

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORPORATION,
Petitioner,

v.

NEURAL AI, LLC,
Patent Owner.

Case No. IPR2025-00609
U.S. Patent No. RE48,438

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

Petition 1

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. 7,861,060 (“Nickolls”) (Ex1004)	3
2. <i>Z. Luo, Artificial Neural Network Computation on Graphic Process Unit</i> (“ANN”) (Ex1006)	3
3. JPH04-237388A (“Tamura”) (Ex1008).....	4
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).....	5
D. How the Challenged Claims Are Unpatentable Under 37 C.F.R. § 42.104(b)(4).....	5

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
E. Supporting Evidence Under 37 C.F.R. § 42.104(b)(5)	5
F. Payment of Fees	6
III. TECHNICAL BACKGROUND	6
A. State of the Art Prior to the '438 Patent	6
1. Background on GPU Acceleration of Non-Graphics Computations and Its Applications	6
2. “Controllers” for Parallel Processing in the Prior Art	8
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	13
IV. GROUNDS OF UNPATENTABILITY	13
A. Grounds 1–4: Nickolls and ANN, Additionally in Combination With Tamura and GPU Gems, Render Obvious Claims 1–14, 16–34, and 40–54	13
1. Overview of Ground and Prior Art	14
a. Summary of Nickolls	14
b. Summary of ANN	17
c. Summary of Tamura	17
d. GPU Gems	19
2. Combinations	21
a. Combination of Nickolls and ANN	21
b. Combination of Tamura with Nickolls/ANN	24

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
3. Detailed Analysis of Grounds 1–4.....	25
a. Claim 1.....	25
i. Element 1[pre]: “A computer system, comprising”	25
ii. Element 1[a]: “a central processing unit to receive input data”	26
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”	27
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”	28
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”	29
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”	31

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”	33
ix. Element 1[d][ii]: “to control performance of the sequence of computations by the at least one graphics processing unit”	36
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”	37
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”	38
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.”	40
c. Claim 3.....	41
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”	41

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”	41
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.”	42
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.”	42
f. Claim 6.....	43
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”	43
ii. Element 6[b]: “a second memory bank to store data specific to at least one computation in the sequence of computations”	44
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.”	45
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.”	46

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.”	47
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.”	48
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.”	48
l. Claim 12.....	49
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”	49
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”	49
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”	50
iv. Element 12[a][ii]: “(ii) accumulating a first value for the output variable in the first texture during a first time step”	51
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”	51
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”	52
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”	53
viii. Element 12[d][i]: “(i) accumulating a second value for the output variable in the second texture during a second time step and”	53
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”	53
m. Claim 13: “The method of claim 12, further comprising: storing the input data in the main memory in response to a user interaction.”	54
n. Claim 14.....	55
i. Element 14[a]: “The method of claim 12, further comprising: receiving the input data at a first rate; and”	55
ii. Element 14[b]: “wherein (A) comprises performing the sequence of computations at a second rate different than the first rate”	55
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.”	55
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.”	55
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.”	56
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.”	56
s. Claim 20.....	56
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”	56
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”	57
t. Claim 21.....	57
i. Element 21[pre]: “A method of performing a sequence of computations representing an artificial neural network, the method comprising:”	57
ii. Element 21[a]: “receiving, at a central processing unit (CPU), first input data acquired from an external system in real time” ...	57

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”	57
iv. Element 21[c]: “transferring the first input data received by the CPU to the memory operably coupled to the GPU”	58
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”	58
vi. Element 21[e]: “storing, in the memory operably coupled to the GPU, the first input data and the first output data”	58
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”	59

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
u. Claim 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.”	59
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.”	59
w. Claim 24.....	60
i. Element 24[a]: “The method of claim 23, further comprising: storing intermediate results of the sequence of computations in the memory, and”	60
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”	60
x. Claim 25: “The method of claim 23, wherein transferring the second input data and transferring the first output data occurs in parallel.”	60
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.”	61
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.”	61

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
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 sequence of computations, and the output of the computation in the sequence of computations.”	62
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.”	64
cc. Claim 30.....	64
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”	64
ii. Element 30[b]: “accessing, by the GPU, the output of the previous computation during performance of the computation in the sequence of computations”	64
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.”	65
ee. Claim 32: “The method of claim 31, wherein all neurons in the layer of neurons are described by the same equation.”	65
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.”	65

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
gg. Claim 34: “The method of claim 21, further comprising: loading the second input data from disk.”	66
hh. Claim 40.....	66
i. Element 40[pre]: “A system for executing an artificial neural network, the system comprising:”	66
ii. Element 40[a]: “a central processing unit (CPU) to provide first input data;”	66
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”	66
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”	68
v. Element 40[d]: “a controller, operably coupled to the processing unit and the memory, to:”	68
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	

TABLE OF CONTENTS (continued)

	<u>Page</u>
layer of neurons of the artificial neural network”	69
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.”	71
ii. Claim 41: “The system of claim 40, wherein the processing unit comprises a graphics processing unit.”	72
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.”	72
kk. Claim 43.....	72
i. Element 43[a]: “The system of claim 40, wherein the memory further comprises: a third partition to store internal variables; and”	72
ii. Element 43[b]: “a fourth partition to store data used as input at a particular layer of neurons of the artificial neural network”	73
ll. Claim 44.....	74
i. Element 44[pre]: “A computer system, comprising:”	74
ii. Element 44[a]: “a central processing unit to receive input data acquired from an external system”	74

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
iii. Element 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”	74
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”	74
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”	74
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”	74
vii. Element 44[d]: “a controller, operably coupled to the at least one processing unit and the accelerator memory”	75
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”	75
ix. Element 44[d][ii]: “to control transfer at least a portion of the output data from the accelerator memory to the main memory	

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
during performance of the intermediate computations in the sequence of computations by the at least one processing unit”	75
x. Element 44[d][iii]: “to control performance of the sequence of computations by the at least one processing unit”	75
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.”	75
nn. Claim 46.....	75
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”	75
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”	75
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.”	76
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.”	76
qq. Claim 49.....	76

TABLE OF CONTENTS *(continued)*

	<u>Page</u>
i. Element 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”	76
ii. Element 49[b]: “a second memory partition to store data specific to at least one computation in the sequence of computations”	76
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.”	76
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.”	76
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.”	77
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.”	77

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.”	77
V. ENABLEMENT AND SECONDARY CONSIDERATIONS	77
VI. DISCRETIONARY DENIAL	Error! Bookmark not defined.
VII. CONCLUSION.....	78
APPENDIX: CHALLENGED CLAIM LISTING	81

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)	77

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	<i>Z. Luo, Artificial Neural Network Computation on Graphic Process Unit (IEEE 2005) (Ex1006)</i>
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 (IPR2025-00610). 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 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. To the contrary, 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 “ANN” article describes using a multi-layer artificial neural network on an NVIDIA GPU to track a soccer ball in real time.

The reissue claims also recite a litany of well-known techniques, such as timing data transfers so that the GPU is not idle while those transfers occur. The combination of Nickolls and ANN 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 ANN 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.

Petitioner therefore respectfully requests the Board institute IPR and invalidate all claims of this patent.

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 Nickolls in view of ANN
2	1–14, 16–34, 40–54	Obviousness over Nickolls in view of ANN and Tamura

¹ All obviousness grounds include the knowledge of a POSITA.

Ground	Claims	Basis for Rejection¹
3	40–43	Obviousness over Nickolls in view of ANN and GPU Gems
4	40–43	Obviousness over Nickolls in view of ANN, Tamura, and GPU Gems

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

1. U.S. Patent No. 7,861,060 (“Nickolls”) (Ex1004)

Nickolls issued on December 28, 2010, from a non-provisional application filed on December 15, 2005. Ex1004, 1. Nickolls is assigned to Petitioner NVIDIA. Nickolls qualifies as prior art to the ’438 patent under pre-AIA 35 U.S.C. § 102(e). Nickolls was of record during prosecution of the ’438 patent but was not substantively discussed or applied by the Examiner.

2. Z. Luo, *Artificial Neural Network Computation on Graphic Process Unit* (“ANN”) (Ex1006)

ANN was presented at the 2005 IEEE International Conference on Neural Networks, from July 31 to August 4, 2005, in Montreal, Canada. Ex1006, 622; Ex1023, ¶¶1–19, 20–37; Ex1024. ANN was made available to conference participants by the last day of the conference. Ex1023, ¶¶20–37. ANN was added to IEEE Xplore on December 27, 2005, and ANN would have been located by searching for terms such as “artificial neural networks” or “graphics processing unit.” *Id.* ANN, as part of the published conference proceedings, was cataloged in the University of Arizona’s library by January 4, 2006. *Id.* A POSITA, exercising

reasonable diligence and searching for references regarding “neural networks” or the like would have located ANN around January 4, 2006. *Id.* ANN therefore qualifies as prior art to the ’438 patent under at least pre-AIA 35 U.S.C. § 102(a) and/or (b). ANN was of record during prosecution of the ’438 patent 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(b). Tamura was not of record during prosecution of the ’438 patent.

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, 38–48. 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 and searching for references regarding programming GPUs, would have located GPU Gems in July 2005. *Id.* A skilled researcher would have located GPU Gems 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 therefore qualifies as prior art to the '438 patent 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 to the Challenged Claims, including an identification of specific portions of the evidence that support the challenge, are provided below. An exhibit list with 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 explained in this Petition are further supported by the Declaration of Prof. Tajana Rosing. Ex1003, ¶¶1–406.

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. *See* Ex1003, ¶¶33–35, *see also id.* ¶¶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. As discussed in the prior art references in this Petition, 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.* For example, 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. For example, 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.* The book *GPU Gems* provides multiple examples of pointer swapping with GPUs, such as in a chemical reaction-diffusion. *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.

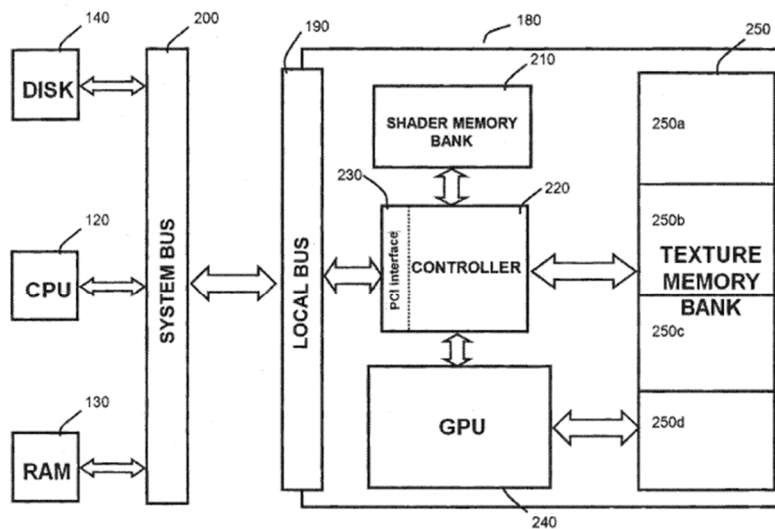


FIG. 2

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 (“the '828 patent”). 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 examiner rejected the pending claims in four office actions on several bases, including indefiniteness (§ 112),

improper recapture (§ 251), anticipation (§ 102), and obviousness (§ 103). *Id.*, 498–550, 697–752, 825–27, 841–93, 956–89, 1024–26.

Throughout 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 at least one prior art reference (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.” *E.g., id.* at 850.

To gain allowance, the applicant amended claims to recite the claimed “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, which were amended as explained above (the remaining pending claims

having been canceled). *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, with 56 claims. 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: Nickolls and ANN, Additionally in Combination With Tamura and GPU Gems, Render Obvious Claims 1–14, 16–34, and 40–54

Nickolls’ GPU accelerator and ANN’s artificial neural network for tracking a soccer ball in real time, 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. Summary of Nickolls

Nickolls discloses a computer system for “parallel data processing” that provides “flexible, general-purpose computational capacity in a GPU that may be used for computations in any field.” Ex1004, 1:21–26, 2:21–23, 30:25–31. Nickolls includes CPU 102 (red), system memory 104 (blue), and GPU 122 (purple), all of which communicate over bus 113. *Id.*, 6:9–34.

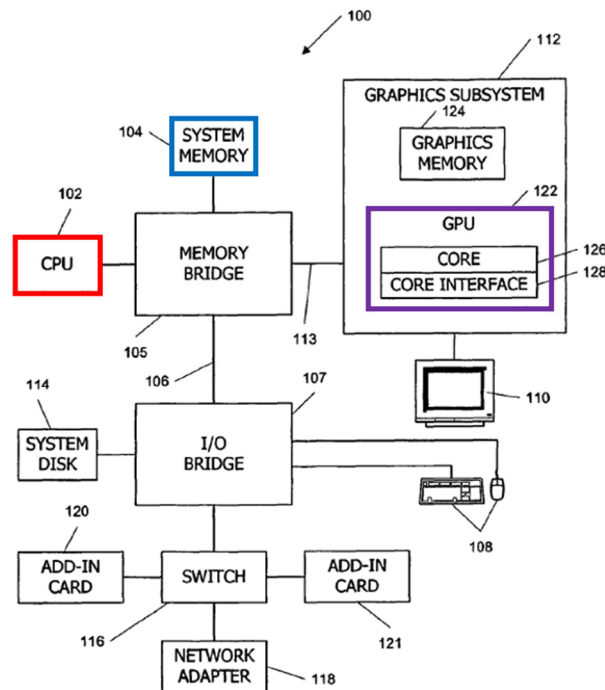


FIG. 1

Id., Fig. 1 (annotated).

GPU 122 includes at least one processing core 126 (green) and core interface 128 (orange). *Id.*, 6:47–55. Each core 126 “includes multiple parallel processing engines that can be used to execute various shader programs” and “can also be

leveraged to perform general-purpose computations.” *Id.*, 6:49–55. Each processing engine is further configured to execute multiple “threads” in parallel. *Id.*, 10:24–36, 2:22–27.

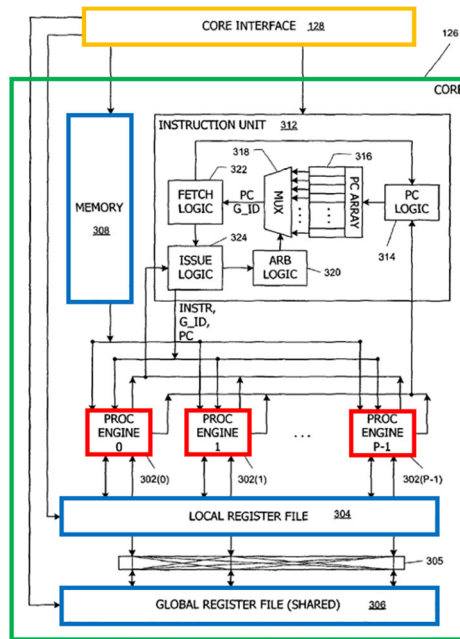


FIG. 3

Id., Fig. 3 (annotated).

Core 126 includes multiple memory modules: local register file 304, global register file 306, and shared memory 308. *Id.*, 10:57–11:18. The processing engines can read from and write to memory, which may include (at various times) shader programs or other instructions, input data, intermediate results, and output data. *Id.*, 7:27–30, 10:57–61, 20:35–45. The processing engines (and their respective threads) are configured to share data, including intermediate results of computations, with each other. *Id.*, 1:22–26, 2:33–35, 8:10–14, 25:8–10.

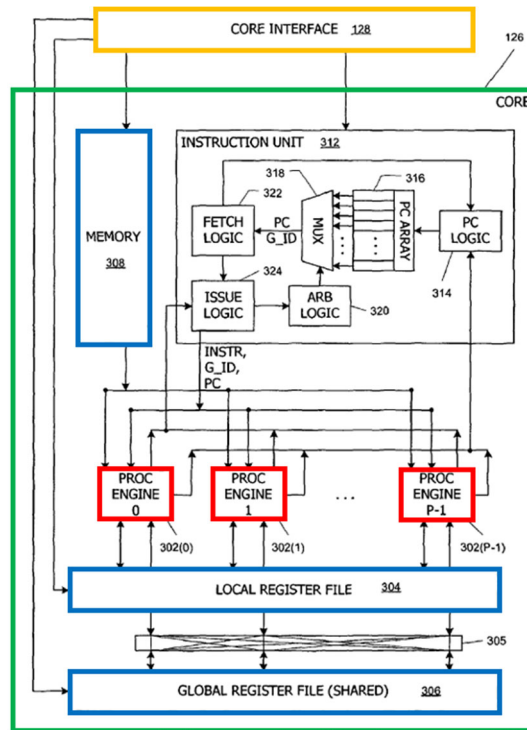


FIG. 3

Id., Fig. 3 (annotated).

Core interface 128 “controls operation of core 126.” *Id.*, 6:47–49, 14:20–22. Its functions include receiving “state information, data, and commands” from CPU 102, *id.*, 14:29–31, “load[ing] the input data into (shared) global register file 306 of core 126,” *id.*, 19:44–50, and loading data and instructions for each processing core into local register file 304 and instructing the core to launch processing. *Id.*, 19:51–61. Core interface 128 also uses instruction unit 312 of core 126 to distribute and execute instructions in GPU 122. *Id.*, 10:37–56, 11:27–29, 12:19–33, 21:37–55; Ex1003, ¶¶126–36.

b. Summary of ANN

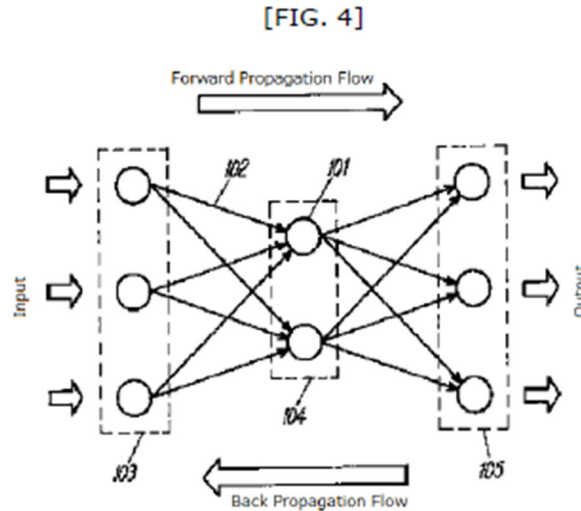
ANN teaches that artificial neural networks are “widely used in pattern recognition,” and are beneficial where “computational load is very heavy” or where “real time process is required.” Ex1006, 622. ANN explains GPUs are well suited for artificial neural networks due to their inherently parallel and repetitive nature. *Id.* ANN applies these principles to the use of artificial neural networks on GPUs for real-time ball tracking in robot soccer. *Id.*, 2–5. The authors used a “[t]hree layer MLP neural network . . . to recognize the ball,” with an “input layer consist[ing] of seven nodes,” a “hidden layer consist[ing] of three nodes,” and an “output layer consist[ing] of just one node.” *Id.*, 3. The cores of the GPU perform matrix multiplication at each layer. *Id.* The researchers used three passes through the multi-layer artificial neural network for the final output. *Id.*, 3–4.

The researchers ran the artificial neural network on “commodity graphic hardware,” including an NVIDIA GF6000 GPU. *Id.*, 4. To improve efficiency, they used the feature “Render to Texture” to “save the intermediate result in a texture on GPU and reuse it as an input data” in a next step. *Id.*; Ex1003, ¶¶137–39.

c. Summary of 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

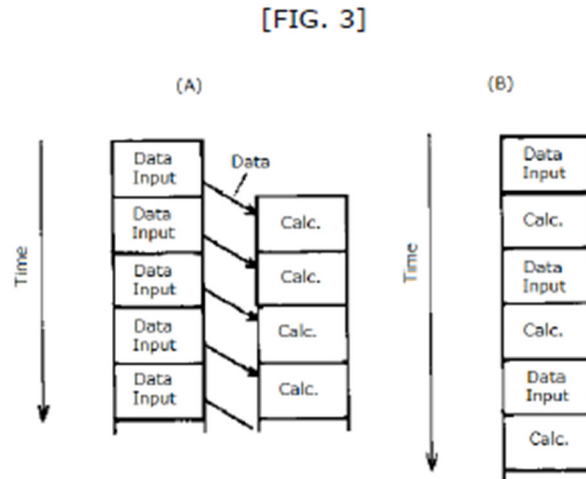
create a new type of computer that excels at tasks such as recognition, association, optimization, and speech synthesis.” *Id.*, ¶2. Figure 4 shows the simulated neural network, with circles 101 as neurons and lines 102 as connections between neurons in the first (input), intermediate, and output layers. *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 problem, Tamura proposes inputting data and writing it to memory while calculations are being performed on previously inputted data. *Id.*, ¶29. By carrying out the “data input operation in parallel with calculations,” “the total processing time is reduced.” *Id.*, ¶¶30, 35, Fig. 3. Tamura also emphasizes that 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.” *Id.*, ¶37.



Id., Fig. 3; Ex1003, ¶¶140–43.

d. GPU Gems

GPU Gems includes 48 chapters describing the state of the art in GPU programming, including a “Primer” on “General-Purpose Computation on GPUs.” Ex1008, xx, xxxi–xxxii, 453. GPU Gems describes swapping pointers of input and output data. Chapter 31, written by Mark Harris of NVIDIA, 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.*

Mr. Harris 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, ¶¶144–47.

2. Combinations

a. Combination of Nickolls and ANN

Before the '438 patent, it was well known that artificial neural networks could be effectively and efficiently implemented on GPUs. Section III.A.1. One such example is ANN's artificial neural network for tracking a soccer ball in real time implemented on an NVIDIA GPU. Ex1003, ¶¶156–64. A POSITA would have been motivated to implement ANN's artificial neural network on other NVIDIA GPU hardware, such as that in Nickolls. *Id.*, ¶156.

ANN discloses a “[t]hree layer MLP neural network” to “recognize and trace [a] ball in real time” for robot soccer. Ex1006, 624. ANN emphasizes that its artificial neural network should be executed on “commodity graphic hardware.” *Id.*, 5. ANN showed that “GPU based [neural network] computation is about 200 times faster than that of CPU.” *Id.*, 4.

Although ANN used NVIDIA's GF6000, a POSITA would have understood that ANN is agnostic as to the specific GPU used. Ex1003, ¶158. Thus, to achieve even further efficiency in computation, a POSITA would have been motivated to implement ANN's artificial neural network on state-of-the-art GPU-accelerated parallel processing systems, such as that in Nickolls. *Id.* The system of Nickolls is specifically designed to perform the kinds of parallel computations required for artificial neural networks, as it “may be used for computations in any field,”

including “matrix algebra.” Ex1004, 28:66–29:5, 30:25–31. As ANN explains and a POSITA would have understood, a key mathematical operation in artificial neural networks is matrix multiplication. Ex1006, 624; Ex1003, ¶158.

A POSITA would have understood numerous advantages to applying the artificial neural network of ANN on the hardware of Nickolls. Ex1003, ¶159. Specifically, Nickolls would significantly reduce execution time compared to other processing architectures because its multiple processing engines can execute the same program on different portions of input data at the same time. *Id.*

The system of Nickolls would also be ideal for the artificial neural network of ANN because of its capabilities for data sharing among threads. *Id.*, ¶160. To increase efficiency, ANN’s artificial neural network “save[s] the intermediate result [of a calculation] in a texture on GPU and reuse[s] it as an input data.” Ex1006, 625. With its shared local and global memory, Nickolls would have allowed the iterative computations of ANN’s artificial neural network in which the outputs of one stage of computations become inputs to subsequent stages. *Id.*

Further, Nickolls is assigned to NVIDIA, who was a leader in the industry in GPUs development at the time of the ’438 patent. Ex1004, Cover; Ex1003, ¶161. A POSITA would have expected that the most advanced GPUs for performing GPGPU computations (including for artificial neural networks) would have come

from NVIDIA.² *Id.* When seeking to execute complex artificial neural networks (such as that in ANN), a POSITA would have sought out advanced systems with programmable GPUs, including those from NVIDIA. *Id.*

A POSITA would have understood the combination of Nickolls and ANN simply applied a known technique (*e.g.*, the artificial neural network of ANN) to a known device (*e.g.*, the GPU-accelerated parallel processing system of Nickolls) ready for improvement to yield predictable results (*e.g.*, fast and efficient computations for an artificial neural network with real-time data input). *Id.*, ¶162.

A POSITA would have had a reasonable expectation of success in combining Nickolls and ANN. *Id.*, ¶163. Nickolls teaches that it is suited for “matrix algebra,” which a POSITA would have understood is one of the most computationally intensive tasks for an artificial neural network. Furthermore, the system of Nickolls was designed for running many tasks (threads) at the same time. Since simulating multi-layer artificial neural networks require dividing the computation into smaller, discrete tasks, a POSITA would have expected the system of Nickolls to be able to use parallel processing to execute the computations required in ANN. *Id.*

² Indeed, the researchers in ANN used an NVIDIA GPU. Ex1006, 625.

b. Combination of Tamura with Nickolls/ANN

Prior to the '438 patent, techniques had already been developed to accelerate computations in artificial neural networks, such as Tamura's simultaneous input and data processing in a neural network processor. A POSITA would have been motivated to apply Tamura's data transfer and computation techniques for artificial neural networks to accelerate the system of Nickolls/ANN. Ex1003, ¶¶165–68.

Tamura is directed to an improvement in processing times for an artificial neural network. Ex1008, ¶ 30. Tamura teaches that, while previously input data is processed and calculations are performed, new data is input and saved to memory. *Id.*, ¶29. To the extent Nickolls/ANN does not teach that data processing and data input should occur simultaneously, a POSITA would have motivated to apply the teachings of Tamura to Nickolls/ANN. A POSITA would reasonably expect the combination would improve computational efficiency, as taught in Tamura. Ex1003, ¶166.

For real-time applications such as tracking a soccer ball, it would have been important for a POSITA to implement Nickolls to simultaneously and continuously receive and process data from a camera. *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 teaches 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. A POSITA would have understood Tamura’s teachings would be applicable to Nickolls’ GPU-accelerated system, which can process neural networks. Ex1003, ¶167.

A POSITA would have understood the combination of Tamura with Nickolls/ANN 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 Nickolls/ANN) ready for improvement to yield predictable results (*e.g.*, fast and efficient computations for an artificial neural network with real-time data input). *Id.*, ¶¶165–68.

3. Detailed Analysis of Grounds 1–4

a. Claim 1

i. 1[pre]

To the extent the preamble is limiting, Nickolls/ANN discloses it. Ex1003, ¶¶170–71.

Nickolls discloses “systems and methods” for “parallel data processing.” Ex1004, 1:21–26. The computer system is illustrated in Figure 1.

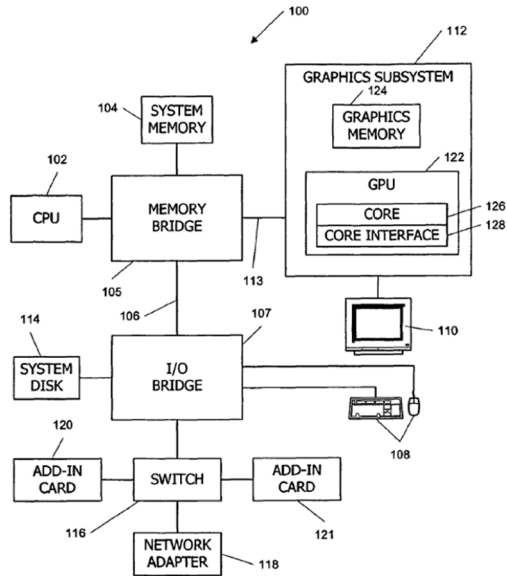


FIG. 1

Id., Fig. 1.

ii. 1[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶172–73.

The computer system of Nickolls includes a central processing unit (*e.g.*, CPU 102). Ex1004, 6:10–14. CPU 102 receives input data, which it supplies to GPU 122 for processing. *Id.*, 6:40–46, 6:40–46, 6:62–7:4. In Nickolls/ANN, the input data is video data from a camera. Ex1006, 624–26.

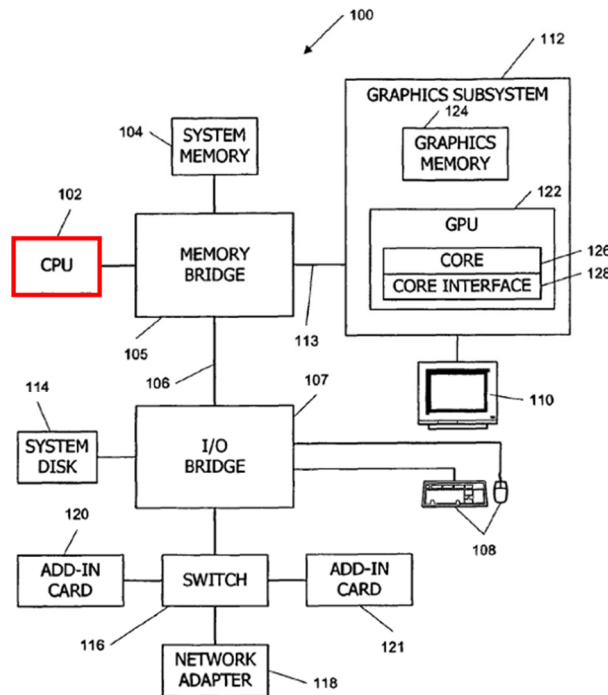


FIG. 1

Ex1004, Fig. 1 (annotated).

iii. 1[b]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶174–75.

Nickolls includes main memory (*e.g.*, system memory 104) operably coupled to CPU 102 via a bus (either through memory bridge 105 or directly). Ex1004, 6:9–18, 6:40–46, 7:8–12. System memory 104 stores input data, which the CPU can access and send to GPU 122. *Id.*, 6:35–46, 6:65–7:12, 11:14–18, 13:16–29.

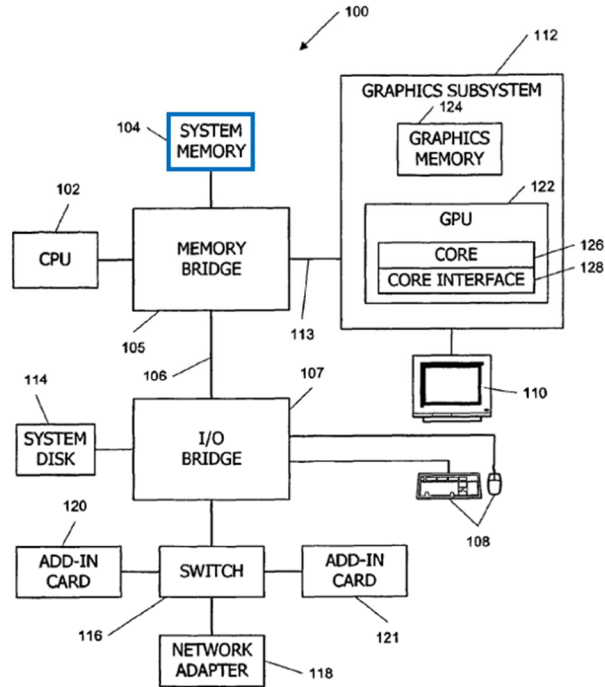


FIG. 1

Id., Fig. 1 (annotated).

iv. 1[c]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶176–77.

Nickolls includes an accelerator (*e.g.*, GPU 122), which is coupled to CPU 102 and system memory 102 via bus 113. Ex1004, 6:9–22, 6:35–55. GPU 122 receives input data from system memory 102 (directed by CPU 102) at input 402 of core interface 128. Ex1004, 6:40–46, 14:29–31, 19:45–50, Figs. 1, 3, 4, 10.

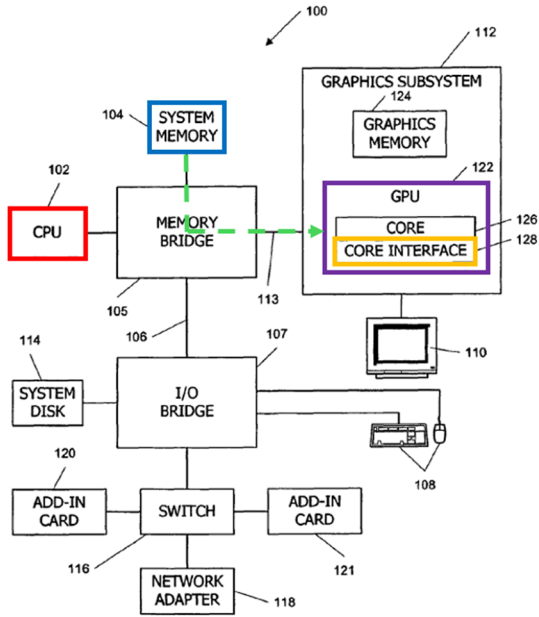


FIG. 1

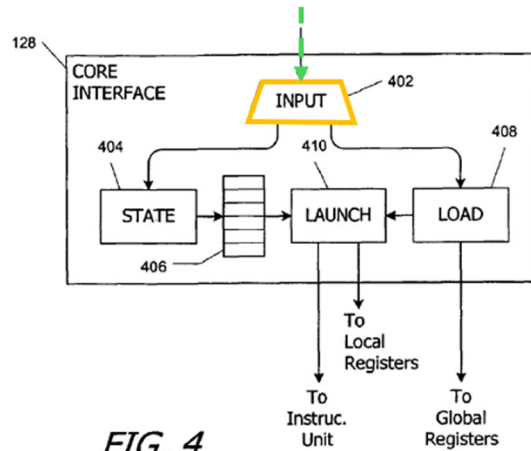


FIG. 4

Ex1003, Figs. 1, 4 (annotated).

v. 1[c][i]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶178–82.

Each core of GPU 122 in Nickolls “includes multiple processing engines” that can “be leveraged to perform general-purpose computations.” Ex1004, 6:35–55, 7:51–65, 30:52–60. Each processing core 126 of GPU 122 (purple) as a whole (green) and the processing engines (red) within core 126 (because they are the processors the perform the graphics and other computational operations) are a “graphics processing unit.” Ex1003, ¶179; Ex1001, 1:32–34 (“[GPUs] are considered highly parallel processors dedicated to fast computation of graphical content.”).

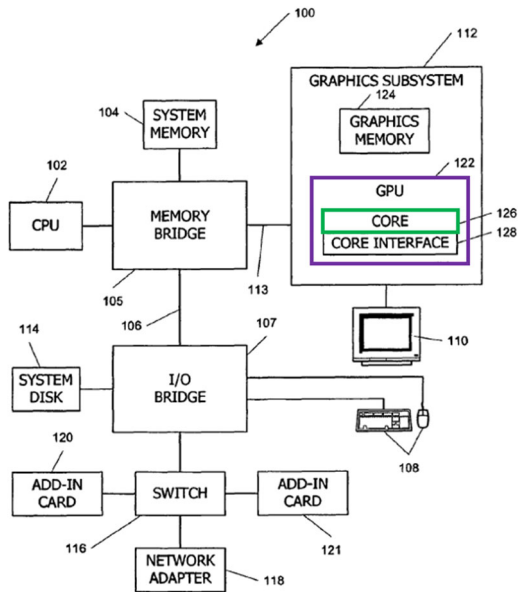


FIG. 1

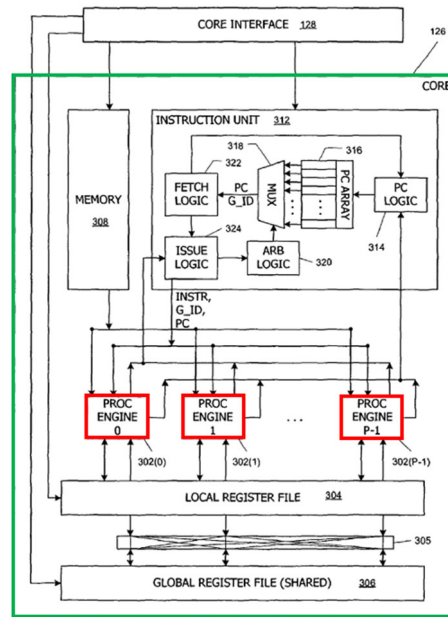


FIG. 3

Ex1004, Figs. 1, 3 (annotated).

In Nickolls, each processing engine can execute instructions (*e.g.*, a sequence of computations) in multiple, parallel threads on a portion of the input data to generate output data. *Id.*, Abstract, 7:51–8:3, 10:24–56, 20:35–45. Each thread of a processing engine is assigned to operate on a segment of the data and accesses it through shared or dedicated memory partitions. *Id.* Once the input data is received, the thread associated with each portion of the input data begins computations simultaneously with the others, each executing their assigned workloads in parallel to generate “output data.” *Id.*, 7:66–8:3, 10:57–59, 19:51–61, 20:20–45, 20:61–21:8.

In Nickolls/ANN, the sequence of computations represents an artificial neural network. Ex1003, ¶181. ANN discloses computations on a GPU that represent a “[t]hree layer MLP neural network” (*i.e.*, an artificial neural network). Ex1006, 622,

624. ANN’s multi-layer neural network includes an input layer with seven nodes (*i.e.*, neurons), a second (hidden) layer with three nodes, and an output layer of just one node, with each layer representing a layer of neurons in the artificial neural network. *Id.*, 624. At each layer, the neural network relies on matrix multiplication and a sigmoid function to determine the output (intermediate results), which are fed into the next layer. *Id.* Thus, in Nickolls/ANN, 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, ¶¶178–82.

vi. 1[c][ii]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶183–84.

GPU 122 of Nickolls includes local register file 304, shared global register file 306, and memory 308 (blue) (*i.e.*, “accelerator memory”)—which is coupled to the graphics processing unit (*e.g.*, core 126 (green) and/or processing engines (red))—for storing input and output data (*e.g.*, the results of the sequence of computations in an artificial neural network). Ex1004, 10:57–11:18, 13:12–25, 20:35–45. For example, each processing engine of GPU 122 “is allocated space in a local register file 304 for storing its local input data, intermediate results, and the like.” Ex1004, 10:57–11:2. Results from computational cycles can also be stored in a dedicated portion of shared global register file 306. *Id.*, 8:14–17, 11:3–18, 20:35–45.

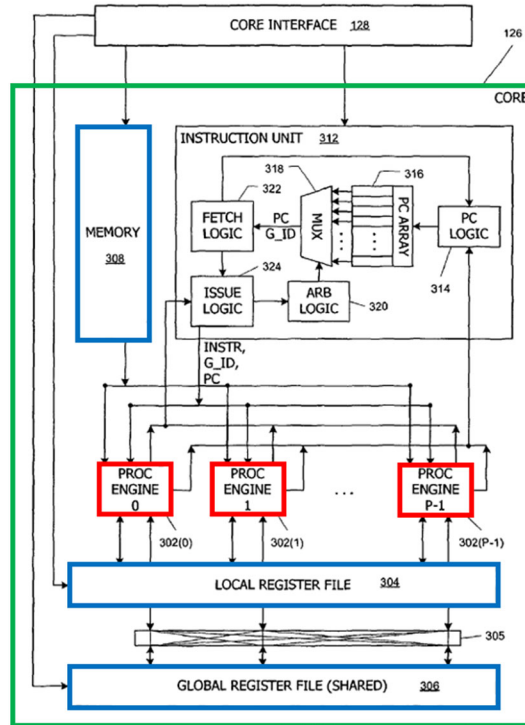


FIG. 3

Id., Fig. 3 (annotated).

vii. 1[d]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶185–86.

Nickolls includes a “controller,” *e.g.*, core interface 128 (orange), which “controls operation of core 126.” *Id.*, 6:47–55, 14:20–22, 13:12–67, 14:17–28, Figs. 3, 4; Ex1003, ¶186. Additionally and alternatively, the “controller” of Nickolls may also include instruction unit 312, which works in conjunction with core interface 128 to supply instructions to the processing engines and manage their execution. Ex1004, 12:19–33, 13:12–45, 21:33–63; Ex1003, ¶186. Like the “controller” in the ’438 patent, the controller of Nickolls 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. The controller of Nickolls is coupled to the graphics processing units (e.g., core 126 and/or processing engines) and the accelerator memory. Ex1004, 10:57–11:18, 13:12–25.

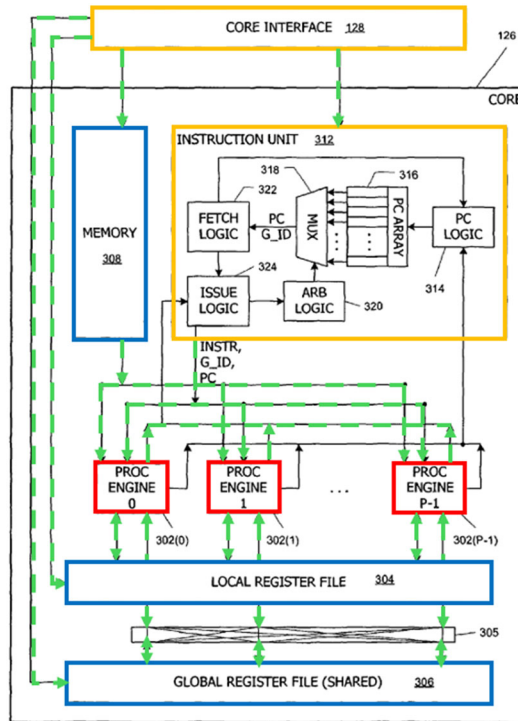


FIG. 3

Id., Fig. 3 (annotated).

viii. 1[d][i]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶187–94.

When Nickolls is used for numerical computations (such as for an artificial neural network in Nickolls/ANN), core interface 128 initializes textures and shaders in memory of GPU 122. Although Figure 4 illustrates information transmitted to local and global registers, Nickolls teaches that data and instructions can be sent to and stored in any memory in GPU 122. *Id.*, 10:57–11:18, 20:35–45.

Initializing Textures. Load module 408 of core interface 128 “receives input parameters for a CTA³ and loads the input parameters into (shared) global register file 306 (FIG. 3) of core 126,” which is accessible to the processing engines to use in computations. Ex1004, 16:63–65; *see also id.*, 1:30–33, 2:22–35, 3:20–57, 4:16–37, 8:14–30, 10:57–11:2, 13:12–45, 16:63–17:19, 19:44–50, Figs. 3, 4, 10. The received parameters include “the input data to be processed,” which are “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, ¶¶187–89. When the processing engines are “leveraged to perform general-purpose computations,” data is stored in the “textures” typically used for graphics. Ex1004, 6:35–55, 7:51–65; Ex1003, ¶¶187–89. The system further initializes textures (data arrays) for storing data output from computations. Ex1004, 4:16–37, 7:51–8:9, 20:35–21:8; Ex1003, ¶¶187–89.

Initializing Shaders. Core interface 128 additionally initializes the programs to be executed in memory, making them available to the processing engines to be used. *Id.* For example, Nickolls discloses instructions (programs) stored in memory

³ “CTA” is a “cooperative thread array,” “a group of multiple threads that concurrently execute the same program on an input data set to produce an output data set.” Ex1003, 2:22–27.

that is initialized by its memory address being loaded in core interface 128. Ex1004, 14:44–48, 13:41–45; *see also id.*, 10:37–56, 12:34–13:11, 20:20–21:8. These “various shader programs” are executed in core 126. *Id.*, 6:49–55. The “program” in memory is a “shader” because it is a “GPU program.” Ex1001, 5:24–26 (“[T]he term ‘shader’ in this document refers to a GPU program unless otherwise specified.”); Ex1004, 6:47–55, 8:3–6, 28:16–21. When the processing engines are “leveraged to perform general-purpose computations,” the shader programs for graphics are repurposed to contain the “shader programs” for the general-purpose computations. Ex1004, 6:35–55, 7:51–65; Ex1003, ¶¶190–91.

In Nickolls/ANN, the “shaders” initialized in memory of GPU 122 would include the equations to be applied at each layer of the multi-layer neural network, and the “textures” initialized in memory of GPU 122 would include the initial neural network parameters, input data, and the weights to be applied at each layer. Ex1003, ¶¶191–93.

Additionally, it would have been obvious to a POSITA to initialize textures (data arrays) and shaders (programs) in memory of the accelerator, making them available to the processing cores. Ex1003, ¶194. These ideas are fundamental to GPGPU and were well known prior to the ’438 patent. Section III.A.1; Ex1003, ¶¶31–86. For the processing engines 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 engines. *Id.*

ix. 1[d][ii]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶195–97.

Core interface 128 of Nickolls, alone and with instruction unit 312, “controls operation of core 126,” which includes a GPU (*e.g.*, processing engines). Ex1004, 6:47–49, 14:20–22; *id.*, 13:12–67, 14:17–28, 19:51–61, Fig. 4; Ex1003, ¶196. Each core 126 “includes multiple parallel processing engines that can be used to execute various shader programs” and “can also be leveraged to perform general-purpose computations.” Ex1004, 6:49–55. Instruction unit 312, under the direction of core interface 128, distributes instructions to the processing engines and helps manage synchronization of the engines as they execute instructions. *Id.*, 11:27–29, 12:19–33, 12:43–55, 21:37–55, 23:61–67. Further, the transfer of data to the memory of core 126, which is used in the sequence of computations, is controlled by core interface 128. *Id.*, 13:16–25, 16:63–17:22, 19:44–50.

Thus, because the instructions for performing the sequence of computations in processing engines (GPU) of core 126 and the data for such computations are provided by core interface and instruction unit 312 (*i.e.*, the “controller”), the controller “control[s] performance of the sequence of computations by the at least one graphics processing unit.” Ex1003, ¶197.

x. 1[d][iii]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶198–201.

In Nickolls/ANN, 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 layer of the multi-layer neural network). Ex1003, ¶199; *see also* Ex1004, 26:55–62 (teaching having the CPU perform functions while a CTA is being processed by GPU 122), 6:62–7:4 (GPU 122 “executes commands asynchronously with operation of CPU 102”). Because the input in ANN is a real-time video feed of soccer, after input 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 in the layers of the artificial neural network for the earlier input data to keep up with the real-time data input. Ex1006, 624–26.

As explained in 1[c], 1[c][ii], GPU 122 of Nickolls includes local register file 304, shared global register file 306, and memory 308 for storing input and output data, *i.e.*, the “accelerator memory.” Ex1004, 10:57–11:18, 13:12–28, 16:63–65, 19:44–50, 20:35–45. The controller (*e.g.*, core interface 128, alone and with instruction unit 312) transfers input data into the accelerator memory. *Id.*; Ex1003, ¶200.

Further, Tamura also discloses this limitation. *Id.*, ¶201. Tamura teaches, in the context of simulating an artificial 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 Nickolls/ANN, additional frames from the real-time video input is transferred into accelerator memory during performance (by GPU 122) of intermediate computations in the artificial neural network on one or more earlier frames of input data. Ex1003, ¶201. It would have been obvious to a POSITA, in view of Tamura, to implement Nickolls/ANN 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. *Id.*; Ex1008, ¶¶30, 35, Fig. 3.

xi. 1[d][iv]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶202–06.

In Nickolls/ANN, 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.*, one or more video frames) by the graphics processing unit. Ex1003, ¶203; Ex1004, 20:35–45; Ex1006,

625–26. Following computations, 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. Ex1003, ¶203. Nickolls provides for large flexibility in when and where data is stored, *see* Ex1004, 20:35–45, and describes high-speed interconnects between GPU memory and system memory that are capable of quickly transferring data, *id.*, 6:27–33 (*e.g.*, PCI-E). Where the artificial neural network is processing real-time 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. Ex1006, 624–25; Ex1003, ¶203. In short, in Nickolls/ANN, there are overlapping computation and data transfers.

The transfer of data occurs based on instructions received from the “controller”—*i.e.*, core interface 128, including through instruction unit 312. Ex1004, 20:35–45, 10:37–40, 13:41–45. As in the ’438 patent, the “controller” of Nickolls/ANN is part of the “accelerator” (*i.e.*, GPU 122), which provides accelerated, parallel data processing, separately from the CPU. Ex1003, ¶145; Ex1001, 3:40 (referring to a “GPU accelerator”); Ex1003, ¶204.

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. *Id.*, ¶205. The objective of ANN is to track a soccer ball in real time. Ex1006, 622. Unless there are overlapping computations and data transfer back to main memory, the system in Nickolls/ANN may not be able to provide data that can be used by the CPU to track the soccer ball. Ex1003, ¶205; Ex1004, 20:67–21:3 (output data transferred to system memory 104, “making it available to application programs executing on CPU 102”). Because accelerator memory is more closely coupled to the processing engines and outputs can be used as inputs in subsequent time steps, the engines 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. Ex1004, 20:35–45; Ex1006, 625–26; Ex1003, ¶¶202–06.

b. Claim 2

Nickolls/ANN discloses this limitation. Ex1003, ¶¶207–08.

Nickolls discloses an “application program interface (API)” on the CPU “for defining and executing CTAs” to “allow application programmers to access CTA functionality.” Ex1004, 25:66–26:4. The API allows “an application programmer [to] invoke the GPU functions by including suitable function calls from the API at appropriate places in the program code.” *Id.*, 26:5–11. One of the functions a programmer (user) could invoke is for the CPU to receive data. Ex1003, ¶208; *see*

1[a]. Thus, through the API, CPU 102 is configured to receive input data in response to user interaction (*e.g.*, a programmer invoking the API).

c. Claim 3

i. 3[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶209–10.

In Nickolls/ANN, the input data is live video from a camera. Ex1006, 624–26. Video can be input at many different rates, such as 30 frames per second, and at different resolutions. Ex1003, ¶210. The CPU in Nickolls/ANN receives the video input at a first rate (*e.g.*, 30 fps or any other known video rate). *Id.* For example, if the input data is uncompressed, each frame may 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.*

ii. 3[b]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶211–12.

ANN discloses various rates for performing computations in the artificial neural network, including depending on how many passes are made and what scheme is selected for the minimum finding procedure. Ex1006, 625 (Tables I and II). Because in ANN the “GPU computation is fast enough for the locating of the ball in real time,” the rate of the sequence of computations is different (*i.e.*, faster)

than the rate of receiving input data. *Id.*, 625; Ex1003, ¶212. For example, ANN discloses an embodiment where each sequence of computations (*e.g.*, for one frame) takes 46 ms, which equates to around 22 operations per second. Ex1006, 625; Ex1003, ¶212. Furthermore, when ANN is combined with Nickolls, a POSITA would have understood multiple potential rates of computation for the GPU in Nickolls, depending on the precise algorithm implemented (as ANN discloses several). Ex1003, ¶212. Thus, Nickolls/ANN discloses that GPU 122 performs the sequence of computations at a different rate than the rate of data input.

d. Claim 4

Nickolls/ANN discloses this limitation. Ex1003, ¶¶213–14.

See 1[d][iv]. Nickolls teaches storing outputs from computations in both accelerator memory and system memory 104. Ex1004, 20:35–45. For example, according to Nickolls, “[i]ntermediate results may be written to global register file 306 and/or other memory such as graphics memory 124 or system memory 104 of FIG. 1.” *Id.* Thus, main memory (system memory 104) is configured to store a copy of the same output data stored in accelerator memory.

e. Claim 5

Nickolls/ANN discloses this limitation. Ex1003, ¶¶215–16.

See 1[c][i]. ANN discloses computations on a GPU that represent a “[t]hree layer MLP neural network” (*i.e.*, an artificial neural network). Ex1006, 622, 624.

“The input layer consists of seven nodes,” (*i.e.*, neurons), “[t]he hidden layer consists of three nodes, and the output layer consists of just one node.” *Id.* As explained in Section III.A.3, at each layer, the system performs matrix multiplication and a sigmoid function to determine an output for each node, each of which represents a neuron in the artificial neural network. Ex1003, ¶216.

f. Claim 6

i. 6[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶217–18.

In Nickolls/ANN, the system includes a first memory bank (*e.g.*, a logical portion of global register file 306 or local register file 304) to store parameters (*e.g.*, weights and equations for the artificial neural network being executed) common to all of the computations in the sequence of computations. The data stored in global register file 306 for the processing cores includes “input parameters,” such as information specific to that thread. Ex 1004, 13:16–45, 16:63–17:3, 24:39–47. In the Nickolls/ANN combination, the “input parameters” include the variables, weights, and equations for the multi-layer artificial neural network. Ex1006, 623–25. Because all threads execute the same instructions and equations simultaneously, the “input parameters” are common to all of the computations in the sequence of computations. Ex1004, 1:43–52, 2:45–54; Ex1003, ¶218. Similarly, memory 308 (also a first memory bank) “is advantageously used to store data that is expected to

be used in multiple threads, such as coefficients of attribute equations.” Ex1004, 11:11–14. These parameters (equations and coefficients) are common to all computations in the series of computations for an artificial neural network. Ex1003, ¶218.

ii. 6[b]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶219–21.

See 1[d][iv]. In Nickolls/ANN, the system includes a second memory bank (*e.g.*, a logical portion of global register file 306 or local register file 304) 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, the data stored in global register file 306 for the processing cores includes “the input data to be processed by the program.” Ex1004, 13:16–45. Input data may also be stored in a bank of local register file 304 *Id.*, 10:57–11:2. Input data (which would be one or more frames being processed) is specific to at least one computation in the sequence of computations. Ex1006, 623–25; Ex1003, ¶220. Additionally, a logical portion of shared memory 308 may be “used to store data that is expected to be used in multiple threads, such as coefficients of attribute equations.” Ex1004, 11:8–14. Thus, shared memory 308 may also be considered the claimed “second memory bank” because the coefficients of attribute equations are specific to at least one computation (*e.g.*, all computations) in the sequence of computations. Ex1003, ¶220.

Nickolls discloses “lanes” (*i.e.*, logical partitions or banks) within a single memory (*e.g.*, local register file 304), and thus both the claimed “first memory bank” and “second memory bank” may exist in local register file 304. Ex1004, 10:57–11:2. Alternatively, one or more of the claimed memory banks may be in global register file 306 or memory 308. *E.g., id.*, 13:16–45; Ex1003, ¶221.

g. Claim 7

Nickolls/ANN discloses this limitation. Ex1003, ¶¶222–25.

See 1[d][iv]. In Nickolls/ANN, “intermediate results” (*e.g.*, the output of the first or hidden layer 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 Nickolls (*e.g.*, core interface 128, along with instruction unit 312) is configured to transfer outputs from computations from accelerator memory (*e.g.*, local register file 304, global register file 306, and/or memory 308) to main memory (*e.g.*, system memory 104). Ex1004, 20:35–45, 10:37–40, 13:41–45. ANN teaches that such transfers occur without transferring intermediate results. ANN teaches implementing “Render to Texture,” where “intermediate result[s]” are saved in a texture and reused as input data,” rather than being transferred to system memory. Ex1006, 625; Ex1003, ¶¶223–24. ANN teaches that this “decrease[s] the data exchange between CPU and GPU,” *i.e.*, reduces data transfer over the bus

between the CPU and GPU. Ex1006, 625. Thus, in Nickolls/ANN, output data (such as the result of a pass through the artificial neural network) would be transferred from accelerator memory back to the main memory without transferring intermediate results (*e.g.*, the output of a first or hidden layer of the artificial neural network) to reduce data transfer. Ex1003, ¶224.

Additionally, it would have been obvious to a POSITA, including in view of ANN's instruction to "decrease the data exchange between CPU and GPU," to only transfer final outputs to main memory. *Id.*, ¶225. It would not benefit the soccer ball tracking of ANN to send intermediate results (*e.g.*, outputs from intermediate layers) to main memory. *Id.*

h. Claim 8

Nickolls/ANN discloses this limitation. Ex1003, ¶¶226–29.

See 1[d][iv]. In Nickolls/ANN, where the artificial neural network is processing real-time data, the transfer of output data to main memory occurs after the GPU (*e.g.*, processing engines) has begun to perform another sequence of computations of the artificial neural network. The controller of Nickolls (*e.g.*, core interface 128, along with instruction unit 312) is configured to transfer outputs from computations from accelerator memory (*e.g.*, local register file 304, global register file 306, and/or memory 308) to main memory (*e.g.*, system memory 104). Ex1004, 20:35–45, 10:37–40, 13:41–45. For processing of real-time data, as in ANN, where

the output data is from the third (output) layer of the artificial neural network, it is sent to system memory after the processing engine begins performing another sequence of computations (*e.g.*, the computations in the layers of the multi-layer neural network) on new input data. Ex1006, 624–26; Ex1003, ¶¶227–28.

To the extent not disclosed in Nickolls/ANN, it would have been obvious to a POSITA 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. Ex1003, ¶229. Doing so would allow the Nickolls/ANN system to keep up with real-time data input and allow the system to immediately utilize the output data to track a soccer ball in real time. *Id.*

i. Claim 9

Nickolls/ANN discloses this limitation. Ex1003, ¶¶230–31.

See 1[d][iii], 1[d][iv]; Claim 8. The controller of Nickolls (*e.g.*, core interface 128, along with instruction unit 312) is configured to initiate transfer of input data to GPU 122 (*see* 1[d][iii]) and to transfer outputs from computations from accelerator memory to main memory (*see* 1[d][iv]). In Nickolls/ANN, where the artificial neural network is processing real-time data, the transfer of input data and output data referenced above occur in parallel with the GPU (*e.g.*, processing engines) performing another sequence of computations of the artificial neural network. For example, as explained with respect to 1[d][iii], data is continually

transferred into the GPU to track a soccer ball in real time. Thus, as input data is transferred into 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. Ex1006, 624–26; Ex1003, ¶231. Doing so allows the GPU to be “fast enough for the locating of the ball in real time.” Ex1006, 625.

j. Claim 10

Nickolls/ANN discloses this limitation. Ex1003, ¶¶232–33.

See 1[d][ii]. The controller of Nickolls (*e.g.*, core interface 128, alone and with instruction unit 312) “controls operation of core 126,” which includes the GPU (*e.g.*, processing engines). Ex1004, 6:47–49, 14:20–22; *id.*, 13:12–67, 14:17–28, 19:51–61, Fig. 4; Ex1003, ¶233. Each core 126 “includes multiple parallel processing engines that can be used to execute various shader programs” and “can also be leveraged to perform general-purpose computations.” Ex1004, 6:49–55. Instruction unit 312, under the direction of core interface 128, distributes instructions to the processing engines and helps manage synchronization of the engines as they execute instructions. *Id.*, 11:27–29, 12:19–33, 12:43–55, 21:37–55, 23:61–67; *see also* Ex1001, 2:36–37.

k. Claim 11

Nickolls/ANN discloses this limitation. Ex1003, ¶¶234–35.

See 1[a]. ANN discloses using a video camera to acquire input data in real time. Ex1006, 624–26. ANN discloses acquiring the input data in the host system and then transferring the data to the GPU and, in an alternative embodiment, using a “video in” function to directly feed the video into GPU. *Id.*, 626. In Nickolls/ANN, where the video input is acquired in the host system and then transferred to the GPU, the video camera is operably coupled to the CPU. Ex1003, ¶235. Additionally, it would have been obvious to a POSITA to use a video camera to be able to capture the real-time video of a soccer ball, and it would have been obvious to couple the video camera to the CPU for acquiring the data as was known and common for external inputs. *Id.*

i. Claim 12

i. 12[pre]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶236–37.

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]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶238–39.

See 1[c][i]. In Nickolls/ANN, the sequence of computations on the GPU represents layers of an artificial neural network. ANN’s multi-layer neural network

includes an input layer with seven nodes (*i.e.*, neurons), a second (hidden) layer with three nodes, and an output layer of just one node, with each layer representing a layer of neurons in the artificial neural network. Ex1006, 622, 624. For example, in the input and hidden layers, the artificial neural network relies on matrix multiplication and a sigmoid function (*i.e.*, intermediate computations) to determine the output (*i.e.*, an intermediate result). *Id.* 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, ¶239.

iii. 12[a][i]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶240–43.

In Nickolls/ANN, data (both inputs and outputs) are stored in “textures.” Ex1004, 3–4. Thus, outputs from one computational element (*e.g.*, a sequence of computations representing a first layer of the artificial neural network) are mapped to be 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) are mapped to be stored in a “second texture.” *Id.*, 3–4. Each texture is assigned an output variable, which means that the result of the computations performed by a neuron at that layer is designated to be stored in that texture. Ex1003,

¶242. ANN teaches assigning output variables to textures using “Render to Texture” functionality, which “save[s] the intermediate result in a texture on GPU and reuse it as an input data.” Ex1006, 625. At the time of the ’438 patent, “Render to Texture” was a well-known function through which outputs of computational elements (*e.g.*, computations representing a neuron) are stored in a “texture” (*i.e.*, a data array) on the GPU to be used in subsequent computations. Ex1003, ¶242.

iv. 12[a][ii]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶244–45.

See 12[a][i]. For the computations in Nickolls/ANN, in a first time step, the GPU first 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 (*i.e.*, data array), as is done in the “Render to Texture” functionality used in ANN. Ex1006, 624–25. The first texture is then passed as the input for computations at the next layer, where the result is stored in a second texture. *Id.*; Ex1003, ¶245.

v. 12[b]

Nickolls/ANN (and optionally Tamura) discloses this limitation. Ex1003, ¶¶246–47.

See 1[d][iii]; Claim 9. In Nickolls/ANN (and as additionally taught in Tamura), where the artificial neural network is processing real-time data, the transfer of input data and output data referenced above occur in parallel with the GPU (*e.g.*,

processing engines) performing a sequence of computations of the artificial neural network. As explained for 1[d][iii], data (*i.e.*, video frames) is continually transferred into the GPU to track a soccer ball in real time. Thus, as a second portion of input data (*i.e.*, additional video frames) is transferred into the GPU from main memory, the GPU continues to perform computations on prior input data (*i.e.*, prior video frames). Ex1006, 624–26; Ex1003, ¶247.

vi. 12[c]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶248–49.

See 1[d][iv]; Claim 8. The accelerator of Nickolls transfers outputs of computations from accelerator memory to main memory via a bus. Ex1004, 20:35–45. In Nickolls/ANN, where the artificial neural network is processing real-time 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 (*e.g.*, processing engine) 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 ANN (*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. Ex1006, 624–26; Ex1003, ¶249.

vii. 12[d]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶250–51.

See 12[a], 12[b]. In Nickolls/ANN, the GPU (*e.g.*, processing engines) perform 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 as explained with respect to 12[a].

viii. 12[d][i]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶252–53.

See 12[a][i], 12[a][ii]. For the computations in the artificial neural network of ANN, 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. Ex1006, 624–25. 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, ¶253.

ix. 12[d][ii]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶254–57.

See 1[c][i], 12[d][i]. In Nickolls/ANN, 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 ANN’s artificial neural network) during the second time step. Ex1006, 625. Nickolls describes using the output of one computational cycle (an “intermediate result”) as an input for another cycle of computation. Ex1004, 2:33–35, 25:8–10. ANN likewise teaches this technique through the use of “Render to Texture” functionality to “save the intermediate result in a texture on GPU and reuse it as an input data.” Ex1006, 625. As explained with respect to 1[c][ii], Nickolls includes multiple memories in the accelerator, with flexibility in storing input and output data, including memory that can store output data from one thread that is shared with other threads, such that the values of output variables from one thread are accessible to the other computational elements (*e.g.*, threads) during a second time step. Ex1004, 8:14–17, 10:57–11:18, 13:12–25, 20:35–45.

m. Claim 13

Nickolls/ANN discloses this limitation. Ex1003, ¶¶258–59.

The system of Nickolls 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), the system of Nickolls is

designed to store input data in main memory in response to a user interaction (*e.g.*, through a programmer invoking an API). Ex1004, 25:66–26:11.

n. Claim 14

i. 14[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶260–61.

See 3[a].

ii. 14[b]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶262–63.

See 3[b].

o. Claim 16

Nickolls/ANN discloses this limitation. Ex1003, ¶¶264–65.

See Claim 7. In Nickolls/ANN, 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.*, system memory 104) 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. Ex1004, 20:35–45, 10:37–40, 13:41–45; Ex1006, 625.

p. Claim 17

Nickolls/ANN discloses this limitation. Ex1003, ¶¶266–67.

See Claim 8. In Nickolls/ANN, 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.*, system memory 104) after a GPU (*e.g.*, processing engine) begins to perform another sequence of computations (*e.g.*, another pass through the artificial neural network). Ex1004, 20:35–45, 10:37–40, 13:41–45; Ex1006, 624–26.

q. Claim 18

Nickolls/ANN discloses this limitation. Ex1003, ¶¶268–69.

See Claim 9. In Nickolls/ANN, where the artificial neural network on the GPU is processing real-time data, the transfer of output data (including “the second portion of the output data,” *see* 12[c]) occurs in parallel with the GPU (*e.g.*, processing engines) performing another sequence of computations of the artificial neural network. Ex1006, 624–26.

r. Claim 19

Nickolls/ANN discloses this limitation. Ex1003, ¶¶270–71.

See Claim 11.

s. Claim 20

i. 20[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶272–73.

See 6[a].

ii. 20[b]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶274–75.

See 6[b].

t. Claim 21

i. 21[pre]

To the extent the preamble is limiting, Nickolls/ANN discloses it. Ex1003, ¶¶276–77.

See 1[c][i], 12[pre]. ANN discloses a sequence of computations in an artificial neural network. Ex1006, 624–26. The sequence of computations may include one or more passes through the artificial neural network. *Id.*, 625.

ii. 21[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶278–79.

See 1[a]; Claim 11.

iii. 21[b]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶280–81.

See 1[c][ii], 1[d], 1[d][i]. Nickolls discloses a controller (*e.g.*, core interface 128, along with instruction unit 312) coupled to a GPU (*e.g.*, processing engines) and configured to initialize textures and shaders in the accelerator memory (*e.g.*, local register file 304, global register file 306, and/or memory 308), which is also coupled to the GPU.

iv. 21[c]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶282–83.

See 1[b], 1[c], 1[d][iii]. Nickolls discloses memory coupled to the GPU (*e.g.*, local register file 304, global register file 306, and/or memory 308), which receives first input data (*e.g.*, data from a video camera) that was received by the CPU.

v. 21[d]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶284–85.

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

vi. 21[e]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶286–87.

See 1[c][ii], 1[d][iii], 1[d][iv], 12[a]. Nickolls discloses memory coupled to the GPU (*e.g.*, local register file 304, global register file 306, and/or memory 308), which stores first input data (*e.g.*, data from an attached video camera) and first output data (*e.g.*, results of the computations described above).

vii. 21[f]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶288–89.

See 1[d][iii], 1[d][iv], 12[b], 12[c]; Claim 11. In Nickolls/ANN, second input data (*e.g.*, additional data from a video camera) is transferred to memory in the accelerator (*e.g.*, local register file 304, global register file 306, and/or memory 308) 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 ANN, the computations (*e.g.*, matrix multiplication and sigmoid functions) represent outputs of neurons (including a “second neuron”) in layers (including a “second layer”) of a multi-layer artificial neural network. Ex1003, ¶289.

u. Claim 22

Nickolls/ANN discloses this limitation. Ex1003, ¶¶290–91.

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

Nickolls/ANN discloses this limitation. Ex1003, ¶¶292–93.

See 1[d][iv]. In Nickolls/ANN, first output data (*e.g.*, the result of the computations in an output layer of the artificial neural network of ANN) is

transferred from one memory (*e.g.*, local register file 304, global register file 306, or memory 308) to another memory (*e.g.*, a different memory in the accelerator or system memory 104). This transfer occurs while computations continue in the artificial neural network. Ex1003, ¶293.

w. Claim 24

i. 24[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶294–95.

See 12[a]; Claims 16, 21. Nickolls stores intermediate results in memory, including in local register file 304 and global register file 306. Ex1004, 8:14–17, 10:57–11:18, 20:35–45. ANN also discloses storing “intermediate result[s]” in texture memory. Ex1006, 625.

ii. 24[b]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶296–97.

See Claims 7, 16.

x. Claim 25

Nickolls/ANN discloses this limitation. Ex1003, ¶¶298–99.

See 12[b], 12[c]; Claim 9. In Nickolls/ANN, where the artificial neural network is processing real-time data, the transfer of input data and output data occur in parallel. Ex1006, 624–26; Ex1003, ¶299.

y. Claim 26

Nickolls/ANN discloses this limitation. Ex1003, ¶¶300–02.

See 6[a]. 6[a] recites a “first memory bank,” while this limitation recites a “first memory partition.” A POSITA would have understood the “first memory bank” described for 6[a]—*e.g.*, logical portions of local register file 304 or global register file 306 storing the common parameters—is a “first memory partition.” Ex1003, ¶301. For example, Nickolls teaches that “[e]ach processing engine 302 is allocated space in a local register file 304 for storing its local input data, intermediate results, and the like.” Ex1004, 10:57–65 (also referring to the allocated space as a “lane,” *i.e.*, a “partition”). Nickolls further teaches that input data is stored in a specific “local register file space” of shared memory, with a pre-determined, addressable portion for each thread of a processing engine. *Id.*, 8:10–30, 14:41–57. Nickolls thus teaches logically allocating memory partitions to hold specific data for specific processing engines or threads, separate from other data. Ex1003, ¶¶301–02; Ex1001, 5:33–48 (“partitioning” can “be done logically,” and that different portions of a single memory bank can be used to store different types of data, such as “input data, output data, intermediate results, and parameters”).

z. Claim 27

Nickolls/ANN discloses this limitation. Ex1003, ¶¶303–05.

See 6[b]. 6[b] recites a “second memory bank,” while this limitation recites a “second memory partition.” For the same reasons a POSITA would have understood the “first memory bank” above to be a “first memory partition,” he or she would have understood the “second memory bank” of 6[b] to be a “second memory partition.” Ex1003, ¶¶304–05; *see also* 6[a].

aa. Claim 28

Nickolls/ANN discloses this limitation. Ex1003, ¶¶306–09.

In Nickolls/ANN, the second memory partition in GPU 122 (*e.g.*, in either local register file 304 or global register file 306), 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. Ex1003, ¶307.

A partition in local register file 304 or global register file 306 (*i.e.*, memory in GPU 122) can store external input data patterns, such as the input from an external camera. Ex1004, 10:57–65, 13:19–29; Ex1006, 622, 625–26; 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 ANN’s multi-layer neural network, which are used as an input for the next layer. Ex1004, 2:33–35, 10:57–65, 25:8–10;

Ex1006, 625; Ex1001, 5:60–62 (“partition ... to hold the data textures representing internal variables”). As explained for 1[c][ii] and 1[d][iii], local register file 304 or global register file 306 also stores inputs and outputs of the computations. Ex1004, 8:14–30, 10:57–11:2, 13:16–25, 19:44–50. Nickolls is flexible in where data is and can be stored, for example, allowing data to be stored in either local register file 304 or global register file 306 so it can be accessed by multiple threads. Ex1004, 4:29–31, 10:57–11:18, 20:35–45; Ex1003, ¶308. As explained with respect to Claim 26, Nickolls teaches that data is stored in a specific “local register file space” of shared memory, and allows logically allocating memory partitions to hold specific data for specific processing engines or threads, separate from other data. Ex1004, 4:29–31, 10:57–11:18, 20:35–45; Ex1003, ¶308. As such, Nickolls discloses that all of these elements can be stored in the “second memory partition.” Ex1003, ¶308.

Additionally, it would have been obvious to a POSITA to store each of the claimed parameters in Claim 28 in the “second memory partition.” Nickolls discloses various memories in GPU 122 in which data and other parameters can be stored, including local register file 304 or global register file 306, and a POSITA would have been able to and motivated to use a partition in any of those memories. Ex1004, 4:29–31, 10:57–11:18, 20:35–45; Ex1003, ¶309. Because each of these pieces of data are used or output by a single thread, 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 a POSITA to try storing these parameters in the “second memory partition.” *Id.*

bb. Claim 29

Nickolls/ANN discloses this limitation. Ex1003, ¶¶310–11.

See 12[a][i], 12[a][ii]. In Nickolls/ANN, the GPU first executes the computations in a first layer and stores the result (*i.e.*, an output of the computational elements) in memory. Ex1006, 624–25.

cc. Claim 30

i. 30[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶312–13.

See 12[a][i], 12[a][ii]. In Nickolls/ANN, 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. Ex1006, 624–25.

ii. 30[b]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶314–15.

See 12[d][i], 12[d][ii]. In Nickolls/ANN, after a first output is stored in memory, the first output is passed as an input for computations in a subsequent layer. Ex1006, 624–25. Nickolls additionally describes the GPU (*e.g.*, processing engines) using the output of one computational cycle as an input for another cycle of computation. Ex1004, 2:33–35, 25:8–10.

dd. Claim 31

Nickolls/ANN discloses this limitation. Ex1003, ¶¶316–17.

See 1[c][i], 12[a][i]. ANN’s multi-layer neural network includes an input layer with seven nodes (i.e., neurons), a second (hidden) layer with three nodes, and an output layer of just one node, with the computations in each layer representing a layer of neurons in the artificial neural network. Ex1006, 624.

ee. Claim 32

Nickolls/ANN discloses this limitation. Ex1003, ¶¶318–19.

In Nickolls/ANN, 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. Ex1006, 625; Ex1003, ¶319. Nickolls explains that in its SIMD architecture, all threads in the GPU (*e.g.*, processing engines) execute the same shader (*i.e.*, the same equations). Ex1004, Abstract, 2:22–27, 5:31–36, 11:28–30.

ff. Claim 33

Nickolls/ANN discloses this limitation. Ex1003, ¶¶320–21.

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, ¶321.

gg. Claim 34

Nickolls/ANN discloses this limitation. Ex1003, ¶¶399–400.

During training of the artificial neural network of ANN, input data would have been loaded from disk (*e.g.*, a hard drive). Ex1006, 624 (referring to the “train set”). Nickolls discloses system disk 114, which can store input data. Ex1004, 5:31–44, 6:9–34, Fig. 1. Additionally, it would have been obvious to load input data from disk. Before the artificial neural network in ANN would be able to track a soccer ball, it would be trained on existing data. Ex1006, 624–25. It would have been obvious to and within the ordinary skill of a POSITA to store the training data on a disk drive and then load it into the accelerator.

hh. Claim 40

i. 40[pre]

To the extent the preamble is limiting, Nickolls/ANN discloses it. Ex1003, ¶¶322–23.

See 21[pre].

ii. 40[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶324–25.

See 1[a]; Claim 11.

iii. 40[b]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶326–330.

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.” Nickolls teaches computing an address location in the shared memory from which data will be read. Ex1004, 8:22–25, 20:46–60. The variable holding the computed address is a “pointer” to the first partition. Ex1003, ¶327; Ex1001, 6:16–18 (referring to “address pointers”), 11:38–40 (referring to an “ID” of a texture (array) as a “pointer”). Similarly, where input data is stored in local register file 304, the system would maintain a “memory address” to identify the location of data, which is implemented as a “pointer” to a first partition. Ex1004, 14:45–48, 3:54–62; Ex1003, ¶327. 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, ¶327.

Additionally, referencing memory by using pointers would have been obvious, including in further view of GPU Gems. Prior to the ’438 patent, pointers were a fundamental construct of computer programming to efficiently manage memory and data access. Ex1003, ¶328; Ex1009, 88; Ex1032, 499–508.

GPU Gems, for example, teaches storing first input data in a first partition, referenced by a first pointer, before computations begin. Ex1003, ¶329. 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 POSITA would have understood that,

in the context of the '438 patent, a “buffer” is a “partition” because it a logical allocation of memory to hold specific data separate from other data. Ex1003, ¶329. 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
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);
}
```

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, ¶330. Given the widespread use of pointers in managing data, including in GPU Gems, their application to the parallel GPU processing in Nickolls/ANN would have been a routine choice for skilled practitioners in the field. *Id.*, ¶330.

iv. 40[c]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶331–32.

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

v. 40[d]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶333–34.

See 1[d].

vi. 40[d][i]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶335–40.

See 12[a][i], 21[d], 40[b]. Output data from the processing unit (*e.g.*, processing engine) is stored in a “second partition” of memory (*e.g.*, a partition in local register file 304, global register file 306, and/or memory 308). As explained with respect to 40[b], a POSITA would have understood the partition of memory would be referenced by a pointer (or it would have been obvious to do so).

In Nickolls/ANN, the first output data from a layer of the artificial neural network becomes an input for a second layer of neurons. Ex1006, 625; Ex1004, 2:33–35, 25:8–10. A POSITA would have understood that this would have been achieved by swapping the first pointer, which was a known technique to efficiently swap inputs for outputs. Ex1003, ¶337.

It would have been obvious to a POSITA to implement an algorithm on the hardware in Nickolls in which the first and second pointers are swapped at the end of a computational cycle, including in further view of GPU Gems. Because Nickolls and ANN describe using intermediate results as inputs to a new computational cycle (*see supra*), and because both disclose using pointers, a POSITA would have been motivated to use pointer swapping to effectuate this efficiently (*e.g.*, by avoiding copying large chunks of memory). Ex1003, ¶338. Pointer swapping as a technique

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 (disclosing pointer swapping in computations on GPUs). For example, 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 ANN’s neural network on the system of Nickolls 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, ¶¶335–40.

vii. 40[d][ii]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶341–42.

See 1[d][iii], 1[d][iv], 12[c]. In Nickolls/ANN, 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]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶343–44.

The controller of Nickolls (*e.g.*, core interface 128, alone and in combination with instruction unit 312) dictates an order of instructions to the processing engines. For example, core interface 128 includes a “launch” module that “selects” a “first SIMD group to be launched,” and then loads the instructions to locations corresponding to those threads. Ex1003, 19:51–61, 17:37–41, 17:63–18:5. Additionally, instruction unit 312 includes “[a]rbitration logic 320 and multiplexer 318 [to] determine the order in which instructions are fetched,” as well as “selection logic 1110 that selects a next instruction to issue.” *Id.*, 12:43–55, 21:37–52, 23:61–67. In Nickolls/ANN, the instructions to the processing unit include instructions to perform computations in all layers of neurons in the artificial neural network. Ex1006, 622–25; Ex1003, ¶344.

ii. Claim 41

Nickolls/ANN discloses this limitation. Ex1003, ¶¶345–46.

See 1[c][i].

jj. Claim 42

Nickolls/ANN discloses this limitation. Ex1003, ¶¶347–48.

See 1[d][ii]. Core interface 128 of Nickolls, alone and with instruction unit 312, “controls operation of core 126,” including distributing instructions to the processing engines. Ex1004, 6:47–49, 11:27–29, 12:19–33, 12:43–55, 13:12–67, 14:20–22, 19:51–61.

kk. Claim 43

i. 43[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶349–51.

See 6[a]; Claim 28 (describing storing specific information in “partitions” in Nickolls/ANN). Nickolls discloses a “third partition” (*e.g.*, a logical partition in memory in GPU 122) for storing “internal variables.” For example, the data stored in shared memory 306 for the processing cores includes “input parameters.” Ex1004, 13:16–45, 16:63–17:3, 24:39–47. The GPU memory can also store representations of internal variables, such as outputs of a hidden layer in ANN’s multi-layer neural network, which are used as an input for the next layer. *Id.*, 2:33–35, 10:57–65, 25:8–10; Ex1006, 625. A POSITA would have also understood these

parameters to be “internal variables” stored in a “third partition.” Ex1003, ¶¶350–51. Additionally, 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]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶352–54.

See 6[b]; Claim 28. Nickolls discloses a “fourth partition” (*e.g.*, a logical partition in memory in GPU 122) for storing “data used as input at a particular layer of neurons of the artificial neural network.” For example, global register file 306 stores “the input data to be processed by the program,” and the logical portion storing such information would be a “partition” as explained for Claim 28. Ex1004, 13:16–25. Additionally and alternatively, a partition of shared memory 308 may be “used to store data that is expected to be used in multiple threads, such as coefficients of attribute equations,” which are data used as input at a particular layer of neurons (*e.g.*, weights used at a particular layer). *Id.*, 11:8–14. 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, ¶¶353–54.

II. Claim 44

i. 44[pre]

To the extent the preamble is limiting, Nickolls/ANN discloses it. Ex1003, ¶¶355–56.

See 1[pre].

ii. 44[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶357–58.

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

iii. 44[b]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶359–60.

See 1[b].

iv. 44[c]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶361–62.

See 1[c].

v. 44[c][i]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶363–64.

See 1[c][i].

vi. 44[c][ii]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶365–66.

See 1[c][ii].

vii. 44[d]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶367–68.

See 1[d].

viii. 44[d][i]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶369–70.

See 1[d][iii].

ix. 44[d][ii]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶371–72.

See 1[d][iv].

x. 44[d][iii]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶373–74.

See 1[d][ii].

mm. Claim 45

Nickolls/ANN discloses this limitation. Ex1003, ¶¶375–76.

See Claim 2.

nn. Claim 46

i. 46[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶377–78.

See 3[a], Claim 44.

ii. 46[b]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶379–80.

See 3[b].

oo. Claim 47

Nickolls/ANN discloses this limitation. Ex1003, ¶¶381–82.

See Claim 4.

pp. Claim 48

Nickolls/ANN discloses this limitation. Ex1003, ¶¶383–84.

See Claim 5.

qq. Claim 49

i. 49[a]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶385–86.

See 6[a]; Claims 26, 44.

ii. 49[b]

Nickolls/ANN discloses this limitation. Ex1003, ¶¶387–88.

See 6[b]; Claim 27.

rr. Claim 50

Nickolls/ANN discloses this limitation. Ex1003, ¶¶389–90.

See Claim 7.

ss. Claim 51

Nickolls/ANN discloses this limitation. Ex1003, ¶¶391–92.

See Claim 8.

tt. Claim 52

Nickolls/ANN discloses this limitation. Ex1003, ¶¶393–94.

See Claim 9.

uu. Claim 53

Nickolls/ANN discloses this limitation. Ex1003, ¶¶395–96.

See Claim 10.

vv. Claim 54

Nickolls/ANN discloses this limitation. Ex1003, ¶¶397–98.

See Claim 11.

V. ENABLEMENT AND SECONDARY CONSIDERATIONS

The prior art in this Petition is enabled. Ex1003, ¶¶136, 139, 143, 147.

Petitioner is not aware of any secondary considerations. *Id.*, ¶401. To the extent 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 respectfully 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
1700 M Street, 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 13,986 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
1700 M Street, 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
1700 M Street, 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.