

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORPORATION,
Petitioner,

v.

NEURAL AI, LLC,
Patent Owner.

Case No. IPR2025-00610
U.S. Patent No. RE48,438

Petition for *Inter Partes* Review of U.S. Patent No. RE48,438

Petition 2

Mail Stop **PATENT BOARD**
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

	<u>Page</u>
TABLE OF AUTHORITIES	xix
PETITIONER’S EXHIBIT LIST	xx
TABLE OF ABBREVIATIONS	xxii
MANDATORY NOTICES.....	xxiii
A. Real Parties-in-Interest.....	xxiii
B. Related Matters [37 C.F.R. § 42.8(b)(2)].....	xxiii
C. Lead and Back-Up Counsel [37 C.F.R. § 42.8(b)(3)].....	xxiii
D. Service Information [37 C.F.R. § 42.8(b)(4)].....	xxiv
I. INTRODUCTION	1
II. REQUIREMENTS OF <i>INTER PARTES</i> REVIEW	2
A. Standing.....	2
B. Identification of Challenge and Relief Requested	2
1. U.S. Patent No. 7,139,003 (“Kirk”) (Ex1005).....	3
2. K. Oh, <i>GPU implementation of neural networks</i> (“Oh”) (Ex1007).....	3
3. JPH04-237388A (“Tamura”) (Ex1008).....	3
4. GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (“GPU Gems”) (Ex1032).....	4
C. How the Challenged Claims Are To Be Construed Under 37 C.F.R. § 42.104(b)(3).....	4
D. How the Challenged Claims Are Unpatentable Under 37 C.F.R. § 42.104(b)(4).....	4

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
E. Supporting Evidence Under 37 C.F.R. § 42.104(b)(5)	5
F. Payment of Fees	5
III. TECHNICAL BACKGROUND	5
A. State of the Art Prior to the '438 Patent	5
1. Background on GPU Acceleration of Non-Graphics Computations and Its Applications	5
2. “Controllers” for Parallel Processing in the Prior Art	7
3. Accelerating Computations by Using Outputs as Inputs	8
B. Alleged Invention of the '438 Patent	9
C. Prosecution History of the '438 Patent	11
D. Person of Ordinary Skill in the Art	12
IV. GROUNDS OF UNPATENTABILITY	12
A. Grounds 1–4: Kirk and Oh, Additionally in Combination With Tamura and GPU Gems, Render Obvious Claims 1–14, 16–34, and 40–54	12
1. Overview of Ground and Prior Art	13
a. Summary of Kirk	13
b. Summary of Oh	17
c. Summary of Tamura	18
d. GPU Gems	19
2. Combinations	21
a. Combination of Kirk and Oh	21
b. Combination of Tamura with Kirk/Oh	23

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
3. Detailed Analysis of Grounds 1–4.....	24
a. Claim 1.....	24
i. Element 1[pre]: “A computer system, comprising”	24
ii. Element 1[a]: “a central processing unit to receive input data”	25
iii. Element 1[b]: “main memory, operably coupled to the central processing unit via a bus, to store the input data received by the central processing unit”	26
iv. Element 1[c]: “an accelerator, operably coupled to the central processing unit and the main memory via the bus, to receive at least a portion of the input data from the main memory, the accelerator comprising”	27
v. Element 1[c][i]: “at least one graphics processing unit to perform a sequence of computations on the at least a portion of the input data so as to generate output data, the sequence of computations representing an artificial neural network, intermediate computations in the sequence of computations representing respective layers of the artificial neural network and yielding intermediate results”	28
vi. Element 1[c][ii]: “accelerator memory, operably coupled to the at least one graphics processing unit, to store the results of the sequence of computations”	30

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
vii. Element 1[d]: “a controller, operably coupled to the at least one graphics processing unit and the accelerator memory”	32
viii. Element 1[d][i]: “to initialize textures and shaders in the accelerator memory for performing the sequence of computations”	35
ix. Element 1[d][ii]: “to control performance of the sequence of computations by the at least one graphics processing unit”	37
x. Element 1[d][iii]: “to transfer the at least a portion of the input data into the accelerator memory during performance of the intermediate computations in the sequence of computations by the at least one graphics processing unit”	38
xi. Element 1[d][iv]: “to transfer at least a portion of the output data from the accelerator memory to the main memory during performance of the intermediate computations in the sequence of computations by the at least one graphics processing unit”	40
b. Claim 2: “The computer system of claim 1, wherein the central processing unit is configured to receive the input data in response to a user interaction.”	42
c. Claim 3.....	43
i. Element 3[a]: “The computer system of claim 1, wherein the central processing unit is configured to receive the input data at a first rate”	43

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
ii. Element 3[b]: “the at least one graphics processing unit is configured to perform the sequence of computations at a second rate different than the first rate”	44
d. Claim 4: “The computer system of claim 1, wherein the main memory is configured to store a copy of the output data stored in the accelerator memory.”	44
e. Claim 5: “The computer system of claim 1, wherein an output of at least one computation in the sequence of computations represents an output of at least one neuron in an artificial neural network.”	45
f. Claim 6.....	45
i. Element 6[a]: “The computer system of claim 1, wherein accelerator memory comprises: a first memory bank to store parameters common to all of the computations in the sequence of computations”	45
ii. Element 6[b]: “a second memory bank to store data specific to at least one computation in the sequence of computations”	46
g. Claim 7: “The computer system of claim 1, wherein the controller is configured to transfer the output data from the accelerator memory to the main memory without transferring any of the intermediate results from the accelerator memory to the main memory so as to reduce data transfer via the bus.”	47
h. Claim 8: “The computer system of claim 1, wherein the controller is configured to transfer at least a portion of the output data from the accelerator memory to the main memory after the at least one graphics processing unit has begun to perform another sequence of computations.”	48

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
i. Claim 9: “The computer system of claim 8, wherein the controller is configured to initiate transfer of the at least a portion of the input data and to transfer the at least a portion of the output data in parallel with performance of at least one computation in the other sequence of computations by the at least one graphics processing unit.”	49
j. Claim 10: “The computer system of claim 1, wherein the controller is configured to control execution of the sequence of computations by the at least one graphics processing unit.”	50
k. Claim 11: “The computer system of claim 1, further comprising: at least one of a video camera, a microphone, or a cell recording electrode, operably coupled to the central processing unit, to acquire the input data in real time.”	51
l. Claim 12.....	51
i. Element 12[pre]: “A method of performing a sequence of computations representing an artificial neural network on a computer system comprising a central processing unit (CPU), a main memory operably coupled to the central processing unit via a bus, an accelerator operably coupled to the CPU and the main memory via the bus, the accelerator comprising a graphics processing unit (GPU) and an accelerator memory, the method comprising”	51
ii. Element 12[a]: “(A) performing, by the GPU, the sequence of computations on a first portion of input data so as to generate a first portion of output data, the first portion of the output data representing an output of a neuron	

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
in a first layer of the artificial neural network, intermediate computations in the sequence of computations yielding intermediate results, wherein performing the sequence of computations on the first portion of the input data comprises”	52
iii. Element 12[a][i]: “(i) assigning an output variable to a first texture and a second texture, the output variable being included in a first computational element of a plurality of computational elements, the plurality of computational elements representing the sequence of computations and”	52
iv. Element 12[a][ii]: “(ii) accumulating a first value for the output variable in the first texture during a first time step”	53
v. Element 12[b]: “(B) in parallel with performing the sequence of computations by the GPU in (A), transferring a second portion of the input data from the main memory to the accelerator via the bus”	53
vi. Element 12[c]: “(C) in parallel with performing the sequence of computations by the GPU in (A), transferring a second portion of the output data from the accelerator memory to the main memory via the bus, the second portion of the output data representing an output of a neuron in a second layer in the artificial neural network”	54
vii. Element 12[d]: “(D) performing, by the GPU, the sequence of computations on the second portion of the input data, wherein performing	

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
the sequence of computations on the second portion of the input data comprises”	55
viii. Element 12[d][i]: “(i) accumulating a second value for the output variable in the second texture during a second time step and”	55
ix. Element 12[d][ii]: “(ii) making the first value of the output variable in the first texture accessible to other computational elements in the plurality of computational elements during the second time step”	55
m. Claim 13: “The method of claim 12, further comprising: storing the input data in the main memory in response to a user interaction.”	56
n. Claim 14.....	57
i. Element 14[a]: “The method of claim 12, further comprising: receiving the input data at a first rate; and”	57
ii. Element 14[b]: “wherein (A) comprises performing the sequence of computations at a second rate different than the first rate”	57
o. Claim 16: “The method of claim 12, wherein (C) comprises: transferring the second portion of the output data from the accelerator memory to the main memory without transferring any of the intermediate results of the plurality of sequential computations from the accelerator memory to the main memory so as to reduce data transfer via the bus.”	57
p. Claim 17: “The method of claim 12, wherein (C) comprises: transferring the second portion of the output data from the accelerator memory to the main	

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
memory after the GPU has begun to perform another sequence of computations.”	58
q. Claim 18: “The method of claim 17, wherein (C) further comprises: initiating transfer of the second portion of the output data in parallel with performance of at least one computation in the other sequence of computations.”	58
r. Claim 19: “The method of claim 12, further comprising: acquiring the input data in real time with at least one of a video camera, a microphone, or a cell recording electrode operably coupled to the CPU.”	58
s. Claim 20.....	58
i. Element 20[a]: “The method of claim 12, further comprising: storing parameters common to all of the computations in the sequence of computations in a first memory bank in the accelerator memory; and”	58
ii. Element 20[b]: “storing data specific to at least one computation in the sequence of computations in a second memory bank in the accelerator memory”	59
t. Claim 21.....	59
i. Element 21[pre]: “A method of performing a sequence of computations representing an artificial neural network, the method comprising:”	59
ii. Element 21[a]: “receiving, at a central processing unit (CPU), first input data acquired from an external system in real time” ...	59

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
iii. Element 21[b]: “initializing, by a controller operably coupled to a graphics processing unit (GPU), textures and shaders in a memory operably coupled to the GPU”	59
iv. Element 21[c]: “transferring the first input data received by the CPU to the memory operably coupled to the GPU”	59
v. Element 21[d]: “performing, by the graphics processing unit (GPU), a first computation in the sequence of computations on the first input data based on the textures and shaders to generate first output data, computations in the sequence of computations representing respective layers of neurons in the artificial neural network, an output of the first computation in the sequence of computations representing an output of a first neuron in a first layer in the artificial neural network”	60
vi. Element 21[e]: “storing, in the memory operably coupled to the GPU, the first input data and the first output data”	60
vii. Element 21[f]: “transferring second input data acquired from the external system in real time into the memory operably coupled to the GPU after the GPU starts the first computation and before the GPU starts a second computation of the sequence of computations, an output of the second computation in the sequence of computations representing an output of a second neuron in a second layer in the artificial neural network”	60
u. Claim 22: “The method of claim 21, wherein transferring the second input data comprises	

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
transferring the second input data via a bus operably coupled to the CPU.”	61
v. Claim 23: “The method of claim 21, further comprising: transferring the first output data from the memory to another memory during the second computation in the sequence of computations.”	61
w. Claim 24.....	62
i. Element 24[a]: “The method of claim 23, further comprising: storing intermediate results of the sequence of computations in the memory, and”	62
ii. Element 24[b]: “wherein transferring the first output data from the memory to the other memory occurs without transferring the intermediate results of the sequence of computations”	62
x. Claim 25: “The method of claim 23, wherein transferring the second input data and transferring the first output data occurs in parallel.”	62
y. Claim 26: “The method of claim 21, further comprising: storing, in a first memory partition of the memory, parameters common to all of the computations in the sequence of computations.”	62
z. Claim 27: “The method of claim 26, further comprising: storing, in a second memory partition of the memory, data specific to the first computation in the sequence of computations.”	63
aa. Claim 28: “The method of claim 27, further comprising: storing, in the second memory partition, external input data patterns, representations of internal variables, an input of the computation in the	

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
sequence of computations, and the output of the computation in the sequence of computations.”.....	63
bb. Claim 29: “The method of claim 21, wherein storing the first output data comprises: accumulating, in the memory, outputs of computational elements executed by the GPU in performing the first computation in the sequence of computations.”.....	65
cc. Claim 30.....	65
i. Element 30[a]: “The method of claim 21, further comprising: storing, in the memory, an output of a previous computation in the sequence of computations”.....	65
ii. Element 30[b]: “accessing, by the GPU, the output of the previous computation during performance of the computation in the sequence of computations”.....	65
dd. Claim 31: “The method of claim 21, wherein performing the first computation comprises executing a plurality of computational elements representing a layer of neurons in an artificial neural network.”.....	66
ee. Claim 32: “The method of claim 31, wherein all neurons in the layer of neurons are described by the same equation.”.....	66
ff. Claim 33: “The method of claim 21, further comprising: acquiring the second input data with at least one of a video camera, a microphone, or a cell recording electrode.”.....	66
gg. Claim 34: “The method of claim 21, further comprising: loading the second input data from disk.”.....	67

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
hh. Claim 40.....	67
i. Element 40[pre]: “A system for executing an artificial neural network, the system comprising:”	67
ii. Element 40[a]: “a central processing unit (CPU) to provide first input data;”	68
iii. Element 40[b]: “a memory, operably coupled to the CPU, to store the first input data in a first partition, referenced by a first pointer, before computing a first layer of neurons of the artificial neural network”	68
iv. Element 40[c]: “a processing unit, operably coupled to the memory, to perform, during computation of the first layer of neurons, at least one calculation on the first input data so as to generate first output data, the first output data representing an output of at least one neuron in the first layer of neurons”	69
v. Element 40[d]: “a controller, operably coupled to the processing unit and the memory, to:”	69
vi. Element 40[d][i]: “store the first output data in a second partition of the memory, the second partition referenced by a second pointer, and to swap the first pointer with the second pointer at the end of the computation of the first layer of neurons, such that the first output data becomes an input for a second layer of neurons of the artificial neural network”	70

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
vii. Element 40[d][ii]: “transfer the first output data to another memory during computation of the second layer of neurons, and”	71
viii. Element 40[d][iii]: “dictate an order of execution of instructions to the processing unit to perform the computation of the first layer of neurons.”	72
ii. Claim 41: “The system of claim 40, wherein the processing unit comprises a graphics processing unit.”	73
jj. Claim 42: “The system of claim 40, wherein the controller is configured to send instructions for performing the at least one calculation to the processing unit.”	73
kk. Claim 43.....	73
i. Element 43[a]: “The system of claim 40, wherein the memory further comprises: a third partition to store internal variables; and”	73
ii. Element 43[b]: “a fourth partition to store data used as input at a particular layer of neurons of the artificial neural network”	74
ll. Claim 44.....	75
i. Element 44[pre]: “A computer system, comprising:”	75
ii. Element 44[a]: “a central processing unit to receive input data acquired from an external system”	75
iii. Element 44[b]: “main memory, operably coupled to the central processing unit via a	

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
bus, to store the input data received by the central processing unit”	75
iv. Element 44[c]: “an accelerator, operably coupled to the central processing unit and the main memory via the bus, to receive at least a portion of the input data from the main memory, the accelerator comprising”	75
v. Element 44[c][i]: “at least one processing unit to perform a sequence of computations representing an artificial neural network on the at least a portion of the input data so as to generate output data, intermediate computations in the sequence of computations representing layers of the neural network and yielding intermediate results”	76
vi. Element 44[c][ii]: “accelerator memory, operably coupled to the at least one processing unit, to store the results of the sequence of computations”	75
vii. Element 44[d]: “a controller, operably coupled to the at least one processing unit and the accelerator memory”	76
viii. Element 44[d][i]: “to control transfer of the at least a portion of the input data into the accelerator memory during performance of the intermediate computations in the sequence of computations by the at least one processing unit”	76
ix. Element 44[d][ii]: “to control transfer at least a portion of the output data from the accelerator memory to the main memory during performance of the intermediate	

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
computations in the sequence of computations by the at least one processing unit”	76
x. Element 44[d][iii]: “to control performance of the sequence of computations by the at least one processing unit”	76
mm. Claim 45: “The computer system of claim 44, wherein the central processing unit is configured to receive the input data in response to a user interaction.”	76
nn. Claim 46.....	77
i. Element 46[a]: “The computer system of claim 44, wherein: the central processing unit is configured to receive the input data at a first rate; and”	77
ii. Element 46[b]: “the at least one processing unit is configured to perform the sequence of computations at a second rate different than the first rate”	77
oo. Claim 47: “The computer system of claim 44, wherein the main memory is configured to store a copy of the output data stored in the accelerator memory.”	77
pp. Claim 48: “The computer system of claim 44, wherein an output of at least one computation in the sequence of computations represents an output of at least one neuron in an artificial neural network.”	77
qq. Claim 49.....	77
i. Element 49[pre]: “The computer system of claim 44, wherein accelerator memory comprises:”	77

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
ii. Element 49[a]: “a first memory partition to store parameters common to all of the computations in the sequence of computations; and”	77
iii. Element 49[b]: “a second memory partition to store data specific to at least one computation in the sequence of computations”	77
rr. Claim 50: “The computer system of claim 44, wherein the controller is configured to transfer the output data from the accelerator memory to the main memory without transferring any of the intermediate results from the accelerator memory to the main memory so as to reduce data transfer via the bus.”	78
ss. Claim 51: “The computer system of claim 44, wherein the controller is configured to transfer at least a portion of the output data from the accelerator memory to the main memory after the at least one processing unit has begun to perform another sequence of computations.”	78
tt. Claim 52: “The computer system of claim 51, wherein the controller is configured to initiate transfer of the at least a portion of the input data and to transfer the at least a portion of the output data in parallel with performance of at least one computation in the other sequence of computations by the at least one processing unit.”	78
uu. Claim 53: “The computer system of claim 44, wherein the controller is configured to control execution of the sequence of computations by the at least one processing unit.”	78

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
vv. Claim 54: “The computer system of claim 44, further comprising: at least one of a video camera, a microphone, or a cell recording electrode, operably coupled to the central processing unit, to acquire the input data in real time.”	78
V. ENABLEMENT AND SECONDARY CONSIDERATIONS	78
VI. DISCRETIONARY DENIAL	79
VII. CONCLUSION.....	79
APPENDIX: CHALLENGED CLAIM LISTING	82

TABLE OF AUTHORITIES

	<u>Page(s)</u>
Cases	
<i>Ohio Willow Wood Co. v. Alps South, LLC</i> , 735 F.3d 1333 (Fed. Cir. 2013)	78

PETITIONER’S EXHIBIT LIST

Exhibit	Description
1001	U.S. Patent No. RE48,438
1002	File History for U.S. Patent No. RE48,438 (Appl. No. 15/808,201)
1003	Declaration of Prof. Tajana Rosing, Ph.D.
1004	U.S. Patent No. 7,861,060 (“Nickolls”)
1005	U.S. Patent No. 7,139,003 (“Kirk”)
1006	Z. Luo, <i>Artificial Neural Network Computation on Graphic Process Unit</i> (IEEE 2005) (“ANN”)
1007	K. Oh, GPU implementation of neural networks (2004) (“Oh”)
1008	Japanese Unexamined Patent Appl. No. H04-237388A (“Tamura”)
1009	<i>The C programming Language</i> (1988)
1010	Excerpts of Patent Owner’s Infringement Contentions
1011	Numerical Recipes in C (2d ed. 2002)
1012	Jeanne Martin, <i>Fortran 90 Pointers vs. “Cray” Pointers</i> , 11 ACM SIGPLAN Fortran Forum (1992)
1013	Arthur Veen, <i>Dataflow Machine Architecture</i> (1986)
1014	Michael Flynn, <i>Some Computer Organizations and Their Effectiveness</i> (1972)
1015	Press Release – NVIDIA Launches the World’s First Graphics Processing Unit; GeForce 256 (Aug. 31, 1999)
1016	Excerpts of OpenGL Shading Language (2004)
1017	Excerpts of The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics (2003)
1018	OpenGL 2.1 Reference Pages
1019	Advanced Image Processing with DirectX 9 Pixel Shaders (2004)
1020	GPGPU: Basic Math Tutorial
1021	Ian Buck, <i>Data Parallel Computation on Graphics Hardware</i> (2003)
1022	Youquan Liu, <i>Real-Time 3D Fluid Simulation on GPU with Complex Obstacles</i> (2004)

Exhibit	Description
1023	Declaration of Dr. Mary Bolin
1024	Declaration of Gordon McPherson
1025	Thomas Rolfes, <i>Artificial Neural Networks on Programmable Graphics Hardware</i> in Game Programming Gems 4 (2004)
1026	P.J.G. Lisboa, <i>A review of evidence of health benefits from artificial neural networks in medical intervention</i> (2002)
1027	U.S. Patent Publ. No. 2003/0140179 (“Wilt”)
1028	Bertil Svensson, <i>SIMD Processor Array Architectures</i> in PARALLEL PROCESSING IN INDUSTRIAL REAL-TIME APPLICATIONS (1992)
1029	Michael Glover, <i>A Massively-Parallel SIMD Processor for Neural Network and Machine Vision Applications</i> (1993)
1030	Francisco Mesa-Martinez, <i>The UCSC Kestrel High Performance SIMD Processor: Present and Future</i> (2003)
1031	<i>Sotera Stipulation</i>
1032	GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (2005)

TABLE OF ABBREVIATIONS

Abbreviation	Term
'201 application	U.S. Patent Appl. No. 15/808,201
'438 patent	U.S. Patent No. RE48,438
'828 patent	U.S. Patent No. 9,189,828
ANN	Z. Luo, <i>Artificial Neural Network Computation on Graphic Process Unit</i> (IEEE 2005) (Ex1006)
Board	Patent Trial and Appeal Board
Challenged Claims	Claims 1–14, 16–34, and 40–54 of U.S. Patent No. RE48,438
Ex.	Exhibit
Fig.	Figure
GPU Gems	GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (2005) (Ex1032)
IPR	<i>inter partes</i> review
Kirk	U.S. Patent No. 7,139,003 (Ex1005)
Litigation	<i>Neural AI, LLC v. NVIDIA Corporation</i> , No. 7:24-cv-00221 (W.D. Tex.)
Nickolls	U.S. Patent No. 7,861,060 (Ex1004)
Oh	K. Oh, <i>GPU implementation of neural networks</i> (2004) (Ex1007)
Patent Owner	Neural AI, LLC
Petitioner	NVIDIA Corporation
POSITA[s]	person[s] of ordinary skill in the art
Tamura	Japanese Unexamined Patent Appl. No. H04-237388A (Ex1008)
USPTO	United States Patent and Trademark Office

MANDATORY NOTICES

A. Real Parties-in-Interest

NVIDIA Corporation is the real party-in-interest.

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

The '438 patent is asserted in *Neural AI, LLC v. NVIDIA Corporation*, No. 7:24-cv-00221 (W.D. Tex.) (“the Litigation”). Petitioner is concurrently filing a second petition for IPR challenging the claims of the '438 patent based on different prior art grounds (IPR2025-00609). Additionally, Petitioner has filed petitions in the following proceedings as to patents related to the '438 patent and asserted in the Litigation: IPR2025-00606 (U.S. Patent No. 8,648,867) and IPR2025-00608 (U.S. Patent No. RE49,461).

C. Lead and Back-Up Counsel [37 C.F.R. § 42.8(b)(3)]

Pursuant to 37 C.F.R. §§ 42.8(b)(3), 42.8(b)(4), and 42.10(a), Petitioner provides the following designation of counsel:

Lead Counsel for Petitioner	Back-Up Counsel for Petitioner
Brian M. Buroker (Reg. No. 39,125) Gibson, Dunn & Crutcher LLP 1050 Connecticut Ave. NW Washington, DC 20036 Phone: (202) 955-8500 Fax: (202) 467-0539 Email: bburoker@gibsondunn.com	L. Kieran Kieckhefer (<i>pro hac vice</i> forthcoming) One Embarcadero Center Suite 2600 San Francisco, CA 94111 Phone: (415) 393-8200 Email: kkieckhefer@gibsondunn.com

Lead Counsel for Petitioner	Back-Up Counsel for Petitioner
	Nathan Curtis (Reg. No. 70,471) Gibson, Dunn & Crutcher LLP 2001 Ross Avenue Suite 2100 Dallas, TX 75201 Phone: (214) 698-3423 Fax: (214) 571-2961 Email: ncurtis@gibsondunn.com Vivian Lu (Reg. No. 74,443) Gibson, Dunn & Crutcher LLP 200 Park Avenue New York, NY 10166 Phone: (212) 351-3827 Email: vlu@gibsondunn.com

A Power of Attorney accompanies this Petition in accordance with 37 C.F.R. § 42.10(b).

D. Service Information [37 C.F.R. § 42.8(b)(4)]

Service via hand delivery or postal mail may be made at the addresses of the lead and back-up counsel above. Petitioner hereby consents to electronic service, and service by email may be made at:

- GDC-NVIDIA-IPR@gibsondunn.com

I. INTRODUCTION

Petitioner requests IPR of U.S. Patent No. RE48,438 and cancellation of claims 1–14, 16–34, and 40–54 as unpatentable under 35 U.S.C. § 103.

The '438 patent is the second of three patents in a family directed to a system for performing *general-purpose* computing (such as numerical simulations) on traditionally *special-purpose* graphics processing units (“GPUs”). The '438 patent’s pre-reissue claims were each amended or discarded entirely during reissue proceedings. But the shift to reciting the use of GPUs to perform computations “representing an artificial neural network,” including on data received in “real time,” does not save the new claims: Leveraging the parallel processing abilities of GPUs to perform “computationally expensive algorithms,” Ex1001, 1:38–42, such as processing real-time data in artificial neural networks, was well known prior to the '438 patent. For example, the prior art “Oh” article describes using a multi-layer artificial neural network on a commodity GPU to identify text in video in real time.

The reissue claims also recite well-known techniques, including timing data transfers so that the GPU is not idle while those transfers are occurring. The Kirk/Oh combination teaches transferring input and output data between the host system and GPU while the GPU performs computations for an artificial neural network, making the computations on real-time data in Oh more efficient. Similarly, Tamura discloses inputting data and writing it to memory while calculations are being performed on previously inputted data. And GPU Gems discloses the well-known

technique of swapping pointers, rather than unnecessarily moving large amounts of data, recited in a handful of claims.

II. REQUIREMENTS OF *INTER PARTES* REVIEW

A. Standing

Pursuant to 37 C.F.R. § 42.104(a), Petitioner certifies that the '438 patent is available for IPR and that Petitioner is not barred or estopped from requesting an IPR on the grounds identified herein.

B. Identification of Challenge and Relief Requested

Pursuant to 37 C.F.R. § 42.104(b), Petitioner requests the Board institute IPR of claims 1–14, 16–34, and 40–54 under pre-AIA 35 U.S.C. § 103.

The precise relief requested by Petitioner is that the Challenged Claims be canceled based on the grounds below:

Ground	Claims	Basis for Rejection¹
1	1–14, 16–34, 40–54	Obviousness over Kirk in view of Oh
2	1–14, 16–34, 40–54	Obviousness over Kirk in view of Oh and Tamura
3	40–43	Obviousness over Kirk in view of Oh and GPU Gems
4	40–43	Obviousness over Kirk in view of Oh, Tamura, and GPU Gems

¹ All obviousness grounds include the knowledge of a POSITA.

The above challenges are made in view of the following prior art references:

1. U.S. Patent No. 7,139,003 (“Kirk”) (Ex1005)

Kirk issued on November 21, 2006, from a non-provisional application filed on December 15, 2003. Kirk is assigned to NVIDIA. Kirk qualifies as prior art under pre-AIA 35 U.S.C. § 102(e). Kirk was not of record during prosecution.

2. K. Oh, *GPU implementation of neural networks* (“Oh”) (Ex1007)

Oh was published in 2004 in The Journal of the Pattern Recognition Society. Oh (with a copyright date of 2004 by reputable publisher Elsevier Ltd.) was available around April 2004. Ex1023, ¶¶1–19, 20–30. A skilled researcher searching for the subject matter of Oh using relevant terms such as “GPU” and “neural network” would have located Oh. *Id.* Several other references in the field cited Oh before August 2005, confirming it was locatable by interested persons. *Id.* Oh qualifies as prior art under pre-AIA 35 U.S.C. § 102(b). Oh was of record during prosecution but was not substantively discussed or applied by the Examiner.

3. JPH04-237388A (“Tamura”) (Ex1008)

Japanese Patent Appl. No. H04-237388 was filed on January 22, 1991, and published on August 25, 1992. Ex1008, 6. Tamura qualifies as prior art to the ’438 patent under pre-AIA 35 U.S.C. § 102(a) and (b). Tamura was not of record during prosecution.

4. GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (“GPU Gems”) (Ex1032)

GPU Gems was published by NVIDIA in March 2005 and publicly available no later than July 2005. Ex1023, ¶¶1–19, 31–41. GPU Gems’ “[f]irst printing” was March 2005 and it was cataloged and indexed at the University of Texas–Austin by July 18, 2005. *Id.* A POSITA, exercising reasonable diligence, would have located GPU Gems in July 2005 by searching for keywords like “GPU” and “programming,” which are contained in the searchable title, or “[c]omputer graphics” or “[r]eal-time programming,” which are contained in the searchable subject headings for GPU Gems. *Id.* GPU Gems qualifies as prior art under pre-AIA 35 U.S.C. § 102(a) and/or (b). GPU Gems was not of record during prosecution.

C. How the Challenged Claims Are to Be Construed Under 37 C.F.R. § 42.104(b)(3)

Petitioner submits that no construction of any claim term is necessary for the Board to resolve, and the Challenged Claims would have been obvious under any reasonable construction.

D. How the Challenged Claims Are Unpatentable Under 37 C.F.R. § 42.104(b)(4)

An explanation of how the Challenged Claims are unpatentable under the grounds identified above, including the identification of where each element of the claim is found in the prior art patents or printed publications, is provided below.

E. Supporting Evidence Under 37 C.F.R. § 42.104(b)(5)

The exhibit numbers of the supporting evidence relied upon and the relevance of the evidence, including an identification of specific portions of the evidence that support the challenge, are provided below. A list of exhibits with the exhibit number and a brief description of each exhibit is also included in this Petition pursuant to 37 C.F.R. § 42.63(e). The technical information and grounds for rejection are further supported by the Declaration of Prof. Tajana Rosing. Ex1003, ¶¶1–419.

F. Payment of Fees

Pursuant to 37 C.F.R. §§ 42.103 and 42.15(a), the required fee is being submitted herewith. The Office is authorized to charge any fee deficiency, or credit overpayment, to deposit account no. 50-1408.

III. TECHNICAL BACKGROUND

A. State of the Art Prior to the '438 Patent

1. Background on GPU Acceleration of Non-Graphics Computations and Its Applications

Parallel processing, for both graphics and general-purpose computing, was known for decades before the '438 patent. Ex1003, ¶¶33–35, ¶¶31–86. In 1999, NVIDIA developed the first consumer-level graphics card that could offload the graphics pipeline from the CPU, dubbing it a “Graphics Processing Unit,” or “GPU.” *Id.* Unlike CPUs, which excel at handling a few tasks sequentially, GPUs are

optimized for parallel processing, making them ideal for the thousands of simultaneous calculations required to render graphics. *Id.*, ¶¶36–39.

GPUs were used almost immediately to perform parallelized computations for non-graphics applications. *Id.*, ¶¶33–51. This type of computing has been referred to as “general-purpose computing on GPUs,” or “GPGPU.” *Id.* Traditionally, GPUs processed “textures” (stored as data arrays) by using “shaders” (*i.e.*, programs) to modify pixel data and create graphics for display. *Id.* In the early 2000s, researchers (including at and supported by NVIDIA) began repurposing shaders in programmable graphics pipelines for non-graphics purposes, such as scientific computing and mathematical simulations. *Id.* For these non-graphics applications, the programs are written as shaders and the data stored in textures. *Id.*

The parallel nature of GPU programming contributed directly to the rise in popularity of using GPUs for non-graphics applications, including artificial neural networks. *Id.* Artificial neural networks are computational systems modeled after the way the human brain’s biological neural networks operate. *Id.*, ¶¶73–79. They are built from layers of interconnected units, called nodes or “neurons,” which mimic the brain’s nerve cells. *Id.* Each node processes data and produces an output passed to other nodes. *Id.* By training on vast amounts of data, these networks learn to recognize patterns and make predictions by fine-tuning the strength of connections between nodes. *Id.* The training process for neural networks is computationally

intensive because it involves repeatedly adjusting the connection strengths between nodes across massive datasets, requiring millions or even billions of complex mathematical calculations, like matrix multiplication. *Id.* This makes GPUs well-suited for the task, as their parallel processing capabilities can handle many calculations simultaneously, significantly speeding up the training process. *Id.*

Similarly, since the early days of GPGPU, it was understood that GPUs' parallel processing capabilities made them suitable for real-time inputs. Researchers and engineers used real-time inputs, like video streams from cameras, to GPUs for tasks such as image processing and object tracking. *Id.*, ¶¶80–81. For example, the researchers in ANN used an artificial neural network on a GPU to track a ball in real time for robot soccer. Ex1006, 622. It was known in these applications processing real-time data to input data and write it to memory while calculations are performed on previously input data. Ex1003, ¶81.

2. “Controllers” for Parallel Processing in the Prior Art

Before the '438 patent, POSITAs created effective designs for parallel processing. Ex1003, ¶¶44–51. One common feature was a “controller” that coordinated the function of the parallel processors. *Id.* These systems often included a host computer or processor, an accelerator, and a controller separate from the CPU. *Id.* It was known that having the controller, rather than the CPU, manage the processors frees up the CPU to perform other functions, which is one of the benefits

the inventors allege for the “controller” in the ’438 patent. *Id.*; Ex1001, 2:19–20, 13:32–39; Ex1003, ¶¶44–51; Ex1029, 845; Ex1030, 2. These same principles were already well known in systems involving both a CPU and GPU.

3. Accelerating Computations by Using Outputs as Inputs

From the beginning, parallel processing using GPUs used well-known parallel processing techniques. For example, swapping address pointers—where the output of one computational step becomes the input for the next—was a basic technique in computer programming long before the ’438 patent. Ex1003, ¶¶65–70. This technique had been widely employed in high-performance computing, numerical simulations, and graphics processing. *Id.* Instead of physically moving large amounts of data between memory locations, incurring significant time and resource costs, pointer swapping allows programs to simply update the memory reference so the output data already in memory is immediately treated as the new input data. *Id.*

Pointer swapping existed before the development of GPUs. *Id.* The 1988 textbook *The C programming Language* teaches when two pieces of data need to be “swapped,” it is efficient to swap *pointers* to the data, rather than copying the data. Ex1009, 88. These techniques (also known as “double buffering” or “ping-pong rendering”) were used in GPUs prior to the ’438 patent. Ex1003, ¶¶65–70. The “double buffering” functionality in OpenGL used two framebuffers (dedicated areas of memory) and swapped pointers between them at the end of each frame. *Id.* GPU

Gems provides multiple examples of pointer swapping with GPUs. *Id.*; Ex1032, 507–08.

B. Alleged Invention of the '438 Patent

The '438 patent acknowledges “manufacturers of GPUs have included general purpose programmability into the GPU architecture leading to the increased popularity of using GPUs for highly parallelizable and computationally expensive algorithms outside of the computer graphics domain.” Ex1001, 1:38–42. However, the patent asserted “general purpose GPU (GPGPU) applications are not able to achieve optimal performance” because “[t]here is overhead for graphics-related features and algorithms that are not necessary for these non-video applications.” *Id.*, 1:42–47.

To address these alleged drawbacks, the patent describes an “accelerator” (Figure 2, below) having at least “one or more graphics processing units” (GPU 240), “two or more associated memory banks that are logically or physically partitioned” (shader memory bank 210 and texture memory bank 250), and “a specialized controller” (controller 220). *Id.*, 2:24–32. This arrangement allegedly improves performance because, rather than the system CPU, “[t]he controller handles most of the primitive operations needed to set up and control GPU computation.” *Id.*, 2:34–36. As a result, “the CPU is freed from this function and is dedicated to other tasks.” *Id.*, 2:36–37.

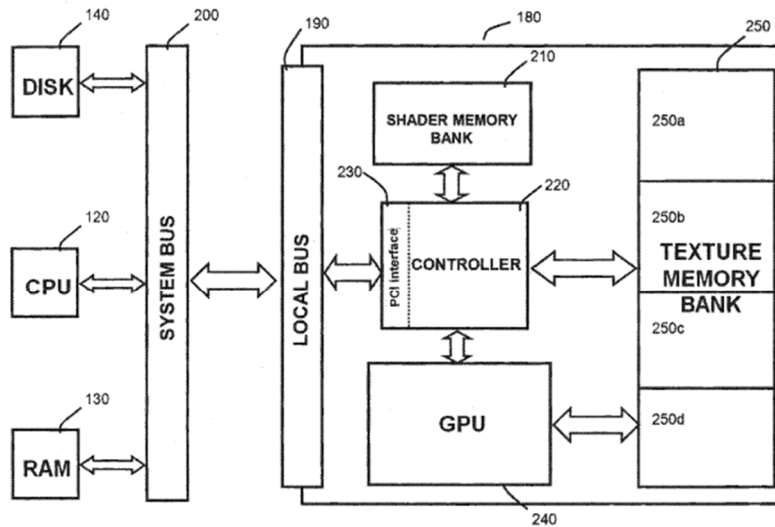


FIG. 2

During reissue, however, the '438 claims patent moved away from this alleged novelty. Instead, the reissued claims focus on well-known applications and techniques for the claimed "accelerator," such as using an accelerator to perform computations "representing an artificial neural network" and transferring input and output data between the accelerator and the host system *while* the GPU computes the layers of the artificial neural network. The '438 patent acknowledges that, prior to the patent, it was known that "neural networks" "can benefit from GPGPU computation." Ex1001, 1:51–55. According to the patent, each layer of the neural network outputs a result (representing a neuron) that is used as an input for the next layer. *Id.*, 6:13–19. The '438 patent also describes receiving input "in real time from a recording device," which is provided to the accelerator memory while the accelerator processes information. *Id.*, 8:29–35, 9:36–39, 13:23–39.

C. Prosecution History of the '438 Patent

The '438 patent is a reissue of U.S. Patent No. 9,189,828. None of the '828 patent's 20 claims recited artificial neural networks. Ex1001, 14:50–17:15.

On November 9, 2017, a reissue of the '828 patent was filed. *Id.*, Cover. The reissue application amended existing claims and added over a hundred new claims. Ex1002, 30–70. Over three years of prosecution, the applicant maintained the prior art did not disclose various limitations relating to the transfer of data to and from the GPU while the GPU performs computations. *Id.*, 640–69, 800–17, 934–48, 1006–08. The Examiner disagreed, identifying Diard as “demonstrating that it was known at the time of the invention to transfer between an accelerator memory and the main memory while the GPU is performing other operations.” *Id.*, 850.

To gain allowance, the applicant added “sequence of computations represent[s] an artificial neural network,” “intermediate computations in the sequence of computations represent[] respective layers of the artificial neural network,” and input and output of data to the GPU occurs “during performance of the intermediate computations.” *Id.*, 914–48. The applicant argued the amended claims “recite[d] transferring input data from a CPU to a GPU, transferring output data from neurons in a neural network from the GPU to the CPU, or both *while the GPU executes the layers of the neural network*,” and the prior art of record was “silent” on those features. *Id.*, 938 (emphasis in original).

On September 30, 2023, the Examiner issued a Notice of Allowance for 56 claims. *Id.*, 1035–44. In describing the “Allowable Subject Matter,” the Examiner explained that the prior art of record did not disclose the combination of the artificial neural network limitations with the limitations on transferring data to and from the GPU. *Id.*, 1040–44. The ’438 patent issued on February 16, 2021. Ex1001, Cover, 14:51–21:9.

D. Person of Ordinary Skill in the Art

The ’438 patent relates to the field of computer systems and architecture, including the use of GPUs. Ex1001, 1:30–47. A POSITA in the field around 2005–2006 would have had a Bachelor’s degree in electrical or computer engineering (or equivalent discipline) and at least two years’ experience in research, design and/or development of computer systems, including experience with GPUs. Ex1003, ¶¶15–20. Additional education could substitute for experience, and vice versa. *Id.*

IV. GROUNDS OF UNPATENTABILITY

A. Grounds 1–4: Kirk and Oh, Additionally in Combination With Tamura and GPU Gems, Render Obvious Claims 1–14, 16–34, and 40–54

Kirk’s GPU accelerator and Oh’s artificial neural network for analyzing real-time video, optionally with Tamura’s disclosure of inputting data simultaneously with processing data and/or GPU Gems’ disclosure of pointer swapping, discloses or renders obvious all elements of the Challenged Claims.

1. Overview of Grounds and Prior Art

a. Kirk

Kirk discloses a “Computing System” with a “programmable graphics processor” for “processing of data.” Ex1005, 1:40–47, 26:24–39, 26:56–61, Figs. 12A, 12B.² Kirk explains the “graphics data” it can process “is any data that is input to or output from computation units within Programmable Graphics Processor 105.” *Id.*, 3:35–41. Kirk includes Host Processor 114 (red), Host Memory 112 (blue), and Graphics Subsystem 107 (purple), all of which communicate through System Interface 115. *Id.*, 3:17–33.

² Many components in Figures 12A and 12B have the same reference numerals as in earlier figures. Thus, this Petition relies on disclosure provided in the context of those other figures to understand Figure 12.

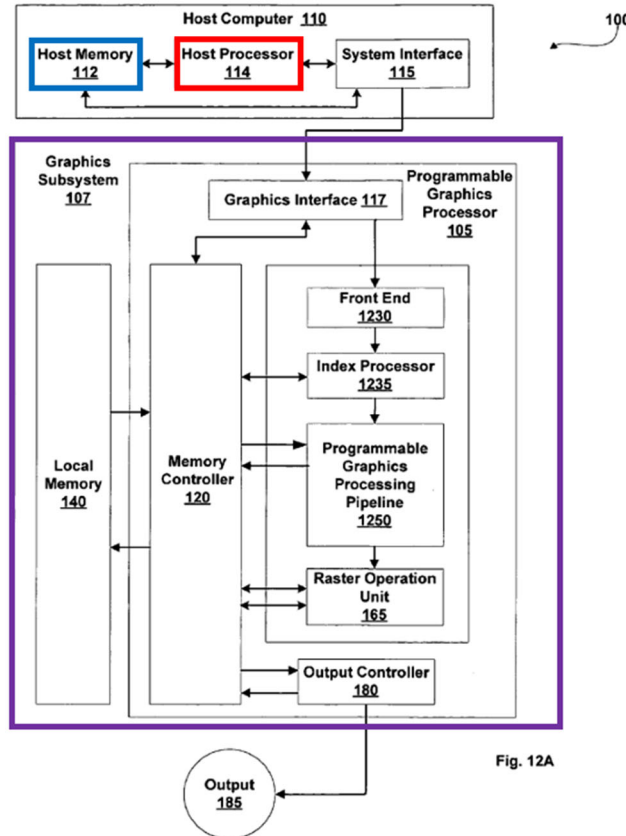
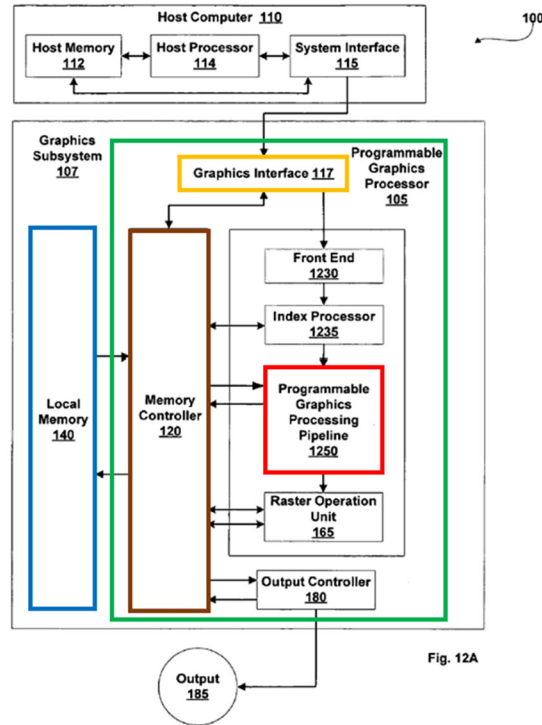


Fig. 12A

Id., Fig. 12A (annotated).

Graphics Subsystem 107 includes Local Memory 140 (blue) and Programmable Graphics Processor 105 (green). *Id.*, 3:32–46, 26:24–39. Programmable Graphics Processor 105 interfaces with the host system through Graphics Interface 117 (orange) and with Local Memory 140 (blue) through Memory Controller 120 (brown). *Id.*, 3:32–67, 26:24–39. Local Memory 140 stores input data, output data, and program instructions for graphics/data processing. *Id.*, 3:34–46. Programmable Graphics Processor 105 includes Programmable Graphics Processing Pipeline 1250 (red) in which data is processed according to instructions in memory. *Id.*, 26:32–61.



Id., Fig. 12A (annotated).

Programmable Graphics Processing Pipeline 1250 includes Execution Pipelines 1240, each of which “includes at least one multithreaded processing unit.” *Id.*, 27:2–11. Each execution pipeline communicates with graphics memory (including Local Memory 140) “to read program instructions and graphics data stored in buffers in graphics memory.” *Id.*, 27:12–28. Programmable Graphics Processing Pipeline 1250 also contains on-board memory to store data and instructions. *Id.*

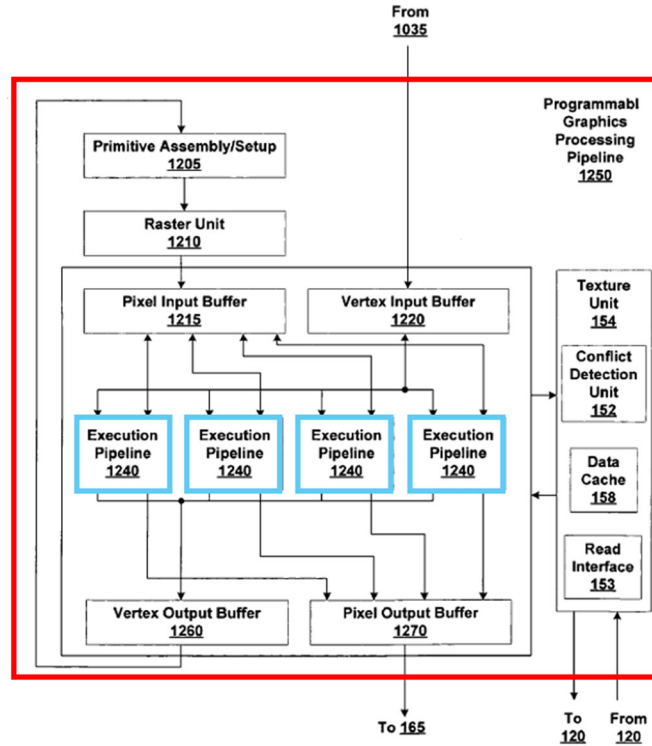


Fig. 12B

Id., Fig. 12B (annotated).

In operation, “[a] graphics program ... is executed within one or more Execution Pipelines 1240 as a plurality of threads where each vertex or fragment to be processed by the program is assigned to a thread.” *Id.*, 27:40–43. Results from the computations are sent to memory, such as Local Memory 140, Host Memory 112, or memory within Programmable Graphics Processing Pipeline. *Id.*, 3:35–46. Data produced by the processing pipeline “may be written to a buffer in graphics memory to be read from during a subsequent pass.” *Id.*, 4:16–20; Ex1003, ¶¶126–33.

b. Oh

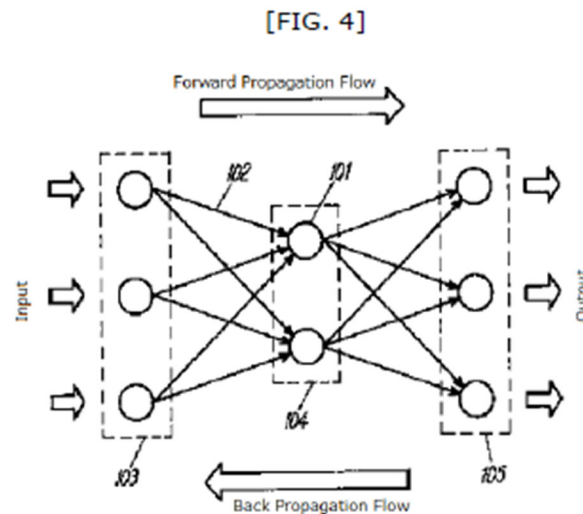
Oh “presents a faster [neural network] using common graphics hardware GPU.” Ex1007, 1. Oh explains that prior neural networks lacked the ability for “real-time processing.” *Id.* Oh thus “focuses on using a GPU to implement a multilayer perceptron” artificial neural network for real-time text detection in video. *Id.*, 1, 3. Oh teaches that a GPU’s “extended capabilities in supporting complex operations have also become useful in non-graphics applications,” including for neural networks. *Id.*, 2.

Oh’s multi-layer neural network performs matrix multiplication and a sigmoid function at each layer, yielding an output result. *Id.*, 2–3. In Oh’s neural network, “[t]he result of the previous layer is saved in the form of a render target texture, which is then used as an input for the next layer.” *Id.*, 3.

Oh applied these principles to program a neural network to “classify the pixels of input images, whereby the feature extraction and pattern recognition stage are integrated in the neural network.” *Id.* The system in Oh, which was run on commodity GPU hardware, took “image and video data” as input and analyzed the frames to identify where in each frame text was located. *Id.*, 3–4. The processing time for Oh’s neural network on a GPU as compared to a CPU was greatly reduced. *Id.*, 3–4; Ex1003, ¶¶134–38.

c. Tamura

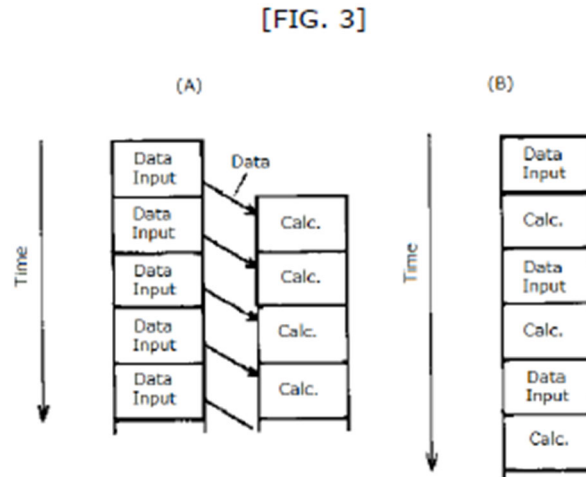
Tamura is directed to a processing unit for simulating neural networks, referred to as a “neuroprocessor.” Ex1008, ¶1. Tamura teaches that, “[b]y modeling and imitating the workings of neurons in the human brain, neural networks aim to create a new type of computer that excels at tasks such as recognition, association, optimization, and speech synthesis.” *Id.*, ¶2. Figure 4 shows the neural network, with circles 101 as neurons and lines 102 as connections between neurons in the first (input), intermediate, and output layers. *Id.*, ¶4.



Id., Fig. 4.

Tamura recognized, however, that “neural network calculations require a lot of processing time because they involve a very large amount of computation,” and the neural network processors of the prior art were inefficient because they would sequentially input data, perform calculations, input further data, and so forth. *Id.*, ¶¶26–27. To solve this, Tamura proposes inputting data and writing it to memory

while calculations are being performed on previously inputted data, such that “the total processing time is reduced.” *Id.*, ¶¶29, 30, 35, Fig. 3.



Id., Fig. 3; Ex1003, ¶¶139–42.

d. GPU Gems

GPU Gems describes the state of the art in GPU programming, including swapping pointers of input and output data. Ex1008, xx, xxxi–xxxii, 453. Chapter 31 explains that, to allow the output of one computation to be used as an input to another, one can use GPUs’ “render-to-texture” feature, where results of one step of a computation are written to a “texture,” *i.e.*, a data array or “buffer” in memory, which is identified by a pointer. *Id.*, 499, 501, 504. The buffer pointer can then be passed as the input to a later computation. *Id.*

Chapter 31 provides code to implement these concepts in a GPU. *Id.*, 505–08. The code “[c]reates a ‘double buffered’ PUGBuffer,” which “allow[s] the simulation to alternate using one as input, one as output.” *Id.*, 507.

Listing 31-2. C++ Code to Set Up and Run a Reaction-Diffusion Simulation on the GPU
See the source code on the accompanying CD for more details.

```
PUGBuffer *rdBuffer; PUGProgram *rdProgram;

void init_rd(){ // Call this once.
    pugInit(); // Start up the GPU framework

    // Create a "double-buffered" PUGBuffer. Two buffers allow the
    // simulation to alternate using one as input, one as output
    PUGBuffer *rdBuffer = pugAllocateBuffer(width, height,
                                           PUG_READWRITE, 4, true);

    PUGProgram *rdProgram = pugLoadProgram("rd.cg", "rd");

    pugBindFloat(rdProgram, "DuDv", du, dv); // bind parameters
    pugBindFloat(rdProgram, "F", F);
    pugBindFloat(rdProgram, "k", k);
```

Id., 507. At the end of each iteration, the code calls the “swap ()” function, which swaps the pointer for the currentSource buffer with that of currentTarget, allowing the output of one computational cycle to become the input for the next.

Listing 31-2 (continued). C++ Code to Set Up and Run a Reaction-Diffusion Simulation on the GPU

```
// Initialize the state of the simulation with values in array
pugInitBuffer(rdBuffer, array);
}

void update_rd(){ // Call this every iteration
    pugBindStream(rdProgram, "concentration", rdBuffer, currentSource);
    PUGRect range(0, width, 0, height);
    pugRunProgram(rdProgram, rdBuffer, range, currentTarget);

    std::swap(currentSource, currentTarget);
}
```

Id., 508 (annotated), 44, 713, 743; Ex1003, ¶¶143–47.

2. Combinations

a. Combination of Kirk and Oh

Before the '438 patent, it was well known that artificial neural networks—such as Oh’s for recognizing text in video—could be effectively implemented on commodity GPUs. Section III.A.1. A POSITA would have been motivated to

implement Oh's artificial neural network on NVIDIA GPU hardware, such as the GPU-accelerated parallel processing system in Kirk. Ex1003, ¶¶155–64.

Oh discloses a multi-layer neural network that takes in “video data” to detect text. Ex1007, 3. Although Oh used a RADEON GPU, Oh is agnostic as to the GPU used. Ex1003, ¶158. Thus, to achieve further efficiency in computation, a POSITA would have been motivated to implement Oh's neural network on state-of-the-art GPU-accelerated parallel processing systems, such as that in Kirk. *Id.*, ¶159. Through its multiple execution pipelines, Kirk would significantly reduce execution time compared to other processing architectures. *Id.*

A POSITA would have understood advantages to applying the artificial neural network of Oh on Kirk. *Id.*, ¶160. Kirk is designed to perform the parallel computations required for neural networks. For example, while Kirk focuses on processing of “graphics data,” Kirk explains that “graphics data is any data that is input to or output from computation units within [the] Programmable Graphics Processor 105.” Ex1005, 3:35–41. Thus, Kirk's parallel execution pipelines can process data associated with Oh's artificial neural network with reduced execution times compared to other architectures. Ex1003, ¶160.

Kirk would also be ideal for Oh's neural network because of its capabilities for data sharing. *Id.*, ¶161. To increase efficiency, in Oh's artificial neural network, “[t]he result of the previous layer is saved in the form of a render target texture,

which is then used as an input for the next layer.” Ex1007, 3. With its shared memory (Local Memory 140), Kirk would have allowed the iterative computations of Oh’s artificial neural network. Ex1003, ¶161.

Further, Kirk is assigned to NVIDIA, who was a leader in the industry in GPUs development. Ex1005, Cover; Ex1003, ¶162. A POSITA would have expected that the most advanced GPUs for performing GPGPU computations would have come from NVIDIA. Ex1003, ¶162. When seeking to execute complex artificial neural networks (such as that in Oh), a POSITA would have sought out advanced systems, including from NVIDIA. *Id.*

Kirk/Oh simply applied a known technique (*e.g.*, the artificial neural network of Oh) to a known device (*e.g.*, the GPU-accelerated parallel processing system of Kirk) ready for improvement to yield predictable results (*e.g.*, fast and efficient computations for an artificial neural network with real-time data). *Id.*, ¶163.

A POSITA would have had a reasonable expectation of success in combining Oh and Kirk. *Id.*, ¶164. Kirk was designed for running many tasks (threads) at the same time. Since simulating multi-layer artificial neural networks requires dividing the computation into smaller, discrete tasks, a POSITA would have expected Kirk to be able to use parallel processing to execute the computations required in Oh. *Id.*, ¶¶155–64.

b. Combination of Tamura with Kirk/Oh

Prior to the '438 patent, techniques existed to accelerate computations in neural networks, such as taught in Tamura. A POSITA would applied Tamura's data transfer and computation techniques to accelerate Kirk/Oh. Ex1003, ¶¶165–68.

To the extent Kirk/Oh does not teach that data processing and data input occur simultaneously, a POSITA would have motivated to apply the teachings of Tamura to Kirk/Oh. Tamura teaches that for an artificial neural network, while previously input data is processed and calculations are performed, new data is input and saved to memory. Ex1008, ¶¶29–30. A POSITA would reasonably expect the combination would improve computational efficiency. Ex1003, ¶166.

For real-time data, it would have been important to implement Kirk to simultaneously and continuously receive and process input. *Id.*, ¶167. A POSITA would have looked to the art for solutions to keep up with real-time data input. *Id.* Tamura discloses one such technique. *Id.* Tamura emphasizes its “invention is unrelated to the structure of the neural network arithmetic processing unit, and can be applied effectively to any neural network arithmetic processing unit.” Ex1008, ¶37. Tamura's teachings would be applicable to Kirk's GPU-accelerated system, which can process neural networks. Ex1003, ¶167.

The combination of Tamura with Kirk/Oh simply applied a known technique (*e.g.*, efficient data processing and data input in an artificial neural network) to a

known device (e.g., the GPU-accelerated parallel processing of Kirk/Oh) ready for improvement to yield predictable results (e.g., fast and efficient computations for an artificial neural network with real-time data input). *Id.*, ¶¶165–67.

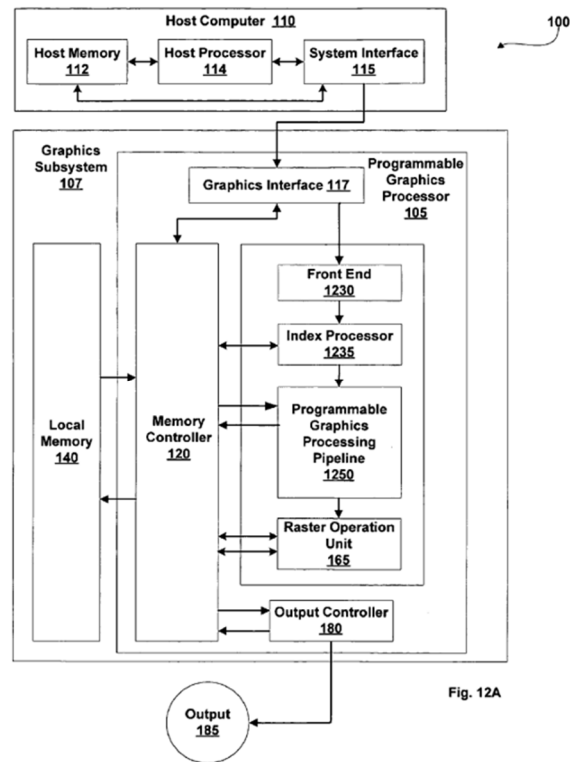
3. Detailed Analysis of Grounds 1–4

a. Claim 1

i. 1[pre]

If the preamble is limiting, Kirk/Oh discloses it. Ex1003, ¶¶170–71.

Kirk discloses a “Computing System” with a “programmable graphics processor” for “processing of data.” Ex1005, 1:40–47, 3:35–41, 26:24–39, 26:56–61, Figs. 12A, 12B.



Id., Fig. 12A.

Kirk includes Host Memory 112 (*i.e.*, “main memory”), which is coupled to Host Processor 114 via a bus (depicted as a black arrows in Fig. 12A), which were well known at the time. Ex1005, 3:17–31. Host Memory 112 stores input data (such as the video data for analysis), which the CPU can access and send to Graphics Subsystem 107 for processing. Ex1005, 3:32–34, 3:50–67, 4:26–31, 5:1–2.

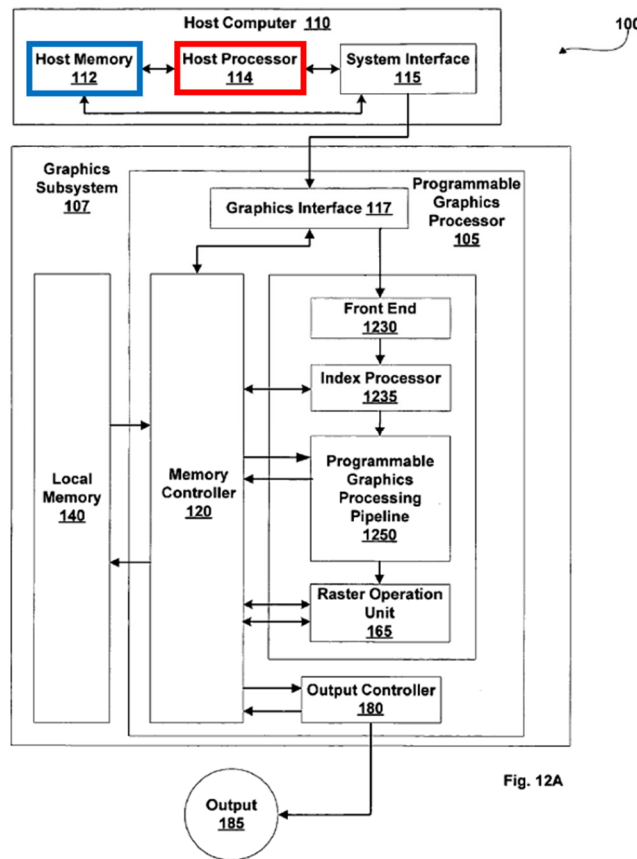


Fig. 12A

Ex1005, Fig. 12A (annotated).

iv. 1[c]

Kirk/Oh discloses this limitation. Ex1003, ¶¶177–79.

Kirk includes an accelerator (*e.g.*, Graphics Subsystem 107), which is coupled to the CPU (*e.g.*, Host Processor 114) and the main memory (*e.g.*, Host Memory

112) via the bus. Ex1005, 3:17–46, Fig. 12A. Host Memory 112 stores input data, which the CPU can access and send to Graphics Subsystem 107 for processing. *Id.*, 3:32–34, 3:50–67, 4:26–31. Host Processor 114 communicates with Host Memory 112 through a bus and with Graphics Interface 117 (and thus Graphics Subsystem 107) through System Interface 115 and the bus. *Id.* at 3:22–31. Thus, Graphics Subsystem 107 (*i.e.*, an accelerator) receives at least a portion of the input data from Host Processor 114 (*i.e.*, main memory).

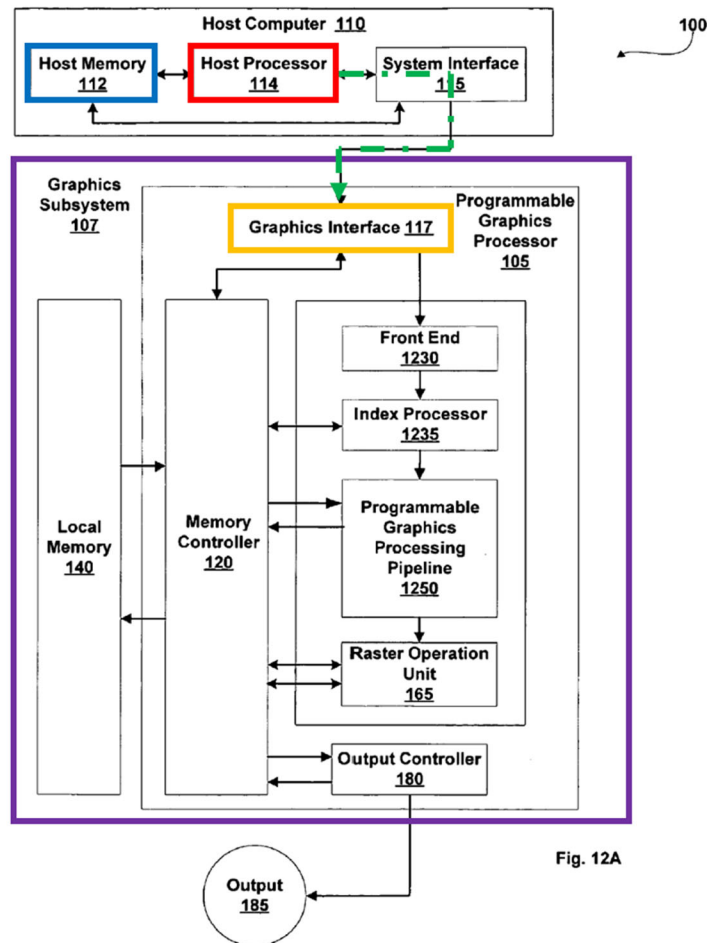


Fig. 12A

Ex1005, Fig. 12A (annotated).

v. 1[c][i]

Kirk/Oh discloses this limitation. Ex1003, ¶¶180–85.

Within Graphics Subsystem 107 (purple) of Kirk is Programmable Graphics Processor 105 (green), which includes Programmable Graphics Processing Pipeline 1250 (red). Ex1005, 3:32–46, 26:24–39. Programmable Graphics Processor 105 as a whole and Programmable Graphics Processing Pipeline 1250 each are a “graphics processing unit.” *Id.*, 26:24–61; Ex1003, ¶181; Ex1001, 1:32–34.

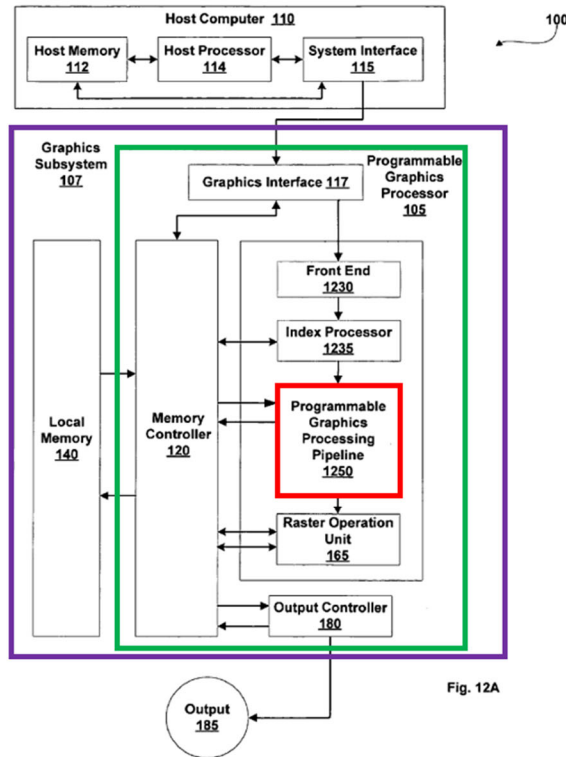


Fig. 12A

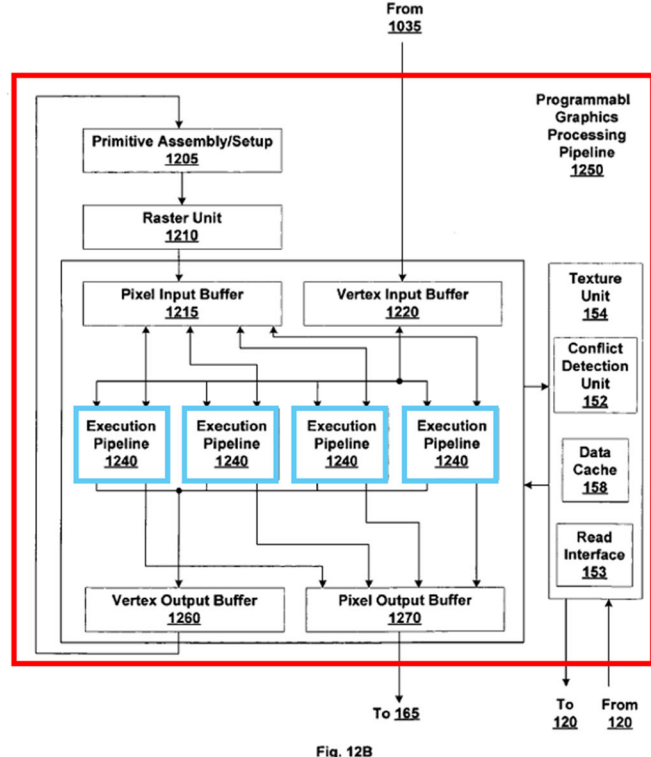


Fig. 12B

Id., Figs. 12A, 12B (annotated).

Programmable Graphics Processing Pipeline 1250 has multiple Execution Pipelines 1240, each of which “includes at least one multithreaded processing unit.”

Id., 27:2–11. Each execution pipeline communicates with graphics memory “to read

program instructions and graphics data stored in buffers in graphics memory.” *Id.*, 27:12–28. In operation, “[a] graphics program . . . is executed within one or more Execution Pipelines 1240 as a plurality of threads where each vertex or fragment to be processed by the program is assigned to a thread.” *Id.*, 27:40–43. Results from the computations are sent to memory, such as Local Memory 140, Host Memory 112, or memory within the programmable graphics processing pipeline. *Id.*, 3:35–46. Data produced by the processing pipeline (*i.e.*, “output data”) “may be written to a buffer in graphics memory to be read from during a subsequent pass.” *Id.*, 4:16–20. Programmable Graphics Processor 105, including its Programmable Graphics Processing Pipeline 1250, therefore performs a sequence of computations on the at least a portion of the input data so as to generate output data. EX1003, ¶¶182–83.

In Kirk/Oh, the sequence of computations represents an artificial neural network. *Id.*, ¶184. Oh’s multi-layer neural network includes an input layer with nodes (*i.e.*, neurons), multiple hidden layers with 30 nodes each, and an output layer with a single node, with each layer representing a layer of neurons in the artificial neural network. Ex1007, 1, 3. At each layer, the neural network relies on matrix multiplication and a sigmoid function to determine the output, which are fed into the next layer. *Id.* Thus, in Kirk/Oh, the intermediate computations represent layers of the artificial neural network and yield intermediate results (*e.g.*, the results of a layer of neurons other than the output layer). Ex1003, ¶¶180–85.

vi. 1[c][ii]

Kirk/Oh discloses this limitation. Ex1003, ¶¶186–89.

Kirk discloses that Graphics Subsystem 107 includes an accelerator memory (e.g., Local Memory 140) (blue) coupled to the graphics processing unit (e.g., Programmable Graphics Processor 105 or Programmable Graphics Processing Pipeline 1250). Ex1005, 3:32–46, 26:24–39. Programmable Graphics Processor 105 interfaces with the host system through Graphics Interface 117 (orange) and with Local Memory 140 (blue) through Memory Controller 120 (brown). *Id.*, 3:32–67, 26:24–39. Local Memory 140 stores the results of the sequences of computations performed in Programmable Graphics Processing Pipeline 1250 (red). *Id.*, 3:32–46, 5:8–11, 5:28–32, 7:17–25, 26:32–61.

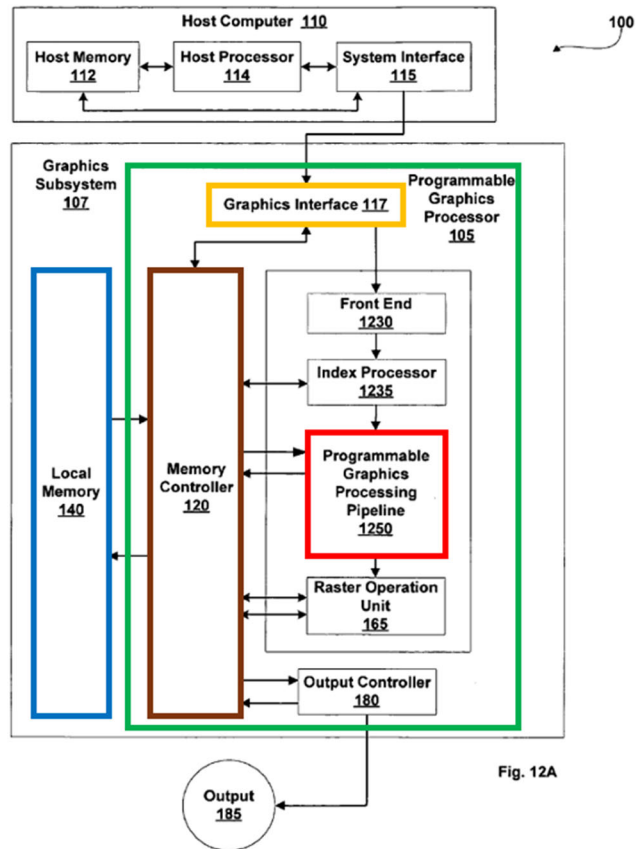


Fig. 12A

Id., Fig. 12A (annotated).

“Execution Pipelines 1240 are optionally configured . . . such that data processing operations are performed in multiple passes through at least one multithreaded processing unit within Execution Pipelines 1240.” *Id.*, 28:33–37. Thus, results of a sequence of computations can also be stored in memory within Programmable Graphics Processing Pipeline 1250, such as Vertex Output Buffer 1260 or Texture Unit 154. *Id.*, 27:29–31, 27:24–28, Fig. 12B. Because these memory locations are within Graphics Subsystem 107 (an “accelerator”), a POSITA would have considered them to be “accelerator memory.” Ex1003, ¶¶186–89.

vii. 1[d]

Kirk/Oh discloses this limitation. Ex1003, ¶¶190–93.

Kirk includes a “controller,” *e.g.*, Graphics Interface 117 along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180. Ex1005, 3:32–67, 26:24–61. These components (individually and collectively) are a “controller” because they control operation of Programmable Graphics Processor 105, including receiving, interpreting, formatting, and sending data and commands to the Programmable Graphics Processing Pipeline 1250 for execution. *Id.* Like the “controller” in the ’438 patent, the controller of Kirk handles the set up and control of the accelerator’s computations, so that “the CPU is freed from this function and is dedicated to other tasks.” Ex1001, 2:36–37, 5:17–29. Graphics Interface 117, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180 are coupled to Local Memory 140 (“accelerator memory”) through Memory Controller 120 and to Programmable Graphics Processing Pipeline 1250. Ex1005, 3:47–67, 26:24–61, Fig. 12A. Graphics Interface 117, Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180 are coupled to Programmable Graphics Processor 105 as they are part of it. Ex1003, ¶192.

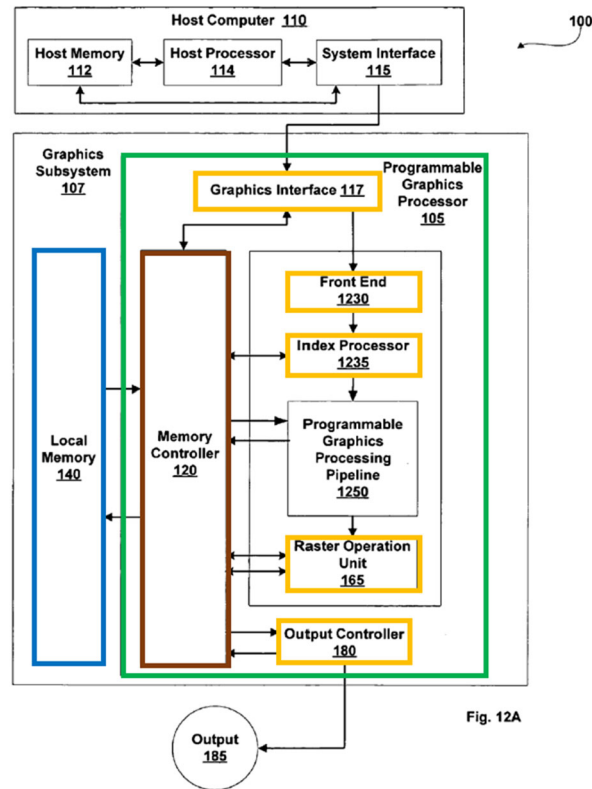
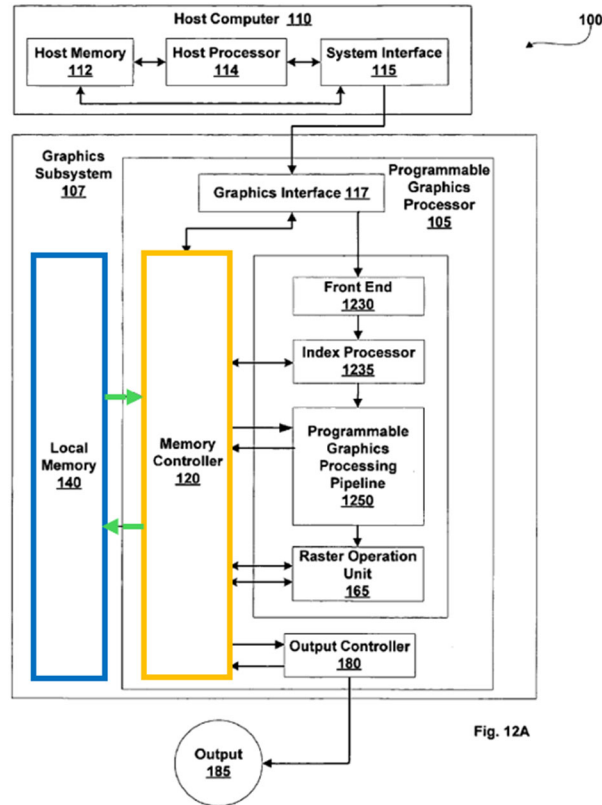


Fig. 12A

Id., Fig. 12A (annotated).

Memory Controller 120 transfers data into and out of Local Memory 140 and communicates with the Programmable Graphics Processing Pipeline 1250. Ex1005, 3:47–67, 4:26–31, 26:24–39. In the Litigation, Patent Owner asserts in its Infringement Contentions that a memory controller in the accused products (like Kirk’s Memory Controller 120) is the claimed “controller.” Ex1010, 55–56 (“Maxwell-architecture GPUs . . . include ‘memory controllers.’ . . . (e.g., *controller, operably coupled to the at least one graphics processing unit and the accelerator memory*).”); Ex1003, ¶193. Thus, additionally and alternatively, under Patent Owner’s view of the claims, Kirk’s memory controller (either alone or in

combination with Graphics Interface 117, Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180) is the claimed “controller.”³



Id., Fig. 12A.

viii. 1[d][i]

Kirk/Oh discloses this limitation. Ex1003, ¶¶194–200.

³ Petitioner disagrees with Patent Owner that a standard memory controller is a “controller” within the meaning of the ’438 patent. Nevertheless, for purpose of this Petition only, Petitioner applies Patent Owner’s allegation.

When Kirk is used for numerical computations (e.g., an artificial neural network in Kirk/Oh), the “controller” initializes textures and shaders in memory of Graphics Subsystem 107 (e.g., Local Memory 140 and/or memory within Programmable Graphics Processing Pipeline 1250). Ex1005, 3:35–67, 4:21–56, 6:4–25.

Initializing Textures. “Data ... received at Graphics Interface 117 can be ... written to Local Memory 140 through Memory Controller 120.” *Id.*, 3:50–54, 3:32–46, 4:57–5:11, 5:12–33, 6:25–33, 26:62–27:11. The data in Local Memory 140 is then available for use in computations within Programmable Graphics Processor 105, including in Programmable Graphics Processing Pipeline 1250. *Id.*, 3:35–46, 6:4–25. The received data comprise “textures” within the meaning of the ’438 patent because they are stored in data arrays. Ex1001, 5:20–22 (“[T]he term ‘texture’ in this document refers to a data array”); Ex1003, ¶196. The system further initializes textures (data arrays) for storing output data from computations. Ex1005, 5:24–45, 6:4–55, 11:22–41, 28:21–32; Ex1003, ¶196.

Initializing Shaders. “[P]rogram instructions received at Graphics Interface 117 can be ... written to Local Memory 140 through Memory Controller 120.” Ex1005, 3:50–54. The program instructions are then available to be loaded into the Programmable Graphics Processing Pipeline to be executed. *Id.*, 3:35–46, 5:34–45, 6:4–43, 26:24–39, 27:12–39, 28:34–37. The “program instructions” in memory are

“shaders” because they are “GPU program[s].” Ex1001, 5:24–26 (“[T]he term ‘shader’ in this document refers to a GPU program unless otherwise specified.”); EX1003, ¶197.

In Kirk/Oh, the “shaders” initialized in memory of Graphics Subsystem 107 would include the equations to be applied at each layer of the multi-layer neural network, and the “textures” initialized in memory of Graphics Subsystem 107 would include the initial neural network parameters, the weights to be applied at each layer, and the input video data. *Id.*, ¶198.

In the mapping in which Memory Controller 120 is part of the “controller,” and under Patent Owner’s theories of infringement, initialization of textures and shaders in the accelerator memory is performed by Memory Controller 120 because access to Local Memory 140 (including data and program instructions) occurs “via Memory Controller 120.” Ex1005, 6:25–30, 3:47–67, 4:26–31, 6:34–55, 26:24–39; Ex1003, ¶199.

Additionally, it would have been obvious to initialize textures (data arrays) and shaders (programs) in memory of the accelerator, making them available to the components performing computations. *Id.*, ¶200. These ideas are fundamental to GPGPU and were well known prior to the ’438 patent. Section III.A.1. For Programmable Graphics Processing Pipeline 1250 to process the input data using

the designated sequence of computations (provided in shaders), they would need to be loaded in memory available to the processing pipeline. *Id.*

ix. 1[d][ii]

Kirk/Oh discloses this limitation. Ex1003, ¶¶201–04.

Kirk’s “controller” (*i.e.*, Graphics Interface 117 along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180) controls performance of the computations in Programmable Graphics Processing Pipeline 1250. Ex1005, 3:35–67, 4:21–56, 27:2–28. Programmable Graphics Processing Pipeline 1250 comprises multiple Execution Pipelines 1240, each of which “includes at least one multithreaded processing unit.” *Id.*, 27:2–11. Each execution pipeline communicates with graphics memory (including Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) “to read program instructions and graphics data stored in buffers in graphics memory.” *Id.*, 27:12–28. Programmable Graphics Processing Pipeline 1250 “contain[s] one or more programmable processing units to perform a variety of specialized functions,” such as “table lookup, scalar and vector addition, multiplication, division, coordinate-system mapping, calculation of vector normal, tessellation, calculation of derivatives, interpolation, and the like.” *Id.*, 26:44–51.

Thus, because the instructions for performing the sequence of computations in Programmable Graphics Processing Pipeline 1250 (*i.e.*, a graphics processing

unit) and the data for such computations are provided based on instructions from the “controller,” the controller “control[s] performance of the sequence of computations by the at least one graphics processing unit.” Ex1003, ¶203.

In the mapping in which Memory Controller 120 is part of the “controller,” and under Patent Owner’s theories of infringement, the control of the performance of the sequence of computations by the at least one graphics processing unit is performed by Memory Controller 120 because the data and program instructions for such computations are provided “via Memory Controller 120.” Ex1005, 6:25–30, 3:47–67, 4:26–31, 6:34–55, 26:24–39; Ex1003, ¶204.

x. 1[d][iii]

Kirk/Oh discloses this limitation. Ex1003, ¶¶205–09.

In Kirk/Oh, input data (*e.g.*, one or more video frames) is transferred into accelerator memory during performance of the intermediate computations in the sequence of computations (*e.g.*, computations in the hidden layers of the multi-layer neural network). Ex1003, ¶207. In Kirk/Oh, because the input of Oh’s neural network is “video data,” which may even be real-time video data, after data for a first frame is transferred into accelerator memory and computations begin, input data for subsequent frames would be transferred into accelerator memory during computations on the earlier input data to keep up with the real-time data input. Ex1007, 3–5; Ex1005, 3:32–67; Ex1003, ¶207.

As explained in 1[c] and 1[c][ii], Graphics Subsystem 107 includes “accelerator memory” (*e.g.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) and a graphics processing unit (*e.g.*, Programmable Graphics Processing Pipeline 1250 or Programmable Graphics Processor 105) in which data is processed according to instructions stored in memory. Ex1005, 3:32–46, 26:24–61. In Kirk, input and output data, as well as commands, are stored in Local Memory 140. *Id.*, 3:34–46. When input data is received in Graphics Subsystem 107 from the host computer, Graphics Interface 117 transfers the data to Local Memory 140. *Id.*, 3:50–58; Ex1003, ¶206.

Tamura also discloses this limitation. Ex1003, ¶208. Tamura teaches, in the context of simulating a neural network, inputting data and writing it to memory at the same time that other input data is processed through the layers of an artificial neural network. Ex1008, ¶¶29–30, 35. When this teaching of Tamura is applied to Kirk/Oh, additional frames from the video input is transferred into accelerator memory during performance (by the accelerator) of intermediate computations in the artificial neural network on one or more earlier frames of input data. Ex1003, ¶208. It would have been obvious, in view of Tamura, to implement Kirk/Oh such that additional input data (*e.g.*, one or more frames) is stored in accelerator memory during performance of the intermediate computations in the sequence of computations (*e.g.*, layers of the artificial neural network) on prior input data (*e.g.*,

one or more prior frames). *Id.* Doing so would reduce processing time, as taught in Tamura, which is important in processing real-time data. Ex1008, ¶¶30, 35, Fig. 3; Ex1003, ¶209.

xi. 1[d][iv]

Kirk/Oh discloses this limitation. Ex1003, ¶¶210–17.

In Kirk/Oh, the transfer of output data (*e.g.*, results from the output layer of the artificial neural network) to main memory occurs as intermediate computations continue to be performed on input data (*e.g.*, video frames) by the graphics processing unit. Ex1003, ¶¶210–15; Ex1005, 5:8–11, 6:49–52, 5:60–6:3. In circumstances where results are written to Host Memory 112, the output is *first* written to memory in the accelerator (because that memory is closer to the processing engines and faster to access) and *then* copied (transferred) to system memory. *Id.*, 4:16–20, 5:8–11; Ex1003, ¶215. Kirk provides for large flexibility in when and where data is stored. Ex1005, 5:8–11, 6:49–52, 5:60–6:3. Where the artificial neural network is processing video data, output data from one pass through the multi-layer neural network (*i.e.*, “a portion of the output data”) is transferred to system memory while the processing cores continue to perform computations in the layers of the artificial neural network on input data. Ex1007, 2–45; Ex1003, ¶215. In short, in Kirk/Oh, there are overlapping computation and data transfers.

To the extent Kirk does not expressly teach transferring the output data from Local Memory 140 to main memory 112 (as opposed to writing directly to main memory 112), a POSITA would have found this to have been obvious in view of Kirk. Ex1003, ¶214. It would have been efficient to write output data to Local Memory 140 so it can be used in further calculations (*see id.* at 4:16–20) and, thereafter, transfer the data to Host Memory 112 while further calculations occur (*e.g.*, during a second computational cycle).

Data transfer occurs based on instructions received from Kirk’s “controller” (*i.e.*, Graphics Interface 117 along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180). Ex1005, 5:8–11, 6:49–52, 5:60–6:3, 3:64–67, Fig. 12B. For example, data from Local Memory 140 to Host Memory 112 is transferred through Graphics Interface 117. *Id.*, 11:32–41, Fig. 12A. As in the ’438 patent, the “controller” of Kirk/Oh is part of the “accelerator,” which provides accelerated, parallel data processing, separately from the CPU. Ex1003, ¶213; Ex1001, 3:40 (referring to a “GPU accelerator”). Furthermore, in the alternative mapping in which Memory Controller 120 is part of the “controller,” and under Patent Owner’s theories of infringement, Memory Controller 120 transfers output data to Host Memory 112, through Graphics Interface 117. *Id.*, 11:32–41; Ex1003, ¶213.

Further, it would have been obvious to transfer output data in accelerator memory from one or more passes through the artificial neural network to main memory while intermediate computations occur in other threads or on other input data. Ex1003, ¶216. The objective of Oh is to provide text detection in real-time video. Ex1007, 3–5. Unless there are overlapping computations and data transfer back to main memory, the system in Kirk/Oh, would not be able to achieve its intended objective. Ex1003, ¶216. To meet this goal, a POSITA would have designed the system to transfer output data from Graphics Subsystem 107 to main memory while computations continue on later-received data. Ex1003, ¶217. Indeed, because accelerator memory is more closely coupled to the processing engines and outputs can be used as inputs in subsequent time steps, it would have been obvious for the engines to first write outputs to accelerator memory and then transfer them to the main memory, which occurs while further intermediate computations in the artificial neural network occur. Ex1005, 4:16–20; Ex1007, 2–3; Ex1003, ¶217.

b. Claim 2

Kirk/Oh discloses this limitation. Ex1003, ¶¶218–19.

Kirk discloses “[a]n application programming interface for a programmable graphics processor” on the CPU, including allowing the data for processing to be “generated . . . by a human, or the like.” Ex1005, 1:40–42, 6:30–33, 18:60–19:67, Figs. 10A–10C. Just like the user interaction of the ’438 patent (Ex1001, 8:10–46),

an API like that disclosed in Kirk allows setting and editing of the parameters and configurations for the computations and triggering the system to acquire the inputs. Ex1005, 6:30–33, 18:60–19:3; Ex1003, ¶219. At a minimum, because an API is disclosed in Kirk, it would have been obvious for a POSITA to have used an API that, through user interaction, to allows the user to trigger the CPU to receive input data—so the data can be received and sent to the GPU for processing. *Id.* Systems such as those in Kirk require a triggering event to begin the input and processing of data, and it would have been an obvious design choice for a POSITA to implement the trigger as a user-interactive GUI or command interface. *Id.*

c. Claim 3

i. 3[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶220–21.

In Kirk/Oh, one of the possible inputs is “video data.” Ex1007, 3. Video can be input at many different rates, such as 30 frames per second, and at different resolutions. Ex1003, ¶221. The CPU in Kirk/Oh receives the video input at a first rate (*e.g.*, up to 30 fps or any other known video rate). *Id.* For example, if the input data is uncompressed, each frame may be around 1 MB in size, resulting in an input data rate of around 30 MB/s, though other rates of input would be possible and known to those skilled in the art depending on specific needs and capabilities of the camera and system. *Id.*, ¶221.

ii. 3[b]

Kirk/Oh discloses this limitation. Ex1003, ¶¶222–23.

Oh’s text detection can be implemented using different $M \times M$ pixel regions for processing. Ex1007, 3. Depending on the size of the region, the rate of computations in the neural network of Oh will differ. Ex1003, ¶222. Thus, in Kirk/Oh, different rates of computation are disclosed, depending on the precise algorithm implemented. *Id.* For example, Oh discloses an example in which the total processing time in the artificial neural network on a GPU is 0.061 seconds (excluding the time it took to create a texture). Ex1007, 4. This speed equates to about 16 sequences of operations per second. Ex1003, ¶222. Thus, Kirk/Oh discloses that the GPU of Kirk performs the sequence of computations at a different rate than the rate of data input, *i.e.*, a second rate. *Id.*, ¶223.

d. Claim 4

Kirk/Oh discloses this limitation. Ex1003, ¶¶224–225.

See 1[d][iv]. Kirk discloses that “[g]raphics memory is any memory used to store graphics data or program instructions to be executed by Programmable Graphics Processor 105. Graphics memory may include portions of Host Memory 112, [or] Local Memory 140 directly coupled to Programmable Graphics Processor 105.” Ex1005, 3:39-46. Kirk thus discloses storing a copy of the output data in either or both of Host Memory 112 (*i.e.*, main memory) or Local Memory 140 in

Graphics Subsystem 107, depending on user preference and programming; indeed, it would have been obvious to a POSITA to store a copy of the output data in both memories. Ex1003, ¶225.

e. Claim 5

Kirk/Oh discloses this limitation. Ex1003, ¶¶226–27.

See 1[c][i]. Oh discloses implementing an “NN” (*i.e.*, neural network) on a GPU to classify input data. Ex1007, 1, 3. Oh’s artificial neural network includes an input layer, multiple hidden layers (with up to 30 nodes or “neurons” each), and an output layer. *Id.*, 3. At each layer, the neural network performs computations. *Id.*, 2–3. The intermediate results from each layer of computations are fed into the subsequent layer. *Id.* The outputs of the computations at each layer represent outputs of a neuron. *Id.*; Ex1003, ¶227.

f. Claim 6

i. 6[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶228–30.

In Kirk/Oh, the system includes a first memory bank (*e.g.*, a logical portion of Local Memory 140) to store parameters (*e.g.*, weights and equations for the artificial neural network) common to all of the computations in the sequence of computations. For example, memory (*i.e.*, Local Memory 140) stores “graphics data,” which includes “any data that is input to or output from computation units.”

Ex1005, 3:32–38, 3:50–58. In Kirk/Oh, the “graphics data” would include any parameters common to all of the computations in the sequence of computations, such as the weights to be used in the neural network of Oh. Ex1003, ¶229. In an artificial neural network, certain data is common to all computations in a series of computations. *Id.* Kirk further teaches data being stored within “buffers” (*i.e.*, individual memory banks) within “graphics memory.” Ex1005, 1:26–33, 5:2–11, Fig. 3B. Thus, the portion of Local Memory 140 (or buffer) storing this data would comprise a “first memory bank.” Ex1003, ¶¶229–30.

ii. 6[b]

Kirk/Oh discloses this limitation. Ex1003, ¶¶231–33.

See 1[d][iv]. In Kirk/Oh, the system includes a second memory bank (*e.g.*, a logical portion of Local Memory 140) to store data specific to at least one computation in the sequence of computations (*e.g.*, input data, such as a portion of a video frame). For example, in Kirk, “[g]raphics memory is any memory used to store graphics data or program instructions to be executed by Programmable Graphics Processor 105. Graphics memory may include portions of Host Memory 112, [or] Local Memory 140 directly coupled to Programmable Graphics Processor 105.” Ex1005, 3:39–46. When implementing the neural network of Oh in Kirk, the “graphics data” stored in Local Memory 140 would include input data (such as video data) for the neural network. This input data is specific to at least one computation

in the sequence of computations. Ex1007, 2–4; Ex1003, ¶232. As explained for 6[a], Kirk teaches data being stored within “buffers” within “graphics memory.” Ex1005, 1:26–33, 5:2–11, Fig. 3B. The portions of Local Memory 140 for the data described above would each be a “second memory bank.” Ex1003, ¶233.

g. Claim 7

Kirk/Oh discloses this limitation. Ex1003, ¶¶234–38.

See 1[d][iv]. In Kirk/Oh, “intermediate results” (e.g., the output of the first or hidden layers of the artificial neural network) are not transferred back to the system memory. Rather, as explained above, only the outputs of the final layer of the artificial neural network are transferred to main memory.

The controller of Kirk (e.g., Graphics Interface 117 along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180, and alternatively with Memory Controller 130) is configured to transfer outputs from computations from accelerator memory (e.g., Local Memory 140) to main memory (e.g., Host Memory 112). Ex1005, 5:8–11, 6:49–52, 5:60–6:3, 11:32–41, Fig. 12A. It was desirable to avoid unnecessary data exchange between Graphics Subsystem 107 and Host Computer 110. Ex1003, ¶¶235–36. For example, intermediate results from a multi-layer neural network (e.g., outputs of a particular middle layer) are not used by the host system in the Kirk/Oh host system, and therefore transferring such intermediate results would be undesirable. *Id.*; Ex1007, 3–5. Thus, in Kirk/Oh,

output data (such as the result of a pass through the neural network) would be transferred from accelerator memory back to the main memory without transferring intermediate results . Ex1003, ¶236. Not transferring intermediate results from a neural network back to main memory would reduce data transfer via the bus. *Id.*

Additionally, it would have been obvious to transfer only final outputs to main memory. *Id.*, ¶238. It would not benefit the object detection of Oh to send intermediate results (*e.g.*, outputs from intermediate layers) to main memory, and a POSITA would have been motivated to avoid unnecessary data transfer. *Id.*

h. Claim 8

Kirk/Oh discloses this limitation. Ex1003, ¶¶239–43.

See 1[d][iv]. In Kirk/Oh, where the artificial neural network is processing video data with multiple frames, the transfer of output data to main memory occurs after the GPU (*e.g.*, Programmable Graphics Processing Pipeline 1250 or Programmable Graphics Processor 105) has begun to perform another sequence of computations of the artificial neural network. Ex1003, ¶240. The controller of Kirk is configured to transfer outputs from computations from accelerator memory (*e.g.*, Local Memory 140) to main memory (*e.g.*, Host Memory 112). Ex1005, 5:8–11, 6:49–52, 5:60–6:3, 11:32–41, Fig. 12A. For processing of video data, where the output data is from the output layer of the artificial neural network, it is sent to system memory after the processing pipeline begins performing another sequence of

computations (*e.g.*, the computations in the layers of the multi-layer neural network) on new input data. Ex1007, 2–4; Ex1003, ¶¶241–42.

To the extent not disclosed in Kirk/Oh, it would have been obvious that the system could be designed to perform a sequence of computations on newly input data (*e.g.*, a new video frame) while the results from computations on a prior frame are transferred to main memory. *Id.*, ¶243. Doing so would allow the system to keep up with the video data input and allow the system to immediately utilize the output data to display results to a system user. *Id.*

i. Claim 9

Kirk/Oh discloses this limitation. Ex1003, ¶¶244–46.

See 1[d][iii], 1[d][iv]; Claim 8. The controller of Kirk (*e.g.*, Graphics Interface 117, along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180, and alternatively with Memory Controller 130) is configured to initiate transfer of input data to Programmable Graphics Processing Pipeline 1250 (*see* 1[d][iii]) and to transfer outputs from computations from accelerator memory to main memory (*see* 1[d][iv]). In Kirk/Oh, where the artificial neural network is processing video data with multiple frames (including potentially real-time data), the transfer of input and output data referenced above occur in parallel with the GPU (*e.g.*, Programmable Graphics Processing Pipeline 1250 or Programmable Graphics Processor 105) performing another sequence of

computations of the artificial neural network. For example, as explained for 1[d][iii], data is continually transferred to the GPU to identify text in video frames. Thus, as input data is transferred to the GPU, the GPU continues to perform computations in another sequence of computations on the new input data and outputs from the sequences of computations are transferred to memory. Ex1007, 2–4; Ex1003, ¶246. Doing so allows the GPU to perform its intended function of locating text in video. *Id.*

j. Claim 10

Kirk/Oh discloses this limitation. Ex1003, ¶¶247–48.

See 1[d][i], 1[d][ii]. Kirk discloses a “controller” (*e.g.*, Graphics Interface 117, along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180, and alternatively with Memory Controller 130), which is operably coupled to the accelerator memory (*e.g.*, Local Memory 140) through Memory Controller 120. Ex1005, 3:17–67. The controller of Kirk controls execution of the sequence of computations by sending the instructions to Programmable Graphics Processing Pipeline 1250 for execution. *Id.*, 26:24–39; Ex1001, 2:36–37; Ex1003, ¶248.

k. Claim 11

Kirk/Oh discloses this limitation. Ex1003, ¶¶249–52.

See 1[a]. Oh discloses “video data” (which can be real-time data) as input. Ex1007, 1, 3. A POSITA would have understood that video data can be captured by a video camera connected to Host Computer 110, which includes Host Processor 114, to acquire the input data in real time. Ex1003, ¶251. Where the video input is acquired via a camera connected to Host Computer 110 and then transferred to the GPU, the video camera is operably coupled to the CPU in the host system. *Id.* Additionally, it would have been obvious to use a video camera to capture video in which text must be recognized. *Id.*, ¶252. An obvious application of the text detection algorithm of Oh is in a real-time system, with a camera. *Id.* In that combination, the video camera would be coupled to the CPU in order to obtain the video input. *Id.*

i. Claim 12

i. 12[pre]

Kirk/Oh discloses this limitation. Ex1003, ¶¶253–54.

See 1[a] (central processing unit), 1[b] (main memory), 1[c] (accelerator), 1[c][i] (GPU performing computations representing an artificial neural network), 1[c][ii] (accelerator memory).

ii. 12[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶255–56.

See 1[c][i]. As discussed above, in Kirk/Oh, the sequence of computations represents the layers of an artificial neural network, which is implemented on a GPU to classify input data. Ex1007, 1, 3. For example, in Oh's input and hidden layers, the artificial neural network performs computations, (*i.e.*, intermediate computations) to determine the output of that layer (*i.e.*, an intermediate result). *Id.*, 2–3. The input and hidden layers of the artificial neural network are each a “first layer of the artificial neural network” as claimed. A portion of the output of this “first layer” of the artificial neural network represents an output of a neuron because each node in each layer of the artificial neural network represents a neuron. Ex1003, ¶256.

iii. 12[a][i]

Kirk/Oh discloses this limitation. Ex1003, ¶¶257–59.

In Kirk/Oh, data (both inputs and outputs) are stored in “textures.” Ex1003, ¶258. Oh discloses storing output data in a “texture,” where “[t]he result of the previous layer is saved in the form of a render target texture, which is then used as an input for the next layer.” Ex1007, 3. This technique, known at the time of the '438 patent as “Render to Texture,” was a well-known function through which outputs of computations are stored in a “texture” (*i.e.*, a data array) on the GPU that can be used in subsequent computations. Ex1003, ¶258. In Oh, outputs from one computational element (*e.g.*, a sequence of computations representing the first layer

of the artificial neural network) is stored in a “first texture,” and outputs from another computational element (*e.g.*, a sequence of computations representing a second layer of the artificial neural network) is stored in a “second texture.” Ex1007, 2–3. Each texture is assigned an output variable (*e.g.*, a result of the computations performed by a neuron at that layer). Ex1003, ¶259.

iv. 12[a][ii]

Kirk/Oh discloses this limitation. Ex1003, ¶¶260–61.

See 12[a][i]. For the computations in Kirk/Oh, in a first time step, the GPU executes the computations in the first layer and stores (accumulates) the result (*i.e.*, a first value for the output variable) in a first texture, as is done in “Render to Texture” functionality. Ex1007, 2–3; Ex1003, ¶261. The first texture is then passed as the input for computations at the next layer, where the result is stored in a second texture. Ex1007, 2–3.

v. 12[b]

Kirk/Oh (and optionally Tamura) discloses this limitation. Ex1003, ¶¶262–63.

See 1[d][iii]; Claim 9. In Kirk/Oh (and in Tamura), where the artificial neural network is processing real-time data, the transfer of input data and output data over the bus occurs in parallel with the GPU performing a sequence of computations of the artificial neural network. As explained for 1[d][iii], data (*e.g.*, video frames) is

continually transferred to the GPU. Thus, as a second portion of the input data (*i.e.*, additional frames) is transferred into the GPU from main memory, the GPU continues to perform computations on prior input data (*i.e.*, prior video frames). Ex1007, 2–4; Ex1003, ¶263.

vi. 12[c]

Kirk/Oh discloses this limitation. Ex1003, ¶¶264–65.

See 1[d][iv]; Claim 8. The accelerator of Kirk transfers outputs of computations from accelerator memory to main memory via a bus. Ex1005, Fig. 12A. In Kirk/Oh, where the artificial neural network is processing video data, the transfer to main memory of the output layer of the computations of one pass of the artificial neural network occurs in parallel with the GPU performing computations in another pass of the artificial neural network. For example, where the “second portion of the output data” is from the output layer of the artificial neural network in Oh (*i.e.*, “an output of a neuron in a second layer in the artificial neural network”), it is sent to system memory in parallel with the processing engines (GPU) performing another sequence of computations (*i.e.*, the computations in the layers of the multi-layer neural network) on new input data. Ex1007, 2–4; Ex1003, ¶265.

vii. 12[d]

Kirk/Oh discloses this limitation. Ex1003, ¶¶266–67.

See 12[a], 12[b]. In Kirk/Oh, the GPU performs computations on the second portion of the input data (*e.g.*, one or more additional video frames) received while the first portion of the input data (*e.g.*, one or more prior received video frames) is being processed in the artificial neural network of the GPU in the same way the computations are performed on the first portion of the input data for 12[a].

viii. 12[d][i]

Kirk/Oh discloses this limitation. Ex1003, ¶¶268–69.

See 12[a][i], 12[a][ii]. For the computations in the artificial neural network of Oh, in a first-time step, the GPU first executes the computations in the first layer and stores the result (*i.e.*, a first value for the output variable) in the first texture. Ex1007, 2–3. Then, in a second-time step (*i.e.*, after the first output is determined), the first texture is passed as the input for computations at a second layer, and the resulting value (*i.e.*, “second value”) is stored (*i.e.*, accumulated) in the second texture. *Id.*; Ex1003, ¶269.

ix. 12[d][ii]

Kirk/Oh discloses this limitation. Ex1003, ¶¶270–72.

See 1[c][i], 12[d][i]. In Kirk/Oh, the first value of the output variable in the first texture is accessible to other computational elements (including the computations in another layer of Oh’s neural network) during the second time step. Ex1007, 2–4. Kirk discloses that “[a] graphics program . . . is executed within one

or more Execution Pipelines 1240 as a plurality of threads where each vertex or fragment to be processed by the program is assigned to a thread.” Ex1005, 27:40–43. Results from the computations are sent to memory, such as Local Memory 140, Host Memory 112, or memory within the programmable graphics processing pipeline. *Id.*, 3:35–46. Data produced by the processing pipeline “may be written to a buffer in graphics memory to be read from during a subsequent pass.” *Id.*, 4:16–20. The data arrays in Kirk in which input and output data are stored are “textures.” Ex1003, ¶271. Relatedly, Oh discloses storing output data in a “texture,” where “[t]he result of the previous layer is saved in the form of a render target texture, which is then used as an input for the next layer.” Ex1007, 3. Thus, in Kirk/Oh, the first value of the output variable in the first texture is accessible to other computational elements (including the computations in the second layer of Oh’s neural network) during the second time step. Ex1007, 2–4.; Ex1003, ¶272.

m. Claim 13

Kirk/Oh discloses this limitation. Ex1003, ¶¶273–74.

Kirk includes main memory that stores input data received by the CPU. *See* 1[b]. For the same reasons the CPU is configured to receive input data in response to a user interaction (*see* Claim 2), Kirk is designed to store input data in main memory in response to a user interaction (or at a minimum it would have been

obvious to do so for the same reasons explained above). Ex1005, 6:30–33, 18:60–19:3.

n. Claim 14

i. 14[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶275–76.

See 3[a].

ii. 14[b]

Kirk/Oh discloses this limitation. Ex1003, ¶¶277–78.

See 3[b].

o. Claim 16

Kirk/Oh discloses this limitation. Ex1003, ¶¶279–80.

See Claim 7. In Kirk/Oh, the second portion of the output data (*e.g.*, the result of a pass through the artificial neural network) is transferred from accelerator memory to main memory (*e.g.*, Host Memory 112) without transferring any of the intermediate results of the plurality of sequential computations (such as the output of the sequence of computations in a first layer of the artificial neural network) to the main memory. Ex1005, 5:8–11, 6:49–52, 5:60–6:3, 11:32–41, Fig. 12A; Ex1007, 3–5; Ex1003, ¶280.

p. Claim 17

Kirk/Oh discloses this limitation. Ex1003, ¶¶281–82.

See Claim 8. In Kirk/Oh, the second portion of the output data (*e.g.*, the result of the output layer of the artificial neural network) is transferred from accelerator memory to main memory (*e.g.*, Host Memory 112) after the GPU begins to perform another sequence of computations (*e.g.*, another pass through the artificial neural network). Ex1005, 5:8–11, 6:49–52, 5:60–6:3, 11:32–41, Fig. 12A; Ex1007, 3–4; Ex1003, ¶282.

q. Claim 18

Kirk/Oh discloses this limitation. Ex1003, ¶¶283–84.

See Claim 9. In Kirk/Oh, where the artificial neural network on the GPU is processing video data, the transfer of output data (including “the second portion of the output data,” *see* 12[c]) occurs in parallel with the GPU performing another sequence of computations of the artificial neural network. Ex1007, 1, 3–4.

r. Claim 19

Kirk/Oh discloses this limitation. Ex1003, ¶¶285–86.

See Claim 11.

s. Claim 20

i. 20[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶287–88.

See 6[a].

ii. 20[b]

Kirk/Oh discloses this limitation. Ex1003, ¶¶289–90.

See 6[b].

t. Claim 21

i. 21[pre]

If the preamble is limiting, Kirk/Oh discloses it. Ex1003, ¶¶291–92.

See 1[c][i], 12[pre]; Ex1007, 3–4.

ii. 21[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶293–94.

See 1[a]; Claim 11.

iii. 21[b]

Kirk/Oh discloses this limitation. Ex1003, ¶¶295–96.

See 1[c][ii], 1[d], 1[d][i]. Kirk discloses a controller coupled to a GPU (*e.g.*, Programmable Graphics Processing Pipeline 1250 or Programmable Graphics Processor 105) that is configured to initialize textures and shaders in the accelerator memory (*e.g.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250), which is also coupled to the GPU, by sending the data and instructions.

iv. 21[c]

Kirk/Oh discloses this limitation. Ex1003, ¶¶297–98.

See 1[b], 1[c], 1[d][iii]. Kirk discloses memory coupled to the GPU (*e.g.*, Local Memory 140), which receives first input data (*e.g.*, data from a video camera) that was received by the CPU (*e.g.*, Host Processor 114).

v. 21[d]

Kirk/Oh discloses this limitation. Ex1003, ¶¶299–300.

See 1[c][i], 12[a]. The computations in the artificial neural network of Oh are based on the textures (data arrays) and shaders (GPU programs) previously initialized. The computations in the artificial neural network represent layer of neurons. Ex1003, ¶300. For example, the “first output data” is the result of a first computations in a layer of the artificial neural network of Oh, or the result of a first pass through the layers of the artificial neural network. *Id.*

vi. 21[e]

Kirk/Oh discloses this limitation. Ex1003, ¶¶301–02.

See 1[c][ii], 1[d][iii], 12[a]. Kirk discloses memory coupled to the GPU (*e.g.*, Local Memory 140), which stores first input data (*e.g.*, data from a video camera) and first output data (*e.g.*, results of the computations described above).

vii. 21[f]

Kirk/Oh discloses this limitation. Ex1003, ¶¶303–04.

See 1[d][iii], 1[d][iv], 12[b], 12[c]; Claim 11. In Kirk/Oh, second input data (*e.g.*, additional video data) is transferred to memory in the accelerator (*e.g.*, Local

Memory 140) after the GPU starts the first computation (*e.g.*, computations in a layer in a first pass of the artificial neural network) and before the GPU starts a second computation of the sequence of computations (*e.g.*, a subsequent computation in a layer of the artificial neural network, including in a second pass of the artificial neural network). In Oh, the computations represent outputs of neurons (including a “second neuron”) in layers (including a “second layer”) of a multi-layer artificial neural network. Ex1003, ¶304.

u. Claim 22

Kirk/Oh discloses this limitation. Ex1003, ¶¶305–06.

See 1[c], 12[b]. The second input data (*e.g.*, additional data from a video camera) is transferred via a bus from the CPU to the accelerator.

v. Claim 23

Kirk/Oh discloses this limitation. Ex1003, ¶¶307–08.

See 1[d][iv]. In Kirk/Oh, first output data (*e.g.*, the result of the computations in an output layer of the artificial neural network of Oh) is transferred from one memory (*e.g.*, Local Memory 140) to another memory (*e.g.*, Host Memory 112). This transfer occurs while computations continue in the artificial neural network. Ex1003, ¶308.

w. Claim 24

i. 24[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶309–10.

See 1[d][iii], 12[a]; Claim 16. Kirk stores intermediate results in memory. Ex1005, 5:8–11, 6:49–52, 5:60–6:3. Oh also discloses storing intermediate results, such as the result of a single pass through a layer, in a texture (*i.e.*, in memory). Ex1007, 3–4.

ii. 24[b]

Kirk/Oh discloses this limitation. Ex1003, ¶¶311–12.

See Claims 7, 16.

x. Claim 25

Kirk/Oh discloses this limitation. Ex1003, ¶¶313–14.

See 12[b], 12[c]; Claim 9. In Kirk/Oh, where the artificial neural network is processing real-time data, the transfer of input data and output data occur in parallel. Ex1003, ¶314.

y. Claim 26

Kirk/Oh discloses this limitation. Ex1003, ¶¶315–17.

See 6[a]. 6[a] recites a “first memory bank,” while this limitation recites a “first memory partition.” The “first memory bank” described for 6[a]—*e.g.*, logical portions of Local Memory 140 storing the common parameters—is a “first memory

partition.” Ex1003, ¶316. Kirk teaches identifying data by an “address for a location in graphics memory.” Ex1005, 8:3–7, 9:41–55. Figures 3A and 3B, for example, illustrate storing data in buffers, such as “Buffer 320.” *Id.*, 5:1–3, 9:32–55, Figs. 3A, 3B. The dedicated memory block of Local Memory 140 in which the parameters common to all computations in a sequence of computations is stored (*i.e.*, one or more buffers) is a “first partition.” Ex1003, ¶316.

z. Claim 27

Kirk/Oh discloses this limitation. Ex1003, ¶¶318–20.

See 6[b]; Claim 26. 6[b] recites a “second memory bank,” while this limitation recites a “second memory partition.” For the same reasons the “first memory bank” above is a “first memory partition,” the “second memory bank” of 6[b] is a “second memory partition.” Ex1003, ¶319; 6[a].

aa. Claim 28

Kirk/Oh discloses this limitation. Ex1003, ¶¶321–24.

In Kirk/Oh, the second memory partition in the GPU (*e.g.*, a partition in Local Memory 140), including the same logical portion storing “data specific to the first computation” in Claim 27, can store external input data patterns, representations of internal variables, an input of the computation in the sequence of computations, and the output of the computation in the sequence of computations. Ex1005, 3:32–46, 4:57–5:15, 5:33–36, 6:49–55; Ex1003, ¶322.

For example, in Kirk/Oh, Local Memory 140 can store external data input patterns, such as the input from an external camera. Ex1005, 3:32–39, 5:1–8; Ex1007, 1, 3–4; Ex1001, 5:58–60 (“partition ... to hold the external input data patterns”). The same memory, in the same partition, can also store representations of internal variables, such as outputs (stored in variables) of a hidden layer in Oh’s multi-layer neural network, which are used as an input for the next layer. Ex1005, 4:16–20; Ex1007, 2–3; Ex1001, 5:60–62 (“partition ... to hold the data textures representing internal variables”). As explained for 1[c][ii] and 1[d][iii], GPU memory (including Local Memory 140) also stores inputs and outputs of the computations. Kirk is flexible in where data is and can be stored, including in buffers in Local Memory 140, so it can be accessed by multiple threads. Ex1005, 3:32–46, 4:57–5:15, 5:33–36, 6:49–55, 26:24–39; Ex1003, ¶323. As explained with respect to Claim 26, Kirk teaches that data is stored in specific buffers (partitions), and the system allows logically allocating memory to hold specific data for specific processing engines, separate from other data. *Id.* As such, Kirk discloses that each of these elements can be stored in a “second memory partition.” *Id.*

Additionally, it would have been obvious to store each of the claimed parameters in Claim 28 in the “second memory partition.” Kirk discloses memory in the GPU in which data and other parameters can be stored (*e.g.*, Local Memory 140), and a POSITA would have been able to and motivated to use Local Memory

140 to store data in partitions (*e.g.*, buffers). Ex1005, 3:32–46, 4:16–20, 4:57–5:45; Ex1003, ¶324. Because each of these pieces of data are used or output by the same thread in an artificial neural network, a POSITA would have been motivated to store them in a single memory partition, *i.e.*, the claimed second memory partition. *Id.* It thus would have been obvious to try storing these parameters in the “second memory partition.” *Id.*

bb. Claim 29

Kirk/Oh discloses this limitation. Ex1003, ¶¶325–26.

See 12[a][i], 12[a][ii]. In Kirk/Oh, the GPU first executes the computations in a first layer (*i.e.*, the “first computation in the sequence of computations”) and stores the result (*i.e.*, an output of the computational elements) in memory. Ex1007, 2–4; Ex1003, ¶326.

cc. Claim 30

i. 30[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶327–28.

See 12[a][i], 12[a][ii]. In Kirk/Oh, in one time step, the GPU first executes the computations in a layer of the artificial neural network and stores the result (*i.e.*, output) in the memory. Ex1007, 2–4; Ex1003, ¶328.

ii. 30[b]

Kirk/Oh discloses this limitation. Ex1003, ¶¶329–30.

See 12[d][i], 12[d][ii]. In Kirk/Oh, after a first output is stored in memory, the first output is passed as an input for computations in a subsequent layer. Ex1007, 2–3. Output data from one computational cycle can become input data for a later computational cycle. Ex1005, 4:16–20, 8:23–25. The GPU therefore accesses the outputs of previous computations in the sequence of computations. Ex1003, ¶330.

dd. Claim 31

Kirk/Oh discloses this limitation. Ex1003, ¶¶331–32.

See 1[c][i], 12[a][i]. Oh’s multi-layer neural network includes an input layer with nodes (*i.e.*, neurons), multiple hidden layers with 30 nodes each, and an output layer with a single node, with each layer representing a layer of neurons. Ex1007, 3. The layers of neurons are represented by a plurality of computational elements (*e.g.*, computations performed at each neuron). Ex1003, ¶332.

ee. Claim 32

Kirk/Oh discloses this limitation. Ex1003, ¶¶333–34.

In Kirk/Oh, all neurons in a layer of the artificial neural network are described by the same equation—*e.g.*, the same matrix multiplication and sigmoid functions—and execute the same program. Ex1007, 2–3; Ex1003, ¶334.

ff. Claim 33

Kirk/Oh discloses this limitation. Ex1003, ¶¶335–36.

See 1[a], Claim 11. The input data from the camera is both the “first input data” and “second input data,” with the “second input data” being later frames of the video. Ex1003, ¶336.

gg. Claim 34

Kirk/Oh discloses this limitation. Ex1003, ¶¶413–14.

Oh discloses inputting “video data” into its neural network. Ex1007, 2–4. Where the input is not live video from a camera, the “video data” would be loaded from a disk in Kirk/Oh. During training of the artificial neural network of Oh, input video data would have been loaded from disk (*e.g.*, a hard drive) coupled to Host Computer 110. *Id.*; Ex1005, 1:14–33, 3:12–46, 22:1–3, 26:24–39. Additionally, it would have been obvious to load input data from disk. Before the artificial neural network in Oh would be able to identify text in video frames, it would be trained on existing data. Ex1006, 624–25. It would have been obvious and within the ordinary skill of a POSITA to store the training data on a disk drive coupled to the system and then load it into the accelerator.

hh. Claim 40

i. 40[pre]

If the preamble is limiting, Kirk/Oh discloses it. Ex1003, ¶¶337–38.

See 21[pre].

ii. 40[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶339–40.

See 1[a]; Claim 11.

iii. 40[b]

Kirk/Oh discloses this limitation. Ex1003, ¶¶341–44.

See 1[c][ii]. The “memory” of this limitation is the “accelerator memory” of Claim 1. The “first input data” is stored in a first partition of the accelerator memory and is referenced by a “first pointer.” Kirk teaches identifying data by an “address for a location in graphics memory.” Ex1005, 8:3–7, 9:41–55. Figures 3A and 3B, for example, illustrate storing data in buffers, such as “Buffer 320,” which are accessed using a variable holding an “address”—*i.e.* a “pointer.” *Id.*, 9:32–55, Figs. 3A, 3B; Ex1003, ¶342; Ex1001, 6:16–18 (referring to “address pointers”), 11:38–40 (referring to an “ID” of a texture (array) as a “pointer”). To be used for the computations of the artificial neural network, the input data to start the computation is transferred to the first partition before computing a first layer. Ex1003, ¶342.

Additionally, referencing memory by using pointers would have been obvious. Prior to the ’438 patent, pointers were a fundamental construct of computer programming to efficiently manage memory and data access. Ex1003, ¶343; Ex1009, 88; Ex1032, 499–508.

GPU Gems teaches storing first input data in a first partition, referenced by a first pointer, before computations begin. Ex1003, ¶344. GPU Gems explains that “[a]rrays of data in the framework are called buffers,” and that buffers are identified by a “pointer.” Ex1032, 504. A “buffer” is a “partition” because it a logical allocation of memory to hold specific data separate from other data. Ex1003, ¶344. In one example in GPU Gems, before any computational cycles begin, the function `pgInitBuffer()` transfers the input data in `array` to a buffer (partition) that is referenced by a pointer (*i.e.*, `rdBuffer`):

Listing 31-2 (continued). C++ Code to Set Up and Run a Reaction-Diffusion Simulation on the GPU

```
// Initialize the state of the simulation with values in array
pgInitBuffer(rdBuffer, array);
}

void update_rd(){ // Call this every iteration
    pgBindStream(rdProgram, "concentration", rdBuffer, currentSource);
    PUGRect range(0, width, 0, height);
    pgRunProgram(rdProgram, rdBuffer, range, currentTarget);

    std::swap(currentSource, currentTarget);
}
```

Ex1006, 504, 507. By using pointers, developers can directly access and manipulate specific memory locations, allowing for efficient data retrieval and modification without the need for copying large data sets. Ex1003, ¶344. Given the widespread use of pointers in managing data, their application to the parallel GPU processing in Kirk/Oh would have been a routine choice for skilled practitioners in the field. *Id.*

iv. 40[c]

Kirk/Oh discloses this limitation. Ex1003, ¶¶345–46.

See 1[c][i], 21[d].

v. 40[d]

Kirk/Oh discloses this limitation. Ex1003, ¶¶347–48.

See 1[d].

vi. 40[d][i]

Kirk/Oh discloses this limitation. Ex1003, ¶¶349–53.

See 12[a][i], 21[d], 40[b]. Output data from the processing unit is stored in a partition of memory (e.g., a partition in Local Memory 140). As explained with respect to 40[b], the partition of memory would be referenced by a pointer (or it would have been obvious to do so).

A POSITA would have understood Oh to leverage “Render to Texture” functionality to where “[t]he result of the previous layer is saved in the form of a render target texture, which is then used as an input for the next layer.” Ex1007, 3. Thus, in Kirk/Oh, the first output data from a layer of the artificial neural network becomes an input for a second layer of neurons. Ex1007, 2–4; Ex1005, 4:16–20. This would have been achieved by swapping pointers. Ex1003, ¶¶350–51.

It would have been obvious to implement an algorithm in which pointers are swapped at the end of a computational cycle, including in view of GPU Gems. Because Kirk and Oh both describe using intermediate results as inputs to a new computational cycle, and because both disclose using pointers, a POSITA would have used pointer swapping to effectuate this efficiently. Ex1003, ¶352. Pointer

swapping was taught throughout the prior art, and thus would have been obvious to implement and a POSITA would have had a reasonable expectation of success in doing so. *Id.*; Ex1032, 44, 508, 713. GPU Gems describes a numerical simulation on a GPU that swaps pointers to input and output data. The simulation code in GPU Gems “[c]reates a ‘double buffered’ PUGBuffer,” which “allow[s] the simulation to alternate using one as input, one as output.” Ex1006, 507. At the end of each iteration, the code calls the “swap ()” function, which swaps the pointers for the currentSource buffer with that of the currentTarget buffer, thus allowing the output of one computational cycle to become the input for the next computational cycle.

Listing 31-2 (continued). C++ Code to Set Up and Run a Reaction-Diffusion Simulation on the GPU

```
    // Initialize the state of the simulation with values in array
    pugInitBuffer(rdBuffer, array);
}

void update_rd(){ // Call this every iteration
    pugBindStream(rdProgram, "concentration", rdBuffer, currentSource);
    PUGRect range(0, width, 0, height);
    pugRunProgram(rdProgram, rdBuffer, range, currentTarget);

    std::swap(currentSource, currentTarget);
}
```

Id., 508 (annotated). A POSITA implementing Oh’s neural network on Kirk would be motivated to include this known technique, thereby increasing efficiency, and would have had a reasonable expectation of success in doing so because it had already been implemented in GPGPU computations. Ex1003, ¶353.

vii. 40[d][ii]

Kirk/Oh discloses this limitation. Ex1003, ¶¶354–55.

See 1[d][iii], 1[d][iv], 12[c]. In Kirk/Oh, where the artificial neural network is processing real-time data, the transfer to main memory of the final output of the computations of one pass or layer of the artificial neural network occurs after the processing engine has begun to perform computations in another pass or layer of the artificial neural network (*e.g.*, a “second layer of neurons”).

viii. 40[d][iii]

Kirk/Oh discloses this limitation. Ex1003, ¶¶356–58.

The controller of Kirk dictates an order of execution of instructions to Programmable Graphics Processing Pipeline 1250. Ex1005, 26:24–39. For example, “Front End 120 interprets and formats the commands and outputs the formatted commands and data to an Index Processor 1235,” which then get passed to Programmable Graphics Processing Pipeline 1250. *Id.*, 26:29–31. A POSITA would have understood that, by so doing, the front end is dictating the order of execution of the instructions. Ex1003, ¶357. Alternatively, to the extent Memory Controller 120 is considered the “controller,” a POSITA would have understood the memory controller to dictate the order of execution of instructions by passing instructions to Programmable Graphics Processing Pipeline 1250 in a specific order. Ex1005, 26:35–39; Ex1003, ¶358. In Kirk/Oh, these features mean that the

controller “dictate[s] an order of execution of instructions” for the computation on the first layer of neurons because these components determine which instructions should be selected and issued for execution. *Id.* The instructions to the processing engines include instructions to perform computations in all layers of neurons in the artificial neural network. *Id.*; Ex1007, 2–4.

ii. Claim 41

Kirk/Oh discloses this limitation. Ex1003, ¶¶359–60.

See 1[c][i].

jj. Claim 42

Kirk/Oh discloses this limitation. Ex1003, ¶¶361–62.

See 1[d][ii]. The controller sends the data and instructions to Programmable Graphics Processing Pipeline 1250 in Programmable Graphics Processor 105 for execution. Ex1005, 3:50–54, 4:21–42. Programmable Graphics Processing Pipeline 1250 (*i.e.*, a graphics processing unit) performs calculations. Ex1003, ¶362.

kk. Claim 43

i. 43[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶363–65.

See 6[a]; Claim 28 (describing storing specific information in “partitions” in Kirk/Oh). Kirk discloses a “third partition” (*e.g.*, a logical partition in Local Memory 140) for storing “internal variables.” Kirk discloses that the memory in

Graphics Subsystem 107 (*i.e.*, Local Memory 140) stores “graphics data,” which includes “any data that is input to or output from computation units.” Ex1005, 3:32–38, 3:50–58. The memory can also store representations of internal variables, such as outputs of a hidden layer in Oh’s multi-layer neural network, which are used as an input for the next layer. *Id.*, 4:16–20; Ex1007, 2–3. The partition of Local Memory 140 storing these internal variables is a “third partition,” with the first and second partitions having been defined in Claim 40. Ex1003, ¶¶364–65. Additionally, as explained, it would have been obvious to store these parameters in a logical “third partition,” where the data is stored separate from other data, thereby facilitating efficient access to the data. *Id.*

ii. 43[b]

Kirk/Oh discloses this limitation. Ex1003, ¶¶366–68.

See 6[b]; Claim 28. Kirk discloses a “fourth partition” (a logical partition in Local Memory 140) for storing “data used as input at a particular layer of neurons of the artificial neural network.” Kirk discloses that the memory in Graphics Subsystem 107 (*i.e.*, Local Memory 140) stores “graphics data,” which includes “any data that is input to or output from computation units,” and the logical portion storing such information would be a “partition” as explained for Claim 28. Ex1005, 3:32–38, 3:50–58. In Kirk/Oh, the “graphics data” would include the output textures described in Oh, which are used as inputs at a particular layer of the neurons in Oh’s

neural network. Ex1007, 2–3; Ex1003, ¶¶367–68. Additionally and alternatively, Programmable Graphics Processing Pipeline 1250 includes “Pixel Input Buffer” and “Vertex Input Buffer,” which are partitions that store data used as inputs. Ex1005, 26:62–27:11, 28:10–20, Fig. 12B. In Kirk/Oh, these buffers (any one of which would be a “fourth partition”), would store textures used as an input to a layer of neurons in the neural network. Ex1007, 2–3; Ex1003, ¶368. For the same reasons it would have been obvious to store data in a “third partition” (*see* 43[a]), it would have been obvious to store data used as input at a particular layer in a “fourth partition.” Ex1003, ¶368.

II. Claim 44

i. 44[pre]

To the extent the preamble is limiting, Kirk/Oh discloses it. Ex1003, ¶¶369–70.

See 1[pre].

ii. 44[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶371–72.

See 1[a], 21[a]; Claim 11.

iii. 44[b]

Kirk/Oh discloses this limitation. Ex1003, ¶¶373–74.

See 1[b].

iv. 44[c]

Kirk/Oh discloses this limitation. Ex1003, ¶¶375–76.

See 1[c].

v. 44[c][i]

Kirk/Oh discloses this limitation. Ex1003, ¶¶377–78.

See 1[c][i].

vi. 44[c][ii]

Kirk/Oh discloses this limitation. Ex1003, ¶¶379–80.

See 1[c][ii].

vii. 44[d]

Kirk/Oh discloses this limitation. Ex1003, ¶¶381–82.

See 1[d].

viii. 44[d][i]

Kirk/Oh discloses this limitation. Ex1003, ¶¶383–84.

See 1[d][iii].

ix. 44[d][ii]

Kirk/Oh discloses this limitation. Ex1003, ¶¶385–86.

See 1[d][iv].

x. 44[d][iii]

Kirk/Oh discloses this limitation. Ex1003, ¶¶387–88.

See 1[d][ii].

mm. Claim 45

Kirk/Oh discloses this limitation. Ex1003, ¶¶389–90.

See Claim 2.

nn. Claim 46

i. 46[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶391–92.

See 3[a], Claim 44.

ii. 46[b]

Kirk/Oh discloses this limitation. Ex1003, ¶¶393–94.

See 3[b].

oo. Claim 47

Kirk/Oh discloses this limitation. Ex1003, ¶¶395–96.

See Claim 4.

pp. Claim 48

Kirk/Oh discloses this limitation. Ex1003, ¶¶397–98.

See Claim 5.

qq. Claim 49

i. 49[a]

Kirk/Oh discloses this limitation. Ex1003, ¶¶399–400.

See 6[a]; Claims 26, 44.

ii. 49[b]

Kirk/Oh discloses this limitation. Ex1003, ¶¶401–02.

See 6[b]; Claim 27.

rr. Claim 50

Kirk/Oh discloses this limitation. Ex1003, ¶¶403–04.

See Claim 7.

ss. Claim 51

Kirk/Oh discloses this limitation. Ex1003, ¶¶405–06.

See Claim 8.

tt. Claim 52

Kirk/Oh discloses this limitation. Ex1003, ¶¶407–08.

See Claim 9.

uu. Claim 53

Kirk/Oh discloses this limitation. Ex1003, ¶¶409–410.

See Claim 10.

vv. Claim 54

Kirk/Oh discloses this limitation. Ex1003, ¶¶411–12.

See Claim 11.

V. ENABLEMENT AND SECONDARY CONSIDERATIONS

The prior art in this Petition is enabled. Ex1003, ¶¶133, 138, 142, 147.

Petitioner is not aware of any secondary considerations. *Id.*, ¶415. If Patent Owner asserts any secondary considerations, Petitioner may request leave to file a reply. Secondary considerations cannot overcome a strong prima facie case of obviousness. *Ohio Willow Wood Co. v. Alps South, LLC*, 735 F.3d 1333, 1344 (Fed. Cir. 2013).

VI. DISCRETIONARY DENIAL

Petitioner does not address discretionary denial other than to stipulate that, if review is instituted, Petitioner will not pursue in the Litigation the same grounds or any grounds that could have reasonably been raised in this Petition. Ex1031.

VII. CONCLUSION

Petitioner has established a reasonable likelihood the Challenged Claims are unpatentable. Petitioner therefore requests that IPR be instituted.

DATED: April 30, 2025

Respectfully Submitted,

/ Brian M. Buroker /

Brian M. Buroker (Reg. No. 39,125)
GIBSON, DUNN & CRUTCHER LLP
1050 Connecticut Ave. NW
Washington, DC 20036
Phone: (202) 955-8500
Fax: (202) 467-0539
Email: bburoker@gibsondunn.com

Attorney for Petitioner NVIDIA Corporation

CERTIFICATION UNDER 37 C.F.R. § 42.24(d)

Under the provisions of 37 C.F.R. § 42.24(d), the undersigned attorney hereby certifies that the word count for Sections I–VII of the foregoing Petition for *Inter Partes* Review is 13,999 according to the word count tool in Microsoft Word.

DATED: April 30, 2025

Respectfully Submitted,

/ Brian M. Buroker /

Brian M. Buroker (Reg. No. 39,125)
GIBSON, DUNN & CRUTCHER LLP
1050 Connecticut Ave. NW
Washington, DC 20036
Phone: (202) 955-8500
Fax: (202) 467-0539
Email: bburoker@gibsondunn.com

Attorney for Petitioner NVIDIA Corporation

CERTIFICATE OF SERVICE

The undersigned certifies service pursuant to 37 C.F.R. §§ 42.6(e) and 42.105(a), (b) on the Patent Owner via FedEx overnight mail of a copy of this Petition for *Inter Partes* Review, all exhibits and supporting materials, and Power of Attorney at the correspondence address of record for the '438 patent:

SMITH BALUCH LLP
376 BOYLSTON ST
STE 401
BOSTON, MA

A courtesy copy has also been mailed to and served on litigation counsel at:

Max L. Tribble , Jr.
Susman Godfrey L.L.P.
1000 Louisiana
Suite 5100
Houston, TX 77002-5096
mtribble@susmangodfrey.com

DATED: April 30, 2025

Respectfully Submitted,

/ Brian M. Buroker /

Brian M. Buroker (Reg. No. 39,125)
GIBSON, DUNN & CRUTCHER LLP
1050 Connecticut Ave. NW
Washington, DC 20036
Phone: (202) 955-8500
Fax: (202) 467-0539
Email: bburoker@gibsondunn.com

Attorney for Petitioner NVIDIA Corporation

APPENDIX: CHALLENGED CLAIM LISTING

No.	Limitation
1[pre]	A computer system, comprising:
1[a]	a central processing unit to receive input data;
1[b]	main memory, operably coupled to the central processing unit via a bus, to store the input data received by the central processing unit;
1[c]	an accelerator, operably coupled to the central processing unit and the main memory via the bus, to receive at least a portion of the input data from the main memory, the accelerator comprising:
1[c][i]	at least one graphics processing unit to perform a sequence of computations on the at least a portion of the input data so as to generate output data, the sequence of computations representing an artificial neural network, intermediate computations in the sequence of computations representing respective layers of the artificial neural network and yielding intermediate results; and
1[c][ii]	accelerator memory, operably coupled to the at least one graphics processing unit, to store the results of the sequence of computations; and
1[d]	a controller, operably coupled to the at least one graphics processing unit and the accelerator memory,
1[d][i]	to initialize textures and shaders in the accelerator memory for performing the sequence of computations,
1[d][ii]	to control performance of the sequence of computations by the at least one graphics processing unit,
1[d][iii]	to transfer the at least a portion of the input data into the accelerator memory during performance of the intermediate computations in the sequence of computations by the at least one graphics processing unit, and
1[d][iv]	to transfer at least a portion of the output data from the accelerator memory to the main memory during performance of the intermediate computations in the sequence of computations by the at least one graphics processing unit.

No.	Limitation
2	The computer system of claim 1, wherein the central processing unit is configured to receive the input data in response to a user interaction.
3[a]	The computer system of claim 1, wherein: the central processing unit is configured to receive the input data at a first rate; and
3[b]	the at least one graphics processing unit is configured to perform the sequence of computations at a second rate different than the first rate.
4	The computer system of claim 1, wherein the main memory is configured to store a copy of the output data stored in the accelerator memory.
5	The computer system of claim 1, wherein an output of at least one computation in the sequence of computations represents an output of at least one neuron in an artificial neural network.
6[a]	The computer system of claim 1, wherein accelerator memory comprises: a first memory bank to store parameters common to all of the computations in the sequence of computations; and
6[b]	a second memory bank to store data specific to at least one computation in the sequence of computations.
7	The computer system of claim 1, wherein the controller is configured to transfer the output data from the accelerator memory to the main memory without transferring any of the intermediate results from the accelerator memory to the main memory so as to reduce data transfer via the bus.
8	The computer system of claim 1, wherein the controller is configured to transfer at least a portion of the output data from the accelerator memory to the main memory after the at least one graphics processing unit has begun to perform another sequence of computations.

No.	Limitation
9	The computer system of claim 8, wherein the controller is configured to initiate transfer of the at least a portion of the input data and to transfer the at least a portion of the output data in parallel with performance of at least one computation in the other sequence of computations by the at least one graphics processing unit.
10	The computer system of claim 1, wherein the controller is configured to control execution of the sequence of computations by the at least one graphics processing unit.
11	The computer system of claim 1, further comprising: at least one of a video camera, a microphone, or a cell recording electrode, operably coupled to the central processing unit, to acquire the input data in real time.
12[pre]	A method of performing a sequence of computations representing an artificial neural network on a computer system comprising a central processing unit (CPU), a main memory operably coupled to the central processing unit via a bus, an accelerator operably coupled to the CPU and the main memory via the bus, the accelerator comprising a graphics processing unit (GPU) and an accelerator memory, the method comprising:
12[a]	(A) performing, by the GPU, the sequence of computations on a first portion of input data so as to generate a first portion of output data, the first portion of the output data representing an output of a neuron in a first layer of the artificial neural network, intermediate computations in the sequence of computations yielding intermediate results, wherein performing the sequence of computations on the first portion of the input data comprises
12[a][i]	(i) assigning an output variable to a first texture and a second texture, the output variable being included in a first computational element of a plurality of computational elements, the plurality of computational elements representing the sequence of computations and
12[a][ii]	(ii) accumulating a first value for the output variable in the first texture during a first time step;

No.	Limitation
12[b]	(B) in parallel with performing the sequence of computations by the GPU in (A), transferring a second portion of the input data from the main memory to the accelerator via the bus;
12[c]	(C) in parallel with performing the sequence of computations by the GPU in (A), transferring a second portion of the output data from the accelerator memory to the main memory via the bus, the second portion of the output data representing an output of a neuron in a second layer in the artificial neural network; and
12[d]	(D) performing, by the GPU, the sequence of computations on the second portion of the input data, wherein performing the sequence of computations on the second portion of the input data comprises
12[d][i]	(i) accumulating a second value for the output variable in the second texture during a second time step and
12[d][ii]	(ii) making the first value of the output variable in the first texture accessible to other computational elements in the plurality of computational elements during the second time step.
13	The method of claim 12, further comprising: storing the input data in the main memory in response to a user interaction.
14[a]	The method of claim 12, further comprising: receiving the input data at a first rate; and
14[b]	wherein (A) comprises performing the sequence of computations at a second rate different than the first rate.
16	The method of claim 12, wherein (C) comprises: transferring the second portion of the output data from the accelerator memory to the main memory without transferring any of the intermediate results of the plurality of sequential computations from the accelerator memory to the main memory so as to reduce data transfer via the bus.
17	The method of claim 12, wherein (C) comprises: transferring the second portion of the output data from the accelerator memory to the main memory after the GPU has begun to perform another sequence of computations.

No.	Limitation
18	The method of claim 17, wherein (C) further comprises: initiating transfer of the second portion of the output data in parallel with performance of at least one computation in the other sequence of computations.
19	The method of claim 12, further comprising: acquiring the input data in real time with at least one of a video camera, a microphone, or a cell recording electrode operably coupled to the CPU.
20[a]	The method of claim 12, further comprising: storing parameters common to all of the computations in the sequence of computations in a first memory bank in the accelerator memory; and
20[b]	storing data specific to at least one computation in the sequence of computations in a second memory bank in the accelerator memory.
21[pre]	A method of performing a sequence of computations representing an artificial neural network, the method comprising:
21[a]	receiving, at a central processing unit (CPU), first input data acquired from an external system in real time;
21[b]	initializing, by a controller operably coupled to a graphics processing unit (GPU), textures and shaders in a memory operably coupled to the GPU;
21[c]	transferring the first input data received by the CPU to the memory operably coupled to the GPU;
21[d]	performing, by the graphics processing unit (GPU), a first computation in the sequence of computations on the first input data based on the textures and shaders to generate first output data, computations in the sequence of computations representing respective layers of neurons in the artificial neural network, an output of the first computation in the sequence of computations representing an output of a first neuron in a first layer in the artificial neural network;
21[e]	storing, in the memory operably coupled to the GPU, the first input data and the first output data; and

No.	Limitation
21[f]	transferring second input data acquired from the external system in real time into the memory operably coupled to the GPU after the GPU starts the first computation and before the GPU starts a second computation of the sequence of computations, an output of the second computation in the sequence of computations representing an output of a second neuron in a second layer in the artificial neural network.
22	The method of claim 21, wherein transferring the second input data comprises transferring the second input data via a bus operably coupled to the CPU.
23	The method of claim 21, further comprising: transferring the first output data from the memory to another memory during the second computation in the sequence of computations.
24[a]	The method of claim 23, further comprising: storing intermediate results of the sequence of computations in the memory, and
25[b]	wherein transferring the first output data from the memory to the other memory occurs without transferring the intermediate results of the sequence of computations.
25	The method of claim 23, wherein transferring the second input data and transferring the first output data occurs in parallel.
26	The method of claim 21, further comprising: storing, in a first memory partition of the memory, parameters common to all of the computations in the sequence of computations.
27	The method of claim 26, further comprising: storing, in a second memory partition of the memory, data specific to the first computation in the sequence of computations.
28	The method of claim 27, further comprising: storing, in the second memory partition, external input data patterns, representations of internal variables, an input of the computation in the sequence of computations, and the output of the computation in the sequence of computations.

No.	Limitation
29	The method of claim 21, wherein storing the first output data comprises: accumulating, in the memory, outputs of computational elements executed by the GPU in performing the first computation in the sequence of computations.
30[a]	The method of claim 21, further comprising: storing, in the memory, an output of a previous computation in the sequence of computations; and
30[b]	accessing, by the GPU, the output of the previous computation during performance of the computation in the sequence of computations.
31	The method of claim 21, wherein performing the first computation comprises executing a plurality of computational elements representing a layer of neurons in an artificial neural network.
32	The method of claim 31, wherein all neurons in the layer of neurons are described by the same equation.
33	The method of claim 21, further comprising: acquiring the second input data with at least one of a video camera, a microphone, or a cell recording electrode.
34	The method of claim 21, further comprising: loading the second input data from disk.
40[pre]	A system for executing an artificial neural network, the system comprising:
40[a]	a central processing unit (CPU) to provide first input data;
40[b]	a memory, operably coupled to the CPU, to store the first input data in a first partition, referenced by a first pointer, before computing a first layer of neurons of the artificial neural network;
40[c]	a processing unit, operably coupled to the memory, to perform, during computation of the first layer of neurons, at least one calculation on the first input data so as to generate first output data, the first output data representing an output of at least one neuron in the first layer of neurons; and

No.	Limitation
40[d]	a controller, operably coupled to the processing unit and the memory, to:
40[d][i]	store the first output data in a second partition of the memory, the second partition referenced by a second pointer, and to swap the first pointer with the second pointer at the end of the computation of the first layer of neurons, such that the first output data becomes an input for a second layer of neurons of the artificial neural network,
40[d][ii]	transfer the first output data to another memory during computation of the second layer of neurons, and
40[d][iii]	dictate an order of execution of instructions to the processing unit to perform the computation of the first layer of neurons.
41	The system of claim 40, wherein the processing unit comprises a graphics processing unit.
42	The system of claim 40, wherein the controller is configured to send instructions for performing the at least one calculation to the processing unit.
43[a]	The system of claim 40, wherein the memory further comprises: a third partition to store internal variables; and
43[b]	a fourth partition to store data used as input at a particular layer of neurons of the artificial neural network.
44[pre]	A computer system, comprising:
44[a]	a central processing unit to receive input data acquired from an external system;
44[b]	main memory, operably coupled to the central processing unit via a bus, to store the input data received by the central processing unit;
44[c]	an accelerator, operably coupled to the central processing unit and the main memory via the bus, to receive at least a portion of the input data from the main memory, the accelerator comprising:
44[c][i]	at least one processing unit to perform a sequence of computations representing an artificial neural network on the at least a portion of the input data so as to generate output data, intermediate computations in the sequence of computations representing layers of the neural network and yielding intermediate results; and

No.	Limitation
44[c][ii]	accelerator memory, operably coupled to the at least one processing unit, to store the results of the sequence of computations; and
44[d]	a controller, operably coupled to the at least one processing unit and the accelerator memory,
44[d][i]	to control transfer of the at least a portion of the input data into the accelerator memory during performance of the intermediate computations in the sequence of computations by the at least one processing unit,
44[d][ii]	to control transfer at least a portion of the output data from the accelerator memory to the main memory during performance of the intermediate computations in the sequence of computations by the at least one processing unit, and
44[d][iii]	to control performance of the sequence of computations by the at least one processing unit.
45	The computer system of claim 44, wherein the central processing unit is configured to receive the input data in response to a user interaction.
46[a]	The computer system of claim 44, wherein: the central processing unit is configured to receive the input data at a first rate; and
46[b]	the at least one processing unit is configured to perform the sequence of computations at a second rate different than the first rate.
47	The computer system of claim 44, wherein the main memory is configured to store a copy of the output data stored in the accelerator memory.
48	The computer system of claim 44, wherein an output of at least one computation in the sequence of computations represents an output of at least one neuron in an artificial neural network.
49[a]	The computer system of claim 44, wherein accelerator memory comprises: a first memory partition to store parameters common to all of the computations in the sequence of computations; and
49[b]	a second memory partition to store data specific to at least one computation in the sequence of computations.

No.	Limitation
50	The computer system of claim 44, wherein the controller is configured to transfer the output data from the accelerator memory to the main memory without transferring any of the intermediate results from the accelerator memory to the main memory so as to reduce data transfer via the bus.
51	The computer system of claim 44, wherein the controller is configured to transfer at least a portion of the output data from the accelerator memory to the main memory after the at least one processing unit has begun to perform another sequence of computations.
52	The computer system of claim 51, wherein the controller is configured to initiate transfer of the at least a portion of the input data and to transfer the at least a portion of the output data in parallel with performance of at least one computation in the other sequence of computations by the at least one processing unit.
53	The computer system of claim 44, wherein the controller is configured to control execution of the sequence of computations by the at least one processing unit.
54	The computer system of claim 44, further comprising: at least one of a video camera, a microphone, or a cell recording electrode, operably coupled to the central processing unit, to acquire the input data in real time.