IN THE PETITION

A. Declaration of Michael Ian Shamos (EX1003)

Petitioners rely heavily on the Declaration of Michael Ian Shamos (EX1003) to supply technical “POSITA would have understood” bridges and other reasoning that must be grounded in record evidence, particularly where Petitioners’ sole ground is anticipation. The expert declaration is not prior art. It cannot substitute for a missing disclosure in the asserted reference, and it should be afforded little weight to the extent it merely repeats Petitioners’ attorney argument without independent technical analysis.

The Board has held, in a precedential decision, that where an expert declaration largely restates a petition’s conclusory assertions without providing additional substantive explanation, the declaration warrants little evidentiary weight. *Xerox Corp. v. Bytemark, Inc.*, IPR2022-00624, Paper 9 at 15-17 (P.T.A.B. Aug. 24, 2022) (precedential). The Federal Circuit has likewise affirmed discounting expert testimony that “merely repeated” a petitioner’s arguments “nearly verbatim” without meaningful independent analysis. *Facebook, Inc. v. Windy City Innovations, LLC*, 973 F.3d 1321, 1340-41 (Fed. Cir. 2020). These principles apply with full force here

because Petitioners repeatedly cite EX1003 for the very “bridging” inferences that must be supported by what the asserted prior art actually teaches.

A comparison of the Petition and the expert declaration confirms pervasive overlap across virtually every substantive portion of the Petition. This is not limited to shared, high-level background. The Petition’s technical narrative is repeatedly keyed to specific EX1003 paragraph citations, and in multiple places the expert declaration tracks the Petition’s wording and structure so closely that the Petition’s narrative functions as a template for the expert declaration, which lacks any independent expert analysis of its own.

Topic	Petition Location (by page number)	Declaration Location (by paragraph number)
‘080 Patent Overview	§ IV (pp. 5–7)	§ V (¶ 28–35)
Prosecution History Args	§ V (pp. 10–11)	§ VIII.K (¶ 89–90)
Fukuda ‘98 Description	§ X.A (pp. 14–19)	§ VIII.F (¶ 61–73)
Claim 1 Analysis	§ XI.A.1 (pp. 23–38)	§ X.A (¶ 123–160)
Claims 2–18 Analysis	§ XI.A.2–18 (pp. 38–52)	§ X.B–R (¶ 162–239)

Under *Xerox* and *Facebook*, that kind of rote repetition warrants little evidentiary weight because it does not provide independent expert analysis. This is

especially important here because Petitioners invoke EX1003 to support critical inferences that the asserted reference must disclose for anticipation, including foundational “POSITA would have understood” and similar bridging assertions offered in support of Petitioners’ theory of “spawning,” “intermediate condition,” and concurrency. Petition, 33; EX1003, ¶¶146-153.

B. Fukuda ‘98 (EX1008)

Fukuda ‘98 (EX1008), titled “Distributed coordination with MESSENGERS,” describes the MESSENGERS system as a distributed programming paradigm based on autonomous messages, called “Messengers,” that carry their own behavior in program form. EX1008, 291. Fukuda ‘98 explains that Messengers navigate an underlying computational network, communicate, and invoke compiled node-resident C functions at the nodes they visit, such that a distributed application may be viewed as a collection of node functions whose invocation and interoperation are orchestrated by Messengers. EX1008, 291. Fukuda ‘98 further explains that MESSENGERS separates computation from coordination, where computation is performed by individual node functions and coordination is the ordering of those function invocations and the movement of information among them via Messengers. EX1008, 291; EX1008, 305. Fukuda ‘98 also describes a shared “state space” that contains functions and data and is structured by being mapped onto a logical network, and it explains that Messengers have state

information and migrate to nodes containing data they wish to read, write, or modify and to nodes containing functions they wish to invoke. EX1008, 292.

Fukuda '98, Fukuda '01, and Fukuda '99 are part of a single MESSENGERS research and publication lineage addressing the same distributed programming paradigm based on mobile agents ("Messengers") navigating a logical network and coordinating via shared state. Fukuda '98 and Fukuda '01 share the same four authors -- Munehiro Fukuda, Lubomir F. Bic, Michael B. Dillencourt, and Fehmina Merchant. EX1008, 291; EX1009, 95. Fukuda '99 is authored by Fukuda, Bic, and Dillencourt, with Jason M. Cahill. EX2006, 188.

Fukuda '01's reference list expressly includes both "Distributed coordination with MESSENGERS" (1998) and "Messages versus Messengers in distributed programming" (1999), confirming that Fukuda '01 builds on and synthesizes the earlier MESSENGERS disclosures of Fukuda '98 and Fukuda '99. EX1009, 111. In substance, the three publications describe the same foundational MESSENGERS system architecture and programming primitives. Each describes a daemon-based execution environment deployed across physical nodes and an application-level "logical network" used as the navigational space for Messengers. EX1008, 310; EX1009, 96–97; EX2006, 190–191. Each also describes the shared-state variable model that distinguishes messenger variables (carried by a Messenger as it propagates) from node variables (resident at a node and shared by Messengers co-

located at that node), with network variables providing local network information. EX1008, 301; EX1009, 98; EX2006, 191–192.

The publications likewise describe the same navigational and coordination mechanisms used to implement distributed applications through Messenger movement and replication. Fukuda ‘99 describes hop as a navigational statement that moves and replicates Messengers across destination nodes in the logical network. EX2006, 192. Fukuda ‘01 similarly describes the hop statement and also describes “inject” as a task-control mechanism for initiating execution of additional Messengers with parameters and optional starting destinations. EX1009, 98–99. Fukuda ‘98 describes complementary coordination and interaction in the same paradigm, including interaction via shared node variables at logical nodes and invocation of compiled native-mode functions at visited nodes, and it provides examples in which node variables are used for distributed simulation. EX1008, 291–292; EX1008, 302–305.

V. PETITIONERS HAVE NOT MET THEIR BURDEN TO ESTABLISH ANTICIPATION UNDER 35 U.S.C §102

Petitioners bear the burden to demonstrate unpatentability on the sole ground they selected (i.e., that the challenged claims are anticipated by Fukuda ‘98), and they must do so in the Petition itself. As the Supreme Court has explained, “in an *inter partes* review the petitioner is master of its complaint,” and “the statute envisions that a petitioner will seek an *inter partes* review of a particular kind, one

guided by a petition describing ‘each claim challenged’ and ‘the grounds on which the challenge to each claim is based.’” *SAS Inst., Inc. v. Iancu*, 138 S. Ct. 1348, 1355 (2018). Consistent with that framework, “[i]n an *inter partes* review, the burden of persuasion is on the petitioner to prove ‘unpatentability by a preponderance of the evidence,’ 35 U.S.C. § 316(e), and that burden never shifts to the patentee.” *FanDuel, Inc. v. Interactive Games LLC*, 966 F.3d 1334, 1341 (Fed. Cir. 2020). Petitioners, therefore, must come forward with a coherent, record-supported anticipation theory that (i) identifies, for each challenged claim limitation, where the single asserted reference discloses the limitation, either expressly or by necessary implication, and (ii) demonstrates that the single asserted reference discloses each such limitation as arranged in the claim.

The governing regulations impose the same requirement. The Petition must include “[a] full statement of the reasons for relief requested, including a detailed explanation of the significance of the evidence including material facts,” 37 C.F.R. § 42.22(a)(2), and must explain both “[h]ow the challenged claim is to be construed” and “[h]ow the construed claim is unpatentable.” *Id.* § 42.104(b)(3)-(4). These requirements are particularly important where, as here, Petitioners rely on a single anticipation ground under § 102, because anticipation is an exacting, limitation-by-limitation inquiry that cannot be satisfied by shifting theories, conclusory expert ipse dixit, or unmoored “POSITA would have understood” bridges.

Here, Petitioners do not satisfy these statutory requirements. As set forth below, Petitioners’ anticipation theory fails on the merits because it does not provide a coherent, limitation-by-limitation mapping to Fukuda ‘98 that accounts for all of the claim limitations. The purported mappings of Fukuda ‘98 to claim 1 are deeply flawed. In some cases, the Petition either completely fails to address the claim language or fails to explain how Fukuda ‘98 meets the claim language. In other cases, the Petition relies on disparate aspects of Fukuda ‘98 for limitations that are connected by the claim language and, when doing so, fails to explain how Fukuda ‘98 discloses the claimed elements “arranged as in the claim,” as required to establish anticipation. In still other cases, the mappings in the Petition are inconsistent in that when the limitation first appears in the claim, the Petition maps one aspect of Fukuda ‘98 to the claim language, but when the same limitation is referenced later in the claim, the Petition maps a different aspect of Fukuda ‘98 to the limitation. Finally, some aspects of the mappings are so poorly explained that they are downright incoherent. Examples of the deficiencies in the mappings of Fukuda ‘98 to claim 1 are discussed next.

A. The Petition Fails To Explain How Fukuda ‘98 Discloses Establishing At Least Distributed Shared Variable That is A Single Variable That Includes Several Variables

Element [1a] requires “at least one distributed shared variable, wherein the at least one distributed shared variable is a single variable that includes several

variables.” Yet, in Ground 1 (the sole ground in the Petition), the Petition fails to address this requirement entirely. Indeed, the Petition’s discussion of Element [1a] - which spans pages 23-26 -- is devoid of any mention of the language in Element [1a] reciting that “the at least one distributed shared variable is a single variable that includes several variables.”

In purporting to map Element [1a] to Fukuda ‘98, the Petition maps the “distributed shared variable” recited in Element [1a] to a “node variable” or the “virtual time” variable in Fukuda ‘98:

Fukuda ‘98 discloses the “distributed shared variable” of claim [1a]. Consistent with the disclosures in the ‘080 specification and the distributed data of Pan, relied on by the Examiner, Fukuda ‘98 discloses “node variables,” which are examples of “distributed shared variables”: “Node variables are resident in nodes and shared by all Messengers currently running on the same logical node.”...

Fukuda ‘98 describes an example of a distributed shared variable, *i.e., virtual time*, which is “a global time line consistent over all physical nodes,” that is, “a globally shared variable that simulates the passage of time.” Petition, 23-24 (emphasis added, citations omitted).

However, the Petition nowhere alleges that either a “node variable” or the “virtual time” variable in Fukuda ‘98 is “a single variable that includes several variables,” as recited in Element [1a]. Despite having the burden to do so, the Petition provides no explanation whatsoever of how either a “node variable” or the “virtual time” variable in Fukuda ‘98 -- which the Petition maps to the “distributed shared variable” in Element [1a] -- is “a single variable that includes several variables.”

Petitioners had the burden of explaining how the challenged claims were to be interpreted, as well as how Fukuda ‘98 disclosed each and every limitation in claim 1. With respect to the requirement that “the at least one distributed shared variable is a single variable that includes several variables” in Element [1a], Petitioners clearly failed to meet this burden, because the Petition completely fails to address this aspect of the claim language.

For this reason alone, the Petition fails to explain how Fukuda ‘98 discloses all of the limitations of claim 1.

B. The Mappings In The Petition Impermissibly Pick And Choose Various Disclosures From Fukuda ‘98 And Fail To Demonstrate That The Reference Discloses All Of The Elements Of Claim 1 As Arranged In The Claim

Element [1c] recites:

transforming the at least one distributed sequential computing program into at least one distributed parallel computing program by spawning at least one child distributed sequential computing program from the at least one distributed sequential computing program when at least one intermediate condition occurs within the at least one distributed sequential computing program ...

In connection with Element [1c], the Petition (i) relies on the exec() statement from Fukuda ‘98 for “spawning at least one child,”¹ and (ii) separately relies on the

¹ Petition, 32 (one of the statements disclosed in Fukuda ‘98, “exec(), contains a main() function “which is spawned as a separate Unix process, i.e., operating concurrently with the invoking Messenger.”)

distributed simulation of “styrene inhalation” shown in Figure 4 of Fukuda ‘98 (and discussed at 304-305 of the reference) for “the at least one distributed sequential computing program” from which the child program is spawned.² Significantly, however, the distributed simulation of “styrene inhalation” (relied on in the Petition for the distributed sequential computing program from which the child program is spawned) does not use the *exec()* function (which the Petition alleges spawns the child program). The script for the distributed simulation of “styrene inhalation” is set forth in Figure 4 of Fukuda ‘98 and reproduced below:

```
(1) styrene_sim(n_iter, inhaled_sty) {
(2)   c_veinal == 0;
(3)   for (i = 0; i < n_iter; i++) {
(4)     func(name="new_lung_conc"; in=c_veinal, inhaled_sty; out = c_art);
(5)     n_msgs = number_of_organs;
(6)     hop(link="arterial");
(7)     func(name="new_organ_conc"; in=c_art; out=c_ven_o);
(8)     hop(link="veinal");
(9)     c_veinal += c_ven_o;
(10)    if (--n_msgs > 0)
(11)      exit;
(12)  }
(13) }
```

Fig. 4. Messenger script for styrene inhalation.

As is plain from the script set forth above, the distributed simulation of “styrene inhalation” as disclosed in Fukuda ‘98 does not disclose that the *main()* function contained an *exec()* statement. In fact, there is no mention at all of the

² Petition, 32-33 (“The distributed simulation example in Fukuda ‘98 demonstrates claim [1c]. ... “the distributed simulation is a distributed sequential computing program.”)

main() function contained in an *exec()* statement in any of the lines of the distributed simulation of “styrene inhalation.” Simply put, the Petition fails to explain how the *main()* function contained in the *exec()* statement disclosed in Fukuda ‘98 is used to spawn a child program from the distributed simulation of “styrene inhalation” disclosed in the reference, because there is no mention at all of the *main()* function contained in an *exec()* statement in any of the lines of the distributed simulation of “styrene inhalation.”

The sole ground in the Petition alleges that claim 1 is anticipated by Fukuda ‘98. “Because the hallmark of anticipation is prior invention, the prior art reference -- in order to anticipate under 35 U.S.C. § 102 -- must not only disclose all elements of the claim within the four corners of the document, but must also disclose those elements “arranged as in the claim.” *Net MoneyIN, Inc. v Verisign, Inc.*, 545 F.3d 1359, 1369 (Fed. Cir. 2008) (emphasis added), *citing*, *Connell v. Sears, Roebuck & Co.*, 722 F.2d 1542, 1548 (Fed. Cir. 1983). In order to anticipate, the prior art reference “must clearly and unequivocally disclose” the claimed invention “without any need for picking, choosing, and combining various disclosures not directly related to each other” in the reference. *Net MoneyIN*, 545 F.3d 1359, 1371 (citations omitted).

Thus, in order to establish anticipation, it was not enough for the Petition to point to the *main()* function contained in the *exec()* statement disclosed in one part

of Fukuda ‘98 for functionality that spawns a child program, and separately point to the distributed simulation of “styrene inhalation” disclosed in another part of Fukuda ‘98 for the distributed sequential computing program from which the child program is spawned. Instead, the Petition was additionally required to show that Fukuda ‘98 disclosed these teachings “arranged as in the claim.” The Petition clearly failed to carry this burden because the Petition fails to explain how Fukuda ‘98 discloses the use of the *main()* function contained in an *exec()* statement within the distributed simulation of “styrene inhalation.”

For this additional reason, the Petition fails to demonstrate that Fukuda ‘98 anticipates claim 1.

C. The Petition Fails To Present A Coherent Or Consistent Mapping Of The At Least One Intermediate Condition Recited Throughout Claim 1

The “at least one intermediate condition” is introduced in Element [1c] and referenced later in Element [1e] which refers back to “the at least one intermediate condition.” In connection with Element [1c], the Petition (at 33) alleges that the “intermediate condition” corresponds to “the arguments specified by in” in the *exec()* statement of Fukuda ‘98, while in connection with Element [1e], the Petition (at 37) alleges that the same “intermediate condition” comprises “the arguments specified by out” in the *exec()* statement. These mappings are inconsistent. Petitioners had the burden to identify a coherent, single “intermediate condition” in Fukuda ‘98 that

satisfies both limitations together, not one “intermediate condition” for element 1[c] and a different “intermediate condition” for element 1[e]. The Petition clearly failed to meet that burden.

While the Petition (at 33) alleges in connection with Element [1c] that the “intermediate condition” corresponds to “the arguments specified by *in*,” the Petition fails to explain how “the arguments specified by *in*” correspond to an “intermediate condition” as that term is used in the context of the ‘080 Patent. The Petitioners state that they “apply the *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) claim construction standard, but the Petition fails to provide a clear explanation of what Petitioners deem to be an “intermediate condition” or how “the arguments specified by *in*” in the *exec()* statement of Fukuda ‘98 equate to an “intermediate condition.” Petition, 12. In discussing the ‘080 Patent, the Petition (at 31) alleges that “the intermediate condition is the receipt of the intermediate result from part of the computation on the current node,” but the Petition never connects the dots by explaining how that proposition applies to “the arguments specified by *in*.” If the Petitioners’ position for Element [1c] was that “the receipt of” the “arguments specified by *in*” by the *main()* function corresponds to the “intermediate condition,” that point was nowhere clearly articulated in the Petition as required by the rules. More importantly, had the Petitioners actually taken the position in connection with Element [1c] that “the receipt of” the “arguments specified by *in*” by the *main()*

function corresponds to the “intermediate condition,” doing so would have clearly conflicted with Petitioners’ mapping of the “intermediate condition” in Element [1e] where Petitioners allege that the same “intermediate condition” comprises “the arguments specified by out.” Petition, 37.

Unlike “the receipt of” the “arguments specified by in” which are provided to the *main* function as inputs at the time when the function is invoked, the “arguments specified by out” are provided as outputs at a later point in time when the function has completed its work. The Petition fails to explain how “the intermediate condition” which is alleged to occur at one point in time when “each cycle begins” can correspond to events that occur at different points in time, i.e., when the *main* function is invoked and again when the *main* function has completed its work.

These inconsistencies are outcome-determinative for anticipation. Petitioners cannot satisfy element 1[c] by treating the “intermediate condition” as something that occurs at the beginning of a cycle or at process creation (“in”), while separately satisfying element 1[e] by redefining the same “intermediate condition” as a later, extended window that ends only after a decrement and by shifting the relevant artifact to “out” and post-return updates. Petition, 17, 33, 37. The claim requires a single “intermediate condition” that both triggers spawning and comprises an intermediate result required by the spawned child to continue computation. EX1001, 11:13-20, 11:24-27. Because Petitioners’ Petition advances inconsistent

identifications of the “intermediate condition” across elements 1[c] and 1[e], Petitioners have not provided the coherent, limitation-by-limitation showing required to establish anticipation. EX1001, 11:13-20, 11:24-27; Petition, 17, 33, 37.

Finally, the Petition appears to offer an alternative “distributed simulation cycle” narrative for the “intermediate condition” but fails to tie that theory to the remainder of the language of claim 1. Petitioners assert that the intermediate condition begins when each cycle begins and ends when the nmsgsr variable decrements, and they present this window as satisfying the “intermediate result” requirement. Petition, 37. Yet Petitioners do not identify the “one intermediate result” within that window, nor explain why any identified quantity is “required” by the spawned child to continue computation. EX1001, 11:22-27; Petition, 37.

In short, Petitioners fail to present a coherent and consistent identification of the claimed “one intermediate result” in Fukuda ‘98. For this additional reason, the Petition fails to explain how Fukuda ‘98 discloses all of the limitations of claim 1.

D. The Petition Fails To Explain How Fukuda ‘98 Performs The Parallel Processing And/Or Operations Recited In Claim 1

Element [1d] recites: “wherein the at least one distributed parallel computing program concurrently uses the at least one distributed sequential computing program and the at least one spawned child distributed sequential computing program to perform parallel processing and/or operations.” According to the

Petition, this limitation was the “point of novelty over the prior art.” Petition, 14. However, the Petition has failed to explain how Fukuda ‘98 discloses Element [1d].

As noted above, the Petition relies on the distributed simulation “for styrene inhalation” (set forth at 304-305 of Fukuda ‘98) in connection with the “spawned” program first introduced into the claim in Element [1c]. Petition, 33 (“The distributed simulation example in Fukuda ‘98 demonstrates claim [1c]”). The script for the styrene inhalation simulation in Fukuda ‘98 is reproduced again below (red annotations added). As shown in the red boxes below, the script for the styrene inhalation simulation in Fukuda ‘98 (at 305) does not use an *exec()* statement but instead uses two *func* statements:

```
(1) styrene_sim(n_iter, inhaled_sty) {  
(2)   c_veinal == 0;  
(3)   for (i = 0; i < n_iter; i++) {  
(4)     func(name="new_lung_conc"; in=c_veinal, inhaled_sty; out = c_art);  
(5)     n_msgrs = number_of_organs;  
(6)     hop(link="arterial");  
(7)     func(name="new_organ_conc"; in=c_art; out=c_ven_o);  
(8)     hop(link="veinal");  
(9)     c_veinal += c_ven_o;  
(10)    if (--n_msgrs > 0)  
(11)      exit;  
(12)  }  
(13) }
```

Significantly, Fukuda ‘98 explains the operation of the *func* statement as follows:

func(file=filename; name=func_name[;in=argument][;out=argument]):
Similar to *exec*, the parameter *filename* is a Unix path name specifying a file. This must contain one or more executable C functions (but not necessarily a complete program with a *main()* function). The function specified by the parameter *func_name* is invoked and the arguments

specified by in are passed to it. Unlike with *exec*, the function executes as part of the current Messenger, that it, as part of the MESSENGERS interpreter process. The invoking Messenger is blocked until the function terminates and returns its results to the Messenger script via the out argument. Fukuda '98 [EX1008], 302 (emphasis added).

Thus, in the distributed simulation “for styrene inhalation” (set forth at 304-305 of Fukuda '98), the *func* statements do not result in parallel processing being performed by the “spawned” program and the invoking Messenger because, when the *func* statement “spawns” another program, the invoking Messenger is blocked until the function terminates. Given that the invoking Messenger and “spawned” program in the distributed simulation do not operate in parallel, the Petition fails to explain how the distributed simulation in Fukuda '98 “concurrently uses the at least one distributed sequential computing program and the at least one spawned child distributed sequential computing program to perform parallel processing and/or operations,” as recited in Element [1d].

In a last ditch attempt to show that Element [1d] is met by Fukuda '98, the Petition cites to Figure 3 of Fukuda '98 (reproduced below) and alleges that it discloses Messengers that “are running concurrently.” Petition, 35. However, nothing about Figure 3 -- or the description of that figure in Fukuda '98 (at 303-304) -- explicitly or inherently discloses Messengers that “are running concurrently.”

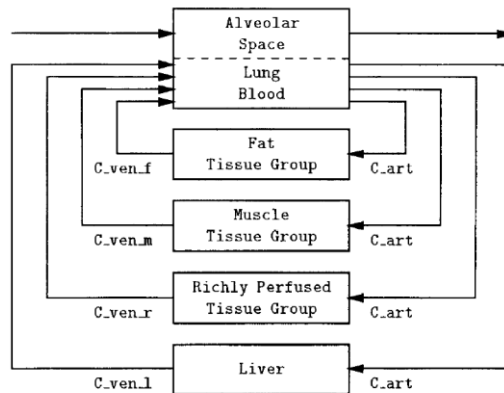


Fig. 3. A toxicology model.

Fukuda refers to Figure 3 as “A toxicology model.” What this model shows is that outputs of various boxes are provided as inputs to other boxes. However, the figure fails to expressly or inherently disclose that Messengers are running concurrently. On the contrary, as explained above, the script disclosed in Fukuda ‘98 (in Figure 4) for implementing the simulation illustrated in Figure 3 uses the *func* statement which does not result in concurrent processing being performed by the “spawned” program and the invoking Messenger.

For these additional reasons, the Petition fails to explain how Fukuda ‘98 discloses all of the limitations of claim 1.

E. The Petition’s Erroneous Allegation That Fukuda ‘98 Is Not Cumulative Of The Art In Front Of The Examiner Is Predicated On A False Factual Assertion

Petitioners’ assertion that “Ground 1 is not cumulative of the art of record” is predicated on Petitioners’ false statement that: “When examining the ‘080 patent application, the Examiner did not have Fukuda ‘01.” Petition, 13 and 20. That

statement is demonstrably false as evidenced by the face of the ‘080 Patent (reproduced below) which lists Fukuda ‘01 (highlighted in yellow) in the “Other Publications” section, with an asterisk indicating it was cited by the Examiner during examination.

(12) United States Patent Pan et al.	(10) Patent No.: US 7,712,080 B2 (45) Date of Patent: May 4, 2010
(54) SYSTEMS AND METHODS FOR PARALLEL DISTRIBUTED PROGRAMMING	Fukuda et al., Messengers: Distributed Programming Using Mobile Agents, Transactions of the SDPS, Dec. 2001, vol. 5, No. 4, pp. 95-112.*
(75) Inventors: Lei Pan , Irvine, CA (US); Lubomir R. Bic , Irvine, CA (US); Michael B. Dillencourt , Irvine, CA (US)	Stanislaw Chrobot, Sharing Variables in Distributed Memory, Fundamenta Informaticae 44 (2000) 63-81, IOS Press, pp. 63-81.* Suzuki et al., Self-Migrating Threads for Multi-Agent Applications, publication data: 1999, published by 1st IEEE computer societey international workshop, pp. 1-8.*
(73) Assignee: The Regents of the University of California , Oakland, CA (US)	Fukuda et al., Messengers versus Messengers in Distributed Programming, published by Journal of Parallel and Distributed Computing vol. 57, Issue 2 May 1999, pp. 1-8.*
(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1013 days.	Lei Pan, et al., <i>Distributed Sequential Computing</i> (pp. 223-239), Applied Parallel and Distributed Computing (Advances in Computation: Theory and Practice, vol. 16), (Y. Pan and L.T. Yang, Eds.), Nova Science Publishers, Inc., New York, 2004.
(21) Appl. No.: 10/850,842	Lei Pan, et al., <i>Facilitating Agent Navigation Using DSM—High Level Designs</i> (11 pages), The 7th Int'l Conf. on Integrated Design & Process Technology (IDPT03), Austin, Texas, Dec. 2003.
(22) Filed: May 21, 2004	
(65) Prior Publication Data US 2005/0039159 A1 Feb. 17, 2005	

(Continued)

The Examiner identified Fukuda ‘01 as “pertinent to the applicant’s disclosure” in the May 21, 2007 Office Action. EX1002, 171. The Examiner also cited Fukuda ‘99 (highlighted in green above) during the prosecution history, which is likewise identified on the face of the ‘080 Patent with an asterisk. EX2006.

Fukuda ‘98, Fukuda ‘01, and Fukuda ‘99 are part of a single MESSENGERS research and publication lineage addressing the same distributed programming paradigm based on mobile agents (“Messengers”) navigating a logical network and coordinating via shared state. Fukuda ‘98 and Fukuda ‘01 share the same four authors, Munehiro Fukuda, Lubomir F. Bic, Michael B. Dillencourt, and Fehmina Merchant. EX1008, 291; EX1009, 95. Fukuda ‘99 is authored by Fukuda, Bic, and Dillencourt, with Jason M. Cahill. EX2006, 188. In substance, the three publications

describe the same foundational MESSENGERS system architecture and programming primitives. Each describes a daemon-based execution environment deployed across physical nodes and an application-level “logical network” used as the navigational space for Messengers. EX1008, 310; EX1009, 96–97; EX2006, 190–191. Each also describes the shared-state variable model that distinguishes messenger variables (carried by a Messenger as it propagates) from node variables (resident at a node and shared by Messengers co-located at that node), with network variables providing local network information. EX1008, 301; EX1009, 98; EX2006, 191–192.

The publications likewise describe the same navigational and coordination mechanisms used to implement distributed applications through Messenger movement and replication. Fukuda ‘99 describes hop as a navigational statement that moves and replicates Messengers across destination nodes in the logical network. EX2006, 192. Fukuda ‘01 similarly describes hop and also describes inject as a task-control mechanism for initiating execution of additional Messengers with parameters and optional starting destinations. EX1009, 98–99. Fukuda ‘98 describes complementary coordination and interaction in the same paradigm, including interaction via shared node variables at logical nodes and invocation of compiled native-mode functions at visited nodes, and it provides examples in which node

variables are used as shared coordination state for distributed simulation. EX1008, 291–292; EX1008, 302–305.

The Petition itself acknowledges that Fukuda ‘01 (identified as EX1009) is cumulative of Fukuda ‘98. See, e.g., Petition, 20 (“Like Fukuda ‘98, Fukuda ‘01 describes a programming paradigm called Messengers”) and 23 (“Fukuda ‘01 describes the principles of operation and the programming of the MESSENGER system.”) The Petition then uses Fukuda ‘01 (falsely alleged as not being before the Examiner) in an attempt to explain what Fukuda ‘98 means. See, e.g., Petition, 33, note 6 (“A POSITA would have confirmed, from Ex.1009 at 99, that the “inject” statement in MESSENGERS “initiates (injects) the execution of one or more instances of another Messenger”) and 34, note 7 (“A POSITA would have confirmed, from Ex.1009 at 99, that, when invoking the “inject” statement in MESSENGERS, parameters are passed to the newly created Messenger: “[t]he parameters specify the Messenger script to be injected and the initial parameters to be passed to the newly created Messenger”).

While the Petition (at 13) proffers a conclusory allegation that “Ground 1 is not cumulative of the art of record,” the Petition never explains what Fukuda ‘98 allegedly adds over the MESSENGERS publications, such as Fukuda ‘99 and Fukuda ‘01, which the Examiner identified and found wanting. Given the numerous deficiencies in Petitioners’ anticipation analysis based on Fukuda ‘98 -- which has

not been shown to be materially different from the other MESSENGERS publications like Fukuda '99 and Fukuda '01 which the Examiner found -- it is clear that the Examiner correctly concluded that the body of prior art encompassed by the MESSENGERS references fails to render the claims of the '080 Patent unpatentable. The fact that the Petition (at 20) needed to resort to a false narrative that "the Examiner did not have Fukuda '01" exemplifies the weaknesses in the case put forward by the Petitioners, and underscores why the Petition fails to present a reasonable likelihood of success on the merits.

F. The Petition Fails To Present A Reasonable Likelihood Of Success With Respect To Any Of The Challenged Claims

For the reasons set forth in Sections V(A)-(E) above, the Petition has failed to demonstrate a reasonable likelihood of success with respect to independent claim 1. In connection with independent claim 9 which contains substantially the same language of claim 1, the Petition (at 44-47) relies on its analysis of claim 1. The Petition's mappings for claim 9 therefore suffer from the same deficiencies as those identified above for claim 1, and, for the same reasons, the Petition has failed to demonstrate a reasonable likelihood of success with respect to independent claim 9. The Petition's mappings for the remaining Challenged Claims (i.e., dependent claims 2-8 and 10-18) similarly fail to demonstrate a reasonable likelihood of success because the mapping for each such dependent claim is predicated on a deficient mapping of the base claim.

VI. CONCLUSION

For the reasons set forth above, the Petition does not demonstrate a reasonable likelihood that Petitioners will prevail in proving that any challenged claim is anticipated by Fukuda '98 (EX1008) under 35 U.S.C. § 102. Petitioners chose a single-reference anticipation theory and, as the master of their Petition, were required to provide a complete and internally coherent statement of the reasons for relief, including a consistent claim construction premise and a limitation-by-limitation mapping grounded in what Fukuda '98 discloses. The Petition does not satisfy that burden.

First, the Petition fails on the claim limitations that matter most to the challenged independent claims. Petitioners do not identify what the claimed “at least one intermediate condition” is in Fukuda '98 in a manner that is tethered to the claim language requiring that the intermediate condition occur “within” the distributed sequential computing program. Instead, the Petition offers shifting descriptions, including a “cycle begins” formulation and other generalized characterizations, without explaining how those concepts constitute the claimed intermediate condition in the dependency-driven sense required by the claims. Relatedly, Petitioners’ mapping is internally inconsistent: Petitioners use one conception of “intermediate condition” to satisfy the spawning requirement in element 1[c], but then rely on a

different conception or time window to satisfy element 1[e], even though the claim uses the same “intermediate condition” in both limitations. This inconsistency alone confirms that Petitioners have not provided the coherent, limitation-by-limitation showing required for anticipation.

Second, Petitioners never identify a concrete “one intermediate result” in Fukuda ‘98 that meets the claim’s express dependency requirement. The Petition does not explain what specific “intermediate result” is produced at the intermediate condition and why that particular intermediate result is “required by the spawned child” to continue computation. Petitioners’ reliance on process I/O concepts, counters, or cycle boundaries does not satisfy the claim language, and the Petition never closes the gap between generic concurrency and the claims’ requirement that the child’s continued computation depends on the intermediate result. In a single-reference anticipation case, that missing explanation is fatal.

Third, Petitioners’ anticipation showing is not supported by a consistent, legally sound claim scope position. Petitioner Southwest, has taken the position in district court that key claim terms are indefinite because they are open to multiple interpretations and render the scope uncertain, while simultaneously taking the opposite approach here by asserting the same terms can be construed under *Phillips* and mapped to Fukuda ‘98. Petitioners provide no explanation for why these

inconsistent claim scope positions are warranted. That failure further undermines Petitioners’ attempt to meet their burden in this proceeding.

Fourth, the Petition also fails to support the threshold prosecution-history propositions that Petitioners chose to advance. Petitioners assert that their ground is “not cumulative of the art of record” because Fukuda ‘98 was not before the Examiner, yet the prosecution history reflects that the Examiner considered closely related MESSENGERS publications, including Fukuda ‘01 (EX1009) and Fukuda ‘99 (EX2006), and the Petition never provides a comparison-based explanation of what Fukuda ‘98 allegedly adds beyond those record references and why any difference matters to the challenged claims. Petitioners’ own reliance on Fukuda ‘01 to supply core MESSENGERS semantics only heightens the need for such an explanation. The Petition does not provide one.

Finally, Petitioners’ heavy reliance on their expert declaration that parrots attorney argument verbatim does not cure the Petition’s deficiencies. The Board and the Federal Circuit have repeatedly held that expert testimony is entitled to little weight where it merely repeats the Petition’s advocacy without providing meaningful independent analysis, and an expert declaration cannot substitute for a missing disclosure in the asserted prior art in an anticipation challenge. Here, the declaration largely tracks the Petition’s narrative and is cited to supply “POSITA

would have understood” bridges where the Petition’s single-reference showing is otherwise lacking.

For all of these reasons, Petitioners have not met their burden to establish a reasonable likelihood of prevailing on their sole anticipation ground. Patent Owner therefore respectfully requests that the Board deny institution.

Respectfully submitted,

Dated: December 29, 2025

By: /Brandon R. Theiss/
Brandon R. Theiss
Registration No. 70,507
Counsel for Patent Owner
Intellectual Ventures I LLC

CERTIFICATE OF COMPLIANCE WITH WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), I certify that this Patent Owner Preliminary Response complies with the type-volume limits of 37 C.F.R. § 42.24(b)(1) because it contains 8,637 words, excluding the parts of this Patent Owner's Preliminary Response that are exempted by 37 C.F.R. § 42.24(a), according to the word processing system used to prepare this Patent Owner's Preliminary Response.

Respectfully submitted,

Dated: December 29, 2025

By: /Brandon R. Theiss/
Brandon R. Theiss
Registration No. 70,507
Counsel for Patent Owner
Intellectual Ventures I LLC

CERTIFICATE OF SERVICE

The undersigned certifies that pursuant to 37 C.F.R. § 42.6(e), a copy of the foregoing **PATENT OWNER'S PRELIMINARY RESPONSE** was served via email (as consented to by counsel) on December 29, 2025 to lead and backup counsel of record for Petitioners as follows:

John B. Campbell
jcampbell@McKoolSmith.com
MCKOOL SMITH, P.C.
303 Colorado Street, Suite 2100
Austin, TX 78701

Alan Block
ablock@mckoolsmith.com
McKool Smith, P.C.
MCKOOL SMITH, P.C.
300 South Grand Avenue, Suite 2900
Los Angeles, California 90071

Casey Shomaker
cshomaker@mckoolsmith.com
McKool Smith, P.C.
300 Crescent Court, Suite 1200
Dallas, TX 75201

Keith D. Harden
kharden@munckwilson.com
Munck Wilson Mandala, LLP
2000 McKinney Ave., Ste. 1900
Dallas, Texas 75201

S. Wallace Dunwoody
wdunwoody@munckwilson.com
Munck Wilson Mandala, LLP
2000 McKinney Ave., Ste. 1900
Dallas, Texas 75201

Michael C. Wilson
mwilson@munckwilson.com
Munck Wilson Mandala, LLP
2000 McKinney Ave., Ste. 1900
Dallas, Texas 75201

AA_Intellectual_Ventures@mckoolsmith.com

SWA-IV@munckwilson.com

Dated: December 29, 2025

By: /Brandon R. Theiss/
Brandon R. Theiss
Registration No. 70,507
Counsel for Patent Owner
Intellectual Ventures I LLC