

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent of: Sandstrom
U.S. Patent No.: 11,150,948 Attorney Docket No.: 42342-0120IP1
Issue Date: October 19, 2021
Appl. Serial No.: 17/212,903
Filing Date: March 25, 2021
Title: MANAGING PROGRAMMABLE LOGIC-BASED PROCESSING
UNIT ALLOCATION ON A PARALLEL DATA PROCESSING
PLATFORM

Mail Stop Patent Board

Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

**PETITION FOR *INTER PARTES* REVIEW OF UNITED STATES PATENT
NO. 11,150,948 PURSUANT TO 35 U.S.C. §§ 311–319, 37 C.F.R. § 42**

TABLE OF CONTENTS

I.	REQUIREMENTS FOR IPR UNDER 37 C.F.R. § 42.104.....	1
A.	Standing.....	1
B.	Challenge and Relief Requested	1
C.	Priority Date	2
II.	PTAB DISCRETION SHOULD NOT PRECLUDE INSTITUTION	3
A.	The <i>Fintiv</i> Factors Weigh in Favor of Institution— 35 U.S.C. § 314(a).....	3
B.	Petition’s New Art Warrants Institution—35 U.S.C. § 325(d)	5
III.	SUMMARY OF THE ’948 PATENT	9
A.	Brief Description	9
B.	Prosecution History	13
C.	Level of Ordinary Skill in the Art	15
IV.	CLAIM CONSTRUCTION	15
V.	THE CHALLENGED CLAIMS ARE UNPATENTABLE.....	16
A.	[GROUND 1] – Nollet in view of Tuan and Seo Renders Claims 1-15 Obvious.....	16
1.	Overview of Nollet (EX1004)	16
2.	Overview of Tuan (EX1005)	22
3.	Overview of Seo (EX1006)	28
4.	The Nollet-Tuan Combination.....	30
5.	The Nollet-Tuan-Seo Combination	34
6.	Predictable Results.....	36

7.	Claim 1	37
8.	Claim 2	59
9.	Claim 3	60
10.	Claim 4	61
11.	Claim 5	61
12.	Claim 6	62
13.	Claim 7	64
14.	Claim 8	65
15.	Claim 9	65
16.	Claim 10	65
17.	Claim 11	65
18.	Claim 12	68
19.	Claim 13	68
20.	Claim 14	68
21.	Claim 15	68
VI.	MANDATORY NOTICES UNDER 37 C.F.R. §42.8	69
A.	Real Parties-In-Interest Under 37 C.F.R. §42.8(b)(1)	69
B.	Related Matters Under 37 C.F.R. §42.8(b)(2)	69
C.	Lead And Back-Up Counsel Under 37 C.F.R. §42.8(b)(3)	69
D.	Service Information	70
VII.	PAYMENT OF FEES – 37 C.F.R. §42.103	70
VIII.	CONCLUSION	70

EXHIBITS

- EX1001 U.S. Patent No. 11,150,948 to Sandstrom (“the ’948 patent”)
- EX1002 Excerpts from the Prosecution History of the ’948 patent (“the Prosecution History”)
- EX1003 Declaration of Dr. Henry Houh
- EX1004 U.S. Patent Application Publication No. 20090187756A1 to Nollet et al. (“Nollet”)
- EX1005 Vu Manh Tuan. “A Study on a Multitasking Environment for Dynamically Reconfigurable Processors.” (2009) (“Tuan”)
- EX1006 Euseong Seo *et al.*, *Energy Efficient Scheduling of Real-Time Tasks on Multicore Processors*, 19 IEEE Transactions On Parallel and Distributed Systems 1540 (Nov. 2008) (“Seo”)
- EX1007 Redaelli, Francesco, et al. “Reconfiguration aware task scheduling for multi-fpga systems.” 4th HiPEAC Workshop on Reconfigurable Computing (2010) (“Redaelli”)
- EX1008 Ewerson Luiz de Souza Carvalho *et al.*, *Dynamic Task Mapping for MPSoCs*, 27 IEEE Design & Test of Computers 26 (Nov. 2010)
- EX1009 Jose Luis Nunez-Yanez *et al.*, *Dynamic Voltage Scaling in a FPGA-Based System-On-Chip*, 2007 International Conference on Field Programmable Logic and Applications, 459-462 (Aug. 2007)
- EX1010 U.S. Pub. No. 2012/0303809 to Parveen Patel *et al.* (“Patel”)

EX1011	Excerpts from Kai Hwang, Zhiwei Xu, SCALABLE PARALLEL COMPUTING – TECHNOLOGY, ARCHITECTURE PROGRAMMING (1998)
EX1012	Declaration of June Ann Munford Dated March 30, 2022
EX1013	Kuen Hung Tsoi and Wayne Luk, <i>Axel: A Heterogeneous Cluster with FPGAs and GPUs</i> , FPGA '10: Proceedings of the 18th annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 115-124 (Feb. 2010)
EX1014	Amir H. Hormati, <i>Flexstream: Adaptive Compilation of Streaming Applications for Heterogeneous Architectures</i> , 2009 18th International Conference on Parallel Architectures and Compilation Techniques (Sep. 2009)
EX1015	Declaration of June Ann Munford Dated April 14, 2023
EX1100	Complaint for Patent Infringement in <i>ThroughPuter, Inc. v. Microsoft Corp.</i> , No. 2:22-CV-00492 (W.D. Wash.)
EX1101	First Amended Complaint for Patent Infringement, <i>ThroughPuter, Inc. v. Microsoft Corp.</i> , Consolidated Case Nos. 2:22-cv-00344-BJR and 2:22-cv-00492-BJR (June 3, 2022)
EX1102	Order Staying Case Pending <i>Inter Partes</i> Review, <i>ThroughPuter, Inc. v. Microsoft Corp.</i> , Case No. 2:22-cv-00344-BJR (August 18, 2022)

CLAIM LISTING

Element	Claim Language
[1.0]	A method on a parallel data processing platform comprising a plurality of programmable logic-based processing units for accelerating a service comprising at least a first program and a second program, different than the first program, the method comprising:
[1.1]	forming a first set of inter-task communication paths connecting a first set of programmable logic-based processing units of the plurality of programmable logic-based processing units into a first multi-stage program instance,
[1.2]	wherein the first multi-stage program instance is configured to accelerate the first program, and
[1.3]	wherein the first set of programmable logic-based processing units are interconnected using reconfigurable cross-connects;
[1.4]	forming a second set of inter-task communication paths connecting a second set of programmable logic-based processing units of the plurality of programmable logic-based processing units to a second multi-stage program instance,
[1.5]	wherein the second multi-stage program instance is configured to accelerate the second program, and
[1.6]	wherein the second set of programmable logic-based processing units are interconnected using reconfigurable cross-connects;
[1.7]	in response to a first increased demand for the first multi-stage program instance determined by monitoring a first characteristic corresponding to the first set of programmable logic-based processing units, based on an adaptive optimized resource allocation policy, adding at least one more programmable logic-based processing unit to the first set of programmable logic-based processing units from the plurality of programmable logic-based processing units; and
[1.8]	in response to a second increased demand for the second multi-stage program instance determined by monitoring a second characteristic different from the first characteristic, corresponding to the second set

Element	Claim Language
	of programmable logic-based processing units, based on the adaptive optimized resource allocation policy, adding at least one more programmable logic-based processing unit to the second set of programmable logic-based processing units from the plurality of programmable logic-based processing units.
[2.0]	The method of claim 1, wherein the allocation policy relates to balancing load among programmable logic-based processing units.
[3.0]	The method of claim 1, wherein each of the first set of programmable logic-based processing units comprises a field programmable gate array (FPGA).
[4.0]	The method of claim 1, wherein each of the second set of programmable logic-based processing units comprises a field programmable gate array (FPGA).
[5.0]	The method of claim 1, wherein the first characteristic or the second characteristic is load or utilization.
[6.0]	A non-transitory computer program product comprising instructions configured to perform a method parallel data processing platform comprising a plurality of programmable logic-based processing units for accelerating a service comprising at least a first program and a second program different than the first program, the method comprising:
[6.1]	forming a first set of inter-task communication paths connecting a first set of programmable logic-based processing units of the plurality of programmable logic-based processing units into a first multi-stage program instance,
[6.2]	wherein the first multi-stage program instance is configured to accelerate the first program, and
[6.3]	wherein the first set of programmable logic-based processing units are interconnected using reconfigurable cross-connects;

Element	Claim Language
[6.4]	forming a second set of inter-task communication paths connecting a second set of programmable logic-based processing units of the plurality of programmable logic-based processing units to a second multi-stage program instance,
[6.5]	wherein the second multi-stage program instance is configured to accelerate the second program, and
[6.6]	wherein the second set of programmable logic-based processing units are interconnected using reconfigurable cross-connects;
[6.7]	in response to a first increased demand for the first multi-stage program instance determined by monitoring a first characteristic corresponding to the first set of programmable logic-based processing units, based on an adaptive optimized resource allocation policy, adding at least one more programmable logic-based processing unit to the first set of programmable logic-based processing units from the plurality of programmable logic-based processing units; and
[6.8]	in response to a second increased demand for the second multi-stage program instance determined by monitoring a second characteristic different from the first characteristic, corresponding to the second set of programmable logic-based processing units, based on the adaptive optimized resource allocation policy, adding at least one more programmable logic-based processing unit to the second set of programmable logic-based processing units from the plurality of programmable logic-based processing units.
[7.0]	The non-transitory computer program product of claim 6, wherein the allocation policy relates to balancing load among programmable logic-based processing units.
[8.0]	The non-transitory computer program product of claim 6, wherein each of the first set of programmable logic-based processing units comprises a field programmable gate array (FPGA).
[9.0]	The non-transitory computer program product of claim 6, wherein each of the second set of programmable logic-based processing units comprises a field programmable gate array (FPGA).

Element	Claim Language
[10.0]	The non-transitory computer program product of claim 6, wherein the first characteristic or the second characteristic is load or utilization.
[11.0]	A system comprising:
[11.1]	a plurality of programmable logic-based processing units, each processing unit being configurable in at least a first configuration and a second configuration, different from the first configuration;
[11.2]	a first set of inter-task communication paths connecting a first set of programmable logic-based processing units of the plurality of programmable logic-based processing units into a first multi-stage program instance,
[11.3]	wherein the first-multi-stage program instance is configured to accelerate the first program, and
[11.4]	wherein the first set of programmable logic-based processing units are interconnected using reconfigurable cross-connects;
[11.5]	a second set of inter-task communication paths connecting a second set of programmable logic-based processing units of the plurality of programmable logic-based processing units to a second multi-stage program instance configured to execute the second program, and
[11.6]	wherein the second set of programmable logic-based processing units are interconnected using reconfigurable cross-connects; and
[11.7]	a hardware logic and/or one or more processors executing computer-executable instructions configured to
[11.8]	(1) add, based on an allocation policy, in response to a first increased demand for the first multi-stage program instance determined by monitoring a first characteristic corresponding to the first set of programmable logic-based processing units, at least one more programmable logic-based processing unit to the first set of programmable logic-based processing units from the plurality of programmable logic-based processing units, and

Element	Claim Language
[11.9]	(2) add, based on the allocation policy, in response to a second increased demand for the second multi-stage program instance determined by monitoring a second characteristic, different from the first characteristic, corresponding to the second set of programmable logic-based processing units, at least one more programmable logic-based processing unit to the second set of programmable logic-based processing units from the plurality of programmable logic-based processing units.
[12.0]	The system of claim 11, wherein the allocation policy relates to balancing load among programmable logic-based processing units.
[13.0]	The system of claim 11, wherein each of the first set of programmable logic-based processing units comprises a field programmable gate array (FPGA).
[14.0]	The system of claim 11, wherein each of the second set of programmable logic-based processing units comprises a field programmable gate array (FPGA).
[15.0]	The system of claim 11, wherein the first characteristic or the second characteristic is load or utilization.

Microsoft Corporation (“Petitioner” or “Microsoft”) petitions for *Inter Partes* Review (“IPR”) of claims 1-15 (“Challenged Claims”) of U.S. Patent No. 11,150,948 (“the ’948 patent”). Microsoft submits that this Petition demonstrates a reasonable likelihood of prevailing with respect to at least one Challenged Claim, and respectfully requests institution of an IPR and cancellation of all Challenged Claims as unpatentable.

I. REQUIREMENTS FOR IPR UNDER 37 C.F.R. § 42.104

A. Standing

Microsoft certifies that the ’948 patent is available for IPR. The present Petition is being filed within one year of service of a complaint against Microsoft in *ThroughPuter, Inc. v. Microsoft Corp.* (WDWA-2-22-cv-00492). Microsoft is not barred or estopped from requesting this review challenging the Challenged Claims on the below-identified ground.

B. Challenge and Relief Requested

Microsoft requests IPR and cancellation of the Challenged Claims on the ground set forth in the table shown below. Additional explanation and support for the ground is set forth in the declaration of Dr. Henry H. Houh (EX1003), referenced throughout this Petition.

Ground	Claims	Basis
1	1-15	§103: Nollet, Tuan, and Seo

Applied references are prior art on the bases below:

Reference	Date(s)	Basis
Nollet	July 23, 2009	§102(b)
Tuan	October 2009 ¹	§102(b)
Seo	June 17, 2008 ²	§102(b)

C. Priority Date

The '948 patent claims priority to several patent applications, the earliest of which is U.S. Patent Application No. 13/297,455, filed on November 16, 2011.

The patent also claims priority to several provisional applications, the earliest of which is provisional application No. 62/556,065, filed on November 4, 2011.

¹ EX1012, ¶8 (authenticating public availability date); EX1015, ¶¶7-9.

² EX1006, 1 (“Manuscript received 29 Oct. 2007; accepted 13 June 2008; published online 17 June 2008.”); EX1015, ¶¶10-13; *Hulu, LLC v. Sound View Innovations, LLC*, IPR2018-01039, Paper 29 at 16 (PTAB Dec. 20, 2019) (precedential) (noting that Petitioner had met its burden sufficiently for institution by submitting a copy of a reference that bore “a copyright date of 1990, a printing date of November 1992, and an ISBN date of 8/94,” and was published by “an established publisher”); *VidStream LLC v. Twitter, Inc.*, 981 F.3d 1060, 1065 (Fed. Cir. 2020) (“When there is an established publisher there is a presumption of public accessibility as of the publication date.”).

EX1001, Cover page. Thus, the earliest possible priority date of the '948 patent is November 4, 2011 ("Critical Date").³

Each prior art reference applied in this petition predates the Critical Date.

II. PTAB DISCRETION SHOULD NOT PRECLUDE INSTITUTION

A. The *Fintiv* Factors Weigh in Favor of Institution— 35 U.S.C. § 314(a)

The *Fintiv* factors weigh heavily against exercise of the Board's discretion to deny institution under 35 U.S.C. § 314(a). *See Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (PTAB Mar. 20, 2020) (precedential) ("*Fintiv*").

Factor 1 strongly favors institution. ThroughPuter currently asserts the '948 Patent against Microsoft in litigation before the U.S. District Court for the Western District Washington (Case No. 2:22-cv-344-BJR) ("the Related Litigation"). MS-1100 (original complaint); MS-1101 (first amended complaint in proceeding consolidated with the Related Litigation). Microsoft's motion for a stay of the Related Litigation (filed April 18, 2022) was granted on August 18, 2022. MS-1102. Spe-

³ Microsoft reserves the right to challenge the '948 patent's priority claims in any appropriate forum. Solely for purposes of this petition, it is assumed that the priority date of the '948 patent is November 4, 2011, and the applied references qualify as prior art under that assumption.

cifically, the Court’s order stayed the Related Litigation “pending resolution of Microsoft’s petitions for *inter partes* review.” *Id.* The stay of the Related Litigation “strongly weigh[s] against exercising the authority to deny institution.” *Fintiv*, 6. As the Board recognized in *Fintiv*, “[a] district court stay of the litigation pending resolution of the PTAB trial allays concerns about inefficiency and duplication of efforts.”

Factor 2 strongly favors institution. With the Related Litigation stayed pending resolution of Microsoft’s IPRs, no trial date is currently set in the district court proceeding. The Board’s final written decision in the present IPR will necessarily precede any district court trial while the stay remains in place pending the PTAB’s consideration of Microsoft’s IPRs.

Factor 3 strongly favors institution. At the time the stay was ordered, the Related Litigation was still in an early phase. *Markman* had not yet transpired, and indeed the parties had not even exchanged terms for construction let alone begun claim construction briefing before the stay. Likewise, no experts had weighed in on any issue in the proceeding, and Microsoft had not served any invalidity contentions. The parties’ and the Court’s investment in the Related Litigation has been minimal.

Factor 4 favors institution. With the Related Litigation having been stayed, there will be no overlapping issues between an IPR instituted based on this Petition

and the district court proceeding. And, as noted in Factor 3, Microsoft has served no invalidity contentions in the Related Litigation pertaining to the '948 Patent.

Factor 5 (whether petitioner is also the defendant in district court) is neutral.

Factor 6 strongly favors institution. As demonstrated below, the merits of this Petition are particularly strong. This is also Microsoft's only challenge to the '948 Patent that has ever been or is currently before the Board, making considerations related to follow-on petitions moot.

In short, not one of the *Fintiv* factors weighs against institution, and at least five factors weigh heavily in favor of institution. Taken together, the Board should not exercise discretion to deny institution of IPR.

B. Petition's New Art Warrants Institution—35 U.S.C. § 325(d)

The *Advanced Bionics* framework sets forth a two-part inquiry to determine whether a §325(d) discretionary denial is warranted. *Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6 (PTAB Feb. 13, 2020) (precedential); *Becton, Dickinson & Co. v. B. Braun Melsungen AG*, IPR2017-01586, Paper 8 (PTAB Dec. 15, 2017) (precedential as to Section III.C.5, first paragraph). Non-limiting factors (a), (b), and (d) articulated in *Becton, Dickinson* apply to the first part of the framework, while factors (c), (e), and (f) apply to

the second part. *Advanced Bionics*, IPR2019-01469, Paper 6, 8. As explained below, application of the *Advanced Bionics* framework to this case reveals that discretionary denial under §325(d) is not warranted.

Advanced Bionics Part 1—The Office never considered the disclosure found in Seo or Tuan – two of the three references advanced in combination – when allowing the ’948 patent to issue. EX1003, ¶¶47-48. Nor did the Office address, through this or substantially similar prior art, the combination that is advanced by this Petition through integration of Seo and Tuan into Nollet. *Id.* Equally significant, the new disclosures of Seo and Tuan are materially different from art relied upon by the Examiner. *Id.*

Notably, the only reference cited during prosecution was U.S. Pub. No. 2012/0303809 (Patel). EX1002, 322. The Patel reference was listed as “prior art of record and not relied upon,” and Patel was never specifically discussed by either the Examiner or Applicant. *Id.* Even so, Patel relates to the communication infrastructure (e.g., packet handling) used for load balancing in a distributed network-based computing system. *See* EX1010, Abstract, claim 1. This is an entirely different context than found in Nollet, Tuan, and Seo, which all relate to parallel processing using multicore/multinode processors in mobile systems. *See* Section V.A.1-3, *infra*.

Becton Factors [a], [b], and [d] counsel in favor of institution under Part 1 of

the *Advanced Bionics* framework. The Petition relies on Tuan and/or Seo—in combination with Nollet—with respect to every feature of the '948 patent. *See, e.g., SHDS, Inc., et al. v. Truinject Corp.*, IPR2020-00935, Paper 12 at 9-12 (PTAB Nov. 17, 2020) (declining to deny institution under 325(d) where uncited references in the petition addressed a limitation that led to allowance). Neither Tuan nor Seo were cited or discussed during prosecution (*Becton* Factor [a]); nor are they cumulative of other art relied on by the Examiner, such as Nollet (*Becton* Factor [b]). There are also no overlapping arguments (*Becton* Factor [d]) between the prosecution record and the Petition given that the Examiner raised no specific prior art arguments. Thus, *Becton, Dickinson* factors (a) and (b) favor Petitioner.⁴

⁴ In IPR2022-00757, the Board denied institution of a petition against U.S. Patent No. 10,963,306, to which the '948 Patent claims priority. *Microsoft Corp. v. Throughputer, Inc.*, IPR2022-00757, Paper 10 (PTAB Nov. 1, 2022). The Board's prior institution decision focused on a limitation of the '306 Patent that recited: "wherein the load-adaptive allocation policy is configured to facilitate minimizing reconfiguring the plurality of reconfigurable logic-based processing units." *Id.* at 17-20. Neither that limitation nor any similar limitation is present in any of the claims of the '948 Patent, so the conclusion of the Institution Decision rendered with respect to the '306 Patent is irrelevant in this proceeding.

Advanced Bionics Part 2—The Board need not reach part 2 of the *Advanced Bionics* framework since part 1 is not satisfied. Nonetheless, each of *Becton, Dickinson* factors (c), (e), and (f) favors Petitioner.

The analysis of factor (c) is straightforward. The Office was not made aware of Seo or Tuan during examination. *See* EX1001, 1-6. Thus, Seo and Tuan were never evaluated, nor were either – singularly or in combination with each other or another reference – the basis of any prior art rejection during examination.

By not considering Seo, Tuan or combinations based on them, the Office erred in its evaluation of the prior art, thus warranting reconsideration. *Advanced Bionics*, Paper 6, 10 (clarifying that a petitioner need not prove material error through any affirmative act of the Patent Office; instead “if the record of the Office’s previous consideration of the art is not well developed or silent, then a petitioner may show the Office erred by overlooking something persuasive under factors (e) and (f).”). Indeed, the Office issued no substantive action showing or explaining its consideration of the art, leaving no record against which Petitioner can assess the error of any affirmative acts or conclusions. *Id.*

Ground 1 discussed in Section V demonstrates how further consideration is warranted. There is no evidence within the prosecution history that the Examiner considered teachings like those cited with respect to Ground 1, much less that the Examiner considered the proposed combination of these teachings as supported by

the evidence cited throughout this Petition and Dr. Houh’s accompanying declaration. *See* EX1003, ¶¶47-48.

Thus, *Becton, Dickinson* factors (c), (e), and (f) weigh in favor of institution.

III. SUMMARY OF THE ’948 PATENT

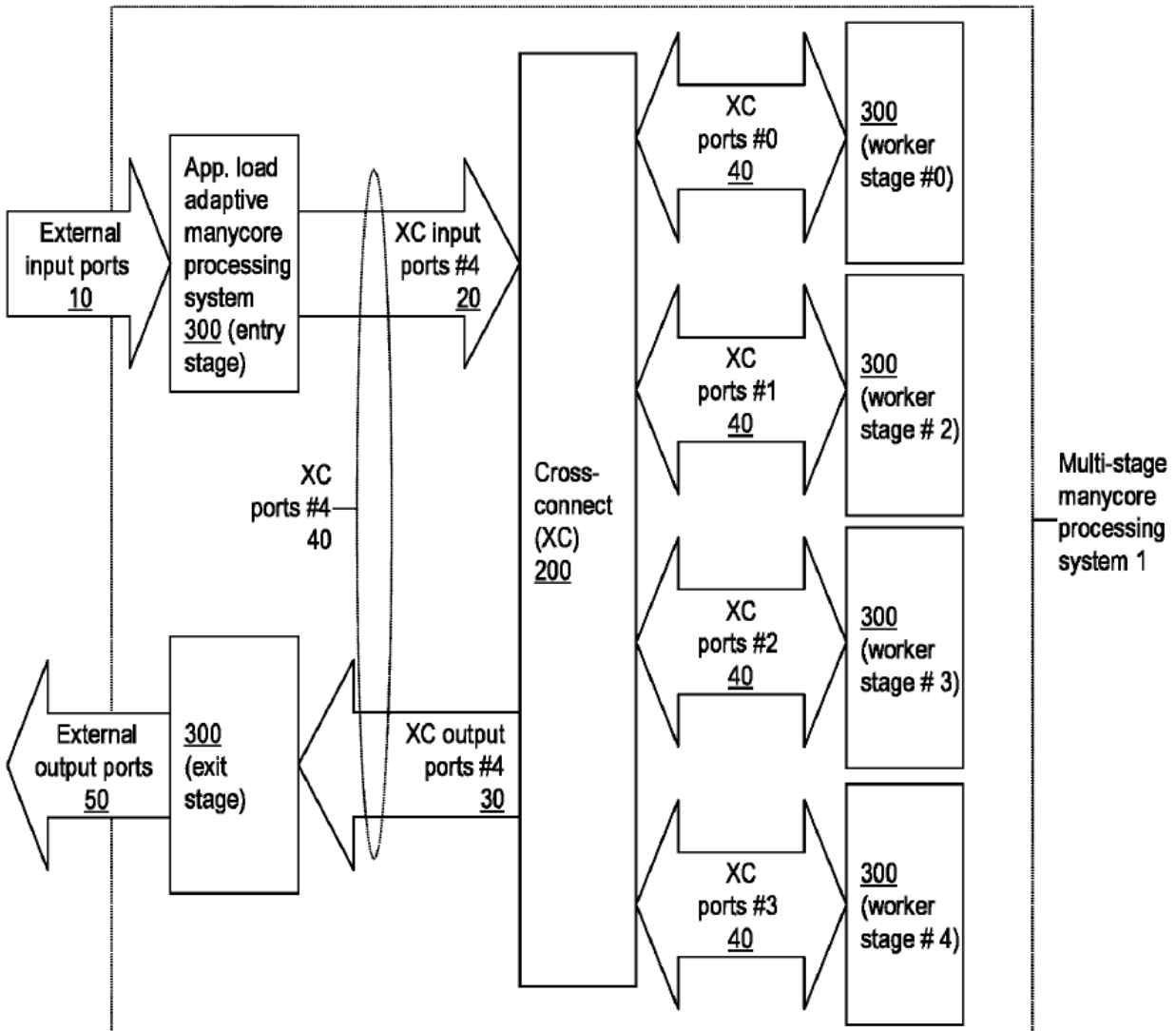
A. Brief Description

The ’948 patent is directed to “techniques for connecting tasks of parallelized programs running on multi-stage manycore processor with each other as well as with external parties with high resource efficiency and high data processing throughput rate.” EX1001, 1:66-2:4.

In its discussion of related art, the ’948 patent identifies two well understood objectives in advancing computing technology: (1) “maximizing the processing speed of one given computationally intensive program running on a dedicated hardware comprising a large number of parallel processing elements” and (2) “most efficiently shar[ing] a given pool of computing hardware resources among a large number of user application programs.” *Id.*, 2:8-18. In doing so, the ’948 patent recognizes within the prior art “a need for a new manycore processor system data flow and IO architecture whose operation, while providing IO data throughout performance, causes little or no overhead in terms of usage of the computation units of the processor.” *Id.*, 3:18-22. The ’948 patent contends however that “[t]here thus exists a need for a new parallel computing architecture, which, at the same time, enables increasing the speed of executing application programs, including through

execution of a given application in parallel across multiple processor cores, as well as improving the utilization of the computing resources available, thereby maximizing the collective application processing throughput for a given cost budget.” *Id.*, 3:46-53.

To allegedly address these issues, the ’948 patent describes the “multi-stage manycore processor system” shown in the functional block diagram of Figure 1 (reproduced below). *Id.*, 6:19-21. The block diagram includes “a set of application processing load adaptive manycore processing stages interconnected by a packet destination app-task-inst controlled cross connect.” *Id.*, 8:44-48.



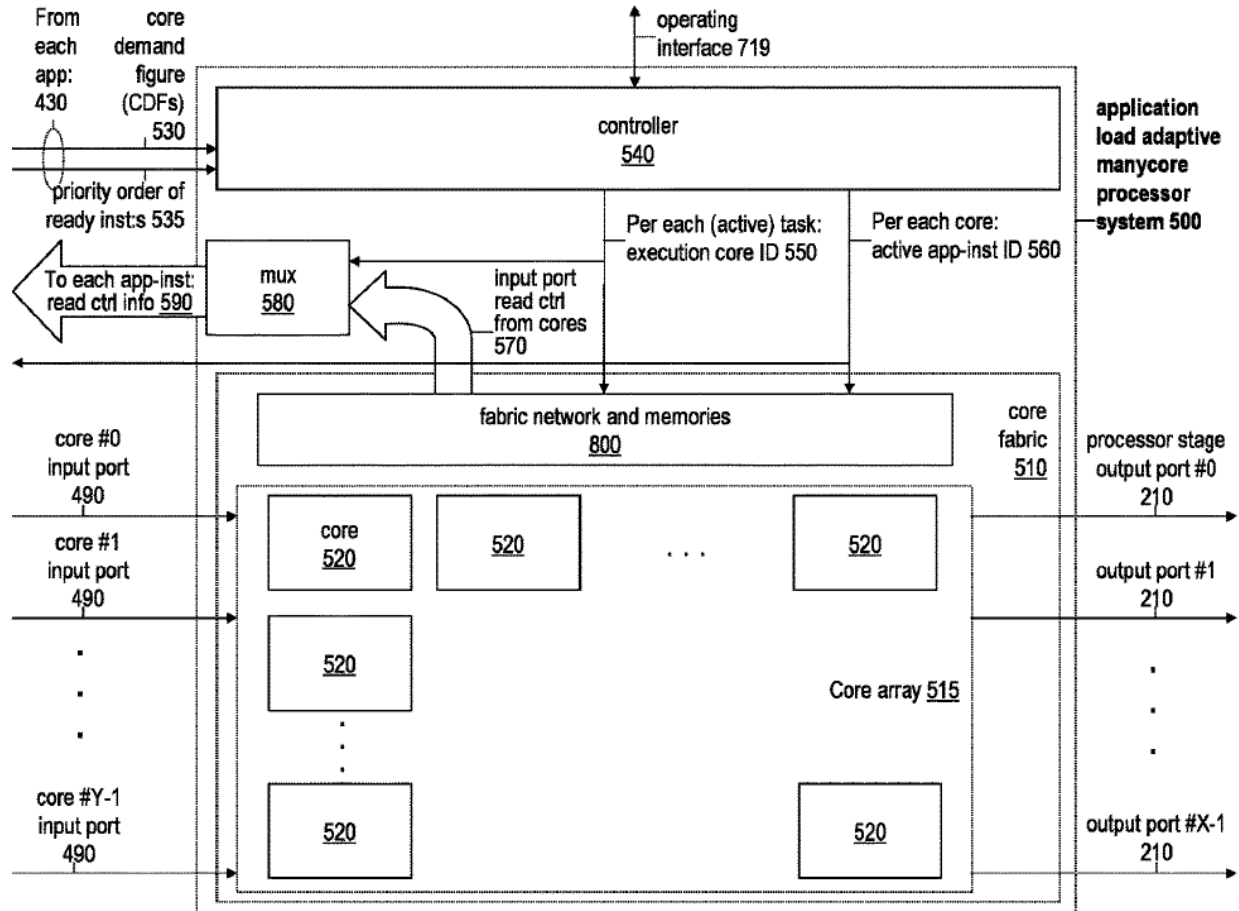
Id., FIG. 1.

In Figure 1, the “system 1 provides data processing services to be used by external parties (e.g. client portions of programs whose server portions run on the system 1) over networks,” “receives data units (e.g. messages, requests, data packets or streams to be processed) from its users through its RX network ports 10, and transmits the processing results to the relevant parties through its TX network ports 50.” *Id.*, 8:55-64. Data can be passed in the system through the cross-connect (XC)

200, which include first-in first out buffers (FIFOs) 260, as illustrated in Figure 2.

Id., 10:45-11:12, FIG. 2.

Figure 5 (reproduced below) shows a functional block diagram for the application load adaptive parallel data processing subsystem 500 for a given manycore processing system 300. *Id.*, 6:33-37. The processor system 500 “comprises an array 515 of processing cores 520, which are dynamically shared among a[sic] the locally hosted tasks of a set of application programs configured to run on the system 1.” *Id.*, 15:12-15. The demand information 430 issued to write “associated applications’ capacity demand indicators 530,” also referred to as “core-demand-figures (CDFs),” to the controller. *Id.*, 15:16-22. The CDFs “express how many cores 520 their associated app-task is presently able to utilize for its ready to execute instances.” *Id.*



Id., FIG. 5.

“A hardware logic based controller module 540 within the processor system 500, through a periodic process, allocates and assigns the cores 520 of the processor 500 among the set of applications 610 (FIG. 6) and their instances, at least in part based on the CDFs 530 of the applications.” *Id.*, 15:33-37.

B. Prosecution History

The '948 patent issued October 19, 2021, from U.S. Patent Application No. 17/212,903 (“the '903 application”), filed March 25, 2021. EX1001, Cover page. The '903 application was filed as a continuation of several patent applications and

claims priority to several provisional applications dating back to November 4, 2011.

EX1002, 410-418.

Substantively, the '948 patent prosecution history involves no prior art rejections. *See, generally, id.* Applicant filed the '903 application with one claim that Applicant canceled in a subsequently-filed amendment, adding new claims 2-16. *Id.*, 393, 402, 503. Applicant also filed a Track One request, which the Office granted. *Id.*, 368-369, 505.

In a non-final Office action, the Office indicated that claims 2-11 were allowable while claims 12-16 were rejected as directed to non-eligible subject matter. *Id.*, 318-322. Following an examiner interview, the applicant amended independent claim 12 to add “one or more processors executing” computer-executable instructions, as discussed in the interview. *Id.*, 298-306.

The Office subsequently issued a Notice of Allowance with no substantive comments concerning the allowance. *Id.*, 279-283. After payment of the issue fee, the applicant filed a petition to withdraw from issuance to cite prior art references, which was granted. *Id.*, 252, 268, 274-278. The Office again issued a Notice of Allowance with no substantive comments concerning the second allowance. *Id.*, 6-7.

As discussed in Section II.B, while the Examiner initialed an IDS that cited Nollet, neither Tuan nor Seo were ever presented to or cited by the Examiner during

prosecution. Thus, the grounds presented herein are entirely new and have never been previously considered by the Office.

C. Level of Ordinary Skill in the Art

A person of ordinary skill in the art in this matter (“POSITA”) would have had at least a combination of a Bachelor’s Degree (or a similar Master’s Degree, or higher degree) in an academic area emphasizing computer science, or a related field, and two or more years of work experience with parallel processing technologies. EX1003, ¶¶20-22. Additional education or industry experience may compensate for a deficit in one of the other aspects of the requirements stated above. *Id.*

IV. CLAIM CONSTRUCTION

All claim terms should be construed according to the *Phillips* standard. *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005); 37 C.F.R. §42.100. Microsoft submits that constructions for claim terms are presently unnecessary to resolve issues of controversy in Ground 1 of the Petition.⁵ *Wellman, Inc. v. Eastman Chem. Co.*, 642 F.3d 1355, 1361 (Fed. Cir. 2011).

⁵ Microsoft reserves the right to assert all available invalidity defenses under 35 U.S.C. § 112 including failing to comply with the written description and enablement requirements and indefiniteness.

V. THE CHALLENGED CLAIMS ARE UNPATENTABLE

A. [GROUND 1] – Nollet in view of Tuan and Seo Renders Claims 1-15 Obvious

1. Overview of Nollet (EX1004)

Nollet discloses a multi-processor computing platform that includes an instruction set processor (ISP) for running “software tasks” and a reconfigurable device (e.g., a Field Programmable Gate Array (FPGA)) for running “hardware tasks.” EX1004, [0057]-[0061]. Such a multi-processor computing platform is said by Nollet to be particularly useful for “multimedia applications, e.g., on portable devices like a personal digital assistant (PDA) or a mobile phone” *Id.*, [0080]; *see also id.*, [0005], [0011], [0107].

Nollet’s multi-processor system relies on an operating systems (OS) that can “provide an environment where several tasks can run concurrently, with minimal of interference between them, but with support for safe data sharing.” *Id.*, [0006]. “The operating system schedules tasks, both on a programmable processor such as an Instruction Set Processor (ISP) and on the reconfigurable logic device and supports a uniform communication framework, which allows tasks to send/receive messages, regardless of their execution location.” *Id.*, [0060]. The ISP can be a standard programmable CPU. *See id.*, [0079]. Thus, the problem addressed by the OS is “how to decide what task to run and where, in case there are multiple processors [] available,” and the OS can base these decisions on “parameters like task priority,

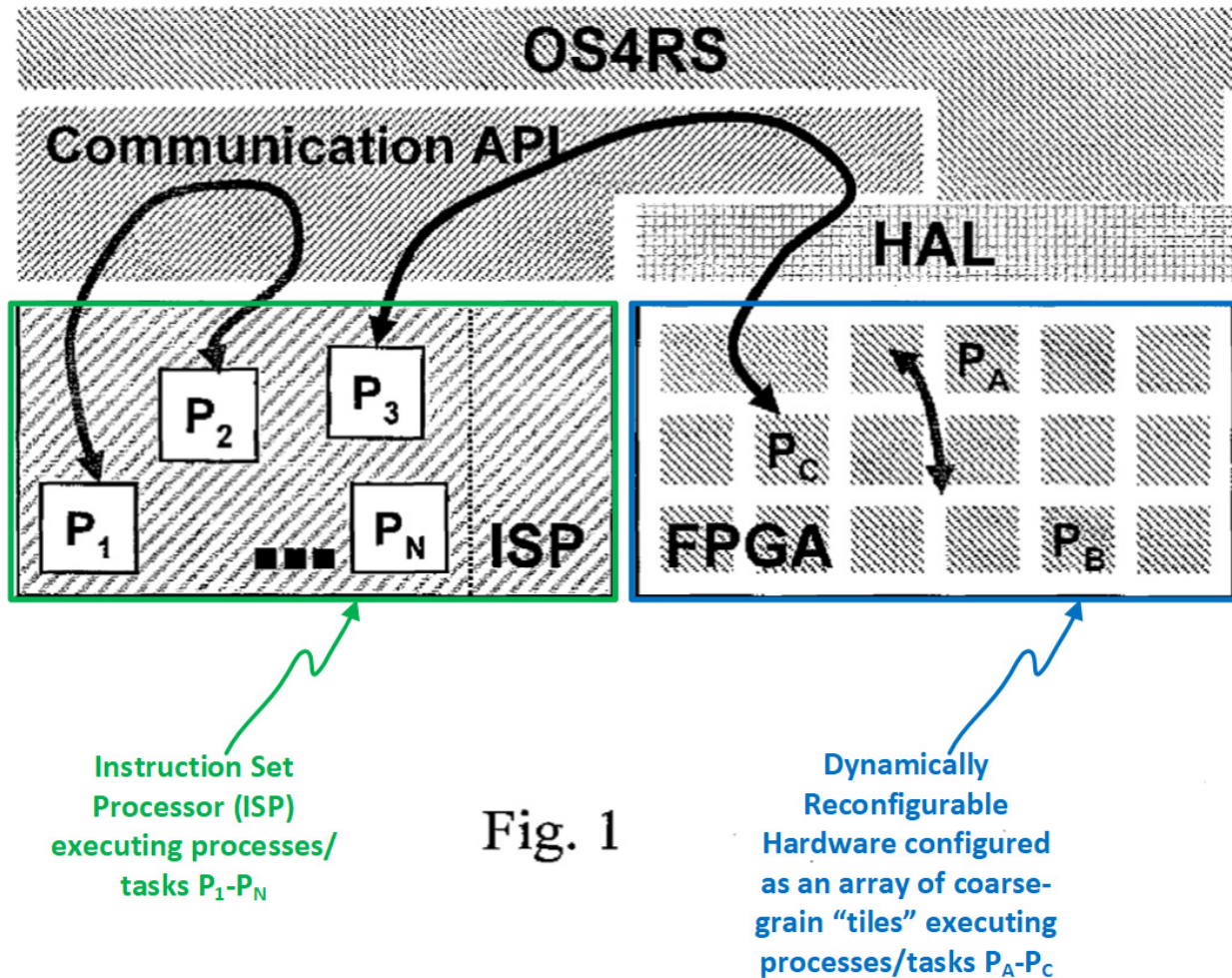
processor affinity, etc.” *Id.*, [0082]; EX1003, ¶50.

Nollet’s multi-tasking environment is “capable of executing at least one application, composed of a plurality of tasks.” *Id.*, [0054]-[0055]. This environment includes a “reconfigurable hardware device, compris[ing] a plurality of small configurable logic blocks and an interconnect structure for interconnecting the configurable logic blocks,” where the hardware device “can be a logic gate array, e.g. a Field Programmable Gate Array (FPGA).” *Id.*, [0057]. When the hardware is reconfigured, the “configuration or function of the logic blocks and/or the interconnection between the configurable logic blocks” are changed. *Id.*, EX1003, ¶51.

The logic blocks are typically arranged in an array and referred to as “tiles” that are “placed edge-to-edge in a tiled arrangement.” *Id.*, [0058]-[0059]; *see also id.*, [0013] (“a reconfigurable matrix (for instance a gate array, especially an FPGA)”); EX1003, ¶52. Each tile is “configurable in a variety of configurations,” and each configuration is “capable of running a hardware component or task” so that the “functionality of the hardware reconfigurable device” can be modified from running a first task in a first configuration to running a “second (different) task” in a “second (different) configuration.” *Id.*, [0059].

An example of Nollet’s architecture is illustrated in FIG. 1, where various tasks/processes of one or more applications are executed simultaneously on each of

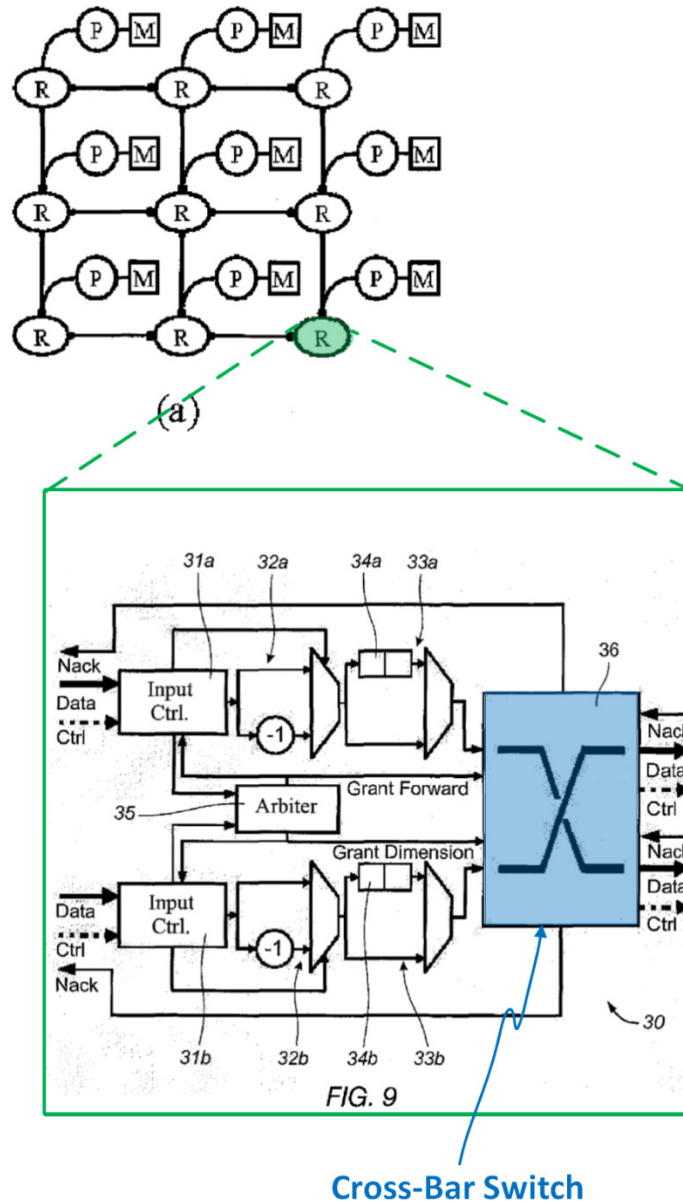
the ISP and the tiles of the FPGA-based reconfigurable hardware. EX1003, ¶53.



Nollet (EX1004), Annotated FIG. 1

Nollet describes that these various tasks are able to communicate with each other. EX1004, [0028]. Tasks are able “to send/receive messages regardless of whether they are executed as hardware or as software tasks.” *Id.* To facilitate the execution of the tasks and the communication between them, each tile may act as a “processor,” and the processors are connected via an interconnection network (ICN). *See id.*, [0035], [0103]-[0104]. The ICN can take “various forms of architectures,

e.g. k-ary n-cubes, hypercubes, butterflies, crossbars and routing policies, e.g. virtual cut-through, wormhole, mad-postman switching.” *Id.*, [0104]. In one example, the network can be composed of 1D-routers, like those shown in FIG. 9, interconnected in a mesh topology, as illustrated in FIGS. 7a and 7b. EX1003, ¶54.



Nollet (EX1004), Annotated FIGS. 7a and 9

Each router may include an “output controller 36 [] composed of a 2-input ***crossbar-switch*** and the NACK.” EX1004, [0114].⁶ Nollet describes that the network can itself “partly comprise logic blocks,” and is thus reconfigurable with the entire device. *Id.*, [0076]. During operation, though, Nollet describes that the interconnection network is “fixed” in order to provide for dynamic partial reconfiguration of the tiles (*see id.*, [0097]-[0100]). EX1003, ¶55. And still, Nollet contemplates a crossbar-switch that is itself “reconfigurable” in the sense that it “provide[s] ***dynamic*** connections between all (source, destination) pairs,” and can be “set on or off ***dynamically*** upon program control.” EX1011, 86-88, EX1003, ¶55.

In an example of task switching and hardware reconfiguration, Nollet describes a scenario where a device user wishes to run two separate applications simultaneously. EX1004, [0107]. The user first starts playing a movie on a movie player, causing a movie decoder to begin performing hardware tasks. *Id.* “When the user is starting his movie decoder the first time, the reconfigurable hardware is free and completely available” so that “the movie player can be started on it, as a hardware task,” and the “computational power of the reconfigurable hardware allows playing the movie at full quality.” *Id.* Later, the user decides to play a 3D game on the device while still playing the movie, but “there are not enough resources

⁶ Emphasis added throughout, unless otherwise noted.

available on the reconfigurable hardware to run both applications.” *Id.* As such, “the movie can be downscaled in resolution and frame rate, resulting in reduced computational requirements for the player,” which “frees up some hardware resources and allows the 3D game to be started as a hardware task.” *Id.* Because these applications are run on reconfigurable hardware, this problem can be solved without requiring different architectures for different multimedia content. *Id.*; EX1003, ¶56.

To manage the different applications running on the hardware/software multitasking platform, Nollet teaches an operating system that manages the various tasks. *Id.*, [0086]. Each application on the platform includes several tasks run on an available hardware/software resource. *Id.* “The operating system preferably keeps track of the available resources, meaning location of all the tasks running. For example, the OS can keep track of which tile of an FPGA is being used to execute a task.” *Id.* In this way, Nollet provides a way of keeping track of where each task is being executed (*i.e.*, which task is being run on which FPGA tiles). *Id.*; EX1003, ¶57.

The system (e.g., via middleware running on top of the OS) is configured to provide Quality of Service (QoS) for the applications running on the system. EX1004, [0080], [0093]. The system is able to reschedule tasks by “determining the best tradeoff in terms of QoS for the different applications that run simultaneously

on the platform.” *Id.*, [0096]. Thus, the system is able to vary the resources allocated to a given application and its tasks to satisfy the QoS levels required by the various applications. *Id.* “If a task requires, for example, for QoS reasons, to be executed in hardware, one of the less critical tasks running on a hardware tile will be reschedule onto an ISP, providing a free resource for the new task.” *Id.*, [0099]. However, in providing the necessary QoS, Nollet teaches that its system balances the computational power of the system with its power consumption. *Id.*, [0086] (“When a new task is started or an old task is deleted, some of the existing tasks might have to be rescheduled onto another resource (e.g. to provide more computational power to the new task or to decrease the power consumption).”); EX1003, ¶58. In this way, Nollet’s system can be “powerful, flexible, *energy conscious* and inexpensive.” EX1004, [0081].

However, Nollet does not constrain the scheduling mechanisms/techniques employed by the OS and middleware to any specific algorithm. EX1003, ¶59. Rather, Nollet leaves such implementation details and choices to the POSITA. *Id.* Tuan and Seo provide examples of the types of task scheduling mechanisms/techniques that a POSITA would have found it obvious to employ when implementing Nollet’s system. *Id.*

2. Overview of Tuan (EX1005)

Like the ’948 patent and Nollet, Tuan is directed to “multitasking on

dynamically reconfigurable processors” and a parallel processing hardware architecture. EX1005, 2, 25-26, 102-103, FIG. 6.1. Like Nollet, Tuan recognizes that “[i]mplementing applications on a reconfigurable fabric allows [sic] to improve performance while minimizing power consumption especially for stream applications.” EX1005, 15; EX1004, [0080]-[0081]. Tuan also recognizes that “reconfigurable platforms are not efficient for control and load/store operations; instead they are often used for accelerating certain parts of an application like data-flow kernels.” EX1005, 29; EX1004, [0080]-[0081]. Thus, like Nollet, Tuan describes coupling the reconfigurable platform with a certain type of microprocessor acting as a general-purpose processor (GPP). *Id.* In other words, the SoC architecture contemplated by Tuan is very similar—if not exactly the same—as that described in Nollet. EX1003, ¶60.

Like the '948 patent, Tuan considers “how to improve the throughput in data flow drive applications by using an appropriate application model and a mapping algorithm to map a target application onto a[n] underlying coarse-grained reconfigurable device consisting of multiple hardware executing units.” EX1005, 2; EX1003, ¶61. It also considers “how to extend the multi-thread execution model in a tile-based architecture into a multicore architecture.” *Id.*

For example, when describing methods for mapping tasks onto tiles and handling task preemption, Tuan uses the architecture of NEC’s DRP-1. *Id.*, 64, 72,

93. The tile-based architecture of the DRP-1 is illustrated in FIG. 6.2, where each reconfigurable tile (i.e., “DRP core”) is connected to a mesh network of routers. This architecture is very similar to that shown in Nollet’s FIG. 7a. EX1003, ¶62. Indeed, like Nollet, the routers in the NEC DRP-1 architecture comprise crossbar switches. EX1005, 103.

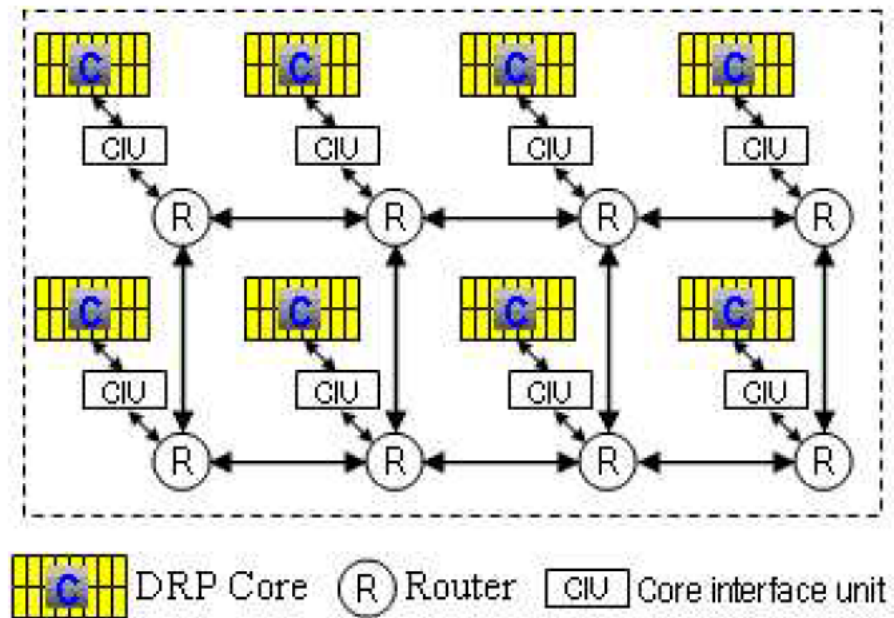


Fig. 6.2: Multicore architecture

Tuan (EX1005), Fig. 6.2

With respect to this type of architecture, Tuan describes that tasks can be mapped to a tile, and further explains that the tasks assigned to the tiles can be switched. *Id.*, 30-33, FIG. 2.9; EX1003, ¶63. In particular, referencing Figure 2.9 (reproduced below), Tuan shows “task-switching” by the reconfigurable array, explaining that when tasks “finish or are preempted, their correspondent cores are

freed” and new tasks “are assigned to these cores.” *Id.*, 33.

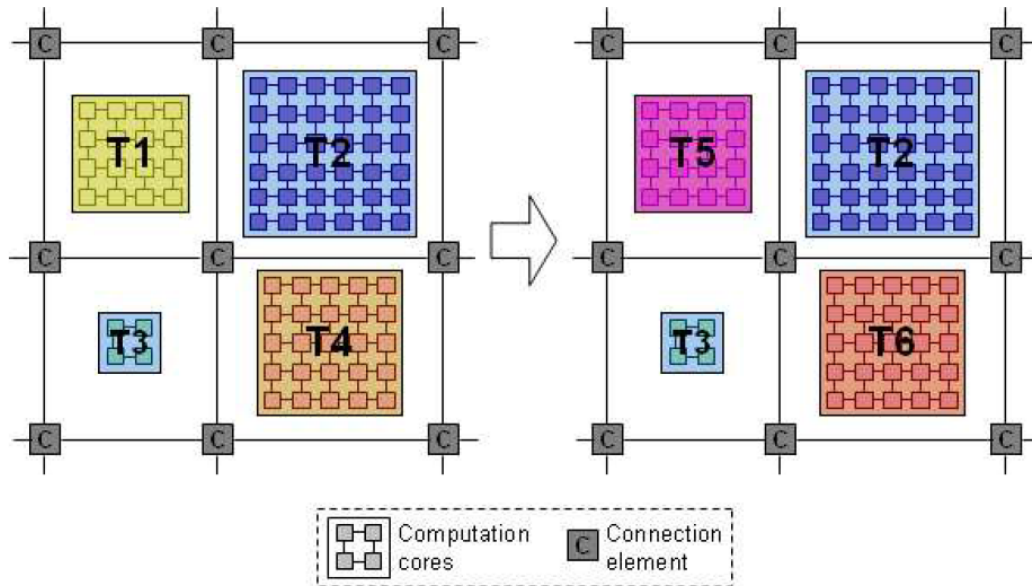


Fig. 2.9: Time-sharing, multicore partitioning model

Tuan (EX1005), FIG. 2.9

Referencing Figure 2.9, Tuan explains:

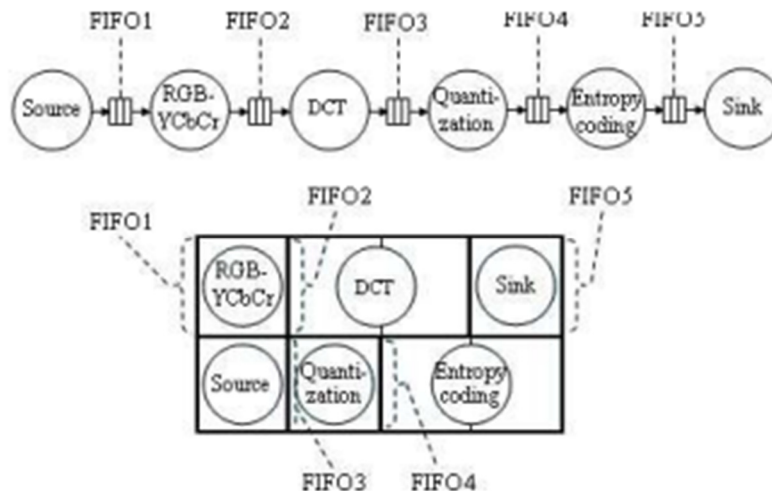
[A]t the beginning, four tasks T1, T2, T3, and T4 are allocated for execution according to their sizes. When tasks T1 and T4 finish or are preempted, their correspondent cores are freed and tasks T5 and T6 are assigned these cores.

Id., 33. Task switching occurs when task T1 is preempted and removed and task T5 occupies the processing unit—that is, task T1 is switched out for task T5. *Id.*; EX1003, ¶64.

To implement its parallel processing platform, Tuan specifies a mapping step where target applications are “mapped onto a candidate architecture for evaluation.” EX1005, 65. In this mapping process, “a total job” of an application “is represented

with multiple processing tasks that can be executed in a pipelined manner.” *Id.*, 66.

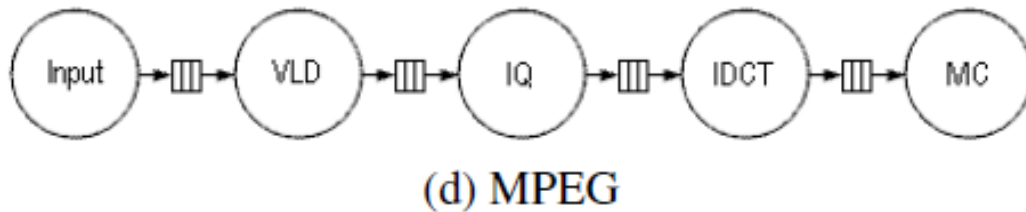
For example, a JPEG encoder is mapped in Figure 4.3 (below) to include multiple processing tasks: a source stage, a sink stage, and multiple intermediate processing stages, including RGB-YCbCr conversion, two-dimensional DCT, quantization, and entropy coding stages. *Id.*, 67, 73-74; EX1003, ¶65. Each task (stage) is executed by a set of reconfigurable logic-based processing units (e.g., tiles), and passes data blocks that are being processed through a set of FIFOs. *Id.*, 73, 75.



EX1005, FIG. 4.3

In another mapping example, Tuan maps an MPEG decoder—a type of video decoder—in Figure 4.9(d) (reproduced below). *Id.*, 74-75, FIG. 4.9. Like the JPEG encoder, the MPEG decoder is mapped to multiple processing tasks: “input, variable length decode (VLD), inverse quantization (IQ), inverse discrete cosine transform (IDCT), and motion compensation (MC),” each of which is separated from a

neighboring task by a FIFO. *Id.*, 75; EX1003, ¶66.



Tuan (EX1005), FIG. 4.9

Tuan teaches that the execution time of a task can be improved by allocating more resources to the task. EX1005, 68, 76. Depending on the amount of parallelism involved in a task, assigning more tiles to perform a given task can lower the tasks execution time, and increase its performance. *Id.* However, at a certain point, allocating more tiles to perform a task no longer improves execution time and may actually degrade performance. *Id.*, 76-77. “Moreover, since using larger numbers of Tiles with low usage will increase the leakage and the clock distribution power, an optimum number of tiles should be carefully selected for each implementation.” *Id.* In other words, like Nollet, Tuan teaches the need to consider power consumption when mapping tasks to tiles. EX1005, 76; EX1004, [0086]. Thus, Tuan suggests that task mapping/scheduling should be a balance between execution performance and power consumption. *Id.*; EX1003, ¶67.

However, Tuan does not limit how the task mapping/scheduling mechanism in the system implements this balance between execution performance and power consumption. EX1003, ¶68. Seo provides one explanation for how to implement

this balance in a multicore processor. *Id.*

3. Overview of Seo (EX1006)

Seo recognized that high-throughput multicore processors were gaining popularity in mobile devices. EX1006, 1. However, as recognized in Nollet and Tuan, Seo also recognized that energy efficiency in mobile devices is an “important issue.” *Id.*; EX1003, ¶69. To account for this problem, Seo describes a real-time task scheduling mechanism for a multicore processor that attempts to optimize power consumption while maintaining the necessary performance to satisfy the demands of the running applications. *Id.*

The algorithm presented in Seo utilizes dynamic voltage scaling (DVS), which “adjusts the clock frequency and operating voltage on the fly to meet changes in the performance demand.” EX1006, 1; EX1003, ¶70. Seo’s algorithm has two parts: (1) dynamic repartitioning, and (2) dynamic core scaling. *Id.*, 2. Dynamic repartitioning “tries to keep the *performance demands of each core* balanced by migrating tasks from the core with the *highest demand* to the one with the *lowest demand*.” *Id.* Dynamic core scaling “decides on *the optimal number of cores for the current performance demand* and tries to meet this criterion as far as all deadlines are guaranteed.” *Id.* “While under high task loads, having more cores is always more efficient; below a certain task load, having more cores actually consumes more energy.” *Id.*, 7. Thus, dynamic core scaling can “save a

considerable amount of leakage power in low load periods.” *Id.*

With respect to the dynamic core scaling portion of the algorithm, Seo describes:

determin[ing] the optimal core number for given [load] L , which is newly updated with each change in the task state. If the optimal number is smaller than the current number of active cores, *try_to_shrink* will be called to reduce the number of active cores. If the optimal number exceeds the current number of active cores, the *Activate* function will be called to increase the number of active cores.

EX1006, 9. Seo’s algorithm determines the optimal core number by considering “the relationship between task set utilization, the number of active cores, and the total power consumption.” *Id.*, 8. Seo calls this the “power consumption expectation function X ,” and it “can be used to determine the number of active cores n that minimizes X for a given [load] L .” *Id.*

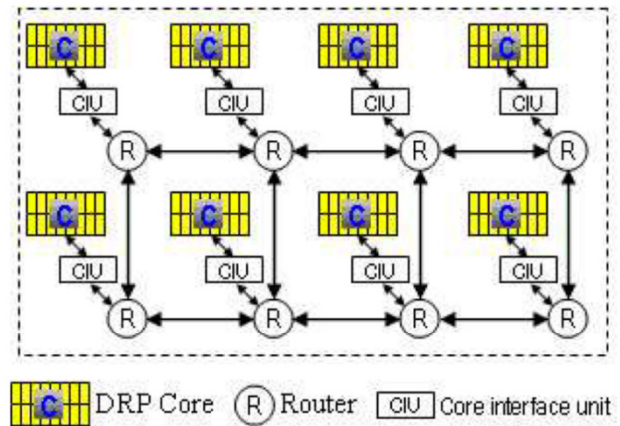
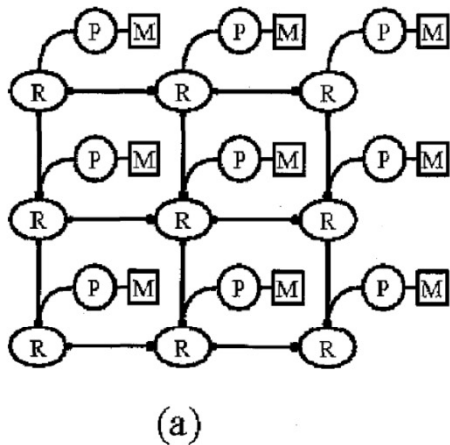
Seo defines the equation for the dynamic utilization L_n (i.e., load) of a given core C_n in equation (9). *Id.*, 5. Load L_n is based on the most recent actual execution time cc_i of all finished tasks and the Worst Case Execution Time (WCET) w_i of all unfinished tasks, where w_i “is defined as the maximum execution time required to complete [task] τ_i at the highest possible processor frequency.” *Id.*, 4-5. Thus, in order to determine the load L_n of a given core C_n , Seo’s algorithm must monitor at least two characteristics for each core: the most recent actual execution time cc_i of

all finished tasks assigned to that core and the Worst Case Execution Time (WCET)

w_i of all unfinished tasks assigned to that core. EX1003, ¶72.

4. The Nollet-Tuan Combination

As described above, Nollet and Tuan both relate to parallel processing on dynamically-reconfigurable hardware, including assigning individually-programmable tiles to tasks of an application. EX1004, Abstract, [0017], [0058], FIG. 1; EX1005, 2, 25-26, 30-32, 73, 102-103, FIG. 6.1; *see* Sections V.A.1-2 *supra*; EX1003, ¶74. Indeed, the hardware architecture leveraged in each has common features, apparent from examination of the following Figures. Compare EX1004, 7a to EX1005, FIG. 6.2.



Nollet (EX1004), FIG. 7(a)

Tuan (EX1005), FIG. 6.2

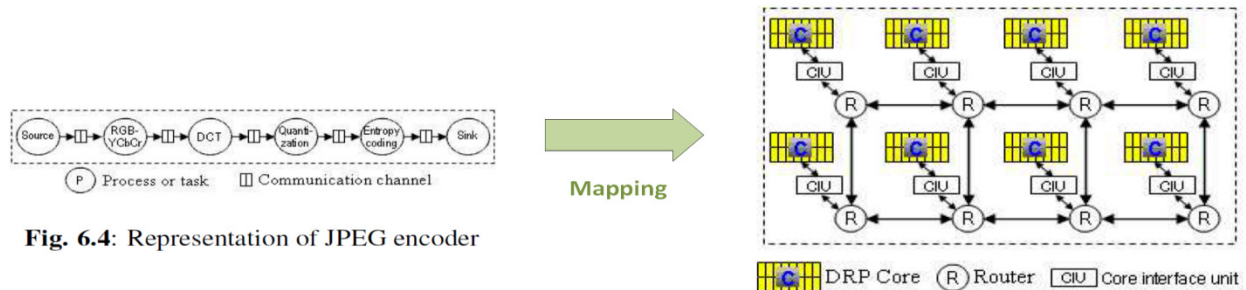
As described in Section V.A.1, *supra*, Nollet does not constrain the scheduling mechanisms/techniques employed by its OS and middleware to any specific

algorithm. EX1003, ¶75. Rather, Nollet leaves such implementation details and choices to the POSITA. *Id.* Tuan and Seo provide examples of the types of task scheduling mechanisms/techniques that a POSITA would have found it obvious to employ when implementing Nollet's system. *Id.*

Nollet describes "allowing a designer to make a representation of an application as communicating threads." EX1004, [0068]. This is accomplished by breaking the application into a plurality of communicating tasks, which can each be mapped on either the reconfigurable matrix/FPGA (denoted hardware tasks) or the ISP (denoted software tasks). EX1004, [0014], [0082], [0086]-[0087], [0102], [0132]-[0133], FIG. 1. And yet, Nollet leaves open to a POSITA how those tasks would actually be mapped onto the reconfigurable matrix and the specific details of that mapping. *See* EX1004, FIG. 2a-2b, [0144]-[0145].

Tuan explicitly teaches representing the processes of an application as multiple stages or tasks and provides example implementation details about how to map those tasks onto a reconfigurable matrix. *See generally* EX1005, 66-79, FIGS. 4.3, 4.9; EX1003, ¶77. A POSITA would have found it obvious to implement the reconfigurable multitasking computing platform of Nollet with the task mapping architecture, network manager, and memory of Tuan. EX1003, ¶77. A system based on the teachings of both Nollet and Tuan would thus facilitate mapping tiles to tasks of an application by implementing the tasks of an application in a pipelined manner,

which “makes it easier to partition an application into a set of parallel communicating tasks,” which is “suitable to be implemented on reconfigurable arrays,” such as Nollet’s reconfigurable device. EX1005, 66. The combined system would further include FIFOs, network manager, and memory to facilitate Nollet’s teaching of inter-task communication and to help manage “information about where tasks are located” after tasks are mapped to tiles or cores. EX1004, [0129]-[0133], FIG. 1; EX1005, 15-16, 27, 28-29, 66, 72-73, 75, 101-111. A POSITA would have been motivated to incorporate the mapping approach taught by Tuan into Nollet “to improve throughput and reduce fragmentation” and to “support[] inter-task communication and input/output operations.” EX1003, ¶78; EX1005, 28-29. Indeed, Tuan provides specific implementation details of how to map tasks communicating via switches with FIFO buffers onto a tile-based reconfigurable hardware system utilizing a switch/router-based interconnection network, like the reconfigurable hardware architecture shown in FIG. 7 of Nollet. *See* EX1005, 103-111.



Tuan (EX1005), Annotated FIGS. 6.2 and 6.4

Tuan further explains that its reconfigurable logic-based processing units are “*often used as an accelerator* for a host processor.” *Id.*, 38 (“A system consisting of a microprocessor and a DRPA could achieve both the high performance with hardware-accelerated execution on the reconfigurable array and the flexibility with software execution on the GPP.”), 29 (“reconfigurable platforms are not efficient for control and load/store operations; instead, they are *often used for accelerating* certain parts of an application like data-flow kernels”). A POSITA would have thus recognized the benefits of using the reconfigurable logic-based processing units of Nollet as an accelerator to the execution of various programs such as a movie player or a 3D game. EX1003, ¶79.

Combining the teachings of Nollet with Tuan would have merely involved combining prior art elements according to known methods to yield predictable results. EX1003, ¶80; *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 416 (2007). Nollet and Tuan are directed to similar systems that assign reconfigurable tiles/cores of a hardware device to tasks of an application. *Id.* A POSITA would have been motivated to modify Nollet’s platform with Tuan’s explicit inclusion of FIFOs, a network manager, and memory to facilitate communications between hardware tasks, keep track of the locations of tiles where each application task is assigned, and managing distribution of tasks to hardware resources. EX1003, ¶80; EX1005, 15-16, 27-29, 66, 72-73, 75, 101-111. The use of these components in Tuan would align

with Nollet's teaching or suggestion that its platform supports each of these functions. EX1003, ¶80; EX1004, [0081] (discussing "manag[ing] the applications by distributing the different tasks on the available resources"), [0086] (discussing "keep[ing] track of which tile of an FPGA is being used to execute a task"), FIG. 1 (showing communication paths between tasks, suggesting multi-stage programs). A POSITA would have understood Nollet's teaching of communication paths between tasks as suggesting the inclusion of multi-stage programs and naturally looked to related references such as Tuan that describe how programs can be partitioned into multiple "tasks that can be executed in a pipelined manner," which "allows for parallelism and pipelined processing, which are suitable to be implemented on reconfigurable arrays," such as those taught in Nollet. EX1003, ¶80; EX1005, 66. Moreover, a POSITA would have understood Nollet's discussion of keeping track of which FGPA tile is being used to execute a task as requiring the use of a system for managing this information and thus naturally looked to related references such as Tuan that describe how "information about where tasks are located" for execution by FPGA tiles is stored and updated using switches for inter-task communication, a network manager, and memory. EX1003, ¶80; EX1005, 15-16, 27-29, 66, 72-73, 75, 101-111.

5. The Nollet-Tuan-Seo Combination

As describes in Sections V.A.1-2, *supra*, each of Nollet and Tuan teach the

need to consider power consumption when mapping tasks to tiles. EX1004, [0086]; EX1005, 76. Together and alone, they suggest that task mapping/scheduling should balance between execution performance and power consumption. *Id.*; EX1003, ¶81. Seo explains how this balance might be implemented in a multicore processor. EX1003, ¶81. Specifically, Seo describes a two-part algorithm for scheduling tasks on a multicore processor, and the algorithm reduces power consumption while maintaining the required deadlines of the tasks. EX1006, 1-2; EX1003, ¶81.

A POSITA would have found it obvious to utilize Seo's approach to dynamic repartitioning and core scaling in the context of the Nollet-Tuan multicore reconfigurable hardware matrix architecture in order to "optimize power consumption during execution" and "reduce leakage power consumption under low load conditions." EX1006, 1-2; EX1003, ¶82. Utilizing the concepts of Seo's tasking scheduling algorithm would have been consistent with Nollet's teaching of "manag[ing] the available hardware resources in a consistent, efficient and fair way." EX1004, [0006].

According to Seo, dynamic task scheduling algorithm can be useful for multicore processors that utilize dynamic voltage scaling (DVS). EX1006, 1. While Seo does not specifically mention that the multicore processor can be implemented using reconfigurable hardware, like the tiled reconfigurable fabric described in Nollet and Tuan, POSITAs would have known that dynamic task scheduling

algorithms that utilize DVS and are able to activate/deactivate tiles/cores were applicable to reconfigurable fabric. *See, e.g.*, EX1007, 73 (teaching that “it is possible to save power in situations of a low workload: the FPGAs that are not needed can be turned off”); EX1003, ¶83. Indeed, a POSITA would have known that the type of reconfigurable fabric described in Nollet would have employed DVS. *See, e.g.*, EX1009, 1 (describing that Virtex-4 Xilinx FPGAs include dynamically reconfigurable clock synthesis technology); EX1004, [0098] (describing the use of Xilinx Virtex™ series hardware for Nollet’s FPGA); EX1003, ¶83.

6. Predictable Results

The combination of Nollet, Tuan, and Seo would have been predictable and foreseeable because (1) Nollet and Tuan both teach parallel processing on dynamically-reconfigurable hardware arrays by assigning individually-reconfigurable tiles to tasks of an application, (2) Nollet, Tuan, and Seo all seek to schedule tasks in a way that considers “efficiency” and “power consumption,” and (3) the combination merely involves incorporating known features that are explicitly disclosed in Tuan and/or Seo and would have been understood by a POSITA as likely present but not explicitly disclosed by Nollet. EX1003, ¶84; *see, e.g.*, EX1004, [0087], [0159], FIG.1. The combination is predictable, at least in part, because elements of the combined system would perform similar functions that had been known to perform prior to the combination. EX1003, ¶84. For example, the computing platform of Nollet

would still perform the same or similar functions, including assigning tasks of applications to FPGA tiles, as it did prior to the combination with Tuan and Seo. *Id.* Applying and combining these concepts would have been obvious from a POSITA's perspective, and a POSITA would have had a reasonable expectation of success in doing so. *Id.*

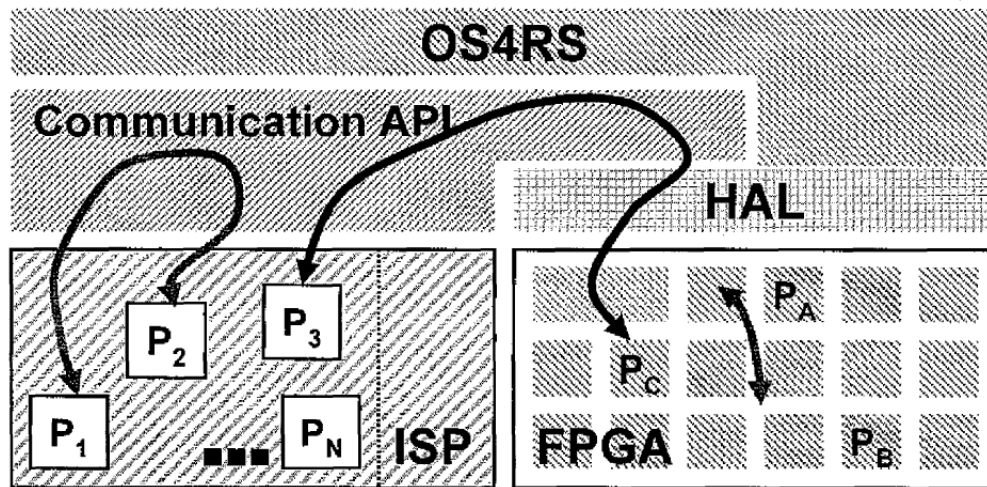
The combined teachings of Nollet, Tuan, and Seo would have rendered obvious claims 1-15 of the '948 patent, and the combination discloses each limitation of the claims as described below. *Id.*

7. Claim 1

[1.0] A method on a parallel data processing platform comprising a plurality of programmable logic-based processing units for accelerating a service comprising at least a first program and a second program, different than the first program, the method comprising:

To the extent the preamble is limiting, it is rendered obvious by the Nollet-Tuan-Seo combination. EX1003, ¶¶87-90. Nollet is directed to “multitasking computing systems and methods adapted for reconfigurable computing platforms as well as reconfigurable hardware devices and software, such as operating systems, middleware, or drivers adapted for supporting and executing the methods.” EX1004, [0003]. Nollet indicates that its parallel data processing platform comprises a “reconfigurable matrix [that] can be subdivided into smaller processing units or ‘tiles.’” *Id.*, [0017]. Nollet explains, “[p]referably, the reconfigurable device, comprises a plurality of tiles, each tile being configurable in a variety of

configurations, each of the tile configurations being capable of running a hardware task.” *Id.*, [0020]. Figure 1 (reproduced below) shows an exemplary hardware architecture comprising an FPGA array having a plurality of reconfigurable tiles. *Id.*, [0037], FIG. 1. Because each tile is “configurable in a variety of configurations” and is based on a Field *Programmable Gate* Array (FPGA), a POSITA would have understood that each tile is a programmable logic-based processing unit, because a gate is a type of logic. *See* EX1004, [0081] (“The provided platform is thus a heterogeneous multi-processor platform, containing one or more instruction set processors (ISP) and one or more *reconfigurable* matrices (for instance a *logic gate array*, especially an FPGA).”); EX1003, ¶¶87. Nollet also discloses a first program (3D game) and a second program (movie player). EX1004, [0107].



Id., FIG. 1.

A POSITA would have understood that the FPGA array of Nollet would act to accelerate a service (e.g., programs running on the device) as FPGAs are well

known to do. EX1003, ¶88 (citing EX1013, 2 (“In the UNNS approach, each node has a single type of PE such that the installation and management of hardware *accelerators (e.g. FPGA and GPU)* are simpler in a segment of the cluster.”)). Indeed, Nollet teaches that the “introduction of the reconfigurable hardware device provides low power consumption for *high computation power by its hardware characteristics* on the one hand, and flexibility by its reconfiguration aspects on the other hand.” EX1004, [0081].

For example, Nollet discusses an MP3 player and a movie player. EX1004, [0080]. The MP3 player would necessarily include an MP3 decoder, and the movie player would similarly include a multimedia decoder (e.g., movie decoder). *Id.*, [0107]; *see also* EX1005, 15, 43, 54, 74 (Table 4.2), 95 (Table 5.1); EX1003, ¶89. The FPGA array of Nollet would be used to accelerate the particular decoder (e.g., MP3 decoder or movie decoder) of the respective program. EX1004, [0107] (“The computational power of the reconfigurable hardware allows playing the movie at full quality.”). Moreover, the combination of Nollet and Tuan teaches that using the FPGA array to accelerate the programs on the device would have been obvious. *Id.*; EX1003, ¶89.

Like Nollet, Tuan discloses a parallel processing hardware architecture. EX1003, ¶90; EX1005, 25-26, 101-111, FIG. 6.2. In particular, Figure 6.2 of Tuan discloses a “computational cores for a multicore architecture” where “tasks are

dynamically assigned into and removed from cores” and “the exact place of a certain task . . . is often changed from time to time either when the system is defragmented or when the task is removed and later resumed.” EX1005, 101-111, FIG. 6.2. Tuan further explains that its reconfigurable logic-based processing units are “often used as an accelerator for a host processor.” *Id.*, 38 (“A system consisting of a microprocessor and a DRPA could achieve both the high performance with **hardware-accelerated execution** on the reconfigurable array and the flexibility with software execution on the GPP.”), 29 (“reconfigurable platforms are not efficient for control and load/store operations; instead, they are often used for **accelerating certain parts of an application** like data-flow kernels”).

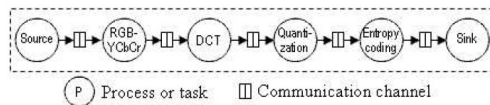


Fig. 6.4: Representation of JPEG encoder

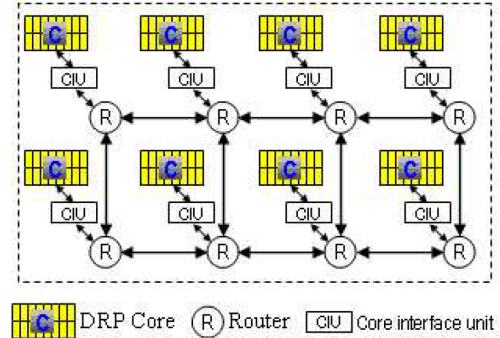


Fig. 6.2: Multicore architecture

Tuan (EX1005), Annotated FIGS. 6.2 and 6.4

[1.1] forming a first set of inter-task communication paths connecting a first set of programmable logic-based processing units of the plurality of programmable logic-based processing units into a first multi-stage program instance,

The Nollet-Tuan-Seo combination discloses this element. EX1003, ¶¶91-95.

Nollet provides “a method for execution of **an application** or a plurality of

applications, the execution requiring running substantially simultaneously a ***plurality of tasks*** (multi-tasking), on the platform.” EX1004, [0017]. This method provides for dynamic instantiation of hardware tasks “by exploiting partial reconfiguration” through subdividing the reconfigurable device into ***tiles that are individually programmable***. *Id.* That is, Nollet’s platform allows individual logic-based tiles of a hardware device to be programmed to perform different hardware tasks of one or more applications. *Id.*, [0017], FIG. 1 (showing different tiles executing different hardware tasks P_A, P_B, and P_C on an FPGA); EX1003, ¶91.

In an example scenario, Nollet discloses programming a first set of FPGA tiles to execute a first multi-stage program instance (e.g., a multimedia decoder) when a user starts watching a movie. *Id.*, [0107]. Nollet also discloses forming a first set of inter-task communication paths connecting a first set of programmable logic-based processing units (e.g., FPGA tiles/cores) to change the configuration or function of the logic units. *Id.*, [0057]. Nollet further explains that “[m]essages 5 between tasks, both scheduled in hardware 8, are routed inside the interconnection network,” where example communication paths between tasks are shown in Figure 1. *Id.*, [0132], FIG. 1. A POSITA would have understood that because the hardware tasks (e.g., the constituent tasks of a multimedia decoder) communicate, the tasks are likely part of a multi-stage program instance. EX1003, ¶92. Specifically, Nollet discloses an example video decoder where “a send thread 100 passes the coded

frames one by one to the decoder thread 102, 106,” which “decodes the frames and sends them, one macroblock at a time, to a receive thread 104 that reconstructs the images and displays them.” EX1004, [0148]-[0149]. As such, a POSITA would have understood that the multi-stage nature of the program instances in Nollet. *Id.*; EX1004, [0148]-[0149]; EX1003, ¶92.

Tuan also discloses the use of a multi-stage program instance in the context of a reconfigurable multicore processor. EX1003, ¶93. Preliminarily, Tuan discloses forming a first set of inter-task communication paths that connect a first set of programmable tiles into a multi-stage program instance by reconfiguring the communication paths. EX1005, 66, 101-111, FIGS. 4.3, 4.9, 6.2, 6.4, 6.6; EX1003, ¶93. For example, Tuan shows that a JPEG encoder in Figure 6.4, which includes a source stage, a sink stage, and multiple intermediate processing stages, including RGB-YCbCr conversion, two-dimensional DCT, quantization, and entropy coding stages. *Id.*, 67, 73-74, 104-105. Each of these stages is executed by a set of programmable logic-based processing units (e.g., cores). *Id.*, 105 (“With the multicore architecture, tasks are mapped onto cores . . . , [and] all tasks belong[ing] to a target application are arranged for executing in a pipelined manner.”); EX1003, ¶93.

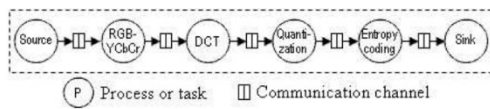


Fig. 6.4: Representation of JPEG encoder



Mapping

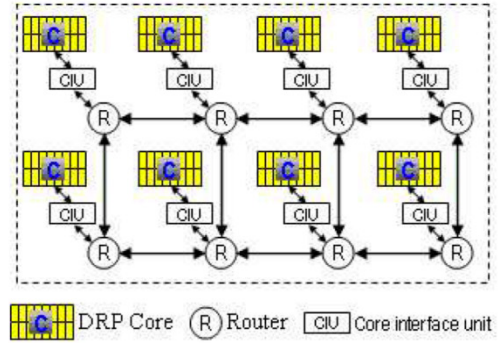
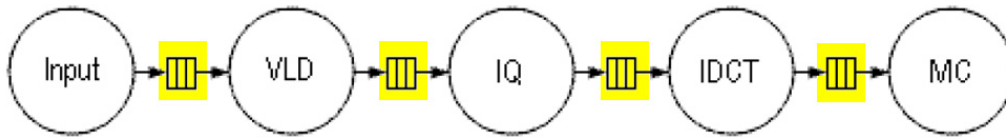


Fig. 6.2: Multicore architecture

Tuan (EX1005), Annotated FIGS. 6.2 and 6.4

In another example of mapping programmable tiles to applications—like JPEG encoding, Tuan discusses movie decoders. *Id.*, 74-75. In particular, Figure 4.9(d) of Tuan (reproduced below) shows an example of a movie decoder—such as the movie decoder of Nolle—*in the form of an MPEG decoder*. *Id.*, 74-75, FIG. 4.9. Like the JPEG encoder, the MPEG decoder of Tuan is a multi-stage program instance that includes multiple computation stages including “input, variable length decode (VLD), inverse quantization (IQ), inverse discrete cosine transform (IDCT), and motion compensation (MC).” *Id.*, 75; EX1003, ¶94. Thus, a POSITA would have understood that Nolle’s movie decoder is a multi-stage program instance, at least based on Tuan’s more detailed teachings involving the multiple tasks of movie decoders. EX1003, ¶94.



(d) MPEG

Tuan (EX1005), FIG. 4.9 (highlighted)

The Nollet-Tuan-Seo combination further discloses that the linking of the processing units is performed through a first set of inter-task communication paths. EX1005, 28-29, 32-33, 35-36, 103-105, FIGS. 4.9, 6.2, 6.4. In particular, the MPEG decoder includes a number of stages, and though not explicitly labeled in Figure 4.9(d), each of the stages of the MPEG decoder is connected to an adjacent stage via a FIFO (highlighted above), through which communication is performed. *Id.*, 66-67, 73, 75, FIG. 4.9(d); EX1003, ¶95. More details of the inter-task communication paths connecting the programmable logic-based processing units is further provided with respect to [1.3], *infra*, the description of which is also incorporated here with respect to [1.1].

[1.2] wherein the first multi-stage program instance is configured to accelerate the first program, and

The Nollet-Tuan-Seo combination discloses this element. EX1003, ¶¶96-98. The '948 patent does not describe a program instance that is configured to accelerate a program. *See, generally*, EX1001; EX1003, ¶96. To the extent that it discusses accelerating a program, the '948 patent states the following in its description of

related art:

Thus, in effect, one branch of computing technology advancement effort has been seeking to **effectively use a large number of parallel processors to accelerate execution of a single application program**, while another branch of the effort has been seeking to efficiently share a single pool of computing capacity among a large number of user applications to improve the utilization of the computing resources.

Id., 2:18-25.

There thus exists a need for a new parallel computing architecture, which, at the same time, **enables increasing the speed of executing application programs, including through execution of a given application in parallel across multiple processor cores**, as well as improving the utilization of the computing resources available, thereby maximizing the collective application processing throughput for a given cost budget.

Id., 2:46-53.

The Nollet-Tuan-Seo combination discloses at least this acceleration of execution of an application program by parallel processors. In particular, as discussed above for element [1.0], a POSITA would have understood that the FPGA array of Nollet would act to accelerate a service (e.g., programs running on the device) as FPGAs are well known to do. EX1003, ¶97 (citing EX1013, 2). In fact, Tuan teaches, “As being mentioned in many research, reconfigurable platforms are not efficient for control and load/store operations; instead, they are often used for

accelerating certain parts of an application, like data-flow kernels.” EX1005, 29

. Moreover, the combination of Nollet and Tuan teaches that using the FPGA array to accelerate the programs on the device would have been obvious. *Id.*; EX1005, 38 (“A system consisting of a microprocessor and a DRPA could achieve both the high performance with hardware-accelerated execution on the reconfigurable array and the flexibility with software execution on the GPP.”), 29 (“reconfigurable platforms are not efficient for control and load/store operations; instead, they are often used for accelerating certain parts of an application like data-flow kernels”).

To the extent that Element [1.2] is interpreted to mean that the first multi-stage program instance itself accelerates the first program, the combination of Nollet and Tuan renders obvious that the first multi-stage program instance accelerates execution of the first program. EX1003, ¶98. One reason a POSITA would have found this obvious is because, as described with respect to [1.1], utilizing a multi-stage program instance breaks a program or portion of a program into a set of discrete tasks, some of which can be executed in parallel. *Id.*

[1.3] wherein the first set of programmable logic-based processing units are interconnected using reconfigurable cross-connects;

The Nollet-Tuan-Seo combination discloses this element. EX1003, ¶¶99-104. With respect to this element, the '948 patent states that its system includes “a hardware implemented data packet switching cross-connect (XC) connecting data packets from an output port of a processing stage to an input port of a given

processing stage if a destination software program task of the data packet is hosted at the given processing stage.” EX1001, 3:40-59.

Nollet discloses such a reconfigurable cross-connect for interconnecting programmable logic-based processing units. EX1005, [0110]-[0112]; EX1003, ¶100. Specifically, Nollet discloses a “reconfigurable hardware device, compris[ing] typically a plurality of small configurable logic blocks and *an interconnect structure for interconnecting the configurable logic blocks*,” and states that “[r]econfiguring the hardware device means changing the configuration or function of the logic blocks and/or the *interconnection between the configurable logic blocks*.” EX1005, [0057]. Nollet also describes a specific implementation of its reconfigurable multitasking computing platform that includes FPGAs as the reconfigurable hardware device “having a packet-switched *interconnection network*” provided by “a 2D torus using wormhole routing,” for example. *Id.*, [0110]; *see also id.*, [0034] (“The communication network can be adapted for supporting packet-switched interconnection network, in particular for supporting message passing communication between tasks, executed by the tiles. ... The communication network may comprise a plurality of routers. Interfaces, for decoupling the tasks from the network may be provided.”), [0035] (“A routing structure for interconnecting the configurable logic blocks (CLB’s) may be provided. A communication network defined on the routing structure for providing

communication between the tiles can also be provided. The communication network can comprise a plurality of routers.”).

Figure 6 of Nollet further illustrates such a reconfigurable cross-connect. EX1003, ¶101. Nollet explains that “[o]n a 2D torus, rows are equivalent to columns, therefore the torus can be decomposed into rows of 1D toruses connected to columns of 1D toruses as shown in FIG. 6.” EX1005, [0113]. “Two simple 1D routers 28 can be chained to obtain a 2D router and produce a more modular network.” *Id.*

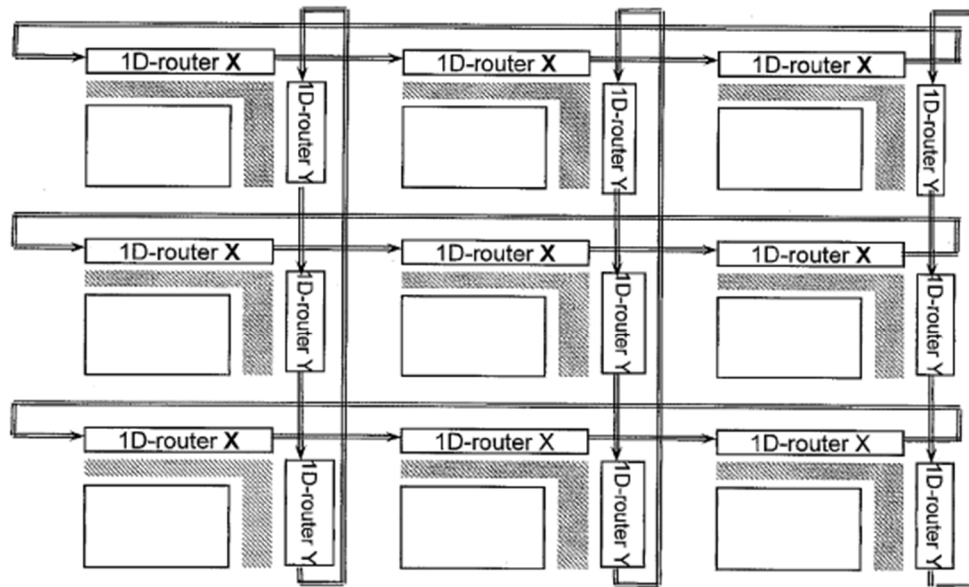


Fig. 6

Nollet (EX1004), FIG. 6

Tuan describes mapping a multi-stage program onto a multicore processing architecture utilizing switches to communicate between cores/tasks—an architecture that operates in the same manner as that disclosed in Nollet. EX1005, 99-105, FIGS.

6.2, 6.4; EX1004, [0035], [0103]-[0104], [110]-[114]; EX1003, ¶102. Thus, a POSITA would have understood that the stages of the MPEG decoder are connected via the first set of inter-task communication paths (i.e., the 2D switch-based interconnection network described in each of Nollet and Tuan) in view of Tuan's description with respect to FIGS. 6.2 and 6.4. EX1003, ¶102; EX1004, FIGS. 7a-7b (showing communication paths between processors/cores/tiles).

Nollet describes that each router may include an "output controller 36 [] composed of a 2-input *crossbar-switch* and the NACK." EX1004, [0114]. Tuan similarly describes its router architecture as relying upon a crossbar switch. EX1005, 103-104, FIG. 6.3. As noted above, Nollet describes that the network can itself "partly comprise logic blocks," and is thus reconfigurable with the entire device. *Id.*, [0076]. This is consistent with Tuan, which describes how the same NEC DRP-1 chip is able to be reconfigured to operate as either a tile-based architecture or a multicore architecture that relies upon a "fixed" interconnection network based on a 2D mesh network of switches. EX1005, 101-104; EX1003, ¶103.

During operation, Nollet describes that the interconnection network is "fixed" in order to provide for dynamic partial reconfiguration of the tiles (*see id.*, [0097]-[0100]). EX1003, ¶104. However, a crossbar-switch is itself "reconfigurable" in the sense that it "provide[s] *dynamic* connections between all

(source, destination) pairs,” and can be “set on or off *dynamically* upon program control.” EX1011, 86-88, EX1003, ¶104. In other words, one or more crossbar switches are addressable to effectively provide a reconfigurable connection between cores/tiles. EX1003, ¶104. This type of “reconfigurable” crossbar switch-based network operates in a very similar manner to cross-connect (XC 830) described in the ’948 Patent, which shows the XC 830 relying upon addressable multiplexors 910. *See* EX1001, 18:20-27, FIG. 9; EX1003, ¶104.

[1.4] forming a second set of inter-task communication paths connecting a second set of programmable logic-based processing units of the plurality of programmable logic-based processing units to a second multi-stage program instance,

The Nollet-Tuan-Seo combination discloses this element. EX1003, ¶¶105-107. Nollet discloses connecting a second set of programmable logic-based processing units (e.g., FPGA tiles) to a second multi-stage program instance of a second program (3D game). EX1004, [0017], [0107]; *see supra* discussion of element [1.1]. Specifically, Nollet teaches “dynamically reconfiguring a computing platform in a process for executing *at least one application* on the platform.” EX1004, [0019]. For example, Nollet discloses reconfiguring one or more processing units to run 3D gaming tasks of a 3D game. *Id.*, [0105], [0107] (“This frees up some hardware resources and allows the 3D game to be started as a hardware task.”).

Tuan discloses a number of different applications that are partitioned and

modeled as multi-stage program instances. EX1005, 72-75. For example, Tuan discloses a Viterbi decoder, a JPEG encoder, an MPEG decoder, a Turbo encoder, a G721 encoder, and a G721 decoder. *Id.*, 73-75. It would have been obvious to a POSITA to partition and model the 3D gaming tasks of Nollet as multi-stage program instances on dynamically reconfigurable processors, as taught in Tuan, at least because such an approach “enables efficient sharing of resources due to adequate task scheduling, allows dynamic changes of the application due to introduction of new tasks during run-time and greatly simplifies development since tasks can be developed quite independent from each other.” *Id.*, 16; EX1003, ¶106.

The Nollet-Tuan-Seo combination further discloses that the linking of the processing units is performed through forming a second set of inter-task communication paths that connect the processing units. EX1005, 66-67, 73-75, FIGS. 4.3 and 4.9. In particular, the 3D gaming tasks of the Nollet-Tuan-Seo combination includes a number of stages, where each of the stages is connected to an adjacent stage via a communication pathway through which communication would be performed. *Id.*, 66-67, 73, 75, 101-110, FIGS. 4.9(d), 6.4; EX1003, ¶107. Tuan further describes mapping these multi-stage programs onto the multicore architecture described with respect to FIG. 6.2 such that each stage is executed by a set of programmable logic-based processing units (e.g., cores). *Id.*, 105 (“With the multicore architecture, tasks are mapped onto cores . . . , [and] all tasks belong[ing]

to a target application are arranged for executing in a pipelined manner.”); EX1003,

¶107. A POSITA would have understood that the stages of the 3D gaming tasks would be connected via a second set of inter-task communication paths (i.e., the 2D switch-based interconnection network described in each of Nollet and Tuan) in view of Tuan’s description with respect to FIGS. 6.2 and 6.4. EX1003, ¶107.

For at least the reasons discussed above in Section V.A.1-6, a POSITA would have been motivated combine Nollet and Tuan.

[1.5] wherein the second multi-stage program instance is configured to accelerate the second program, and

For the same reasons as discussed above for element [1.2], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶108.

[1.6] wherein the second set of programmable logic-based processing units are interconnected using reconfigurable cross-connects;

For the same reasons as discussed above for element [1.3], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶109.

[1.7] in response to a first increased demand for the first multi-stage program instance determined by monitoring a first characteristic corresponding to the first set of programmable logic-based processing units, based on an adaptive optimized resource allocation policy, adding at least one more programmable logic-based processing unit to the first set of programmable logic-based processing units from the plurality of programmable logic-based processing units; and

The Nollet-Tuan-Seo combination discloses this element. EX1003, ¶¶110-118. As discussed above for elements [1.1]-[1.6], Nollet discloses dynamically reconfiguring a computing platform for executing at least two programs, including

an example where a user is watching a movie (second multi-stage program instance) on a portable device until he decides he wants to play a 3D game (first multi-stage program instance) while a scene in the movie is playing. EX1004, [0019], [0107]; EX1003, ¶110. When the user first starts watching the movie, the movie decoder starts and reconfigurable hardware is used for this task. *Id.* “When the user decides to start the 3D game, there are not enough resources available on the reconfigurable hardware to run both [the movie and 3D game] applications,” but “the movie can be downscaled in resolution and frame rate” to reduce computational requirements, “free[ing] up some hardware resources and allow[ing] the 3D game to be started as a hardware task.” *Id.*, [0107].

At a higher level, Nollet discloses reallocating at least one processing unit “based on parameters like task priority and processor affinity.” *Id.*, [0008] (“how to decide what task to run and where”), [0082]. Nollet further discloses task preemption, or reallocation of at least one processing unit, and explains that the applications comprise “multiple, possible independent, tasks running in parallel on the reconfigurable matrix.” *Id.*, [0082]. When preemption occurs, “the context of the current task can be saved, in order for this task to continue whenever additional computing resources are assigned to it.” *Id.* Nollet also discloses monitoring the state of the computing resource through a number of variables, which “mainly includes *the load of the computing unit.*” *Id.*, [0160]. In the example above, when

the processing load in the FPGA is too high to accommodate a first program instance (3D game), the movie tasks are reallocated to the CPU to make room for the 3D gaming tasks that have just been started, thus experiencing increased demand. *Id.*, [0107]. That is, in response to the detected increase in demand determined by monitoring the load on the processing units, at least one or more programmable logic-based processing units is allocated or added to the first program instance (3D game). *Id.*, [0107], [0160]; EX1003, ¶112.

Nollet further discloses that the reallocation or adding of reconfigurable processing units is performed by “an operating system that transparently manages the complexity of hierarchical reconfiguration.” *Id.*, [0158] (“the invention provides a filtering application on top of a microcontroller and a parameterizable filter block, both instantiated inside a reconfigurable hardware device such as an FPGA”), [0159] (“the operating system will have to keep track of the available computing resources as well as the executing OS4RS tasks.”), [0086] (“The operating system preferably keeps track of the available resources, meaning location of all the tasks running.”). Nollet explains that for managing preemption, “a kind of memory controller/programmer is provided” that “takes care of saving/reloading context and configuration of the ... tiles.” *Id.*, [0085].

As noted in Section V.A.1, *supra*, Nollet does not constrain the scheduling mechanisms/techniques employed by the OS and middleware to any specific

algorithm. EX1003, ¶113. Rather, Nollet leaves such implementation details and choices to the POSITA. *Id.* Seo provides an example of a task scheduling algorithm that a POSITA would have found it obvious to employ when implementing Nollet's system. *Id.*

As explained in Section V.A.3, Seo describes a real-time task scheduling mechanism for a multicore processor that attempts to optimize power consumption while maintaining the necessary performance to satisfy the demands of the running applications. EX1006, 1; EX1003, ¶114. One part of Seo's scheduling algorithm, called dynamic core scaling, "decides on *the optimal number of cores for the current performance demand*" and tries to meet this criterion as far as all deadlines are guaranteed." *Id.*, 2. "While under high task loads, having more cores is always more efficient; below a certain task load, having more cores actually consumes more energy." *Id.*, 7. Thus, dynamic core scaling can "save a considerable amount of leakage power in low load periods." *Id.*

Seo's dynamic core scaling algorithm is based on the "power consumption expectation function X ." *Id.*, 8. Seo's dynamic core scaling algorithm attempts to minimize the power consumption expectation function X by determining the optimal core number n for given load L . *Id.*, 8-9; EX1003, ¶115. Thus, Seo's dynamic core scaling algorithm requires that the multicore system track the dynamic utilization L_n (i.e., load) of every core C_n in the system. *Id.*

Seo defines the equation for the dynamic utilization L_n (i.e., load) of a given core C_n in equation (9). *Id.*, 5. Load L_n is based on the most recent actual execution time cc_i of all finished tasks and the Worst Case Execution Time (WCET) w_i of all unfinished tasks, where w_i “is defined as the maximum execution time required to complete [task] τ_i at the highest possible processor frequency.” *Id.*, 4-5. Thus, in order to determine the load L_n of a given core C_n , Seo’s algorithm must monitor at least two different characteristics for each core: the most recent actual execution time cc_i of all finished tasks assigned to that core and the Worst Case Execution Time (WCET) w_i of all unfinished tasks assigned to that core. EX1003, ¶116. A POSITA would have recognized that each of cc_i (first characteristic) and w_i (second characteristic) are characteristics corresponding to the cores running the tasks associated with each of the executing applications (which, in the context of Nollet’s reconfigurable hardware would be both the first and second set of programmable logic-based processing units”) and that their use in evaluating L_n means that these characteristics are used to determine increases in demand for the applications (i.e., “program instances”) associated with the tasks executing on each of the cores. EX1003, ¶116.

Seo’s algorithm then determines “the *optimal* core number for given [load] L , which is newly updated with each change in the task state.” EX1006, 9. In other words, Seo’s dynamic core scaling algorithm is “an adaptive optimized resource

allocation policy,” as claimed. EX1003, ¶117. Specifically:

If the *optimal* number is smaller than the current number of active cores, *try_to_shrink* will be called to reduce the number of active cores.

If the *optimal* number exceeds the current number of active cores, the *Activate* function will be called to increase the number of active cores.

Id. In this way, the dynamic core scaling algorithm “*adjusts* the number of active cores to reduce leakage power consumption under low load conditions.” *Id.*, 1.

When applied to the context of Nollet’s reconfigurable hardware, a POSITA would have found it obvious to utilize Seo’s dynamic core scaling algorithm to determine the optimal number of active tiles/cores in Nollet’s dynamically reconfigurable hardware. EX1003, ¶118. If the demand on the tiles/cores by the active tasks exceeds a threshold, the dynamic core scaling algorithm would add an active core/tile (i.e., adding at least one more programmable logic-based processing unit to the first set of programmable logic-based processing units from the plurality of programmable logic-based processing units). EX1003, ¶118.

For at least the reasons discussed above in Section V.A.1-6, a POSITA would have found it obvious and been motivated to utilize the concepts of Seo’s dynamic task scheduling algorithms in the context of the Nollet and Tuan’s reconfigurable multi-core/tile hardware systems.

[1.8] in response to a second increased demand for the second multi-stage program instance determined by monitoring a second characteristic different from the first characteristic, corresponding to the second set of programmable logic-

based processing units, based on the adaptive optimized resource allocation policy, adding at least one more programmable logic-based processing unit to the second set of programmable logic-based processing units from the plurality of programmable logic-based processing units.

The Nollet-Tuan-Seo combination discloses this element. EX1003, ¶¶119-121. In particular, as discussed above with respect to Element [1.7], Nollet discloses dynamically reconfiguring a computing platform for executing at least two programs, including an example where a user is watching a movie (second multi-stage program instance) on a portable device until he decides he wants to play a 3D game (first multi-stage program instance) while a scene in the movie is playing. EX1004, [0019], [0107].

Seo's dynamic core scaling algorithm must monitor at least two different characteristics for each core: the most recent actual execution time cc_i of all finished tasks assigned to that core and the Worst Case Execution Time (WCET) w_i of all unfinished tasks assigned to that core. See EX1006, 4-5, 8-9; EX1003, ¶120. A POSITA would have recognized that each of cc_i (first characteristic) and w_i (second characteristic) are characteristics corresponding to the cores running the tasks associated with each of the executing applications (which, in the context of Nollet's reconfigurable hardware would be both the first and second "set of programmable logic-based processing units") and that their use in evaluating L_n means that these characteristics are used to determine increases in demand for the applications (i.e., "program instances") associated with the tasks executing on each of the cores.

When applied to the context of Nollet's reconfigurable hardware, a POSITA would have found it obvious to utilize Seo's dynamic core scaling algorithm to determine the optimal number of active tiles/cores in Nollet's dynamically reconfigurable hardware. EX1003, ¶121. If the demand on the tiles/cores by the active tasks exceeds a threshold, the dynamic core scaling algorithm would add an active core/tile (i.e., adding at least one more programmable logic-based processing unit to the second set of programmable logic-based processing units from the plurality of programmable logic-based processing units). EX1003, ¶121.

8. Claim 2

[2.0] The method of claim 1, wherein the allocation policy relates to balancing load among programmable logic-based processing units.

The Nollet-Tuan-Seo combination discloses this element. EX1003, ¶¶122-125. Seo describes that:

An activation of a slept core occurs when the current active core number is less than the power-optimal number. The decision of whether or not to expand is made whenever L increases. It is also done when a task was newly released, and it is unable to be scheduled in C_{min} [i.e., the core with the lowest L].

EX1006, 9. After the activation operation is complete, Seo describes calling "the repartitioning function," which "balances the L values of the cores" by moving "the earliest unfinished deadline task in C_{max} to the C_{src} ." *Id.* Thus, Seo's dynamic core scaling algorithm also relates to balancing load among programmable logic-based

processing units, as recited in [2.0]. EX1003, ¶124.

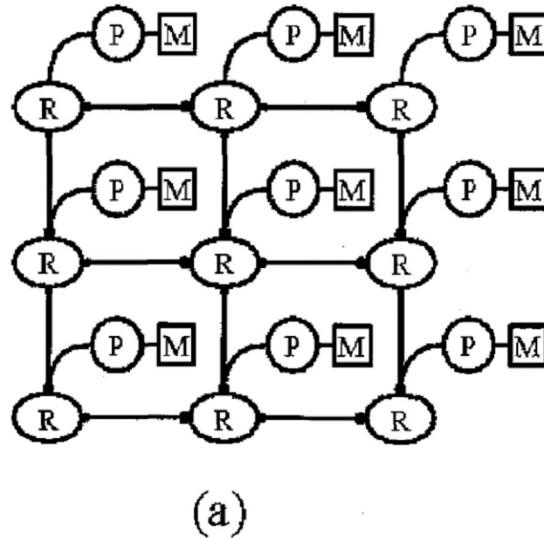
Moreover, as explained in Section V.A.3, there is another aspect of Seo's task assignment algorithm called dynamic repartitioning. EX1006, 2. Dynamic repartitioning "tries to keep the *performance demands of each core* balanced by migrating tasks from the core with the *highest demand* to the one with the *lowest demand*." *Id.* Thus, the dynamic repartitioning portion of Seo's algorithm relates to balancing load among programmable logic-based processing units, as recited in [2.0]. EX1003, ¶125.

9. Claim 3

[3.0] The method of claim 1, wherein each of the first set of programmable logic-based processing units comprises a field programmable gate array (FPGA).

The Nollet-Tuan-Seo combination discloses this element. EX1003, ¶¶126-130. Nollet describes that its "apparatus comprises a plurality of resources, more in particular at least of one programmable processor and a reconfigurable hardware device." EX1004, [0056]. The "reconfigurable hardware device can be a logic gate array, e.g. a Field Programmable Gate Array (FPGA)." EX1004, [0057]. Nollet describes that the "the operating system is aware of the tile usage at any moment," and that, as a result, "it can spawn a new task without placement overhead by replacing the tile content through partial reconfiguration of the FPGA." EX1004, [0065]. "For example, the OS can keep track of which tile of an FPGA is being used to execute a task." EX1004, [0086].

As shown in FIG. 7(a), “[e]mbodiments of the invention are implemented as a whole network with all the processors on the same FPGA.” EX1004, [0104].



Nollet (EX1004), FIG. 7(a)

In other words, each of the first set of programmable logic-based processing units comprises a field programmable gate array (FPGA), because a portion of a field programmable gate array is itself a field programmable gate array. EX1003, ¶130.

10. Claim 4

[4.0] The method of claim 1, wherein each of the second set of programmable logic-based processing units comprises a field programmable gate array (FPGA).

For at least the same reasons discussed with respect to claim 3, the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶¶131-133.

11. Claim 5

[5.0] The method of claim 1, wherein the first characteristic or the second characteristic is load or utilization.

The Nollet-Tuan-Seo combination discloses this element. EX1003, ¶¶134-

136. As discussed with respect to Elements [1.7], Seo defines the equation for the **dynamic utilization** L_n (i.e., load) of a given core C_n in equation (9). EX1006, 5. Load L_n is based on the most recent actual execution time cc_i of all finished tasks and the Worst Case Execution Time (WCET) w_i of all unfinished tasks, where w_i “is defined as the maximum execution time required to complete [task] τ_i at the highest possible processor frequency.” *Id.*, 4-5. Thus, in order to determine the load L_n of a given core C_n , Seo’s algorithm must monitor at least two different characteristics for each core: the most recent actual execution time cc_i of all finished tasks assigned to that core and the Worst Case Execution Time (WCET) w_i of all unfinished tasks assigned to that core. EX1003, ¶136. Thus, the first and second characteristics are constituent parts of **dynamic utilization** L_n (i.e., load) of a given core C_n . EX1003, ¶136.

12. Claim 6

[6.0] A non-transitory computer program product comprising instructions configured to perform a method parallel data processing platform comprising a plurality of programmable logic-based processing units for accelerating a service comprising at least a first program and a second program different than the first program, the method comprising:

The Nollet-Tuan-Seo combination discloses this element. EX1003, ¶139. Nollet explains that the “operating system software can be supplied as a computer program product which includes code segments which when executed on a computing device execute methods in accordance with one aspect of the invention.”

EX1004, [0088]. Moreover, as discussed above with respect to Element [1.0], the Nollet-Tuan-Seo combination discloses a method on a parallel data processing platform comprising a plurality of programmable logic-based processing units for accelerating a service comprising at least a first program and a second program, different than the first program. *Supra* Section V.A (Element [1.0]).

[6.1] forming a first set of inter-task communication paths connecting a first set of programmable logic-based processing units of the plurality of programmable logic-based processing units into a first multi-stage program instance,

For the same reasons as discussed above for element [1.1], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶140.

[6.2] wherein the first multi-stage program instance is configured to accelerate the first program, and

For the same reasons as discussed above for element [1.2], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶141.

[6.3] wherein the first set of programmable logic-based processing units are interconnected using reconfigurable cross-connects;

For the same reasons as discussed above for element [1.3], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶142

[6.4] forming a second set of inter-task communication paths connecting a second set of programmable logic-based processing units of the plurality of programmable logic-based processing units to a second multi-stage program instance,

For the same reasons as discussed above for element [1.4], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶143.

[6.5] wherein the second multi-stage program instance is configured to accelerate the second program, and

For the same reasons as discussed above for element [1.5], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶144.

[6.6] wherein the second set of programmable logic-based processing units are interconnected using reconfigurable cross-connects;

For the same reasons as discussed above for element [1.6], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶145

[6.7] in response to a first increased demand for the first multi-stage program instance determined by monitoring a first characteristic corresponding to the first set of programmable logic-based processing units, based on an adaptive optimized resource allocation policy, adding at least one more programmable logic-based processing unit to the first set of programmable logic-based processing units from the plurality of programmable logic-based processing units; and

For the same reasons as discussed above for element [1.7], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶146.

[6.8] in response to a second increased demand for the second multi-stage program instance determined by monitoring a second characteristic different from the first characteristic, corresponding to the second set of programmable logic-based processing units, based on the adaptive optimized resource allocation policy, adding at least one more programmable logic-based processing unit to the second set of programmable logic-based processing units from the plurality of programmable logic-based processing units.

For the same reasons as discussed above for element [1.8], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶147.

13. Claim 7

[7.0] The non-transitory computer program product of claim 6, wherein the allocation policy relates to balancing load among programmable logic-based processing units.

For the same reasons as discussed above for element [2.0], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶149.

14. Claim 8

[8.0] The non-transitory computer program product of claim 6, wherein each of the first set of programmable logic-based processing units comprises a field programmable gate array (FPGA).

For the same reasons as discussed above for element [3.0], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶151.

15. Claim 9

[9.0] The non-transitory computer program product of claim 6, wherein each of the second set of programmable logic-based processing units comprises a field programmable gate array (FPGA).

For the same reasons as discussed above for element [4.0], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶153.

16. Claim 10

[10.0] The non-transitory computer program product of claim 6, wherein the first characteristic or the second characteristic is load or utilization.

For the same reasons as discussed above for element [5.0], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶155.

17. Claim 11

[11.0] A system comprising:

The Nollet-Tuan-Seo combination discloses this element. EX1003, ¶158. As

discussed above with respect to Element [1.0], the Nollet-Tuan-Seo combination discloses a parallel data processing platform, which is a system. *Supra* Section V.A (Element [1.0]).

[11.1] a plurality of programmable logic-based processing units, each processing unit being configurable in at least a first configuration and a second configuration, different from the first configuration;

For the same reasons as discussed above for element [1.1], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶159.

[11.2] a first set of inter-task communication paths connecting a first set of programmable logic-based processing units of the plurality of programmable logic-based processing units into a first multi-stage program instance,

For the same reasons as discussed above for element [1.1], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶160.

[11.3] wherein the first-multi-stage program instance is configured to accelerate the first program, and

For the same reasons as discussed above for element [1.2], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶161.

[11.4] wherein the first set of programmable logic-based processing units are interconnected using reconfigurable cross-connects;

For the same reasons as discussed above for element [1.3], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶162.

[11.5] a second set of inter-task communication paths connecting a second set of programmable logic-based processing units of the plurality of programmable

logic-based processing units to a second multi-stage program instance configured to execute the second program, and

For the same reasons as discussed above for elements [1.4]-[1.5], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶163.

[11.6] wherein the second set of programmable logic-based processing units are interconnected using reconfigurable cross-connects; and

For the same reasons as discussed above for element [1.6], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶164.

[11.7] a hardware logic and/or one or more processors executing computer-executable instructions configured to

For the same reasons as discussed above for element [6.0], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶165.

[11.8] (1) add, based on an allocation policy, in response to a first increased demand for the first multi-stage program instance determined by monitoring a first characteristic corresponding to the first set of programmable logic-based processing units, at least one more programmable logic-based processing unit to the first set of programmable logic-based processing units from the plurality of programmable logic-based processing units, and

For the same reasons as discussed above for element [1.7], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶166.

[11.9] (2) add, based on the allocation policy, in response to a second increased demand for the second multi-stage program instance determined by monitoring a second characteristic, different from the first characteristic, corresponding to the second set of programmable logic-based processing units, at least one more programmable logic-based processing unit to the second set of programmable logic-based processing units from the plurality of programmable logic-based processing units.

For the same reasons as discussed above for element [1.8], the Nollet-Tuan-

Seo combination discloses this element. EX1003, ¶167.

18. Claim 12

[12.0] The system of claim 11, wherein the allocation policy relates to balancing load among programmable logic-based processing units.

For the same reasons as discussed above for element [2.0], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶169.

19. Claim 13

[13.0] The system of claim 11, wherein each of the first set of programmable logic-based processing units comprises a field programmable gate array (FPGA).

For the same reasons as discussed above for element [3.0], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶171.

20. Claim 14

[14.0] The system of claim 11, wherein each of the second set of programmable logic-based processing units comprises a field programmable gate array (FPGA).

For the same reasons as discussed above for element [4.0], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶173.

21. Claim 15

[15.0] The system of claim 11, wherein the first characteristic or the second characteristic is load or utilization.

For the same reasons as discussed above for element [5.0], the Nollet-Tuan-Seo combination discloses this element. EX1003, ¶175.

VI. MANDATORY NOTICES UNDER 37 C.F.R. §42.8

A. Real Parties-In-Interest Under 37 C.F.R. §42.8(b)(1)

Microsoft is the real party-in-interest. No other parties had access to or control over the present Petition, and no other parties funded the present Petition.

B. Related Matters Under 37 C.F.R. §42.8(b)(2)

ThroughPuter, Inc. (hereinafter, “ThroughPuter”)—the alleged Patent Owner—filed a complaint on April 13, 2022, in the U.S. District Court for the Western District of Washington (Case No. 2:22-CV-00492) against Microsoft, asserting two patents, including the ’948 patent. The complaint was served on April 19, 2022.

C. Lead And Back-Up Counsel Under 37 C.F.R. §42.8(b)(3)

Petitioner provides the following designation of counsel.

LEAD COUNSEL	BACK-UP COUNSEL
W. Karl Renner, Reg. No. 41,265 Fish & Richardson P.C. 60 South Sixth Street, Suite 3200 Minneapolis, MN 55402 Tel: 202-783-5070 Fax: 877-769-7945 Email: IPR42342-0120IP1@fr.com	David L. Holt, Reg. No. 65,161 Jennifer Huang, Reg. No. 64,297 Nicholas Stephens, Reg. No. 74,320 60 South Sixth Street, Suite 3200 Minneapolis, MN 55402 Tel: 202-783-5070 Fax: 877-769-7945 PTABInbound@fr.com

D. Service Information

Please address all correspondence and service to the address listed above. Petitioner consents to electronic service by email at IPR42342-0120IP1@fr.com (referencing No. 42342-0120IP1 and cc'ing PTABInbound@fr.com, axf-ptab@fr.com, holt2@fr.com, jjh@fr.com, and nstephens@fr.com).

VII. PAYMENT OF FEES – 37 C.F.R. §42.103

Petitioner authorizes the Office to charge Deposit Account No. 06-1050 for the petition fee set in 37 C.F.R. §42.15(a) and for any other required fees.

VIII. CONCLUSION

Accordingly, Petitioner respectfully requests IPR of the Challenged Claims based upon the Ground 1 presented herein.

Respectfully submitted,

Dated: April 17, 2023

/David L. Holt/
W. Karl Renner, Reg. No. 41,265
David L. Holt, Reg. No. 65,161
Jennifer Huang, Reg. No. 64,297
Nicholas Stephens, Reg. No. 74,320
Fish & Richardson P.C.
60 South Sixth Street, Suite 3200
Minneapolis, MN 55402
Tel: 202-783-5070
Fax: 877-769-7945

(Control No. IPR2023-00852)

Attorneys for Petitioner

CERTIFICATION UNDER 37 CFR § 42.24

Under the provisions of 37 CFR § 42.24(d), the undersigned hereby certifies that the word count for the foregoing Petition for *Inter partes* Review totals 13,641 words, which is less than the 14,000 allowed under 37 CFR § 42.24.

Dated : April 17, 2023

/David L. Holt/

W. Karl Renner, Reg. No. 41,265
David L. Holt, Reg. No. 65,161
Jennifer Huang, Reg. No. 64,297
Nicholas Stephens, Reg. No. 74,320
Fish & Richardson P.C.
60 South Sixth Street, Suite 3200
Minneapolis, MN 55402
Tel: 202-783-5070
Fax: 877-769-7945

Attorneys for Petitioner

CERTIFICATE OF SERVICE

Pursuant to 37 CFR §§ 42.6(e)(4)(i) *et seq.* and 42.105(b), the undersigned certifies that on April 17, 2023, a complete and entire copy of this Petition for *Inter Partes* Review and all supporting exhibits were provided by Federal Express, to the Patent Owner by serving the correspondence address of record as follows:

Gardella Grace P.A.
80 M Street SE, 1st Floor
Washington, DC 20003

/Diana Bradley/
Diana Bradley
Fish & Richardson P.C.
60 South Sixth Street, Suite 3200
Minneapolis, MN 55402
(858) 678-5667