

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

NVIDIA CORPORATION,  
Petitioner,

v.

NEURAL AI, LLC,  
Patent Owner.

---

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

---

**Declaration of Tajana S. Rosing, Ph.D.  
in Support of Petitioner in the  
Petition for *Inter Partes* Review of U.S. Patent No. RE48,438**

## TABLE OF CONTENTS

|   | <u>Page</u> |
|---|-------------|
| I. INTRODUCTION .....   | 1           |
| II. BASIS FOR OPINIONS .....  | 2           |
| A. Qualifications .....   | 2           |
| III. LEGAL UNDERSTANDINGS.....  | 5           |
| A. A Person of Ordinary Skill in the Art .....                        | 5           |
| B. General Claim Construction Principles.....                         | 7           |
| C. Obviousness.....   | 9           |
| IV. BACKGROUND TECHNOLOGY.....  | 11          |
| A. GPUs for General Purpose Computing .....                           | 12          |
| 1. History of Graphics Processing Units.....                          | 13          |
| 2. Development of GPUs for General Purpose Computing.....             | 15          |
| B. Known Features of GPUs Prior to the '438 Patent .....              | 17          |
| 1. Prior Art Parallel Computing and “Controllers” .....               | 17          |
| 2. The Stream Programming Model of GPUs.....                          | 19          |
| 3. APIs and Programming GPUs .....                                    | 21          |
| 4. Known Memory Management Techniques.....                            | 24          |
| 5. Artificial Neural Networks and Other Applications of<br>GPGPU..... | 31          |
| 6. Real-Time Data Input.....  | 34          |
| 7. Task and Resource Management on a GPU .....                        | 35          |
| V. THE '438 PATENT.....   | 38          |
| A. Summary of the '438 Patent.....                                    | 38          |
| B. Summary of the Prosecution History of the '438 Patent .....        | 46          |
| C. Overview of the Challenged Claims .....                            | 51          |
| VI. PRIOR ART CONSIDERED .....  | 61          |
| A. General Overview of the Primary Prior Art References .....         | 62          |
| 1. Kirk .....   | 62          |

|      |  |     |
|------|--|-----|
| 2.   | Oh.....  | 67  |
| 3.   | Tamura .....   | 69  |
| 4.   | GPU Gems .....   | 71  |
| VII. | ANALYSIS OF THE CHALLENGED CLAIMS AGAINST THE<br>PRIOR ART.....  | 74  |
| A.   | Claim Construction.....  | 74  |
| B.   | Application of the Prior Art to the Claims .....   | 75  |
| C.   | Grounds 1–4: Kirk and Oh, Additionally in Combination With<br>Tamura and GPU Gems, Render Obvious Claims 1–14, 16–34,<br>and 40–54 .....   | 76  |
| 1.   | Combination of Kirk and Oh .....   | 77  |
| 2.   | Combination of Tamura with Kirk and Oh.....  | 81  |
| 3.   | Element-by-Element Analysis .....  | 83  |
| a)   | Claim 1.....   | 83  |
| b)   | Claim 2: “The computer system of claim 1, wherein the<br>central processing unit is configured to receive the input<br>data in response to a user interaction.” .....  | 110 |
| c)   | Claim 3.....   | 111 |
| d)   | Claim 4: “The computer system of claim 1, wherein the<br>main memory is configured to store a copy of the output<br>data stored in the accelerator memory.” .....  | 112 |
| e)   | Claim 5: “The computer system of claim 1, wherein an<br>output of at least one computation in the sequence of<br>computations represents an output of at least one neuron<br>in an artificial neural network.” .....   | 113 |
| f)   | Claim 6.....   | 114 |
| g)   | Claim 7: “The computer system of claim 1, wherein the<br>controller is configured to transfer the output data from<br>the accelerator memory to the main memory without<br>transferring any of the intermediate results from the<br>accelerator memory to the main memory so as to reduce<br>data transfer via the bus.” ..... | 116 |

|    |   |     |
|----|---|-----|
| 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.”<br>.....  | 117 |
| 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.” ..... | 120 |
| 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.” .....   | 121 |
| 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.” .....   | 121 |
| l) | Claim 12.....   | 123 |
| m) | Claim 13: “The method of claim 12, further comprising: storing the input data in the main memory in response to a user interaction.” .....  | 130 |
| n) | Claim 14.....   | 131 |
| 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.” ..... | 131 |
| 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 memory   |     |

|     |   |     |
|-----|---|-----|
|     | after the GPU has begun to perform another sequence of computations.” .....   | 132 |
| 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.” . | 132 |
| 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.” .....                 | 133 |
| s)  | Claim 20.....   | 133 |
| t)  | Claim 21.....   | 134 |
| u)  | Claim 22: “The method of claim 21, further comprising: transferring the second input data comprises transferring the second input data via a bus operably coupled to the CPU.” .....  | 138 |
| 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.” .....                                 | 138 |
| w)  | Claim 24.....   | 139 |
| x)  | Claim 25: “The method of claim 23, wherein transferring the second input data and transferring the first output data occurs in parallel.” .....   | 139 |
| 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.” .....                                   | 140 |
| 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.” .....  | 141 |
| 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.” .....  | 141 |
| 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.” ..... | 143 |
| cc) | Claim 30.....  | 144 |
| 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.” .....                                      | 145 |
| ee) | Claim 32: “The method of claim 31, wherein all neurons in the layer of neurons are described by the same equation.” .....  | 145 |
| 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.” .....  | 146 |
| gg) | Claim 40.....  | 146 |
| hh) | Claim 41: “The system of claim 40, wherein the controller is configured to send instructions for performing the at least one calculation to the processing unit.” .....  | 154 |
| ii) | 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.” .....  | 154 |
| jj) | Claim 43.....  | 155 |
| kk) | Claim 44.....  | 157 |
| ll) | 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.”.....   | 160 |

|     |  |     |
|-----|--|-----|
| mm) | Claim 46.....  | 160 |
| nn) | 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.”.....   | 161 |
| oo) | 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.”.....   | 161 |
| pp) | Claim 49.....  | 161 |
| qq) | 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.”.....                                       | 162 |
| rr) | 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.”   | 162 |
| ss) | 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.” ..... | 162 |
| tt) | 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.” .....  | 162 |
| uu) | 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.”.....  | 163 |

|       |   |     |
|-------|---|-----|
| vv)   | Claim 34: “The method of claim 21, further comprising:<br>loading the second input data from disk.” ..... | 163 |
| VIII. | SECONDARY CONSIDERATIONS .....  | 164 |
| IX.   | RIGHT TO SUPPLEMENT .....   | 164 |
| X.    | CONCLUSION.....   | 165 |

## 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    | RESERVED  |
| 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)                                  |

| Exhibit | Description   |
|---------|---|
| 1022    | Youquan Liu, <i>Real-Time 3D Fluid Simulation on GPU with Complex Obstacles</i> (2004)  |
| 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</i> Stipulation   |
| 1032    | GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (2005)                       |

I, Dr. Tajana S. Rosing, Ph.D., do hereby declare as follows.

## **I. INTRODUCTION**

1. I am making this Declaration at the request of NVIDIA Corporation, (“Petitioner” or “NVIDIA”) in the matter of the *Inter Partes* Review of U.S. Patent No. RE48,438 (“the ’438 patent”).

2. I am being compensated for my work in this matter at my standard hourly rate. I am also being reimbursed for reasonable and customary expenses associated with my work and testimony in this proceeding. My compensation is not contingent on the outcome of this matter or the specifics of my testimony.

3. I have been asked to provide my opinions regarding whether the subject matter of claims 1–14, 16–34, and 40–54 (“the Challenged Claims”) of the ’438 patent would have been obvious to a person having ordinary skill in the art (“POSITA”) at the time of the alleged invention, in light of the prior art. It is my opinion that the Challenged Claims would have been obvious to a POSITA.

4. In forming the opinions expressed below and preparing this Declaration, I have considered: the challenged ’438 patent; its file history; each of the exhibits identified in the list of exhibits; the relevant legal standards, including the standard for obviousness; and my own knowledge and experience based upon my work related to the field of the ’438 patent.

5. Unless otherwise noted, all **emphasis** in any quoted material has been added. Claim terms are *italicized*.

## II. BASIS FOR OPINIONS

### A. Qualifications

6. My complete qualifications and professional experience are described in my Curriculum Vitae, a copy of which can be found as Appendix A. The following is a brief summary of my relevant qualifications and professional experience. Among other things, my CV includes a list of my issued patents and publications for at least the last 30 years.

7. I am a distinguished professor, IEEE and ACM Fellow, and the Fratamico Endowed Chair in the Computer Science and Engineering Department at the University of California, San Diego (UCSD), and adjunct distinguished professor in the Electrical and Computer Engineering Department at UCSD. My research focuses on energy-efficient computing. My place of business is located at 9500 Gilman Drive, La Jolla, California. I am over the age of eighteen, and am a citizen of the United States.

8. I am currently a Director of System Energy Efficiency Lab where I am leading a diverse research team on projects relating to system energy-efficiency. I have over thirty years of academic and industry experience in applying, designing, studying, teaching, and writing about energy-efficient computing, including

datacenters and their interaction with smart grid and renewable energy resources. My experience has spanned both hardware and software, along with measurements and simulations related to datacenters, smart grid, renewable energy resources, power, thermal and reliability management (for which I was nominated both IEEE and ACM Fellow).

9. I received an Electrical Engineering Ph.D. degree in 2001 from Stanford University; my thesis was titled “Energy Efficient System Design and Utilization.” Before that, I received an Electrical Engineering B.S. degree in 1992 from Northern Arizona University, an Electrical and Computer Engineering M.S. degree in 1993 from the University of Arizona, and an Engineering Management M.S. degree in 2000 from Stanford University.

10. While at Stanford University, I worked in the same office and on the same machines where Yahoo was started, and just down the hall from the office where Google was started. After completing my M.S. in Electrical and Computer Engineering, I worked at Altera Corporation, which is now part of Intel Corporation, as a senior design engineer for four years. During and after completing my Ph.D. degree, I worked with Stanford University and Hewlett-Packard Labs leading a team of researchers to develop products for the wireless portable devices market. At HP Labs, my team’s efforts also included work related to content delivery networks,

streaming video over wireless, videoconferencing using mobile platforms, and energy efficiency of chips, mobile as well as server systems.

11. Through the research and work experience described above, I have significant experience in the area of computer systems and architecture, including the use of graphic processing units (GPUs), as well as with artificial neural networks. I have many publications related to accelerating machine learning algorithms, including neural networks, using GPUs, FPGAs, ASICs and in/near memory/storage computing devices. This work has led to me winning \$56.5M from DARPA and Semiconductor Research Corporation JUMP2.0 center grant that focuses on next generation computing architecture which leverage processing in and near memory for big data workloads such as machine learning. The center I am leading, PRISM, Processing with Intelligent Storage and Memory, has 24 PIs and more than 230 PhD students across 13 top universities in the USA. Furthermore, I am a co-PI of DARPA/SRC JUMP2.0 CoCoSys center which focuses on acceleration and design of next generation cognitive computing architectures, which push the envelope of machine learning into neuromorphic, symbolic, stochastic and hyperdimensional computing. Through my work on Hyperdimensional computing, a new form of machine learning, and its acceleration on GPUs, FPGAs, ASICs and in/near memory/storage, I was awarded a top University Researcher for Design in 2022, an award given by the Semiconductor Industry Association which represents 99% of

the world's semiconductor industry by revenue. This award is given to only one individual per year.

12. My publication history is detailed in my attached CV. I have published over 400 publications and received a number of best paper awards and nominations. I have also been an invited speaker at numerous academic and industry conferences. I am an inventor on several U.S. and foreign patents related to my work.

13. Based on my experience and education, I believe that I am qualified to opine as to the knowledge and level of skill of a POSITA at the time of the '438 patent (which I further describe below), in addition to the general state of the art during that time.

### **III. LEGAL UNDERSTANDINGS**

14. In this section, I describe my understanding of certain legal standards that I may have relied upon in forming my opinions set forth in this Declaration. I have been informed of these legal standards by Petitioner's attorneys. Although I am not an attorney and have not researched the law on patent invalidity, I have formed the following understanding about the legal standards.

#### **A. A Person of Ordinary Skill in the Art**

15. I understand there are multiple factors relevant to determining the level or ordinary skill in the pertinent art, including (1) the levels of education and experience of persons working in the field at the time of the invention; (2) the

sophistication of the technology; (3) the types of problems encountered in the field; and (4) the prior art solutions to those problems.

16. I further understand that the hypothetical POSITA is presumed to have knowledge of all references that are sufficiently related to one another and to the pertinent art, and to have knowledge of all arts reasonably pertinent to the particular problem that the claimed invention purports to address.

17. In my opinion, as it relates to the '438 patent, a POSITA on or around September 25, 2006, would have had at least a bachelor's degree in electrical or computer engineering (or an equivalent discipline) and at least two years' worth of experience in research, design and/or development of computer systems, including experience with graphical processing units. Additional education could substitute for experience, and vice versa. For example, a master's degree or higher would have been acceptable in place of the work experience described above.

18. I meet the criteria described above and believe myself to be a person with at least ordinary skill in the art pertaining to the '438 patent. I would have qualified as such a person by at least September 25, 2006, which I understand is the date the provisional application to which the '438 patent was filed claims priority.

19. For the purposes of this Declaration, in general, and unless otherwise noted, my statements and opinions, such as those regarding my own experience and what a POSITA would have understood or known generally (and specifically related

to the references I consulted herein), reflect the knowledge that existed in the relevant field as of the priority date of the '438 patent.

20. As I discuss in detail below, the '438 patent relates to the use of GPUs to accelerate parallel computations, which is relevant to many fields that perform numerical simulations, including artificial neural networks. The '438 patent explains that “[n]umerical simulations, e.g., finite element analysis, of large systems of similar elements (e.g. neural networks, genetic algorithms, particle systems, mechanical systems) . . . can benefit from GPGPU computation.” Ex1001, 1:51–55. In my opinion, prior to the '438 patent, a POSITA interested in this subject matter would have searched for books and publications by keywords such as “graphics processing unit,” “GPU,” “general-purpose GPU,” “GPGPU,” or “neural networks.”

### **B. General Claim Construction Principles**

21. I understand that there are two types of claims: independent claims and dependent claims. I understand that an independent claim stands alone and includes only the limitations it recites. I understand that a dependent claim is a claim that depends on another claim. I understand that a dependent claim includes all of the limitations recited in the dependent claim as well as any limitations included in the corresponding independent claim.

22. I understand that before determining whether the Challenged Claims are invalid, the claims must be construed on a limitation-by-limitation basis. I

understand that claim terms are to be afforded their plain and ordinary meaning as used in the claims and as would be understood by a POSITA at the time of the invention. I understand that this person skilled in the art at the time of the invention is assumed to have read the claim terms in light of the entire patent record, including the other claims, the patent specification, and the patent prosecution history. I have interpreted the claim terms in the '438 patent in light of this understanding.

23. I understand that to properly understand the meaning of claim terms, one should first consider the intrinsic evidence, which includes the claim language itself, the patent specification, and the patent's prosecution, as well as any post-issuance history before the Patent Office. For example, the patent specification may show that the inventor used words or terms in a manner inconsistent with their plain and ordinary meaning. Specifically, I understand that where the specification reveals a special definition given to a claim term by the inventor that differs from the meaning it would otherwise possess, the inventor's lexicography governs. I understand that the prosecution history of the patent may also provide guidance in construing a claim term. For example, the prosecution history may show that the patent applicant might have limited the scope of some or all of the claims during prosecution, either affirmatively or by implication.

24. I understand that if the intrinsic evidence is not conclusive regarding the meaning of a particular claim term, extrinsic evidence may also be used to

determine its meaning. I understand that extrinsic evidence may be used, for example, to help determine what a person skilled in the art at the time of the invention would have understood the claim term to mean. Extrinsic evidence may include dictionaries, technical treatises and books, articles, or expert testimony.

### **C. Obviousness**

25. I understand that for a claim to be invalid as obvious, the party asserting invalidity must identify prior art references that alone or in combination with other prior art references would have rendered the claim obvious to one of ordinary skill in the art at the time of the invention. I understand that the analysis of obviousness looks at (1) the scope and content of the prior art; (2) the differences between the claims and the prior art; (3) the level of ordinary skill in the art; and (4) secondary considerations of non-obviousness.

26. I understand that a single prior art reference can render a claim obvious where it would have been obvious to modify that reference to arrive at the patented invention. I understand that it is appropriate to look to the knowledge of a POSITA in assessing whether it would have been obvious to modify a single prior art reference to produce the claimed invention. I also understand that a conclusion of obviousness may be based on a combination of prior art references. I understand that it may be helpful to identify a reason that would have prompted a person of ordinary

skill in the relevant field to combine the elements in a way that the claimed invention does.

27. I understand that in determining whether a combination of prior art references renders a claim obvious, it may be helpful to consider whether there is some teaching, suggestion, or motivation to modify or combine the references and whether there would have been a reasonable expectation of success in doing so.

28. I understand that there are several recognized rationales for combining references or modifying a reference to show obviousness. These rationales include: (a) combining prior art elements according to known methods to yield predictable results; (b) simple substitution of one known element for another to obtain predictable results; (c) use of a known technique to improve a similar device (method, or product) in the same way; (d) applying a known technique to a known device (method, or product) ready for improvement to yield predictable results; (e) choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and (f) some teaching, suggestion, or motivation in the prior art that would have led a POSITA to modify the prior art or to combine prior art teachings to arrive at the claimed invention.

29. I understand that in evaluating whether patent claims are invalid as obvious, secondary considerations of non-obviousness are considered, including considerations such as commercial success, long-felt but unsolved needs, copying,

praise, teaching away from the claimed invention by the prior art, unexpected results, industry acceptance, failure of others, and skepticism by experts. I understand that the secondary considerations are considered with the balance of obviousness evidence in the record to act as a check against impermissible hindsight bias. I further understand that there must be a nexus between the merits of the claimed invention and the evidence of secondary considerations in order to give such considerations substantial weight in the determination of obviousness.

30. I understand that the burden of presenting evidence of secondary considerations is on the patentee. I also understand that the proponent of the evidence of secondary considerations bears the burden of showing that a nexus exists between the claimed features of the invention and the evidence offered to show non-obviousness.

#### **IV. BACKGROUND TECHNOLOGY**

31. As I have explained in more detail in Section V, the '438 patent relates to a “graphic processor based accelerator system and method.” Ex1001, Title. The '438 patent claims recite systems and methods for performing “a sequence of computations representing an artificial neural network.” *Id.*, 14:50–21:9. In particular, the claims focus on the timing of transfer of input data acquired from an external system in real time to GPU memory. *Id.* At a high level, the CPU executes receives input data acquired from an external system in real time and transfers this

input data into GPU memory after the GPU begins a first computation in the sequence of computations, but before it begins a second computation in the sequence. *Id.*, 17:16–45. In some claims, this transfer must occur “in parallel with performing the sequence of computations,” which would “reduc[e] the impacts of this transfer” on the computational efficiency and performance of the system. *Id.*, 14:14–20, 16:24–27. The claims also relate to techniques for memory handling, such as swapping pointers to input and output data to facilitate interdependencies between the neural network layers. To put in context my opinions on the invalidity of the ’438 patent, in this section I provide a brief discussion of the relevant technological background.

32. I begin by discussing the history of GPUs and their use for numerical computations (such as for artificial neural networks), an area in which Petitioner NVIDIA and others were at the forefront long before the ’438 patent. I then describe the ideas and methodologies claimed in the ’438 patent for performing numerical simulations and computations, such as simulating artificial neural networks, on GPUs that were well known and practiced in the art before the ’438 patent.

#### **A. GPUs for General Purpose Computing**

33. Fundamentally, computers are machines that perform mathematical operations on input data to create output data. Every task a computer performs—whether it’s processing text, rendering an image, or running complex simulations—

boils down to executing calculations based on binary logic. This fundamental capability makes computers inherently suited for mathematical and scientific computations. Thus, from the earliest days of computers to now, scientists and engineers have relied on computers' speed and accuracy to solve difficult problems.

34. Originally, numerical computations were handled by central processing units (CPUs), which excel at executing instructions sequentially, *i.e.*, one step at a time. While they were highly effective for many applications, the CPU's ability to perform large-scale numerical simulations was limited by its sequential nature, leading researchers to explore alternative hardware solutions for faster and more efficient computation.

35. Parallel processing, both for graphics and general-purpose computing, was known for decades before the '438 patent. In 1999, NVIDIA developed the first consumer-level graphics card that could offload essentially the entire graphics pipeline from the CPU, dubbing it a "Graphics Processing Unit," or "GPU." 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.

## **1. History of Graphics Processing Units**

36. A graphics processing unit (GPU) is an integrated circuit with computer hardware designed for graphics. GPUs were developed in the 1990s by NVIDIA to

accelerate graphics rendering. The '438 patent recognizes that “[v]arious brand of GPU [we]re relevant” at the time, including “GPU’s based on the GeForce series from NVIDIA Corporation.” Ex1001, 4:40–42. NVIDIA released the world’s first programmable consumer-level GPU, the GeForce 256, in 1999. Ex1015, 1. At the time of release, the GeForce 256 had the equivalent parallel processing performance of a “maximum configuration 256-processor Cray T3D,” one of the leading parallel processing systems as the time. *Id.* NVIDIA developed and released several other GPUs prior to 2005, including the GeForce 2 (2000), GeForce 3 (2001), GeForce 4 (2002), GeForce FX (2003), and GeForce 6 (2004) GPUs. *E.g.*, Ex1032, 471–91 (“GeForce 6 Series GPUs provide the GPU programmer with unparalleled flexibility and performance . . . .”); *see also* Ex1017, 10–11 (describing the “generations” of GPUs).

37. Unlike CPUs, which were optimized to handle tasks sequentially, GPUs were designed with parallel processing in mind. GPUs were originally intended to leverage their parallel processing architecture to accelerate graphics rendering in desktop computers, particularly real-time rendering of three-dimensional environments. Ex1015, 1.

38. At a high level, traditional GPUs processed “textures” (stored as data arrays) by using “shaders” (programs) to modify pixel data and create graphics for display. *See* Ex1032, 500–01. Shaders operated in a highly parallel manner,

executing mathematical operations on individual texture elements (sometimes called texels or pixels) to perform tasks like shading, filtering, lighting computations, and blending. These operations were ideal for parallelism because the computations for each element were typically independent of others, allowing GPUs to process thousands of elements simultaneously.

39. After their release, GPUs quickly evolved to include programmable shaders, replacing the older fixed-function pipeline. This advancement gave developers the ability to write custom shader programs, enabling them to manipulate graphics at a much more granular level. For example, programmable shaders allowed for more complex visual effects, such as per-pixel lighting, procedural textures, and realistic surface details, revolutionizing graphics rendering in video games, simulations, and visual effects. These innovations also laid the groundwork for general-purpose computing on GPUs (GPGPU), discussed in more detail below, enabling GPUs to handle non-graphics tasks like scientific and numerical simulations and machine learning.

## **2. Development of GPUs for General Purpose Computing**

40. Not long after GPUs were released for graphics processing, some in the industry—including many at and supported by NVIDIA—realized the power of GPUs as versatile, low-cost parallel computing platforms. Because graphics processing is an inherently parallelizable task, GPUs were designed with parallelism

baked into their structure. *See* Ex1032, 457–70. Additionally, GPUs did not need to support as many different sorts of functions and datatypes as general-purpose CPUs; by omitting the circuitry needed to support these features, GPU circuits were able to fit more processing power onto a single chip die an equivalent CPU.

41. GPUs were used almost immediately to perform parallelized computations for non-graphics applications, such as numerical simulations. In the early 2000s, this type of computing was often referred to as “general-purpose computing on GPUs,” sometimes referred to as “GPGPU.”

42. GPGPU works by using “textures” to store data and repurposing “shaders” to perform computations on the data. Ex1032, 497–502. The techniques enabled GPUs to handle tasks like scientific computing, numerical simulations, and neural network training. NVIDIA and others led GPGPU research and development. For example, NVIDIA not only improved hardware and software for programmable GPUs—such as the designs described in the Nickolls and Kirk prior art references discussed in this Declaration—but also actively supported the GPGPU community through educational efforts. NVIDIA published books like GPU Gems, which provided practical guidance and techniques for using GPUs in general-purpose applications, helping developers harness the parallel processing power of GPUs for a wide range of fields. *See id.*, 451 (“General-Purpose Computation on GPUs: A Primer”). One recognized benefit of GPGPU applications was that, with the

architecture described above, all calculations occurring in parallel on the GPU, they “free up the CPU for other tasks.” Ex1032, 650.

## **B. Known Features of GPUs Prior to the '438 Patent**

43. The '438 patent describes several features and functions of GPGPU, some of which are recited in the claims. As explained below, these features and functions were all well known to those skilled in the art prior to the '438 patent.

### **1. Prior Art Parallel Computing and “Controllers”**

44. The principles of parallel processing were known prior to the '438 patent, and researchers and engineers in the field had applied these parallelization techniques to GPUs, which are particularly suited to perform parallel computations with their parallel processing engines. Researchers and engineers had also settled on effective designs for parallel processing systems, including a “controller” for managing the parallel computing process.

45. Parallel computing involves dividing a computational task among more than one processor. When appropriately configured, a system using many processors can perform challenging calculations more quickly than a single-processor system by instructing these many processors to work on different parts of the problem simultaneously.

46. Decades before the '438 patent, parallel computing was of particular interest to the scientific computing community, who expected that speed gains

through parallelization would make new classes of experiments and computations possible. *Id.*, 365–67. It was found, however, that parallel processes lose efficiency when individual processors sit idle or when they compete for memory resources.

47. Thus, parallel computing system designers proposed architectures to address such bottlenecks that may arise in such systems. A common solution involved using one set of computer hardware (*e.g.*, a controller) to coordinate the function and results of a second set of hardware that performs the core computations (*e.g.*, processors).

48. For example, one class of control architecture for parallel computation is Single-Instruction-Multiple-Data (“SIMD”), which has been known in the art since at least the 1970s. Ex1014, 1, 13. There are many examples in the prior art of SIMD architectures (including for neural networks) with a host computer, controller separate from the CPU, and parallel processors; indeed it was known that having the controller, rather than the CPU, manage the processors frees up the CPU to perform other functions. Ex1013, 375–89; Ex1028, 1–2; Ex1029, 845; Ex1030, 2.

49. Another class of control architecture for parallel processing is the Multiple Instruction Multiple Data (“MIMD”) architecture. Ex1014, 957–58. In such a system, operations are split into units whose outputs do not depend on each other, and which might require different calculations. *Id.*; Ex1032, 547–48 (“[I]n a MIMD-parallel architecture, different processors may simultaneously execute

different instructions.”). Like SIMD, prior art MIMD systems used a controller. *See* Ex1014, 957–58.

50. Beginning in the late 1990s, those skilled in the art recognized that the existing GPU pipeline could be adapted to known parallel processing paradigms, like SIMD and MIMD. Ex1032, 547–49. Texture memory, originally designed to store a matrix of pixels with float values at each location, closely resembled the vector structures commonly used in numerical analysis. By reimagining its GPU architecture in the context of established parallel computing models, NVIDIA made its GPUs highly suitable for these frameworks. By early 2005, NVIDIA GPUs supported features like “MIMD branching, SIMD branching, and condition codes,” all of which were well-understood architectures in the art of parallel computing. Ex1032, 547–49.

51. Companies like NVIDIA had also applied the well-known “controller” framework for directing parallel computations. For example, two of the references discussed in this Declaration—Nickolls and Kirk—disclose improved GPU architectures with a “controller” to direct the distribution and execution of instructions in the processing cores of the GPU.

## **2. The Stream Programming Model of GPUs**

52. Prior to the ’438 patent, it was known in the art that GPUs operated based on the “stream programming model,” where data is represented as a *stream*—

an ordered set of elements of the same data type. Ex1032, 464 (“In the stream programming model, all data is represented as a *stream*—which we define as an ordered set of data of the same data type. That data type can be simple (a stream of integers or floating-point numbers) or complex (a stream of points or triangles or transformation matrices).”). Streams are processed by *kernels*, which is a program or function that “tak[es] one or more streams as inputs and produc[es] one or more streams as outputs.” *Id.*

53. A key feature of this stream programming model is that kernels—the computational units in the pipeline—process entire streams of data rather than individual elements. *Id.* This design ensures that computations on individual stream elements are entirely independent of one another, enabling what seems like a serial calculation to be executed efficiently on data-parallel hardware. *Id.* Moreover, because the outputs of a kernel depend solely on its inputs, all the data required for execution is known upfront, allowing for highly optimized data storage, access, and management. This architecture aligned perfectly with the existing graphics pipeline in GPUs (including NVIDIA’s), which relied on data parallelism and element independence. *Id.*

54. In the stream programming model, applications are built by chaining multiple kernels together. *Id.*, 465. By the time of the ’438 patent, GPUs featured high-performance data-parallel processors capable of implementing two essential

kernels in the graphics pipeline: the vertex program and the fragment program. *Id.*, 467. These programs could sustain high computation rates on user-defined tasks with the precision necessary for solving general-purpose computing challenges. In effect, they transformed GPUs into programmable stream processors, making them suitable for far more than graphics rendering. *Id.* Their architecture excelled in tasks with “high arithmetic intensity,” such as solving systems of linear equations, making GPUs an attractive solution for a wide range of computationally demanding applications beyond their traditional graphics domain. *Id.*, 467, 495.

### **3. APIs and Programming GPUs**

55. Prior to the '438 patent, as adoption of GPGPU spread and demand grew, graphics hardware manufacturers such as NVIDIA developed improved software tools for programmable GPUs, such as advanced application programming interfaces (APIs) and programming languages. These tools facilitated interactions with GPU pipelines, including programmable shaders for GPGPU tasks. Graphics APIs act as a bridge between software applications and the graphics hardware by providing high-level, simple-to-use functions for developers while handling low-level communication with the hardware.

56. For example, programmers made use of NVIDIA’s GPUs through graphics Application Programming Interfaces (APIs), such as Direct3D and OpenGL, as well as GPU languages such as HLSL, GLSL, and Cg. Ex1032, xxxii; *see also*

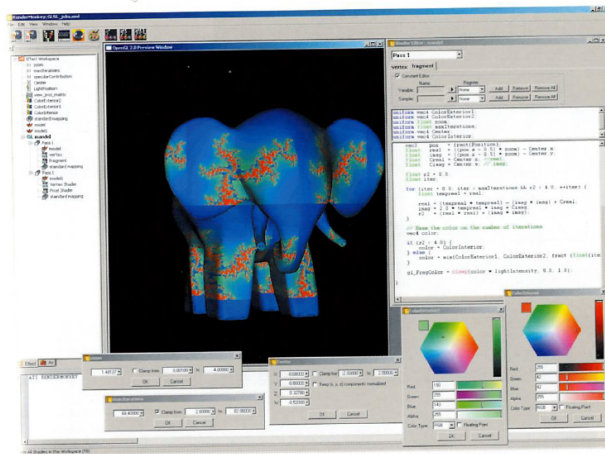
Ex1001, 1:62–64 (“A GPU on a conventional video card is usually controlled through OpenGL, DirectX, or similar graphic application programming interfaces (APIs).”). On release, the GeForce 256 GPU used Microsoft’s DirectX 7.0 API. Ex1015, 2. By 2004, NVIDIA’s GPUs supported OpenGL 2.1, then developed by Silicon Graphics, Inc. NVIDIA worked in tandem with API designers to enable programmers to make better use of their GPUs.

57. OpenGL, short for Open Graphics Library, is a cross-platform API developed in 1992 by Silicon Graphics to standardize the way developers program hardware to render graphics. It provided a hardware-agnostic interface, allowing developers to create complex graphical applications without worrying about the specifics of the underlying hardware. Initially designed for high-performance workstations used in industries like CAD, scientific visualization, and video production, OpenGL soon became a key tool in game development and real-time rendering. With the introduction of programmable shaders in GPUs during the late 1990s and early 2000s (discussed in further detail below), OpenGL evolved to support these features, enabling developers to write custom shader programs in GLSL (OpenGL Shading Language). Ex1017, 26–27.

58. Other high-level programming languages prior to the ’438 patent included HLSL (High-Level Shader Language) and Cg (C for Graphics). HLSL, developed by Microsoft, was introduced as part of DirectX to provide developers

with a tool for creating custom shaders on Windows platforms. Cg, developed by NVIDIA, aimed to simplify GPU programming by providing a C-like syntax, making it accessible to a broader range of developers. Ex1017, 2–3 (“Cg program can perform physical simulation, compositing, and other nonshading tasks.”).

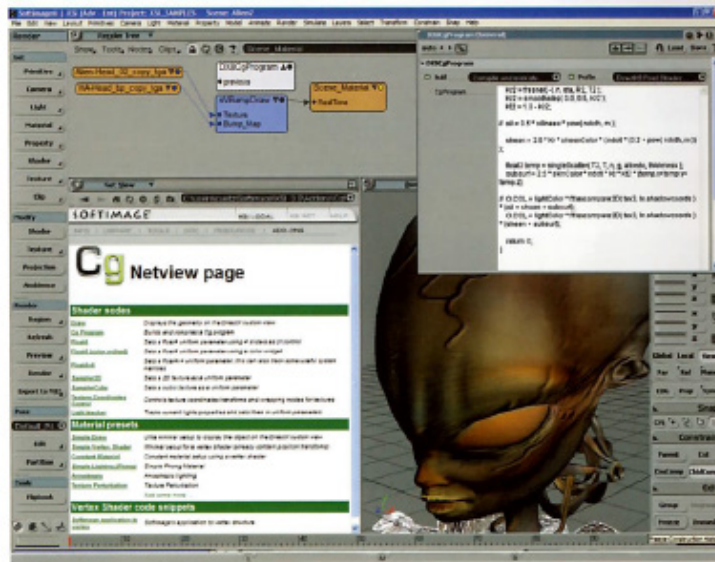
59. Visual tools were also created to interact with GPUs. For example, the book *OpenGL Shading Language* shows a user interface that allowed a user to interactively adjust parameters used by the GPU for shading an object.



Color Plate 1 Screen shot of the RenderMonkey IDE user interface. The shader that is procedurally generating a Julia set on the surface of the elephant is shown in the source code window. Color selection tools and user interface elements for manipulating user-defined uniform variables are also shown.

Ex1016, 281.

60. Similarly, the book *Cg: The Cg Tutorial* describes and shows various Cg viewers that can be used to facilitate user interaction.



Ex1017, 32.

#### 4. Known Memory Management Techniques

61. As GPUs became more powerful, with increasing clock speeds and chip sizes, the amount of time it took to transfer data (cost of communication) increased compared to the amount of time it took to perform computations (cost of computation). Ex1032, 459–60. Thus, memory management techniques were already an area of focus for developers. The memory management features described and claimed in the '438 patent, such as using pointers for memory addresses and “swapping” pointers to allow the output of one process to be the input of another, were well known in the art decades before the '438 patent.

##### a) Using Pointers for Memory Management

62. Computers store information in memory. Memory is not an infinite resource, so programmers are careful to avoid storing data that will not be needed

later. Further, memory is more limited the closer it is to the processor. A processor generally has a local memory cache with less space, and a larger memory (such as DRAM, which is main memory, SSD, which is Flash-based storage, or a hard drive) with more space. But transferring data to and from main memory or storage can be much slower than moving data in the local cache, and so programmers avoid unnecessary such transfers. Reading and writing data takes time, even in the processor cache, so programmers avoid unnecessary copying (which both reads and writes data).

63. The C programming language, first released in 1977, has long offered sophisticated tools to handle memory use and access. Data used during the operation of a program is called runtime data, and these are generally stored in memory cache. The C programming language includes high-level functions to allocate and deallocate memory in the cache for variables, and it uses automatic processes that deallocate local variables when a function closes. *See, e.g.*, Ex1009, 86–116. To pass an array (*i.e.*, data) to a function, C instructs programmers to create a pointer variable which stores the address of the array in the memory, and then pass that address to the function. *Id.*, 88–90. This approach allows the function to read data from the space in memory directly and to write data back to that same space directly. Without using pointers, a system would have to copy into a memory location accessed by the function and then copy the resulting data back at the end of the function, which

would require a lot of time and system resources. *Id.*, 89. These design features reduce the number of unnecessary read/write cycles and reduce the risk that the processor cache will run out of memory.

64. Similar memory management features quickly became common among computer programming languages, particularly those developed for numerical simulation and computation. *See, e.g.*, Ex1012, 17–20.

#### **b) Using Outputs as Inputs and Pointer Swapping**

65. It was well known that in many numerical simulations and computations, the output of one computational step is used as the input for the next step. For example, in a simulation of a physical system over time, the result (output) of one time step is input into the simulation as an input for the next time step, creating a feedback loop that allows the model to evolve accurately over time. Prior to the '438 patent, those skilled in the art had already developed effective solutions for implementing these kinds of computations in computers.

66. A simple approach is to copy the output data into a new array for use in the subsequent step, but this copying process incurs a performance cost due to the time and memory bandwidth required to duplicate potentially large datasets. A more efficient alternative—which was well known and used in the art prior to the '438 patent—is to avoid copying altogether by swapping the pointers of the input and output data arrays. By simply redirecting the pointer that indicates which array

to use as input, the simulation can proceed without duplicating data, significantly reducing overhead. This approach not only saves time but also minimizes memory usage, making it especially valuable in high-performance computing applications where memory and processing power are critical resources.

67. Programming techniques for swapping pointers existed in computer programming long before the '438 patent. For example, one textbook relating to the C programming language teaches a “swap” function, which swaps *pointers* to values, rather than swapping the values themselves, for efficiency.

The way to obtain the desired effect is for the calling program to pass *pointers* to the values to be changed:

```
swap(
```

Since the operator & produces the address of a variable, &a is a pointer to a. In swap itself, the parameters are declared as pointers, and the operands are accessed indirectly through them.

```
void swap(int *px, int *py) /* interchange *px and *py */
{
    int temp;

    temp = *px;
    *px = *py;
    *py = temp;
}
```

Ex1009, 88.

68. Prior to the '438 patent, GPUs had been designed in ways to easily allow the above principles to be applied. For example, GPUs (such as NVIDIA's) allowed shaders to render results to a buffer (memory) rather than a display. It was known to use these buffers to perform iterative computations, using the output of one rendering step as the input to the next, until a final frame was ready to render to the display (in the case of graphics) or final result was passed back to a CPU and/or

system memory (in the case of GPGPU). It was well understood that programmers could use these buffers to solve iterative numerical simulations with limited read-write cycles between CPU and GPU. Ex1032, 501–08. One known way to accomplish this was through “render-to-texture,” through which the results of a program (shader) are written to local texture memory “so they can be used as input to future programs.” *Id.*, 501.

69. The concept of “pointer swapping” was well known in the graphics and GPGPU space prior to the ’438 patent as a way to leverage the output of one computation as the input to the next. For example, in the book GPU Gems (which I understand is prior art to the ’438 patent), NVIDIA employee and author Mark Harris provides C++ code for simulating a reaction-diffusion system on a GPU that includes a “swap()” function at the end of each iteration that allows the output of one step of the simulation to become the input of the next step. The code defines one buffer (*i.e.*, portion of memory) as the input (`currentSource`) and another as the output (`currentTarget`). That is, the input data for a step is stored in a memory location pointed to by the variable `currentSource` and the output data is saved in a memory location pointed to by the variable `currentTarget`. At the end of computational cycle, the function `swap()` swaps the pointers for the `currentSource` buffer with that of the `currentTarget` buffer, so that the data

pointed to by `currentTarget` becomes the input pointed to by `currentSource` for the next 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);
}
```

Ex1032, 508 (annotated). GPU Gems contains several other examples of pointer swapping during numerical computations. *E.g., id.*, 44 (“Then, we swap the roles of the two textures and iterate the process . . .”), 713 (“swap rho and newrho pointers” at the conclusion of each step of a “Conjugate Gradient Solver”), 743 (“To sort the entire field, we just call the next list and swap the buffers until we’re done.”).

70. In this art, these techniques were sometimes known as “double buffering” or “ping-pong rendering.” For example, OpenGL’s “double buffering” used two framebuffers (which were dedicated areas of memory)—one for rendering and one for display—and swaps pointers between them at the end of each frame. In fact, by 2004, OpenGL 2.1 supported quadruple buffering. Ex1018, 1. DirectX 9 likewise supported double buffering at least by 2003. Ex1019, 1, 18, 23; *see also* Ex1020, 17 (describing the “ping pong technique,” in which the user “alternately

use[s] the output of a given rendering pass as an input in the next one”); Ex1021, 323–24.

### **c) Transferring Data Between System Memories**

71. Prior to the '438 patent, it was known that in numerical simulations and other computer applications, a user did not need to preserve all information from each time step. Rather, a user could elect when and how to store output data, including intermediate results of calculations.

72. A user running a numerical simulation would understand that runtime data is primarily stored in the processor's cache as long as it fits within the available cache size. If the cache capacity is exceeded, data is transferred to other levels of memory, such as DRAM, which offers larger storage but with increased access latency. Since transferring data between different memory hierarchies incurs computational overhead in terms of time and resource management, the user must balance the need for storing output data with the cost of moving it. For example, the user might only require a steady-state solution at the final step of the simulation, minimizing storage needs. Alternatively, he or she might need to track the evolution of a concentration profile over time, choosing an appropriate step size to achieve a desired accuracy. In some cases, a variable step-size method might be employed to maintain numerical stability, while still ensuring output is recorded at regular intervals. In each scenario, the user configures the simulation to store output

selectively, discarding or overwriting intermediate results as necessary. Ultimately, the user must strike an appropriate balance between data retention, computational efficiency, and storage constraints, based on the specific objectives of the simulation. *See generally* Ex1011, 710–11 (showing a selection of step size according to tolerance for error and demand for speed).

## **5. Artificial Neural Networks and Other Applications of GPGPU**

73. The parallel nature of GPU programming contributed directly to the rise in popularity of using GPUs for non-graphics applications, including artificial neural networks. By early 2005, GPUs had already been used for a variety of traditional scientific computing applications. This included, for example, solving systems of linear equations (Ex1032, 703–18); reaction-diffusion simulations (Ex1032, 505–08); protein folding and structure prediction (Ex1021, 8; Ex1032, 695–702); and fluid dynamics and flow simulation (Ex1022, 1; Ex1032, 747–63), among many others. *See generally* Ex1032 (“Part VI: Simulation and Numerical Algorithms”).

74. One significant application that arose from early GPGPU advancements was the ability to simulate artificial neural networks on GPUs. *See, e.g.*, Ex1023, 322–24; Ex1006, 622. An artificial neural network is a computer-based model designed to mimic the way the human brain processes information. In the human brain, neurons are organized into layers that work together to process information. Neurons communicate by sending signals to each other, forming

pathways that support thinking, learning, and memory. Information flows through layers of neurons, with each layer refining and interpreting the data. As pathways are used more frequently, they strengthen, allowing the brain to recognize patterns. The collection of layers of neurons is referred to as a “neural network.”

75. An “artificial neural network” mimics a natural neural network in a computer. The artificial neural network is made up of layers of nodes—each acting like a neuron—that receive data, perform calculations, and pass the results to the next layer of nodes. Every node is connected to nodes in the next layer with parameters called weights. By adjusting these weights, or connections, between these layers of nodes during training, the network gradually learns to recognize patterns and make decisions, much like how repeated experiences in the human brain strengthen neural pathways.

76. 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. This makes GPUs well-suited for the task, as their parallel processing capabilities can handle many calculations simultaneously, significantly speeding up the training process. Ex1032, 494.

77. In one simple illustrative example, the inputs to a neural network are represented as a vector or matrix of values. Each input is then multiplied by a

corresponding weight (representing the connection between layers of nodes). The sum of these weighted inputs is then calculated and that resulting vector is passed through an activation function that determines whether a node (neuron) should be activated (fire) based on the weighted sum of its inputs. Activation functions introduce non-linearity to the system, and enable the system to learn complex patterns. A bias term can be added to the weighted sum of inputs prior to passing through the activation function to shift the results to better fit the data. These computations are computed for all nodes in the layer, and the results are passed as inputs to the next layer.

78. An artificial neural network is usually trained by feeding a model large amounts of labelled data. For example, to identify handwritten numbers, the network would be provided with a large number of labelled images of numbers (positive examples) and other objects that are not numbers (negative examples). The computer simulating the neural network would then make predictions (*i.e.*, what number is represented in the image) by running the training data through its layers of nodes. The predictions are compared to the labels (the correct answers) to calculate an error. This error is propagated backward through the network—a process called backpropagation—which adjusts the internal weights to improve accuracy. Because this process requires performing millions of complex calculations (like matrix multiplication) across many layers and iterations, it is computationally intensive.

79. Prior to the '438 patent, it was known that the parallel processing power of GPUs was crucial for efficiently training these networks, as GPUs are particularly well-suited for the matrix operations and numerical calculations required for neural network training. Ex1023, 322–23 (“[M]any inner product operations can be replaced with matrix multiplication, which is more appropriate for GPU implementation.”). For example, before the '438 patent, GPUs were already being utilized to power artificial neural networks in a variety of applications, including tasks such as image and pattern recognition (Ex1023), object detection (Ex1006, 622), medical diagnostics (Ex1026), and checkers (Ex1025, 376). The rapid progress in this field owed much to the computational capabilities provided by early GPGPU innovations, including innovations from NVIDIA and others. *See generally* Ex1032.

## **6. Real-Time Data Input**

80. Prior to the '438 patent, the idea of using GPUs to process real-time streams of input, such as video data from a camera, was being explored in numerous research and development contexts. For example, GPU-based computer vision was well developed, with “[t]he single-instruction, multiple-data (SIMD) capability of the GPU mak[ing] it suitable for running computer vision tasks.” Ex1032, 649.

81. It was thus well understood at the time that the parallel processing capabilities of GPUs made them particularly suitable for real-time tasks. Researchers

and engineers had been experimenting with receiving real-time inputs, like live video streams from cameras, and sending them to GPUs for tasks such as image processing, object tracking, and motion detection. Ex1023; Ex1006. As one example, researchers had used artificial neural networks in a GPU to track a soccer ball in real time for robot soccer. Ex1006, 622. The concept of inputting data and writing it to memory in parallel with calculations being performed on previously input data was well-known and used in these applications for processing real-time data.

## **7. Task and Resource Management on a GPU**

82. The operating system is the central software that manages hardware components in a computer, including CPUs and GPUs, ensuring that they work together smoothly and efficiently. One key part of this management is handled by the “scheduler,” a mechanism built into the operating system that decides which tasks—whether they are individual threads or complete processes—get to use the computer’s resources at any given time. *E.g.*, Ex1027. The scheduler works by assigning priorities and time slices to tasks based on factors like urgency, resource demands, and overall system load, ensuring that critical operations run promptly while also balancing the needs of less urgent tasks.

83. Schedulers enable shared resource utilization by determining which tasks should run at any given moment, making necessary computational resources

available when required. For instance, once all the resources needed for a specific task are ready, the scheduler can prioritize and select that task for execution. Scheduling policies govern how these decisions are made and take into account factors like task priority, resource demands, and timing. These policies are designed to meet various objectives, such as minimizing latency or response time, maximizing throughput, or meeting deadlines in real-time systems.

84. Prior to the '438 patent, one of the most widely used scheduling strategies was priority scheduling, where each task was assigned a priority value. In the situation that a set of two or more tasks have the same priority value, the scheduling policy can shift to using round robin scheduling for that set of tasks, assigning time slices to each task in equal portions and in circular order so that each task has an equal chance of getting resources. *Id.* The scheduler would then use these priority levels to allocate resources, giving preference to higher-priority tasks. In systems that supported what is known as “preemption,” higher-priority tasks could interrupt lower-priority ones, temporarily suspending their execution. When a task is preempted, the resources it was using are freed up for the higher-priority task. This dynamic allocation ensures that critical tasks are executed promptly while maintaining system responsiveness.

85. The concept of queuing tasks for executing was fundamental to task scheduling. When a resource required for a particular task is currently in use, the

scheduler will queue the task for later execution. Schedulers queue tasks as they come in by assigning each task a position in the queue based on a scheduling algorithm. For example, if a GPU is currently executing a computation, the scheduler can queue the task by placing it in a specific position in the queue based on factors such as task priority and resource availability. The queuing of tasks allowed operating systems to handle multitasking effectively, balancing the load and ensuring that high-priority tasks received timely attention. By managing tasks that require access to the same data, computational resources currently in-use, or have data dependencies, schedulers optimize efficiency, balance computational loads, and improve data coherency in a system. Ex1027.

86. In sum, the concepts discussed in the '438 patent are not new. Prior to the '438 patent, the industry had already advanced and transformed GPUs into a general-purpose computing platform using well-known techniques such as SIMD parallelization, memory management (*e.g.*, pointer swapping to use outputs as inputs), and scheduling. These techniques, individually and collectively, allowed GPUs to perform computationally intensive tasks, like large scale numerical simulations (such as reaction-diffusion systems) or artificial neural networks. NVIDIA and others led this effort, not Patent Owner. And NVIDIA published its methods (both in books and in patents) before Patent Owner filed its patent.

## V. THE '438 PATENT

### A. Summary of the '438 Patent

87. The '438 patent, titled “Graphic Processor Based Accelerator System and Method,” was filed on November 9, 2017, and issued on February 16, 2021. U.S. Patent No. RE49,438 (Ex1001), Cover.

88. The '438 patent belongs to a family of patents directed to an optimized system for performing general-purpose computations on a graphics processing unit (GPU)-accelerated system. The '438 patent family relates to an “accelerator” for performing computations in a computer system, such as numerical simulations. Ex1001, 1:59–2:10. In the Abstract, the '438 patent is summarized as follows:

89. An accelerator system is implemented on an expansion card comprising a printed circuit board having (a) one or more graphics processing units (GPUs), (b) two or more associated memory banks (logically or physically partitioned), (c) a specialized controller, and (d) a local bus providing signal coupling compatible with the PCI industry standards. The controller handles most of the primitive operations to set up and control GPU computation. Thus, the computer's central processing unit (CPU) can be dedicated to other tasks. In this case a few controls (simulation start and stop signals from the CPU and the simulation completion signal back to CPU), GPU programs and input/output data are exchanged between CPU and the expansion card. Moreover, since on every time step of the simulation the results from the

previous time step are used but not changed, the results are preferably transferred back to CPU in parallel with the computation. *Id.*, Abstract.

90. In the Background of the Invention, the '438 patent explains that “Graphics Processing Units (GPUs) are found in video adapters (video cards) of most personal computers (PCs), video game consoles, workstations, etc. and are considered highly parallel processors dedicated to fast computation of graphical content.” *Id.*, 1:30–34. The patent further acknowledges that “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.” *Id.*, 1:34–42. These applications are referred to as “general purpose GPU” or “GPGPU.” *Id.*, 1:42–45. As an example, the '438 patent says that “[n]umerical simulations, e.g., finite element analysis, of large systems of similar elements (e.g. neural networks, genetic algorithms, particle systems, mechanical systems) . . . can benefit from GPGPU computation.” *Id.*, 1:51–55.

91. The '438 patent specification does not claim to have invented the performance of general-purpose computing on GPUs. *Id.*, 1:38–47. In fact, the '438 patent states that GPGPU applications have “increased [in] popularity” and notes that 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.*, 1:38–47.

92. To overcome this drawback, the ’438 patent purports to invent a “hardware implementation” having “(a) one or more graphics processing units, (b) two or more associated memory banks that are logically or physically partitioned, (c) a specialized controller, and (d) a local bus providing signal coupling.” *Id.*, 2:24–32.

93. The ’438 patent does not purport to have invented any specialized hardware. *Id.*, 14:38–42. Instead, the ’438 patent describes a purportedly novel architecture using “currently available and reliable” hardware components. *Id.*, 2:42–45. This hardware implementation of “the computational stream of execution” replaces “thread and context initialization” with “hardware initialization.” *Id.* According to the ’438 patent, this arrangement allegedly improves overall system 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.*, 2:41–43. As a result, “the CPU is freed from this function and is dedicated to other tasks.” *Id.*, 2:43–44.

94. Figure 2 (below) is illustrative of the allegedly inventive “GPU accelerator system.” *Id.*, 5:1–8. On the left-hand side of the system are “one or more CPU’s 120,” “main, or system memory 130,” and “mass/non volatile data storage

140.” *Id.*, 4:9–25. Those components connect to “expansion card 180” over one or more buses. *Id.*, 4:9–41. Expansion card 180 includes one GPU 240, memories 210 (shader memory bank) and 250 (texture memory bank), and “accelerator controller 220.” *Id.*, 5:9–57.

95. In the embodiment of Figure 2, “one memory bank, the shader memory bank 210, is used to store the shaders needed for the execution of the required computations and the parameters that are common for all computational elements,” and “[t]he second memory, the texture memory bank 250, is used to store all the necessary data that are specific for every computational element (including, but not limited to, input data, output data, intermediate results, and parameters).” *Id.*, 5:35–57. The ’438 patent explains that “the term ‘texture’ in this document refers to a data array unless specific otherwise and each element of the texture is a pixel of color information” and “the term ‘shader’ in this document refers to a GPU program unless specified otherwise.” *Id.*, 5:35–57.

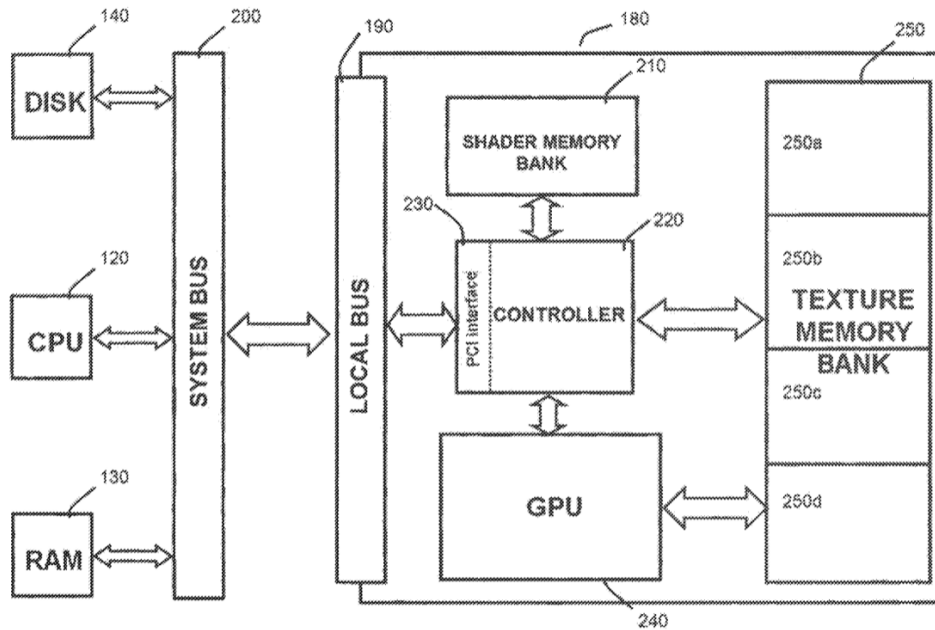


FIG. 2

*Id.*, Fig. 2.

96. Accelerator controller 220 is described as “specifically designed” and can be implemented as “field programmable gate array (FPGA) logic, or custom built application-specific (ASIC),” or “even fully implemented in software.” *Id.*, 5:9–17. The controller “commands the storage and retrieval of arrays of data,” execution of GPU programs,” and “data transfer between the system bus 200 and the expansion card 180.” *Id.*, 5:18–29.

97. The “system comprises three ‘peripheral’ components: user interactive hardware, disk hardware, and computational hardware.” *Id.*, 2:3–11. The CPU establishes communication and synchronization between the peripheral components, and “[e]ach of the peripherals is preferably controlled by a dedicated thread that is

executed in parallel with minimal interactions and dependencies on the other threads.” *Id.*

98. The ’438 patent describes various applications and techniques for its accelerator architecture. For example, the ’438 patent says that “neural networks” are “one example of an application that can benefit from GPGPU computation.” Ex1001, 1:51–55. In that example application, each layer of the neural network outputs a result (representing the outputs of the neurons in that layer) that is used as an input for the next layer. *Id.*, 6:13–19. The ’438 patent also says that the input for its calculations can be received “in real time from a recording device.” *Id.*, 8:29–35, 9:36–39.

99. 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.

100. 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.

101. In a numerical simulation like a neural network, “[e]ach computational element discussed above has output variables that affect the rest of the system.” *Id.*, 6:13–22. In some implementations, these “new values should be copied over old values so that they are used as input during the next time step,” but “[c]opying textures” is a computationally expensive operation. *Id.*, 11:59–12:5. The '438 patent explains that its system allows the value of the output variable that “corresponds to the value computed on the previous, not the current time step,” to be accessed by “other elements of the system” by “dedicating two textures to output variables.” *Id.*, 6:23–38. One texture “holds the value computed during the previous time step and is accessible to all computational elements during the current time step” while the other “is not accessible to other elements and is used to accumulate new values for the variable computed during the current time step.” *Id.* Because the “textures are referred to by texture IDs (pointers),” the system “swap[s] the address pointers” to the respective textures between computational steps “so that newly accumulated values serve as accessible input during the next time step, while the old input is replaced with new values of the variable.” *Id.*, 6:23–38, 11:59–12:4. Swapping

pointers instead of copying textures “achieves the same result at a much lesser cost.”  
*Id.*, 11:59–12:4.

102. A “[c]ontroller-driven data exchange or input parser texture generator 316 allows the user to change input that is generated on the fly during the simulation,” allowing for monitoring of “real time” input “that is coming from a recording device,” such as a video camera. *Id.*, 9:34–44. The controller-driven data exchange “preprocesses the input...and generates textures.” *Id.* The textures created by the controller-driven data exchange “are transferred to hardware not whenever ready but upon the request of the controller.” *Id.*

103. The ’438 patent focuses on this idea of transferring input data acquired in real time from external systems to the GPU memory. *Id.* The claims recite transferring input data acquired in real time from external systems to GPU memory “after the GPU starts the first computation and before the GPU starts a second computation of the sequence of computations,” and, more specifically “*during* performance of the intermediate computations in the sequence of computations.” *Id.*, cls. 1, 21.

104. In other words, the claims focus on the idea of performing necessary data transfer “in parallel with the computation, thus reducing the impact of this transfer on the performance” of the system. *Id.*, 14:14–20.

105. As I explain in detail below, all of these features were well-known and widely used in the field of GPGPU as of the priority date of the '438 patent.

**B. Summary of the Prosecution History of the '438 Patent**

106. The '438 patent is a reissue of U.S. Patent No. 9,189,828 (“the '828 patent”). The '438 patent originated from reissue Application No. 15/808,201 (“the '201 application”) filed on November 9, 2017 and claims priority from Provisional Application No. 60/826,892 filed on September 25, 2006—more than 10 years prior to the filing of the '201 application.

107. The '828 patent issued with 20 claims. Ex1002, (specification of the '828). The Applicant included a preliminary amendment with the reissue application amending the issued claims of the '828 patent and adding more than one hundred new claims. Ex1002, (11/9/17 preliminary amendment). The Applicant also included reissue declarations, broadly stating that “Applicant erred by claiming less than it had a right to claim” in the '828 patent. *Id.*, (11/9/17 reissue declaration).

108. As presented in the preliminary amendment, claim 42, which is generally representative, recited:

A method of performing a sequence of computations representing an artificial neural network, the method comprising:

performing, by a graphics processing unit (GPU), a first computation in the sequence of computations on first input data to generate first output data, an output of a computation in the sequence of computations representing an output of a neuron in the artificial neural network;

storing, in a memory operably coupled to the GPU, the first input data and the first output data; and

transferring second input data into the memory during performance of the sequence of computations by the GPU.

*Id.*, (11/9/17 preliminary amendment).

109. In the first Office Action, the Examiner objected to the reissue declarations and rejected the claims under § 112, § 251, § 102, and § 103. *Id.*, (11/20/18 NFOA). The Examiner cited U.S. Patent No. 7,219,085 to Buck (“Buck ’085”); U.S. Patent Application Publication No. 2001/0010034 to Burton; U.S. Patent No. 7,525,547 to Diard; U.S. Patent No. 7,119,810 to Sumanaweera; U.S. Patent Application Publication No. 2006/0129506 to Edelman; and U.S. Patent No. 5,142,665 to Bigus.

110. In particular, the Examiner rejected five of the six independent claims and many of their respective dependent claims as anticipated under § 102 in view of Buck ’085. *Id.*, (11/20/18 NFOA). The Examiner also rejected various claims under § 103 in view of Buck ’085 and Diard, noting that Diard “demonstrat[ed] that it was known at the time of the invention to transfer between an accelerator memory and the main memory while the GPU is performing other operations.” *Id.*

111. In response, the Applicant argued that Buck ’085 failed to teach the claim limitations requiring transfer of “at least a portion of the input data into the accelerator memory...during performance of the sequence of computations by the at least one graphics processing unit.” *Id.*, (3/20/19 response). Instead, Buck “only

discloses transferring data into GPU memory *before* the GPU performs any computations” and that “transfer and computations occur in an alternating fashion.” *Id.*, (3/20/19 response). The Applicant argued that Buck ’085 is “silent on transferring data to a GPU or GPU memory while the GPU is performing a sequence of computations,” because all of the CPU to GPU memory transfers that Buck discloses “occur before the GPU begins executing the shader programs.” *Id.* The Applicant also argued that Buck ’085 does not disclose “acquiring input data in real time with a video camera.” *Id.*

112. With respect to recapture, Applicant argued that the newly added reissue claims are “materially narrower” than the claims of the ’828 patent because “an artificial neural network is a particular type of the sequence of computations claimed broadly in claim 1.” *Id.*, (3/20/19 response).

113. Applicant amended the claims to recite, among other limitations, “initializing, by a controller operably coupled to a graphics processing unit (GPU), textures and shaders in a memory operably coupled to the GPU; transferring the first input data received by the CPU to the memory operably coupled to the GPU; [and] 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.” *Id.*, (9/13/19 response).

114. The Examiner then rejected many of the independent claims over Buck '085 in view of Diard and/or Johnson, noting that Johnson demonstrates that “it was known at the time of invention to provide a controller ‘operably coupled to the at least one graphics processing unit and the accelerator memory, to transfer the at least a portion of the output data from the accelerator memory to the main memory,’” and that Diard demonstrates that “it was known at the time of invention to transfer between an accelerator memory and the main memory while the GPU is performing other operations.” *Id.*, (12/16/19 NFOA).

115. Again, the Applicant amended the claims, specifying that the “first input data [is] acquired from an external system *in real time*,” that “an output of the first computation in the sequence of computations represent[s] an output of a first neuron in a *first layer* in the artificial neural network,” and that “transferring second input data acquired from the external system in real time into the memory” occurs “*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.” *Id.*, (4/16/20 response) (emphasis added).

116. Applicant also argued that “neither Johnson nor Diard says anything about transferring output data from a GPU to a CPU.” *Id.*, (4/16/20 response).

Applicant argues that Johnson discloses “transferring output data from the graphics processor 400 to either a display device or another graphics processor,” but not to the main memory. *Id.*, (4/16/20 response). Applicant additionally argued that Diard discloses “data transfer from the CPU to the GPU,” but not “transferring output data from the GPU to the CPU.” *Id.*, (4/16/20 response).

117. In a Final Office Action dated June 5, 2020, the Examiner identified allowable subject matter and, in response, the Applicant accepted the allowable subject matter and cancelled all claims that were rejected. Ex1002, (6/5/20 FOA). Finally, on September 30, 2020, the Examiner issued a Notice of Allowance for the remaining claims. Ex1002, (9/30/20 NOA). In the reasons for allowance, the Examiner stated that the “cited prior art of record” did not “reasonably disclose or suggest” a number of features across the six independent claims. *Id.*

118. The representative limitations included:

“a controller...to control performance of the sequence of computations by the at least one graphics processing unit, 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 the computations by the at least one graphics processing unit, and to transfer the 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” (claim 1);

“performing, by the graphics processing unit (GPU), a first computation in the sequence of computations on the first input data...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” (claim 21);*

*“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” (claim 21);*

*“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” (claim 12);*

storing the first output data in a second partition of the memory and *“transfer[ring] the first output data to another memory during computation of the second layer of neurons” (claim 112, which issued as claim 40); and*

*“a camera to generate input data in real time” (claim 95, which issued as claim 35).*

*Id.*

119. The ’438 patent issued on February 16, 2021, with 56 claims. Ex1001, Cover, 14:51–21:9.

### **C. Overview of the Challenged Claims**

120. There are six independent claims in the ’438 patent: claims 1, 12, 21, 35, 40, and 44. Claims 12 and 21 are method claims and claims 1, 35, 40, and 44 are system claims.

121. I understand that Petitioners are challenging claims 1–14, 16–34, and 40–54 of the ’438 patent (the “Challenged Claims”). I have analyzed each of these claims.

|        |  |
|--------|--|
| 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.  |
| 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.   |
| 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:<br>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;   |
| 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:<br>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:<br>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:<br>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.   |
| 18        | The method of claim 17, wherein (C) further comprises:<br>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:<br>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   |
| 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:<br>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:<br>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:<br>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:<br>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. |
| 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:<br>acquiring the second input data with at least one of a video camera, a microphone, or a cell recording electrode.  |
| 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   |
| 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 |
| 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.  |
| 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:<br>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. |
|----|---|

122. For the reasons I detail below, it is my opinion that each of these claims is unpatentable over the prior art discussed below.

## **VI. PRIOR ART CONSIDERED**

123. In this Declaration, I primarily discuss three prior art references with exemplary citations identifying the relevant features related to the claim language of the Challenged Claims. I describe combinations of these prior art references that, when combined, render the Challenged Claims obvious. I also discuss other documents and references to help establish what was known to a POSITA prior to the '438 patent. It is my opinion that the Challenged Claims are rendered obvious in light of the prior art specifically discussed in this Declaration, in view of the knowledge of a POSITA at the time of the '438 patent.

124. I understand that Patent Owner Neural AI, LLC (“Patent Owner”) asserts that it conceived of the purported inventions claimed in the '438 patent at least as early as February 25, 2005. I have been asked to assume that the references discussed in this Declaration are prior art to the '438 patent. In this Declaration I offer no opinions regarding whether Patent Owner is in fact entitled to a conception date prior to the filing date of the '438 patent.

## **A. General Overview of the Primary Prior Art References**

125. The following sections provide a brief description of the anticipating prior art systems and software that invalidate the '438 patent. This discussion is not meant to be exhaustive, and my full element-by-element analysis is provided in more detail below in the analysis section. I cite representative portions of the prior art references to make my point below, though in many instances there are other portions of the prior art references that reinforce the same points. I have personally reviewed the references cited in this Declaration.

### **1. Kirk**

126. U.S. Patent No. 7,139,003 (“Kirk”) is titled “Methods of Processing Graphics Data Including Reading and Writing Buffers” and issued on November 21, 2006, from an application filed on December 15, 2003. Kirk is assigned to Petitioner NVIDIA. It was well known to manage data transfer between a host system and an accelerator, such as a GPU, while the GPU executes the layers of the neural network. This feature is present in Kirk, which simultaneously transfers input and output data between the host system and GPU in order to achieve its text recognition.

127. Kirk discloses a “Computing System” with a “programmable graphics processor” for “processing of data.” Ex1005, 1:40–47, 26:24–39, 26:56–61 (“Programmable Graphics Processing Pipeline 1250 is programmed to operate on surface, primitive, vertex, fragment, pixel, sample, or any other data.”), Figs. 12A,

12B. In this Declaration, I focus on Figures 12A, 12B, and 12C, which depict an “alternate embodiment.” *Id.*, 26:24–26. Many of the components in this embodiment (and associated figures) have the same reference numerals as in other embodiments in Kirk, and thus disclosures provided in the context of those other embodiments also apply to the embodiment depicted in Figures 12A, 12B and 12C. I therefore rely on disclosures provided in the context of those other embodiments as well.

128. Kirk explains the “graphics data” it can process “is any data that is input to or output from computation units within [the] Programmable Graphics Processor 105.” *Id.*, 3:35–41. Thus, in my opinion, a POSITA would have understood that, in addition to standard graphics, Kirk’s programmable graphics processor is able to process numerical data for numerical simulations.

129. The computer system of Kirk is illustrated in the annotated version of Figure 12A below. Kirk’s computer system includes Host Processor 114 (red), Host Memory 112 (blue), and Graphics Subsystem 107 (purple), which communicate with each other through System Interface 115 and one or more buses. *Id.*, 3:17–33.



Fig. 12A

*Id.*, Fig. 12A (annotated).

130. Graphics Subsystem 107 includes Local Memory 140 (blue) and Programmable Graphics Processor 105 (green). *Id.*, 3:32–46, 26:24–39. Although not depicted in Figure 12A, Programmable Graphics Processing Pipeline 1250 contains additional memory for the execution pipelines (*e.g.*, in Texture Unit 154 and the input/output buffers). *Id.*, 6:22–38, 27:19–28. Programmable Graphics Processor 105 interfaces with the host system through Graphics Interface 117 (orange) and with Local Memory 140 (blue) through Memory Controller 120 (brown). *Id.*, 3:32–67, 26:24–39. Local Memory 140 stores input data, output data, and program instructions for graphics/data processing. *Id.*, 3:34–46. Programmable Graphics Processor 105 includes Programmable Graphics Processing Pipeline 1250

(red) in which data is processed according to instructions stored in memory. *Id.*, 26:32–61.

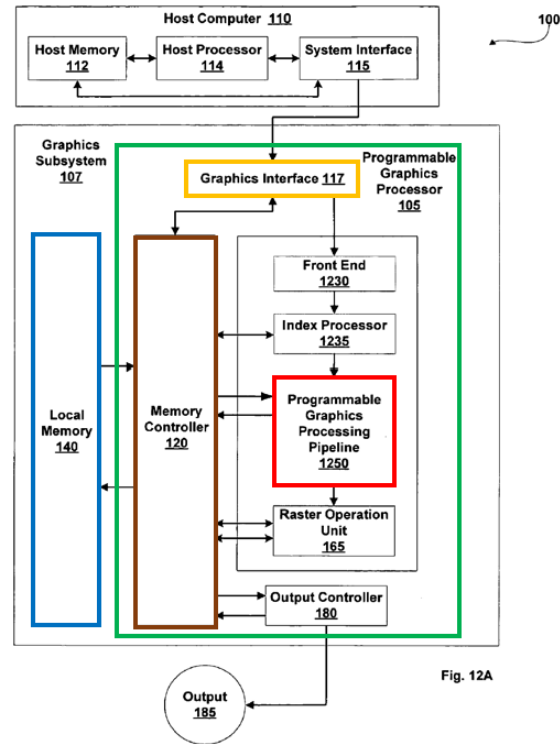


Fig. 12A

*Id.*, Fig. 12A (annotated).

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

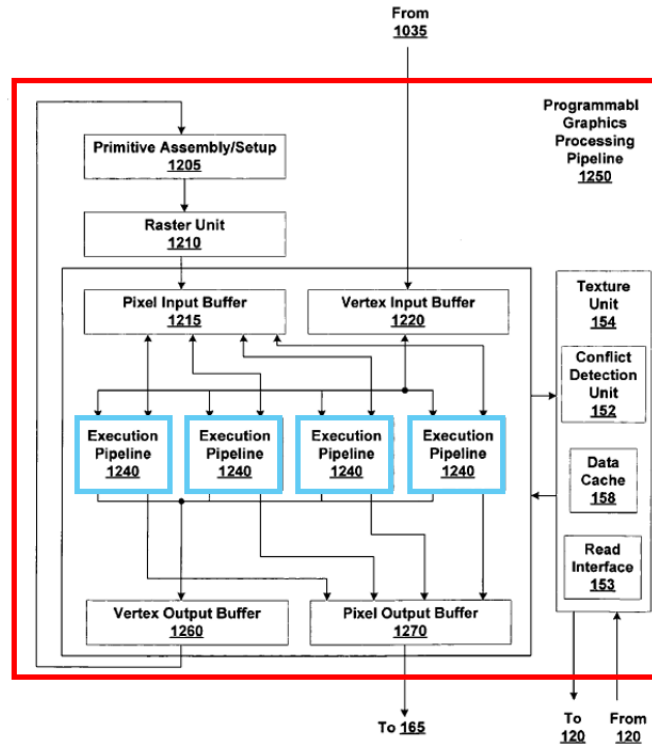


Fig. 12B

*Id.*, Fig. 12B (annotated).

132. Kirk’s Host Computer 110 communicates with Graphics Subsystem 107 through System Interface 115 and Graphics Interface 117 to provide data and instructions. *Id.*, 3:32–46. The Graphics Processing Pipeline 103 “receives a stream of program instructions (vertex program instructions and fragment program instructions) and data” through Graphics Interface 117 or Memory Controller 120, and performs vector floating-point operations or other processing operations using the data.” *Id.*, 4:21–42. In operation, “[a] graphics program . . . is executed within one or more Execution Pipelines 1240 as a plurality of threads where each vertex or fragment to be processed by the program is assigned to a thread.” *Id.*, 27:40–43. Results from the computations are sent to memory, such as Local Memory 140, Host

Memory 112, or memory within the Programmable Graphics Processing Pipeline. *Id.*, 3:35–46. Kirk teaches that data produced by the processing pipeline “may be written to a buffer in graphics memory to be read from during a subsequent pass.” *Id.*, 4:16–20.

133. In my opinion, Kirk is enabled, in that a POSITA exercising reasonable diligence could make and use the invention described in Kirk without undue experimentation. For example, Kirk provides more than 27 columns of disclosure and several figures that a POSITA could have relied on to develop an operational system.

## **2. Oh**

134. Oh was published in 2004 in The Journal of the Pattern Recognition Society. Oh also discloses simultaneous transfer of input and output data between the host system and GPU in order to achieve text recognition. As supported by Dr. Mary Bolin, Oh (which bears a copyright date of 2004 by reputable publisher Elsevier Ltd.) was available around April 2004. Ex1023, ¶¶24–27. A skilled researcher searching for the subject matter of Oh using relevant terms such as “GPU” and “neural network” would have located Oh. *Id.*, ¶¶70–74. Several other references in the field cited Oh before August 2005, confirming it was locatable by interested persons in the art. *Id.*, ¶¶20–30.

135. Oh is a research article that presents an implementation of “a faster NN [neural network] using common graphics hardware GPU.” Ex1007, 1. Oh explains that prior neural networks lacked the ability for “real-time processing” and “focuses on using a GPU to implement a multilayer perceptron” artificial neural network for text detection in video. *Id.*, 1, 3. Oh teaches that a GPU’s “extended capabilities in supporting complex operations have also become useful in non-graphics applications,” including for neural networks. *Id.*, 2. As Oh recognizes, there is significant potential for using GPUs instead of CPUs in applications in which “repeated operations are common,” due to the better “parallelism and more pipelined” structure of GPUs. *Id.*

136. Oh describes the process for performing matrix multiplication and applying a sigmoid function at each layer of a multi-layer neural network. *Id.*, 2–3. The implementation in Oh used a three-layer artificial neural network. *Id.* For a multi-layer neural network, such as the one in Oh, “[t]he result of the previous layer is saved in the form of a render target texture, which is then used as an input for the next layer.” *Id.*, 3.

137. The researchers in Oh applied these principles to use a neural network to “classify the pixels of input images, whereby the feature extraction and pattern recognition stage are integrated in the neural network.” *Id.* Oh’s system, which was run on readily available, commodity GPU hardware, received “image and video

data” as input and analyzed the frames to identify where in each frame text was located. *Id.*, 3–4. The researchers demonstrated that the processing time for Oh’s neural network on a GPU was greatly reduced as compared to the processing time on a CPU. *Id.*, 3–4.

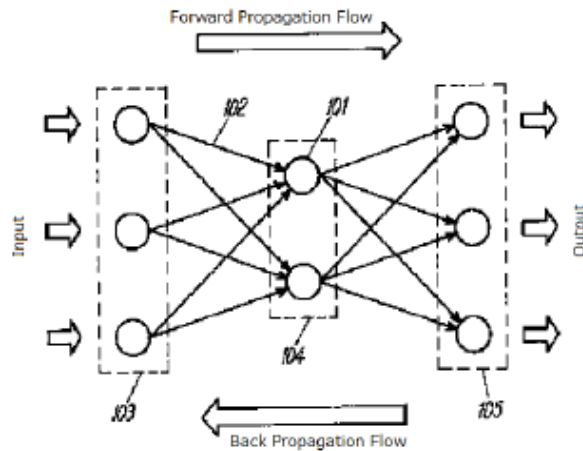
138. In my opinion, Oh is enabled, in that a POSITA exercising reasonable diligence could make and use the invention described in Oh without undue experimentation. For example, Oh provides pages of disclosure, implementation details such as hardware specifications, and several figures that a POSITA could have relied on to develop an operational system.

### **3. Tamura**

139. Japanese Patent Appl. No. H04-237388, titled “Neuroprocessor,” was filed on January 22, 1991, and was published on August 25, 1992. Ex1008, 6. Tamura was not of record during prosecution of the ’438 patent.

140. Tamura is directed to a “neuroprocessor” for “forward propagation and learning processing” in a high speed neural network. Ex1008, ¶1. Tamura explains that the use of “neural networks” in computing aims “to create a new type of computer that excels at tasks such as recognition, association, optimization, and speech synthesis.” *Id.*, ¶2. Figure 4 (below) shows an example of a neural network with first, intermediate, and output layers. *Id.*, ¶4.

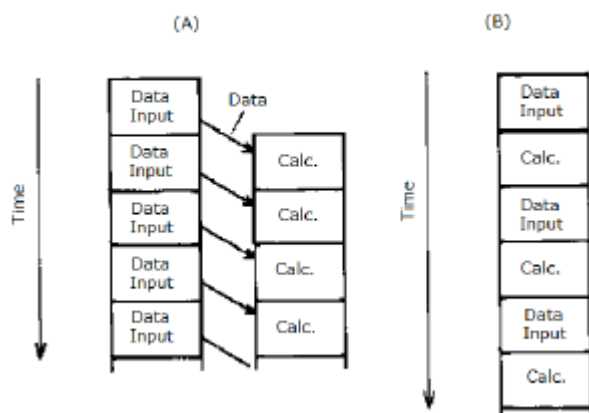
[FIG. 4]



*Id.*, Fig. 4 (with neurons 101 and lines 102 connecting the neurons).

141. Tamura recognized that “neural network calculations require a lot of processing time because they involve a very large amount of computation,” and the neuroprocessors of the prior art were inefficient because they would sequentially input data, perform calculations, input further data, and so forth. *Id.*, ¶¶26–27. To address this alleged shortcoming, 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; *id.*, ¶ 35 (“FIG. 3 (A) is a diagram showing processing flow. Data input and calculations are performed using data inputted in the previous step.”). 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.

[FIG. 3]



*Id.*, Fig. 3.

142. In my opinion, Tamura is enabled, in that a POSITA exercising reasonable diligence could make and use the invention described in Tamura without undue experimentation. For example, Tamura walks through several figures and example embodiments that a POSITA could have relied on to develop an operational system.

#### 4. GPU Gems

143. GPU Gems includes 48 chapters describing the state of the art in GPU programming. Ex1008, xxxi–xxxii. Part IV is a “Primer” on “General-Purpose Computation on GPUs,” which “aims to provide a gentle introduction to the world of general-purpose computation on graphics processing units, or ‘GPGPU.’” *Id.*, xx, 453. As supported by Dr. Mary Bolin, GPU Gems (which bears a copyright date of 2005 by reputable publisher Addison-Wesley) was available by July 2005. Ex1023, ¶¶35–38. A skilled researcher searching for the subject matter of ANN using relevant

terms such as “computer graphics” and “real-time programming” would have located GPU Gems. *Id.*, ¶¶31–41.

144. GPU Gems describes the concept of swapping pointers of input and output data, as claimed in claim 40 of the ’438 patent. Chapter 31, written by Mark Harris of NVIDIA, explains that “most computations are broken into steps,” where “[e]ach step depends on the output of previous steps.” Ex1008, 501. To implement “direct feedback of GPU output to input,” Mr. Harris explains that one can use GPUs’ “render-to-texture” feature. *Id.*, 499, 501. With this method, 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.*, 501, 504. The buffer pointer can then be passed as the input to a later computation. *Id.* Mr. Harris applies these concepts, including passing outputs to inputs, to “a simulation of a phenomenon known as chemical reaction-diffusion,” and provides code to implement the simulation. *Id.*, 505–08. The simulation 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.

145. 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).

146. Other chapters of GPU Gems also include examples of swapping pointers between outputs and inputs. *E.g.*, *id.*, 44, 713 (“swap rho and newrho

pointers”), 743 (“To sort the entire field, we just call the next list and swap the buffers until we’re done.”).

147. In my opinion, GPU Gems is enabled, in that a POSITA exercising reasonable diligence could make and use the invention described in GPU Gems without undue experimentation. For example, GPU Gems provides 48 chapters and over 800 pages of disclosure and figures that a POSITA could have relied on to develop an operational system.

## **VII. ANALYSIS OF THE CHALLENGED CLAIMS AGAINST THE PRIOR ART**

### **A. Claim Construction**

148. Unless specifically noted otherwise below, I have accorded claim terms their ordinary and customary meaning as understood by a POSITA and in light of the patent specification and prosecution history.

149. I have been informed by counsel and understand that claim terms need not be construed when there is no dispute as to the meaning of a term that could affect the outcome of the proceeding. It is my opinion that none of the claims of the ’438 patent require construction to determine their validity in view of the prior art I have analyzed in this Declaration. Thus, for purposes of this proceeding, it is my opinion that no formal claim construction is necessary.

**B. Application of the Prior Art to the Claims**

150. Based upon my review of the prior art, particularly the references detailed in this Declaration, it is my opinion that the subject matter of each of the Challenged Claims would have been obvious to one of ordinary skills in the art in view of the known prior art references.

151. Based upon my analysis, it is my opinion that the Challenged Claims of the '438 patent are rendered obvious by the following prior art grounds. These obviousness grounds also include the knowledge of a POSITA:

| <b>Ground</b> | <b>Claims</b>      | <b>Basis for Rejection</b>                                |
|---------------|--------------------|---|
| 1             | 1–14, 16–34, 40–54 | Obviousness over Kirk in view of Oh                       |
| 2             | 1–14, 16–34, 40–54 | Obviousness over Kirk in view of Oh and Tamura            |
| 3             | 40–43              | Obviousness over Kirk in view of Oh and GPU Gems          |
| 4             | 40–43              | Obviousness over Kirk in view of Oh, Tamura, and GPU Gems |

152. 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 “Oh” article describes using a multi-layer artificial neural network on a commodity GPU to identify text in video in real time.

153. 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 are occurring. The combination of Kirk and Oh teaches transferring input and output data between the host system and GPU while the GPU performs computations for an artificial neural network, making the computations on real-time data in Oh more efficient. Ex1005; Ex1007. Similarly, Tamura discloses inputting data and writing it to memory while calculations are being performed on previously inputted data. Ex1008. 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. Ex1032.

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

154. In my opinion, claims 1–14, 16–34, and 40–54 of the ’438 patent would have been obvious over Kirk in view of Oh and additionally in view of Tamura and

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

### **1. Combination of Kirk and Oh**

155. In my opinion, a POSITA would have been motivated to combine Kirk with Oh.

156. Before the '438 patent, it was well known that artificial neural networks could be effectively and efficiently implemented on GPUs. Section IV.B.5. One such example is Oh's artificial neural network for recognizing text in video, which was implemented on an off-the-shelf GPU. In my opinion, a POSITA would have been motivated to implement Oh's artificial neural network on NVIDIA GPU hardware, such as the GPU-accelerated parallel processing system in Kirk.

157. More specifically, it is my opinion that a POSITA seeking to execute the multi-layer artificial neural network for real-time pattern recognition disclosed in Oh would have sought out advanced systems with a programmable GPU, such as that disclosed in Kirk (assigned to GPU-industry leader NVIDIA). Thus, in my opinion, a POSITA would have been motivated to implement Oh's artificial neural network on the GPU-accelerated parallel processing system in Kirk. *Id.*

158. As I have previously explained, Oh discloses a multi-layer neural network that uses “video data” (including real time video data) to detect text in that data. Ex1007, 3. Oh emphasizes that its artificial neural network can be implemented using a “common graphics hardware GPU,” which achieved a “20-fold performance enhancement . . . compared to CPU-only processing.” *Id.*, 1, 4. In my opinion, Oh is indifferent as to the specific GPU architecture used for implementing its artificial neural network—although Oh used ATI’s RADEON 9700 PRO GPU, the paper’s discussion of “common graphics hardware” indicates that its neural network could have been implemented on any commonly available graphics hardware.

159. To achieve even further computational efficiency than possible with off-the-shelf hardware, it is my opinion that a POSITA would have been motivated to implement the artificial neural network of Oh using a state-of-the-art GPU-accelerated parallel processing system, such as that disclosed in Kirk. Indeed, because Kirk describes a system that uses multiple execution pipelines that can execute the same program on different portions of an input data set at the same time, a POSITA would have understood that the system of Kirk would greatly reduce execution time as compared to other processing architectures on which Oh’s neural network could be implemented.

160. In my opinion, a POSITA would have understood numerous advantages to applying the artificial neural network of Oh on the hardware of Kirk. The system

described in Kirk performs the types of parallel computations required for artificial neural networks to operate. Although Kirk's description focuses on processing "graphics data," Kirk explains that "graphics data is any data that is input to or output from computation units within [the] Programmable Graphics Processor 105." Ex1005, 3:35–41. Therefore, Kirk's parallel execution pipelines (1240) can process data associated with Oh's artificial neural network.

161. Moreover, it is my opinion that the system described in Kirk would be ideal for implementing the artificial neural network described in Oh. Kirk's system allows the results of its processing pipeline to be "written to a buffer in graphics memory to be read from during a subsequent pass," as is required for Oh's artificial neural network. Ex1005, 4:16–20. And Oh describes that, in order to increase efficiency in the artificial neural network, "[t]he result of the previous layer is saved in the form of a render target texture, which is then used as an input for the next layer." Ex1007, 3. Because Kirk's system leverages shared memory (*e.g.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250), and explicitly discusses the use of the results of one pass in a subsequent pass, it is my opinion that Kirk's system would have allowed the iterative computations described in Oh.

162. Additionally, I understand that Kirk is assigned to NVIDIA, who pioneered GPUs and GPGPU computing in the early 2000s. Ex1005, Cover. In my

opinion, a POSITA would have expected that the most advanced GPUs for performing GPGPU computations (including for artificial neural networks) to come from NVIDIA. Thus, it is my opinion that it would have been particularly important for a POSITA to seek out state-of-the-art GPU acceleration hardware, including hardware from NVIDIA, such as that disclosed in Kirk, which is assigned to NVIDIA, when implementing a real-time simulation like that disclosed in Oh.

163. In my opinion, a POSITA would have understood that the Kirk and Oh combination simply applied a known technique (*e.g.*, the artificial neural network of Oh) to a known device (*e.g.*, the GPU-accelerated parallel processing system of Kirk) ready for improvement to yield predictable results (*e.g.*, fast and efficient computations for an artificial neural network requiring real-time data processing).

164. Finally, in my opinion, a POSITA would have had a reasonable expectation of success in implementing the artificial neural network of Oh on the system of Kirk. As noted above, Kirk's system was designed to run many tasks (threads) at the same time. Simulating multi-layer artificial neural networks requires dividing the computation into smaller, discrete tasks, and thus, in my opinion, a POSITA would have expected the system of Kirk to be able to use parallel processing to execute the computations required in Oh. Furthermore, it is my opinion that a POSITA would have expected success in the combination because Kirk's system has a memory architecture (*e.g.*, shared memory) suitable for swapping

pointers of outputs and inputs, as a POSITA would have done when implementing Oh's teachings in Kirk's system.

## **2. Combination of Tamura with Kirk and Oh**

165. In my opinion, a POSITA would have been additionally motivated to combine Tamura with Kirk and Oh. For applications with real-time inputs, such as an implementation of Oh's multi-layer neural network for real-time video recognition, a POSITA would have been motivated to implement the data transfer and computation techniques for artificial neural networks of Tamura, which improve processing time. 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. In my opinion, a POSITA would have been motivated to apply Tamura's data transfer and computation techniques for artificial neural networks to accelerate the system of the Kirk and Oh combination.

166. As I have previously explained, Tamura describes a system that provides improved processing times for artificial neural networks. Ex1008, ¶ 30. Tamura teaches that while data previously input is processed and calculations are performed, new data is input and saved to memory. *Id.*, ¶29. In other words, Tamura teaches that data input should occur simultaneously with computations. Thus, to the extent that Kirk and Oh do not explicitly teach a POSITA that data processing and

data input should occur simultaneously, it is my opinion that a POSITA would have been motivated to apply the teachings of Tamura with Kirk and Oh. It is also my opinion that a POSITA would have reasonably expected the combination of Tamura with Kirk and Oh would improve computational efficiency of an artificial neural network simulation, as taught in Tamura.

167. For real-time applications that perform functions such as video analysis, a POSITA would have understood that it is of particular importance to design a system to be able to simultaneously and continuously receive and process data from video, including from a video camera. Therefore, it is my opinion that a POSITA would have looked to the available art to ensure the ability for a system to handle and keep up with real-time data input. Tamura discloses one such technique. Tamura 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.” Ex1008, ¶37. Therefore, it is my opinion that a POSITA would have understood that the technique described in Tamura could be applied to Kirk’s GPU-accelerated system.

168. In sum, it is my opinion that a POSITA would have understood that the combination of Tamura with Kirk and Oh simply applied a known technique (*e.g.*, efficient data processing and data input in an artificial neural network) to the known device(*e.g.*, the GPU-accelerated parallel processing system of the Kirk and Oh

combination) ready for improvement to yield the predictable results (*e.g.*, fast and efficient computations for an artificial neural network requiring real-time data processing).

### **3. Element-by-Element Analysis**

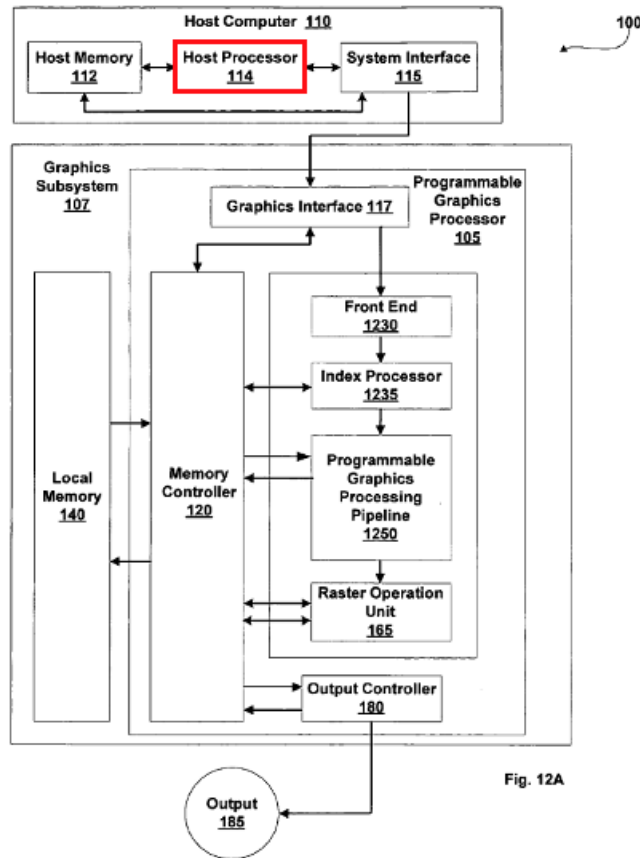
169. As I explain below, it is my opinion that the combination of Kirk and Oh (Ground 1), Kirk, Oh, and Tamura (Ground 2), Kirk in view of Oh and GPU Gems (Ground 3), and Kirk in view of Oh, Tamura, and GPU Gems (Ground 4) teach and render obvious all the elements of challenged claims.

#### **a) Claim 1**

##### **i. Element 1[pre]: “A computer system, comprising”**

170. To the extent the preamble is limiting, in my opinion, the combination of Kirk and Oh discloses Element 1[pre].

171. Kirk discloses, for example, a “Computer System” with a “programmable graphics processor” for processing data.” Ex1005, 1:40-47. Kirk’s “Programmable Graphics Processor 105 uses memory to store graphics data and program instructions, where graphics data is any data that is input to or output from computation units within Programmable Graphics Processor 105.” *Id.*, 3:35–41, *see also id.*, 26:24–39, 26:56–61, Figs. 12A, 12B.



*Id.*, Fig. 12A (annotated).

**ii. Element 1[a]: “a central processing unit to receive input data”**

172. In my opinion, the combination of Kirk and Oh discloses Element 1[a].

173. Kirk includes *a central processing unit* (e.g., Host Processor 114). *Id.*, 3:17–46, Figs. 1A, 12A. In my opinion, a POSITA would have understood that, in order for data to be processed by the Graphics Subsystem 107, Host Processor 114 would receive input data and process and pass that input data to the Graphics Subsystem 107. *Id.*, 1:40–47, 3:35–41, 26:24–39, 26:56–61, Figs. 12A, 12B. In the

combined Kirk and Oh system, the Host Processor 114 could receive “video data”, e.g., from a camera, as input. Ex1007, 1, 3.

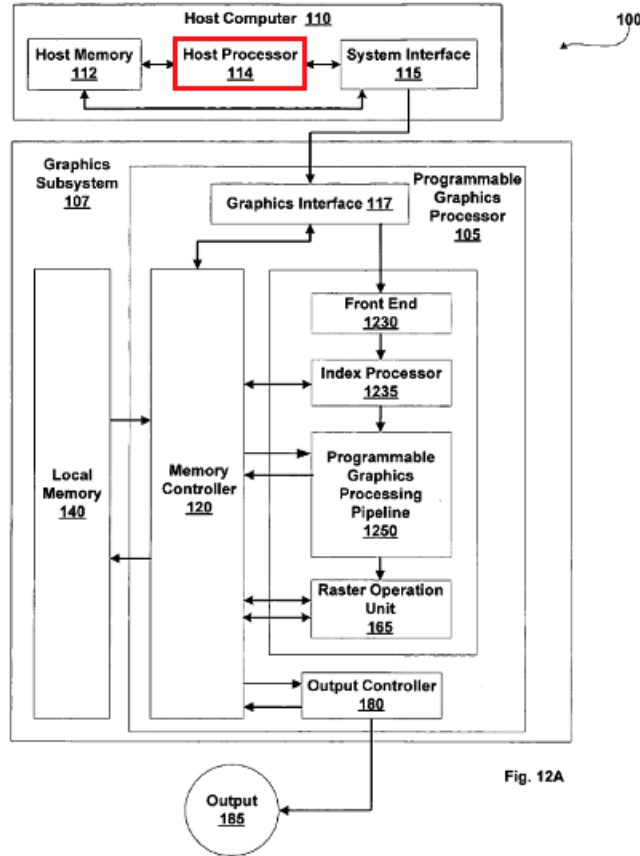


Fig. 12A

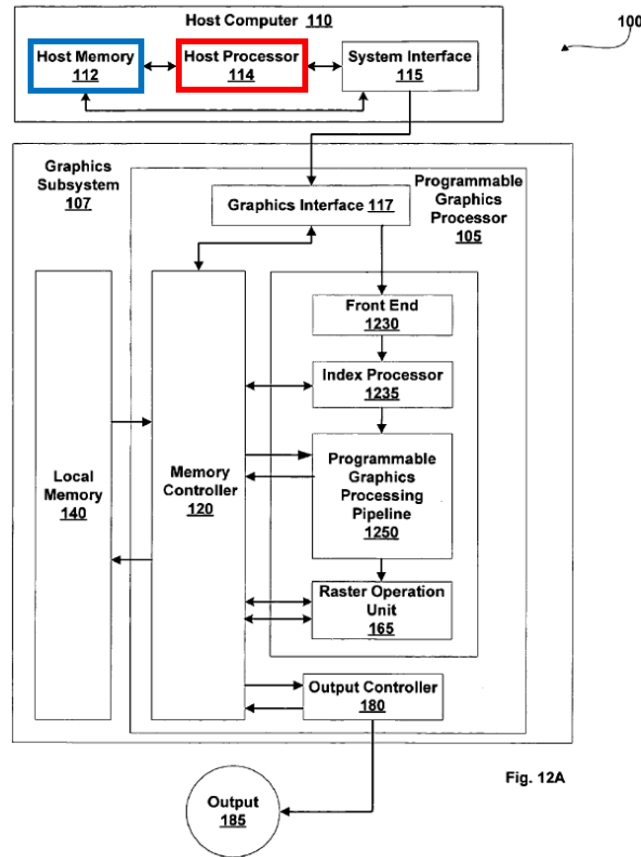
*Id.*, Fig. 12A (annotated).

- 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”**

174. In my opinion, the combination of Kirk and Oh discloses Element 1[b].

175. As I illustrate in annotated Figure 21A of Kirk below, Kirk discloses a *main memory*, Host Memory 112, which is connected to the Host Processor 114 via a bus, which were well known at the time. Ex1005, 3:17–31. Kirk explains that Host

Memory 112 stores input data (e.g., video data) for analysis, which the CPU can access and send to Graphics Subsystem 107 for processing. *Id.*,3:32-34, 3:50–67, 4:26–31, 5:1-2 (“data stored in . . . Host Memory 112”).



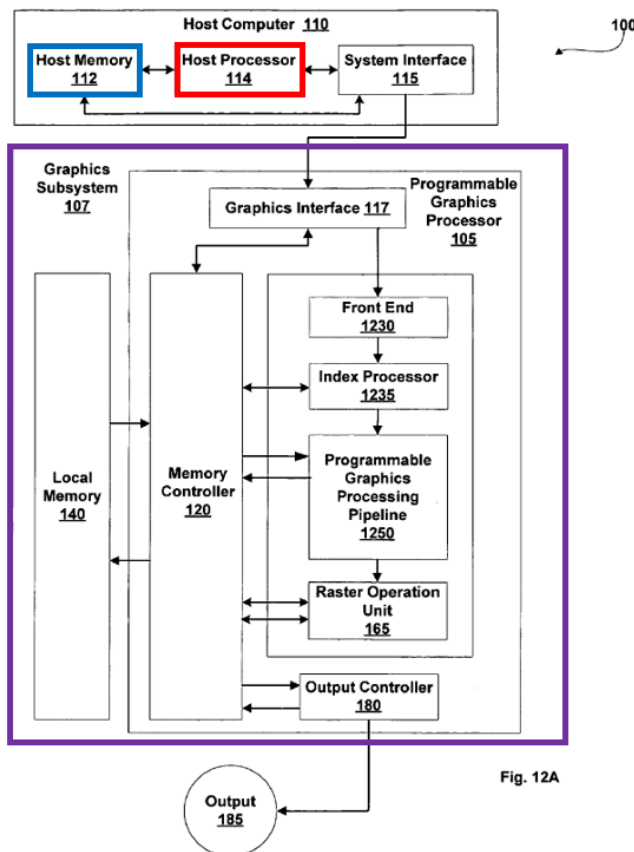
*Id.*, Fig. 12A (annotated).

176. Kirk explains that the Host Memory 112 can store “program instructions and data.” *Id.*, 5:1-2 (“data stored in . . . Host Memory 112”). Given these disclosures in Kirk, it is my opinion that a POSITA would have understood that Host Memory 112 (*main memory*) is operably coupled to Host Processor 114 (*central processing unit*) via a bus (depicted as black arrows in Fig. 12a).

- 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”**

177. In my opinion, the combination of Kirk and Oh discloses Element 1[c].

178. As I illustrate in annotated Figure 12A of Kirk below, the Kirk computer system includes an *accelerator*, Graphics Subsystem 107, that is coupled to Host Processor 114 (*central processing unit*) and the Host Memory 112 (*main memory*) via the bus:



*Id.*, Fig. 12A (annotated).

179. Kirk explains that the CPU can access and send data stored in Host Memory 112 to Graphics Subsystem 107 for processing. *Id.*, 3:32–34, 3:50–67, 4:26–31. The Host Processor 114 (red) communicates with Host Memory 112 (blue) through a bus (depicted as black arrows) and with Graphics Interface 117 (orange)—and therefore, the Graphics Subsystem 107 (purple)—through System Interface 115 (communication depicted as green arrow) and the bus (black arrows). *Id.* at 3:22–31 (“Host computer 110 includes Host Processor 114 that may include a system memory controller to interface directly to Host Memory 112 or may communicate with Host Memory 112 through a System Interface 115. System Interface 115 may be an I/O (input/output) interface or a bridge device including the system memory controller to interface directly to Host Memory 112.”). Thus, Graphics Subsystem 107 (i.e., an accelerator) receives at least a portion of the input data from Host Processor 114 (i.e., main memory).

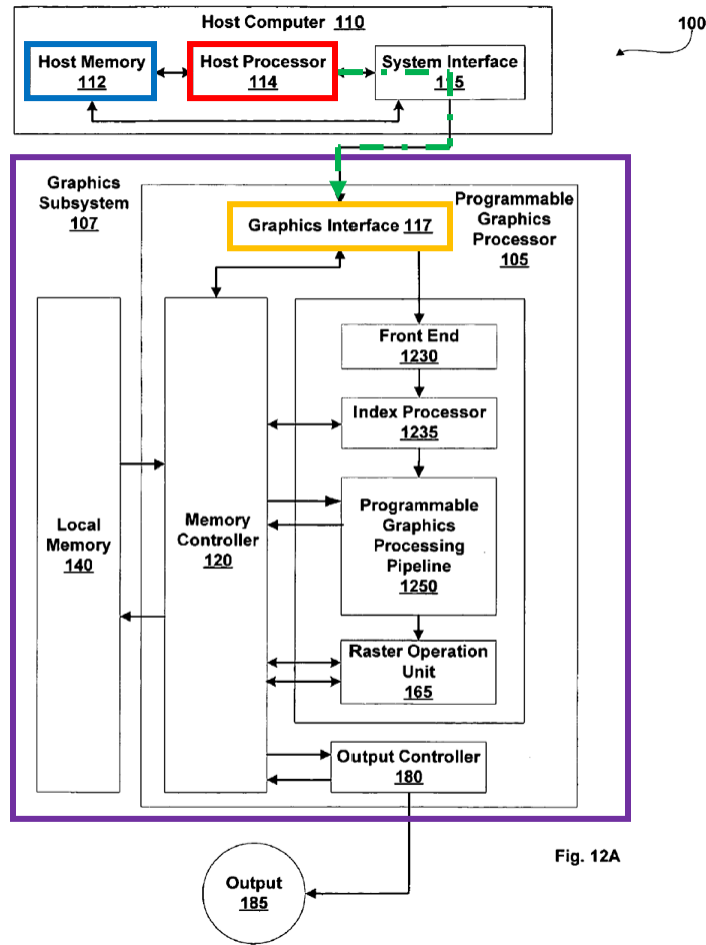


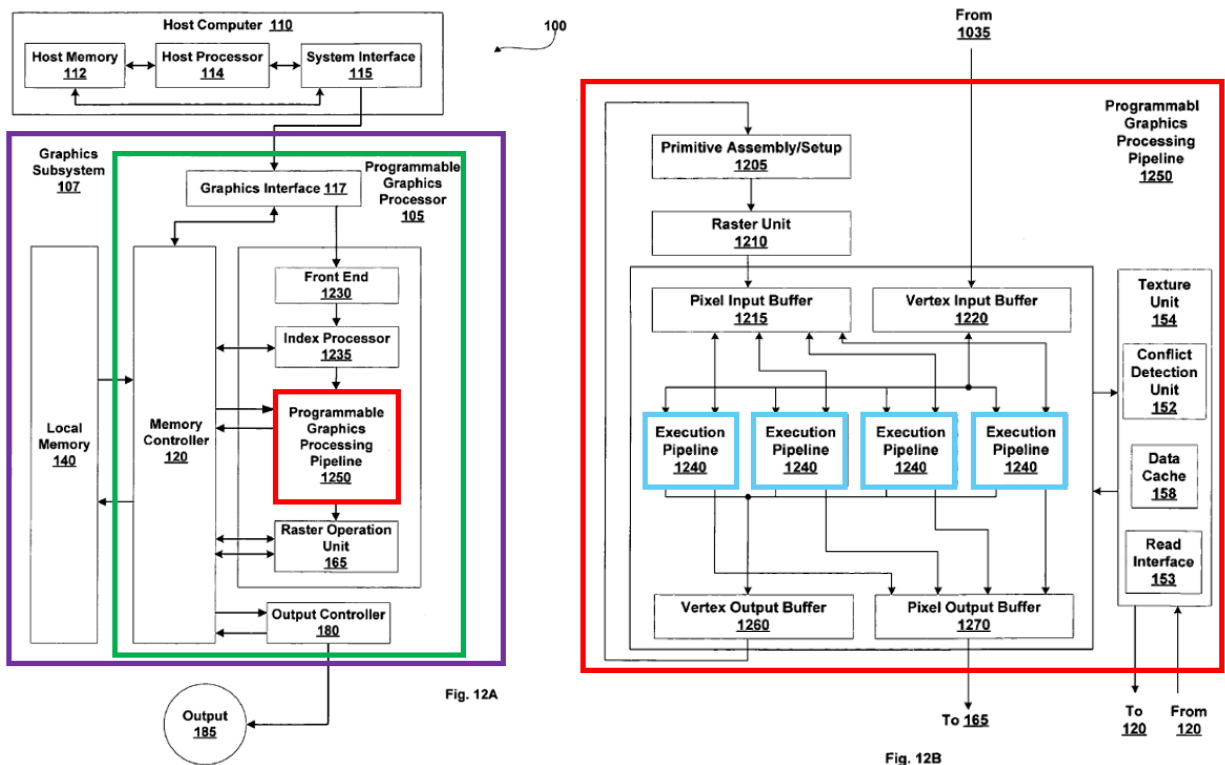
Fig. 12A

*Id.*, Fig. 12A (annotated).

- 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”**

180. In my opinion, the combination of Kirk and Oh discloses Element 1[c][i].

181. As I illustrate in annotated Figure 12A of Kirk below, Kirk discloses Graphics Subsystem 107 (purple), Programmable Graphics Processor 105 (green), and Programmable Graphics Processing Pipeline 1250 (red). *Id.*, 3:32–46, 26:24–39. Programmable Graphics Processor 105 (green), which includes Programmable Graphics Processing Pipeline 1250 (red), is within Graphics Subsystem 107 (purple). Programmable Graphics Processor 105 as a whole and Programmable Graphics Processing Pipeline 1250 each are a “graphics processing unit.” *Id.*, 26:24–61; Ex1001, 1:32–34 (“Graphics Processing Units (GPUs) ... are considered highly parallel processors dedicated to fast computation of graphical content.”).



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

182. Kirk provides additional details concerning the Programmable Graphics Processing Pipeline 1250 in Figure 12B, which I have annotated and reproduced above. Kirk explains that the Programmable Graphics Processing Pipeline 1250 (red) includes multiple Execution Pipelines 1240 (blue). Ex1005, 27:2–11. Each of these pipelines “includes at least one multithreaded processing unit.” *Id.* Kirk also explains that each of these execution pipelines communicates with graphics memory (including Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) “to read program instructions and graphics data stored in buffers in graphics memory.” *Id.*, 27:12–28.

183. Kirk explains that, in operation, “[a] graphics program . . . is executed within one or more Execution Pipelines 1240 as a plurality of threads where each vertex or fragment to be processed by the program is assigned to a thread.” *Id.*, 27:40–43. The results of the computations are then sent to memory, such as Local Memory 140, Host Memory 112, or memory within the programmable graphics processing pipeline. *Id.*, 3:35–46. Kirk explains that data produced by the processing pipeline, (*i.e.*, “output data”) “may be written to a buffer in graphics memory to be read from during a subsequent pass.” *Id.*, 4:16–20. Given these teachings in Kirk, it is my opinion that a POSITA would have understood that Programmable Graphics Processor 105, including its Programmable Graphics Processing Pipeline 1250,

therefore performs a sequence of computations on at least a portion of the input data so as to generate output data.

184. In the Kirk and Oh combination, the sequence of computations represents an artificial neural network. Oh discloses computations, executed on a GPU, that represent a multi-layer artificial neural network. Ex1007, 1, 3. Oh's multi-layer network includes an input layer including nodes (*i.e.*, neurons), multiple hidden layers comprising 30 nodes each, and an output later having a single node. *Id.*, 3. Each of these layers represents a layer of neurons in the artificial neural network. *Id.* At each layer, the artificial neural network relies on matrix multiplication and a sigmoid function to determine the output which are then input into the next layer. *Id.*

185. Therefore, when Kirk is combined with Oh, it is my opinion that a POSITA would understand the sequence of computations on Programmable Graphics Processing Pipeline 1250 of Kirk represents an artificial neural network, with the intermediate results (the intermediate computations in the sequence) representing the respective layers of the multi-layer artificial neural network described in Oh.

- 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”**

186. In my opinion, the combination of Kirk and Oh discloses Element 1[c][ii].

187. As I have previously explained, and as illustrated in annotated Figure 12A below, Kirk's computer system includes a Graphics Subsystem 107 that includes an *accelerator memory* (e.g., Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) (blue) coupled to the graphics processing unit (e.g., Programmable Graphics Processor 105 or Programmable Graphics Processing Pipeline 1250). Ex1005, 3:32–46, 26:24–39. Programmable Graphics Processor 105 interfaces with the host system through Graphics Interface 117 (orange) and with Local Memory 140 (blue) through Memory Controller 120 (brown). *Id.*, 3:32–67, 26:24–39. Local Memory 140 stores the results of the sequences of computations performed in Programmable Graphics Pipeline 1250 (red). *Id.*, 3:32–46, 5:8–11, 5:28–32, 7:17–25, 26:32–61.

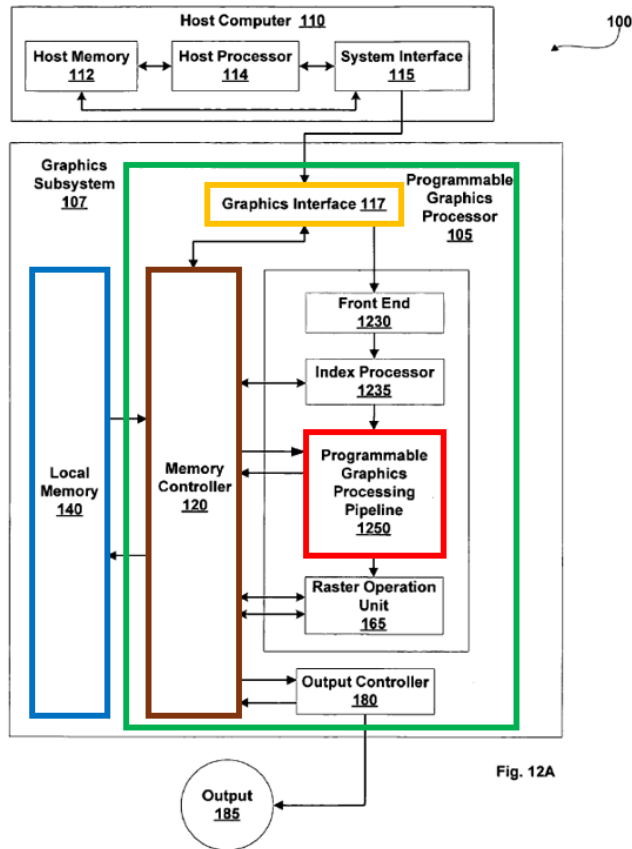


Fig. 12A

*Id.*, Fig. 12A (annotated).

188. Additionally, Kirk explains that “Execution Pipelines 1240 are optionally configured . . . such that data processing operations are performed in multiple passes through at least one multithreaded processing unit within Execution Pipelines 1240.” *Id.*, 28:33–37. Thus, results of a sequence of computations can also be stored in memory within Programmable Graphics Processing Pipeline 1250, such as Vertex Output Buffer 1260 or Texture Unit 154. *Id.*, 27:29–31, 27:24–28, Fig. 12B. For example, Kirk discloses that the outputs of Execution Pipelines 1240 are “processed samples, such as vertices, that are stored in a Vertex Output Buffer 1260 in a register file, FIFO memory, cache, or the like (not shown).” *Id.*, 27:29–31. “In

another alternate embodiment,” Kirk discloses that “a Texture Unit 154 is included in each Execution Pipeline 1240.” *Id.*, 27:24–28.

189. Thus, it is my opinion that a POSITA would have understood this memory to be *accelerator memory* as claimed because these memory locations of Kirk’s computer system (including Local Memory 140 and components not illustrated, such as memory within Programmable Graphics Processing Pipeline 1250 (*e.g.*, Vertex Output Buffer 1260)), are located within Graphics Subsystem 107 (an “accelerator”).

**vii. Element 1[d]: “a controller, operably coupled to the at least one graphics processing unit and the accelerator memory”**

190. In my opinion, the combination of Kirk and Oh discloses Element 1[d].

191. Kirk’s system includes a “controller” *e.g.*, Graphics Interface 117, a Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180. *Id.*, 3:32–67, 26:24–61. These components (individually and collectively, annotated in yellow) are a “controller” because they control operation of Programmable Graphics Processor 105, including receiving, interpreting, formatting, and sending data and commands to the Programmable Graphics Processing Pipeline 1250 for execution. *Id.*; *see also* Ex1001, 5:17–29 (describing “controller” in the ’438 patent). Like the “controller” in the ’438 patent, the controller of Kirk handles the set up and control of the accelerator’s computations,

so that “the CPU is freed from this function and is dedicated to other tasks.” Ex1001, 2:36–37, 5:17–29 (describing “controller” in the ’438 patent).

192. As I have illustrated below in annotated Figure 12A of Kirk, the Graphics Interface 117, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180 are coupled to the accelerator memory (*i.e.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) through the Memory Controller 120 and to Programmable Graphics Processing Pipeline 1250. Ex1005, 3:47–67, 26:24–61, Fig. 12A. And Graphics Interface 117, Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180 are coupled to programmable Graphics Processor 105 as they are a part of it. Thus, in my opinion, a POSITA would have understood that Graphics Interface 117, Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180 collectively and individually would constitute a “*controller*.”

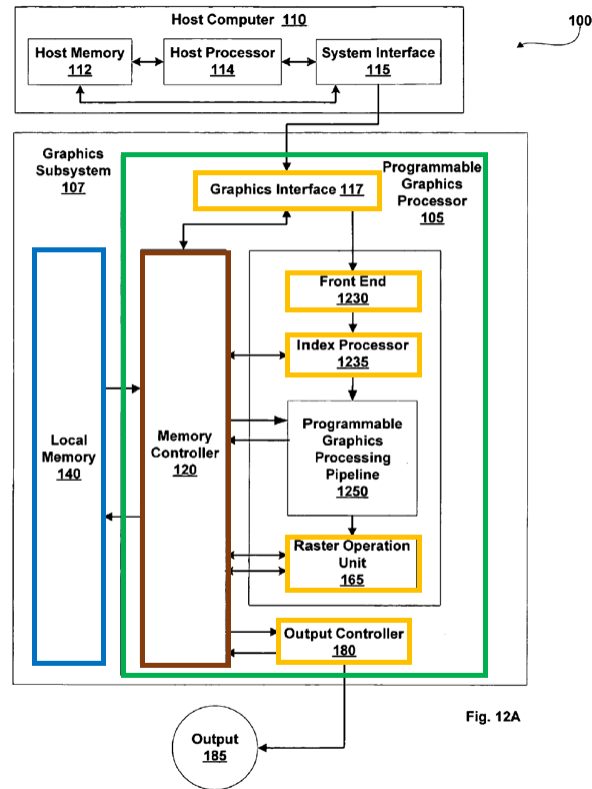
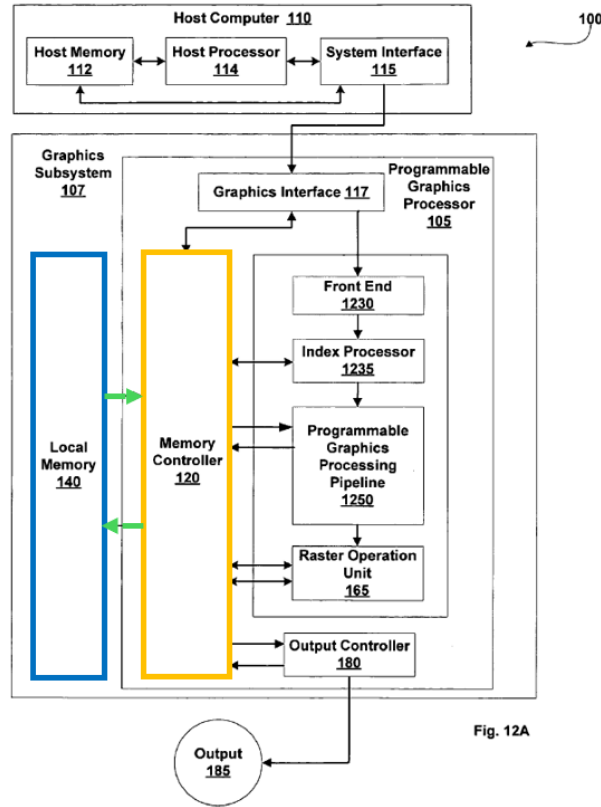


Fig. 12A

*Id.*, Fig. 12A (annotated).

193. Additionally, as I have previously explained, Kirk’s computer system includes Memory Controller 120 (annotated in yellow below), which communicates with the Programmable Graphics Processing Pipeline 1250, and transfers data into and out of Local Memory 140 (annotated in blue below). *Id.*, 3:47–67, 4:26–31, 26:24–39. I understand that in the litigation between Petitioner NVIDIA and Patent Owner Neural AI, Patent Owner has contended in its Infringement Contentions that a memory controller in the accused products (like Memory Controller 120 in Kirk) is the claimed “controller.” Ex1010, 55–56. Patent Owner asserts that “the GPU architecture of the Accused Products implement *memory*. As an example, Maxwell-

architecture GPUs . . . includes ‘memory controllers’ (*e.g., accelerator controller*) and ‘DRAM’ (*e.g., accelerator memory*).” *Id.* Therefore, additionally and alternatively, under Patent Owner’s contentions and view of the claims, the memory controller disclosed in Kirk (either alone or in combination with Kirk’s Graphics Interface 117, Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180) can be the claimed “accelerator controller.” I disagree with Patent Owner’s contentions that a standard memory controller constitutes a “controller” within the meaning of the claims of the ’438 patent and understand that Petitioner disagrees as well. However, for purposes of this Petition only, I understand that Petitioner accepts Patent Owner’s allegation that “controller” is broad enough to include the standard memory controller accused by Patent Owner.



*Id.*, Fig. 12A.

**viii. Element 1[d][i]: “to initialize textures and shaders in the accelerator memory for performing the sequence of computations”**

194. In my opinion, the combination of Kirk and Oh discloses Element 1[d][i].

195. When Kirk is used for numerical computations (*e.g.*, an artificial neural network), the “controller” (*i.e.*, Graphics Interface 117 along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180) initializes textures and shaders in memory of Graphics Subsystem 107 (*e.g.*, Local Memory 140 and/or memory within Programmable Graphics Processing Pipeline

1250). Ex1005, 3:35–67, 4:21–56 (“The program instructions configure subunits within Geometry Processor 130, Rasterizer 150 and Fragment Processing Pipeline 160. The program instructions and data are stored in graphics memory.”), 6:4–25.

196. For example, with respect to initializing textures, “[d]ata and program instructions received at Graphics Interface 117 can be . . . written to Local Memory 140 through Memory Controller 120.” *Id.*, 3:50–54; *see also id.*, 3:32–46, 4:57–5:11, 5:12–33, 6:25–33, 26:62–27:11. The data and program instructions in Local Memory 140 are then available for use by computation units within Programmable Graphics Processor 105, such as Programmable Graphics Processing Pipeline 1250. *Id.*, 3:35–46. The “data” referenced above are “textures” within the meaning of the ’438 patent: The ’438 patent defines a “texture” as a “data array.” Ex1001, 5:20–22 (“[T]he term ‘texture’ in this document refers to a data array . . . .”). Kirk’s system further initializes textures (data arrays) for storing output data from computations. Ex1005, 5:24–45, 6:4–55, 11:22–41, 28:21–32.

197. With respect to initializing shaders, Kirk’s “program instructions” are “shaders” within the meaning of the ’438 patent. “[P]rogram instructions received at Graphics Interface 117 can be . . . written to Local Memory 140 through Memory Controller 120.” Ex1005, 3:50–54. The program instructions are then available to be loaded into the Programmable Graphics Processing Pipeline to be executed. *Id.*, 3:35–46, 5:34–45, 6:4–43, 26:24–39, 27:12–39, 28:34–37. The ’438 patent defines

a “shader” as a “GPU program.” Ex1001, 5:24–26 (“[T]he term ‘shader’ in this document refers to a GPU program unless otherwise specified.”).

198. In the Kirk and Oh combination, the “shaders” that are initialized in the memory of Graphics Subsystem 107 would include the equations that would be applied to each layer of the multi-layer neural network. The “textures” initialized in memory of Graphics Subsystem 107 would include the initial neural network parameters and the weights to be applied at each layer. Additionally, textures and shaders may be stored in Kirk’s Texture Unit 154 and/or Conflict Detection Unit 152. Ex1005, 6:4–25, Fig. 12B.

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

200. To the extent not explicitly disclosed in Kirk or Oh, in my opinion, 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 components performing computations. In order for Programmable Graphics Processing Pipeline 1250 to process the input data using the designated sequence of computations

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

**ix. Element 1[d][ii]: “to control performance of the sequence of computations by the at least one graphics processing unit”**

201. In my opinion, the combination of Kirk and Oh discloses Element 1[d][ii].

202. As I have previously explained, Kirk’s “controller” (i.e., Graphics Interface 117 along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180) controls performance of the computations in Programmable Graphics Processing Pipeline 1250. Ex1005, 3:35–67, 4:21–56, 27:2–28. Programmable Graphics Processing Pipeline 1250 comprises multiple Execution Pipelines 1240, each of which “includes at least one multithreaded processing unit.” Ex1005, 27:2–11. Each of the execution pipelines communicates with the graphics memory (including Local Memory 104) “to read program instructions and graphics data stored in buffers in graphics memory.” *Id.*, 27:12–28. Programmable Graphics Processing Pipeline 1250 “contain[s] one or more programmable processing units to perform a variety of specialized functions,” including “table lookup, scalar and vector addition, multiplication, division, coordinate-system mapping, calculation of vector normal, tessellation, calculation of derivatives, interpolation, and the like.” *Id.*, 26:44–51.

203. From this disclosure, it is my opinion that a POSITA would have understood that because the instructions for performing the sequence of computations in Programmable Graphics Processing Pipeline 1250 (*i.e.*, a graphics processing unit) and the data for such computations are provided based on instructions from the “controller,” the controller “control[s] performance of the sequence of computations by the at least one graphics processing unit.”

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

- 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”**

205. In my opinion, the combination of Kirk and Oh (and optionally in view of Tamura) discloses Element 1[d][iii].

206. As I have previously explained with respect to Elements 1[c] and 1[c][ii], Graphics Subsystem 107 includes “*accelerator memory*” (*e.g.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250)

and a *graphics processing unit* (e.g., Programmable Graphics Processing Pipeline 1250 or Programmable Graphics Processor 105) in which data is processed according to instructions stored in memory. *Id.*, 3:32–46, 26:24–61. Kirk explains that input data, output data, and commands, are stored in Local Memory 140. *Id.*, 3:34–46. Furthermore, when Graphics Subsystem 107 receives input data from the host computer, Graphics Interface 117 transfers the data to Local Memory 140. *Id.*, 3:50–58.

207. In the Kirk and Oh combination, the input data, which can be one or more video frames, is transferred into the accelerator memory during performance of the intermediate computations in the sequence of computations. Ex1005, 3:32–67. For example, because the input of Oh’s neural network is “video data,” which may even be real-time video data, in order for Oh’s system to keep up with the real-time data input, once video data for a first frame is transferred into accelerator memory and the artificial neural network computations have begun, input data for subsequent frames would be transferred into accelerator memory while the computations are being performed on the earlier input data. Ex1007, 3–5.

208. Tamura also teaches “transfer[ring] 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” as claimed. Tamura teaches, in the context of simulating an artificial neural

network, receiving input data and writing it to memory while calculations are performed on previously received input data (i.e., the previously received input data is processed through the layers of an artificial neural network). Ex1008, ¶¶29–30, 35. In the Kirk, Oh, and Tamura combination, the real-time data on which image recognition is performed would have been transferred into the accelerator memory while the programmable graphics processing pipeline performs the intermediate computations of the artificial neural network described in Oh. In my opinion, a POSITA would have understood that when this teaching of Tamura is applied to the Kirk and Oh combination, additional frames from the video input is transferred into accelerator memory during performance (by the accelerator) of intermediate computations in the artificial neural network on one or more earlier frames of input data.

209. In my opinion, it would have been obvious to a POSITA, in view of Tamura, to implement the Kirk and Oh combination 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. 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”**

210. In my opinion, the combination of Kirk and Oh discloses Element 1[d][iv].

211. Kirk’s “controller” (*i.e.*, Graphics Interface 117 along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180) transfers output data from accelerator memory (*i.e.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) to main memory (*i.e.*, Host Memory 112). Ex1005, 5:8–11. For example, Kirk discloses that “output data from Raster Operation Unit 165 is written back to Local Memory 140 or Host Memory 112 at the x,y position associated with the output data.” *Id.*, 6:49–52.

212. In another example, Kirk discloses that “the read interface in Output Controller 180 generate[s] address and control signals to Memory Controller 120.” *Id.*, 3:64–67, Fig. 12B. In yet another example, Kirk discloses that “Write Interface 157 within Raster Operation Unit 165 writes the optionally processed shaded fragment data to a buffer stored in Local Memory 140 or Host Memory 112, via Memory Controller 120.” *Id.*, 5:60–6:3. For example, data from Local Memory 140 to Host Memory 112 is transferred through Graphics Interface 117. *Id.*, 11:32–41, Fig. 12A. Because Kirk discloses that its output data can be written to Local Memory

140 or Host Memory 112, in my opinion, a POSITA would have understood that Kirk suggests that output data written to Local Memory 140 (*accelerator memory*) could be transferred to Host Memory 112 (*main memory*).

213. The transfer of data occurs based on instructions received from Kirk's "controller" (i.e., Graphics Interface 117 along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180). Ex1005, 5:8–11 ("The output data from Raster Operation Unit 165 is written back to Local Memory 140 or Host Memory 112 at the x,y position associated with the output data."), 6:49–52 ("A Write Interface 157 within Raster Operation Unit 165 writes the optionally processed shaded fragment data to a buffer stored in Local Memory 140 or Host Memory 112, via Memory Controller 120."), 5:60–6:3. For example, data from Local Memory 140 to Host Memory 112 is transferred through Graphics Interface 117. *Id.*, 11:32–41, Fig. 12A. As in the '438 patent, the "controller" of the Kirk and Oh combination is part of the "accelerator," which provides accelerated, parallel data processing, separately from the CPU. Ex1001, 3:40 (referring to a "GPU accelerator"). Furthermore, in the alternative mapping in which Memory Controller 120 is part of the "controller," and under Patent Owner's theories of infringement, Memory Controller 120 transfers output data to Host Memory 112, through Graphics Interface 117. *Id.*, 11:32–41.

214. To the extent Kirk does not expressly disclose transferring output data from Local Memory 140 (*accelerator memory*) to Main Memory 112 (*main memory*), as opposed to writing the data directly to Main Memory 112, a POSITA would have considered this obvious to in view of Kirk's own disclosure. Indeed, it would have been efficient to write output data to Local Memory 140 so that it can be used in other calculations (*see id.* at 4:16–20) and, thereafter, copy that output data to Host Memory 112 (i.e., *transfer the output data to main memory*) while the further calculations occur (e.g., during a second computational cycle).

215. Moreover, in the Kirk and Oh combination, the transfer of output data to main memory occurs during performance of the intermediate computations in the sequence of computations by the at least one graphics processing unit. Where an artificial network, such as the one described in Oh, is processing data in real-time, output data from one pass through the multi-layer neural network is transferred to system memory while the GPU continues to perform computations (i.e., there are overlapping computation and data transfers). Ex1007, 2–4. Kirk's system provides flexibility in when and where data is stored. *Id.*, 5:8–11, 6:49–52, 5:60–6:3. Thus, in the Kirk and Oh combination, when results are written to Host Memory 112, it is my opinion that a POSITA would have understood Kirk to teach *first* writing data to memory in the accelerator (because that memory is closer to the processing engines and faster to access) and *then* copying (transferring) the data to system memory. *Id.*,

4:16–20, 5:8–11. In short, in the Kirk and Oh combination, there are overlapping computation and data transfers.

216. At a minimum, it is my opinion that a POSITA would have considered it obvious for the system of the Kirk and Oh combination to transfer at least a portion of the output data from the accelerator memory from one pass through the artificial neural network to the main memory during performance of the intermediate computations in other threads or on other input data. The objective of Oh is to provide text detection in real-time video. Ex1007, 3-5. Unless there are overlapping computations and data transfer back to main memory, the system in the Kirk and Oh combination, would not be able to achieve its intended objective.

217. Indeed, in my opinion, a POSITA implementing the artificial neural network described in Oh for video data would see a benefit in receiving real-time results of the pattern and text recognition in the data. To accomplish this, it is my opinion that a POSITA would have designed the system to transfer output data from Graphics Subsystem 107 to main memory while performing computations on later-received data. In my opinion, because accelerator memory is more closely coupled to the processing engines and outputs can be used as inputs in subsequent time steps, a POSITA would have found it obvious to design the system such that 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. Ex1005, 4:16–20; Ex1007, 2–3.

**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.”**

218. In my opinion, the combination of Kirk and Oh discloses Claim 2.

219. Kirk discloses “[a]n application programming interface for a programmable graphics processor” on the CPU, including allowing the data for processing to be “generated . . . by a human, or the like.” Ex1005, 1:40-42, 6:30–33, 18:60–19:67, Figs. 10A–10C. Just like the user interaction of the ’438 patent (Ex1001, 8:10–46), an API like that disclosed in Kirk allows setting and editing of the parameters and configurations for the computations and triggering the system to acquire the inputs. Ex1005, 6:30–33, 18:60–19:3. At a minimum, because Kirk already discloses an API, in my opinion, it would have been obvious for a POSITA to implement an API that, through user interaction, to allows the user to trigger the CPU to receive input data—*i.e.*, so the data can be received and sent to the GPU for processing. *Id.* Systems such as those in Kirk require a triggering event to begin the input and processing of data, and it is my opinion that it would have been an obvious design choice for a POSITA to implement a user-interactive GUI or command interface with which a user can interact and in response to which the CPU can receive input data. *Id.*

c) **Claim 3**

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”**

220. In my opinion, the combination of Kirk and Oh discloses Element 3[a].

221. One of the possible inputs to the combined system of Kirk and Oh, is “video data.” Ex1007, 3. The CPU in the Kirk and Oh combination receives the video input at a first rate (*e.g.*, up to 30 fps or any other known video rate). Video can be input at many different rates, such as 30 frames per second, and at different resolutions, and all data that is transferred has a data transfer rate—that rate is simply the speed at which the data is moved between the origin and the destination. 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. Thus, in my opinion, a POSITA would have understood the CPU in the combination of Kirk and Oh to receive the video input at a first rate. *Id.*

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”**

222. In my opinion, the combination of Kirk and Oh discloses Element 3[b].

Oh discloses that its text detection can be implemented using different M×M pixel region size selected for processing. Ex1007, 3. For example, Oh discloses one example in which the total processing time in the artificial neural network on a GPU is 0.061 seconds (excluding the time it took to create a texture). Ex1007, 4. This speed equates to about 16 sequences of operations per second. Thus, the Kirk and Oh combination discloses that the GPU of Kirk performs the sequence of computations at a different rate than the rate of data input, *i.e.*, a second rate.

223. Thus, in my opinion, the combination of Kirk and Oh discloses that its GPU performs the sequence of computations at a different rate (*second rate*) than the rate at which video data is received by its CPU (*first rate*).

**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.”**

224. In my opinion, the combination of Kirk and Oh disclose Claim 5.

225. *See* Element 1[d][iv]. Kirk discloses that “[g]raphics memory is any memory used to store graphics data or program instructions to be executed by Programmable Graphics Processor 105. Graphics memory may include portions of Host Memory 112, [and] Local Memory 140 directly coupled to Programmable Graphics Processor 105.” Ex1005, 3:39-46. Thus, in my opinion, Kirk discloses storing a copy of the output data in either or both of Host Memory 112 (*main memory*) or Local Memory 140 (*accelerator memory*) in Graphics Subsystem 107,

depending on user preference and programming. Indeed, in my opinion, it would have been obvious to a POSITA that a copy of the output data could be stored in both of Host Memory 112 and Local Memory 140 to reduce access speeds, maintain redundant copies to reduce the effect of data corruption, and/or allow concurrent access.

- 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.”**

226. In my opinion, the combination of Kirk and Oh discloses Claim 5.

227. *See* Element 1[c][i]. Oh discloses implementing a “NN” (neural network) on a GPU to classify input data. Ex1007, 1, 3. Oh’s artificial neural network includes multiple layers—an input layer, multiple hidden layers (with up to 30 nodes each), and an output layer. *Id.*, 3. At each layer, the artificial neural network performs computations, including matrix multiplication and a sigmoid function. *Id.*, 2–3. The intermediate results from each layer of computations are input into the subsequent layer. *Id.* In my opinion, a POSITA would have readily understood that the outputs of the computations at each layer represent outputs of a neuron in an artificial neural network. *Id.*

**f) Claim 6**

- 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”**

228. In my opinion, the combination of Kirk and Oh discloses Element 6[a].

229. Kirk discloses a first memory bank (*e.g.* a logical portion of Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) 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. For example, memory in Graphics Subsystem 107 (*i.e.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) stores “graphics data,” which includes “any data that is input to or output from computation units.” Ex1005, 3:32–38, 3:50–58. In my opinion, a POSITA would have understood that in the Kirk and Oh combination, the “graphics data” would include any parameters common to all of the computations in the sequence of computations, such as the weights to be used in the neural network of Oh. In an artificial neural network, certain data is common to all computations in a series of computations.

230. Kirk further teaches data being stored within “buffers” (*i.e.*, individual memory banks) within “graphics memory.” Ex1005, 1:26-33, 5:2-11, Fig. 3B. Thus,

in my opinion, a POSITA would understand that the portion of Local Memory 140 (“or buffer”) storing this data would comprise a “*first memory bank*.”

**ii. Element 6[b]: “a second memory bank to store data specific to at least one computation in the sequence of computations”**

231. In my opinion, the combination of Kirk and Oh discloses Element 6[b].

232. See Claim 4, Element 1[d][iv], Element 6[a]. Kirk discloses a second memory bank (*e.g.*, a logical portion of Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) 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). In Kirk, “[g]raphics memory is any memory used to store graphics data or program instructions to be executed by Programmable Graphics Processor 105. Graphics memory may include portions of Host Memory 112, [or] Local Memory 140 directly coupled to Programmable Graphics Processor 105.” Ex1005, 3:39–46. As I have explained, when implementing the neural network of Oh on Kirk’s computer system, the “graphics data” stored in Local Memory 140 would include input data (such as video data) for the neural network. This input data is specific to at least one computation in the sequence of computations. Ex1007, 2–4.

233. And as I have explained for Element 6[a], Kirk teaches data being stored within “buffers” within “graphics memory.” Ex1005, 1:26-33, 5:2-11, Fig.

3B. The portions of Local Memory 140 for the data I described above would be a “*second memory bank*.”

- 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.”**

234. In my opinion, the combination of Kirk and Oh discloses Claim 7.

235. *See* Element 1[d][iv]. As I have explained above, Kirk’s controller (*e.g.*, Graphics Interface 117 along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180, and alternatively with Memory Controller 130) is configured to transfer outputs from computations from accelerator memory (*e.g.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) to main memory (*e.g.*, Host Memory 112). Ex1005, 5:8–11, 6:49–52, 5:60–6:3, 11:32–41, Fig. 12A.

236. It is my opinion that a POSITA would have understood that it was desirable to avoid unnecessary data exchange between Graphics Subsystem 107 and Host Computer 110. For example, intermediate results from a multi-layer neural network (*e.g.*, outputs of a particular middle layer) are not used by the host system in the Kirk and Oh combination host system, and therefore transferring such intermediate results would be undesirable. *Id.*; Ex1007, 3–5. Thus, in the Kirk and Oh combination, 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). In my opinion, a POSITA would have understood that not transferring intermediate results from a neural network back to main memory would reduce data transfer via the bus.

237. Furthermore, in my opinion, in the combination of Kirk and Oh, 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 layer of the artificial neural network). Instead, as explained above, only the outputs of the final layer of the artificial neural network are transferred to main memory.

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

- 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.”**

239. In my opinion, the combination of Kirk and Oh discloses Claim 8.

240. See Element 1[d][iv]. In the Kirk and Oh combination, where the artificial neural network is processing video data with multiple frames (including potentially real-time data), the transfer of output data to main memory occurs after the GPU (e.g., Programmable Graphics Processing Pipeline 1250 or Programmable Graphics Processor 105) has begun to perform another sequence of computations of the artificial neural network.

241. As I have explained above, Kirk's controller (e.g., Graphics Interface 117 along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180, and alternatively with Memory Controller 130) is configured to transfer outputs from computations from accelerator memory to main memory. Kirk's system transfers output data from *accelerator memory* (i.e., Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) to *main memory* (i.e., Host Memory 112). Ex1005, 5:8–11, 6:49–52, 5:60–6:3, 11:32–41, Fig. 12A. When processing video data, as in Oh, where the output data is from the output layer of the artificial neural network, it is sent to system memory after the processing pipeline begins performing another sequence of computations (e.g., the computations in the layers of the multi-layer neural network) on new input data. Ex1007, 2–4.

242. In the combined Kirk and Oh system, the output data from the output layer of the artificial neural network can be sent to main 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. Ex1007, 2–4. Thus, in the combination of Kirk and Oh, where the artificial neural network is processing video data with multiple frames (including potentially real-time data), the transfer of output data to main memory would occur after the GPU (*e.g.*, Programmable Graphics Processing Pipeline 1250 or Programmable Graphics Processor 105) has begun to perform another sequence of computations of the artificial neural network. For example, where the output data is from the output layer of the artificial neural network, it is sent to system memory after the processing pipeline begins performing another sequence of computations (*e.g.*, the computations in the layers of the multi-layer neural network) on new input data. Ex1007, 2–4.

243. To the extent not disclosed in the Kirk and Oh combination, 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. Doing so would allow the Kirk and Oh combination system to keep up with the video data input and allow the system to immediately utilize the output data to display results to a system user.

- 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.”**

244. In my opinion, the combination of Kirk and Oh discloses Claim 9.

245. *See* Elements 1[d][iii], 1[d][iv]; Claim 8. As I have explained above, Kirk’s controller (*e.g.*, Graphics Interface 117, along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180, and alternatively with Memory Controller 130) is configured to initiate transfer of input data to Programmable Graphics Processing Pipeline 1250 (*see* Element 1[d][iii]) and to transfer outputs from computations from accelerator memory to main memory (*see* Element 1[d][iv]).

246. In the combination of Kirk and Oh, where the artificial neural network is processing video data with multiple frames (including potentially real-time data), the transfer of input and output data would occur in parallel with the GPU (*e.g.*, Programmable Graphics Processing Pipeline 1250 or Programmable Graphics Processor 105) performing another sequence of computations of the artificial neural network. For example, as I have explained above for Element 1[d][iii], data is continually transferred into the GPU to identify text in video frames. 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—doing so allows the GPU to perform its intended function of locating text in video. Ex1007, 2–4.

**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.”**

247. In my opinion, the combination of Kirk and Oh discloses Claim 10.

248. *See* Elements 1[d], 1[d][ii]. As I have explained above, Kirk discloses a “controller,” (*e.g.*, Graphics Interface 117, along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180, and alternatively with Memory Controller 130), which is operably coupled to the accelerator memory (*e.g.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) through Memory Controller 120. Ex1005, 3:17–67. Kirk’s controller controls execution of the sequence of computations by sending instructions for executing a sequence of computations to Programmable Graphics Processing Pipeline 1250 (*at least one graphics processing unit*). *Id.*, 26:24–39.

**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.”**

249. In my opinion, the combination of Kirk and Oh discloses Claim 11.

250. *See* Element 1[a]. Oh discloses “video data” (*i.e.*, real-time data) as input. Ex1007, 1 (“real-time processing”), 3.

251. In my opinion, a POSITA would have understood that video data can be captured by a video camera connected to Host Computer 110, which includes Host Processor 114, to acquire the input data in real time. This video camera would be connected to the combined Kirk and Oh system through, for example, a peripheral device interface and/or a bus, to provide input data to the CPU, and where video input is acquired via a camera connected to Host Computer 110 and then transferred to the GPU, the video camera is operably coupled to the CPU in the host system.

252. Furthermore, to the extent this limitation is not expressly disclosed, it would have been obvious. For example, in my opinion, it would have been obvious to a POSITA to use a video camera to capture video in which text must be recognized—the POSITA would have understood that an obvious application of Oh's text detection algorithm would be in a real-time system, with a video camera providing real-time input. In that combination, the video camera would be operably coupled to the CPU in order to obtain the video input.

**l) Claim 12**

- 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”**

253. To the extent the preamble is limiting, it is my opinion that the combination of Kirk and Oh discloses Element 12[pre].

254. *See* Elements 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. 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 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 ”**

255. In my opinion, the combination of Kirk and Oh discloses Element 12[a].

256. See Element 1[c][i]. As I have explained above, in the combination of Kirk and Oh the sequence of computations represents the layers of an artificial neural network. Oh discloses implementing a “NN” (neural network) on a GPU to classify input data. Ex1007, 1, 3. Oh’s artificial neural network includes multiple layers—an input layer, multiple hidden layers (with up to 30 nodes or “neurons” each), and an output layer. *Id.*, 3. At each layer, the artificial neural network performs calculations including matrix multiplication and a sigmoid function (*i.e.*, intermediate computations) to determine the output of the layer (*i.e.*, an intermediate result). *Id.*, 2-3. Because each node in each layer of the artificial neural network represents a neuron, a portion of the output of the first layer of the artificial neural network represents an output of a neuron.

- 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”**

257. In my opinion, the combination of Kirk and Oh discloses Element 12[a][i].

258. In the combination of Kirk and Oh, data, including both inputs and outputs, is stored in “textures.” For example, Oh discloses storing output data in a “texture,” where “[t]he result of the previous layer is saved in the form of a render

target texture, which is then used as an input for the next layer.” Ex1007, 3. This technique, known at the time of the ’438 patent as “Render to Texture,” was a well-known function through which outputs of computations are stored in a “texture” (i.e., a data array) on the GPU that can be used in subsequent computations. As I have previously noted, the ’438 patent defines a “texture” as a “data array.” Ex1001, 5:20–22 (“[T]he term ‘texture’ in this document refers to a data array . . .”).

259. In Oh, outputs from one computational element (*e.g.*, a sequence of computations representing the first layer of the artificial neural network) are 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 stored in a “second texture.” Ex1007, 2–3. Each texture is assigned an output variable (*e.g.*, a result of the computations performed by a neuron at that layer). Thus, in my opinion, the combination of Kirk and Oh discloses assigning an output variable included in a first computational element of the sequence of computations to a first texture and a second texture.

**iv. Element 12[a][ii]: “(ii) accumulating a first value for the output variable in the first texture during a first time step”**

260. In my opinion, the combination of Kirk and Oh discloses Element 12[a][ii].

261. See Element 12[a][i]. In the combined system of Kirk and Oh, the GPU executes the computations representing Oh's artificial neural network in the first layer during a first time step and stores (*i.e.*, accumulates) the result (*i.e.*, a first value for the output variable) in the first texture as is done in "Render to Texture" functionality. Ex1007, 2–3. The system then uses this first texture as the input for computations at the next layer, where the result is stored in a second texture. *Id.*

- 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"**

262. In my opinion, the combination of Kirk and Oh (and optionally Tamura) discloses Element 12[b].

263. See Element 1[d][iii]; Claim 9. In the combination of Kirk and Oh (and as additionally taught in Tamura), while the GPU is performing computations representing the artificial neural network on data received in real-time, the transfer of input data and output data over the bus occurs in parallel. As I have explained above with respect to Element 1[d][iii], in the combined Kirk and Oh (and Tamura) system, real-time input data (*i.e.*, video frames) is continually transferred to the GPU to identify text video. Thus, as a second portion of the input data (*i.e.*, additional video frames) is transferred to the GPU from main memory, the GPU continues to

perform computations on the first portion of input data (*i.e.*, prior frames). Ex1007, 2–4.

- 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”**

264. In my opinion, the combination of Kirk and Oh discloses Element 12[c].

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

**vii. Element 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”**

266. In my opinion, the combination of Kirk and Oh discloses Element 12[d].

267. *See* Elements 12[a], 12[b]. In the combination of Kirk and Oh, the GPU performs computations on the second portion of the input data (*e.g.*, the real-time input data received while the first portion of the input data is being processed by the GPU, such as 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 simulated by the GPU in the same way the computations are performed on the first portion of the input data as I have explained with respect to Element 12[a].

**viii. Element 12[d][i]: “(i) accumulating a second value for the output variable in the second texture during a second time step and”**

268. In my opinion, the combination of Kirk and Oh discloses Element 12[d][i].

269. *See* Element 12[a][i], 12[a][ii]. As I have explained above, the GPU first executes the computations for in the first layer of Oh’s artificial neural network during a first time step and stores the result (*i.e.*, a first value for the output variable)

in the first texture. Ex1007, 2–3. Then, the first texture is passed as the input for computations at second layer during a second time step, and the result of those computations (*i.e.*, “second value”) is stored in a second texture. *Id.*

**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”**

270. In my opinion, the combination of Kirk and Oh discloses Element 12[d][ii].

271. *See* Elements 1[c][i], 12[d][i]. In the Kirk and Oh combination, the first value of the output variable in the first texture is accessible to other computational elements (including the computations in the second layer of Oh’s artificial neural network) during the second time step. Ex1007, 2–4. Kirk discloses that “[a] graphics program . . . is executed within one or more Execution Pipelines 1240 as a plurality of threads where each vertex or fragment to be processed by the program is assigned to a thread.” *Id.*, 27:40–43. Results from the computations are sent to memory, such as Local Memory 140, Host Memory 112, or memory within the programmable graphics processing pipeline. *Id.*, 3:35–46. Kirk teaches that data produced by the processing pipeline “may be written to a buffer in graphics memory to be read from during a subsequent pass.” *Id.*, 4:16–20. The data arrays in Kirk in which input and output data are stored are “textures”.

272. Relatedly, Oh discloses storing output data in a “texture,” where “[t]he result of the previous layer is saved in the form of a render target texture, which is then used as an input for the next layer.” Ex1007, 3. Thus, in my opinion, in the combination of Kirk and Oh, the first value of the output variable in the first texture is accessible to other computational elements (including the computations in the second layer of Oh’s neural network) during the second time step. Ex1007, 2–4.

**m) Claim 13: “The method of claim 12, further comprising: storing the input data in the main memory in response to a user interaction.”**

273. In my opinion, the combination of Kirk and Oh disclose Claim 13.

274. *See* Element 1[b]. As I have explained, Kirk’s system includes a main memory that stores input data received by the CPU. *See* Element 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 Kirk is designed to store input data in main memory in response to a user interaction (*e.g.*, through a programmer invoking an API) or, at a minimum, it is my opinion that it would have been obvious to a POSITA to do so for the same reasons explained above). Ex1005, 6:30–33, 18:60–19:3.

n) **Claim 14**

- i. **Element 14[a]: “The method of claim 12, further comprising: receiving the input data at a first rate; and”**

275. In my opinion, the combination of Kirk and Oh discloses Element 14[a].

276. *See* Element 3[a].

- ii. **Element 14[b]: “wherein (A) comprises performing the sequence of computations at a second rate different than the first rate”**

277. In my opinion, the combination of Kirk and Oh discloses Element 14[b].

278. *See* Element 3[b].

- 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.”**

279. In my opinion, the combination of Kirk and Oh discloses Claim 16.

280. *See* Claim 7. In the Kirk and Oh combination, the second portion of the output data (*e.g.*, the result of a pass through the artificial neural network) is transferred from accelerator memory to main memory (*e.g.*, Host Memory 112) without transferring any of the intermediate results (such as the output of a first layer

of the artificial neural network). Ex1005, 5:8–11, 6:49–52, 5:60–6:3, 11:32–41, Fig. 12A; Ex1007, 3–5.

**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 memory after the GPU has begun to perform another sequence of computations.”**

281. In my opinion, the combination of Kirk and Oh discloses Claim 17.

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

**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.”**

283. In my opinion, the combination of Kirk and Oh discloses Claim 18.

284. *See* Claim 9. In the Kirk and Oh combination, where the artificial neural network on the GPU is processing video data, the transfer of output data (including “the second portion of the output data,” *see* 12[c]) occurs in parallel with the GPU

performing another sequence of computations of the artificial neural network.

Ex1007, 1, 3–4.

- r) **Claim 19: “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.”**

285. In my opinion, the combination of Kirk and Oh discloses Claim 19.

286. *See* Claim 11.

- s) **Claim 20**

- 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”**

287. In my opinion, the combination of Kirk and Oh discloses Element 20[a].

288. *See* Element 6[a]. Kirk discloses a first memory bank to store parameters common to all of the computations in the sequence of computations.

- 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”**

289. In my opinion, the combination of Kirk and Oh discloses Element 20[b].

290. *See* Element 6[b]. Kirk discloses a second memory bank to store data specific to at least one computation in the sequence of computations.

t) **Claim 21**

- i. **Element 21[pre]: “A method of performing a sequence of computations representing an artificial neural network, the method comprising:”**

291. To the extent the preamble is limiting, it is my opinion that Kirk and Oh disclose 21[pre].

292. *See* Elements 1[c][i], 12[pre] Oh discloses a sequence of combinations in an artificial neural network). Ex1007, 3–4.

- ii. **Element 21[a]: “receiving, at a central processing unit (CPU), first input data acquired from an external system in real time”**

293. In my opinion, the combination of Kirk and Oh discloses Element 21[a].

294. *See* Element 1[a]; Claim 11. As I have explained, Kirk’s CPU (*e.g.*, Host Processor 114) receives input data, and in the combination of Kirk and Oh, input data can be acquired from an external system (*e.g.*, a video camera) in real time.

- 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”**

295. In my opinion, the combination of Kirk and Oh discloses Element 21[b].

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

**iv. Element 21[c]: “transferring the first input data received by the CPU to the memory operably coupled to the GPU”**

297. In my opinion, the combination of Kirk and Oh discloses Element 21[c].

298. *See* Elements 1[b], 1[c], 1[d][iii]. Kirk discloses memory coupled to the GPU (*e.g.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250), which receives first input data (*e.g.*, data from a video camera) that was received by the CPU (*e.g.*, Host Processor 114).

- 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”**

299. In my opinion, the combination of Kirk and Oh discloses Element 21[d].

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

- vi. **Element 21[e]: “storing, in the memory operably coupled to the GPU, the first input data and the first output data”**

301. In my opinion, the combination of Kirk and Oh discloses Element 21[e].

302. *See* Elements 1[c][ii], 1[d][iii], 12[a]. Kirk discloses memory coupled to the GPU (*e.g.*, Local Memory 140 and/or memory of Programmable Graphics

Processing Pipeline 1250), which stores first input data (*e.g.*, data from a video camera) and first output data (*e.g.*, results of the computations described above).

- vii. **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”**

303. In my opinion, the combination of Kirk and Oh discloses Element 21[f].

304. *See* Elements 1[d][iii], 1[d][iv], 12[b], 12[c]; Claim 11. In the combination of Kirk and Oh, second input data (*e.g.*, additional video data) is transferred to memory in the accelerator (*e.g.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) after the GPU starts the first computation (*e.g.*, computations in a layer in a first pass of the artificial neural network) and before the GPU starts a second computation of the sequence of computations (*e.g.*, a subsequent computation in a layer of the artificial neural network, including in a second pass of the artificial neural network). In Oh, the computations (*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.

- u) **Claim 22: “The method of claim 21, further comprising: transferring the second input data comprises transferring the second input data via a bus operably coupled to the CPU.”**

305. In my opinion, the combination of Kirk and Oh discloses Claim 22.

306. *See* Elements 1[c], 12[b]. In the combination of Kirk and Oh, the second input data (*e.g.*, additional video data from a video camera) is transferred via a bus from the CPU to the accelerator.

- 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.”**

307. In my opinion, the combination of Kirk and Oh discloses Claim 23.

308. *See* Element 1[d][iv]. The Kirk and Oh combination teaches that output data (*e.g.*, the result of the computations in a layer of the artificial neural network of Oh or the result of a pass through the layers of the artificial neural network) is transferred from one memory (*e.g.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) to another memory (*e.g.*, Host Memory 112). This transfer occurs while computations continue in the artificial neural network.

w) **Claim 24**

- i. **Element 24[a]: “The method of claim 23, further comprising: storing intermediate results of the sequence of computations in the memory, and”**

309. In my opinion, the combination of Kirk and Oh discloses Element 24[a].

310. *See* Elements 1[d][iii], 12[a]; Claim 16. Kirk’s system stores intermediate results in memory, including in Local Memory 140 and Host Memory 112. Ex1005, 5:8–11, 6:49-52, 5:60–6:3. Oh also discloses storing intermediate results, such as the result of a single pass through a layer, in a texture (*i.e.*, in memory). Ex1007, 3–4.

- ii. **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”**

311. In my opinion, the combination of Kirk and Oh discloses Element 24[b].

312. *See* Claims 7, 16.

- x) **Claim 25: “The method of claim 23, wherein transferring the second input data and transferring the first output data occurs in parallel.”**

313. In my opinion, the combination of Kirk and Oh discloses Claim 25.

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

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.”**

315. In my opinion, the combination of Kirk and Oh discloses Claim 26.

316. See Element 6[a]. Element 6[a] recites a “first memory bank,” while this limitation recites a “first memory partition.” In my opinion, a POSITA would have understood the “first memory bank” described for Element 6[a]—*e.g.*, logical portions of Local Memory 140 storing the common parameters—is a “first memory partition.” Kirk teaches identifying data by an “address for a location in graphics memory.” Ex1005, 8:3–7, 9:41–55. Figures 3A and 3B, for example, illustrate storing data in buffers, such as “Buffer 320.” *Id.*, 5:1–3, 9:32–55, Figs. 3A, 3B. The dedicated memory block of Local Memory 140 in which the parameters common to all computations in a sequence of computations is stored (*i.e.*, one or more buffers) is a “first partition.”

317. Prior to the ’438 patent, it was well-known in the art that a computer program must allocate memory to store data. By allocating memory, a programmer reserves a portion of memory to store data and prevents the system from overwriting that data inadvertently. Thus, when the load module in the core interface of Kirk

loads input data into on-chip memory, it creates a partition and stores the data within that partition.

**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.”**

318. In my opinion, the combination of Kirk and Oh discloses Claim 27.

319. *See* Element 6[b]; Claim 26. Element 6[b] recites a “second memory bank,” while this limitation recites a “second memory partition.” For the same reasons I have provided above that a POSITA would have understood the “first memory bank” above to be a “first memory partition,” it is my opinion that he or she would have understood the “second memory bank” of Element 6[b] to be a “second memory partition.” *See also* Element 6[a].

320. It is my opinion that, under the interpretation of “partition” that Neural AI has put forward in the litigation, Kirk discloses “storing, in a second memory partition of the memory, data specific to the first computation in the sequence of computations” as claimed.

**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.”**

321. In my opinion, the combination of Kirk and Oh discloses Claim 28.

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

323. In the Kirk and Oh combination, Local Memory 140 can store external data input patterns, such as the input from an external camera. *Id.*, 3:32–39, 5:1–8; Ex1007, 1, 3–4. The same 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. Ex1005, 4:16–20; Ex1007, 2–3. As I have explained for Elements 1[c][ii] and 1[d][iii], GPU memory also stores inputs and outputs of the computations. Kirk is flexible in where data is and can be stored, including in buffers in Local Memory 140, and thus Kirk discloses that each of these elements are stored in a “second memory partition.” Ex1005, 3:32–46, 4:57–5:15, 5:33–36, 6:49–55, 26:24–39. And as I have explained with respect to Claim 26, Kirk teaches that data is stored in specific buffers (partitions), and the system allows logically allocating memory to hold specific data for specific processing engines, separate from other data. As such, Kirk discloses that each of these elements can be stored in a “second memory partition.”

324. To the extent not expressly disclosed in the Kirk and Oh combination, it would have been obvious to a POSITA to store each of the claimed parameters in this Claim 28 in the “second memory partition.” Kirk discloses memory in the GPU in which data and other parameters can be stored (*e.g.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250), and a POSITA would have been able to and motivated to use Local Memory 140 to store data in partitions (*e.g.*, buffers). Ex1005, 3:32–46, 4:16–20, 4:57–5:45. Because each of these pieces of data are used or output by the same thread in an artificial neural network, a POSITA would have been motivated to store them in a single memory partition, *i.e.*, the claimed second memory partition. It thus would have been obvious to a POSITA to try storing these parameters in the “second memory partition.”

**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.”**

325. In my opinion, the combination of Kirk and Oh discloses Claim 29.

326. *See* Elements 12[a][i], 12[a][ii]. In Kirk’s system, the memory in Graphics Subsystem 107 (*i.e.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) stores “graphics data,” which includes “any data that is input to or output from computation units.” Ex1005, 3:32–38, 3:50–58. In my opinion, a POSITA would have understood that the “graphics data” would include

outputs of computational elements executed by the GPU in performing the first computation in the sequence of computations. For the computations in the artificial neural network of Oh, the GPU first executes the computations in a first layer (“*i.e.*, the “first computation in the sequence of computations”) and stores the result (*i.e.*, an output of the computational elements) in memory. Ex1007, 2–4.

**cc) Claim 30**

- 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”**

327. In my opinion, the combination of Kirk and Oh discloses Element 30[a].

328. *See* Elements 12[a][i], 12[a][ii]. In the combined Kirk and Oh system, in one time step, the GPU first executes the computations in a layer of the artificial neural network and stores the result (*i.e.*, output) in the memory. Ex1007, 2–4. Thus, in a subsequent time step, an output of a previous computation is stored in memory.

*Id.*

- ii. Element 30[b]: “accessing, by the GPU, the output of the previous computation during performance of the computation in the sequence of computations”**

329. In my opinion, the combination of Kirk and Oh discloses Element 30[b].

330. See Elements 12[d][i], 12[d][ii]. In the combined Kirk and Oh system, after a first output of the GPU computations representing the artificial neural network is stored in memory, the first output is passed as an input for computations in a subsequent layer. Ex1007, 2–3. Kirk additionally teaches that output data from one computational cycle can become input data for a later computational cycle. Ex1005, 4:16–20, 8:23–25. The GPU therefore accesses the outputs of previous computations in the sequence of computations.

**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.”**

331. In my opinion, the combination of Kirk and Oh discloses Claim 31.

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

**ee) Claim 32: “The method of claim 31, wherein all neurons in the layer of neurons are described by the same equation.”**

333. In my opinion, the combination of Kirk and Oh discloses Claim 32.

334. In my opinion, a POSITA would have readily understood that, in the combination of Kirk and Oh, all neurons in a layer of the artificial neural network are described by the same equation—*e.g.*, the same matrix multiplication and sigmoid functions—and execute the same program. Ex1007, 2–3.

**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.”**

335. In my opinion, the combination of Kirk and Oh discloses Claim 33.

336. *See* Element 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.

**gg) Claim 40**

**i. Element 40[pre]: “A system for executing an artificial neural network, the system comprising:”**

337. To the extent the preamble is limiting, it is my opinion that Kirk and Oh disclose Element 40[pre].

338. *See* Elements 21[pre].

**ii. Element 40[a]: “a central processing unit (CPU) to provide first input data;”**

339. In my opinion, the combination of Kirk and Oh discloses Element 40[a].

340. *See* Element 1[a]; Claim 11.

**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”**

341. In my opinion, the combination of Kirk and Oh discloses Element 40[b].

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

343. To the extent the Kirk and Oh combination does not expressly teach referencing memory by using pointers, it would have been obvious to a POSITA. Prior to the ’438 patent, pointers were a fundamental construct of computer programming to efficiently manage memory and data access. The use of pointers are

taught, for example, in a book on the C programming language (Ex1009, 88) and NVIDIA’s book (GPU Gems) on performing general-purpose computations on GPUs (Ex1032, 499–508).

344. GPU Gems teaches storing first input data in a first partition, referenced by a first pointer, before computations begin. GPU Gems explains that “[a]rrays of data in the framework are called buffers,” and that buffers are identified by a “pointer.” Ex1032, 504. In my opinion, 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. 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. Given the widespread use of pointers in managing data, including in GPU Gems, their application to the parallel GPU

processing in the combination of Kirk and Oh would have been a routine choice for skilled practitioners in the field.

- 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”**

345. In my opinion, the combination of Kirk and Oh discloses Element 40[c].

346. *See* Elements 1[c][i], 21[d]. In the combination of Kirk and Oh, the data output by the GPU (*i.e.*, processing unit) represents an output of at least one neuron in a first layer of neurons.

- v. **Element 40[d]: “a controller, operably coupled to the processing unit and the memory, to:”**

347. In my opinion, the combination of Kirk and Oh discloses Element 40[d].

348. *See* Element 1[d].

- vi. **Element 40[d][i]: “store the first output data in a second partition of the memory, the second partition referenced by a second pointer, and to swap the first pointer with the second pointer at the end of the computation of the first layer of neurons, such that the first output data becomes an input for a second layer of neurons of the artificial neural network”**

349. In my opinion, the combination of Kirk and Oh discloses Element 40[d][i].

350. *See* Elements 12[a][i], 21[d], 40[b]. Output data from the processing unit is stored in a partition of memory (*e.g.*, a partition in Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250). As I have explained with respect to Element 40[b], a POSITA would have understood that partition of memory would be referenced by a pointer (or it would have been obvious to do so).

351. As I have explained above, in my opinion, a POSITA would have understood Oh to use “Render to Texture” functionality such that “[t]he result of the previous layer is saved in the form of a render target texture, which is then used as an input for the next layer.” Ex1007, 3. Thus, in the combination of Kirk and Oh, the first output data from a layer of the artificial neural network becomes an input for a second layer of neurons. Ex1007, 2–4; Ex1005, 4:16–20. In my opinion, 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.

352. In my opinion, it would have been obvious to a POSITA to implement an algorithm on the hardware in Kirk in which the first and second pointers are swapped at the end of a computational cycle. Because Kirk and Oh both describe using intermediate results as inputs to a new computational cycle (*see supra*), and because both disclose using pointers, it is my opinion that a POSITA would have been motivated to use pointer swapping to effectuate this efficiently (*e.g.*, by avoiding copying large chunks of memory). 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. Ex1032, 44, 508, 713 (disclosing pointer swapping in computations on GPUs).

353. 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). In my opinion, a POSITA implementing Oh’s neural network on the system of Kirk would be motivated to include this known technique, thereby increasing efficiency, and would have had a reasonable expectation of success in doing so because it had already been implemented in GPGPU computations.

**vii. Element 40[d][ii]: “transfer the first output data to another memory during computation of the second layer of neurons, and”**

354. In my opinion, the combination of Kirk and Oh discloses Element 40[d][ii].

355. *See* Elements 1[d][iii], 1[d][iv], 12[c]. In the combination of Kirk and Oh, where the artificial neural network is processing real-time data, the final output of the computations of one pass or layer of the artificial neural network is transferred to main memory 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. 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.”**

356. In my opinion, the combination of Kirk and Oh discloses Element 40[d][iii].

357. Kirk’s controller (*e.g.*, Graphics Interface 117 along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180) dictates an order of execution of instructions to Programmable Graphics Processing Pipeline 1250. Ex1005, 26:24–39. For example, “Front End 120 interprets and formats the commands and outputs the formatted commands and data to an Index Processor 1235,” which then get passed to Programmable Graphics Processing Pipeline 1250. *Id.*, 26:29–31. In my opinion, a POSITA would have understood that, by performing those actions, the front end is dictating the order of execution of the instructions.

358. Alternatively, to the extent Memory Controller 120 is considered the “controller,” it is my opinion that a POSITA would have understood the memory controller to dictate the order of execution of instructions by passing instructions to Programmable Graphics Processing Pipeline 1250 in a specific order. Ex1005, 26:35–39. In the combination of Kirk and Oh, it is my opinion that a POSITA would have understood these features to mean that the controller “dictate[s] an order of execution of instructions” for the computation on the first layer of neurons because

these components determine which instructions should be selected and issued for execution. Indeed, the instructions to the processing engines include instructions to perform computations in all layers of neurons in the artificial neural network. Ex1007, 2–4.

**hh) Claim 41: “The system of claim 40, wherein the controller is configured to send instructions for performing the at least one calculation to the processing unit.”**

359. In my opinion, the combination of Kirk and Oh discloses Claim 41.

360. *See* Element 1[c][i].

**ii) 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.”**

361. In my opinion, the combination of Kirk and Oh discloses Claim 42.

362. *See* Element 1[d][ii]. The Kirk and Oh combination teaches that the controller (*e.g.*, Graphics Interface 117, along with Front End 1230, Index Processor 1235, Raster Operation Unit 165, and Output Controller 180) sends the data and instructions to Programmable Graphics Processing Pipeline 1250 in Programmable Graphics Processor 105 for execution. Ex1005, 3:50–54, 4:21–42. Programmable Graphics Processing Pipeline 1250 (*i.e.*, a graphics processing unit) performs calculations.

**jj) Claim 43**

- i. Element 43[a]: “The system of claim 40, wherein the memory further comprises: a third partition to store internal variables; and”**

363. In my opinion, the combination of Kirk and Oh discloses Element 43[a].

364. *See* Element 6[a]; Claim 28 (describing storing specific information in “partitions” in the Kirk and Oh combination). Kirk discloses a “third partition” (*e.g.*, a partition in Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) for storing “internal variables.” Kirk discloses that the memory in Graphics Subsystem 107 (*i.e.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) stores “graphics data,” which includes “any data that is input to or output from computation units.” Ex1005, 3:32–38, 3:50–58.

365. In my opinion, a POSITA would have understood that the memory can also store representations of internal variables, such as outputs of a hidden layer in Oh’s multi-layer neural network, which are used as an input for the next layer. *Id.*, 4:16–20; Ex1007, 2–3. Indeed, it is my opinion that a POSITA would have understood the partition of Local Memory 140 storing these internal variables is a “third partition,” with the first and second partitions having been defined in Claim 40. *Id.* To the extent not disclosed in the Kirk and Oh combination, 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.

**ii. Element 43[b]: “a fourth partition to store data used as input at a particular layer of neurons of the artificial neural network”**

366. In my opinion, the combination of Kirk and Oh discloses Element 43[b].

367. *See* Element 6[b]; Claim 28 (describing storing specific information in “partitions” in the Kirk and Oh combination). Kirk discloses a “fourth partition” (*e.g.*, a partition in Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) for storing “data used as input at a particular layer of neurons of the artificial neural network.” Kirk discloses that the memory in Graphics Subsystem 107 (*i.e.*, Local Memory 140 and/or memory of Programmable Graphics Processing Pipeline 1250) stores “graphics data,” which includes “any data that is input to or output from computation units.” Ex1005, 3:32–38, 3:50–58.

368. In my opinion, a POSITA would have understood that, in the combination of Kirk and Oh, the “graphics data” would include the output textures described in Oh, which are used as inputs at a particular layer of the neurons in Oh’s neural network. Ex1007, 2–3. Additionally and alternatively, Programmable Graphics Processing Pipeline 1250 includes “Pixel Input Buffer” and “Vertex Input Buffer,” which are partitions that store data used as inputs. Ex1005, 26:62–27:11,

28:10–20, Fig. 12B. In the combination of Kirk and Oh, these buffers (any one of which would be a “fourth partition”), would store textures used as an input to a layer of neurons in the neural network. Ex1007, 2–3. Furthermore, for the same reasons it would have been obvious to store data in a “third partition” (*see* my discussion of Element 43[a]), in my opinion, it would have been obvious to a POSITA to store data used as input at a particular layer in a “fourth partition.”

**kk) Claim 44**

**i. Element 44[pre]: “A computer system, comprising:”**

369. To the extent the preamble is limiting, it is my opinion that Kirk and Oh disclose Element 44[pre].

370. *See* Element 1[pre].

**ii. Element 44[a]: “a central processing unit to receive input data acquired from an external system”**

371. In my opinion, the combination of Kirk and Oh discloses Element 44[a].

372. *See* Elements 1[a], 21[a]; Claim 11.

**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”**

373. In my opinion, the combination of Kirk and Oh discloses Element 44[b].

374. *See* Element 1[b].

- 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”**

375. In my opinion, the combination of Kirk and Oh discloses Element 44[c].

376. *See* Element 1[c].

- 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”**

377. In my opinion, the combination of Kirk and Oh discloses Element 44[c][i].

378. *See* Element 1[c][i].

- 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”**

379. In my opinion, the combination of Kirk and Oh discloses Element 44[c][ii].

380. *See* Element 1[c][ii].

- vii. **Element 44[d]: “a controller, operably coupled to the at least one processing unit and the accelerator memory”**

381. In my opinion, the combination of Kirk and Oh discloses Element 44[d].

382. *See* Element 1[d].

- 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”**

383. In my opinion, the combination of Kirk and Oh discloses Element 44[d][i].

384. *See* Element 1[d][iii].

- ix. **Element 44[d][ii]: “to control transfer at least a portion of the output data from the accelerator memory to the main memory during performance of the intermediate computations in the sequence of computations by the at least one processing unit”**

385. In my opinion, the combination of Kirk and Oh discloses Element 44[d][ii].

386. *See* Element 1[d][iv].

- x. **Element 44[d][iii]: “to control performance of the sequence of computations by the at least one processing unit”**

387. In my opinion, the combination of Kirk and Oh discloses Element 44[d][iii].

388. *See* Element 1[d][ii].

- ll) **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.”**

389. In my opinion, the combination of Kirk and Oh discloses Claim 45.

390. *See* Claim 2.

**mm) Claim 46**

- 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”**

391. In my opinion, the combination of Kirk and Oh discloses Element 46[a].

392. *See* Element 3[a]; Claim 44.

- 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”**

393. In my opinion, the combination of Kirk and Oh discloses Element 46[b].

394. *See* Element 3[b].

**nn) 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.”**

395. In my opinion, the combination of Kirk and Oh discloses Claim 47.

396. *See* Claim 4.

**oo) 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.”**

397. In my opinion, the combination of Kirk and Oh discloses Claim 48.

398. *See* Claim 5.

**pp) Claim 49**

**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”**

399. In my opinion, the combination of Kirk and Oh discloses Element 49[a].

400. *See* Element 6[a]; Claims 26, 44.

**ii. Element 49[b]: “a second memory partition to store data specific to at least one computation in the sequence of computations”**

401. In my opinion, the combination of Kirk and Oh discloses Element 49[b].

402. *See* Element 6[b]; Claim 27.

qq) **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.”

403. In my opinion, the combination of Kirk and Oh discloses Claim 50.

404. *See* Claim 7.

rr) **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.”

405. In my opinion, the combination of Kirk and Oh discloses Claim 51.

406. *See* Claim 8.

ss) **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.”

407. In my opinion, the combination of Kirk and Oh discloses Claim 52.

408. *See* Claim 9.

tt) **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.”

409. In my opinion, the combination of Kirk and Oh discloses Claim 53.

410. *See* Claim 10.

**uu) 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.”**

411. In my opinion, the combination of Kirk and Oh discloses Claim 54.

412. *See* Claim 11.

**vv) Claim 34: “The method of claim 21, further comprising: loading the second input data from disk.”**

413. In my opinion, the combination of Kirk and Oh discloses Claim 34.

414. Oh discloses using “video data” as input into its artificial neural network. Ex1007, 2–4. Where the input is not live video from a camera, the “video data” would be loaded from a disk in Kirk/Oh. During training of the artificial neural network of Oh, input video data would have been loaded from disk (e.g., a hard drive) coupled to Host Computer 110. *Id.*; Ex1005, 1:14–33, 3:12–46, 22:1–3, 26:24–39. Additionally, in my opinion, it would have been obvious to load input data from disk. Indeed, before the artificial neural network in Oh would be able to identify text in video frames, it would need to be trained using, for example, existing data. Ex1006, 624–25. It would have been obvious and within the ordinary skill of a POSITA to store the training data on a disk drive coupled to the system and then load it into the accelerator. At the time of the ’438 patent, hard drives were common storage media on which data could be stored and retrieved. At a minimum, it is my

opinion that it would have been obvious to a POSITA to load input data from disk and storing the training data on a disk drive and then loading it into the accelerator for training the artificial neural network would have been within the ordinary skill of a POSITA.

## **VIII. SECONDARY CONSIDERATIONS**

415. I am not aware of any evidence of secondary considerations of nonobviousness relied upon by Patent Owner. To the extent Patent Owner is permitted to provide any evidence of secondary considerations to rebut my opinions expressed herein, I reserve the right to respond by supplementing and/or amending this Declaration or submitting a reply declaration to consider the alleged secondary considerations of nonobviousness.

## **IX. RIGHT TO SUPPLEMENT**

416. I reserve the right to supplement this Declaration, if appropriate, should additional information come to light that is relevant to my opinions in this Declaration. I also reserve the right to consider and comment on additional evidence, (*e.g.*, objective evidence of nonobviousness), that may be presented by Patent Owner or its experts.

417. I will continue my study and evaluation of relevant prior art as these proceedings continue, at least up until the end of the period prescribed by the relevant rules and regulations. In the event further issues arise that concern the subject matter

in my Declaration, I expressly reserve the right to supplement, modify, or amend my opinions as stated in this Declaration as well as the bases supporting my opinions. I specifically reserve the right to supplement and/or amend this Declaration at such time that the Board defines the disputed terms in the Challenged Claims, particularly if the Board adopts alternate instructions not advanced by Petitioners. I further reserve the right to supplement and/or amend this Declaration to submit a reply Declaration based on any information or contentions expressed by Patent Owner or its experts attempting to rebut my opinions in this Declaration. In addition, I reserve the right to use demonstratives or other exhibits at trial in support of my opinions in any of my expert declarations and/or supplements thereto.

## **X. CONCLUSION**

418. For the reasons set forth above, it is my opinion that the Challenged Claims of the '438 patent are invalid.

419. I hereby declare that all statements made herein of my own knowledge are true, and that all statements made on information and belief are believed to be true, and that these statements were made with knowledge that willful false statements and the like are so made punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the Petition for *inter partes* review.

Executed on April 28, 2025

San Diego, California

A handwritten signature in black ink, appearing to read "Tajana", is enclosed within a thin black rectangular border.

By: \_\_\_\_\_

Dr. Tajana S. Rosing, Ph.D.

# Tajana Šimunić Rosing

tajana@ucsd.edu

<http://www.cse.ucsd.edu/~trosing/>

( 858 ) 534-4868

**INTERESTS** energy-efficient computing, computer architecture, embedded systems

## PROFESSIONAL EXPERIENCE

05-pres. **UCSD** Full Professor and Frattamico Endowed Chair in the CSE & ECE Departments; IEEE & ACM Fellow

08-pres Executive board member of San Diego Supercomputing Center

- leading a diverse research team on projects related to system energy-efficiency:
  - management and optimization of computing systems, acceleration of big data workloads
    - director of \$50M DARPA/SRC JUMP 2.0 PRISM Processing with Intelligent Storage and Memory Center
    - lead of Hyperdimensional computing work in \$40M DARPA/SRC JUMP 2.0 Cognitive Computing Center
    - hyperdimensional computing grants from DARPA, SRC, NSF; TSMC funded HDC chip tapeout
    - lead Organization & Hierarchy; and Precision Medicine in \$40M JUMP CRISP center
    - NSF AI TILOS Research Institute thrusts on chips and networks
    - head of the Large Scale Systems thrust in MuSyC center, with focus on datacenters
    - 10 year NSF ERC CIAN; lead of thrust on energy efficient datacenters; 30 companies, 12 universities
    - funded by NSF, DARPA, SRC, Oracle, Google, Microsoft, TI, Cisco, Qualcomm, CEC, Intel, Panasonic, Ericson, Broadcom, IBM and many others
  - Computing and resource management at and beyond the edge
    - director of \$16.3M AI for Healthy Living Center
    - \$28M TerraSwarm center; 22 faculty from 10 top institutions in the USA; leading the SmartCities theme
    - led context-aware distributed optimization in the Smart Grid as a part of ARPA-E NODES grant
    - NSF Early wildfire detection with drones and IoT; NSF MetaSense & CitiSense projects focused on mobile air quality sensing; coverage in NY Times and the Wall Street Journal
    - funded by DARPA, NSF, NIH, ARPA-E, DOE, LANL, CNS, Intel, IBM, TI, UTC, Raytheon, Oracle, Qualcomm, Panasonic, Broadcom, Ericson, Google, Microsoft, and others.
- published over ~420 publications; got a nomination for one of the best papers in 10 years of DATE, and TODAES journal paper was the top most downloaded paper, received many of best paper awards and nominations, and numerous invited talks in academia and industry
- led projects as a PI, co-PI or senior personnel totaling more than \$350M
- teaching embedded systems and computer engineering classes

## 97 – 04 STANFORD UNIVERSITY & HEWLETT-PACKARD LABS

- led a team of researchers developing products for wireless media market, interfaced with HP divisions; 5 patents
- obtained project funding for university collaborations and led collaboration at Stanford

## 93 – 97 ALTERA CORPORATION

- patented a testing methodology for FPLDs that enabled Altera to get to market 4 months sooner
- designed, evaluated and managed simulation and testing of 5 product families

## 92 – 93 UNIVERSITY OF ARIZONA

- design automation of high-speed VLSI interconnects; the simulator has been used by SRC member companies

## 88 – 92 NORTHERN ARIZONA UNIVERSITY

- modeled tether dynamics for orbiting stations to aid in the design of orbiting telescopes; designed an image processing environment for MRIs; designed an award-winning switched capacitor filter for TI

**EDUCATION** PhD in Electrical Engineering, Stanford University, 2001.

Thesis: *Energy Efficient System Design and Utilization*

MS in Engineering Management, Stanford University, 2000.

MS in Electrical and Computer Engineering, University of Arizona, 1993.

BS in Electrical Engineering, Northern Arizona University, 1992.

**JOURNAL PAPERS**

1. Weihong Xu, Saransh Gupta, Niema Moshiri, and Tajana Rosing, “RAPIDx: High-performance ReRAM Processing in-Memory Accelerator for Sequence Alignment,” IEEE TCAD, 2023.
2. Alireza Amirshahi, Anthony Thomas, Amir Aminifar, Tajana Rosing, David Atienza. “M2D2: Maximum-Mean-Discrepancy Decoder for Temporal Localization of Epileptic Brain Activities,” IEEE Journal of Biomedical and Health Informatics, 2022
3. X. Yu, K. Ergun, X. Song, L. Cherkasova, T. Rosing, “Automating and Optimizing Reliability-Driven Deployment in Energy-Harvesting IoT Networks,” IEEE Transactions on Network and Service Management, 2022
4. O. Gungor, T. Rosing, B. Aksanli, “STEWART: STacking Ensemble for White-Box Adversarial Attacks Towards more resilient data-driven predictive maintenance.” Computers in Industry, Elsevier 2022.
5. M Ostertag, N. Atanasov, T. Rosing, “Trajectory Planning and Optimization for Minimizing Uncertainty in Persistent Monitoring Applications,” Journal of Intelligent & Robotic Systems, 2022.
6. K. Ergun, R. Ayoub, P. Mercati, T. Rosing, “Dynamic Reliability Management of Multi-Gateway IoT Edge Computing Systems,” Special Issue of IEEE Internet of Things Journal, 2022.
7. J. Kang, B. Khaleghi, Y. Kim, T. Rosing, “OpenHD: A GPU-Powered Framework for Hyperdimensional Computing,” Special Issue on Software, Hardware, and Applications for Neuromorphic Computing in IEEE Transactions on Computing, 2022.
8. K. Ergun, R. Ayoub, P. Mercati, T. Rosing, “Reinforcement Learning Based Reliability-Aware Routing in IoT Networks”, Elsevier Ad Hoc Networks, 2022.
9. Justin Morris, Yilun Hao, Saransh Gupta, Behnam Khaleghi, Baris Aksanli and Tajana Rosing, “Stochastic-HD: Leveraging Stochastic Computing on the Hyper-Dimensional Computing Pipeline,” Special Issue of Frontiers in Neuroscience, section on Neuromorphic Engineering, 2022.
10. Jones, Derek; Allen, Jonathan; Yang, Yue; Bennett, W.F.; Gokhale, Maya; Moshiri, Niema; Rosing, Tajana, “Accelerators for Classical Molecular Dynamics Simulations of Biomolecules,” Journal of Chemical Theory and Computation, 2022.
11. George Armstrong , Cameron Martino , Justin Morris , Behnam Khaleghi , Jaeyoung Kang , Jeff Dereus , Mr. Qiyun Zhu , Daniel Roush , Daniel McDonald , Dr. Antonio Gonzalez , Dr. Justin P Shaffer , Carolina Carpenter , Dr. Mehrbod Estaki , Dr. Stephen Wandro , Sean Eilert , Ameen Akel , Justin Eno , Ken Curewitz , Austin D Swafford , Niema Moshiri , Tajana Rosing , Rob Knight, “ Swapping metagenomics preprocessing pipeline components offers speed and sensitivity increases,” American Society for Microbiology mSystems Journal, 2022. 6.496 impact factor
12. J. Morris, K. Ergun, B. Khaleghi, M. Imani, B. Aksanli, T. Rosing, “HyDREA: Utilizing Hyperdimensional Computing For A More Robust and Efficient Machine Learning System,” Special issue of ACM TECS’22.
13. S. Gupta, B. Khaleghi, S. Salamat, J. Morris, R. Ramkumar, J. Yu, A. Tiwari, M. Imani, B. Aksanli, T. Rosing, “Store-n-Learn: Classification and Clustering with Hyperdimensional Computing across Flash Hierarchy,” Special issue of ACM TECS, 2022.
14. O. Gungor, T. Rosing, B. Aksanli, "RESPIRE++: Robust Indoor Sensor Placement Optimization under Distance Uncertainty”, **Invited paper to IEEE Sensors Journal, 2022.**
15. A. Khachiyani, A. Thomas, H. Zhou, G. Hanson, A. Cloninger, T. Rosing, A. Khandelwal, “Using Neural Networks to Predict Micro-Spatial Economic Growth,” American Economic Review Insights, 2022.
16. A. Thomas, S. Dasgupta, T. Rosing, “A Theoretical Perspective on Hyperdimensional Computing,” Journal of Artificial Intelligence Research, 2021.
17. S. Gupta, M. Imani, J. Sim, A. Huang, F. Wu, Y. Kim, T. Rosing, “COSMO: Computing with Stochastic Numbers in Memory,” JETC 2021.
18. O. Gungor, T. Rosing, B. Aksanli, "DOWELL: Diversity-induced Optimally Weighted Ensemble Learner for Predictive Maintenance of Industrial Internet of Things Devices", IEEE IoT Journal 2021.
19. S. Salamat, H. Zhang, YS Ki, T. Rosing, “NASCENT2: Generic Near-storage Sort Accelerator for Data Analytics on SmartSSD,” IEEE Transactions on Reconfigurable Technology and Systems, 2021.
20. J. Morris, Y. Hao, M. Imani, B. Aksanli, T. Rosing, “Locality-based Encoder and Model Quantization for Efficient Hyper-Dimensional Computing,” IEEE TCAD 2020.
21. X. Yu, K. Ergun, L. Cherkasova, T. Rosing, “Optimizing Sensor Deployment and Maintenance Costs for Large-Scale Environmental Monitoring,” IEEE TCAD, 2020.
22. Sahand Salamat, Mohsen Imani, and Tajana Rosing, “Accelerating hyperdimensional computing on FPGAs by exploiting computational reuse,” IEEE Transactions on Computing, 2020.

23. M. Imani, S. Bosch, S. Datta, S. Ramakrishna, S. Salamat, J. Rabaey, T. Rosing, "QuantHD: A Quantization Framework for Hyperdimensional Computing", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2019.
24. M. Imani, S. Gupta, S. Sarama, T. Rosing, "NVQuery: Efficient Query Processing in Non-Volatile Memory," IEEE TCAD, 2019.
25. M. Imani, X. Yin, J. Messerly, S. Gupta, M. Nemier, X. S. Hu, T. Rosing, "SearchHD: A Memory-Centric Hyperdimensional Computing with Stochastic Training", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2019.
26. D. Peroni, M. Imani, H. Nejatollahi, N. Dutt, T. Rosing, "Data Reuse for Accelerated Approximate Warps", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2019.
27. Yeseong Kim, Mohsen Imani, and Tajana S. Rosing, "Image Recognition Accelerator Design Using In-Memory Processing," IEEE MICRO, 2019.
28. M. Imani, R. Garcia, S. Gupta, T. Rosing, "Hardware-Software Co-design to Accelerate Neural Network Applications", ACM Journal on Emerging Technologies in Computing (JETC), 2019.
29. M. Imani, J. Morris, H. Shu, S. Li, T. Rosing, "Efficient Associative Search in Brain-Inspired Hyperdimensional Computing", IEEE Design & Test (D&T), 2019.
30. S. Vikram, A. Collier-Oxandale, M. Osterhout, M. Menarini, C. Chermak, S. Dasgupta, T. S. Rosing, M. Hannigan, W. Griswald, T. S. Rosing, "Evaluating and Improving the Reliability of Gas-Phase Sensor System Calibrations Across Locations for Ambient Measurements and Personal Exposure Monitoring," Atmospheric Measurement Techniques Journal, 2019.
31. S. Gupta, M. Imani, T. S. Rosing, "NNPIM: A Processing In-Memory Architecture for Neural Network Acceleration," IEEE Transaction on Computers, 2019.
32. D. Peroni, M. Imani, T. Rosing, "Runtime Efficiency-Accuracy Trade-off Using Configurable Floating Point Multiplier", IEEE TCAD, 2018.
33. M. Imani, S. Gupta, S. Sharma, T. Rosing, "NVQuery: Efficient Query Processing in Non-Volatile Memory", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2018.
34. Christine Chan, Alper Sinan Akyurek, Baris Aksanli, and Tajana S. Rosing. "Optimal Performance-Aware Cooling on Enterprise Servers," IEEE TCAD'18.
35. A. Sinan Akyurek and Tajana S. Rosing, "Optimal Packet Aggregation in Wireless Networks", IEEE Transactions on Mobile Computing, 2018.
36. M. Imani, Shruti Patil, T. Rosing, "Approximate Computing using Multiple-Access Single-Charge Associative Memory" IEEE Transaction on Emerging Topics in Computing (TETC), 2017.
37. M. Imani, D. Peroni, T. Rosing "NVALT: Approximate Lookup Table for GPU Acceleration", IEEE Embedded System Letter (ESL), 2017.
38. M. Imani, A. Rahimi, Hwang, T. S. Rosing, J.M. Rabaey, "Low-Power Sparse Hyperdimensional Encoder for Language Recognition," IEEE Design & Test, 2017.
39. M. Imani, A. Rahimi, P. Mercati, T.S. Rosing, "Multi-stage Tunable Approximate Search in Resistive Associative Memory", IEEE Transactions on Multi-Scale Computing Systems (TMSCS), 2017.
40. M. Imani, D. Peroni, A. Rahimi, T.S. Rosing, "Resistive CAM Acceleration for Tunable Approximate Computing", IEEE Transactions on Emerging Topics in Computing (TETC), 2017.
41. B. Aksanli, J. Venkatesh, C. Chan, A. S. Akyurek, T. S. Rosing, "Modular and Personalized Smart Health Application Design in a Smart City Environment," IEEE Internet of Things Journal, 2017.
42. B. Aksanli, T. S. Rosing, "User Behavior and Flexibility based Energy Management in Residential Neighborhoods," Special Issue of IEEE Transactions on Emerging Topics in Computing, 2017.
43. Jinseok Yang, A. Sinan Akyurek, Sameer Tilak and Tajana S. Rosing, "Design of transmission manager in heterogeneous WSNs", IEEE Transactions on Emerging Topics in Computing 2017.
44. A. Sinan Akyurek and Tajana Simunic Rosing, "Optimal Distributed Nonlinear Battery Control", IEEE Journal of Emerging and Selected Topics in Power Electronics, 2017.
45. Jagannathan Venkatesh, Baris Aksanli, Christine Chan, Alper S. Akyurek, Tajana S. Rosing, "Scalable Application Design for the Internet of Things", Special Issue of IEEE Software, 2017.
46. P. Mercati, F. Paterna, A. Bartolini, L. Benini, T. Simunic Rosing, "WARM: Workload-Aware Reliability Management in Linux/Android," IEEE TCAD 2016.

47. M. Imani, Shruti Patil, T. Rosing, "Approximate Computing using Multiple-Access Single-Charge Associative Memory" IEEE Transaction on Emerging Topics in Computing (IEEE TETC), 2016.
48. Mohsen Imani, Shruti Patil, Tajana S. Rosing, "Ultra-low power FinFET based SRAM cell employing sharing current concept", Microelectronic Reliability Elsevier Journal, 2015
49. Lucas Wanner, Liangzhen Lai, Abbas Rahimi, Mark Gottscho, Pietro Mercati, Chu-Hsiang Huang, Frederic Sala, Yuvraj Agarwal, Lara Dolecek, Nikil Dutt, Puneet Gupta, Rajesh Gupta, Ranjit Jhala, Rakesh Kumar, Sorin Lerner, Subhasish Mitra, Alexandru Nicolau, Tajana Simunic Rosing, Mani B Srivastava, Steve Swanson, Dennis Sylvester, Yuanyuan Zhou. "NSF Expedition of Variability-Aware Software:Recent Results and Contributions", Information Technology, Special Issue on Dependable Embedded Systems, 57(3), 2015, pp. 181-198., 2015.
50. E.A. Lee, J Rabaey, D. Blaauw, P. Dutta, K. Fu, C. Guestrin, B. Hartmann, R. Jafari, D. Jones, J. Kubiawicz, V. Kumar, R. Mangharam, R. Murray, G. Pappas, K. Pister, A. Rowe, A. Sangiovanni-Vincentelli, S.A. Seshia, T. Simunic Rosing, B. Taskar, J. Wawrzynnek, D. Wessel, "The Swarm at the Edge of the Cloud", IEEE Design & Test, 2014.
51. R. Ayoub, R. Nath, T. Simunic Rosing, "CoMETC: Coordinated Management of Energy, Thermal, & Cooling in Servers" TODAES'13.
52. A. Kahng, S. Kang, T. Simunic Rosing, R. Strong, "Many-Core Token-Based Adaptive Power Gating," IEEE TCAD'13.
53. P. Aghera, J. Yang, P. Zappi, D. Krishnaswamy, A. Coskun, and T. Simunic Rosing," Energy Management in Wireless Mobile Systems Using Dynamic Task Assignment," JOLPE'13.
54. S. Sharifi, T. Simunic Rosing, "PROMETHEUS: A Proactive Method for Thermal Management of Heterogeneous MPSoCs," IEEE TCAD'13.
55. P. Gupta, Y. Agarwal, L. Dolecek, N. Dutt, R. K. Gupta, R. Kumar, S. Mitra, A. Nicolau, T. Simunic Rosing, M. B. Srivastava, S. Swanson, D. Sylvester: "Underdesigned and Opportunistic Computing in Presence of Hardware Variability," IEEE TCAD'13.
56. B. Aksanli, J. Venkatesh, T. Simunic Rosing, "Datacenter Modeling and Simulation with Focus on Energy Efficiency and Green Energy Integration," IEEE Computer Special Issue on Modeling and Simulation of Smart and Green Computing Systems, 2012.
57. Mohamed M. Sabry, Ayse K. Coskun, David Atienza, Tajana Simunic Rosing, Thomas Brunschwiler, "Energy-Efficient Multi-Objective Thermal Control for Liquid-Cooled 3D Stacked Architectures," IEEE TCAD, 2011.
58. R. Ayoub, K. Indukuri, T. Simunic Rosing, "Temperature Aware Dynamic Workload Scheduling in Multisocket CPU Servers," IEEE TCAD, 2011.
59. E. Regini, D. Lim, T. Simunic Rosing, "Energy management in heterogeneous wireless sensor networks," JOLPE, 2011.
60. G. Dhiman, G. Marchetti, T. Simunic Rosing, "vGreen: A System for Energy Efficient Management of Virtualized Environments," Special Issue of ACM TODAES, 2010. **Top most downloaded paper '10-'11**
61. J. Recas, C. B, T. Simunic Rosing, D. Atienza, "HOLLOWS: A Power-aware Task Scheduler for Energy Harvesting Sensor Nodes", Journal of Intelligent Material Systems and Structures, 2010.
62. S. Sharifi, T. Simunic Rosing, "Accurate direct and indirect on-chip temperature sensing for efficient dynamic thermal management," IEEE TCAD, 2010.
63. A. Coskun, T. Simunic Rosing, "Utilizing Predictors for Efficient Thermal Management in Multiprocessor SoCs," IEEE TCAD, 2009.
64. G. Dhiman, T. Simunic Rosing, "Using online learning for system level power management," IEEE TCAD, 2009.
65. A. Coskun, T. Simunic Rosing, K. Whisnant, K. Gross, "Static and dynamic temperature-aware scheduling for multiprocessor SoCs," IEEE TVLSI, 2008.
66. T. Simunic Rosing, K. Mihic, G. De Micheli, "Power and reliability management of SOCs," IEEE Transactions on VLSI, 2007.
67. G. Park, T. Simunic Rosing, M. Todd, C. Farrar, W. Hodgkiss, "Energy Harvesting for Structural Health Monitoring in Sensor Networks," ASCE Journal, 2007.
68. A. Coskun, T. Simunic Rosing, K. Mihic, G. De Micheli, Y. Leblebici, "Analysis and Optimization of MPSoC Reliability," Invited paper to Journal of Low-Power Electronics, April 2006.
69. T. Simunic, S. Boyd, P. Glynn: "Managing Power Consumption in Networks on Chips," IEEE Transactions on VLSI, pp. 96- 107, Jan 2004.

70. A. Acquaviva, T. Simunic, V. Deolalikar, S. Roy: "Remote Power Control of Wireless Network Interfaces", Special Issue of Journal of Embedded Computing, No. 3, 2004.
71. B. Delaney, T. Simunic, N. Jayant: "Power Aware Distributed Speech Recognition for Wireless Mobile Devices," Special Issue on Embedded Systems for Multimedia, IEEE Design & Test, 2004.
72. A. Peymandoust, T. Simunic, G. De Micheli: "Complex Instruction and Software Library Mapping for Embedded Software Using Symbolic Algebra," Special Issue of IEEE Transactions on CAD, pp.964-975, August 2003.
73. T. Simunic, L. Benini, P. Glynn, G. De Micheli: "Event-Driven Power Management", IEEE Transactions on CAD, pp.840-857, July 2001.
74. T. Simunic, M. Smith: "Dynamic Power Management at HP", Invited Paper in Special Issue of Design and Test Journal, 2001.
75. T. Simunic, L. Benini, G. De Micheli: "Energy-Efficient Design of Battery-Powered Embedded Systems", Special Issue of IEEE Transactions on VLSI, 2001.
76. T. Simunic, J. Rozenblit, J. Brews: "VLSI Interconnect Design Automation Using Qualitative and Symbolic Techniques"; IEEE Transactions on Components, Packaging, and Manufacturing Technology Part B: Advanced Packaging, 1996.

#### CONFERENCE PAPERS

1. W. Xu, J. Kang and T. Rosing, "FSL-HD: Accelerating Few-Shot Learning on ReRAM using Hyperdimensional Computing," DATE 2023
2. M. Zhou, X. Wang, T. Rosing, "OverlaPIM: Overlap Optimization for Processing In-Memory Neural Network Acceleration," DATE 2023.
3. B. Khaleghi, T. Zhang, C. Martino, G. Armstrong, A. Akel, K. Curewitz, J. Eno, S. Eilert, R. Knight, N. Moshiri, T. Rosing, "SALIENT: Ultra-Fast FPGA-based Short Read Alignment", **best paper nomination**, FPT 2022.
4. J. Kang, W. Xu, W. Bittremieux, T. Rosing, "Massively Parallel Open Modification Spectral Library Searching with HD Computing," PACT'22.
5. J. Kang, M. Zhou, A. Bhansali, W. Xu, A. Thomas and T. Rosing, "RelHD: A Lightweight Graph-based Learning with Hyperdimensional Computing", ICCD 2022.
6. U. Mallappa, P. Gangwar, B. Khaleghi, and T. Rosing, "TermiNETor: Early Convolution Termination for Efficient Deep Neural Networks", ICCD 2022
7. B. Khaleghi, T. Zhang, N. Shao, A. Akel, K. Curewitz, J. Eno, S. Eilert, N. Moshiri, T. Rosing, "FAST: FPGA-based Acceleration of Genomic Sequence Trimming", IEEE BioCAS, 2022.
8. O. Gungor, T. Rosing, B. Aksanli, "DENSE-DEFENSE: Diversity Promoting Ensemble Adversarial Training Towards Effective Defense", IEEE Sensors 2022.
9. J. Liu, X. Yu, T. Rosing, "Self-Train: Self-Supervised On-Device Training for Post-Deployment Adaptation", IEEE SmartIoT, 2022
10. Q. Zhao, K. Lee, J. Liu, M. Huzaifa, X. Yu, T. Rosing, "FedHD - Federated Learning with Hyperdimensional Computing Demo", Mobicom, 2022
11. E. Ekaireb, X. Yu, K. Ergun, Q. Zhao, K. Lee, M. Huzaifa, T. Rosing, "ns3-fl: Simulating Federated Learning with ns-3", Workshop on ns-3 (WNS3), 2022
12. Arpan Dutta, Saransh Gupta, Behnam Khaleghi, Rishikanth Chandrasekaran, Weihong Xu, Tajana Rosing, "HDnn PIM: Efficient in Memory Design of Hyperdimensional Computing with Feature Extraction," GLVLSI'22.
13. A. Thomas, S. Dasgupta, T. Rosing, "A Theoretical Perspective on Hyperdimensional Computing," **Invited paper to a special session of the International Joint Conference on Artificial Intelligence, IJCAI 2022**
14. Weihong Xu, Jaeyoung Kang and Tajana Rosing, "A Near-Storage Framework for Boosted Data Preprocessing of Mass Spectrum Clustering," DAC, 2022
15. Behnam Khaleghi, Uday Mallappa, Duygu Nur Yaldiz, Haichao Yang, Monil Shah, Jaeyoung Kang and Tajana Rosing, "PatterNet: Explore and Exploit Filter Patterns for Efficient Deep Neural Networks," DAC'22
16. Rishikanth Chandrasekaran, Kazim Ergun, Jihyun (Lucy) Lee, Dhanush Nanjunda, Jaeyoung Kang and Tajana Rosing, "FHDnn: Communication Efficient and Robust Federated Learning for AIoT Networks," DAC'22.
17. Behnam Khaleghi, Jaeyoung Kang, Hanyang Xu, Justin Morris and Tajana Rosing, "GENERIC: Highly Efficient Learning Engine on Edge using Hyperdimensional Computing," DAC'22.
18. Minxuan Zhou, Weihong Xu, Jaeyoung Kang, and Tajana Rosing, "TransPIM: A Memory-based Acceleration via Software-Hardware Co-Design for Transformers", HPCA'22.

19. Yizhou Wei, Minxuan Zhou, Sihang Liu, Korakit Seemakhupt, Tajana Rosing and Samira Khan. “PIMProf: An Automated Program Profiler for Processing-in-Memory Offloading Decisions”, DATE’22.
20. Yang Ni, Yeseong Kim, Tajana Rosing and Mohsen Imani, “Online Performance and Power Prediction for Edge TPU via Comprehensive Characterization,” DATE’22.
21. Yang Ni, Yeseong Kim, Tajana Rosing and Mohsen Imani, “Algorithm-Hardware Co-Design for Efficient Brain-Inspired Hyperdimensional Learning on Edge,” **Best Paper Award** at DATE’22.
22. Justin Morris, Hin Wai Lui, Kenneth Stewart, Behnam Khaleghi, Anthony Thomas, Thiago Marback, Baris Aksanli, Emre Nefteci, and Tajana Rosing, “HyperSpike: HyperDimensional Computing for More Efficient and Robust Spiking Neural Networks, DATE’22.
23. Michael Ostertag, Jason Ma, Tajana S. Rosing, “Remote Sensing with UAV and mobile recharging vehicle rendezvous,” DATE’22.
24. S. Xia, R. Chandrasekaran, Y. Liu, C. Yang, T. Rosing, X. Jiang, “Drone-based System for Intelligent and Autonomous Homes,” ACM SenSys 2021; **Best demo award**
25. J. Kang, B. Khaleghi, Y. Kim, T. Rosing, “XCelHD: Efficient GPU-Powered Hyperdimensional Computing with Parallelized Training,” ASPDAC’22.
26. X. Liu, M. Zhou, R. Ausavarungrun, S. Eilert, A. Akel, T. Rosing, V. Narayanan, J. Zhao, “Mirage: A Highly Parallel and Flexible RRAM Accelerator,” HPCA’21
27. S. Gupta, R. Camarota, T. Rosing, “Accelerating Fully Homomorphic Encryption with Processing in Memory,” **Invited paper to special session at DAC’21.**
28. M. Zhou, Y. Guo, W. Xu, B. Li, K. Eliceiri, T. Rosing, “MAT: Processing In-Memory Acceleration for Long-Sequence Attention,” DAC’21.
29. X. Yu, W. Xu, L. Cherkasova, T. Rosing, “Automating Reliable and Fault-Tolerant Design of LoRa-based IoT Networks,” CNSM’21. **Best paper award**
30. Yeseong Kim, M. Imani, S. Gupta, M. Zhou, T. Rosing, “Massively Parallel Big Data Classification on a Programmable Processing In-Memory Architecture,” ICCAD 21.
31. A. Paul, B. Khaleghi, G. Hota, Y. Xu, T. Rosing, Gert Cauwenberghs, “Attention State Classification with In-Ear EEG,” BioCAS’21.
32. S. Salamat, N. Moshiri, T. Rosing, “FPGA Acceleration of Pairwise Distance Calculation for Viral Transmission Clustering” BIOCAS’21.
33. O. Gungor, B. Aksanli, T. Rosing, “CAHEROS: Constraint-Aware Heuristic Approach for Robust Sensor Placement” IEEE SENSORS’21.
34. O. Gungor, B. Aksanli, T. Rosing, “ENFES: Ensemble FEw-Shot Learning for Intelligent Fault Diagnosis with Limited Data,” IEEE SENSORS’21.
35. Justin Morris, Si Thu Kaung Set, Gadi Rosen, Mohsen Imani, Baris Aksanli and Tajana Rosing, “AdaptBit-HD: Adaptive Model Bitwidth for Hyperdimensional Computing,” ICCD’21
36. Yilun Hao, Saransh Gupta, Justin Morris, Behnam Khaleghi, Baris Aksanli and Tajana Rosing: “Stochastic-HD: Leveraging Stochastic Computing on Hyper-Dimensional Computing,” ICCD’21.
37. A Sokolova, M Imani, A Huang, R Garcia, J Morris, T Rosing, B Aksanli, “MACcelerator: Approximate Arithmetic Unit for Computational Acceleration,” ISQED’21.
38. M. Zhou, L. Wu, M. Li, N. Moshiri, K. Skadron, T. Rosing, “Ultra Efficient Acceleration for De Novo Genome Assembly via Near-Memory Computing,” PACT’21
39. M. Zhou, G. Chen, M. Imani, S. Gupta, W. Zhang, T. Rosing, “PIM-DL: Boosting DNN Inference on Digital Processing In-Memory Architectures via Data Layout Optimizations,” PACT’21.
40. S. Gupta, R. Camarota, T. Rosing, “Priv-PIM: Privacy-Preserved Processing in-Memory,” SPSL’21.
41. Xiao Liu, Minxuan Zhou, Rachata Ausavarungrun, Sean Eilert, Ameen Akel, Tajana Rosing, Vijaykrishnan Narayanan, Jishen Zhao, “FPRA: A Fine-grained Parallel RRAM Architecture,” ISLPED’21.
42. K. Ergun, R. Ayoub, P. Mercati, T. Rosing, “Improving Mean Time to Failure of IoT Networks with Reliability-Aware Routing,” CPSIoT 2021.
43. J. Ma, M. Ostertag, D. Bharadia, T. Rosing, “Frequency-aware Trajectory and Power Control for Multi-UAV Systems,” INFOCOM-DroneCom’21.
44. Khaleghi B, Akel A, Curewitz K, Eno J, Eilert S, Moshiri N, Rosing T., “FPGA-based acceleration of primer trimming,” 28th International Dynamics & Evolution of Human Viruses Conference 2021.

45. Kang J, Young C, Morris J, Akel A, Eilert S, Eno J, Curewitz K, Moshiri N, Rosing T., "GPU-Powered Phylogenetic Analysis for Large-scale Genomic Sequences," Int'l Dynamics & Evolution of Human Viruses Conf, 2021. Poster.
46. S. Salamat, N. Moshiri, T. Rosing, "FPGA-based acceleration of pairwise distance calculation for viral transmission clustering," International Dynamics & Evolution of Human Viruses Conf., 2021. Poster
47. O. Gungor, T. Rosing, B. Aksanli, "OPELRUL: Optimally Weighted Ensemble Learner for Remaining Useful Life Prediction of Industrial Machinery", IEEE International Conference on Prognostics and Health Management, 2021.
48. Mohsen Imani\*, Zhuowen Zou, Samuel Bosch, Sanjay Anantha Rao, Sahand Salamat, Venkatesh Kumar, Yeseong Kim\*, and Tajana Rosing, "Revisiting HyperDimensional Learning for FPGA and Low-Power Architectures," HPCA'21.
49. Namiko Matsumoto, Anthony Thomas, Tara Javidi and Tajana Rosing, "Hyperdimensional Computing and Spectral Learning," CogArch'21.
50. Sahand Salamat, Behnam Khaleghi, Armin Haj Aboutalebi, Joo Hwan Lee, Yang Seok Ki, Tajana Rosing, "NASCENT: Near-Storage Acceleration of Database Sort on SmartSSD," FPGA 2021.
51. M. Zhou, M. Li, M. Imani, T. Rosing, "HyGraph: Accelerating Graph Processing with Hybrid Memory-centric Computing," DATE'21.
52. Sahand Salamat, Jaeyoung Kang, Yeseong Kim, Mohsen Imani, Niema Moshiri and Tajana Rosing, "FPGA Acceleration of Protein Back-translation and Alignment," DATE'21.
53. Behnam K., Hanyang Xu, Justin Morris, Tajana S. Rosing, "tiny-HD: Ultra-Efficient Hyperdimensional Computing Engine for IoT Applications," DATE'21.
54. R. Garcia, F. Asgarinejad, B. Khaleghi, T. Rosing, M. Imani, "TruLook: A Framework for Configurable GPU Approximation", DATE, 2021.
55. Justin Morris, Kazim Ergun, Behnam Khaleghi, Mohsen Imani, Baris Aksanli, Tajana Rosing, "HyDREA: Towards More Robust and Efficient Machine Learning Systems with Hyperdimensional Computing," DATE'21.
56. M. Zhou, S. Gupta, M. Imani, Y. Kim, T. Rosing, "DP-Sim: A Full-stack Simulation Infrastructure for Digital Processing In-Memory Architecture," ASPDAC 2021.
57. S. Salamat, S. Shubhi, B. Khaleghi, T. Rosing, "Residue-Net: Multiplication-free Neural Network by In-situ, No-loss Migration to Residue Number Systems", ASP-DAC, 2021.
58. Y. Guo, S. Gupta, J. Kang, M. Imani, Y. Kim, J. Morris, T. Rosing, "HyperRec: Efficient Recommender Systems with Hyperdimensional Computing," ASPDAC 2021.
59. K. Ergun, R. Ayoub, P. Mercati, D. Liu, T. Rosing, "Energy and QoS-Aware Dynamic Reliability Management of IoT Edge Computing Systems," ASPDAC 2021.
60. Saransh Gupta, Mohsen Imani, Behnam Khaleghi, Niema Moshiri, and Tajana S. Rosing, "RAPIDx: A ReRAM Processing in-Memory Architecture for DNA Short Read Alignment," American Society of Human Genetics (ASHG), 2020. Poster
61. Salamat S, Kang J, Kim Y, Imani M, Moshiri N, Rosing T., "FPGA Acceleration of Protein Back-Translation and Alignment," American Society of Human Genetics (ASHG) 2020. Poster.
62. F. Asgarinejad, A. Thomas, T. Rosing. Detection of Epileptic Seizures from Surface EEG using Hyperdimensional Computing, EMBC, 2020.
63. B. Khaleghi, S. Salamat, T. Rosing, "Revisiting FPGA Routing under Varying Operating Conditions", FPT, 2020.
64. Y. Guo, X. Yu, K. Chaudhuri, T. Rosing, "Efficient Distributed Training in Heterogeneous Mobile Networks with Active Sampling," MSN'20.
65. X. Yu, X. Song, L. Cherkasova, T. Rosing, "Reliability-Driven Deployment in Energy-Harvesting Sensor Networks," CNSM'20.
66. K. Ergun, X. Yu, N. Nagesh, L. Cherkasova, P. Mercati, R. Ayoub, T. Rosing, "Simulating Reliability of IoT Networks with RelIoT", IEEE 50th Conference on Dependable Systems and Networks (DSN), 2020.
67. K. Ergun, X. Yu, N. Nagesh, L. Cherkasova, P. Mercati, R. Ayoub, T. Rosing, "RelIoT: Reliability Simulator for IoT Networks", International Conference on Internet of Things (ICIOT), 2020.
68. M. Imani, S. Pampana, S. Gupta, M. Zhou, Y. Kim, T. Rosing, "DUAL: Acceleration of Clustering Algorithms using Digital-based Processing In-Memory," MICRO 2020.
69. J. Morris, Y. Hao, S. Gupta, R. Ramkumar, J. Yu, M. Imani, B. Aksanli, T. Rosing, "Multi-label HD Classification in 3D Flash," VLSI-SOC 2020, **Invited paper**.

70. Y. Guo, M. Liu, T. Yang, T. S. Rosing, “Improved Schemes for Episodic Memory based Lifelong Learning Algorithm,” NeurIPS 2020, **Spotlight presentation**.
71. O. Gungor, T. Rosing, B. Aksanli, “RESPIRE: Robust SEnSor Placement OptImization in PRobabilistic Environments,” IEEE Sensors 2020. **Best paper nomination**
72. O. Gungor, J. Garnier, T. Rosing, B. Aksanli, “LENARD: Lightweight ENsemble LeARner for MeDium-term Electricity Consumption Prediction,” IEEE SmartGridComm, 2020.
73. Y. Guo, X. Yu, K. Choudhuri, T. Rosing, “Efficient Distributed Training in Heterogeneous Mobile Networks with Active Sampling,” IEEE Mobility, Sensing and Networking 2020.
74. Saransh Gupta, Mohsen Imani, Hengyu Zhao, Fan Wu, Jishen Zhao, and Tajana Rosing, “Implementing Binary Neural Networks in Memory with Approximate Accumulation,” ISLPED 2020.
75. B. Khaleghi, S. Salamat, A. Thomas, F. Asgarinejad, Y. Kim, T. Rosing, "SHEARer: Highly-Efficient Hyperdimensional Computing by Software-Hardware Enabled Multifold AppRoximation", ISLPED, 2020.
76. Saransh Gupta, Justin Morris, Mohsen Imani, Ranganathan Ramkumar, Jeffrey Yu, Aniket Tiwari, Baris Aksanli, Tajana Rosing , "THRIFTY: Training with Hyperdimensional Computing across Flash Hierarchy," ICCAD 2020.
77. M. Ostertag, S. Al-Doweeshi, T. Rosing, “Efficient Training on Edge Devices Using Online Quantization,” DATE 2020.
78. Yunhui Guo, Yandong Li, Liqiang Wang, Tajana Rosing, "AdaFilter: Adaptive Filter Fine-tuning for Deep Transfer Learning," AAAI 2020.
79. Yunhui Guo, Noel C. Codella, Leonid Karlinsky, James V. Codella, John R. Smith, Kate Saenko, Tajana Rosing, Rogerio Feris , "A Broader Study of Cross-Domain Few-Shot Learning," ECCV 2020.
80. M. Imani, M. Samragh, Y. Kim, S. Gupta, F. Koushanfar, T. Rosing “Deep Learning Acceleration with Neuron-to-Memory Transformation”, HPCA, 2020.
81. Mohsen Imani\*, Saikishan Pampana, Saransh Gupta, Minxuan Zhou, Yeseong Kim\*, and Tajana Rosing, "DUAL: Acceleration of Clustering Algorithms using Digital-based Processing In-Memory," HPCA'20.
82. B. Khaleghi, M. Imani, T. Rosing “Prive-HD: Privacy-Preserved Hyperdimensional Computing”, DAC, 2020.
83. H. Nejatollahi, S. Gupta, M. Imani, R. Cammarota\*, T. Rosing, N. Dutt “CryptoPIM: In-Memory Acceleration for RLWE Lattice-based Cryptography”, **best paper award**, DAC, 2020.
84. S. Gupta, M. Imani, J. Sim, A. Huang, F. Wu, H. Najafi, T. Rosing, “SCRIMP: A General Stochastic Computing Architecture using ReRAM in-Memory Processing”, DATE, 2020.
85. Y. Kim, M. Imani, N. Moshiri\*, T. Rosing, “GenieHD: Efficient DNA Pattern Matching Accelerator Using Hyperdimensional Computing”, **best paper nomination**, DATE, 2020.
86. Behnam Khaleghi and Tajana S. Rosing, "Thermal-Aware Design and Flow for FPGA Performance Improvement", IEEE/ACM Design Automation and Test in Europe Conference (DATE), 2019
87. B. Khaleghi, S. Salamat, M. Imani, T. Rosing, "FPGA Energy Efficiency by Leveraging Thermal Margin", ICCD, 2019.
88. K. Ergun, R. Ayoub, P. Mercati, T. Rosing, “Dynamic Optimization of Battery Health in IoT Networks”, ICCD’19.
89. R. Chandrasekaran, Y. Guo, A. Thomas, M. Menarini, M. Ostertag, Y. Kim, T. S. Rosing, “Efficient Sparse Processing in Smart Home Applications,” ACM SenSys ML, 2019.
90. M. Imani, J. Morris, S. Bosch, H. Shu, G. De Micheli, T. S. Rosing, “AdaptHD: Adaptive Efficient Training for Brain-Inspired Hyperdimensional Computing,” BioCAS, 2019.
91. M. Imani, S. Gupta, T. Rosing “Digital-based Processing In-Memory: A Highly-Parallel Accelerator for Data Intensive Applications”, ACM International Symposium on Memory Systems (MEMSYS), 2019.
92. Mohsen Imani, Tarek Nassar, Tajana Rosing, "Moving Toward Real-Time Diagnostics using Brain-Inspired Hyperdimensional Computing", AACR conference on Artificial Intelligence, Big Data, and Prediction in Cancer 2019
93. M Imani, S Gupta, Y Kim, T Rosing, “Deep Learning Acceleration using Digital-Based Processing In-Memory,” SOCC 2020.
94. S. Salamat, B. Khaleghi, M. Imani, T. Rosing, "Workload-Aware Opportunistic Energy Efficiency in Multi-FPGA Platforms", ICCAD, 2019.
95. M Imani, S Bosch, M Javaheripi, B Rouhani, X Wu, F Koushanfar, T. Rosing, “Semihd: Semi-supervised learning using hyperdimensional computing,” ICCAD 2019.
96. S Gupta, M Imani, T Rosing, “Exploring processing in-memory for different technologies,” GLVLSI, 2019.

97. J.Morris, M. Imani, S. Bosch, A. Thomas, H. Shu, T. Rosing, "CompHD: Efficient Hyperdimensional Computing Using Model Compression". IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), 2019.
98. J. Sim, M. Kim, Y. Kim, S. Gupta, B. Khaleghi, T. Rosing, "Multi-bit Parallelized Sensing for Processing in Non-volatile Memory", NVMW, 2019.
99. J Sim, M Imani, W Choi, Y Kim, T Rosing, "Current-sensing efficient adder for processing-in-memory design," NVM 2019.
100. Mohsen Imani, Saransh Gupta, Yeseong Kim, and Tajana S. Rosing, "FloatPIM: In-Memory Acceleration of Deep Neural Network Training with High Precision", ISCA, 2019
101. Mohsen Imani, Yeseong Kim, Sadegh Riyazi, John Merssely, Patrick Liu, Farinaz Koushanfar, and Tajana S. Rosing, "A Framework for Collaborative Learning in Secure High-Dimensional Space", IEEE Cloud Computing (CLOUD), 2019
102. Mohsen Imani, Yeseong Kim, Thomas Worley, Sarangh Gupta, and Tajana S. Rosing, "HDCluster: An Accurate Clustering Using Brain-Inspired High-Dimensional Computing", IEEE/ACM Design Automation and Test in Europe Conference (DATE), 2019
103. Mohsen Imani, John Merssely, Fan Wu, Wang Pi, and Tajana S. Rosing, "A Binary Learning Framework for Hyperdimensional Computing", IEEE/ACM Design Automation and Test in Europe Conference (DATE), 2019
104. Mohsen Imani, Ricardo Garcia, Andrew Huang, and Tajana S. Rosing, "CADE: Configurable Approximate Divider for Energy Efficiency", IEEE/ACM Design Automation and Test in Europe Conference (DATE), 2019
105. Xiao Liu, Minxuan Zhou, Tajana Rosing, and Jishen Zhao, "HR3AM: A Heat Resilient Design for RRAM-based Neuromorphic Computing", ISLPED 2019
106. Minxuan Zhou, Mohsen Imani, Saransh Gupta, Yeseong Kim, and Tajana Rosing, "Thermal-Aware Design and Management for Search-based In-Memory Acceleration", IEEE/ACM Design Automation Conference (DAC), 2019
107. Mohsen Imani, Tarek Nassar, Justin Morris, Tajana Rosing, "DNA Sequencing using Brain-inspired Hyperdimensional Computing", GOMACTech Conference, 2019
108. Mohsen Imani, Yeseong Kim, Saransh Gupta, Daniel Peroni, and Tajana S. Rosing, "In-Memory Acceleration of Deep Neural Network", GOMACTech Conference, 2019
109. Mohsen Imani, Tarek Nassar, Tajana Rosing, "Brain-Inspired Hyperdimensional Computing for Real-Time Health Analysis", IEEE International Conference on Biomedical and Health Informatics (BHI), 2019
110. Mohsen Imani, Saransh Gupta, Yeseong Kim, Minxuan Zhou, and Tajana S. Rosing, "DigitalPIM: Digital-based Processing In-Memory for Big Data Acceleration", ACM Great lakes symposium on VLSI (GLSVLSI), 2019
111. Anthony Thomas, Yunhui Guo, Yeseong Kim, Baris Aksanli, Arun Kumar, and Tajana Rosing, "Hierarchical and Distributed Machine Learning Inference Beyond the Edge", IEEE International Conference on Networking, Sensing and Control (ICNSC), 2019.
112. Daniel Peroni, Mohsen Imani, Hamid Nejatollahi, Nikil Dutt, and Tajana S. Rosing, "ARGA: Approximate Reuse for GPGPU Acceleration", IEEE/ACM Design Automation Conference (DAC), 2019
113. Mohsen Imani, Alice Sokolova, Ricardo Garcia, Andrew Huang, Fan Wu, and Tajana S. Rosing, "ApproxLP: Approximate Multiplication with Linearization and Iterative Error Control", IEEE/ACM Design Automation Conference (DAC), 2019
114. Mohsen Imani, Justin Morris, John Merssely, Helen Shu, Yaobang Deng, and Tajana S. Rosing, "BRIC: Locality-based Encoding for Energy-Efficient Brain-Inspired Hyperdimensional Computing", IEEE/ACM Design Automation Conference (DAC), 2019. **Best paper nomination**
115. Mohsen Imani, Sahand Salamat, Saransh Gupta, Jiani Huang, and Tajana S. Rosing, "FACH: FPGA-based Acceleration of Hyperdimensional Computing by Reducing Computational Complexity", IEEE Asia and South Pacific Design Automation Conference (ASP-DAC), 2019
116. Michael H Ostertag, Nikolay Atanasov, Tajana Simunic Rosing, "Robust Velocity Control for Minimum Steady State Uncertainty in Persistent Monitoring Applications", American Controls Conference, 2019. **Best poster award**
117. Saransh Gupta, Mohsen Imani, Behnam Khaleghi, Venkatesh Kumar, and Tajana S. Rosing, "RAPID: A ReRAM Processing in Memory Architecture for DNA Sequence Alignment", International Symposium on Low Power Electronics and Design (ISLPED), 2019
118. Mohsen Imani, Sahand Salamat, Behnam Khaleghi, Mohammad Samragh, Farinaz Koushanfar, and Tajana S. Rosing, "SparseHD: Algorithm-Hardware Co-Optimization for Efficient High-Dimensional Computing", International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2019

119. Sahand Salamat, Mohsen Imani, Behnam Khaleghi, and Tajana S. Rosing, "F5-HD: Fast Flexible FPGA-based Framework for Refreshing Hyperdimensional Computing", ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), 2019
120. Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, Rogério Schmidt Feris, "SpotTune: Transfer Learning through Adaptive Fine-tuning", In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019
121. Yunhui Guo, Yandong Li, Liqiang Wang, Tajana Rosing, "Depthwise Convolution is All You Need for Learning Multiple Visual Domains", The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI), 2019
122. Joonseop Sim, Saransh Gupta, Mohsen Imani, Yeseong Kim, and Tajana S. Rosing, "UPIM: Unipolar Switching Logic for High Density Processing-in-Memory Applications", ACM Great lakes symposium on VLSI (GLSVLSI), 2019
123. Yeseong Kim, Ankit More, Emily Shriver, and Tajana S. Rosing, "Application Performance Prediction and Optimization Under Cache Allocation Technology", IEEE/ACM Design Automation and Test in Europe Conference (DATE), 2019
124. Joonseop Sim, Minsu Kim, Yeseong Kim, Saransh Gupta, Behnam Khaleghi and Tajana Rosing, "MAPIM: Mat Parallelism for High Performance Processing in Non-volatile Memory Architecture", IEEE International Symposium on Quality Electronic Design (ISQED), 2019
125. M. Imani, M. Samragh, Y. Kim, S. Gupta, F. Koushanfar, T. Rosing, "A Fully Digital In-Memory Acceleration of Deep Neural Network" Non-Volatile Memory Workshop (NVMW). 2019.
126. Daniel Peroni, Mohsen Imani, Tajana Rosing, "ALook: Adaptive Lookup for GPGPU Acceleration", IEEE Asia and South Pacific Design Automation Conference (ASP-DAC), 2019
127. Minxuan Zhou, Mohsen Imani, Saransh Gupta, Tajana Rosing, "GRAM: Graph Processing in a ReRAM-based Computational Memory", IEEE Asia and South Pacific Design Automation Conference (ASP-DAC), 2019
128. Mohsen Imani, Sahand Salamat, Jiani Huang, Saransh Gupta, Tajana Rosing, "FACH: FPGA-based Acceleration of Hyperdimensional Computing by Reducing Computational Complexity", IEEE Asia and South Pacific Design Automation Conference (ASP-DAC), 2019
129. Joonseop Sim, Mohsen Imani, Woojin Choi, Yeseong Kim, Tajana Rosing, "LUPIS: Latch-Up Based Ultra Efficient Processing-in-Memory System", **Best paper nomination**, ISQED'18.
130. Minxuan Zhou, Mohsen Imani, Saransh Gupta, and Tajana Rosing, "GAS: A Heterogeneous Memory Architecture for Graph Processing," ISLPED '18.
131. Mohsen Imani, Tarek Nassar, Abbas Rahimi, Tajana Rosing "HDNA: Energy-Efficient DNA Sequencing Using Hyperdimensional Computing", IEEE International Conference on Biomedical and Health Informatics (BHI), 2018.
132. M. Imani, Y. Kim, T. Rosing, "Visual Object Recognition Accelerator Based on Approximate In-Memory Processing", Non-Volatile Memory Workshop (NVMW), 2018.
133. M. Imani, A. Rahimi, D. Kong, T. Rosing, Jan Rabaey, "Non-Volatile Associative Memory to Accelerate Brain-inspired Hyperdimensional Computing" Non-Volatile Memory Workshop (NVMW), 2018 .
134. M. Imani, S. Gupta, T. Rosing, "Accelerating Multiplication and Parallelizing Operations in Non-Volatile Memory" Non-Volatile Memory Workshop (NVMW), 2018.
135. M. Imani, C. Huang , D. Kong, T. Rosing, "Hierarchical Hyperdimensional Computing for Energy Efficient Classification", IEEE/ACM Design Automation Conference (DAC), 2018.
136. M. Imani, R. Garcia , S. Gupta, T. Rosing, "Configurable Floating Point Multiplier for Approximate Computing", IEEE/ACM Design Automation Conference (DAC), 2018.
137. S. Gupta, M. Imani, T. Rosing, "Processing In-Memory Architecture for Multiple Memory Technology", IEEE/ACM Design Automation Conference (DAC), 2018.
138. M. Imani, R. Garcia, S. Gupta, T. Rosing "RMAC: Runtime Configurable Floating Point Multiplier for Approximate Computing", IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), 2018.
139. M. Zhou, M. Imani, S. Gupta, T. Rosing "GAS: A Heterogeneous Memory Acceleration for Graph Processing", IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), 2018.
140. M. Imani, S. Gupta, T. Rosing, "GenPIM: Generalized Processing In-Memory to Accelerate Data Intensive Applications ", IEEE/ACM Design Automation and Test in Europe Conference (DATE), 2018.
141. S. Gupta, M. Imani, T. Rosing, "FELIX: Fast and Energy-Efficient Logic in Memory" IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2018.

142. M. Imani, Y. Kim, T. Rosing, "Brain-Inspired Hyperdimensional Computing: An Efficient Classifier for Embedded Devices" IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2018.
143. Y. Kim, M. Imani, T. Rosing, "Efficient Human Activity Recognition Using Hyperdimensional Computing", IEEE Conference on Internet of Things (IoT), 2018.
144. J.Sim, M. Imani, W. Choi, Y. Kim, T. Rosing, "LUPIS : Latch-Up Based Ultra Efficient Processing In-Memory System", IEEE International Symposium on Quality Electronic Design (ISQED), 2018. (Best paper nomination).
145. M. Imani, M. Masich, D. Peroni, P. Wang, T. Rosing, "CANNA: Neural Network Acceleration using Configurable Approximation on GPGPU", IEEE Asia and South Pacific Design Automation Conference (ASP-DAC), 2018.
146. M. Imani, D. Peroni, T. Rosing, "Resistive Content Addressable Memory for Configurable Approximation", GOMACTech Conference, 2018.
147. M. Imani, M. Masich, T. Rosing, "Training Acceleration of Deep Neural Network on Configurable GPGPU" Techon SRC Conference, 2018 .
148. M. Imani, T. Nassar, T. Rosing, "Moving Toward Real-Time Diagnostics using Brain-Inspired Hyperdimensional Computing", AACR conference on Artificial Intelligence, Big Data, and Prediction in Cancer, 2018.
149. S. Salamat, M. Imani, S. Gupta, T. Rosing, "RNSnet: In-Memory Neural Network Acceleration Using Residue Number System" IEEE International Conference on Rebooting Computing (ICRC), 2018.
150. M. Imani, D. Peroni, T. Rosing, "Program Acceleration Using Nearest Distance Associative Search", IEEE International Symposium on Quality Electronic Design (ISQED), 2018.
151. Anthony Thomas, Yunhui Guo, Yeseong Kim, Baris Aksanli, Arun Kumar, Tajana S. Rosing. "Hierarchical and Distributed Machine Learning Inference Beyond the Edge." In Government Microcircuit Applications & Critical Technology Conference (GOMACTech), 2018.
152. Y. Kim, A. Moore, E. Shriver, T. S. Rosing, "P4: Phase-Based Power/Performance Prediction of Heterogeneous Systems via Neural Networks," ICCAD'17.
153. P. Mercati. R. Ayoub, E. Samson, F. Paterna, M. Beuchat, M. Kisnevsky, T. S. Rosing, "Multi-variable Dynamic Power Management for the GPU Subsystem," DAC'17.
154. C. Chan, M. Ostertag, A. S. Akyurek, T. S. Rosing. "Context-Aware System Design," Invited paper to SPIE 2017.
155. B. Aksanli, J. Venkatesh, C. Chan, A. S. Akyurek, T. S. Rosing, "Context-Aware and User-Centric Residential Energy Management," PerIoT, 2017.
156. N. Mousavi, B. Aksanli, A. S. Akyurek, T. S. Rosing, "Accuracy-Resource Tradeoff for Edge Devices in Internet of Things," SmartEdge 2017.
157. W. Cui, Y. Kim, T. S. Rosing, "Cross-Platform Machine Learning Characterization for Task Allocation in IoT Ecosystems," **Best paper award**, IEEE CCWC, 2017.
158. M. Imani, A. Rahimi, D. Kong, T. Rosing, J. M. Rabaey "Exploring Hyperdimensional Associative Memory", HPCA'17.
159. M. Imani, S. Gupta, T. Rosing "Ultra-Efficient Processing In-Memory for Data Intensive Applications", DAC'17.
160. M. Imani, D. Peroni, T. Rosing "CFPU: Configurable Floating Point Multiplier for Energy-Efficient Computing", DAC'17 (**Best poster at ResearchExpo**).
161. Y. Kim, M. Imani, T. Rosing "ORCHARD: Visual Object Recognition Accelerator Based on Approximate In-Memory Processing", ICCAD'17.
162. Mohsen Imani, Yeseong Kim, and Tajana S. Rosing, "Brain-Inspired Hyperdimensional Computing: An Efficient Classifier for Embedded Devices," ICCAD'17.
163. M. Imani, A. Rahimi, D. Kong, T. Rosing, J. M. Rabaey "Hardware Acceleration of Brain-inspired Hyperdimensional Computing", ICCAD VMC 2017.
164. M. Imani, D. Peroni, Y. Kim, A. Rahimi and T. Rosing, "Efficient Neural Network Acceleration on GPGPU using Content Addressable Memory," DATE'17.
165. M. Samragh, M. Imani, F. Koushanfar and T. Rosing, "LookNN: Neural Network with No Multiplication," DATE'17.
166. M. Imani, S. Gupta, A. Arredondo, T. Rosing "Efficient Query Processing in Crossbar Memory", ISLPED'17.
167. M. Imani, Y. Kim, T. Rosing, "MPIM: Multi-Purpose In-Memory Processing using Configurable Resistive Memory" ASP-DAC'17.
168. M. Imani, D. Kong, A. Rahimi, T. Rosing, "VoiceHD: Hyperdimensional Computing for Efficient Speech Recognition", ICRC'17.

169. M. Imani, Y. Kim, T. Rosing “NNgine: Ultra-Efficient Nearest Neighbor Accelerator Based on In-Memory Computing”, ICRC’17.
170. J. Sim, M. Imani, Y. Kim, T. Rosing “Enabling Efficient System Design Using Vertical Nanowire Transistor Current Mode Logic”, VLSI-SoC’17.
171. M. Imani, D. Kong, A. Rahimi and T. Rosing, Jan Rabaey, "Brain-Inspired Hyperdimensional Computing: Robust, Scalable and Energy Efficient Classifier," Techcon’17.
172. Y. Kim, M. Imani and T. Rosing, "General-Purpose Online Classification Accelerator via In-Memory Computing," Techon’17.
173. M. Imani, T. Rosing, "CAP: Configurable Resistive Associative Processor for Near-Data Computing," IEEE ISQED’17.
174. M. Imani, D. Peroni, A. Rahimi, T. Rosing, “Non-volatile Content Addressable Memory for Computing Acceleration” NVMW’17.
175. M. Imani, Y. Kim, T. Rosing, “In-Memory Processing to Support Search-Based and Bitwise Computation” NVMW’17.
176. Pietro Mercati, Francesco Paterna, Andrea Bartolini, Mohsen Imani, Luca Benini, Tajana S. Rosing, "VarDroid: Online Variability Emulation in Android/Linux Platforms", GLSVLSI, 2016
177. Akanksha Maurya, Alper Sinan Akyurek, Baris Aksanli and Tajana Rosing, "Time-Series Clustering for Data Analysis in Smart Grid", SmartGridComm, 2016.
178. J. Venkatesh, C. Chan, A. S. Akyurek, B. Aksanli, T. S. Rosing, “A Modular Approach to Context-Aware IoT Applications,” IOTDI’16.
179. A. S. Akyurek, T. S. Rosing, “Optimal In-Network Packet Aggregation Policy for Maximum Information Freshness,” EUCNC’16.
180. Jinseok Yang, S. Tilak, T. S. Rosing, "Interactive Context-aware Power Management Technique for Optimizing Sensor Network Lifetime,” SENSORNETS’16, nominated for the **Best paper award**.
181. Yeseong Kim, Pietro Mercati, and Tajana S. Rosing, "Power Efficient, Hierarchical, Introspection Framework for HPC Systems," TECHCON SRC Conference (TECHCON 2016), September 2016
182. M. Imani, D. Peroni, A. Rahimi, T. Rosing, “Resistive CAM Acceleration for Tunable Approximate Computing” ICCD’16. **Best paper nomination, Selected as top ranked conference paper for publishing in IEEE TETC.**
183. M. Imani, Y. Kim, A. Rahimi, T. Rosing, "ACAM: Approximate Computing Based on Adaptive Associative Memory with Online Learning" ISLPED’16.
184. M. Imani, A. Rahimi, Y. Kim, T. Rosing, "A Low-Power Hybrid Magnetic Cache Architecture Exploiting Narrow-Width Values" NVMSA’16.
185. M. Imani, A. Rahimi, T. Rosing, "Resistive Configurable Associative Memory for Approximate Computing" DATE’16.
186. M. Imani, S. Patil, T. Rosing, "MASC: Ultra-Low Energy Multiple-Access Single-Charge TCAM for Approximate Computing" DATE’16.
187. M. Imani, Y. Cheng, T. Rosing, "Processing Acceleration with Resistive Memory-based Computation" MEMSYS’16.
188. M. Imani, P. Mercati, T. Rosing, "ReMAM: Low Energy Resistive Multi-Stage Associative Memory for Energy Efficient Computing" ISQED’16.
189. M. Imani, S. Patil, T. Rosing, "Low Power Data-Aware STT-RAM based Hybrid Cache Architecture" ISQED’16.
190. M. Imani, Y. Kim, A. Rahimi, T. Rosing, "Associative Memory with Online Learning for Approximate Computing" Poster in DAC’16.
191. M. Imani, Y. Cheng, T. Rosing, "Resistive Memory for Approximate Program Acceleration" NVMW’16.
192. P. Mercati, A. Bartolini, F. Paterna, M. Imani, L. Benini and T. Rosing, "VarDroid: Online Variability Emulation in Android/Linux Platforms" GLSVLSI’16.
193. M. Imani, S. Patil, T. Rosing, "DCC: Double Capacity Cache for Narrow-Width Data Values" GLSVLSI’16.
194. M. Imani, A. Rahimi, T. Rosing, "Ultra-Efficient Content Addressable Memory for Tunable GPU Approximation" TECHCON’16.
195. M. Imani, S. Patil, T. Rosing, “Hierarchical Design of Robust and Low Data Dependent FinFET Based SRAM Array”, NANOARCH’15.
196. M. Imani, S. Patil, M. Jafari, T. Rosing, “Ultra-Low Read leakage SRAM Cell Utilizing Independently-Controlled-Gate FinFET”, Poster in DAC’15.

197. M. Imani, S. Patil, T. Rosing, "Using STT-RAM Based Buffers in Digital Circuits", NVMW'15.
198. B. Aksanli, A. S. Akyurek, T. S. Rosing, "User Behavior Modeling for Estimating Residential Energy Consumption," Invited paper at SGSC'15.
199. J. Venkatesh, S. Chen, P. Tinnakornsrisuphap, T. S. Rosing, "Lifetime-dependent Battery Usage Optimization for Grid-Connected Residential Systems", MSCPES, 2015.
200. Mercati P, Hanumaiah V., Kulkarni J., Bloch S. and Rosing T. "BLAST: Battery Lifetime-constrained Adaptation with Selected Target" MOBIQUITOUS 2015.
201. A. S. Akyurek and B. Aksanli and T. S. Rosing, S2Sim: Smart Grid Swarm Simulator, IGSC 2015.
202. Baris Aksanli, Alper Sinan Akyurek, Tajana Simunic Rosing, "Minimizing the Effects of Data Centers on Microgrid Instability", IGSC'15
203. Y. Kim, M. Imani, S. Patil, T. S. Rosing, "CAUSE: Critical Application Usage-Aware Memory System using Non-volatile Memory for Mobile Devices," ICCAD'15.
204. Y. Kim, F. Paterna, T. S. Rosing, "Smartphone Analysis and Optimization based on User Activity Recognition," ICCAD'15.
205. Shruti Patil, Yeseong Kim, Kunal Korgaonkar, Ibrahim Awwal, Tajana S. Rosing, "Characterization of User's Behavior Variations for Design of Replayable Mobile Workloads", MOBICASE 2015.
206. F. Paterna, T. S. Rosing, "Modeling and Mitigation of Extra-SoC Thermal Coupling Effects and Heat Transfer Variations in Mobile Devices," ICCAD'15.
207. H. Rodrigues, R. Strong, T. S. Rosing. "Accurate Emulation of Fast Optical Circuit Switches", ICC'15.
208. H. Rodrigues, R. Strong, A. Akyurek, T. S. Rosing. "Dynamic Optical Switching for Latency Sensitive Applications", ACM/IEEE Symposium on Architectures for Networking and Communications Systems, 2015
209. Y. Chen, S. Patil, T. S. Rosing, "GazeTube: Gaze-Based Adaptive Video Playback for Bandwidth and Power Optimizations," Globecom 2015.
210. Jinseok Yang, S. Tilak, T. S. Rosing, "Transmission manager in heterogeneous applications running WSNs," IEEE Globecom 2015
211. Jagannathan Venkatesh, Christine Chan, Alper Sinan Akyurek, Tajana Simunic Rosing, "A Context-Driven IoT Middleware Architecture", TechCon, 2015.
212. Christine Chan, Alper Sinan Akyurek, Kalyan Vaidyanathan, Kenny Gross, Tajana Rosing, "Optimization of Energy, Cooling and IO Performance for Data-intensive Applications on Enterprise Servers", TECHCON, 2015.
213. Pietro Mercati, Francesco Paterna, Andrea Bartolini, Luca Benini, Tajana Simunic Rosing, "Variability Emulation on Real Linux/Android Devices", TECHCON 2015.
214. Jinseok Yang, S. Tilak, T. S. Rosing, "Leveraging application context for efficient sensing," IEEE ISSNIP 2014
215. Baris Aksanli and Tajana Rosing, Providing Regulation Services and Managing Data Center Peak Power Budgets. Design, Automation and Test in Europe (DATE), 2014.
216. H. Rodrigues, I. Monga, A. Sadasivarao, S. Syed, C. Guok, E. Pouyoul, C. Liou, and T. S. Rosing. "Traffic Optimization in Multi-Layered WANs using SDN." IEEE High-Performance Interconnects, 2014. **Best paper award.**
217. A. Sadasivarao, H. Rodrigues, S. Syed, C. Liou, S. Balakrishnan, A. Lake, E. Poyoul, C. Guok, I. Monga, T. Rosing. , "Enabling Multi-Layer Provisioning and Optimization for Core Transport Networks with Unified Packet-Optical Control Plan", 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI'14.
218. H. Rodrigues, A. Akyurek, T. Rosing, "OCSEMU: SDN Enabled Fast Hybrid Optical Circuit Switch Emulator Platform to Study Application Performance in the Emerging Optical Data Center", OIDA Software Defined Photonic and Data Center Networks Workshop, 2014.
219. H. Rodrigues, R. Strong, T. Rosing, "Scheduling Optical Tunnels to Distributed Applications", USENIX Annual Technical Conference, ATC'14.
220. B. O. Akyurek and A. S. Akyurek and J. Kleissl and T. S. Rosing, TESLA: Taylor Expanded Solar Analog Forecasting, IEEE SmartGridComm 2014
221. Mercati P, Paterna F., Bartolini A, Benini L and Rosing T "Variability Management in Mobile Multicore Processors under Lifetime Constraints", ICCD'14.
222. Mercati P, Bartolini A, Paterna F., Benini L and Rosing T "An On-line Reliability Emulation Framework" in Embedded and Ubiquitous Computing, IEEE Proceedings of the International Conference on (EUC14), 2014.
223. Paterna F., Zanolletti J. and Rosing T. "Ambient variation-tolerant and inter components aware thermal management for mobile system on chips" DATE'14.

224. Mercati P, Bartolini A, Paterna F., Benini L and Rosing T “A Linux-Governor Based Dynamic Reliability Manager for Android Mobile Devices” DATE’14.
225. P. Mercati, T. Simunic Rosing, V. Hanumaiah, J. Kulkarni, S. Bloch, “User-centric Joint Power and Thermal Management for Smartphones,” MOBICASE’14.
226. Baris Aksanli, Alper Sinan Akyurek, Madhur Behl, Meghan Clark, Alexandre Donze, Prabal Dutta, Patrick Lazik, Mehdi Maasoumy, Rahul Mangharam, Truong X. Nghiem, Vasu Raman, Anthony Rowe, Alberto Sangiovanni-Vincentelli, Sanjit A. Seshia, Tajana Simunic Rosing, and Jagannathan Venkatesh. Distributed Control of a Swarm of Buildings Connected to a Smart Grid. 1st ACM International Conference on Embedded Systems For Energy-Efficient Buildings (BuildSys), 2014.
227. Baris Aksanli, Tajana Rosing, "Energy Management and Cost Analysis in Residential Houses using Batteries", SRC TECHCON, 2014
228. F. Seracini, X. Zhang, T. S. Rosing, I. Krueger, “A Proactive Customer-Aware Resource Allocation Approach for Data Centers”, ISPA’14.
229. Yeseong Kim, Francesco Paterna, Tajans S. Rosing, Sameer Tilak, "Fine-grained Analysis and Optimization of Smartphone Applications via Automated Phase Recognition for Improved User Experience," DCOSS'14
230. B. Milosevic, J. Yang, N. Verma, S. S. Tilak, Piero Zappi, Elisabetta Farella, L. Benini, T. Simunic Rosing, “Efficient Energy Management and Data Recovery in Sensor Networks using Latent Variables Based Tensor Factorization”, MSWiM, 2013.
231. A. S. Akyurek, B. Torre, T. S. Rosing, “ECO-DAC Energy Control Over Divide and Control,” IEEE SmartGridComm 2013.
232. B. Aksanli, T.S. Rosing, “Optimal Battery Configuration in a Residential Home with Time-of-Use Pricing,” IEEE SmartGridComm 2013.
233. C. Chan, B. Pan, K. Gross, K. Vaidyanathan, T. Rosing, "Correcting vibration-induced performance degradation in enterprise servers", SIGMETRICS Performance Evaluation Review, 2013. **Best paper award**
234. Baris Aksanli, Eddie Pettis, Tajana Rosing, "Architecting Efficient Peak Power Shaving Using Batteries in Data Centers", International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2013.
235. G. Porter, R. Strong, N. Farrington, A. Forencich, P. Sun, T. Rosing, Y. Fainman, G. Papen, A. Vahdat, “Integrating Microsecond Circuit Switching into the Data Center,” SIGCOMM’13.
236. Rajib Nath, Raid Ayoub, Tajana S. Rosing, "Temperature Aware Thread Block Scheduling in GPGPUs", Design Automation Conference, 2013
237. P. Mercati, A. Bartolini, F. Paterna, T. Simunic Rosing, L. Benini, “Workload and User Experience-Aware Dynamic Reliability Management in Multicore Processors,” DAC 2013.
238. J. Yang, S. Tilak. D. Krishniswamy, T Simunic Rosing, “A novel protocol for adaptive broadcasting of sensor data in urban scenarios,” GLOBECOM, 2013.
239. Baris Aksanli, Eddie Pettis, Tajana Rosing, "Distributed Battery Control for Peak Power Shaving in Data Centers ", International Green Computing Conference (IGCC), 2013.
240. J. Venkatesh, B. Aksanli, Jean-Claude Junqua, Philippe Morin, T. Simunic Rosing, "HomeSim: Comprehensive, Smart, Residential Electrical Energy Simulation and Scheduling", IGCC’13.
241. Baris Aksanli, Jagannathan Venkatesh, Tajana Rosing, and Inder Monga, "A Comprehensive Approach to Reduce the Energy Cost of Network of Datacenters”, ISCC, 2013. **Best paper award**
242. Jagannathan Venkatesh, Baris Aksanli, and Tajana Rosing, "Residential Energy Simulation and Scheduling: A Case Study Approach", International Symposium on Computers and Communications (ISCC), 2013
243. Rajib Nath, Douglas Carmean and Tajana S. Rosing, "Power Modeling and Thermal Management Techniques for Many Core Processors", The IEEE symposium on Computers and Communications (ISCC), 2013.
244. L. Zhang, G. Dhiman, and T. S. Rosing. "vGreenNet: Managing Server and Networking Resources of Co-located Heterogeneous VMs". IEEE International Parallel and Distributed Processing Symposium (IPDPS), High Performance Grid and Cloud Computing, 2013.
245. Filippo Seracini; Xiang Zhang; Ingolf Krueger; Tajana Rosing; Massimiliano Menarini, “Green Web Services: Improving Energy Efficiency in Data Centers via Workload Predictions,” ICSEWS’13 GREENS.
246. Andrew B. Kahng, Siddhartha Nath, Tajana S. Rosing, “On Potential Design Impacts of Electromigration Awareness,” ASPDAC’13.

247. Nima Nikzad, Nakul Verma, Celal Ziftci, Elizabeth Bales, Nichole Quick, Piero Zappi, Kevin Patrick, Sanjoy Dasgupta, Ingolf Krueger, Tajana Simunic Rosing, William G. Griswold. "CitiSense: Improving Geospatial Environmental Assessment of Air Quality Using a Wireless Personal Exposure Monitoring System". Wireless Health 2012. **Best paper award.**
248. G. Dhiman, V. Kontorinis, R. Ayoub, L. Zhang, C. Sadler+, D. Tullsen, T. Simunic Rosing, "Themis: Energy Efficient Management of Workloads in Virtualized Data Centers," EuroPar-VHPC'12.
249. Mohammad Moghimi, Jagannathan Venkatesh, Piero Zappi and Tajana Rosing, "Context-Aware Mobile Power Management Using Fuzzy Inference as a Service," MobiCASE'12.
250. V. Kontorinis, E. Zhang, B. Aksanli, J. Samson, H. Homayoun, E. Pettis, D. Tullsen, T. Simunic Rosing, "Managing Distributed UPS Energy for Effective Power Capping in Data Centers," ISCA 2012.
251. R. Strong, S. Kang, K. Jeong, A. Kahng, T. Simunic Rosing, "TAP: Token-aware Power Gating," ISLPED'12. (Note: authors listed in the order of contribution; the paper had alphabetical order)
252. C. Chan, Y. Jin, YK Wu, K. Gross, K. Vaidyanathan, R. Ayoub, T. Simunic Rosing, "Fan-Speed-Aware Scheduling of Data Intensive Jobs," ISLPED'12.
253. P. Zappi, E. Bales, JH Park, W. Griswold and T. Šimunić Rosing, "The CitiSense Air Quality Monitoring Mobile Sensor Node," IPSN-Mobile Sensing, 2012.
254. R. Herrmann, P. Zappi, T. Simunic Rosing, "Context Aware Power Management of Mobile Systems for Sensing Applications," IPSN-Mobile Sensing, 2012.
255. R. Ayoub, R. Nath, T. Simunic Rosing, "JETC: Joint Energy Thermal and Cooling Management for Memory and CPU Subsystems in Servers," HPCA 2012.
256. Nima Nikzad, Jinseok Yang, Piero Zappi, Tajana Simunic Rosing, and Dilip Krishnaswamy, "Model-driven Adaptive Wireless Sensing for Environmental Healthcare Feedback Systems," IEEE ICC 2012.
257. Baris Aksanli, Tajana S. Rosing, Inder Monga, "Benefits of Green Energy and Proportionality in High Speed Wide Area Networks Connecting Data Centers," DATE 2012.
258. R. Strong, S. Kang, K. Jeong, A. Kahng, T. Simunic Rosing, "MAPG: Memory Access Power Gating," DATE'12. (Note: authors listed in the order of contribution; the paper had alphabetical order)
259. S. Sharifi, R. Ayoub, T. Simunic Rosing, "TempoMP: Integrated Prediction and Management of Temperature in Heterogeneous MPSoCs," DATE'12.
260. Baris Aksanli, Jagannathan Venkatesh, Liuyi Zhang, Tajana Rosing, "Utilizing Green Energy Prediction to Schedule Mixed Batch And Service Jobs in Data Centers," HotPower 2011.
261. R. Ayoub, U. Ogras, E. Gorbatoov, Y. Jin, T. Kam, P. Diefenbough, T. Rosing, "OS-level Power Minimization Under Tight Performance Constraints in General Purpose Systems," ISLPED 2011.
262. Denis Dondi, Piero Zappi, Tajana Šimunić Rosing, "A Scheduling Algorithm for Consistent Monitoring Results with Solar Powered High-Performance Wireless Embedded Systems," ISLPED 2011.
263. Y. Wu, S. Sharifi, T. Simunic Rosing, "Distributed Thermal Management for Embedded Heterogeneous MPSoCs with Dedicated Hardware Accelerators", ICCD 2011.
264. S. Sharifi, Yen-Kuan Wu, T. Simunic Rosing, "Temperature-aware Scheduling for Embedded Heterogeneous MPSoCs with Special Purpose IP Cores," ETMEC 2011.
265. R. Ayoub, K. Indukuri, T. Simunic Rosing, "Energy Efficient Proactive Thermal Management in Memory Subsystem," ISLPED 2010.
266. G. Dhiman, K. Mihic, T. Simunic Rosing, "A system for online power prediction in virtualized environments using Gaussian mixture models," DAC'10.
267. Nichole Quick, Kevin Patrick, Nima Nikzad, Celal Ziftci, Piero Zappi, Priti Aghera, Nakul Verma, Barry Demchak, PJE Quintana, Ingolf Krueger, Tajana Rosing, Sanjoy Dasgupta, Hovav Shacham & William Griswold, "CitiSense – Adaptive Services for Community-Driven Behavioral and Environmental Monitoring to Induce Change, " invited poster at mHealthSummit 2010.
268. Claudiu Farcas, Filippo Seracini, Ingolf Krüger and Tajana Simunic Rosing, "Greening Datacenters through Software," invited poster at NASA Workshop on Global Collaboration in Environmental and Alternative Energy Strategies, 2010.
269. G. Dhiman, V. Kontorinis, D. Tullsen, T. Rosing; E. Saxe, J.Chew, "Dynamic Workload Characterization for Power Efficient Scheduling on CMP Systems," ISLPED 2010.
270. D. Dondi, A. Di Pompeo, C. Tenti, and T. S. Rosing, "SHiMmer: A Wireless Harvesting Embedded System for Active Ultrasonic Structural Health Monitoring, " IEEE Sensors 2010.

271. P. Aghera, D. Krishnaswamy, T. Rosing, "DynAGreen: Hierarchical Dynamic Energy Efficient Task Assignment for Wireless Healthcare Systems," *BodyNets*, 2010.
272. E.B. Flynn, S. Kpotufe, D. Harvey, E. Figueiredo, S. Taylor, D. Dondi, T. Mollov, M.D. Todd, T.S. Rosing, G. Park, and C. Farrar, "SHMTools: a embeddable software package for SHM applications," *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems*, SPIE, 2010.
273. C. Olschanowsky, L. Carrington, M. Tikir, M. Laurenzano, T. Rosing, A. Snively, "Fine-grained Energy Consumption Characterization and Modeling," DOD High Performance Computing Modernization Program User Group Conference, June 2010.
274. R. Ayub, S. Sharifi, T. Simunic Rosing, "GentleCool: cooling aware proactive workload scheduling in multi-machine systems," DATE'10.
275. P. Aghera, A. Coskun, D. Fang, D. Krishnaswamy, T. Simunic Rosing, "DynAHeal: Dynamic energy efficient task assignment for wireless healthcare systems," DATE'10.
276. A. Sitaraman, D. Dondi, T. Simunic Rosing, "DVFS Based Task Scheduling in a Harvesting WSN for Structural Health Monitoring," DATE'10.
277. A. Coskun, D. Atienza, T. Simunic Rosing, "Energy-efficient variable-flow liquid cooling in 3D stacked architectures," DATE'10.
278. R. Ayoub, T. Simunic Rosing, "Cool and Save: Cooling Aware Dynamic Workload Scheduling in Multi-socket CPU Systems," ASPDAC'10.
279. S. Sharifi, A. Coskun, T. Simunic Rosing, "Hybrid Dynamic Energy and Thermal Management in Heterogeneous Embedded Multiprocessors," ASPDAC'10.
280. E. Regini, T. Simunic Rosing, "An Energy Efficient Wireless Communication Mechanism for Sensor Node Cluster Heads," ISSNIP'09.
281. A. Coskun, J. Ayala, D. Atienza, T. Simunic Rosing, "Modeling and Dynamic Management of 3D Multicore Systems with Liquid Cooling," **Best paper award** at VLSI-SOC 2009.
282. A. Coskun, A. Kahng, T. Simunic Rosing, "Temperature- and Cost-Aware Design of 3D Multiprocessor Architectures", DSD'09.
283. A. Coskun, R. Strong, D. Tullsen, T. Simunic Rosing, "Evaluating the Impact of Job Scheduling and Power Management on Processor Lifetime for Chip Multiprocessors," SIGMETRICS'09.
284. R. Ayoub, T. Simunic Rosing, "Predict and Act: Dynamic Thermal Management for Multicore Processors," ISLPED'09.
285. G. Dhiman, R. Ayoub, G. Marchetti, T. Simunic Rosing, "vGreen: A System for Energy Efficient Computing in Virtualized Environments," **Nominated for the best paper award** at ISLPED'09.
286. G. Dhiman, R. Ayoub, T. Simunic Rosing, "PDRM: A hybrid PRAM DRAM main memory system", DAC'09.
287. P. Aghera, D. Fang, T. Simunic Rosing, K. Patrick "Energy management in wireless healthcare systems," IPSN'09.
288. J. Bradely Steck, T. Simunic Rosing, "Adapting Performance in Energy Harvesting Wireless Sensor Networks for Structural Health Monitoring Applications," **Invited paper** at IWSHM'09.
289. J. Bradely Steck, T. Simunic Rosing, "Adapting Task Utility in Externally Triggered Energy Harvesting Wireless Sensing Systems," INSS'09.
290. J. Recas, C. Bergonzini, T. Simunic Rosing, D. Atienza, "Prediction and Management in Energy Harvested Wireless Sensor Nodes," **Invited paper** at Wireless VITAE'09.
291. J. Recas, C. Bergonzini, B. Lee, T. Simunic Rosing, "Solar energy harvesting prediction algorithm," Energy Harvesting Workshop'09.
292. A. K. Coskun, T. Simunic Rosing, J. Ayala, D. Atienza, Y. Leblebici. "Dynamic Thermal Management in 3D Multicore Architectures," DATE 2009.
293. A. Coskun, T. Simunic Rosing, K. Gross, "Proactive temperature balancing for low cost thermal management in MPSOCs," ICCAD'08.
294. G. Dhiman, K. Pusukuri, T. Simunic Rosing, "Analysis of Dynamic Voltage Scaling for System Level Energy Management," USENIX-HotPower'08.
295. E. Regini, D. Lim, T. Simunic Rosing, "Distributed scheduling for heterogeneous wireless sensor networks," IASTEAD'08.
296. A. Coskun, T. Simunic Rosing, K. Gross, "Proactive temperature management in MPSOCs," ISLPED'08.
297. A. Coskun, T. Simunic Rosing, K. Gross, "Temperature management in MPSOCs using online learning," DAC'08.

298. A. Coskun, T. Simunic Rosing, "Temperature-aware MPSOC scheduling for reducing hot spots and gradients," ASPDAC'08.
299. S. Sharifi, T. Simunic Rosing, "An analytical model for the upper bound on temperature differences on a chip," GLVLSI'08.
300. S. Sharifi, T. Simunic Rosing, "Accurate temperature sensing for efficient thermal management," ISQED'08.
301. G. Dhiman, T. Simunic Rosing, "Dynamic Voltage Scaling using Machine Learning," ISLPED'07.
302. O. Milenkovic, R. Baraniuk, and T. Simunic Rosing, "Compressed sensing meets bioinformatics: A novel DNA microarray design," in *Second Annual ITA Workshop*, San Diego, California, January 2007.
303. Todd, M., Mascarenas, D., Flynn, E., Rosing, T., Lee, B., Musiani, D., Dasgupta, S., Kpotufe, S., Hsu, D., Gupta, R., Park, G., Overly, T., Nothnagel, M., Farrar, C., "A different approach to sensor networking for SHM: Remote powering and interrogation with unmanned aerial vehicles", *Keynote at Workshop on Structural Health Monitoring*, 2007.
304. D. Musiani, K. Lin, T. Simunic Rosing, "An Active Sensing Platform for Structural Health Monitoring Application," IPSN-SPOTS'07.
305. A. Coskun, T. Simunic Rosing, "Temperature-aware task scheduling," DATE'07.
306. D. Lim, J. Shim, T. Simunic Rosing, T. Javidi, "Scheduling data delivery in heterogeneous wireless sensor networks," ISM'06.
307. G. Dhiman, T. Simunic Rosing, "Dynamic Power Management Using Machine Learning," *Nominated for the best paper award* at ICCAD'06
308. A. Coskun, T. Simunic Rosing, "A Simulation Methodology for Reliability Analysis in Multi-Core SoCs," GVLSI'06
309. T. Simunic, K. Mihic, G. De Micheli: "Optimization of Reliability and Power Consumption in Systems on a Chip," PATMOS'05.
310. T. Simunic, W. Quadeer, G. De Micheli: "Managing heterogeneous wireless environments via Hotspot servers," MMCN'05.
311. T. Simunic, K. Mihic, G. De Micheli: "Reliability and Power Management of Integrated Systems," *Invited paper* at DSD'04
312. G. Manjunath, V. Krishnan, T. Simunic, J. Tourrilhes, A. McReynolds, D. Das, V. Srinivasmurthy, A. Srinivasan: "Smart Edge Server – going beyond a wireless access point," WMASH'04.
313. O. Celebican, T. S. Rosing, V. J. Mooney: "Energy estimation of peripheral devices in embedded systems," GLVLSI'04.
314. W. Quadeer, T. Simunic, J. Ankcorn, V. Krishnan, G. De Micheli, "Heterogeneous wireless network management", PACS'03.
315. A. Acquaviva, T. Simunic, V. Deolalikar, S. Roy: "Remote Power Control of Wireless Network Interfaces", PATMOS'03.
316. B. Delaney, N. Jayant, T. Simunic: "A WLAN Scheduling Algorithm to Reduce the Energy Consumption of a Distributed Speech Recognition Front-End", ESTIMedia'03.
317. A. Peymandoust, T. Simunic, G. De Micheli: "Complex Software Library Element Mapping with Symbolic Algebra", DAC'02.
318. T. Simunic, S. Boyd: "Managing Power Consumption in Networks on Chips", DATE'02.
319. A. Peymandoust, T. Simunic, G. De Micheli: "Low Power Embedded Software Optimization using Symbolic Algebra", pp. 1052-1057, DATE'02.
320. B. Delaney, N. Jayant, M. Hans, T. Simunic, A. Acquaviva: "Low-Power Fixed-Point Front-End Feature Extraction for Distributed Speech Recognition", ICASSP'02.
321. T. Simunic, L. Benini, A. Acquaviva, P. Glynn, G. De Micheli: "Dynamic Voltage Scaling for Portable Systems", DAC'01.
322. T. Simunic, L. Benini, P. Glynn, G. De Micheli: "Dynamic Power Management of Portable Systems", MOBICOM'00.
323. T. Simunic, L. Benini, G. De Micheli, M. Hans: "Source Code Optimization and Profiling of Energy Consumption in Embedded Systems", *Invited paper* at ISSS'00.
324. T. Simunic, H. Vikalo, P. Glynn, G. De Micheli: "Energy Efficient Design of Portable Wireless Systems", ISLPED'00.
325. T. Simunic, L. Benini, P. Glynn, G. De Micheli: "Dynamic Power Management of Laptop Hard Disk", DATE'00.

326. Y. Lu, E. Chung, T. Simunic, L. Benini, G. De Micheli: “Quantitative Comparison of Power Management Algorithms”, pp.20-26, DATE’00, Selected for publication in *The Most Influential Papers of 10 Years DATE*, Edited by Lauwereins, Rudy; Madsen, Jan, 2008.
327. T. Simunic, L. Benini, G. De Micheli: “Event-driven Power Management of Portable Systems”, ISSS’99.
328. T. Simunic, L. Benini, G. De Micheli: “Energy-efficient design of Battery-Powered Embedded Systems”, ISLPED’99.
329. T. Simunic, L. Benini, G. De Micheli: “Cycle-Accurate Simulation of Energy Consumption in Embedded Systems”, DAC’99.
330. Y. Lu, T. Simunic, G. De Micheli: “Software Controlled Power Management”, CODES’99.
331. J. Rozenblit, T. Simunic: “Techniques for Intelligent VLSI Interconnect Design,” DMC’94.
332. T. Simunic, J. Rozenblit: “Reduction of Signal Delay and Crosstalk in Electronic Packaging,” EPEP’93.
333. T. Simunic, P. Hsu, J. Rozenblit, C. Wolff, J. Prince, A. Cangelaris: “An Integrated Framework for Modeling and Simulation of Electronic Packaging,” TECHCON’93

#### PREPRINTS

1. S Gupta, R Cammarota, T Rosing, “MemFHE: End-to-End Computing with Fully Homomorphic Encryption in Memory,” arXiv:2204.12557
2. R. Fielding-Miller, S. Karthikeyan, T. Gaines, T. Rosing et al., “Wastewater and surface monitoring to detect COVID-19 in elementary school settings: The Safer at School Early Alert project”, Medrxiv, 2021.
3. R Cammarota, M Schunter, A Rajan, F Boemer, Á Kiss, A Treiber, ... T. Rosing, “Trustworthy ai inference systems: An industry research view,” arXiv:2008.04449
4. S Bosch, AS de la Cerda, M Imani, TS Rosing, G De Micheli, “QubitHD: A stochastic acceleration method for HD computing-based machine learning,” arXiv:1911.12446
5. M Imani, M Samragh, Y Kim, S Gupta, F Koushanfar, T Rosing, “Rapidnn: In-memory deep neural network acceleration framework,” arXiv:1806.05794
6. S Salamat, T Rosing, “FPGA Acceleration of Sequence Alignment: A Survey,” arXiv:2002.02394

#### BOOK CHAPTERS

1. M. Imani, T. Rosing, “Approximate CPU and GPU design using emerging memory technologies,” Approximate Circuits, Book chapter in Approximate Circuits by Springer 2019.
2. S. Patil, Y. Kim, K. Korgaonkar, I. Awwal, T. S. Rosing, “Characterization of User’s Behavior Variations for Design of Replayable Mobile Workloads,” in Mobile Computing, Applications, and Services, Volume 162 of the series Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp 51-71, January, 2016.
3. Baris Aksanli, Jagannathan Venkatesh, Inder Monga, and Tajana Rosing, “Reable Energy Prediction for Improved Utilization and Efficiency in Datacenters and Backbone Networks,” Computational Sustainability Springer Book Chapter, 2015.
4. Ayse K. Coskun, J. Ayala, D. Atienza, T. Simunic Rosing: “Thermal Modeling and Management of Liquid-Cooled 3D Stacked Architectures,” Editors: J. Becker, M. Johann and R. Reis. Springer, VLSI-SoC: Technologies for Systems Integration (ISBN: 978-3-642-23119-3), p. 34-55, 2011.
5. G. Dhiman, R. Ayoub, T. Simunic Rosing, “Energy and Thermally Aware Scheduling in Datacenters,” in Energy-Efficient Distributed Computing, Edited by Albert Zomaya & Young Choon Lee, Wiley-Interscience 2010.
6. N. Nikzad, P. Aghera, P. Zappi, T. Simunic Rosing, “Energy Management in Heterogeneous Wireless Healthcare Networks,” in Energy-Efficient Distributed Computing, Edited by Albert Zomaya & Young Choon Lee, Wiley-Interscience 2010.
7. Ayse K. Coskun, J. Ayala, D. Atienza, T. Simunic Rosing. Thermal Modeling and Management of Liquid-Cooled 3D Stacked Architectures. Editors: J. Becker, M. Johann and R. Reis. Springer, VLSI-SoC: Technologies for Systems Integration (ISBN: 978-3-642-23119-3), p. 34-55, 2011.
8. Y. Lu, E. Chung, T. Simunic, L. Benini, G. De Micheli: “Quantitative Comparison of Power Management Algorithms”, in *The Most Influential Papers of 10 Years DATE*, Edited by Lauwereins, Rudy; Madsen, Jan, Springer-Verlag, 2008.

9. J. Kim, T. Simunic Rosing, "Power-aware resource management techniques for low-power embedded systems," in Handbook of Real-Time and Embedded Systems, Edited by S. H. Son, I. Lee, J. Y-T Leung, Taylor-Francis Group LLC, 2006.
10. T. Simunic: "Dynamic Management of Power Consumption" in Power Aware Computing, Edited by R. Graybill, R. Mehlem, Kluwer Academic Publishers pp.102-125, 2002.

#### PATENTS

1. TS Rosing, M Imani, Y Kim, B Khaleghi, AN Moshiri, S Gupta, V Kumar, "Methods, circuits, and articles of manufacture for searching within a genomic reference sequence for queried target sequence using hyper-dimensional computing techniques," US20220059189, Priority 2021-02-24.
2. TS Rosing, J Morris, M Imani, Y Kim, J Messerly, Y Guo, B Khaleghi, "Circuits, methods, and articles of manufacture for hyper-dimensional computing systems and related applications," US20220019441A1, Priority 2020-07-14;
3. S Salamat, M Imani, B Khaleghi, T Rosing," Methods and systems configured to specify resources for hyperdimensional computing implemented in programmable devices using a parameterized template for hyperdimensional computing," US20210334703A1, Priority date 2020-02-21.
4. B Khaleghi, TS Rosing, M Imani, S Salamat, "Methods of providing trained hyperdimensional machine learning models having classes with reduced elements and related computing systems," US20210326756A1, Priority 2020-04-07.
5. M Imani, Y Kim, T Rosing, F Koushanfar, MS Riazi, "Systems, circuits and computer program products providing a framework for secured collaborative training using hyper-dimensional vector based data encoding/decoding and related," US20200410404A1, Priority 2019-06-27.
6. A. Acquaviva, B. Luca, T. Rosing, "Application-drive method and apparatus for limiting power consumption in a processor-controlled hardware platform," US 7272730, Priority 2003-07-31.
7. Tajana S Rosing, Ozgur Celebican, "Arrangement and method for estimating and optimizing energy consumption of a system including I/O devices," WO2005076166A1, Priority 2004-01-30
8. T. Simunic Rosing, "Device and method for identifying a communication interface that performs an operating parameter closer to a desired performance level than another communication interface performs the operating parameter," US7246181B2, Priority date 2004-09-14.
9. V. Deolalikar, T. Simunic: "Method and system for power control in wireless portable devices using wireless channel characteristics",US20050170801A1, Priority date 2004-01-30.
10. T. Simunic, A. Acquaviva, L. Benini, "Application-driven method and apparatus for limiting power consumption in a processor-controlled hardware platform," US7272730B1, Priority date 2003-07-31.
11. T. Simunic, N. Mehta, C. Crome : "Method and Device for Test Vector Analysis"; US6197605B1, Priority date 1996-04-10.

#### THESES

1. B. Khaleghi, "Hardware-Algorithm Co-design for Efficient and Privacy-Preserved Edge Computing," 2022.
2. U. Mallappa, "AI for Design Optimization and Design for AI Acceleration," 2022.
3. K. Ergun, "Energy-Efficient and Reliability-Driven Management of IoT Systems," 2022.
4. J. Morris, "Fast, Efficient, and Robust Learning with Brain-Inspired Hyperdimensional Computing," 2022.
5. S. Gupta, "Efficient and Secure Learning across Memory Hierarchy," 2021.
6. S. Salamat, "Fast and Energy Efficient Big Data Processing on FPGAs," 2021.
7. M. Imani, "Machine Learning in IoT Systems: From Deep Learning to Hyperdimensional Computing," 2020.
8. Y. Kim, "Efficient Learning in Heterogeneous Internet of Things Ecosystems," 2020.
9. Y. Guo, "Efficient Learning across Multiple Domains with Deep Neural Networks," 2020.
10. J. Sim, "Architecting Non-volatile Memory for High Bandwidth Systems," 2019.
11. D. Peroni, "Approximate Computing for GPGPU Acceleration," PhD, 2019.
12. C. Chan, "Context-aware Platform Design and Optimization," PhD, 2017.
13. A. S. Akyurek, "Optimized Energy Control in Power Distribution Systems," PhD, 2017.
14. P. Mercati, "Power, Thermal, Reliability and Variability Management of Mobile Devices," PhD, 2016.
15. J. Venkatesh, "A Context-aware Approach for Automation of End-User Elements in the Smart Grid," PhD, 2016.
16. J. Yang, "Energy Efficient Data Aggregation in Sensor Networks," PhD, 2015.

17. B. Aksanli, "Energy and Cost Efficient Datacenters," PhD, 2015.
18. R. Strong, "Low-Latency Techniques for Improving System Energy Efficiency," PhD, 2013.
19. V. Kontorinis, "Adaptive Architectures for Peak Power Management," PhD, 2013.
20. S. Sharifi, "Accurate Temperature Sensing and Efficient Dynamic Thermal Management in MPSoCs," PhD, 2011.
21. R. Ayoub, "Temperature and Cooling Management in Computing Systems," PhD, 2011.
22. G. Dhiman, "Dynamic Workload Characterization for Energy Efficient Computing," PhD, 2011.
23. R. Herrmann, "Context based energy management for sensing applications," MS, 2011.
24. A. K. Coskun, "Efficient Thermal Management for Multiprocessor Systems," PhD 2009.
25. E. Regini, "Resource management in heterogeneous wireless sensor networks," MS 2009.
26. J. Steck, "Energy and task management in energy harvesting wireless sensor networks for structural health monitoring," MS 2009.
27. C. Bergonzini, "Management of solar harvested energy in actuation based embedded systems," MS 2009.
28. D. Lim, "Distributed proxy-layer scheduling in heterogeneous wireless networks," MS 2007.
29. D. Musiani, "Design of an active sensing platform for wireless structural health monitoring," MS 2007.
30. T. Simunic, "Energy efficient system design and utilization," PhD 2001.
31. T. Simunic, "VLSI interconnect design automation using qualitative and quantitative techniques," MS 1993.

#### ACADEMIC COMMUNITY SERVICE

- Invited to Dagstuhl Seminar on Power and Energy-aware Computing on Heterogeneous Systems, 2022
- Keynote at iTherm, 2022
- Keynote at HPCA-CogArch 2022
- Invited to participate in National AI Research Resource Task Force, August 2021.
- IEEE Fellow Award Committee, 2021.
- DARPA Electronics Resurgence Initiative - Invited speaker & invited demo, 2020, 2021
- NeurIPS reviewer 2021
- Invited talk at Annual Symposium of Academy of Neuroscience & Architecture on Quantified Buildings, Quantified Self, 2021
- Keynote at IGCC, 2021
- Keynote at IEEE Sensors, 2021
- IEEE CEDA Kuh Early Career Award Committee 2018, 2019, 2020
- Distinguished speaker UC Riverside, 2019.
- Distinguished speaker UC Irvine 2019
- TPC track chair, DATE, 2019-2020
- DAC women in EDA invited panel speaker, 2019
- JUMP CBRIC invited speaker, 2019
- JUMP ADA Invited speaker, 2019
- DARPA ERI Invited speaker, 2019
- ICCAD TPC member, 2019
- DATE women in EDA invited panel speaker 2018
- Keynote speaker at IEEE/ACM Workshop on Variability Modeling and Characterization, 2018
- Keynote speaker at IEEE Reliability Symposium 2018
- GLOBECOMM 2015 Executive committee member, Tutorials Chair
- Invited speaker at WIC Panel, 2015.
- ISCC'2013 chair of the TPC executive committee
- DATE 2011-2012, 2012-2013 TPC Track Chair
- ISLPED 2012 TPC Track Chair
- DAC 2013 WIP Chair, ESS Chair
- Associate Editor for IEEE Transactions on Mobile Computing 2008-2012
- Associate Editor for IEEE Transactions on Circuits and Systems 2003-2005
- Invited speaker at WEED-ESSA panel on "Cross-stack Energy-optimization - Fact or Fiction?" on June 9th at ISCA, Portland.

## Tajana Šimunić Rosing

tajana@ucsd.edu

<http://www.cse.ucsd.edu/~trosing/>

( 858 ) 534-4868

- CRA-W workshop presentations
- Session chair for DATE, ISLPED, DAC, ICCAD
- Technical paper committee for many conferences, such as DAC, DATE, ASPLOS, IPSN, ICCAD, ISLPED, ISCA, MMCN, HotPower, SECON.
- Reviewer for a number of publications ranging from Proceedings of the IEEE, to IEEE Transactions on Computers, IEEE Transactions on VLSI, IEEE Transactions on CAD, IEEE Transactions on Mobile Computing, IEEE Computer, IEEE Transactions on Computers, IEEE Micro, ACM TECS, ACM TODAES, ACM TOSN, and conferences such as DATE, DAC, ISLPED, ICCAD, IPSN and many others
- Technical reviewer for Alfred Sloan Grant & Dutch Ministry of Economic Affairs, Estonian NSF

### UNIVERSITY SERVICE

- Committee on Committees, 2019-2022.
- JSOE Dean's Faculty Council, 2019-2022.
- Ad-hoc committees for endowed chair appointments, 2014-2022
- Ad-hoc committees within CSE, 2014-2022
- Invited speaker at CS PhD Information Session, 2021
- CSE Diversity Coordinator 2019-2020
- SP-SOC 2018-2019
- LPSOE recruiting committee, 2017-2019
- HDSI Faculty Member of Clusters on Data Science Theory, Methods and Tools; Cross-cutting areas and systems, Improving Quality of Life, and Enabling Scientific Discovery 2018-pres.
- Dean Pisano Performance Review Committee CSE Representative 2018
- Ad-hoc committee for two endowed chair appointments in SOE, 2018.
- LPSOE recruiting committee, 2017-2019
- SP-SOC committee member 2018-pres.
- Co-creator of Jean Ferrante diversity scholarship (with Arun Kumar) 2018
- School of Engineering Building Committee 2015-2018
- UCSD Undergraduate Council Member 2016-2018
- Precision Medicine SOE recruit committee, 2017-2018
- Chair of Undergraduate Council's Department of Math Review Committee, 2017.
- Chair of Undergraduate Council's Department of Religion Review Committee, 2018.
- Director for \$10M UCSD-IBM AI for Healthy Living Center, 2017-2018.
- Diversity Coordinator for CSE Department. 2016 - Present
- IDEA Center Board Member, 2017-pres.
- University Faculty Recruitments for Sensors, Devices, and Imaging Committee, 2017-18.
- Mental Health & Technology Center executive board faculty member, 2017-pres.
- Qualcomm Institute (CalIT2) ORU Review, 2017.
- Featured Speaker at the UCSD's Founder's Symposium, "An Evening of Nonconventional Wisdom: AI for Healthy Aging," 2017.
- Teaching faculty recruit committee, 2017-pres.
- Board member, IDEA Center, 2017-pres.
- Diversity coordinator for CSE Department, 2017-pres.
- Dean review committee 2018
- Triton Drone Racing club faculty advisor 2016-2018
- UCSD Undergraduate Council Member 2016-2018
- Involved in 10 UCSD Centers: Center for Contextual Robotics, Center for Wearable Sensors, Center for Energy Research, Center for Networked Systems, Center for Wireless and Population Health Systems, Sustainable Power and Energy Center, San Diego Supercomputing Center, Qualcomm Institute
- University Diversity in Precision Medicine Faculty Recruitments for Sensors, Devices, and Imaging Committee, 2015-16
- Eastern Europe Outreach 2015-2016

- School of Engineering Building Committee 2015-2016
- MS comprehensive exam planning committee 2015
- MS project review 2015, 2016
- CSE PhD admission committee 2014-2016
- MAS AESE Program Review 2014-2015
- Graduate students' committee 2012-2014
- Executive board member of San Diego Supercomputing Center 2008-pres.
- University energy initiatives committee, 2012-2013
- JSOE energy faculty search committee, 2012-2013
- Graduate students committee (gradcom) 2012-2013
- Member of the search committee for SDSC director 2009-pres.
- Engineering Wide Initiatives Committee, 2008-09
- Faculty Advisor, Women in Computing, 2005 – pres.
- Calit2 System-on-a-Chip Committee, Chair, 2005 – 06
- Faculty Recruiting Committee, 2005 – 06
- Computer Engineering Program Committee, 2005 – 08
- Masters Students Admissions and Affairs Committee, 2005 – 09
- Computer Engineering Space Committee, Chair, 2005 – 09

#### TEACHING EXPERIENCE

Fall 05 – pres.

##### UCSD – Full professor

- Taught an undergraduate class in logic circuit design (CSE 140); 95% of students said they recommend me as an instructor; 2005-pres.
- Designed, got funding and set up a new graduate level class and lab in embedded systems (CSE237a); 98% of students said they recommend me as instructor for the class; 2005-pres.
- Developed a graduate course on SmartGrid that attracted students, researchers and faculty across CSE, ECE, mechanical engineering, nano engineering and structural engineering departments, 2015.
- Designed and taught a course on Internet of Things, 2016.
- Designed and taught courses on Emerging Computing, and SW for Acceleration, 2018.
- Designed and taught an embedded systems class that is a part of Master of Advanced Studies program in Wireless and Embedded Systems at UCSD
- Taught a course on Hardware acceleration for bioinformatics workloads, 2019
- Designed and taught a course on Hypedimensional computing, 2022.

Winter 02

##### STANFORD UNIVERSITY – lecturer

- taught a graduate course on Logic Synthesis of VLSI Circuits; lead a team of TAs and graders

#### HONORS & AWARDS

- **SIA-SRC University Researcher Award** 2022, the first woman in history to receive one. This award recognizes lifetime research contributions to the U.S. semiconductor industry.
- **ACM Fellow**, 2022
- **Intel's 2021 Outstanding Researcher Award**, received in 2022
- **IEEE Fellow**, 2018
- Awarded John J. and Susan M. **Fratamico Endowed Chair** in CSE Department, 2014.
- Keynote at IEEE International High-Level Design Validation and Test, "Reliability and Maintainability of IoT systems," 2017.
- UCSD Sustainability Award for the Postdoc in my group, 2016.
- FISP Award 2015.
- UCSD Research Expo Best poster awards, 2010, 2012 (two honorable mention), 2013 (two honorable mention).

- Von Liebig Entrepreneurism Center Innovation Award in Information Technology for “SOPRA – A Proactive Service Oriented Self-Adaptive Framework for Data Center Resource Optimization,” 2013
- CitiSense project covered in the NY Times and the Wall Street Journal, December 2012
- TODAES journal paper is the top most downloaded paper in 2010-2011
- Publication selected for inclusion in in *The Most Influential Papers of 10 Years DATE*, Edited by Lauwereins, Rudy; Madsen, Jan, 2008.
- Nominated as one of MIT’s top 100 researchers in 2002
- NSF Design and Manufacturing Grantee and SRC Research Assistantship 1993
- Lowell’s Award for the Best Student in Science at the Northern Arizona University 1992
- NASA Undergraduate Research Fellowship 1991

### SELECT FUNDING (as PI, Co-PI, senior contributor, student support)

1. DARPA & SRC JUMP 2.0 PRISM center, \$50.5M, 2023-2028, PI and Center Director; 20 total PIs
2. DARPA & SRC JUMP 2.0 Cognitive computing center, \$40M, 2023-2028. Sole PI at UCSD, 20 total PIs
3. DARPA DPRIVE subcontract, \$12.3M, 2022-2023
4. NSF MLWiNS, \$775k, 2020-2023
5. NSF Lifelong learning with HD Computing, \$800k, 2023-2028
6. Qualcomm HDnn gift, \$75k, 2021
7. NSF AI TILOS Institute, \$20M, 2021-2026
8. NSF Center for Power Management, \$375k to UCSD, 2021-2023
9. NSF CCRI: ENS: CHASE-CI, \$1.8M, 2021-24
10. NSF CCRI: ABR: Cognitive Hardware and Software Ecosystem Community Infrastructure, \$1M, 2021-23
11. NSF Prototype National Research Platform, \$6M, 2021-26
12. NSF CC\*NPEO REU, \$16k, 2021-22
13. NSF REU NRI, \$14k, 2021-22
14. NSF FET HD REU, \$16k, 2021-22
15. NSF MLWiNS REU, \$16k, 2021-22
16. NSF CNS Nautilus, \$1.8M, 2021-2024
17. SRC HD, \$240k, 2020-2022
18. SRC HDnn, \$270k, 2020-2022
19. SRC FHE-PIM, \$255k, 2020-2022
20. TSMC HD chip, \$300k, 2020-2022
21. NSF RAPID Accelerating COVID-19 Analysis in HW, \$200k, 2020-2021
22. NSF REU, \$16k, 2020-2021
23. MICS Center funds, \$17k, 2020
24. Qualcomm gift for HD computing, \$150k, 2020 & 2021
25. NSF RAID COVID-19 acceleration, \$300k
26. DARPA HyDREA, HD computing, \$1M, 2020-2021
27. GRC HD computing \$240k, 2019-2022
28. NSF FET HD computing \$500k, 2019-2023
29. Russel-Sage Foundation \$175k, 2018-2020
30. Kavli Institute for Brain and Mind Innovative, \$50k, 2019-2020
31. NSF CC-NPEO, \$2.5M, 2018-2021
32. NSF NRI \$2.5M, 2018-2020
33. DARPA/SRC JUMP CRISP \$40M, 2018-2023
34. KACST IoT, \$2.9M, 2017-2019
35. GRC IoT Reliability, \$240k, 2018-2021
36. IBM-UCSD AIHL, \$16.3M, 2017-2022
37. NSF CPS CHASE-CI, \$1M, 2017-2020
38. Samsung IoT, \$300k, 2018
39. Huawei, \$100k, 2018
40. Intel, \$300k, 2016-2018.
41. ARPA-E NODES \$2.5M, 2016-2018.

42. NSF CSR \$450k
43. NSF MetaSense \$1.126M
44. Qualcomm FMA, Energy management of residences in the grid, \$75k, 2014-2015.
45. CEC EISG, Energy management in data centers, \$95k, 2013-2014.
46. NSF CSR, Energy efficient data centers, \$300k, 2013-2016.
47. FCRP center “TerraSwarm,” \$27.5M, 2013-2017
48. NSF SCH: EXP SenseHealth, \$618k, 2013-2016
49. NSF MRI visualization \$2M, 2013-2016
50. Oracle gift, \$62k, 2012
51. Google gift, \$57k, 2012 CSE
52. NSF MRI optical networking \$1.06M, 2012-2014
53. NSF OCI:Sensor-Rocks, \$274k, 2012-2014.
54. NSF CCF, \$450k, 2012-2015.
55. Futurewei gift, \$117k, 2012.
56. Panasonic gift, \$100k, 2012.
57. Qualcomm FMA, \$75k, 2012-2014.
58. SRC, \$150k, 2011-2014.
59. Qualcomm membership, \$150k, 2011.
60. Oracle gift, \$100k, 2011. CNS
61. Google gift, \$75k, 2011. CNS
62. Google, \$125k, 2010. CSE
63. Qualcomm gift, \$35k, 2010.
64. Microsoft gift, \$300k, 2010
65. NSF-ERC CIAN, \$14M total, 2010-2019
66. NSF-Expedition on Variability, \$4.02M, 2010-2015
67. NSF OCI INRC: TransLight/StarLight, \$2.05M, 2010-2014
68. Qualcomm gift, \$75k, 2010-2011
69. CNS, \$75k, 2010.
70. NSF-CPS, \$1.5M, 2009-2013
71. Sun Microsystems Gift, \$120k, 2009
72. Google, \$50k, 2009.
73. Qualcomm Gift, \$15k, 2009
74. NIH PALMS, \$3.2M, 2007-2011
75. MARCO-MuSyC center, \$3M, 2009-2012
76. NSF CCF ARRA, \$476k, 2009-2013
77. NSF GreenLight, \$20M, 2008-2012
78. NSF FlashGordon, \$22M, 2011-2015
79. Cisco Gift, \$80k, 2008-2009
80. MARCO-GSRC Grant, \$154k, 2008-2009
81. CNS Grant for thermal management, \$142k, 2008-2010
82. CNS Grant for healthcare, \$116k, 2008-2010
83. Xilinx gift of 20 XUP DVKs
84. UC Micro Grant, \$30k, 2008-2010
85. Sun Microsystems Gift, \$100k, 2008-2009
86. NSF-CCF \$600k, sensing, 2007-2011
87. Sun Microsystems Gift, \$60k, 2007
88. CNS Grant, \$130k, 2006-2008
89. NSF – HPWREN, \$3M, 2005-2009.
90. LANL Structural Health Monitoring, \$7M, 2005-2009.
91. UC Micro, \$30k, 2006.
92. Sun Microsystems Gift, \$50k, 2005.
93. CNS Grant, \$60k, 2005.
94. Intel Grant, \$200k, 2005.

95. UC Graduate and Travel Grant \$11k, 2005
96. HP Labs - \$100k, 2003-2004

#### RECENT INVITED TALKS

1. SRC Board Invited Speaker, 2022.
2. Keynote at iTherm, 2022
3. Intel Research, OR, "Lifelong learning with HD computing," 2022
4. Invited talk at DATE'22 special session on "Interpretable AI and Nanoelectronics-Based Designs of edge computing systems in the IoT 2.0 Era." Title: "Hyperdimensional computing and applications," 2022
5. Invited talk at ISCA-CogArch on "Accelerating fully homomorphic encryption in memory," 2022
6. Micron Inc. executive leadership committee invited speaker 2022
7. SRC executive committee invited speaker 2022
8. Intel internal workshop, CA, "HD computing and applications," 2021
9. Terradata, CA, "Big data acceleration," 2021
10. Merck, Germany, "Brain-inspired computing using high dimensionality," 2021
11. Raytheon, DC, "Accelerating big data processing in hardware," 2021
12. Xilinx, CA, "Accelerating COVID-19 analysis with FPGAs," 2021
13. Keynote at Annual Symposium of Academy of Neuroscience & Architecture on Quantified Buildings, Quantified Self, (joint with Cognitive Sciences faculty from UCSD) 2021
14. Keynote at IGCC, 2021
15. Keynote at IEEE Sensors, 2021
16. Invited plenary speaker at the Future Chips Forum, Tsinghua University, 2021.
17. Invited to participate in National AI Research Resource Task Force, 2021.
18. DARPA Electronics Resurgence Initiative - Invited speaker & invited demo, 2020, 2021
19. Raytheon, DC, "Accelerating Fully Homomorphic Encryption with Processing in Memory" 2021.
20. Google, CA, "Accelerating big data analysis with in and near memory computing," 2021.
21. Intel Research, OR, "Symbolic reasoning with HD computing," 2021
22. Facebook, CA, "Brain-inspired HD computing," 2021
23. IBM Research, NY, "Acceleration of big data workloads using in memory and in storage computing," 2021
24. Samsung, NV at CES, "Hyperdimensional computing," 2020.
25. MEC/DARPA Invited talk at "Analog Feature Extraction; Rethinking Analog Front-End to Accelerate Digital Inference," Title: "Hyperdimensional computing and its acceleration," 2020.
26. Micron, CA, "Accelerating bioinformatics workloads," 2020.
27. IBM, NY, "Accelerating machine learning & HD computing," 2020.
28. Qualcomm, CA, "Accelerating machine learning & HD computing," 2020.
29. Northrop-Grumman, CA, "Accelerating machine learning & HD computing," 2020.
30. Merck, Germany, "Accelerating workloads using in memory computing," 2020.
31. Merck, Germany, "Acceleration of COVID-19 pipeline," 2020.
32. Intel Research, CA, "Acceleration of COVID-19 pipeline," 2020.
33. Leidos, CA, "HD Computing and its acceleration," 2020.
34. ARM Inc, England, "HD computing," 2020.
35. Intel Research, OR, "Accelerating big data in storage," 2020.
36. EMD, CA, "Accelerating big data in HW," 2020.
37. NXP, CA, "HD computing," 2020.
38. DARPA ERI, DC, invited talk, "Accelerating big data in memory and storage," 2020.
39. LLNL, CA, "Accelerating bioinformatics workloads," 2020
40. Xilinx Inc, CA, "Accelerating bioinformatics workloads on FPGAs," 2020
41. Intel Research, OR, " Computational storage," 2020

42. UCSD TV, “Research Opportunities and Partnerships in the Tech Industry: Industry Panelists Share insights at the CSE Winter 2020 Research Open House,” 2020.
43. VLSI-SOC invited keynote, “Acceleration of big data workloads,” 2020.
44. Intel Research, OR, “Hyperdimensional computing,” 2020
45. University of Wisconsin, WI, distinguished speaker “Accelerating big data workloads,” 2020
46. Invited talk at NSF NDA Panel on AI/ML/Brain-inspired hardware design, 2020
47. Sony, CA, “Accelerating machine learning & HD computing using PIM,” 2019.
48. ARM, CA, “Power, performance, thermal & reliability modeling and management in IoT systems,” 2019.
49. Intel, OR, “Power, performance, thermal & reliability modeling and management in IoT systems,” 2019.
50. EPFL, Switzerland, “Context-aware learning and acceleration,” 2019.
51. Yahoo, CA, ““Context-aware learning and acceleration,” 2019.
52. Workshop on Brain-Inspired Architectures, NV, “Hyperdimensional Computing & Applications,” 2019.
53. Non-volatile Memory Workshop, “Hyperdimensional Computing and Its Applications,” 2019.
54. Altera, CA, “Accelerating Bioinformatics Workloads,” 2019.
55. IBM Research, CA, “Accelerating machine learning & HD computing using PIM,” 2019.
56. Huawei, CA, “Thermal management in mobiles,” 2018.
57. Samsung, CA, “Context-aware management in Smart Homes,” 2018.
58. IBM Research in Austin, TX, “Context-awareness for healthy aging,” 2018.
59. Xconomy, CA, “Big data meets big biology: Accelerating learning for healthy living,” 2018.
60. Huawei, China, “Proactive power and thermal management strategies,” 2018.
61. Samsung, CA, “Smart Homes: context-aware management,” 2018.
62. Intel Research, OR, “Reliability Management for IoT Systems,” 2018.
63. DARPA Electronic Resurgence Initiative, CA, “JUMP CRISP Center Overview,” 2018.
64. Micron, CA, “Accelerating machine learning workloads using PIM,” 2018.
65. JUMP C-BRIC Center, USA, “Accelerating machine learning & HD computing using PIM,” 2018.
66. JUMP ADA Center, USA, “Accelerating machine learning & HD computing using PIM,” 2018.
67. TSMC, Taiwan, “Accelerating machine learning & HD computing using PIM,” 2018.
68. GM, USA, “Context-aware management for IoT systems,” 2018.
69. China government delegation, CA, “Context-aware learning and acceleration,” 2018.
70. NSF CSR Workshop, WA, “Accelerating learning using PIM,” 2018.
71. HLTV Keynote, “Reliability and Maintainability of IoT systems,” 2017.
72. IBM, Austin, TX, “Context-aware IoT Systems,” 2017.
73. Sony, Japan, “Context-aware IoT Systems” & “UCSD-IBM AI for Healthy Living Center”, 2017.
74. D-Link, Japan, “UCSD-IBM AI for Healthy Living Center”, 2017.
75. A-Star, Singapore, “UCSD-IBM AI for Healthy Living Center”, 2017.
76. NRF, Singapore, “UCSD-IBM AI for Healthy Living Center”, 2017.
77. NUS, Singapore, “UCSD-IBM AI for Healthy Living Center”, 2017.
78. IBM, Kawasaki, Japan, “Context-aware IoT Systems, their Acceleration & Management,” 2017.
79. Sony, Tokyo, Japan, “Context-aware IoT Systems, their Acceleration & Management,” 2017.
80. IEEE/ACM Workshop on Variability Modeling and Characterization, Irvine, CA, “Increasing computational efficiency with novel computing paradigm,” <http://www.cerc.utexas.edu/utda/vmc/>, 2017.
81. University of Melbourne, Australia, “Context-aware management for Smart Cities,” 2016.
82. San Diego Port Authority, CA, “Context-aware management for Smart Cities,” 2016.
83. CalTrans, CA, “Context-aware management for Smart Cities,” 2016.
84. Cymer, CA, “System Energy Efficiency,” 2016.
85. Hitachi, CA, “System Energy Efficiency,” 2016.
86. Hewlett-Packard, CA, “System Energy Efficiency for IoT Applications,” 2016.
87. IBM, CA, “Context-awareness for healthcare applications,” 2016.
88. SDGE, CA, “Sensors to Grid,” 2016.