(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
03 March 2022 (03.03.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/040863 A1

(51) International Patent Classification:
H05K 7/20 (2006.01)

(21) International Application Number:
PCT/CN2020/110811

(22) International Filing Date:
24 August 2020 (24.08.2020)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **NVIDIA CORPORATION** [US/US]; 2788 San Tomas Expressway, Santa Clara, CA 95051 (US).

(72) Inventor; and
(71) Applicant *(for BH, KW only)*: **CAI, Xiaozhuo** [CN/US]; 2788 San Tomas Expressway, Santa Clara, CA 95051 (US).

(72) Inventors: **YAN, Shuanghu**; 2788 San Tomas Expressway, Santa Clara, CA 95051 (US). **YI, Dinghai**; 2788 San Tomas Expressway, Santa Clara, CA 95051 (US). **ZHU, Hao**; 2788 San Tomas Expressway, Santa Clara, CA 95051 (US). **FU, Yonghua**; 2788 San Tomas Expressway, Santa Clara, CA 95051 (US).

(74) Agent: **P.C. & ASSOCIATES**; 9th Floor Scitech Tower, No. 22 Jian Guo Men Wai Avenue, Chaoyang District, Beijing 100022 (CN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,

(54) **Title:** INTELLIGENT ADAPTABLE FINS FOR COOLING DATACENTER DEVICES

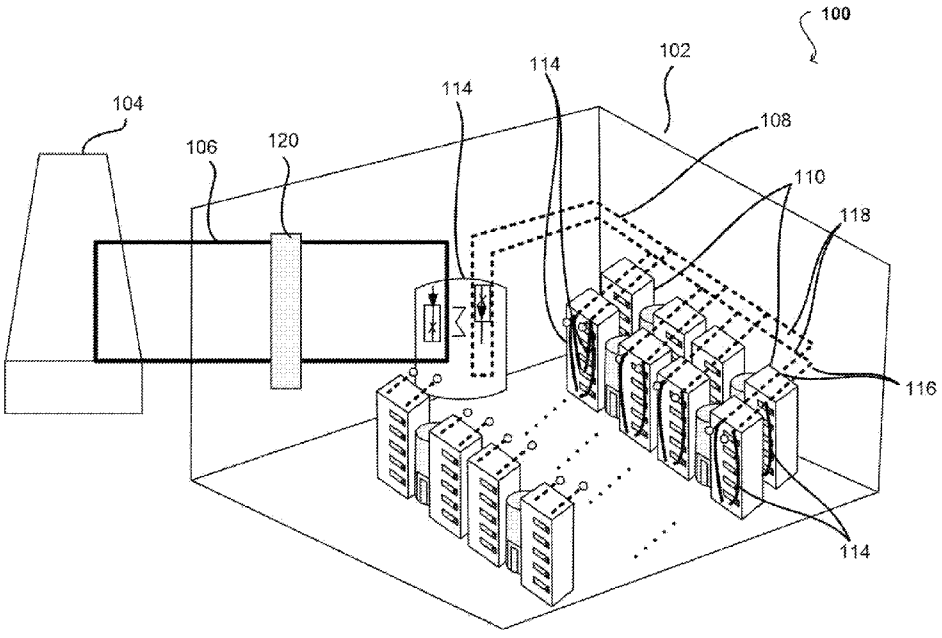

FIG. 1

(57) **Abstract:** A cooling system for a datacenter device is disclosed. Fins (216) are provided between a first plate (210) and a second plate (208) to dissipate a first amount of heat to an environment in a first configuration of the fins (216). The first plate (210) is movable relative to the second plate (208) to expose a surface area of the fins (216) to the environment in a second configuration of the fins (216).

WO 2022/040863 A1

[Continued on next page]

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

INTELLIGENT ADAPTABLE FINS FOR COOLING DATACENTER DEVICES

FIELD

[0001] At least one embodiment pertains to cooling systems for a datacenter device. In at least one embodiment, fins are provided between a first plate and a second plate of a heat sink to dissipate a first amount of heat to an environment in a first configuration of the fins and the first plate is movable relative to the second plate to expose a surface area of the fins to the environment in a second configuration of the fins.

BACKGROUND

[0002] Datacenter cooling systems typically use fans to circulate air through server components. Certain supercomputers or other high capacity computers may use water or other cooling systems than air cooling systems to draw heat away from the server components or racks of the datacenter to an area external to the datacenter. The cooling systems may include a chiller within the datacenter area, including the area external to the datacenter. The area external to the datacenter may be an area including a cooling tower or other external heat exchanger that receives heated coolant from the datacenter and disperses the heat by forced air or other means to the environment (or an external cooling medium) before the cooled coolant is recirculated back into the datacenter. In an example, the chiller and the cooling tower together form a chilling facility with pumps responsive to temperature measured by external devices applied to the datacenter. Air cooling systems alone may not draw sufficient heat to support effective or efficient cooling in datacenters and liquid cooling systems may not be economical for the demands of the datacenter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] Various embodiments in accordance with the present disclosure will be described with reference to the drawings, in which:

[0004] FIG. 1 is a block diagram of an example datacenter having a cooling system subject to improvements described in at least one embodiment;

[0005] FIG. 2A is a block diagram illustrating mobile datacenter features of a cooling system incorporating adaptable fins in a first configuration of the fins, according to at least one embodiment;

[0006] FIG. 2B is a block diagram illustrating a cooling system incorporating adaptable fins in a second configuration of the fins, according to at least one embodiment;

[0007] FIGS. 3A, 3B, and 3C illustrate aspects of adaptable fins, according to at least one embodiment;

[0008] FIGS. 4A, 4B, and 4C illustrate processor-supported subsystems to enable a cooling system incorporating adaptable fins, according to at least one embodiment;

[0009] FIGS. 4D and 4E illustrate plan views of a plate incorporating one or more processor-supported subsystems to enable a cooling system incorporating adaptable fins, according to at least one embodiment;

[0010] FIG. 5 is a process flow of steps available for a method of using or making the cooling system of FIGS. 2A-4E and 6A-17D, according to at least one embodiment;

[0011] FIG. 6A illustrates an example datacenter, in which at least one embodiment from FIGS. 2A-5 may be used;

[0012] FIGS. 6B, 6C illustrate inference and/or training logic, such as used in FIG. 6A and in at least one embodiment of the present disclosure, for enabling and/or supporting a cooling system incorporating adaptable fins, according to various embodiments;

[0013] FIG. 7A is a block diagram illustrating an exemplary computer system, which may be a system with interconnected devices and components, a system-on-a-chip (SOC) or some combination thereof formed with a processor that may include execution units to execute an instruction to support and/or to enable a cooling system incorporating adaptable fins as described herein, according to at least one embodiment;

[0014] FIG. 7B is a block diagram illustrating an electronic device for utilizing a processor to support and/or to enable a cooling system incorporating adaptable fins, according to at least one embodiment;

[0015] FIG. 7C is a block diagram illustrating an electronic device for utilizing a processor to support and/or to enable a cooling system incorporating adaptable fins, according to at least one embodiment;

[0016] FIG. 8 illustrates a further example computer system, according to at least one

embodiment, to implement various processes and methods for a cooling system incorporating adaptable fins as described throughout this disclosure;

[0017] FIG. 9A illustrates an exemplary architecture in which GPUs are communicatively coupled to multi-core processors over high-speed links for enabling and/or supporting a cooling system incorporating adaptable fins, according to at least one embodiment of the disclosure herein;

[0018] FIG. 9B illustrates additional details for an interconnection between a multi-core processor and a graphics acceleration module in accordance with one exemplary embodiment;

[0019] FIG. 9C illustrates another exemplary embodiment in which accelerator integration circuit is integrated within a processor for enabling and/or supporting a cooling system incorporating adaptable fins, according to at least one embodiment of the disclosure herein;

[0020] FIG. 9D illustrates an exemplary accelerator integration slice 990 for enabling and/or supporting a cooling system incorporating adaptable fins, according to at least one embodiment of the disclosure herein;

[0021] FIG. 9E illustrates additional details for one exemplary embodiment of a shared model. to enable and/or support a cooling system incorporating adaptable fins, according to at least one embodiment of the disclosure herein;

[0022] FIG. 9F illustrates additional details for one exemplary embodiment of a unified memory, addressable via a common virtual memory address space used to access physical processor memories and GPU memories to enable and/or support a cooling system incorporating adaptable fins, according to at least one embodiment of the disclosure herein;

[0023] FIG. 10A illustrates exemplary integrated circuits and associated graphics processors, according to embodiments described herein for a cooling system incorporating adaptable fins;

[0024] FIGS. 10B-10C illustrate exemplary integrated circuits and associated graphics processors, according to at least one embodiment, to support and/or to enable a cooling system incorporating adaptable fins;

[0025] FIGS. 10D-10E illustrate additional exemplary graphics processor logic, according to at least one embodiment, to support and/or to enable a cooling system incorporating adaptable fins;

[0026] FIG. 11A is a block diagram illustrating a computing system to support and/or to enable a cooling system incorporating adaptable fins, according to at least one embodiment;

[0027] FIG. 11B illustrates a parallel processor to support and/or to enable a cooling system incorporating adaptable fins according to at least one embodiment;

[0028] FIG. 11C is a block diagram of a partition unit according to at least one embodiment;

[0029] FIG. 11D shows a graphics multiprocessor used for a cooling system incorporating adaptable fins according to at least one embodiment;

[0030] FIG. 11E shows a graphics multiprocessor according to at least one embodiment;

[0031] FIG. 12A illustrates a multi-GPU computing system, according to at least one embodiment;

[0032] FIG. 12B is a block diagram of a graphics processor, according to at least one embodiment;

[0033] FIG. 13 is a block diagram illustrating micro-architecture for a processor that may include logic circuits to perform instructions, according to at least one embodiment;

[0034] FIG. 14 illustrates a deep learning application processor, according to at least one embodiment;

[0035] FIG. 15 is a block diagram of a neuromorphic processor, according to at least one embodiment;

[0036] FIG. 16A is a block diagram of a processing system, according to at least one embodiment;

[0037] FIG. 16B is a block diagram of a processor having one or more processor cores, an integrated memory controller, and an integrated graphics processor, according to at least one embodiment;

[0038] FIG. 16C is a block diagram of hardware logic of a graphics processor core, according to at least one embodiment;

[0039] FIGS. 16D-16E illustrate thread execution logic including an array of processing elements of a graphics processor core according to at least one embodiment;

[0040] FIG. 17A illustrates a parallel processing unit, according to at least one embodiment;

[0041] FIG. 17B illustrates a general processing cluster, according to at least one embodiment;

[0042] FIG. 17C illustrates a memory partition unit of a parallel processing unit, in accordance with at least one embodiment; and

[0043] FIG. 17D illustrates a streaming multi-processor, according to at least one embodiment.

DETAILED DESCRIPTION

[0044] Air cooling of high density servers may not be efficient or may be ineffective in view of sudden high heat requirements caused by changing computing-loads in present day computing components. However, as the requirements are subject to change or tend to range from a minimum to a maximum of different cooling requirements, these requirements must be met in an economical manner. The different cooling requirements also reflect different heat features of the datacenter. In at least one embodiment, heat generated from the components, servers, and racks are cumulatively referred to as a heat feature or a cooling requirement as the cooling requirement must address the heat feature entirely. In at least one embodiment, the heat feature or the cooling requirement for a cooling system is the heat generated or the cooling requirement of the components, servers, or racks associated with the cooling system and may be of a portion of components, servers, and racks in the datacenter.

[0045] In at least one embodiment, deployment of pre-fabricated (such as mobile) datacenters enable housing of datacenter information technology (IT) components within the pre-fabricated datacenters. The use of the pre-fabricated datacenters, in at least one aspect, reduces cost, expedites building, enables flexibility of relocation, and lowers operational costs for a datacenter to be deployed. Moreover, these benefits also promote reliability of operations. Further, as power density increases in the IT equipment, there are various components that feature higher heat generation as computing expectations increase from these components. These components are datacenter devices that may include a graphics processing unit (GPU), a central processing unit (CPU), storage components, storage boxes, switches, networking equipment, and auxiliary equipment. As these components are varied in design and structure, unique challenges exist in terms of cooling design for such IT equipment in pre-fabricated and in permanent datacenters. While air cooling through fins may provide limited dissipation of heat, at least one embodiment

herein provides a way to let the passive fan fins react between a closed (contracted or overlapping) first configuration and an expanded (or exposed) second configuration.

[0046] In at least one embodiment, for economy purposes, when the IT equipment indicates normal heat generation and sensors determine appropriate (and normal) heat dissipation via air cooling of fins, the fins remain contracted in the first configuration. In at least one embodiment, when the IT equipment indicates above normal heat generation, for sudden computation demands, and when sensors determine associated (and above normal) heat dissipation requirements via the air cooling of the fins, then the fins may be expanded in the second configuration to expose more surface area to the environment of an air cooling system. The exposed area is able to exchange or dissipate more heat to the environment than when the fins are in the contracted configuration. In at least one embodiment, the exposed surface area provides proportional heat dissipation. As such, there are intermediate configurations (amount of exposure changes) between the contracted configuration and the expanded configuration of the fins, and there are intermediate heat features (heat dissipation requirements) or cooling requirements that can be addressed in at least one embodiment. In at least one embodiment, each configuration of the fins enable a different cooling capabilities or capacities corresponding to different heat features (heat dissipation requirements) or cooling requirements for a cooling system incorporating the adaptable fins.

[0047] FIG. 1 is a block diagram of an example datacenter 100 having a cooling system subject to improvements described in at least one embodiment. The datacenter 100 may be one or more rooms 102 having racks 110 and auxiliary equipment to house one or more servers on one or more server trays. The datacenter 100 is supported by a cooling tower 104 located external to the datacenter 100. The cooling tower 104 dissipates heat from within the datacenter 100 by acting on a primary cooling loop 106. Further, a cooling distribution unit (CDU) 112 is used between the primary cooling loop 106 and a second or secondary cooling loop 108 to enable extraction of the heat from the second or secondary cooling loop 108 to the primary cooling loop 106. The secondary cooling loop 108 is able to access various plumbing all the way into the server tray as required, in an aspect. The loops 106, 108 are illustrated as line drawings, but a person of ordinary skill would recognize that one or more plumbing features may be used. In an instance, flexible polyvinyl chloride (PVC) pipes may be used along with associated

plumbing to move the fluid along in each of the loops 106, 108. One or more coolant pumps, in at least one embodiment, may be used to maintain pressure differences within the loops 106, 108 to enable the movement of the coolant according to temperature sensors in various locations, including in the room, in one or more racks 110, and/or in server boxes or server trays within the racks 110.

[0048] In at least one embodiment, the coolant in the primary cooling loop 106 and in the secondary cooling loop 108 may be at least water and an additive, for instance, glycol or propylene glycol. In operation, each of the primary and the secondary cooling loops has their own coolant. In an aspect, the coolant in the secondary cooling loops may be proprietary to requirements of the components in the server tray or racks 110. The CDU 112 is capable of sophisticated control of the coolants, independently or concurrently, in the loops 106, 108. For instance, the CDU may be adapted to control the flow rate so that the coolant(s) is appropriately distributed to extract heat generated within the racks 110. Further, more flexible tubing 114 is provided from the secondary cooling loop 108 to enter each server tray and to provide coolant to the electrical and/or computing components. In the present disclosure, the electrical and/or computing components are used interchangeably to refer to the heat-generating components that benefit from the present datacenter cooling system. The tubing 118 that form part of the secondary cooling loop 108 may be referred to as room manifolds. Separately, the tubing 116 extending from tubing 118 may also be part of the secondary cooling loop 108 but may be referred to as row manifolds. The tubing 114 enters the racks as part of the secondary cooling loop 108, but may be referred to as rack cooling manifold. Further, the row manifolds 116 extend to all racks along a row in the datacenter 100. The plumbing of the secondary cooling loop 108, including the manifolds 118, 116, and 114 may be improved by at least one embodiment of the present disclosure. A chiller 120 may be provided in the primary cooling loop within datacenter 102 to support cooling before the cooling tower. To the extent additional loops exist in the primary control loop, a person of ordinary skill would recognize reading the present disclosure that the additional loops provide cooling external to the rack and external to the secondary cooling loop; and may be taken together with the primary cooling loop for this disclosure.

[0049] In at least one embodiment, in operation, heat generated within server trays of the racks

110 may be transferred to a coolant exiting the racks 110 via flexible tubing of the row manifold 114 of the second cooling loop 108. Pertinently, second coolant (in the secondary cooling loop 108) from the CDU 112, for cooling the racks 110, moves towards the racks 110. The second coolant from the CDU 112 passes from on one side of the room manifold having tubing 118, to one side of the rack 110 via row manifold 116, and through one side of the server tray via tubing 114. Spent second coolant (or exiting second coolant carrying the heat from the computing components) exits out of another side of the server tray (such as enter left side of the rack and exits right side of the rack for the server tray after looping through the server tray or through components on the server tray). The spent second coolant that exits the server tray or the rack 110 comes out of different side (such as exiting side) of tubing 114 and moves to a parallel, but also exiting side of the row manifold 116. From the row manifold 116, the spent second coolant moves in a parallel portion of the room manifold 118 going in the opposite direction than the incoming second coolant (which may also be the renewed second coolant), and towards the CDU 112.

[0050] In at least one embodiment, the spent second coolant exchanges its heat with a primary coolant in the primary cooling loop 106 via the CDU 112. The spent second coolant is renewed (such as relatively cooled when compared to the temperature at the spent second coolant stage) and ready to be cycled back to through the second cooling loop 108 to the computing components. Various flow and temperature control features in the CDU 112 enable control of the heat exchanged from the spent second coolant or the flow of the second coolant in and out of the CDU 112. CDU 112 is also able to control a flow of the primary coolant in primary cooling loop 106.

[0051] In at least one embodiment, the economy of using an air-cooled system is improved with the present adaptable fines. As such, it is possible that some components within the servers and the racks that have cooling requirements between what may be afforded by air cooling systems and liquid cooling system cannot receive more than cooling capability or capacity offered by static fins of a heat sink in an air cooling system. Separately, such a problem may also exist in some liquid cooling systems using fins as a heat exchange surface contacting a cooling media, such as a coolant. In at least one embodiment of the present disclosure, adaptable fins increase the cooling capability or capacity of a respective cooling system, such as at least the air cooling system. In at least one embodiment, the adaptable fins also increase the cooling

capability or capacity of a liquid-cooling system and of a hybrid cooling system incorporating both air and liquid cooling systems.

[0052] FIG. 2A is a block diagram illustrating mobile datacenter features of a cooling system 200 incorporating adaptable fins 216 in a first configuration of the fins, according to at least one embodiment. In at least one embodiment, the cooling system 200 is for a datacenter device, such as a GPU, a CPU, storage components, storage boxes, switches, networking equipment, and auxiliary equipment. In at least one embodiment, the cooling system 200 includes a heat sink 202 having fins 216 between a first plate 210 and a second plate 208. In at least one embodiment, fins 216 may be two different fins associated together with a ribbon. In at least one embodiment, the fins 216 are individual unibody structures incorporating a flexible material throughout the fin. In at least one embodiment, only a second plate 208 is provided with fins 216 thereon, where the fins 216 incorporate a material such as a bimorph material that is capable of changing shape or structure upon application of heat. The fins 216 are bent over to incorporate at least an overlapping portion, which is addressed further in at least FIGS. 3A, 3B, and the related discussion.

[0053] In at least one embodiment, as illustrated in FIG. 2A, the fins 216 may be in a first configuration with the overlapping portion being so that only a first surface area is exposed to the environment. In at least one embodiment, FIG. 2A illustrates the bottom-most position 218 for the fins 216. The first surface area may be a default or primary surface area that is an outside portion of two sides of an individual fin of the fins 216. In at least one embodiment, the surface area exposed in the second configuration is more than the default or the primary surface area of the fins that were in the first configuration. In at least one embodiment, the surface area is a reference to the overall surface area of all the fins 216 at a given time. In at least one embodiment, individual fins of the fins 216 contribute to the first surface area with sides of all the fins simultaneously in a singular or uniform first configuration, such as the contracted configuration illustrated in FIG. 2A. In at least one embodiment, each fin has at least two inside surfaces that may be overlapping with a ribbon feature there between, where the ribbon feature forms intermediate surfaces on either of its sides and between the two inside surfaces. In at least one embodiment, when the fin is a unibody structure, a middle portion of the fin may have the intermediate surfaces between the inside surfaces of the unibody structure.

[0054] In at least one embodiment, while the inside surfaces may receive some air in the air cooled or air cooling system, the inside surfaces of the first configuration are not able to suitably exchange heat with the environment. In at least one embodiment, as a result, the fins 216 only dissipate a first amount of heat to the environment in the first configuration of the fins 216. In an example, the environment is in reference to air flowing through the fins 216 from an air cooling system. In at least one embodiment, the environment is in reference to the environment within a server box, a rack, or a datacenter. The environment may be still air that may be moved by a fan, in at least one embodiment. In at least one embodiment, the environment is coolant-driven, coolant, or refrigerant.

[0055] In at least one embodiment, the first plate 210 is movable relative to the second plate 208 in a singular relative direction. In at least one embodiment, a horizontal fin structure may be used where the first plate moves horizontally instead of vertically and the fins are horizontal fins. The goal in either case is to dissipate more heat to the environment when the fins are in an expanded configuration than in a contracted configuration. In at least one embodiment, the second plate 208 is affixed in a suitable manner to datacenter device 206 directly or via an intermediate cooling surface 204. In at least one embodiment, an interface between the heat sink 202 and the datacenter device 206 may be thermal grease that promotes heat transfer from the datacenter device 206 and the heat sink 202. In at least one embodiment, the thermal grease is a silver-based compound. In at least one embodiment, the intermediate cooling surface 204 may be a secondary cooling component in a hybrid cooling system.

[0056] In at least one embodiment, the intermediate cooling surface 204 may be a liquid cooling component that is adapted to allow flow of a coolant through pipelines 220A, 220B. In at least one embodiment, the movement of the first plate 210 relative to the second plate 208 causes the fins 216 to expand and causes the overlapping surfaces (also referred to as inside surfaces) of the individual fins 216 to become non-overlapping with the middle portion or the ribbon feature and with each other (indirectly, in at least one embodiment). In at least one embodiment, it is possible to construct the adaptable fins without the intermediate surface or ribbon feature so that the fin is entire bent and overlapping directly between a top portion and a bottom portion of the fin, having a hinge there between.

[0057] In at least one embodiment, the expansion of the fins 216 causes a surface area of the

fins, previously in the overlapping surfaces, to become exposed to the environment in a similar manner as outer surfaces on either side of the fins 216, illustrated in FIG. 2A. In at least one embodiment, the expansion of the fins 216 results in a second configuration of the fins 216. The surface area newly exposed and previously overlapping enables dissipation of additional heat from the fins to the environment. The additional heat (or a cumulative heat, along with the first amount of heat) is referred to as a second amount of heat that is more than the first amount of heat when the fins are in the first configuration.

[0058] In at least one embodiment, the fins 216 are assisted or unassisted in its change from the first configuration to the second configuration. In an assisted system, in at least one embodiment, a gear subsystem, an electromagnetic subsystem, a thermoelectric generator subsystem, a thermo-reactive subsystem, or a pneumatic subsystem is provided via motion feature 214 of the cooling system. In at least one embodiment, there may be more than one motion feature 214 throughout the second plate 208 to provide level movement of the first surface or to provide equal exposure of all the fins 216 simultaneously. The motion features, when more than one, are designed to address the equal exposure of all of the overlapping surfaces of the fins 216. In at least one embodiment, reference to a single fin of the fins 216 illustrated in FIG. 2A or the other figures herein is understood to apply equitably to all the fins 216 provided for a heat sink 202.

[0059] In at least one embodiment, when one motion feature 214 is provided, there may be one or more support features 212 to assist the motion feature 214. In at least one embodiment, the one or more support features 212 provide stability as the motion feature 214 acts on one area of the first plate 210 to raise the first plate 210. In at least one embodiment, the support features provide at least tension-based support to keep the first plate 210 stable across the four corners as it is raised relative to the second plate 208. In at least one embodiment, the tension-based support is provided via inner springs that are adapted to compress with the entire load of the first plate 210 asserted on the inner springs. In at least one embodiment, the inner springs 210 can, however, maintain position when supported by an upward tension applied to the first plate 210 to reduce the load from the first plate 210.

[0060] In at least one embodiment, in an unassisted system, the cooling system 200 may not incorporate the first plate 210. In at least one embodiment, cooling system 200 may not

incorporate the motion feature 214, and may or may not incorporate the support feature(s) 212. In at least one embodiment, if the support features 212 are provided on the second plate 210, the first plate 208 is provided as the support features 212 function to reduce the load of the second plate 210 on the fins 216, in at least one embodiment of the unassisted system. In at least one embodiment, the unassisted system is enabled by a material incorporated within the entirety of unibody of one or more of the fins 216 or in a ribbon feature of one or more fins 216. In at least one embodiment, the material is a bimorph material that is formed of at least two elements having different coefficients of expansion forming a bi (or multi) metallic strip. This enables the ribbon feature or a middle portion of the unibody structure of the one or more fins 216. In at least one embodiment, the bimorph material is caused to change shape or structure, such as from a bent shape or structure to a relatively straightened shape or structure. In at least one embodiment, heat from an associated datacenter component 206 causes the bimorph material to change shape. This change in the shape or the structure at least exposes the fin 216 to more environment air than in the bent position. In at least one embodiment, the assisted and unassisted system may be used together with the bimorph materials reducing a load on a motion feature or with the bimorph material caused to first act prior to the motion feature coming into play to expand the fins 216.

[0061] FIG. 2B is a block diagram illustrating a cooling system 250 incorporating adaptable fins in a second configuration of the fins 258, according to at least one embodiment. In at least one embodiment, a first plate 254 is illustrated in a top-most position 264 relative to a second plate 252 of a heat sink, such as the heat sink 202 of FIG. 2A. The fins 258 are in a second configuration relative to the fins 216 of FIG. 2A, which are illustrated in the first configuration. In at least one embodiment, previously bent middle portions or ribbon features 262 of the individual fins are relatively straighter than when the fins were in the first configuration, illustrated in FIG.2A. In at least one embodiment, the middle portions or ribbon features 262 may be straight or curved, to the extent that the top-most position 264 is relatively distinct from the bottom-most position 218.

[0062] In at least one embodiment, when the middle portion 262 is a hinge or in a hinge format, the middle portion 262 is not as substantial as illustrated in FIGS 2A, 2B, but the fins are able to have a surface area exposed to the environment, more so in a second (expanded and exposed) configuration than in a first (contracted and overlapping) configuration. In at least one

embodiment, the first and the second configurations may be functionally enabled by a capability or capacity of a set of fins to dissipate a first amount of heat that is different than a second amount of heat, respectively, in one or more cycles of a datacenter device generating a determined amount of heat. The same functionality may be extrapolated to intermediate configurations and to intermediate amounts of heat dissipated for the intermediate configurations.

[0063] In at least one embodiment, intermediate configurations of the fins between the first configuration and the second configuration may exist to provide intermediate surface areas to enable dissipation of third amounts of heat that is relative with respect to individual intermediate configurations of the fins 258 (or 216). In at least one embodiment, in an assisted system, a motion feature 260 may have at least one movable portion 260A and at least one fixed portion 260B. At least one support feature 256 may have similar fixed and movable portions. In at least one embodiment, at least one movement component is positioned within the portions 260A, 260B of the motion feature 260. In at least one embodiment, the at least one movement component is a component that may be associated with the gear subsystem, the electromagnetic subsystem, the thermoelectric generator subsystem, the thermo-reactive subsystem, or the pneumatic subsystem discussed throughout this disclosure.

[0064] In at least one embodiment, the cooling systems 200, 250 illustrate that at least one ribbon feature is associated with the fins to enable individual fins to include an overlapping portion. The overlapping portion includes one or more surfaces of individual fins that is at least partially shielded from the environment in the first configuration and that is separated to expose the overlapping portion to the environment in the second configuration. Furthermore, in at least one embodiment, the at least one ribbon feature associated with the fins is formed in part from a bimorph material to enable the first plate movable relative to the second plate upon a sensed heat acting on the bimorph material. This embodiment may represent the aforementioned unassisted system for the fins movement from the first to the second configurations.

[0065] In at least one embodiment, for the pneumatic subsystem of an assisted system for the fins, the motion feature may utilize coolant from a liquid cooling system to provide the movement of the first plate relative to the second plate. In at least one embodiment, fluid or gas lines (including vapor lines carrying a refrigerant) enable receipt of a cooling fluid (or media) from a cooling loop of a datacenter hosting the datacenter device. In at least one embodiment,

the pneumatic subsystem uses the cooling fluid to extend a piston and to cause the first plate to move relative to the second plate and to expose the surface area of the fins in the second configuration of the fins.

[0066] FIGS. 3A and 3B illustrate aspects of adaptable fins 300; 320; 350, according to at least one embodiment. In at least one embodiment, the fin 300 is either a unibody structure or a combination of portions. In at least one embodiment, the fin 300 has a first portion 302, a second portion 304, and a middle portion 306. In at least one embodiment, the middle portion 306 may be a ribbon feature formed of a different material than the first and the second portions. In at least one embodiment, the middle portion 306 is formed of a same material as the first portion and the second portion but may be a dimensionally different portion so that it can bend more than the first or the second portions. In at least one embodiment, the dimensionally different portion is in reference to a thinner portion of the same or a similar material as used for the first and the second portions, thereby making the middle portion 306 a ribbon feature that is bendable relative to the first and the second portions. In at least one embodiment, the middle portion 306 is composed of a bimorph metal changes shape or structure when heat is applied. This enables the middle portion 306 to move relative to the first portion 302, and causes the attached second portion 304 to also move. An end result is at least an inside surface of one or more of the first and of the second portions of the fins exposed to at least dissipate more heat than when the first were in their first configuration. This, in at least one embodiment, represents an intelligent but processor-less adaptable fin for heat sinks.

[0067] In at least one embodiment, the middle portion has two surfaces that interface between the overlapping surfaces of the first and the second portions 302, 304. In at least one embodiment, there are attachment material 308A, 308B provided to associate the fin 300 to plates of a heat sink, such as the heat sink in FIGS. 2A, 2B. In at least one embodiment, only the attachment material 308B at the bottom section 302 is provided for associating the fin 300 to a bottom plate. This may be the case for an unassisted system fin, which is able to maintain a vertical structure whether the fin 300 is in a contracted or an expanded position. The middle portion has sufficient rigidity to change shape or structure from bent to almost straight without causing the fin to droop, in at least one embodiment.

[0068] In at least one embodiment, the fin 300 may be designed to maintain a vertical bottom

structure and a horizontal (or diagonal) top structure. In at least one embodiment, the first portion 302 is vertical and the second portion 304 is horizontal or diagonal when fin 300 is in a second or expanded configuration. This ensures that at least an inside surface of the first portion 302 is exposed to the environment to cause an increase in heat dissipated from the fin 300. In at least one embodiment, the first portion 302 is vertical and the second portion 304 is also vertical. The top-most position in these different configurations may be a top position of the second portion 304 in the fully extended horizontal, diagonal, or vertical position.

[0069] FIG. 3B illustrates at least a second configuration of a fin 320, distinguished from the first position of the fin 300 in FIG. 3A. FIG. 3B also illustrates a middle portion or a ribbon feature 326 having changed shape or structure. In at least one embodiment, the changed shape is because the middle portion 326 is not bent or curved at an as much an angle as illustrated by the middle portion 306 of FIG. 3A. Even though the middle portion 326 does still include a curve or bend (representing a shape or structure), the curve or bend is not as much as the curve or bend of the middle portion 306 of fin 300. This at least enables the first portion 322 of the fin 320 to expose more of its inside surface 328B and enables the second portion 324 to expose more of its inside surface 328A. While the inside surface 328A of the second portion 324 may have some exposure to the environment because of a gap that may exist to the first portion 322 (see gap between the first and the second portions 302, 304 of fin 300), there is an obstruction to the inside surface 328A as a result of the first configuration and the gap there between. The exposed inside surface 328A is able to dissipate more heat in the second configuration of the fin 320 than the first configuration of the fin 300.

[0070] In at least one embodiment, FIG. 3C illustrates that multiple portions 354A, 54B may be provided over a first portion 352 of a fin 350, different from the fins 300; 320 of FIGS. 3A, 3B. In at least one embodiment, the multiple portions 354A, 354B may be separated by one or more middle portions or ribbon features 356A, 356B. In at least one embodiment, the material strength of the portions may be such that a second portion 354A may be caused to extend (or expose surface area) before a third portion 354B. This adaptation may enable the multiple intermediate configurations for a fin, such as the fin 350 of FIG. 3C, in at least one embodiment. In at least one embodiment, the different portions 354A, B may have different angles in their second configurations. The angles may include a vertical, horizontal, or diagonal extension relative to the first portion 352 or to a second plate that is fixed and that hosts the fin 350.

[0071] FIGS. 4A, 4B, and 4C illustrate processor-supported subsystems 400; 430; 460 to enable a cooling system incorporating adaptable fins, such as the cooling systems 200; 250, and the fins 300; 320; 350 of FIGS. 2A-3C, according to at least one embodiment. The processor-supported subsystems 400; 430; 460 are part of or comprise the motion features referenced in the discussion of at least FIGS. 2A, 2B. The processor-supported subsystems 400; 430; 460 support an assisted system for moving a first plate with respect to a second plate, as described with reference to at least FIGS. 2A, 2B. The processor-supported subsystems 400; 430; 460 are able to send at least a respective input to a respective controller to cause movement of the first plate with respect to the second plate. FIG. 4A is a processor-supported subsystem 400 that is based on a mechanical gear subsystem, which in turn is supported by controller inputs to and from an electromechanical or a mechanical controller 406 to a servo-motor 408. In at least one embodiment, when a processor senses a heat feature or cooling requirement, the processor is able to determine a cooling capacity or capability of an associated one or more heat sinks. The processor is adapted to react by at least activating an air cooling system (if not already active), and by sending input to the electromechanical or the mechanical controller 406 to cause the fins to move to the second configuration. In at least one embodiment, the processor is able to control multiple controllers of multiple heat sinks to concurrently or independently move respective fins to respective second configurations.

[0072] In at least one embodiment that uses the processor-supported subsystem 400 of FIG. 4A, the processor input is received in the electromechanical or the mechanical controller 406 that sends a further control input to a motor, such as a servo-motor 408. In at least one embodiment, the controller 406 is electromechanical and is adapted to provide signals to start, stop, and control a speed of the servo-motor 408. In at least one embodiment, the controller 406 is mechanical, and is adapted to assert or remove a mechanical braking force associated with gears 410. In at least one embodiment, a set of gears 410 translates mechanical output of the servo-motor 408 to lateral movement of a piston 404. In at least one embodiment, the piston 404 has threads that are threaded by at least one gear of the set of gears 410, to move the piston 404 in an up or down lateral movement. In at least one embodiment, the set of gears may include a bevel gear to translate rotational movement to linear movement. In at least one embodiment, the top of the piston 404 may be associated with the first plate that is movable with respect to a second plate. In at least one embodiment, the second plate hosts an enclosure 402 for one or all of the

components 406-410 of the processor-supported subsystems 400 of FIG. 4A. In at least one embodiment, the processor associated with the electromechanical or the mechanical controller 406 may reside within the enclosure 402 and may be part of a distributed control system. In at least one embodiment, the processor may be at least one of processor cores of multi-core processors 905, 906 in FIG. 9A. In at least one embodiment, the processor is able to perform part of detection of heat requirements, determination of configuration requirements, and determination of a heat sink that requires activation to change configuration on its own from within one of the processor-supported subsystems 400; 430; 460 of FIGS. 4A-C.

[0073] In at least one embodiment, FIG. 4B is a processor-supported subsystem 430 that is based on an electromagnetic subsystem supported by processor inputs to and from an electromagnetic controller 436 and electromagnets 440. In at least one embodiment, when a processor senses a heat feature or cooling requirement, the processor is able to determine a cooling capacity or capability of an associated one or more heat sinks. The processor is adapted to react by at least activating an air cooling system (if not already active), and by sending input to the electromagnetic controller 436 to cause the fins to move to the second configuration.

[0074] In at least one embodiment that uses the processor-supported subsystem 430 of FIG. 4B, the processor input is received in the electromagnetic controller 436 that sends a further control input to one or more electromagnets 440. In at least one embodiment, the electromagnets 440 translate electrical input of the electromagnetic controller 436 to magnetic attractive or repulsive forces in an order that causes lateral movement of a piston 434 from within a tube 438. In at least one embodiment, the top of the piston 434 may be associated with the first plate that is movable with respect to a second plate. In at least one embodiment, the second plate hosts an enclosure 432 for one or all of the components 436-440 of the processor-supported subsystems 430 of FIG. 4B. In at least one embodiment, the processor associated with the electromagnetic controller 436 may reside within the enclosure 432 and may be part of a distributed control system. In at least one embodiment, any of the controller 406, 436, 466 is adapted to return an input to the processor or signal in response to query on position of the piston 404, 434, 436. In at least one embodiment, any of the controller 406, 436, 466 is also adapted to return an input to the processor or signal in response to query on status of the controller 406, 436, 466.

[0075] In at least one embodiment, FIG. 4C is a processor-supported subsystem 460 that is

based on a pneumatic subsystem supported by processor inputs to and from a pneumatic controller 466 and fluid assembly 470A. In at least one embodiment, when a processor senses a heat feature or cooling requirement, the processor is able to determine a cooling capacity or capability of an associated one or more heat sinks. The processor is adapted to react by at least activating an air cooling system (if not already active), and by sending input to the pneumatic controller 436 to cause the fins to move to the second configuration. In at least one embodiment, when the processor senses a heat feature or cooling requirement, the processor is able to determine that cooling capacity or capability of an associated one or more heat sinks is improved by a liquid cooling system. The processor is adapted to react by at least activating an air cooling system (if not already active) and a liquid cooling system (if also not already active), and by sending input to the pneumatic controller 436 to cause the fins to move to the second configuration using fluid from the liquid cooling system as a secondary function of the liquid cooling system.

[0076] In at least one embodiment that uses the processor-supported subsystem 460 of FIG. 4C, the processor input is received in the pneumatic controller 466 that sends a further control input via lines 474 to one or more valves 470A. In at least one embodiment, the control input is electrical to electro-pneumatic valves forming the one or more valves 470A, or is pneumatic to a fully pneumatic valves forming the one or more valves 470A. In at least one embodiment, the one or more valves 470A translate electrical or pneumatic input of the pneumatic controller 436 to fluid within tube 468 that causes lateral movement of a piston 464. In at least one embodiment, the fluid is from the liquid cooling system, entering and leaving via lines or piping 472A, 472B, or may be provided from a fin-specific liquid storage tank for the adaptable fins via the same lines 472A, 472B. In at least one embodiment, the top of the piston 464 may be associated with the first plate that is movable with respect to a second plate. In at least one embodiment, the second plate hosts an enclosure 432 for one or all of the components 436-440 of the processor-supported subsystems 430 of FIG. 4B. In at least one embodiment, the processor associated with the pneumatic controller 466 may reside within the enclosure 462 and may be part of a distributed control system. In at least one embodiment, the enclosure 462 is also able to catch fluid leaks and has a detector to inform the processor of fluid leaks as part of the status information queried by the processor. In at least one embodiment, each of the pistons 404, 434, 464 is caused to return to its initial position within a respective tube or enclosure by a

spring acting on a bottom of the piston.

[0077] In at least one embodiment, for the pneumatic controller 466 of FIG. 4C, pneumatic control action may be provided via one or more bypass or inline pumps on either an exit side of the piping 472A, 472B. Furthermore, the pneumatic controller 466 may assert pumping actions on both the entry and the exit sides of the piping 472A, 472B. As such, when the pneumatic controller 466 can assert a suction action rather than a pushing action. In at least one embodiment, non-return valves may be used as the one or more valves 470A to keep fluid within the tube 468 or to keep fluid out of the tube 468 during pump up or down to move from the first to the second configurations or from the second to the first configurations. When the pneumatic controller 466 is causing tandem actions, there are both, suction and pushing actions acting on fluid within the piping 472A, 472B. The flow rates achieved may be higher by the tandem flow caused so that the configuration switches are achieved faster.

[0078] FIGS. 4D and 4E illustrate plan views of a plate 480, 490 incorporating one or more processor-supported subsystems to enable a cooling system incorporating adaptable fins, according to at least one embodiment. In at least one embodiment, the plate is a second plate that remains fixedly or removably coupled to a datacenter device or to an intermediate cooling surface. The plan views illustrate first locations or areas 484 of one or more motion features and second locations or areas 482 one or more support features. In at least one embodiment, the motion features may include any one of the processor-supported subsystem 400, 430, 460. In at least one embodiment, the plan views illustrate locations 484 available within the plate 480, 490 to support the size of at least a respective enclosure 402, 432, 462. In at least one embodiment, the areas 482 for the support features may be remarkably smaller than the areas 484 for the motion features. In at least one embodiment, as illustrated in the plan view of plate 490, there are only two motion features in areas 494, and four support features in areas 492 provided for these features. In at least one embodiment, there may be only one motion feature at the center of the plate 490 and four support features on the four corners of the plate. In at least one embodiment, the layout of the areas for the motion feature and the support feature are addressed in part to maximize the space for the adaptable fins. In the case of hybrid adaptable fins, the fins are able to support some of the weight of the first plate that moves relative to the bottom plate 480; 490. As such, fewer or no support features may be required. In at least one embodiment, when the adaptable fins are unassisted so that no areas are required for the motion or the support

features. The entire plate 480; 490 is provided with the adaptable fins having bimorph capabilities, in at least one embodiment.

[0079] FIG. 5 is a process flow of steps available for a method 500 of using or making the cooling system of FIGS. 2A-4E and 6A-17D, according to at least one embodiment. In at least one embodiment, a step 502 is for providing fins between a first plate and a second plate, where the first plate is movable relative to the second plate. In at least one embodiment, the first plate is movable by assisted or unassisted features provided with the fins. In at least one embodiment, step 504 enables dissipation of a first amount of heat to an environment in a first configuration of the fins. In at least one embodiment, the first configuration is as provided in at least the example of FIG. 2A with the fins having one or more overlapping portions.

[0080] In at least one embodiment, a determination is made via step 506 that an associated datacenter device could benefit from additional heat dissipation (or additional cooling). In at least one embodiment, the determination is made based in part on a current heat generated and current heat dissipated. When the determination is that a positive difference exists between the current heat generated and heat dissipated, then additional cooling may be required. In at least one embodiment, the determination is based in part on active ongoing computation versus potential computation required of the datacenter device. When additional loads are expected so that the potential computation represents an 80% to 100% usage of the datacenter device, it may be anticipated that the heat generated is likely to increase beyond being currently dissipated. In at least one embodiment, a determination is also made whether the current configuration of fins is unable to more heat than the first amount of heat, via step 508. In at least one embodiment, this determination in step 508 may be made by letting the generated heat accumulate and not dissipate for a period, such as a few second; or may be determined by noting a rate of increase of heat generated (and not being dissipated). A positive determination in step 508, enables step 510 to cause the first plate to move relative to the second plate to expose a surface area of the fins to the environment in a second configuration of the fins. In the second configuration, the fins dissipate a second amount of heat more than the first amount of heat to the environment. When the determination in step 508 is negative, a further determination via step 506 is cycled till the second configuration of the fins is required.

[0081] In at least one embodiment, a learning subsystem having at least one processor may be

used to determine when to move from the first to the second configuration and back to the first configuration. In at least one embodiment the learning subsystem is used to evaluate temperatures sensed from components, servers, or racks in the datacenter, with surface area of the fins associated with different amounts of heat dissipation. In at least one embodiment, the learning subsystem is adapted to work with multiple heat sinks concurrently or independently. In at least one embodiment, the learning subsystem may then use the surface areas of the collective fins of the multiple heat sinks or of all the fins of independent heat sinks. In at least one embodiment, the learning subsystem may then use the amounts of heat dissipation associated with the collective fins of the multiple heat sinks or all of the fins of independent heat sinks. In at least tone embodiment, the learning subsystem can provide an output associated with at least a temperature to facilitate the movement of the first plate relative to the second plate to expose the surface area of the fins.

[0082] In at least one embodiment, the learning subsystem may be implemented via the deep learning application processor, such as processor 1400 in FIG. 14 and may use the neurons 1502 and components thereof implemented using circuitry or logic, including one or more arithmetic logic units (ALUs) as described in FIG. 15. As such, the learning subsystem includes at least one processor for evaluating temperatures within servers of the one or more racks with surface area of the fins associated with the different amounts of heat dissipation.

[0083] Once trained, the learning subsystem will be able to provide the output having instructions associated with the surface areas for collective fins of multiple heat sinks or of independent heath sinks. The instructions are provided to the appropriate controllers associated with the appropriate motion features, as discussed in reference to FIGS. 4A-C. The outputs are provided from the learning subsystem that executes a machine learning model, in at least one embodiment. The machine learning model is adapted to process the temperatures using multiple neuron levels of the machine learning model having the temperatures and having prior associated surface areas of the fins. The machine learning model is adapted to provide the output associated with at least a temperature to the at least one controller. The output provided after an evaluation of the prior associated surface areas of the fins and prior associated temperatures.

[0084] Alternatively, in at least one embodiment, the difference in the temperatures and the subsequent temperatures from different heatsinks associated with different adaptable fins, along

with the requisite surface areas in the different configurations available for the different heatsinks is used to train the learning system to recognize when to activate and deactivate associated controllers for different motion features of the different heat sinks. Once trained, the learning subsystem, through appropriate controllers (also referred to herein as a centralized control system or distributed control system) described elsewhere throughout this disclosure, is able to provide one or more outputs to control the motion feature of the different heat sinks in response to temperature received from temperature sensors associated with different components, different servers, and different racks.

[0085] In at least one embodiment, aspects of the processing for the deep learning subsystem may use the collected information processed in line with the features discussed with reference to FIGS. 14, 15. In an example, the processing of the temperatures uses multiple neuron levels of the machine learning model that are loaded with one or more of the collected temperature features noted above and the corresponding surface areas of the different heat sinks (collectively or independently). The learning subsystem performs a training that may be represented as an evaluation of temperature changes associated with prior surface areas (or changes in surface areas) of each heat sink as per adjustments made to the one or more controllers associated with the different heat sinks. The neuron levels may store values associated with the evaluation process and may represent an association or correlation between the temperature changes and the surface areas for each of the different heat sinks.

[0086] In at least one embodiment, the learning subsystem, once trained, is able to determine, in application, surface areas (and appropriate positioning of respective pistons to achieve the surface areas) of the different heat sinks to achieve cooling to a temperature (or a change, such as a reduction in temperature) associated with cooling requirements of different servers, different racks, or different components, for instance. As the cooling requirements have to be within the cooling capabilities of the appropriate cooling subsystem, the learning subsystem is able to select which of the heat sinks (including type of air and/or liquid cool system) to activate to address the temperature sensed from the different servers, the different racks, or the different components. The collected temperatures and prior associated surface areas for the different heat sinks that is used to achieve the collected temperatures (or differences, for instance) may be used by the learning subsystem to provide one or more outputs associated with required surface areas (or piston movements) of different heat sinks to address the different cooling requirements reflected

by a temperature (representing a reduction) than a present temperature for the different servers, the different racks, or the different components.

[0087] In at least one embodiment, a result of the learning subsystem is, in response to a sensed temperature from the different servers, the different racks, or the different components, one or more outputs to the controllers associated with the different heat sinks that modifies a configuration of fins in the respective different heat sinks. The modification of the configuration of fins enables a determined exposed surface area and a corresponding determined dissipation of heat from the different servers, the different racks, or the different components requiring the cooling. The modified surface area may be maintained till the temperature in the requiring area reaches a temperature associated with the surface area that is known to the learning subsystem. In at least one embodiment, the modified surface area may be maintained till the temperature in the area changes by a determined value. In at least one embodiment, the modified surface area may be maintained till the temperature in the area reaches a rated temperature for the different servers, different racks, the different components, or the different surface area possible by configurations of the fins.

[0088] In at least one embodiment, a processor and a controller may function in unison. The processor (also referred to as a centralized or distributed control system) is at least one processor having at least one logic unit to control controllers associated with one or more heat sinks. In at least one embodiment, the at least one processor is within the datacenter, such as a processor 702 of FIG. 7A. The controller facilitates movement of a respective fins associated with a respective heat sink, and facilitates cooling of areas in the datacenter in response to temperatures sensed in the area. In at least one embodiment, the at least one processor is a processor core of a multi-core processor, such as multi-core processors 905, 906 in FIG. 9A. In at least one embodiment, the at least one logic unit may be adapted to receive a temperature values from temperature sensors associated with servers or the one or more racks, and is adapted to facilitate movement of fins associated with one or more heat sinks. In at least one embodiment, the controller has a microprocessor to perform its communication and control roles with its respective motion feature.

[0089] In at least one embodiment, a processor, such as the processor cores of multi-core processors 905, 906 in FIG. 9A may include a learning subsystem for evaluating temperatures from sensors in different locations in the datacenter, such as in different locations associated with

the servers or the racks, or even a component within the server, with surface areas that are associated with at least a heat sink. The learning subsystem provides an output, such as an instruction that is associated with at least a temperature or a surface area to facilitate movement plates having the fins therein so that the fins expand from a first configuration to a second configuration to address the different cooling requirements.

[0090] In at least one embodiment, the learning subsystem executes a machine learning model to process the temperature using multiple neuron levels of the machine learning model having the temperatures and having prior associated surface areas of fins for different heat sinks. The machine learning model may be a implemented using the neurons structure described in FIG. 15 and the deep learning processor as described in FIG. 14. The machine learning model provides the output associated with the surface area, from an evaluation of the prior associated surface areas, to the one or more controllers. Furthermore, an instruction output of the processor, such as a pin of a connector bus or a ball of a ball grid array, enables communication of the output with the one or more controllers to modify a surface area of first fins of a first heat sink, while maintaining second fins of a second heat sink at its existing configuration (existing surface area exposed to the environment, without changes). In at least one embodiment, the learning subsystem is, therefore, able to control surface areas of fins of one or more heat sinks, according to cooling requirements from associated datacenter devices for the one or more heat sinks.

[0091] In at least one embodiment, the present disclosure is to at least one processor for a cooling system or to a system having the at least one processor. The at least one processor includes at least one logic unit to train one or more neural network having hidden layers of neurons to evaluate temperatures sensed from components, servers, or racks in the datacenter, with surface area of fins associated with different amounts of heat dissipation from the fins. The at least one logic unit is also adapted to provide an output associated with at least a temperature to facilitate movement of a first plate relative to a second plate having the fins there between to expose a surface area of the fins. In at least one embodiment, the at least one logic unit is adapted to output at least one instruction associated with at least a temperature to facilitate the movement of the first plate relative to the second plate having the fins there between to expose the surface area of the fins.

[0092] In at least one embodiment, an unassisted system of fins is enabled by the present

disclosure. In at least one embodiment, the unassisted system supports a heat sink having fins that are movable to dissipate heat to an environment by exposing a surface area of the fins in a first configuration that is more than a primary surface area of the fins in a second configuration. In at least one embodiment, the unassisted system is a processorless subsystem to cause the fins to move. In at least one embodiment, the processorless subsystem is enabled by bimorph metal associated with individual fins. In at least one embodiment, the bimorph metal to unfold a portion of the individual fins, thereby exposing the surface area of the fins. As the bimorph material changes shape or structure in response to heat without a need for external instruction, the unassisted system of fins bears intelligence to perform heat dissipation operations on its own. In at least one embodiment, the fins are adapted to expand in response to heat from an associated component being above a threshold and adapted to contract in response to the heat being below the threshold.

[0093] In at least one embodiment, the present disclosure is to a cooling system with adaptable fins that can provide cooling from a relatively low density rack having about 10KW to higher density cooling of about 30KW using the air-base cooling subsystem to a mid-density cooling between 30-50KW, additionally, using the expanded fins; from about 50KW to 60KW using a liquid-based cooling subsystem, but can additionally provide cooling from about 60KW to 80kW using the expandable fins with the liquid cooling system to dissipate heat via a combination of air and liquid cooling media.

[0094] In at least one embodiment, the step 502 of method 500 includes associating at least one ribbon feature with the fins. In step 504, the method 500 enables, by the at least one ribbon feature, individual fins to comprise an overlapping portion that is at least partially shielded from the environment in the first configuration. In the first configuration, therefore, the fins dissipate a first amount of heat from an associated datacenter device. In at least one embodiment, step 510 includes enabling a change in structure of the at least one ribbon feature to expose the overlapping portion to the environment in the second configuration. Further, in at least one embodiment, step 510 is enabled using fluid received from a cooling loop of a datacenter hosting the datacenter device using fluid or gas lines. Step 510 then includes extending a piston associated with a pneumatic subsystem using the cooling fluid to cause the first plate to move relative to the second plate to expose the surface area of the fins in the second configuration of

the fins.

DATACENTER

[0095] FIG. 6A illustrates an example datacenter 600, in which at least one embodiment from FIGS. 2A-5 may be used. In at least one embodiment, datacenter 600 includes a datacenter infrastructure layer 610, a framework layer 620, a software layer 630, and an application layer 640. In at least one embodiment, such as described in respect to FIGS. 2A-5, features in the components of a cooling system incorporating adaptable fins may be performed inside or in collaboration with the example datacenter 600. In at least one embodiment, the infrastructure layer 610, the framework layer 620, the software layer 630, and the application layer 640 may be partly or fully provided via computing components on server trays located in racks 210 of the datacenter 200. This enables cooling systems of the present disclosure to direct cooling to certain ones of the computing components in an efficient and effective manner. Further, aspects of the datacenter, including the datacenter infrastructure layer 610, the framework layer 620, the software layer 630, and the application layer 640 may be used to support intelligent control of the controllers in the adaptable fins cooling system herein discussed with at least reference to FIGS. 2A-5 above. As such, the discussion in reference to FIGS. 6A-17D may be understood to apply to the hardware and software features required to enable or support a cooling system incorporating adaptable fins for the datacenter of FIGS. 2A-5, for instance.

[0096] In at least one embodiment, as in FIG. 6A, datacenter infrastructure layer 610 may include a resource orchestrator 612, grouped computing resources 614, and node computing resources ("node C.R.s") 616(1)-616(N), where "N" represents any whole, positive integer. In at least one embodiment, node C.R.s 616(1)-616(N) may include, but are not limited to, any number of central processing units ("CPUs") or other processors (including accelerators, field programmable gate arrays (FPGAs), graphics processors, etc.), memory devices (such as dynamic read-only memory), storage devices (such as solid state or disk drives), network input/output ("NW I/O") devices, network switches, virtual machines ("VMs"), power modules, and cooling modules, etc. In at least one embodiment, one or more node C.R.s from among node C.R.s 616(1)-616(N) may be a server having one or more of above-mentioned computing resources.

[0097] In at least one embodiment, grouped computing resources 614 may include separate groupings of node C.R.s housed within one or more racks (not shown), or many racks housed in datacenters at various geographical locations (also not shown). Separate groupings of node C.R.s within grouped computing resources 614 may include grouped compute, network, memory or storage resources that may be configured or allocated to support one or more workloads. In at least one embodiment, several node C.R.s including CPUs or processors may grouped within one or more racks to provide compute resources to support one or more workloads. In at least one embodiment, one or more racks may also include any number of power modules, cooling modules, and network switches, in any combination.

[0098] In at least one embodiment, resource orchestrator 612 may configure or otherwise control one or more node C.R.s 616(1)-616(N) and/or grouped computing resources 614. In at least one embodiment, resource orchestrator 612 may include a software design infrastructure ("SDI") management entity for datacenter 600. In at least one embodiment, resource orchestrator may include hardware, software or some combination thereof.

[0099] In at least one embodiment, as shown in FIG. 6A, framework layer 620 includes a job scheduler 622, a configuration manager 624, a resource manager 626 and a distributed file system 628. In at least one embodiment, framework layer 620 may include a framework to support software 632 of software layer 630 and/or one or more application(s) 642 of application layer 640. In at least one embodiment, software 632 or application(s) 642 may respectively include web-based service software or applications, such as those provided by Amazon Web Services, Google Cloud and Microsoft Azure. In at least one embodiment, framework layer 620 may be, but is not limited to, a type of free and open-source software web application framework such as Apache Spark™ (hereinafter "Spark") that may utilize distributed file system 628 for large-scale data processing (such as "big data"). In at least one embodiment, job scheduler 622 may include a Spark driver to facilitate scheduling of workloads supported by various layers of datacenter 600. In at least one embodiment, configuration manager 624 may be capable of configuring different layers such as software layer 630 and framework layer 620 including Spark and distributed file system 628 for supporting large-scale data processing. In at least one embodiment, resource manager 626 may be capable of managing clustered or grouped computing resources mapped to or allocated for support of distributed file system 628 and job

scheduler 622. In at least one embodiment, clustered or grouped computing resources may include grouped computing resource 614 at datacenter infrastructure layer 610. In at least one embodiment, resource manager 626 may coordinate with resource orchestrator 612 to manage these mapped or allocated computing resources.

[0100] In at least one embodiment, software 632 included in software layer 630 may include software used by at least portions of node C.R.s 616(1)-616(N), grouped computing resources 614, and/or distributed file system 628 of framework layer 620. One or more types of software may include, but are not limited to, Internet web page search software, e-mail virus scan software, database software, and streaming video content software.

[0101] In at least one embodiment, application(s) 642 included in application layer 640 may include one or more types of applications used by at least portions of node C.R.s 616(1)-616(N), grouped computing resources 614, and/or distributed file system 628 of framework layer 620. One or more types of applications may include, but are not limited to, any number of a genomics application, a cognitive compute, and a machine learning application, including training or inferencing software, machine learning framework software (such as PyTorch, TensorFlow, Caffe, etc.) or other machine learning applications used in conjunction with one or more embodiments.

[0102] In at least one embodiment, any of configuration manager 624, resource manager 626, and resource orchestrator 612 may implement any number and type of self-modifying actions based on any amount and type of data acquired in any technically feasible fashion. In at least one embodiment, self-modifying actions may relieve a datacenter operator of datacenter 600 from making possibly bad configuration decisions and possibly avoiding underutilized and/or poor performing portions of a datacenter.

[0103] In at least one embodiment, datacenter 600 may include tools, services, software or other resources to train one or more machine learning models or predict or infer information using one or more machine learning models according to one or more embodiments described herein. In at least one embodiment, in at least one embodiment, a machine learning model may be trained by calculating weight parameters according to a neural network architecture using software and computing resources described above with respect to datacenter 600. In at least one embodiment, trained machine learning models corresponding to one or more neural networks

may be used to infer or predict information using resources described above with respect to datacenter 600 by using weight parameters calculated through one or more training techniques described herein. As previously discussed, deep learning techniques may be used to support intelligent control of the controllers in the adaptable fins cooling system herein by monitoring area temperatures of the datacenter. Deep learning may be advanced using any appropriate learning network and the computing capabilities of the datacenter 600. As such, a deep neural network (DNN), a recurrent neural network (RNN) or a convolutional neural network (CNN) may be supported either simultaneously or concurrently using the hardware in the datacenter. Once a network is trained and successfully evaluated to recognize data within a subset or a slice, for instance, the trained network can provide similar representative data for using with the collected data.

[0104] In at least one embodiment, datacenter 600 may use CPUs, application-specific integrated circuits (ASICs), GPUs, FPGAs, or other hardware to perform training and/or inferencing using above-described resources. Moreover, one or more software and/or hardware resources described above may be configured as a service to allow users to train or performing inferencing of information, such as pressure, flow rates, temperature, and location information, or other artificial intelligence services.

INFERENCE AND TRAINING LOGIC

[0105] Inference and/or training logic 615 may be used to perform inferencing and/or training operations associated with one or more embodiments. In at least one embodiment, inference and/or training logic 615 may be used in system FIG. 6A for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein. In at least one embodiment, inference and/or training logic 615 may include, without limitation, hardware logic in which computational resources are dedicated or otherwise exclusively used in conjunction with weight values or other information corresponding to one or more layers of neurons within a neural network. In at least one embodiment, inference and/or training logic 615 may be used in conjunction with an application-specific integrated circuit (ASIC), such as Tensorflow® Processing Unit from Google, an inference processing unit (IPU) from Graphcore™, or a Nervana® (such as "Lake Crest") processor from Intel Corp.

[0106] In at least one embodiment, inference and/or training logic 615 may be used in conjunction with central processing unit (CPU) hardware, graphics processing unit (GPU) hardware or other hardware, such as field programmable gate arrays (FPGAs). In at least one embodiment, inference and/or training logic 615 includes, without limitation, code and/or data storage modules which may be used to store code (such as graph code), weight values and/or other information, including bias values, gradient information, momentum values, and/or other parameter or hyperparameter information. In at least one embodiment, each of the code and/or data storage modules is associated with a dedicated computational resource. In at least one embodiment, the dedicated computational resource includes computational hardware that further include one or more ALUs that perform mathematical functions, such as linear algebraic functions, only on information stored in code and/or data storage modules, and results from which are stored in an activation storage module of the inference and/or training logic 615.

[0107] FIGS. 6B, 6C illustrates inference and/or training logic, such as used in FIG. 6A and in at least one embodiment of the present disclosure, according to at least one embodiment. The inference and/or training logic 615 are used to perform inferencing and/or training operations associated with at least one embodiment. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. The inference and/or training logic 615 of FIGS. 6B and 6C are distinguished by the use of the arithmetic logic units (ALUs) 610 versus the computational hardware 602, 606. In at least one embodiment, each of computational hardware 602 and computational hardware 606 includes one or more ALUs that perform mathematical functions, such as linear algebraic functions, only on information stored in code and/or data storage 601 and code and/or data storage 605, respectively, result of which is stored in activation storage 620. As such, FIGS. 6B and 6C may be alternatives and may be used interchangeably unless stated otherwise.

[0108] In at least one embodiment, inference and/or training logic 615 may include, without limitation, code and/or data storage 601 to store forward and/or output weight and/or input/output data, and/or other parameters to configure neurons or layers of a neural network trained and/or used for inferencing in at least one embodiment. In at least one embodiment, training logic 615 may include, or be coupled to code and/or data storage 601 to store graph code or other software to control timing and/or order, in which weight and/or other parameter

information is to be loaded to configure, logic, including integer and/or floating point units (collectively, arithmetic logic units (ALUs). In at least one embodiment, code, such as graph code, loads weight or other parameter information into processor ALUs based on an architecture of a neural network to which the code corresponds. In at least one embodiment, code and/or data storage 601 stores weight parameters and/or input/output data of each layer of a neural network trained or used in conjunction with at least one embodiment during forward propagation of input/output data and/or weight parameters during training and/or inferencing using at least one embodiment. In at least one embodiment, any portion of code and/or data storage 601 may be included with other on-chip or off-chip data storage, including a processor's L1, L2, or L3 cache or system memory.

[0109] In at least one embodiment, any portion of code and/or data storage 601 may be internal or external to one or more processors or other hardware logic devices or circuits. In at least one embodiment, code and/or code and/or data storage 601 may be cache memory, dynamic randomly addressable memory ("DRAM"), static randomly addressable memory ("SRAM"), non-volatile memory (such as Flash memory), or other storage. In at least one embodiment, choice of whether code and/or code and/or data storage 601 is internal or external to a processor, for example, or included of DRAM, SRAM, Flash or some other storage type may depend on available storage on-chip versus off-chip, latency requirements of training and/or inferencing functions being performed, batch size of data used in inferencing and/or training of a neural network, or some combination of these factors.

[0110] In at least one embodiment, inference and/or training logic 615 may include, without limitation, a code and/or data storage 605 to store backward and/or output weight and/or input/output data corresponding to neurons or layers of a neural network trained and/or used for inferencing in at least one embodiment. In at least one embodiment, code and/or data storage 605 stores weight parameters and/or input/output data of each layer of a neural network trained or used in conjunction with at least one embodiment during backward propagation of input/output data and/or weight parameters during training and/or inferencing using at least one embodiment. In at least one embodiment, training logic 615 may include, or be coupled to code and/or data storage 605 to store graph code or other software to control timing and/or order, in

which weight and/or other parameter information is to be loaded to configure, logic, including integer and/or floating point units (collectively, arithmetic logic units (ALUs).

[0111] In at least one embodiment, code, such as graph code, loads weight or other parameter information into processor ALUs based on an architecture of a neural network to which the code corresponds. In at least one embodiment, any portion of code and/or data storage 605 may be included with other on-chip or off-chip data storage, including a processor's L1, L2, or L3 cache or system memory. In at least one embodiment, any portion of code and/or data storage 605 may be internal or external to on one or more processors or other hardware logic devices or circuits. In at least one embodiment, code and/or data storage 605 may be cache memory, DRAM, SRAM, non-volatile memory (such as Flash memory), or other storage. In at least one embodiment, choice of whether code and/or data storage 605 is internal or external to a processor, for example, or included of DRAM, SRAM, Flash or some other storage type may depend on available storage on-chip versus off-chip, latency requirements of training and/or inferencing functions being performed, batch size of data used in inferencing and/or training of a neural network, or some combination of these factors.

[0112] In at least one embodiment, code and/or data storage 601 and code and/or data storage 605 may be separate storage structures. In at least one embodiment, code and/or data storage 601 and code and/or data storage 605 may be same storage structure. In at least one embodiment, code and/or data storage 601 and code and/or data storage 605 may be partially same storage structure and partially separate storage structures. In at least one embodiment, any portion of code and/or data storage 601and code and/or data storage 605 may be included with other on-chip or off-chip data storage, including a processor's L1, L2, or L3 cache or system memory.

[0113] In at least one embodiment, inference and/or training logic 615 may include, without limitation, one or more arithmetic logic unit(s) ("ALU(s)") 610, including integer and/or floating point units, to perform logical and/or mathematical operations based, at least in part on, or indicated by, training and/or inference code (such as graph code), a result of which may produce activations (such as output values from layers or neurons within a neural network) stored in an activation storage 620 that are functions of input/output and/or weight parameter data stored in code and/or data storage 601 and/or code and/or data storage 605. In at least one embodiment, activations stored in activation storage 620 are generated according to linear algebraic and or

matrix-based mathematics performed by ALU(s) 610 in response to performing instructions or other code, wherein weight values stored in code and/or data storage 605 and/or code and/or data storage 601 are used as operands along with other values, such as bias values, gradient information, momentum values, or other parameters or hyperparameters, any or all of which may be stored in code and/or data storage 605 or code and/or data storage 601 or another storage on or off-chip.

[0114] In at least one embodiment, ALU(s) 610 are included within one or more processors or other hardware logic devices or circuits, whereas in another embodiment, ALU(s) 610 may be external to a processor or other hardware logic device or circuit that uses them (such as a co-processor). In at least one embodiment, ALUs 610 may be included within a processor's execution units or otherwise within a bank of ALUs accessible by a processor's execution units either within same processor or distributed between different processors of different types (such as central processing units, graphics processing units, fixed function units, etc.). In at least one embodiment, code and/or data storage 601, code and/or data storage 605, and activation storage 620 may be on same processor or other hardware logic device or circuit, whereas in another embodiment, they may be in different processors or other hardware logic devices or circuits, or some combination of same and different processors or other hardware logic devices or circuits. In at least one embodiment, any portion of activation storage 620 may be included with other on-chip or off-chip data storage, including a processor's L1, L2, or L3 cache or system memory. Furthermore, inferencing and/or training code may be stored with other code accessible to a processor or other hardware logic or circuit and fetched and/or processed using a processor's fetch, decode, scheduling, execution, retirement and/or other logical circuits.

[0115] In at least one embodiment, activation storage 620 may be cache memory, DRAM, SRAM, non-volatile memory (such as Flash memory), or other storage. In at least one embodiment, activation storage 620 may be completely or partially within or external to one or more processors or other logical circuits. In at least one embodiment, choice of whether activation storage 620 is internal or external to a processor, for example, or included of DRAM, SRAM, Flash or some other storage type may depend on available storage on-chip versus off-chip, latency requirements of training and/or inferencing functions being performed, batch size of data used in inferencing and/or training of a neural network, or some combination of these

factors. In at least one embodiment, inference and/or training logic 615 illustrated in FIG. 6B may be used in conjunction with an application-specific integrated circuit ("ASIC"), such as Tensorflow® Processing Unit from Google, an inference processing unit (IPU) from Graphcore™, or a Nervana® (such as "Lake Crest") processor from Intel Corp. In at least one embodiment, inference and/or training logic 615 illustrated in FIG. 6B may be used in conjunction with central processing unit ("CPU") hardware, graphics processing unit ("GPU") hardware or other hardware, such as field programmable gate arrays ("FPGAs").

[0116] In at least one embodiment, as illustrated in FIG. 6C, inference and/or training logic 615 may include, without limitation, hardware logic in which computational resources are dedicated or otherwise exclusively used in conjunction with weight values or other information corresponding to one or more layers of neurons within a neural network. In at least one embodiment, inference and/or training logic 615 illustrated in FIG. 6C may be used in conjunction with an application-specific integrated circuit (ASIC), such as Tensorflow® Processing Unit from Google, an inference processing unit (IPU) from Graphcore™, or a Nervana® (such as "Lake Crest") processor from Intel Corp. In at least one embodiment, inference and/or training logic 615 illustrated in FIG. 6C may be used in conjunction with central processing unit (CPU) hardware, graphics processing unit (GPU) hardware or other hardware, such as field programmable gate arrays (FPGAs). In at least one embodiment, inference and/or training logic 615 includes, without limitation, code and/or data storage 601 and code and/or data storage 605, which may be used to store code (such as graph code), weight values and/or other information, including bias values, gradient information, momentum values, and/or other parameter or hyperparameter information. In at least one embodiment illustrated in FIG. 6C, each of code and/or data storage 601 and code and/or data storage 605 is associated with a dedicated computational resource, such as computational hardware 602 and computational hardware 606, respectively.

[0117] In at least one embodiment, each of code and/or data storage 601 and 605 and corresponding computational hardware 602 and 606, respectively, correspond to different layers of a neural network, such that resulting activation from one "storage/computational pair 601/602" of code and/or data storage 601 and computational hardware 602 is provided as an input to "storage/computational pair 605/606" of code and/or data storage 605 and computational

hardware 606, in order to mirror conceptual organization of a neural network. In at least one embodiment, each of storage/computational pairs 601/602 and 605/606 may correspond to more than one neural network layer. In at least one embodiment, additional storage/computation pairs (not shown) subsequent to or in parallel with storage computation pairs 601/602 and 605/606 may be included in inference and/or training logic 615.

COMPUTER SYSTEMS

[0118] FIG. 7A is a block diagram illustrating an exemplary computer system 700A, which may be a system with interconnected devices and components, a system-on-a-chip (SOC) or some combination thereof formed with a processor that may include execution units to execute an instruction to support and/or to enable the intelligent control of a cooling system incorporating adaptable fins as described herein, according to at least one embodiment. In at least one embodiment, computer system 700A may include, without limitation, a component, such as a processor 702 to employ execution units including logic to perform algorithms for process data, in accordance with present disclosure, such as in embodiment described herein. In at least one embodiment, computer system 700A may include processors, such as PENTIUM® Processor family, Xeon™, Itanium®, XScale™ and/or StrongARM™, Intel® Core™, or Intel® Nervana™ microprocessors available from Intel Corporation of Santa Clara, California, although other systems (including PCs having other microprocessors, engineering workstations, set-top boxes and like) may also be used. In at least one embodiment, computer system 700B may execute a version of WINDOWS' operating system available from Microsoft Corporation of Redmond, Wash., although other operating systems (UNIX and Linux for example), embedded software, and/or graphical user interfaces, may also be used.

[0119] In at least one embodiment, the exemplary computer system 700A may incorporate one or more of components 110-116 (from FIG. 1) to support processing aspects for the intelligent control for a cooling system incorporating adaptable fins. For at least this reason, in one embodiment, FIG. 7A illustrates a system, which includes interconnected hardware devices or "chips", whereas in other embodiments, FIG. 7A may illustrate an exemplary System on a Chip ("SoC"). In at least one embodiment, devices may be interconnected with proprietary interconnects, standardized interconnects (such as PCIe) or some combination thereof. In at least one embodiment, one or more components of computer system 700B are interconnected using

compute express link (CXL) interconnects. Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments, as previously discussed with respect to FIGS. 6A-C, for instance. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6A-C. In at least one embodiment, inference and/or training logic 615 may be used in system FIG. 7A for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0120] Embodiments may be used in other devices such as handheld devices and embedded applications. Some examples of handheld devices include cellular phones, Internet Protocol devices, digital cameras, personal digital assistants ("PDAs"), and handheld PCs. In at least one embodiment, embedded applications may include a microcontroller, a digital signal processor ("DSP"), system on a chip, network computers ("NetPCs"), set-top boxes, network hubs, wide area network ("WAN") switches, or any other system that may perform one or more instructions in accordance with at least one embodiment.

[0121] In at least one embodiment, computer system 700A may include, without limitation, processor 702 that may include, without limitation, one or more execution units 708 to perform machine learning model training and/or inferencing according to techniques described herein. In at least one embodiment, computer system 700A is a single processor desktop or server system, but in another embodiment computer system 700A may be a multiprocessor system. In at least one embodiment, processor 702 may include, without limitation, a complex instruction set computer ("CISC") microprocessor, a reduced instruction set computing ("RISC") microprocessor, a very long instruction word ("VLIW") microprocessor, a processor implementing a combination of instruction sets, or any other processor device, such as a digital signal processor, for example. In at least one embodiment, processor 702 may be coupled to a processor bus 710 that may transmit data signals between processor 702 and other components in computer system 700A.

[0122] In at least one embodiment, processor 702 may include, without limitation, a Level 1 ("L1") internal cache memory ("cache") 704. In at least one embodiment, processor 702 may have a single internal cache or multiple levels of internal cache. In at least one embodiment,

cache memory may reside external to processor 702. Other embodiments may also include a combination of both internal and external caches depending on particular implementation and needs. In at least one embodiment, register file 706 may store different types of data in various registers including, without limitation, integer registers, floating point registers, status registers, and instruction pointer register.

[0123] In at least one embodiment, execution unit 708, including, without limitation, logic to perform integer and floating point operations, also resides in processor 702. In at least one embodiment, processor 702 may also include a microcode ("ucode") read only memory ("ROM") that stores microcode for certain macro instructions. In at least one embodiment, execution unit 708 may include logic to handle a packed instruction set 709. In at least one embodiment, by including packed instruction set 709 in an instruction set of a general-purpose processor 702, along with associated circuitry to execute instructions, operations used by many multimedia applications may be performed using packed data in a general-purpose processor 702. In one or more embodiments, many multimedia applications may be accelerated and executed more efficiently by using full width of a processor's data bus for performing operations on packed data, which may eliminate need to transfer smaller units of data across processor's data bus to perform one or more operations one data element at a time.

[0124] In at least one embodiment, execution unit 708 may also be used in microcontrollers, embedded processors, graphics devices, DSPs, and other types of logic circuits. In at least one embodiment, computer system 700A may include, without limitation, a memory 720. In at least one embodiment, memory 720 may be implemented as a Dynamic Random Access Memory ("DRAM") device, a Static Random Access Memory ("SRAM") device, flash memory device, or other memory device. In at least one embodiment, memory 720 may store instruction(s) 719 and/or data 721 represented by data signals that may be executed by processor 702.

[0125] In at least one embodiment, system logic chip may be coupled to processor bus 710 and memory 720. In at least one embodiment, system logic chip may include, without limitation, a memory controller hub ("MCH") 716, and processor 702 may communicate with MCH 716 via processor bus 710. In at least one embodiment, MCH 716 may provide a high bandwidth memory path 718 to memory 720 for instruction and data storage and for storage of graphics commands, data and textures. In at least one embodiment, MCH 716 may direct data signals

between processor 702, memory 720, and other components in computer system 700A and to bridge data signals between processor bus 710, memory 720, and a system I/O 722. In at least one embodiment, system logic chip may provide a graphics port for coupling to a graphics controller. In at least one embodiment, MCH 716 may be coupled to memory 720 through a high bandwidth memory path 718 and graphics/video card 712 may be coupled to MCH 716 through an Accelerated Graphics Port ("AGP") interconnect 714.

[0126] In at least one embodiment, computer system 700A may use system I/O 722 that is a proprietary hub interface bus to couple MCH 716 to I/O controller hub ("ICH") 730. In at least one embodiment, ICH 730 may provide direct connections to some I/O devices via a local I/O bus. In at least one embodiment, local I/O bus may include, without limitation, a high-speed I/O bus for connecting peripherals to memory 720, chipset, and processor 702. Examples may include, without limitation, an audio controller 729, a firmware hub ("flash BIOS") 728, a wireless transceiver 726, a data storage 724, a legacy I/O controller 723 containing user input and keyboard interfaces 725, a serial expansion port 727, such as Universal Serial Bus ("USB"), and a network controller 734. Data storage 724 may include a hard disk drive, a floppy disk drive, a CD-ROM device, a flash memory device, or other mass storage device.

[0127] FIG. 7B is a block diagram illustrating an electronic device 700B for utilizing a processor 710 to support and/or to enable intelligent control of a cooling system incorporating adaptable fins as described herein, according to at least one embodiment. In at least one embodiment, electronic device 700B may be, for example and without limitation, a notebook, a tower server, a rack server, a blade server, a laptop, a desktop, a tablet, a mobile device, a phone, an embedded computer, or any other suitable electronic device. In at least one embodiment, the exemplary electronic device 700B may incorporate one or more of components that support processing aspects for the cooling system incorporating adaptable fins.

[0128] In at least one embodiment, system 700B may include, without limitation, processor 710 communicatively coupled to any suitable number or kind of components, peripherals, modules, or devices. In at least one embodiment, processor 710 coupled using a bus or interface, such as a 1°C bus, a System Management Bus ("SMBus"), a Low Pin Count (LPC) bus, a Serial Peripheral Interface ("SPI"), a High Definition Audio ("HDA") bus, a Serial Advance Technology Attachment ("SATA") bus, a Universal Serial Bus ("USB") (versions 1, 2, 3), or a

Universal Asynchronous Receiver/Transmitter ("UART") bus. In at least one embodiment, FIG. 7B illustrates a system, which includes interconnected hardware devices or "chips", whereas in other embodiments, FIG. 7B may illustrate an exemplary System on a Chip ("SoC"). In at least one embodiment, devices illustrated in FIG. 7B may be interconnected with proprietary interconnects, standardized interconnects (such as PCIe) or some combination thereof. In at least one embodiment, one or more components of FIG. 7B are interconnected using compute express link (CXL) interconnects.

[0129] In at least one embodiment, FIG 7B may include a display 724, a touch screen 725, a touch pad 730, a Near Field Communications unit ("NFC") 745, a sensor hub 740, a thermal sensor 746, an Express Chipset ("EC") 735, a Trusted Platform Module ("TPM") 738, BIOS/firmware/flash memory ("BIOS, FW Flash") 722, a DSP 760, a drive 720 such as a Solid State Disk ("SSD") or a Hard Disk Drive ("HDD"), a wireless local area network unit ("WLAN") 750, a Bluetooth unit 752, a Wireless Wide Area Network unit ("WWAN") 756, a Global Positioning System (GPS) 755, a camera ("USB 3.0 camera") 754 such as a USB 3.0 camera, and/or a Low Power Double Data Rate ("LPDDR") memory unit ("LPDDR3") 715 implemented in, for example, LPDDR3 standard. These components may each be implemented in any suitable manner.

[0130] In at least one embodiment, other components may be communicatively coupled to processor 710 through components discussed above. In at least one embodiment, an accelerometer 741, Ambient Light Sensor ("ALS") 742, compass 743, and a gyroscope 744 may be communicatively coupled to sensor hub 740. In at least one embodiment, thermal sensor 739, a fan 737, a keyboard 746, and a touch pad 730 may be communicatively coupled to EC 735. In at least one embodiment, speaker 763, headphones 764, and microphone ("mic") 765 may be communicatively coupled to an audio unit ("audio codec and class d amp") 762, which may in turn be communicatively coupled to DSP 760. In at least one embodiment, audio unit 764 may include, for example and without limitation, an audio coder/decoder ("codec") and a class D amplifier. In at least one embodiment, SIM card ("SIM") 757 may be communicatively coupled to WWAN unit 756. In at least one embodiment, components such as WLAN unit 750 and Bluetooth unit 752, as well as WWAN unit 756 may be implemented in a Next Generation Form Factor ("NGFF").

[0131] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, inference and/or training logic x615 may be used in system FIG. 7B for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0132] FIG. 7C illustrates a computer system 700C, according to at least one embodiment, to support and/or to enable the intelligent control of a cooling system incorporating adaptable fins as described herein. In at least one embodiment, computer system 700C includes, without limitation, a computer 771 and a USB stick 770. In at least one embodiment, computer 771 may include, without limitation, any number and type of processor(s) (not shown) and a memory (not shown). In at least one embodiment, computer 771 includes, without limitation, a server, a cloud instance, a laptop, and a desktop computer.

[0133] In at least one embodiment, USB stick 770 includes, without limitation, a processing unit 772, a USB interface 774, and USB interface logic 773. In at least one embodiment, processing unit 772 may be any instruction execution system, apparatus, or device capable of executing instructions. In at least one embodiment, processing unit 772 may include, without limitation, any number and type of processing cores (not shown). In at least one embodiment, processing unit or core 772 comprises an application specific integrated circuit ("ASIC") that is optimized to perform any amount and type of operations associated with machine learning. For instance, in at least one embodiment, processing core 772 is a tensor processing unit ("TPC") that is optimized to perform machine learning inference operations. In at least one embodiment, processing core 772 is a vision processing unit ("VPU") that is optimized to perform machine vision and machine learning inference operations.

[0134] In at least one embodiment, USB interface 774 may be any type of USB connector or USB socket. For instance, in at least one embodiment, USB interface 774 is a USB 3.0 Type-C socket for data and power. In at least one embodiment, USB interface 774 is a USB 3.0 Type-A connector. In at least one embodiment, USB interface logic 773 may include any amount and

type of logic that enables processing unit 772 to interface with or devices (such as computer 771) via USB connector 774.

[0135] Inference and/or training logic 615, as described with respect to FIGS. 6B and 6C, are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, inference and/or training logic 615 may be used in system FIG. 7C for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0136] FIG. 8 illustrates a further example computer system 800, according to at least one embodiment, to implement various processes and methods for a cooling system incorporating adaptable fins described throughout this disclosure. In at least one embodiment, computer system 800 includes, without limitation, at least one central processing unit ("CPU") 802 that is connected to a communication bus 810 implemented using any suitable protocol, such as PCI ("Peripheral Component Interconnect"), peripheral component interconnect express ("PCI-Express"), AGP ("Accelerated Graphics Port"), HyperTransport, or any other bus or point-to-point communication protocol(s). In at least one embodiment, computer system 800 includes, without limitation, a main memory 804 and control logic (such as implemented as hardware, software, or a combination thereof) and data are stored in main memory 804 which may take form of random access memory ("RAM"). In at least one embodiment, a network interface subsystem ("network interface") 822 provides an interface to other computing devices and networks for receiving data from and transmitting data to other systems from computer system 800.

[0137] In at least one embodiment, computer system 800, in at least one embodiment, includes, without limitation, input devices 808, parallel processing system 812, and display devices 806 which can be implemented using a cathode ray tube ("CRT"), liquid crystal display ("LCD"), light emitting diode ("LED"), plasma display, or other suitable display technologies. In at least one embodiment, user input is received from input devices 808 such as keyboard, mouse, touchpad, microphone, and more. In at least one embodiment, each of foregoing modules can be situated on a single semiconductor platform to form a processing system.

[0138] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments, as previously discussed with respect to FIGS. 6A-C, for instance. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6A-C. In at least one embodiment, inference and/or training logic 615 may be used in system FIG. 8 for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein. In at least one embodiment, inference and/or training logic 615 may be used in system FIG. 8 for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0139] FIG. 9A illustrates an exemplary architecture in which a plurality of GPUs 910-913 is communicatively coupled to a plurality of multi-core processors 905-906 over high-speed links 940-943 (such as buses, point-to-point interconnects, etc.). In one embodiment, high-speed links 940-943 support a communication throughput of 4GB/s, 30GB/s, 80GB/s or higher. Various interconnect protocols may be used including, but not limited to, PCIe 4.0 or 5.0 and NVLink 2.0.

[0140] In addition, and in one embodiment, two or more of GPUs 910-913 are interconnected over high-speed links 929-930, which may be implemented using same or different protocols/links than those used for high-speed links 940-943. Similarly, two or more of multi-core processors 905-906 may be connected over high speed link 928 which may be symmetric multi-processor (SMP) buses operating at 20GB/s, 30GB/s, 120GB/s or higher. Alternatively, all communication between various system components shown in FIG. 9A may be accomplished using same protocols/links (such as over a common interconnection fabric).

[0141] In one embodiment, each multi-core processor 905-906 is communicatively coupled to a processor memory 901-902, via memory interconnects 926-927, respectively, and each GPU 910-913 is communicatively coupled to GPU memory 920-923 over GPU memory interconnects 950-953, respectively. Memory interconnects 926-927 and 950-953 may utilize same or different memory access technologies. By way of example, and not limitation, processor memories 901-902 and GPU memories 920-923 may be volatile memories such as dynamic random access memories (DRAMs) (including stacked DRAMs), Graphics DDR SDRAM

(GDDR) (such as GDDR5, GDDR6), or High Bandwidth Memory (HBM) and/or may be non-volatile memories such as 3D XPoint or Nano-Ram. In one embodiment, some portion of processor memories 901-902 may be volatile memory and another portion may be non-volatile memory (such as using a two-level memory (2LM) hierarchy).

[0142] As described below, although various processors 905-906 and GPUs 910-913 may be physically coupled to a particular memory 901-902, 920-923, respectively, a unified memory architecture may be implemented in which a same virtual system address space (also referred to as "effective address" space) is distributed among various physical memories. In at least one embodiment, processor memories 901-902 may each include 64GB of system memory address space and GPU memories 920-923 may each include 32GB of system memory address space (resulting in a total of 256GB addressable memory in this example).

[0143] As discussed elsewhere in this disclosure, at least flow rates and associated temperatures may be established for a first level of an intelligent learning system, such as a neural network system. As the first level represents the prior data, it also represents a smaller subset of the data that may be available to improve the system by retraining the system. The testing and training may be performed in parallel using the multiple processor units so that the intelligent learning system is robust. An architecture, such as in FIG. 9A, may be used. When convergence is achieved for the intelligent learning system, an amount of data points and the data in the data points used to cause the convergence is noted. The data and data points may be used to control a cooling system incorporating adaptable fins, entirely, as discussed in reference, for instance, to FIGS. 2A-5.

[0144] FIG. 9B illustrates additional details for an interconnection between a multi-core processor 907 and a graphics acceleration module 946 in accordance with one exemplary embodiment. Graphics acceleration module 946 may include one or more GPU chips integrated on a line card which is coupled to processor 907 via high-speed link 940. Alternatively, graphics acceleration module 946 may be integrated on a same package or chip as processor 907.

[0145] In at least one embodiment, illustrated processor 907 includes a plurality of cores 960A-960D, each with a translation lookaside buffer 961A-961D and one or more caches 962A-962D. In at least one embodiment, cores 960A-960D may include various other components for executing instructions and processing data which are not illustrated. Caches 962A-962D may

include level 1 (L1) and level 2 (L2) caches. In addition, one or more shared caches 956 may be included in caches 962A-962D and shared by sets of cores 960A-960D. In at least one embodiment, one embodiment of processor 907 includes 24 cores, each with its own L1 cache, twelve shared L2 caches, and twelve shared L3 caches. In this embodiment, one or more L2 and L3 caches are shared by two adjacent cores. Processor 907 and graphics acceleration module 946 connect with system memory 914, which may include processor memories 901-902 of FIG. 9A.

[0146] Coherency is maintained for data and instructions stored in various caches 962A-962D, 956 and system memory 914 via inter-core communication over a coherence bus 964. In at least one embodiment, each cache may have cache coherency logic/circuitry associated therewith to communicate to over coherence bus 964 in response to detected reads or writes to particular cache lines. In one implementation, a cache snooping protocol is implemented over coherence bus 964 to snoop cache accesses.

[0147] In one embodiment, a proxy circuit 925 communicatively couples graphics acceleration module 946 to coherence bus 964, allowing graphics acceleration module 946 to participate in a cache coherence protocol as a peer of cores 960A-960D. In particular, an interface 935 provides connectivity to proxy circuit 925 over high-speed link 940 (such as a PCIe bus, NVLink, etc.) and an interface 937 connects graphics acceleration module 946 to link 940.

[0148] In one implementation, an accelerator integration circuit 936 provides cache management, memory access, context management, and interrupt management services on behalf of a plurality of graphics processing engines 931, 932, N of graphics acceleration module 946. Graphics processing engines 931, 932, N may each include a separate graphics processing unit (GPU). Alternatively, graphics processing engines 931, 932, N may include different types of graphics processing engines within a GPU such as graphics execution units, media processing engines (such as video encoders/decoders), samplers, and blit engines. In at least one embodiment, graphics acceleration module 946 may be a GPU with a plurality of graphics processing engines 931-932, N or graphics processing engines 931-932, N may be individual GPUs integrated on a common package, line card, or chip. As is the case, the above determination for the reconstruction parameter and the reconstruction algorithm may be performed in GPUs 931-N of FIG. 9B.

[0149] In one embodiment, accelerator integration circuit 936 includes a memory management unit (MMU) 939 for performing various memory management functions such as virtual-to-physical memory translations (also referred to as effective-to-real memory translations) and memory access protocols for accessing system memory 914. MMU 939 may also include a translation lookaside buffer (TLB) (not shown) for caching virtual/effective to physical/real address translations. In one implementation, a cache 938 stores commands and data for efficient access by graphics processing engines 931-932, N. In one embodiment, data stored in cache 938 and graphics memories 933-934, M is kept coherent with core caches 962A-962D, 956, and system memory 914. As mentioned above, this may be accomplished via proxy circuit 925 on behalf of cache 938 and memories 933-934, M (such as sending updates to cache 938 related to modifications/accesses of cache lines on processor caches 962A-962D, 956, and receiving updates from cache 938).

[0150] A set of registers 945 store context data for threads executed by graphics processing engines 931-932, N and a context management circuit 948 manages thread contexts. In at least one embodiment, context management circuit 948 may perform save and restore operations to save and restore contexts of various threads during contexts switches (such as where a first thread is saved and a second thread is stored so that a second thread can be executed by a graphics processing engine). In at least one embodiment, on a context switch, context management circuit 948 may store current register values to a designated region in memory (such as identified by a context pointer). It may then restore register values when returning to a context. In one embodiment, an interrupt management circuit 947 receives and processes interrupts received from system devices.

[0151] In one implementation, virtual/effective addresses from a graphics processing engine 931 are translated to real/physical addresses in system memory 914 by MMU 939. One embodiment of accelerator integration circuit 936 supports multiple (such as 4, 8, 16) graphics accelerator modules 946 and/or other accelerator devices. Graphics accelerator module 946 may be dedicated to a single application executed on processor 907 or may be shared between multiple applications. In one embodiment, a virtualized graphics execution environment is presented in which resources of graphics processing engines 931-932, N are shared with multiple applications or virtual machines (VMs). In at least one embodiment, resources may be

subdivided into "slices" which are allocated to different VMs and/or applications based on processing requirements and priorities associated with VMs and/or applications.

[0152] In at least one embodiment, accelerator integration circuit 936 performs as a bridge to a system for graphics acceleration module 946 and provides address translation and system memory cache services. In addition, accelerator integration circuit 936 may provide virtualization facilities for a host processor to manage virtualization of graphics processing engines 931-932, N, interrupts, and memory management.

[0153] Because hardware resources of graphics processing engines 931-932, N are mapped explicitly to a real address space seen by host processor 907, any host processor can address these resources directly using an effective address value. One function of accelerator integration circuit 936, in one embodiment, is physical separation of graphics processing engines 931-932, N so that they appear to a system as independent units.

[0154] In at least one embodiment, one or more graphics memories 933-934, M are coupled to each of graphics processing engines 931-932, N, respectively. Graphics memories 933-934, M store instructions and data being processed by each of graphics processing engines 931-932, N. Graphics memories 933-934, M may be volatile memories such as DRAMs (including stacked DRAMs), GDDR memory (such as GDDR5, GDDR6), or HBM, and/or may be non-volatile memories such as 3D XPoint or Nano-Ram.

[0155] In one embodiment, to reduce data traffic over link 940, biasing techniques are used to ensure that data stored in graphics memories 933-934, M is data which will be used most frequently by graphics processing engines 931-932, N and may not used by cores 960A-960D (at least not frequently). Similarly, a biasing mechanism attempts to keep data needed by cores (and may not graphics processing engines 931-932, N) within caches 962A-962D, 956 of cores and system memory 914.

[0156] FIG. 9C illustrates another exemplary embodiment in which accelerator integration circuit 936 is integrated within processor 907 for enabling and/or supporting intelligent control of a cooling system incorporating adaptable fins, according to at least one embodiment of the disclosure herein. In at least this embodiment, graphics processing engines 931-932, N communicate directly over high-speed link 940 to accelerator integration circuit 936 via interface

937 and interface 935 (which, again, may be utilize any form of bus or interface protocol). Accelerator integration circuit 936 may perform same operations as those described with respect to FIG. 9B, but potentially at a higher throughput given its close proximity to coherence bus 964 and caches 962A-962D, 956. At least one embodiment supports different programming models including a dedicated-process programming model (no graphics acceleration module virtualization) and shared programming models (with virtualization), which may include programming models which are controlled by accelerator integration circuit 936 and programming models which are controlled by graphics acceleration module 946.

[0157] In at least one embodiment, graphics processing engines 931-932, N are dedicated to a single application or process under a single operating system. In at least one embodiment, a single application can funnel other application requests to graphics processing engines 931-932, N, providing virtualization within a VM/partition.

[0158] In at least one embodiment, graphics processing engines 931-932, N, may be shared by multiple VM/application partitions. In at least one embodiment, shared models may use a system hypervisor to virtualize graphics processing engines 931-932, N to allow access by each operating system. For single-partition systems without a hypervisor, graphics processing engines 931-932, N are owned by an operating system. In at least one embodiment, an operating system can virtualize graphics processing engines 931-932, N to provide access to each process or application.

[0159] In at least one embodiment, graphics acceleration module 946 or an individual graphics processing engine 931-932, N selects a process element using a process handle. In at least one embodiment, process elements are stored in system memory 914 and are addressable using an effective address to real address translation techniques described herein. In at least one embodiment, a process handle may be an implementation-specific value provided to a host process when registering its context with graphics processing engine 931-932, N (that is, calling system software to add a process element to a process element linked list). In at least one embodiment, a lower 16-bits of a process handle may be an offset of a process element within a process element linked list.

[0160] FIG. 9D illustrates an exemplary accelerator integration slice 990 for enabling and/or supporting intelligent control of a cooling system incorporating adaptable fins, according to at

least one embodiment of the disclosure herein. As used herein, a "slice" comprises a specified portion of processing resources of accelerator integration circuit 936. Application effective address space 982 within system memory 914 stores process elements 983. In one embodiment, process elements 983 are stored in response to GPU invocations 981 from applications 980 executed on processor 907. A process element 983 contains process state for corresponding application 980. A work descriptor (WD) 984 contained in process element 983 can be a single job requested by an application or may contain a pointer to a queue of jobs. In at least one embodiment, WD 984 is a pointer to a job request queue in an application's address space 982.

[0161] Graphics acceleration module 946 and/or individual graphics processing engines 931-932, N can be shared by all or a subset of processes in a system. In at least one embodiment, an infrastructure for setting up process state and sending a WD 984 to a graphics acceleration module 946 to start a job in a virtualized environment may be included.

[0162] In at least one embodiment, a dedicated-process programming model is implementation-specific. In this model, a single process owns graphics acceleration module 946 or an individual graphics processing engine 931. Because graphics acceleration module 946 is owned by a single process, a hypervisor initializes accelerator integration circuit 936 for an owning partition and an operating system initializes accelerator integration circuit 936 for an owning process when graphics acceleration module 946 is assigned.

[0163] In operation, a WD fetch unit 991 in accelerator integration slice 990 fetches next WD 984 which includes an indication of work to be done by one or more graphics processing engines of graphics acceleration module 946. Data from WD 984 may be stored in registers 945 and used by MMU 939, interrupt management circuit 947, and/or context management circuit 948 as illustrated. In at least one embodiment, one embodiment of MMU 939 includes segment/page walk circuitry for accessing segment/page tables 986 within OS virtual address space 985. Interrupt management circuit 947 may process interrupt events 992 received from graphics acceleration module 946. When performing graphics operations, an effective address 993 generated by a graphics processing engine 931-932, N is translated to a real address by MMU 939.

[0164] In one embodiment, a same set of registers 945 are duplicated for each graphics processing engine 931-932, N and/or graphics acceleration module 946 and may be initialized by

a hypervisor or operating system. Each of these duplicated registers may be included in an accelerator integration slice 990. Exemplary registers that may be initialized by a hypervisor are shown in Table 1.

Table 1 –Hypervisor Initialized Registers

1	Slice Control Register
2	Real Address (RA) Scheduled Processes Area Pointer
3	Authority Mask Override Register
4	Interrupt Vector Table Entry Offset
5	Interrupt Vector Table Entry Limit
6	State Register
7	Logical Partition ID
8	Real address (RA) Hypervisor Accelerator Utilization Record Pointer
9	Storage Description Register

[0165] Exemplary registers that may be initialized by an operating system are shown in Table 2.

Table 2 –Operating System Initialized Registers

1	Process and Thread Identification
2	Effective Address (EA) Context Save/Restore Pointer
3	Virtual Address (VA) Accelerator Utilization Record Pointer
4	Virtual Address (VA) Storage Segment Table Pointer
5	Authority Mask

6	Work descriptor

[0166] In one embodiment, each WD 984 is specific to a particular graphics acceleration module 946 and/or graphics processing engines 931-932, N. It contains all information required by a graphics processing engine 931-932, N to do work or it can be a pointer to a memory location where an application has set up a command queue of work to be completed.

[0167] FIG. 9E illustrates additional details for one exemplary embodiment of a shared model. This embodiment includes a hypervisor real address space 998 in which a process element list 999 is stored. Hypervisor real address space 998 is accessible via a hypervisor 996 which virtualizes graphics acceleration module engines for operating system 995.

[0168] In at least one embodiment, shared programming models allow for all or a subset of processes from all or a subset of partitions in a system to use a graphics acceleration module 946. There are two programming models where graphics acceleration module 946 is shared by multiple processes and partitions: time-sliced shared and graphics-directed shared.

[0169] In this model, system hypervisor 996 owns graphics acceleration module 946 and makes its function available to all operating systems 995. For a graphics acceleration module 946 to support virtualization by system hypervisor 996, graphics acceleration module 946 may adhere to the following: 1) An application's job request must be autonomous (that is, state does not need to be maintained between jobs), or graphics acceleration module 946 must provide a context save and restore mechanism. 2) An application's job request is guaranteed by graphics acceleration module 946 to complete in a specified amount of time, including any translation faults, or graphics acceleration module 946 provides an ability to preempt processing of a job. 3) Graphics acceleration module 946 must be guaranteed fairness between processes when operating in a directed shared programming model.

[0170] In at least one embodiment, application 980 is required to make an operating system 995 system call with a graphics acceleration module 946 type, a work descriptor (WD), an authority mask register (AMR) value, and a context save/restore area pointer (CSRP). In at least one embodiment, graphics acceleration module 946 type describes a targeted acceleration function for a system call. In at least one embodiment, graphics acceleration module 946 type may be a system-specific value. In at least one embodiment, WD is formatted specifically for

graphics acceleration module 946 and can be in a form of a graphics acceleration module 946 command, an effective address pointer to a user-defined structure, an effective address pointer to a queue of commands, or any other data structure to describe work to be done by graphics acceleration module 946. In one embodiment, an AMR value is an AMR state to use for a current process. In at least one embodiment, a value passed to an operating system is similar to an application setting an AMR. If accelerator integration circuit 936 and graphics acceleration module 946 implementations do not support a User Authority Mask Override Register (UAMOR), an operating system may apply a current UAMOR value to an AMR value before passing an AMR in a hypervisor call. Hypervisor 996 may, in at least one embodiment, apply a current Authority Mask Override Register (AMOR) value before placing an AMR into process element 983. In at least one embodiment, CSRP is one of registers 945 containing an effective address of an area in an application's effective address space 982 for graphics acceleration module 946 to save and restore context state. This pointer is used in at least one embodiment, if no state is required to be saved between jobs or when a job is preempted. In at least one embodiment, context save/restore area may be pinned system memory.

[0171] Upon receiving a system call, operating system 995 may verify that application 980 has registered and been given authority to use graphics acceleration module 946. Operating system 995 then calls hypervisor 996 with information shown in Table 3.

Table 3 –OS to Hypervisor Call Parameters

1	A work descriptor (WD)
2	An Authority Mask Register (AMR) value (potentially masked)
3	An effective address (EA) Context Save/Restore Area Pointer (CSRP)
4	A process ID (PID) and optional thread ID (TID)
5	A virtual address (VA) accelerator utilization record pointer (AURP)
6	Virtual address of storage segment table pointer (SSTP)
7	A logical interrupt service number (LISN)

[0172] Upon receiving a hypervisor call, hypervisor 996 verifies that operating system 995 has registered and been given authority to use graphics acceleration module 946. Hypervisor 996 then puts process element 983 into a process element linked list for a corresponding graphics acceleration module 946 type. A process element may include information shown in Table 4.

Table 4 –Process Element Information

1	A work descriptor (WD)
2	An Authority Mask Register (AMR) value (potentially masked).
3	An effective address (EA) Context Save/Restore Area Pointer (CSRP)
4	A process ID (PID) and optional thread ID (TID)
5	A virtual address (VA) accelerator utilization record pointer (AURP)
6	Virtual address of storage segment table pointer (SSTP)
7	A logical interrupt service number (LISN)
8	Interrupt vector table, derived from hypervisor call parameters
9	A state register (SR) value
10	A logical partition ID (LPID)
11	A real address (RA) hypervisor accelerator utilization record pointer
12	Storage Descriptor Register (SDR)

[0173] In at least one embodiment, hypervisor initializes a plurality of accelerator integration slice 990 registers 945.

[0174] As illustrated in FIG. 9F, in at least one embodiment, a unified memory is used, addressable via a common virtual memory address space used to access physical processor memories 901-902 and GPU memories 920-923. In this implementation, operations executed on GPUs 910-913 utilize a same virtual/effective memory address space to access processor

memories 901-902 and vice versa, thereby simplifying programmability. In one embodiment, a first portion of a virtual/effective address space is allocated to processor memory 901, a second portion to second processor memory 902, a third portion to GPU memory 920, and so on. In at least one embodiment, an entire virtual/effective memory space (sometimes referred to as an effective address space) is thereby distributed across each of processor memories 901-902 and GPU memories 920-923, allowing any processor or GPU to access any physical memory with a virtual address mapped to that memory.

[0175] In one embodiment, bias/coherence management circuitry 994A-994E within one or more of MMUs 939A-939E ensures cache coherence between caches of one or more host processors (such as 905) and GPUs 910-913 and implements biasing techniques indicating physical memories in which certain types of data should be stored. While multiple instances of bias/coherence management circuitry 994A-994E are illustrated in FIG. 9F, bias/coherence circuitry may be implemented within an MMU of one or more host processors 905 and/or within accelerator integration circuit 936.

[0176] One embodiment allows GPU-attached memory 920-923 to be mapped as part of system memory, and accessed using shared virtual memory (SVM) technology, but without suffering performance drawbacks associated with full system cache coherence. In at least one embodiment, an ability for GPU-attached memory 920-923 to be accessed as system memory without onerous cache coherence overhead provides a beneficial operating environment for GPU offload. This arrangement allows host processor 905 software to setup operands and access computation results, without overhead of tradition I/O DMA data copies. Such traditional copies involve driver calls, interrupts and memory mapped I/O (MMIO) accesses that are all inefficient relative to simple memory accesses. In at least one embodiment, an ability to access GPU attached memory 920-923 without cache coherence overheads can be critical to execution time of an offloaded computation. In cases with substantial streaming write memory traffic, for example, cache coherence overhead can significantly reduce an effective write bandwidth seen by a GPU 910-913. In at least one embodiment, efficiency of operand setup, efficiency of results access, and efficiency of GPU computation may play a role in determining effectiveness of a GPU offload.

[0177] In at least one embodiment, selection of GPU bias and host processor bias is driven by a bias tracker data structure. A bias table may be used, for example, which may be a page-granular structure (in at least one embodiment this may be controlled at a granularity of a memory page) that includes 1 or 2 bits per GPU-attached memory page. In at least one embodiment, a bias table may be implemented in a stolen memory range of one or more GPU-attached memories 920-923, with or without a bias cache in GPU 910-913 (such as to cache frequently/recently used entries of a bias table). Alternatively, an entire bias table may be maintained within a GPU.

[0178] In at least one embodiment, a bias table entry associated with each access to GPU-attached memory 920-923 is accessed prior to actual access to a GPU memory, causing the following operations. First, local requests from GPU 910-913 that find their page in GPU bias are forwarded directly to a corresponding GPU memory 920-923. Local requests from a GPU that find their page in host bias are forwarded to processor 905 (such as over a high-speed link as discussed above). In one embodiment, requests from processor 905 that find a requested page in host processor bias complete a request like a normal memory read. Alternatively, requests directed to a GPU-biased page may be forwarded to GPU 910-913. In at least one embodiment, a GPU may then transition a page to a host processor bias if it is not currently using a page. In at least one embodiment, bias state of a page can be changed either by a software-based mechanism, a hardware-assisted software-based mechanism, or, for a limited set of cases, a purely hardware-based mechanism.

[0179] One mechanism for changing bias state employs an API call (such as OpenCL), which, in turn, calls a GPU's device driver which, in turn, sends a message (or enqueues a command descriptor) to a GPU directing it to change a bias state and, for some transitions, perform a cache flushing operation in a host. In at least one embodiment, cache flushing operation is used for a transition from host processor 905 bias to GPU bias, but is not for an opposite transition.

[0180] In one embodiment, cache coherency is maintained by temporarily rendering GPU-biased pages uncacheable by host processor 905. To access these pages, processor 905 may request access from GPU 910 which may or may not grant access right away. Thus, to reduce communication between processor 905 and GPU 910 it is beneficial to ensure that GPU-biased pages are those which are required by a GPU but not host processor 905 and vice versa.

[0181] Inference and/or training logic 615 are used to perform one or more embodiments. Details regarding the inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C.

[0182] FIG. 10A illustrates exemplary integrated circuits and associated graphics processors that may be fabricated using one or more IP cores, according to various embodiments described herein, to support and/or to enable a cooling system incorporating adaptable fins as described herein. In addition to what is illustrated, other logic and circuits may be included in at least one embodiment, including additional graphics processors/cores, peripheral interface controllers, or general-purpose processor cores.

[0183] FIG. 10A is a block diagram illustrating an exemplary system on a chip integrated circuit 1000A that may be fabricated using one or more IP cores, according to at least one embodiment. In at least one embodiment, integrated circuit 1000A includes one or more application processor(s) 1005 (such as CPUs), at least one graphics processor 1010, and may additionally include an image processor 1015 and/or a video processor 1020, any of which may be a modular IP core. In at least one embodiment, integrated circuit 1000A includes peripheral or bus logic including a USB controller 1025, UART controller 1030, an SPI/SDIO controller 1035, and an I^2S/I^2C controller 1040. In at least one embodiment, integrated circuit 1000A can include a display device 1045 coupled to one or more of a high-definition multimedia interface (HDMI) controller 1050 and a mobile industry processor interface (MIPI) display interface 1055. In at least one embodiment, storage may be provided by a flash memory subsystem 1060 including flash memory and a flash memory controller. In at least one embodiment, memory interface may be provided via a memory controller 1065 for access to SDRAM or SRAM memory devices. In at least one embodiment, some integrated circuits additionally include an embedded security engine 1070.

[0184] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, inference and/or training logic 615 may be used in integrated circuit 1000A for inferencing or predicting operations based, at least in part, on weight parameters calculated using

neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0185] FIGS. 10B-10C illustrate exemplary integrated circuits and associated graphics processors that may be fabricated using one or more IP cores, according to various embodiments described herein to support and/or to enable a cooling system incorporating adaptable fins as described herein. In addition to what is illustrated, other logic and circuits may be included in at least one embodiment, including additional graphics processors/cores, peripheral interface controllers, or general-purpose processor cores.

[0186] FIGS. 10B-10C are block diagrams illustrating exemplary graphics processors for use within an SoC, according to embodiments described herein, to support and/or to enable a cooling system incorporating adaptable fins as described herein. In an example, the graphic processors may be used in the intelligent control of a cooling system incorporating adaptable fins because of existing math engines capable of faster processing of multi-level neural networks. FIG. 10B illustrates an exemplary graphics processor 1010 of a system on a chip integrated circuit that may be fabricated using one or more IP cores, according to at least one embodiment. FIG. 10C illustrates an additional exemplary graphics processor 1040 of a system on a chip integrated circuit that may be fabricated using one or more IP cores, according to at least one embodiment. In at least one embodiment, graphics processor 1010 of FIG. 10A is a low power graphics processor core. In at least one embodiment, graphics processor 1040 of FIG. 10C is a higher performance graphics processor core. In at least one embodiment, each of graphics processors 1010, 1040 can be variants of graphics processor 1010 of FIG. 10A.

[0187] In at least one embodiment, graphics processor 1010 includes a vertex processor 1005 and one or more fragment processor(s) 1015A-1015N (such as 1015A, 1015B, 1015C, 1015D, through 1015N-1, and 1015N). In at least one embodiment, graphics processor 1010 can execute different shader programs via separate logic, such that vertex processor 1005 is optimized to execute operations for vertex shader programs, while one or more fragment processor(s) 1015A-1015N execute fragment (such as pixel) shading operations for fragment or pixel shader programs. In at least one embodiment, vertex processor 1005 performs a vertex processing stage of a 3D graphics pipeline and generates primitives and vertex data. In at least one embodiment, fragment processor(s) 1015A-1015N use primitive and vertex data generated by vertex processor

1005 to produce a framebuffer that is displayed on a display device. In at least one embodiment, fragment processor(s) 1015A-1015N are optimized to execute fragment shader programs as provided for in an OpenGL API, which may be used to perform similar operations as a pixel shader program as provided for in a Direct 3D API.

[0188] In at least one embodiment, graphics processor 1010 additionally includes one or more memory management units (MMUs) 1020A-1020B, cache(s) 1025A-1025B, and circuit interconnect(s) 1030A-1030B. In at least one embodiment, one or more MMU(s) 1020A-1020B provide for virtual to physical address mapping for graphics processor 1010, including for vertex processor 1005 and/or fragment processor(s) 1015A-1015N, which may reference vertex or image/texture data stored in memory, in addition to vertex or image/texture data stored in one or more cache(s) 1025A-1025B. In at least one embodiment, one or more MMU(s) 1020A-1020B may be synchronized with other MMUs within system, including one or more MMUs associated with one or more application processor(s) 1005, image processors 1015, and/or video processors 1020 of FIG. 10A, such that each processor 1005-1020 can participate in a shared or unified virtual memory system. In at least one embodiment, one or more circuit interconnect(s) 1030A-1030B enable graphics processor 1010 to interface with other IP cores within SoC, either via an internal bus of SoC or via a direct connection.

[0189] In at least one embodiment, graphics processor 1040 includes one or more MMU(s) 1020A-1020B, cache(s) 1025A-1025B, and circuit interconnect(s) 1030A-1030B of graphics processor 1010 of FIG. 10A. In at least one embodiment, graphics processor 1040 includes one or more shader core(s) 1055A-1055N (such as 1055A, 1055B, 1055C, 1055D, 1055E, 1055F, through 1055N-1, and 1055N), which provides for a unified shader core architecture in which a single core or type or core can execute all types of programmable shader code, including shader program code to implement vertex shaders, fragment shaders, and/or compute shaders. In at least one embodiment, a number of shader cores can vary. In at least one embodiment, graphics processor 1040 includes an inter-core task manager 1045, which acts as a thread dispatcher to dispatch execution threads to one or more shader cores 1055A-1055N and a tiling unit 1058 to accelerate tiling operations for tile-based rendering, in which rendering operations for a scene are subdivided in image space, for example to exploit local spatial coherence within a scene or to optimize use of internal caches.

[0190] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, inference and/or training logic 615 may be used in integrated circuit 10A and/or 10B for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0191] FIGS. 10D-10E illustrate additional exemplary graphics processor logic according to embodiments described herein to support and/or to enable a cooling system incorporating adaptable fins as described herein. FIG. 10D illustrates a graphics core 1000D that may be included within graphics processor 1010 of FIG. 10A, in at least one embodiment, and may be a unified shader core 1055A-1055N as in FIG. 10C in at least one embodiment. FIG. 10B illustrates a highly-parallel general-purpose graphics processing unit 1030 suitable for deployment on a multi-chip module in at least one embodiment.

[0192] In at least one embodiment, graphics core 1000D can include multiple slices 1001A-1001N or partition for each core, and a graphics processor can include multiple instances of graphics core 1000D. Slices 1001A-1001N can include support logic including a local instruction cache 1004A-1004N, a thread scheduler 1006A-1006N, a thread dispatcher 1008A-1008N, and a set of registers 1010A-1010N. In at least one embodiment, slices 1001A-1001N can include a set of additional function units (AFUs 1012A-1012N), floating-point units (FPU 1014A-1014N), integer arithmetic logic units (ALUs 1016-1016N), address computational units (ACU 1013A-1013N), double-precision floating-point units (DPFPU 1015A-1015N), and matrix processing units (MPU 1017A-1017N).

[0193] In at least one embodiment, FPUs 1014A-1014N can perform single-precision (32-bit) and half-precision (16-bit) floating point operations, while DPFPUs 1015A-1015N perform double precision (64-bit) floating point operations. In at least one embodiment, ALUs 1016A-1016N can perform variable precision integer operations at 8-bit, 16-bit, and 32-bit precision, and can be configured for mixed precision operations. In at least one embodiment, MPUs 1017A-1017N can also be configured for mixed precision matrix operations, including half-precision floating point and 8-bit integer operations. In at least one embodiment, MPUs 1017A-

1017N can perform a variety of matrix operations to accelerate machine learning application frameworks, including enabling support for accelerated general matrix to matrix multiplication (GEMM). In at least one embodiment, AFUs 1012A-1012N can perform additional logic operations not supported by floating-point or integer units, including trigonometric operations (such as Sine, Cosine, etc.).

[0194] As discussed elsewhere in this disclosure, inference and/or training logic 615 (referenced at least in FIGS. 6B, 6C) may be used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, inference and/or training logic 615 may be used in graphics core 1000D for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0195] FIG. 11A is a block diagram illustrating a computing system 1100A according to at least one embodiment. In at least one embodiment, computing system 1100A includes a processing subsystem 1101 having one or more processor(s) 1102 and a system memory 1104 communicating via an interconnection path that may include a memory hub 1105. In at least one embodiment, memory hub 1105 may be a separate component within a chipset component or may be integrated within one or more processor(s) 1102. In at least one embodiment, memory hub 1105 couples with an I/O subsystem 1111 via a communication link 1106. In at least one embodiment, I/O subsystem 1111 includes an I/O hub 1107 that can enable computing system 1100A to receive input from one or more input device(s) 1108. In at least one embodiment, I/O hub 1107 can enable a display controller, which may be included in one or more processor(s) 1102, to provide outputs to one or more display device(s) 1110A. In at least one embodiment, one or more display device(s) 1110A coupled with I/O hub 1107 can include a local, internal, or embedded display device.

[0196] In at least one embodiment, processing subsystem 1101 includes one or more parallel processor(s) 1112 coupled to memory hub 1105 via a bus or other communication link 1113. In at least one embodiment, communication link 1113 may be one of any number of standards based communication link technologies or protocols, such as but not limited to PCI Express, or

may be a vendor specific communications interface or communications fabric. In at least one embodiment, one or more parallel processor(s) 1112 form a computationally focused parallel or vector processing system that can include a large number of processing cores and/or processing clusters, such as a many integrated core (MIC) processor. In at least one embodiment, one or more parallel processor(s) 1112 form a graphics processing subsystem that can output pixels to one of one or more display device(s) 1110A coupled via I/O Hub 1107. In at least one embodiment, one or more parallel processor(s) 1112 can also include a display controller and display interface (not shown) to enable a direct connection to one or more display device(s) 1110B.

[0197] In at least one embodiment, a system storage unit 1114 can connect to I/O hub 1107 to provide a storage mechanism for computing system 1100A. In at least one embodiment, an I/O switch 1116 can be used to provide an interface mechanism to enable connections between I/O hub 1107 and other components, such as a network adapter 1118 and/or wireless network adapter 1119 that may be integrated into a platform(s), and various other devices that can be added via one or more add-in device(s) 1120. In at least one embodiment, network adapter 1118 can be an Ethernet adapter or another wired network adapter. In at least one embodiment, wireless network adapter 1119 can include one or more of a Wi-Fi, Bluetooth, near field communication (NFC), or other network device that includes one or more wireless radios.

[0198] In at least one embodiment, computing system 1100A can include other components not explicitly shown, including USB or other port connections, optical storage drives, video capture devices, and so on, may also be connected to I/O hub 1107. In at least one embodiment, communication paths interconnecting various components in FIG. 11A may be implemented using any suitable protocols, such as PCI (Peripheral Component Interconnect) based protocols (such as PCI-Express), or other bus or point-to-point communication interfaces and/or protocol(s), such as NV-Link high-speed interconnect, or interconnect protocols.

[0199] In at least one embodiment, one or more parallel processor(s) 1112 incorporate circuitry optimized for graphics and video processing, including, for example, video output circuitry, and constitutes a graphics processing unit (GPU). In at least one embodiment, one or more parallel processor(s) 1112 incorporate circuitry optimized for general purpose processing. In at least one embodiment, components of computing system 1100A may be integrated with one or more other

system elements on a single integrated circuit. In at least one embodiment, in at least one embodiment, one or more parallel processor(s) 1112, memory hub 1105, processor(s) 1102, and I/O hub 1107 can be integrated into a system on chip (SoC) integrated circuit. In at least one embodiment, components of computing system 1100A can be integrated into a single package to form a system in package (SIP) configuration. In at least one embodiment, at least a portion of components of computing system 1100A can be integrated into a multi-chip module (MCM), which can be interconnected with other multi-chip modules into a modular computing system.

[0200] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, inference and/or training logic 615 may be used in system FIG. 11A for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

PROCESSORS

[0201] FIG. 11B illustrates a parallel processor 1100B according to at least one embodiment. In at least one embodiment, various components of parallel processor 1100B may be implemented using one or more integrated circuit devices, such as programmable processors, application specific integrated circuits (ASICs), or field programmable gate arrays (FPGA). In at least one embodiment, illustrated parallel processor 11100B 100 is a variant of one or more parallel processor(s) 1112 shown in FIG. 11B according to an exemplary embodiment.

[0202] In at least one embodiment, parallel processor 1100B includes a parallel processing unit 1102. In at least one embodiment, parallel processing unit 1102 includes an I/O unit 1104 that enables communication with other devices, including other instances of parallel processing unit 1102. In at least one embodiment, I/O unit 1104 may be directly connected to other devices. In at least one embodiment, I/O unit 1104 connects with other devices via use of a hub or switch interface, such as memory hub 1105. In at least one embodiment, connections between memory hub 1105 and I/O unit 1104 form a communication link 1113. In at least one embodiment, I/O unit 1104 connects with a host interface 1106 and a memory crossbar 1116, where host interface

1106 receives commands directed to performing processing operations and memory crossbar 1116 receives commands directed to performing memory operations.

[0203] In at least one embodiment, when host interface 1106 receives a command buffer via I/O unit 1104, host interface 1106 can direct work operations to perform those commands to a front end 1108. In at least one embodiment, front end 1108 couples with a scheduler 1110, which is configured to distribute commands or other work items to a processing cluster array 1112. In at least one embodiment, scheduler 1110 ensures that processing cluster array 1112 is properly configured and in a valid state before tasks are distributed to processing cluster array 1112. In at least one embodiment, scheduler 1110 is implemented via firmware logic executing on a microcontroller. In at least one embodiment, microcontroller implemented scheduler 1110 is configurable to perform complex scheduling and work distribution operations at coarse and fine granularity, enabling rapid preemption and context switching of threads executing on processing array 1112. In at least one embodiment, host software can prove workloads for scheduling on processing array 1112 via one of multiple graphics processing doorbells. In at least one embodiment, workloads can then be automatically distributed across processing array 1112 by scheduler 1110 logic within a microcontroller including scheduler 1110.

[0204] In at least one embodiment, processing cluster array 1112 can include up to "N" processing clusters (such as cluster 1114A, cluster 1114B, through cluster 1114N). In at least one embodiment, each cluster 1114A-1114N of processing cluster array 1112 can execute a large number of concurrent threads. In at least one embodiment, scheduler 1110 can allocate work to clusters 1114A-1114N of processing cluster array 1112 using various scheduling and/or work distribution algorithms, which may vary depending on workload arising for each type of program or computation. In at least one embodiment, scheduling can be handled dynamically by scheduler 1110, or can be assisted in part by compiler logic during compilation of program logic configured for execution by processing cluster array 1112. In at least one embodiment, different clusters 1114A-1114N of processing cluster array 1112 can be allocated for processing different types of programs or for performing different types of computations.

[0205] In at least one embodiment, processing cluster array 1112 can be configured to perform various types of parallel processing operations. In at least one embodiment, processing cluster array 1112 is configured to perform general-purpose parallel compute operations. In at least one

embodiment, in at least one embodiment, processing cluster array 1112 can include logic to execute processing tasks including filtering of video and/or audio data, performing modeling operations, including physics operations, and performing data transformations.

[0206] In at least one embodiment, processing cluster array 1112 is configured to perform parallel graphics processing operations. In at least one embodiment, processing cluster array 1112 can include additional logic to support execution of such graphics processing operations, including, but not limited to texture sampling logic to perform texture operations, as well as tessellation logic and other vertex processing logic. In at least one embodiment, processing cluster array 1112 can be configured to execute graphics processing related shader programs such as but not limited to vertex shaders, tessellation shaders, geometry shaders, and pixel shaders. In at least one embodiment, parallel processing unit 1102 can transfer data from system memory via I/O unit 1104 for processing. In at least one embodiment, during processing, transferred data can be stored to on-chip memory (such as parallel processor memory 1122) during processing, then written back to system memory.

[0207] In at least one embodiment, when parallel processing unit 1102 is used to perform graphics processing, scheduler 1110 can be configured to divide a processing workload into approximately equal sized tasks, to better enable distribution of graphics processing operations to multiple clusters 1114A-1114N of processing cluster array 1112. In at least one embodiment, portions of processing cluster array 1112 can be configured to perform different types of processing. In at least one embodiment, in at least one embodiment, a first portion may be configured to perform vertex shading and topology generation, a second portion may be configured to perform tessellation and geometry shading, and a third portion may be configured to perform pixel shading or other screen space operations, to produce a rendered image for display if a simulation of valve control for a cooling system incorporating adaptable fins is required. In at least one embodiment, intermediate data produced by one or more of clusters 1114A-1114N may be stored in buffers to allow intermediate data to be transmitted between clusters 1114A-1114N for further processing.

[0208] In at least one embodiment, processing cluster array 1112 can receive processing tasks to be executed via scheduler 1110, which receives commands defining processing tasks from front end 1108. In at least one embodiment, processing tasks can include indices of data to be

processed, such as surface (patch) data, primitive data, vertex data, and/or pixel data, as well as state parameters and commands defining how data is to be processed (such as what program is to be executed). In at least one embodiment, scheduler 1110 may be configured to fetch indices corresponding to tasks or may receive indices from front end 1108. In at least one embodiment, front end 1108 can be configured to ensure processing cluster array 1112 is configured to a valid state before a workload specified by incoming command buffers (such as batch-buffers, push buffers, etc.) is initiated.

[0209] In at least one embodiment, each of one or more instances of parallel processing unit 1102 can couple with parallel processor memory 1122. In at least one embodiment, parallel processor memory 1122 can be accessed via memory crossbar 1116, which can receive memory requests from processing cluster array 1112 as well as I/O unit 1104. In at least one embodiment, memory crossbar 1116 can access parallel processor memory 1122 via a memory interface 1118. In at least one embodiment, memory interface 1118 can include multiple partition units (such as partition unit 1120A, partition unit 1120B, through partition unit 1120N) that can each couple to a portion (such as memory unit) of parallel processor memory 1122. In at least one embodiment, a number of partition units 1120A-1120N is configured to be equal to a number of memory units, such that a first partition unit 1120A has a corresponding first memory unit 1124A, a second partition unit 1120B has a corresponding memory unit 1124B, and a Nth partition unit 1120N has a corresponding Nth memory unit 1124N. In at least one embodiment, a number of partition units 1120A-1120N may not be equal to a number of memory devices.

[0210] In at least one embodiment, memory units 1124A-1124N can include various types of memory devices, including dynamic random access memory (DRAM) or graphics random access memory, such as synchronous graphics random access memory (SGRAM), including graphics double data rate (GDDR) memory. In at least one embodiment, memory units 1124A-1124N may also include 3D stacked memory, including but not limited to high bandwidth memory (HBM). In at least one embodiment, render targets, such as frame buffers or texture maps may be stored across memory units 1124A-1124N, allowing partition units 1120A-1120N to write portions of each render target in parallel to efficiently use available bandwidth of parallel processor memory 1122. In at least one embodiment, a local instance of parallel processor

memory 1122 may be excluded in favor of a unified memory design that utilizes system memory in conjunction with local cache memory.

[0211] In at least one embodiment, any one of clusters 1114A-1114N of processing cluster array 1112 can process data that will be written to any of memory units 1124A-1124N within parallel processor memory 1122. In at least one embodiment, memory crossbar 1116 can be configured to transfer an output of each cluster 1114A-1114N to any partition unit 1120A-1120N or to another cluster 1114A-1114N, which can perform additional processing operations on an output. In at least one embodiment, each cluster 1114A-1114N can communicate with memory interface 1118 through memory crossbar 1116 to read from or write to various external memory devices. In at least one embodiment, memory crossbar 1116 has a connection to memory interface 1118 to communicate with I/O unit 1104, as well as a connection to a local instance of parallel processor memory 1122, enabling processing units within different processing clusters 1114A-1114N to communicate with system memory or other memory that is not local to parallel processing unit 1102. In at least one embodiment, memory crossbar 1116 can use virtual channels to separate traffic streams between clusters 1114A-1114N and partition units 1120A-1120N.

[0212] In at least one embodiment, multiple instances of parallel processing unit 1102 can be provided on a single add-in card, or multiple add-in cards can be interconnected. In at least one embodiment, different instances of parallel processing unit 1102 can be configured to inter-operate even if different instances have different numbers of processing cores, different amounts of local parallel processor memory, and/or other configuration differences. In at least one embodiment, in at least one embodiment, some instances of parallel processing unit 1102 can include higher precision floating point units relative to other instances. In at least one embodiment, systems incorporating one or more instances of parallel processing unit 1102 or parallel processor 1100B can be implemented in a variety of configurations and form factors, including but not limited to desktop, laptop, or handheld personal computers, servers, workstations, game consoles, and/or embedded systems.

[0213] FIG. 11C is a block diagram of a partition unit 1120 according to at least one embodiment. In at least one embodiment, partition unit 1120 is an instance of one of partition units 1120A-1120N of FIG. 11B. In at least one embodiment, partition unit 1120 includes an L2

cache 1121, a frame buffer interface 1125, and a raster operations unit ("ROP") 1126. L2 cache 1121 is a read/write cache that is configured to perform load and store operations received from memory crossbar 1116 and ROP 1126. In at least one embodiment, read misses and urgent write-back requests are output by L2 cache 1121 to frame buffer interface 1125 for processing. In at least one embodiment, updates can also be sent to a frame buffer via frame buffer interface 1125 for processing. In at least one embodiment, frame buffer interface 1125 interfaces with one of memory units in parallel processor memory, such as memory units 1124A-1124N of FIG. 11B (such as within parallel processor memory 1122).

[0214] In at least one embodiment, ROP 1126 is a processing unit that performs raster operations such as stencil, z test, blending, and so forth. In at least one embodiment, ROP 1126 then outputs processed graphics data that is stored in graphics memory. In at least one embodiment, ROP 1126 includes compression logic to compress depth or color data that is written to memory and decompress depth or color data that is read from memory. In at least one embodiment, compression logic can be lossless compression logic that makes use of one or more of multiple compression algorithms. Compression logic that is performed by ROP 1126 can vary based on statistical characteristics of data to be compressed. In at least one embodiment, in at least one embodiment, delta color compression is performed on depth and color data on a per-tile basis.

[0215] In at least one embodiment, ROP 1126 is included within each processing cluster (such as cluster 1114A-1114N of FIG. 11B) instead of within partition unit 1120. In at least one embodiment, read and write requests for pixel data are transmitted over memory crossbar 1116 instead of pixel fragment data. In at least one embodiment, processed graphics data may be displayed on a display device, such as one of one or more display device(s) 1110 of FIG. 11, routed for further processing by processor(s) 1102, or routed for further processing by one of processing entities within parallel processor 1100B of FIG. 11B.

[0216] FIG. 11D is a block diagram of a processing cluster 1114 within a parallel processing unit according to at least one embodiment. In at least one embodiment, a processing cluster is an instance of one of processing clusters 1114A-1114N of FIG. 11B. In at least one embodiment, one of more of processing cluster(s) 1114 can be configured to execute many threads in parallel, where "thread" refers to an instance of a particular program executing on a particular set of input

data. In at least one embodiment, single-instruction, multiple-data (SIMD) instruction issue techniques are used to support parallel execution of a large number of threads without providing multiple independent instruction units. In at least one embodiment, single-instruction, multiple-thread (SIMT) techniques are used to support parallel execution of a large number of synchronized threads, using a common instruction unit configured to issue instructions to a set of processing engines within each one of processing clusters.

[0217] In at least one embodiment, operation of processing cluster 1114 can be controlled via a pipeline manager 1132 that distributes processing tasks to SIMT parallel processors. In at least one embodiment, pipeline manager 1132 receives instructions from scheduler 1110 of FIG. 11B and manages execution of those instructions via a graphics multiprocessor 1134 and/or a texture unit 1136. In at least one embodiment, graphics multiprocessor 1134 is an exemplary instance of a SIMT parallel processor. However, in at least one embodiment, various types of SIMT parallel processors of differing architectures may be included within processing cluster 1114. In at least one embodiment, one or more instances of graphics multiprocessor 1134 can be included within a processing cluster 1114. In at least one embodiment, graphics multiprocessor 1134 can process data and a data crossbar 1140 can be used to distribute processed data to one of multiple possible destinations, including other shader units. In at least one embodiment, pipeline manager 1132 can facilitate distribution of processed data by specifying destinations for processed data to be distributed vis data crossbar 1140.

[0218] In at least one embodiment, each graphics multiprocessor 1134 within processing cluster 1114 can include an identical set of functional execution logic (such as arithmetic logic units, load-store units, etc.). In at least one embodiment, functional execution logic can be configured in a pipelined manner in which new instructions can be issued before previous instructions are complete. In at least one embodiment, functional execution logic supports a variety of operations including integer and floating point arithmetic, comparison operations, Boolean operations, bit-shifting, and computation of various algebraic functions. In at least one embodiment, same functional-unit hardware can be leveraged to perform different operations and any combination of functional units may be present.

[0219] In at least one embodiment, instructions transmitted to processing cluster 1114 constitute a thread. In at least one embodiment, a set of threads executing across a set of parallel

processing engines is a thread group. In at least one embodiment, thread group executes a program on different input data. In at least one embodiment, each thread within a thread group can be assigned to a different processing engine within a graphics multiprocessor 1134. In at least one embodiment, a thread group may include fewer threads than a number of processing engines within graphics multiprocessor 1134. In at least one embodiment, when a thread group includes fewer threads than a number of processing engines, one or more processing engines may be idle during cycles in which that thread group is being processed. In at least one embodiment, a thread group may also include more threads than a number of processing engines within graphics multiprocessor 1134. In at least one embodiment, when a thread group includes more threads than processing engines within graphics multiprocessor 1134, processing can be performed over consecutive clock cycles. In at least one embodiment, multiple thread groups can be executed concurrently on a graphics multiprocessor 1134.

[0220] In at least one embodiment, graphics multiprocessor 1134 includes an internal cache memory to perform load and store operations. In at least one embodiment, graphics multiprocessor 1134 can forego an internal cache and use a cache memory (such as L1 cache 1148) within processing cluster 1114. In at least one embodiment, each graphics multiprocessor 1134 also has access to L2 caches within partition units (such as partition units 1120A-1120N of FIG. 11B) that are shared among all processing clusters 1114 and may be used to transfer data between threads. In at least one embodiment, graphics multiprocessor 1134 may also access off-chip global memory, which can include one or more of local parallel processor memory and/or system memory. In at least one embodiment, any memory external to parallel processing unit 1102 may be used as global memory. In at least one embodiment, processing cluster 1114 includes multiple instances of graphics multiprocessor 1134 can share common instructions and data, which may be stored in L1 cache 1148.

[0221] In at least one embodiment, each processing cluster 1114 may include a memory management unit ("MMU") 1145 that is configured to map virtual addresses into physical addresses. In at least one embodiment, one or more instances of MMU 1145 may reside within memory interface 1118 of FIG. 11B. In at least one embodiment, MMU 1145 includes a set of page table entries (PTEs) used to map a virtual address to a physical address of a tile and, in at least one embodiment, a cache line index. In at least one embodiment, MMU 1145 may include

address translation lookaside buffers (TLB) or caches that may reside within graphics multiprocessor 1134 or L1 cache or processing cluster 1114. In at least one embodiment, physical address is processed to distribute surface data access locality to allow efficient request interleaving among partition units. In at least one embodiment, cache line index may be used to determine whether a request for a cache line is a hit or miss.

[0222] In at least one embodiment, a processing cluster 1114 may be configured such that each graphics multiprocessor 1134 is coupled to a texture unit 1136 for performing texture mapping operations, such as determining texture sample positions, reading texture data, and filtering texture data. In at least one embodiment, texture data is read from an internal texture L1 cache (not shown) or from an L1 cache within graphics multiprocessor 1134 and is fetched from an L2 cache, local parallel processor memory, or system memory, as needed. In at least one embodiment, each graphics multiprocessor 1134 outputs processed tasks to data crossbar 1140 to provide processed task(s) to another processing cluster 1114 for further processing or to store processed task(s) in an L2 cache, local parallel processor memory, or system memory via memory crossbar 1116. In at least one embodiment, preROP 1142 (pre-raster operations unit) is configured to receive data from graphics multiprocessor 1134, direct data to ROP units, which may be located with partition units as described herein (such as partition units 1120A-1120N of FIG. 11B). In at least one embodiment, PreROP 1142 unit can perform optimizations for color blending, organize pixel color data, and perform address translations.

[0223] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, inference and/or training logic 615 may be used in graphics processing cluster 1114 for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0224] FIG. 11E shows a graphics multiprocessor 1134 according to at least one embodiment. In at least one embodiment, graphics multiprocessor 1134 couples with pipeline manager 1132 of processing cluster 1114. In at least one embodiment, graphics multiprocessor 1134 has an execution pipeline including but not limited to an instruction cache 1152, an instruction unit

1154, an address mapping unit 1156, a register file 1158, one or more general purpose graphics processing unit (GPGPU) cores 1162, and one or more load/store units 1166. GPGPU core(s) 1162 and load/store unit(s) 1166 are coupled with cache memory 1172 and shared memory 1170 via a memory and cache interconnect 1168.

[0225] In at least one embodiment, instruction cache 1152 receives a stream of instructions to execute from pipeline manager 1132. In at least one embodiment, instructions are cached in instruction cache 1152 and dispatched for execution by instruction unit 1154. In at least one embodiment, instruction unit 1154 can dispatch instructions as thread groups (such as warps), with each thread group assigned to a different execution unit within GPGPU core(s) 1162. In at least one embodiment, an instruction can access any of a local, shared, or global address space by specifying an address within a unified address space. In at least one embodiment, address mapping unit 1156 can be used to translate addresses in a unified address space into a distinct memory address that can be accessed by load/store unit(s) 1166.

[0226] In at least one embodiment, register file 1158 provides a set of registers for functional units of graphics multiprocessor 1134. In at least one embodiment, register file 1158 provides temporary storage for operands connected to data paths of functional units (such as GPGPU cores 1162, load/store units 1166) of graphics multiprocessor 1134. In at least one embodiment, register file 1158 is divided between each of functional units such that each functional unit is allocated a dedicated portion of register file 1158. In at least one embodiment, register file 1158 is divided between different warps being executed by graphics multiprocessor 1134.

[0227] In at least one embodiment, GPGPU cores 1162 can each include floating point units (FPUs) and/or integer arithmetic logic units (ALUs) that are used to execute instructions of graphics multiprocessor 1134. GPGPU cores 1162 can be similar in architecture or can differ in architecture. In at least one embodiment, a first portion of GPGPU cores 1162 include a single precision FPU and an integer ALU while a second portion of GPGPU cores include a double precision FPU. In at least one embodiment, FPUs can implement IEEE 754-2008 standard for floating point arithmetic or enable variable precision floating point arithmetic. In at least one embodiment, graphics multiprocessor 1134 can additionally include one or more fixed function or special function units to perform specific functions such as copy rectangle or pixel blending

operations. In at least one embodiment one or more of GPGPU cores can also include fixed or special function logic.

[0228] In at least one embodiment, GPGPU cores 1162 include SIMD logic capable of performing a single instruction on multiple sets of data. In at least one embodiment GPGPU cores 1162 can physically execute SIMD4, SIMD8, and SIMD16 instructions and logically execute SIMD1, SIMD2, and SIMD32 instructions. In at least one embodiment, SIMD instructions for GPGPU cores can be generated at compile time by a shader compiler or automatically generated when executing programs written and compiled for single program multiple data (SPMD) or SIMT architectures. In at least one embodiment, multiple threads of a program configured for an SIMT execution model can executed via a single SIMD instruction. In at least one embodiment, in at least one embodiment, eight SIMT threads that perform same or similar operations can be executed in parallel via a single SIMD8 logic unit.

[0229] In at least one embodiment, memory and cache interconnect 1168 is an interconnect network that connects each functional unit of graphics multiprocessor 1134 to register file 1158 and to shared memory 1170. In at least one embodiment, memory and cache interconnect 1168 is a crossbar interconnect that allows load/store unit 1166 to implement load and store operations between shared memory 1170 and register file 1158. In at least one embodiment, register file 1158 can operate at a same frequency as GPGPU cores 1162, thus data transfer between GPGPU cores 1162 and register file 1158 is very low latency. In at least one embodiment, shared memory 1170 can be used to enable communication between threads that execute on functional units within graphics multiprocessor 1134. In at least one embodiment, cache memory 1172 can be used as a data cache for example, to cache texture data communicated between functional units and texture unit 1136. In at least one embodiment, shared memory 1170 can also be used as a program managed cache. In at least one embodiment, threads executing on GPGPU cores 1162 can programmatically store data within shared memory in addition to automatically cached data that is stored within cache memory 1172.

[0230] In at least one embodiment, a parallel processor or GPGPU as described herein is communicatively coupled to host/processor cores to accelerate graphics operations, machine-learning operations, pattern analysis operations, and various general purpose GPU (GPGPU) functions. In at least one embodiment, GPU may be communicatively coupled to host

processor/cores over a bus or other interconnect (such as a high speed interconnect such as PCIe or NVLink). In at least one embodiment, GPU may be integrated on same package or chip as cores and communicatively coupled to cores over an internal processor bus/interconnect (in at least one embodiment, internal to package or chip). In at least one embodiment, regardless of manner in which GPU is connected, processor cores may allocate work to GPU in form of sequences of commands/instructions contained in a work descriptor. In at least one embodiment, GPU then uses dedicated circuitry/logic for efficiently processing these commands/instructions.

[0231] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, inference and/or training logic 615 may be used in graphics multiprocessor 1134 for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0232] FIG. 12A illustrates a multi-GPU computing system 1200A, according to at least one embodiment. In at least one embodiment, multi-GPU computing system 1200A can include a processor 1202 coupled to multiple general purpose graphics processing units (GPGPUs) 1206A-D via a host interface switch 1204. In at least one embodiment, host interface switch 1204 is a PCI express switch device that couples processor 1202 to a PCI express bus over which processor 1202 can communicate with GPGPUs 1206A-D. GPGPUs 1206A-D can interconnect via a set of high-speed point to point GPU to GPU links 1216. In at least one embodiment, GPU to GPU links 1216 connect to each of GPGPUs 1206A-D via a dedicated GPU link. In at least one embodiment, P2P GPU links 1216 enable direct communication between each of GPGPUs 1206A-D without requiring communication over host interface bus 1204 to which processor 1202 is connected. In at least one embodiment, with GPU-to-GPU traffic directed to P2P GPU links 1216, host interface bus 1204 remains available for system memory access or to communicate with other instances of multi-GPU computing system 1200A, for example, via one or more network devices. While in at least one embodiment GPGPUs 1206A-D connect to processor 1202 via host interface switch 1204, in at least one embodiment processor 1202 includes direct support for P2P GPU links 1216 and can connect directly to GPGPUs 1206A-D.

[0233] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, inference and/or training logic 615 may be used in multi-GPU computing system 1200A for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0234] FIG. 12B is a block diagram of a graphics processor 1200B, according to at least one embodiment. In at least one embodiment, graphics processor 1200B includes a ring interconnect 1202, a pipeline front-end 1204, a media engine 1237, and graphics cores 1280A-1280N. In at least one embodiment, ring interconnect 1202 couples graphics processor 1200B to other processing units, including other graphics processors or one or more general-purpose processor cores. In at least one embodiment, graphics processor 1200B is one of many processors integrated within a multi-core processing system.

[0235] In at least one embodiment, graphics processor 1200B receives batches of commands via ring interconnect 1202. In at least one embodiment, incoming commands are interpreted by a command streamer 1203 in pipeline front-end 1204. In at least one embodiment, graphics processor 1200B includes scalable execution logic to perform 3D geometry processing and media processing via graphics core(s) 1280A-1280N. In at least one embodiment, for 3D geometry processing commands, command streamer 1203 supplies commands to geometry pipeline 1236. In at least one embodiment, for at least some media processing commands, command streamer 1203 supplies commands to a video front end 1234, which couples with a media engine 1237. In at least one embodiment, media engine 1237 includes a Video Quality Engine (VQE) 1230 for video and image post-processing and a multi-format encode/decode (MFX) 1233 engine to provide hardware-accelerated media data encode and decode. In at least one embodiment, geometry pipeline 1236 and media engine 1237 each generate execution threads for thread execution resources provided by at least one graphics core 1280A.

[0236] In at least one embodiment, graphics processor 1200B includes scalable thread execution resources featuring modular cores 1280A-1280N (sometimes referred to as core slices), each having multiple sub-cores 1250A-1250N, 1260A-1260N (sometimes referred to as core

sub-slices). In at least one embodiment, graphics processor 1200B can have any number of graphics cores 1280A through 1280N. In at least one embodiment, graphics processor 1200B includes a graphics core 1280A having at least a first sub-core 1250A and a second sub-core 1260A. In at least one embodiment, graphics processor 1200B is a low power processor with a single sub-core (such as 1250A). In at least one embodiment, graphics processor 1200B includes multiple graphics cores 1280A-1280N, each including a set of first sub-cores 1250A-1250N and a set of second sub-cores 1260A-1260N. In at least one embodiment, each sub-core in first sub-cores 1250A-1250N includes at least a first set of execution units 1252A-1252N and media/texture samplers 1254A-1254N. In at least one embodiment, each sub-core in second sub-cores 1260A-1260N includes at least a second set of execution units 1262A-1262N and samplers 1264A-1264N. In at least one embodiment, each sub-core 1250A-1250N, 1260A-1260N shares a set of shared resources 1270A-1270N. In at least one embodiment, shared resources include shared cache memory and pixel operation logic.

[0237] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, inference and/or training logic 615 may be used in graphics processor 1200B for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0238] FIG. 13 is a block diagram illustrating micro-architecture for a processor 1300 that may include logic circuits to perform instructions, according to at least one embodiment. In at least one embodiment, processor 1300 may perform instructions, including x86 instructions, ARM instructions, specialized instructions for application-specific integrated circuits (ASICs), etc. In at least one embodiment, processor 1300 may include registers to store packed data, such as 64-bit wide MMX™ registers in microprocessors enabled with MMX technology from Intel Corporation of Santa Clara, Calif. In at least one embodiment, MMX registers, available in both integer and floating point forms, may operate with packed data elements that accompany single instruction, multiple data ("SIMD") and streaming SIMD extensions ("SSE") instructions. In at least one embodiment, 128-bit wide XMM registers relating to SSE2, SSE3, SSE4, AVX, or

beyond (referred to generically as "SSEx") technology may hold such packed data operands. In at least one embodiment, processor 1300 may perform instructions to accelerate machine learning or deep learning algorithms, training, or inferencing.

[0239] In at least one embodiment, processor 1300 includes an in-order front end ("front end") 1301 to fetch instructions to be executed and prepare instructions to be used later in processor pipeline. In at least one embodiment, front end 1301 may include several units. In at least one embodiment, an instruction prefetcher 1326 fetches instructions from memory and feeds instructions to an instruction decoder 1328 which in turn decodes or interprets instructions. In at least one embodiment, in at least one embodiment, instruction decoder 1328 decodes a received instruction into one or more operations called "micro-instructions" or "micro-operations" (also called "micro ops"or "uops") that machine may execute. In at least one embodiment, instruction decoder 1328 parses instruction into an opcode and corresponding data and control fields that may be used by micro-architecture to perform operations in accordance with at least one embodiment. In at least one embodiment, a trace cache 1330 may assemble decoded uops into program ordered sequences or traces in a uop queue 1334 for execution. In at least one embodiment, when trace cache 1330 encounters a complex instruction, a microcode ROM 1332 provides uops needed to complete operation.

[0240] In at least one embodiment, some instructions may be converted into a single micro-op, whereas others need several micro-ops to complete full operation. In at least one embodiment, if more than four micro-ops are needed to complete an instruction, instruction decoder 1328 may access microcode ROM 1332 to perform instruction. In at least one embodiment, an instruction may be decoded into a small number of micro-ops for processing at instruction decoder 1328. In at least one embodiment, an instruction may be stored within microcode ROM 1332 should a number of micro-ops be needed to accomplish operation. In at least one embodiment, trace cache 1330 refers to an entry point programmable logic array ("PLA") to determine a correct micro-instruction pointer for reading microcode sequences to complete one or more instructions from microcode ROM 1332 in accordance with at least one embodiment. In at least one embodiment, after microcode ROM 1332 finishes sequencing micro-ops for an instruction, front end 1301 of machine may resume fetching micro-ops from trace cache 1330.

[0241] In at least one embodiment, out-of-order execution engine ("out of order engine") 1303 may prepare instructions for execution. In at least one embodiment, out-of-order execution logic has a number of buffers to smooth out and re-order flow of instructions to optimize performance as they go down pipeline and get scheduled for execution. In at least one embodiment, out-of-order execution engine 1303 includes, without limitation, an allocator/register renamer 1340, a memory uop queue 1342, an integer/floating point uop queue 1344, a memory scheduler 1346, a fast scheduler 1302, a slow/general floating point scheduler ("slow/general FP scheduler") 1304, and a simple floating point scheduler ("simple FP scheduler") 1306. In at least one embodiment, fast schedule 1302, slow/general floating point scheduler 1304, and simple floating point scheduler 1306 are also collectively referred to herein as "uop schedulers 1302, 1304, 1306." In at least one embodiment, allocator/register renamer 1340 allocates machine buffers and resources that each uop needs in order to execute. In at least one embodiment, allocator/register renamer 1340 renames logic registers onto entries in a register file. In at least one embodiment, allocator/register renamer 1340 also allocates an entry for each uop in one of two uop queues, memory uop queue 1342 for memory operations and integer/floating point uop queue 1344 for non-memory operations, in front of memory scheduler 1346 and uop schedulers 1302, 1304, 1306. In at least one embodiment, uop schedulers 1302, 1304, 1306 determine when a uop is ready to execute based on readiness of their dependent input register operand sources and availability of execution resources uops need to complete their operation. In at least one embodiment, fast scheduler 1302 of at least one embodiment may schedule on each half of main clock cycle while slow/general floating point scheduler 1304 and simple floating point scheduler 1306 may schedule once per main processor clock cycle. In at least one embodiment, uop schedulers 1302, 1304, 1306 arbitrate for dispatch ports to schedule uops for execution.

[0242] In at least one embodiment, execution block 1311 includes, without limitation, an integer register file/bypass network 1308, a floating point register file/bypass network ("FP register file/bypass network") 1310, address generation units ("AGUs") 1312 and 1314, fast Arithmetic Logic Units (ALUs) ("fast ALUs") 1316 and 1318, a slow Arithmetic Logic Unit ("slow ALU") 1320, a floating point ALU ("FP") 1322, and a floating point move unit ("FP move") 1324. In at least one embodiment, integer register file/bypass network 1308 and floating point register file/bypass network 1310 are also referred to herein as "register files 1308, 1310." In at least one embodiment, AGUs 1312 and 1314, fast ALUs 1316 and 1318, slow ALU 1320,

floating point ALU 1322, and floating point move unit 1324 are also referred to herein as "execution units 1312, 1314, 1316, 1318, 1320, 1322, and 1324." In at least one embodiment, execution block b11 may include, without limitation, any number (including zero) and type of register files, bypass networks, address generation units, and execution units, in any combination.

[0243] In at least one embodiment, register files 1308, 1310 may be arranged between uop schedulers 1302, 1304, 1306, and execution units 1312, 1314, 1316, 1318, 1320, 1322, and 1324. In at least one embodiment, integer register file/bypass network 1308 performs integer operations. In at least one embodiment, floating point register file/bypass network 1310 performs floating point operations. In at least one embodiment, each of register files 1308, 1310 may include, without limitation, a bypass network that may bypass or forward just completed results that have not yet been written into register file to new dependent uops. In at least one embodiment, register files 1308, 1310 may communicate data with each other. In at least one embodiment, integer register file/bypass network 1308 may include, without limitation, two separate register files, one register file for low-order thirty-two bits of data and a second register file for high order thirty-two bits of data. In at least one embodiment, floating point register file/bypass network 1310 may include, without limitation, 128-bit wide entries because floating point instructions typically have operands from 64 to 128 bits in width.

[0244] In at least one embodiment, execution units 1312, 1314, 1316, 1318, 1320, 1322, 1324 may execute instructions. In at least one embodiment, register files 1308, 1310 store integer and floating point data operand values that micro-instructions need to execute. In at least one embodiment, processor 1300 may include, without limitation, any number and combination of execution units 1312, 1314, 1316, 1318, 1320, 1322, 1324. In at least one embodiment, floating point ALU 1322 and floating point move unit 1324, may execute floating point, MMX, SIMD, AVX and SSE, or other operations, including specialized machine learning instructions. In at least one embodiment, floating point ALU 1322 may include, without limitation, a 64-bit by 64-bit floating point divider to execute divide, square root, and remainder micro ops. In at least one embodiment, instructions involving a floating point value may be handled with floating point hardware. In at least one embodiment, ALU operations may be passed to fast ALUs 1316, 1318. In at least one embodiment, fast ALUS 1316, 1318 may execute fast operations with an effective latency of half a clock cycle. In at least one embodiment, most complex integer operations go to

slow ALU 1320 as slow ALU 1320 may include, without limitation, integer execution hardware for long-latency type of operations, such as a multiplier, shifts, flag logic, and branch processing. In at least one embodiment, memory load/store operations may be executed by AGUS 1312, 1314. In at least one embodiment, fast ALU 1316, fast ALU 1318, and slow ALU 1320 may perform integer operations on 64-bit data operands. In at least one embodiment, fast ALU 1316, fast ALU 1318, and slow ALU 1320 may be implemented to support a variety of data bit sizes including sixteen, thirty-two, 128, 256, etc. In at least one embodiment, floating point ALU 1322 and floating point move unit 1324 may be implemented to support a range of operands having bits of various widths. In at least one embodiment, floating point ALU 1322 and floating point move unit 1324 may operate on 128-bit wide packed data operands in conjunction with SIMD and multimedia instructions.

[0245] In at least one embodiment, uop schedulers 1302, 1304, 1306, dispatch dependent operations before parent load has finished executing. In at least one embodiment, as uops may be speculatively scheduled and executed in processor 1300, processor 1300 may also include logic to handle memory misses. In at least one embodiment, if a data load misses in data cache, there may be dependent operations in flight in pipeline that have left scheduler with temporarily incorrect data. In at least one embodiment, a replay mechanism tracks and re-executes instructions that use incorrect data. In at least one embodiment, dependent operations might need to be replayed and independent ones may be allowed to complete. In at least one embodiment, schedulers and replay mechanism of at least one embodiment of a processor may also be designed to catch instruction sequences for text string comparison operations.

[0246] In at least one embodiment, registers may refer to on-board processor storage locations that may be used as part of instructions to identify operands. In at least one embodiment, registers may be those that may be usable from outside of processor (from a programmer's perspective). In at least one embodiment, registers might not be limited to a particular type of circuit. Rather, in at least one embodiment, a register may store data, provide data, and perform functions described herein. In at least one embodiment, registers described herein may be implemented by circuitry within a processor using any number of different techniques, such as dedicated physical registers, dynamically allocated physical registers using register renaming, combinations of dedicated and dynamically allocated physical registers, etc. In at least one

embodiment, integer registers store 32-bit integer data. A register file of at least one embodiment also contains eight multimedia SIMD registers for packed data.

[0247] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment portions or all of inference and/or training logic 615 may be incorporated into execution block 1311 and other memory or registers shown or not shown. In at least one embodiment, in at least one embodiment, training and/or inferencing techniques described herein may use one or more of ALUs illustrated in execution block 1311. Moreover, weight parameters may be stored in on-chip or off-chip memory and/or registers (shown or not shown) that configure ALUs of execution block 1311 to perform one or more machine learning algorithms, neural network architectures, use cases, or training techniques described herein.

[0248] FIG. 14 illustrates a deep learning application processor 1400, according to at least one embodiment. In at least one embodiment, deep learning application processor 1400 uses instructions that, if executed by deep learning application processor 1400, cause deep learning application processor 1400 to perform some or all of processes and techniques described throughout this disclosure. In at least one embodiment, deep learning application processor 1400 is an application-specific integrated circuit (ASIC). In at least one embodiment, application processor 1400 performs matrix multiply operations either "hard-wired" into hardware as a result of performing one or more instructions or both. In at least one embodiment, deep learning application processor 1400 includes, without limitation, processing clusters 1410(1)-1410(12), Inter-Chip Links ("ICLs") 1420(1)-1420(12), Inter-Chip Controllers ("ICCs") 1430(1)-1430(2), memory controllers ("Mem Ctrlrs") 1442(1)-1442(4), high bandwidth memory physical layer ("HBM PHY") 1444(1)-1444(4), a management-controller central processing unit ("management-controller CPU") 1450, a Serial Peripheral Interface, Inter-Integrated Circuit, and General Purpose Input/Output block ("SPI, I2C, GPIO"), a peripheral component interconnect express controller and direct memory access block ("PCIe Controller and DMA") 1470, and a sixteen-lane peripheral component interconnect express port ("PCI Express x 16") 1480.

[0249] In at least one embodiment, processing clusters 1410 may perform deep learning operations, including inference or prediction operations based on weight parameters calculated

one or more training techniques, including those described herein. In at least one embodiment, each processing cluster 1410 may include, without limitation, any number and type of processors. In at least one embodiment, deep learning application processor 1400 may include any number and type of processing clusters 1400. In at least one embodiment, Inter-Chip Links 1420 are bi-directional. In at least one embodiment, Inter-Chip Links 1420 and Inter-Chip Controllers 1430 enable multiple deep learning application processors 1400 to exchange information, including activation information resulting from performing one or more machine learning algorithms embodied in one or more neural networks. In at least one embodiment, deep learning application processor 1400 may include any number (including zero) and type of ICLs 1420 and ICCs 1430.

[0250] In at least one embodiment, HBM2s 1440 provide a total of 32 Gigabytes (GB) of memory. HBM2 1440(i) is associated with both memory controller 1442(i) and HBM PHY 1444(i). In at least one embodiment, any number of HBM2s 1440 may provide any type and total amount of high bandwidth memory and may be associated with any number (including zero) and type of memory controllers 1442 and HBM PHYs 1444. In at least one embodiment, SPI, I2C, GPIO 1460, PCIe Controller and DMA 1470, and/or PCIe 1480 may be replaced with any number and type of blocks that enable any number and type of communication standards in any technically feasible fashion.

[0251] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, deep learning application processor 1400 is used to train a machine learning model, such as a neural network, to predict or infer information provided to deep learning application processor 1400. In at least one embodiment, deep learning application processor 1400 is used to infer or predict information based on a trained machine learning model (such as neural network) that has been trained by another processor or system or by deep learning application processor 1400. In at least one embodiment, processor 1400 may be used to perform one or more neural network use cases described herein.

[0252] FIG. 15 is a block diagram of a neuromorphic processor 1500, according to at least one embodiment. In at least one embodiment, neuromorphic processor 1500 may receive one or more inputs from sources external to neuromorphic processor 1500. In at least one embodiment,

these inputs may be transmitted to one or more neurons 1502 within neuromorphic processor 1500. In at least one embodiment, neurons 1502 and components thereof may be implemented using circuitry or logic, including one or more arithmetic logic units (ALUs). In at least one embodiment, neuromorphic processor 1500 may include, without limitation, thousands or millions of instances of neurons 1502, but any suitable number of neurons 1502 may be used. In at least one embodiment, each instance of neuron 1502 may include a neuron input 1504 and a neuron output 1506. In at least one embodiment, neurons 1502 may generate outputs that may be transmitted to inputs of other instances of neurons 1502. In at least one embodiment, in at least one embodiment, neuron inputs 1504 and neuron outputs 1506 may be interconnected via synapses 1508.

[0253] In at least one embodiment, neurons 1502 and synapses 1508 may be interconnected such that neuromorphic processor 1500 operates to process or analyze information received by neuromorphic processor 1500. In at least one embodiment, neurons 1502 may transmit an output pulse (or "fire" or "spike") when inputs received through neuron input 1504 exceed a threshold. In at least one embodiment, neurons 1502 may sum or integrate signals received at neuron inputs 1504. In at least one embodiment, in at least one embodiment, neurons 1502 may be implemented as leaky integrate-and-fire neurons, wherein if a sum (referred to as a "membrane potential") exceeds a threshold value, neuron 1502 may generate an output (or "fire") using a transfer function such as a sigmoid or threshold function. In at least one embodiment, a leaky integrate-and-fire neuron may sum signals received at neuron inputs 1504 into a membrane potential and may also apply a decay factor (or leak) to reduce a membrane potential. In at least one embodiment, a leaky integrate-and-fire neuron may fire if multiple input signals are received at neuron inputs 1504 rapidly enough to exceed a threshold value (in at least one embodiment, this is before a membrane potential decays too low to fire). In at least one embodiment, neurons 1502 may be implemented using circuits or logic that receive inputs, integrate inputs into a membrane potential, and decay a membrane potential. In at least one embodiment, inputs may be averaged, or any other suitable transfer function may be used. Furthermore, in at least one embodiment, neurons 1502 may include, without limitation, comparator circuits or logic that generate an output spike at neuron output 1506 when result of applying a transfer function to neuron input 1504 exceeds a threshold. In at least one embodiment, once neuron 1502 fires, it may disregard previously received input information by, for example, resetting a membrane

potential to 0 or another suitable default value. In at least one embodiment, once membrane potential is reset to 0, neuron 1502 may resume normal operation after a suitable period of time (or refractory period).

[0254] In at least one embodiment, neurons 1502 may be interconnected through synapses 1508. In at least one embodiment, synapses 1508 may operate to transmit signals from an output of a first neuron 1502 to an input of a second neuron 1502. In at least one embodiment, neurons 1502 may transmit information over more than one instance of synapse 1508. In at least one embodiment, one or more instances of neuron output 1506 may be connected, via an instance of synapse 1508, to an instance of neuron input 1504 in same neuron 1502. In at least one embodiment, an instance of neuron 1502 generating an output to be transmitted over an instance of synapse 1508 may be referred to as a "pre-synaptic neuron" with respect to that instance of synapse 1508. In at least one embodiment, an instance of neuron 1502 receiving an input transmitted over an instance of synapse 1508 may be referred to as a "post-synaptic neuron" with respect to that instance of synapse 1508. Because an instance of neuron 1502 may receive inputs from one or more instances of synapse 1508, and may also transmit outputs over one or more instances of synapse 1508, a single instance of neuron 1502 may therefore be both a "pre-synaptic neuron" and "post-synaptic neuron," with respect to various instances of synapses 1508, in at least one embodiment.

[0255] In at least one embodiment, neurons 1502 may be organized into one or more layers. Each instance of neuron 1502 may have one neuron output 1506 that may fan out through one or more synapses 1508 to one or more neuron inputs 1504. In at least one embodiment, neuron outputs 1506 of neurons 1502 in a first layer 1510 may be connected to neuron inputs 1504 of neurons 1502 in a second layer 1512. In at least one embodiment, layer 1510 may be referred to as a "feed-forward layer." In at least one embodiment, each instance of neuron 1502 in an instance of first layer 1510 may fan out to each instance of neuron 1502 in second layer 1512. In at least one embodiment, first layer 1510 may be referred to as a "fully connected feed-forward layer." In at least one embodiment, each instance of neuron 1502 in an instance of second layer 1512 may fan out to fewer than all instances of neuron 1502 in a third layer 1514. In at least one embodiment, second layer 1512 may be referred to as a "sparsely connected feed-forward layer." In at least one embodiment, neurons 1502 in second layer 1512 may fan out to neurons 1502 in

multiple other layers, including to neurons 1502 in (same) second layer 1512. In at least one embodiment, second layer 1512 may be referred to as a "recurrent layer." In at least one embodiment, neuromorphic processor 1500 may include, without limitation, any suitable combination of recurrent layers and feed-forward layers, including, without limitation, both sparsely connected feed-forward layers and fully connected feed-forward layers.

[0256] In at least one embodiment, neuromorphic processor 1500 may include, without limitation, a reconfigurable interconnect architecture or dedicated hard wired interconnects to connect synapse 1508 to neurons 1502. In at least one embodiment, neuromorphic processor 1500 may include, without limitation, circuitry or logic that allows synapses to be allocated to different neurons 1502 as needed based on neural network topology and neuron fan-in/out. In at least one embodiment, in at least one embodiment, synapses 1508 may be connected to neurons 1502 using an interconnect fabric, such as network-on-chip, or with dedicated connections. In at least one embodiment, synapse interconnections and components thereof may be implemented using circuitry or logic.

[0257] FIG. 16A is a block diagram of a processing system, according to at least one embodiment. In at least one embodiment, system 1600A includes one or more processors 1602 and one or more graphics processors 1608, and may be a single processor desktop system, a multiprocessor workstation system, or a server system having a large number of processors 1602 or processor cores 1607. In at least one embodiment, system 1600A is a processing platform incorporated within a system-on-a-chip (SoC) integrated circuit for use in mobile, handheld, or embedded devices.

[0258] In at least one embodiment, system 1600A can include, or be incorporated within a server-based gaming platform, a game console, including a game and media console, a mobile gaming console, a handheld game console, or an online game console. In at least one embodiment, system 1600A is a mobile phone, smart phone, tablet computing device or mobile Internet device. In at least one embodiment, processing system 1600A can also include, couple with, or be integrated within a wearable device, such as a smart watch wearable device, smart eyewear device, augmented reality device, or virtual reality device. In at least one embodiment, processing system 1600A is a television or set top box device having one or more processors 1602 and a graphical interface generated by one or more graphics processors 1608.

[0259] In at least one embodiment, one or more processors 1602 each include one or more processor cores 1607 to process instructions which, when executed, perform operations for system and user software. In at least one embodiment, each of one or more processor cores 1607 is configured to process a specific instruction set 1609. In at least one embodiment, instruction set 1609 may facilitate Complex Instruction Set Computing (CISC), Reduced Instruction Set Computing (RISC), or computing via a Very Long Instruction Word (VLIW). In at least one embodiment, processor cores 1607 may each process a different instruction set 1609, which may include instructions to facilitate emulation of other instruction sets. In at least one embodiment, processor core 1607 may also include other processing devices, such a Digital Signal Processor (DSP).

[0260] In at least one embodiment, processor 1602 includes cache memory 1604. In at least one embodiment, processor 1602 can have a single internal cache or multiple levels of internal cache. In at least one embodiment, cache memory is shared among various components of processor 1602. In at least one embodiment, processor 1602 also uses an external cache (such as a Level-3 (L3) cache or Last Level Cache (LLC)) (not shown), which may be shared among processor cores 1607 using known cache coherency techniques. In at least one embodiment, register file 1606 is additionally included in processor 1602 which may include different types of registers for storing different types of data (such as integer registers, floating point registers, status registers, and an instruction pointer register). In at least one embodiment, register file 1606 may include general-purpose registers or other registers.

[0261] In at least one embodiment, one or more processor(s) 1602 are coupled with one or more interface bus(es) 1610 to transmit communication signals such as address, data, or control signals between processor 1602 and other components in system 1600A. In at least one embodiment, interface bus 1610, in one embodiment, can be a processor bus, such as a version of a Direct Media Interface (DMI) bus. In at least one embodiment, interface 1610 is not limited to a DMI bus, and may include one or more Peripheral Component Interconnect buses (such as PCI, PCI Express), memory busses, or other types of interface busses. In at least one embodiment processor(s) 1602 include an integrated memory controller 1616 and a platform controller hub 1630. In at least one embodiment, memory controller 1616 facilitates communication between a

memory device and other components of system 1600A, while platform controller hub (PCH) 1630 provides connections to I/O devices via a local I/O bus.

[0262] In at least one embodiment, memory device 1620 can be a dynamic random access memory (DRAM) device, a static random access memory (SRAM) device, flash memory device, phase-change memory device, or some other memory device having suitable performance to serve as process memory. In at least one embodiment memory device 1620 can operate as system memory for system 1600A, to store data 1622 and instructions 1621 for use when one or more processors 1602 executes an application or process. In at least one embodiment, memory controller 1616 also couples with an, in at least one embodiment, external graphics processor 1612, which may communicate with one or more graphics processors 1608 in processors 1602 to perform graphics and media operations. In at least one embodiment, a display device 1611 can connect to processor(s) 1602. In at least one embodiment display device 1611 can include one or more of an internal display device, as in a mobile electronic device or a laptop device or an external display device attached via a display interface (such as DisplayPort, etc.). In at least one embodiment, display device 1611 can include a head mounted display (HMD) such as a stereoscopic display device for use in virtual reality (VR) applications or augmented reality (AR) applications.

[0263] In at least one embodiment, platform controller hub 1630 enables peripherals to connect to memory device 1620 and processor 1602 via a high-speed I/O bus. In at least one embodiment, I/O peripherals include, but are not limited to, an audio controller 1646, a network controller 1634, a firmware interface 1628, a wireless transceiver 1626, touch sensors 1625, a data storage device 1624 (such as hard disk drive, flash memory, etc.). In at least one embodiment, data storage device 1624 can connect via a storage interface (such as SATA) or via a peripheral bus, such as a Peripheral Component Interconnect bus (such as PCI, PCI Express). In at least one embodiment, touch sensors 1625 can include touch screen sensors, pressure sensors, or fingerprint sensors. In at least one embodiment, wireless transceiver 1626 can be a Wi-Fi transceiver, a Bluetooth transceiver, or a mobile network transceiver such as a 3G, 4G, or Long Term Evolution (LTE) transceiver. In at least one embodiment, firmware interface 1628 enables communication with system firmware, and can be, for example, a unified extensible firmware interface (UEFI). In at least one embodiment, network controller 1634 can enable a

network connection to a wired network. In at least one embodiment, a high-performance network controller (not shown) couples with interface bus 1610. In at least one embodiment, audio controller 1646 is a multi-channel high definition audio controller. In at least one embodiment, system 1600A includes a legacy I/O controller 1640 for coupling legacy (such as Personal System 2 (PS/2)) devices to system. In at least one embodiment, platform controller hub 1630 can also connect to one or more Universal Serial Bus (USB) controllers 1642 connect input devices, such as keyboard and mouse 1643 combinations, a camera 1644, or other USB input devices.

[0264] In at least one embodiment, an instance of memory controller 1616 and platform controller hub 1630 may be integrated into a discreet external graphics processor, such as external graphics processor 1612. In at least one embodiment, platform controller hub 1630 and/or memory controller 1616 may be external to one or more processor(s) 1602. In at least one embodiment, in at least one embodiment, system 1600A can include an external memory controller 1616 and platform controller hub 1630, which may be configured as a memory controller hub and peripheral controller hub within a system chipset that is in communication with processor(s) 1602.

[0265] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment portions or all of inference and/or training logic 615 may be incorporated into graphics processor 1600A. In at least one embodiment, in at least one embodiment, training and/or inferencing techniques described herein may use one or more of ALUs embodied in graphics processor 1612. Moreover, in at least one embodiment, inferencing and/or training operations described herein may be done using logic other than logic illustrated in FIGS. 6B or 6C. In at least one embodiment, weight parameters may be stored in on-chip or off-chip memory and/or registers (shown or not shown) that configure ALUs of graphics processor 1600A to perform one or more machine learning algorithms, neural network architectures, use cases, or training techniques described herein.

[0266] FIG. 16B is a block diagram of a processor 1600B having one or more processor cores 1602A-1602N, an integrated memory controller 1614, and an integrated graphics processor 1608,

according to at least one embodiment. In at least one embodiment, processor 1600B can include additional cores up to and including additional core 1602N represented by dashed lined boxes. In at least one embodiment, each of processor cores 1602A-1602N includes one or more internal cache units 1604A-1604N. In at least one embodiment, each processor core also has access to one or more shared cached units 1606.

[0267] In at least one embodiment, internal cache units 1604A-1604N and shared cache units 1606 represent a cache memory hierarchy within processor 1600B. In at least one embodiment, cache memory units 1604A-1604N may include at least one level of instruction and data cache within each processor core and one or more levels of shared mid-level cache, such as a Level 2 (L2), Level 3 (L3), Level 4 (L4), or other levels of cache, where a highest level of cache before external memory is classified as an LLC. In at least one embodiment, cache coherency logic maintains coherency between various cache units 1606 and 1604A-1604N.

[0268] In at least one embodiment, processor 1600B may also include a set of one or more bus controller units 1616 and a system agent core 1610. In at least one embodiment, one or more bus controller units 1616 manage a set of peripheral buses, such as one or more PCI or PCI express busses. In at least one embodiment, system agent core 1610 provides management functionality for various processor components. In at least one embodiment, system agent core 1610 includes one or more integrated memory controllers 1614 to manage access to various external memory devices (not shown).

[0269] In at least one embodiment, one or more of processor cores 1602A-1602N include support for simultaneous multi-threading. In at least one embodiment, system agent core 1610 includes components for coordinating and operating cores 1602A-1602N during multi-threaded processing. In at least one embodiment, system agent core 1610 may additionally include a power control unit (PCU), which includes logic and components to regulate one or more power states of processor cores 1602A-1602N and graphics processor 1608.

[0270] In at least one embodiment, processor 1600B additionally includes graphics processor 1608 to execute graphics processing operations. In at least one embodiment, graphics processor 1608 couples with shared cache units 1606, and system agent core 1610, including one or more integrated memory controllers 1614. In at least one embodiment, system agent core 1610 also includes a display controller 1611 to drive graphics processor output to one or more coupled

displays. In at least one embodiment, display controller 1611 may also be a separate module coupled with graphics processor 1608 via at least one interconnect, or may be integrated within graphics processor 1608.

[0271] In at least one embodiment, a ring based interconnect unit 1612 is used to couple internal components of processor 1600B. In at least one embodiment, an alternative interconnect unit may be used, such as a point-to-point interconnect, a switched interconnect, or other techniques. In at least one embodiment, graphics processor 1608 couples with ring interconnect 1612 via an I/O link 1613.

[0272] In at least one embodiment, I/O link 1613 represents at least one of multiple varieties of I/O interconnects, including an on package I/O interconnect which facilitates communication between various processor components and a high-performance embedded memory module 1618, such as an eDRAM module. In at least one embodiment, each of processor cores 1602A-1602N and graphics processor 1608 use embedded memory modules 1618 as a shared Last Level Cache.

[0273] In at least one embodiment, processor cores 1602A-1602N are homogenous cores executing a common instruction set architecture. In at least one embodiment, processor cores 1602A-1602N are heterogeneous in terms of instruction set architecture (ISA), where one or more of processor cores 1602A-1602N execute a common instruction set, while one or more other cores of processor cores 1602A-16-02N executes a subset of a common instruction set or a different instruction set. In at least one embodiment, processor cores 1602A-1602N are heterogeneous in terms of microarchitecture, where one or more cores having a relatively higher power consumption couple with one or more power cores having a lower power consumption. In at least one embodiment, processor 1600B can be implemented on one or more chips or as an SoC integrated circuit.

[0274] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment portions or all of inference and/or training logic 615 may be incorporated into processor 1600B. In at least one embodiment, in at least one embodiment, training and/or inferencing techniques described herein may use one or more of ALUs embodied in graphics processor 1612, graphics core(s) 1602A-1602N, or other components in FIG. 16. Moreover, in

at least one embodiment, inferencing and/or training operations described herein may be done using logic other than logic illustrated in FIGS. 6B or 6C. In at least one embodiment, weight parameters may be stored in on-chip or off-chip memory and/or registers (shown or not shown) that configure ALUs of graphics processor 1600B to perform one or more machine learning algorithms, neural network architectures, use cases, or training techniques described herein.

[0275] FIG. 16C is a block diagram of hardware logic of a graphics processor core 1600C, according to at least one embodiment described herein. In at least one embodiment, graphics processor core 1600C is included within a graphics core array. In at least one embodiment, graphics processor core 1600C, sometimes referred to as a core slice, can be one or multiple graphics cores within a modular graphics processor. In at least one embodiment, graphics processor core 1600C is exemplary of one graphics core slice, and a graphics processor as described herein may include multiple graphics core slices based on target power and performance envelopes. In at least one embodiment, each graphics core 1600C can include a fixed function block 1630 coupled with multiple sub-cores 1601A-1601F, also referred to as sub-slices, that include modular blocks of general-purpose and fixed function logic.

[0276] In at least one embodiment, fixed function block 1630 includes a geometry/fixed function pipeline 1636 that can be shared by all sub-cores in graphics processor 1600C, for example, in lower performance and/or lower power graphics processor implementations. In at least one embodiment, geometry/fixed function pipeline 1636 includes a 3D fixed function pipeline, a video front-end unit, a thread spawner and thread dispatcher, and a unified return buffer manager, which manages unified return buffers.

[0277] In at least one embodiment fixed, function block 1630 also includes a graphics SoC interface 1637, a graphics microcontroller 1638, and a media pipeline 1639. In at least one embodiment fixed, graphics SoC interface 1637 provides an interface between graphics core 1600C and other processor cores within a system on a chip integrated circuit. In at least one embodiment, graphics microcontroller 1638 is a programmable sub-processor that is configurable to manage various functions of graphics processor 1600C, including thread dispatch, scheduling, and pre-emption. In at least one embodiment, media pipeline 1639 includes logic to facilitate decoding, encoding, pre-processing, and/or post-processing of multimedia data,

including image and video data. In at least one embodiment, media pipeline 1639 implements media operations via requests to compute or sampling logic within sub-cores 1601-1601F.

[0278] In at least one embodiment, SoC interface 1637 enables graphics core 1600C to communicate with general-purpose application processor cores (such as CPUs) and/or other components within an SoC, including memory hierarchy elements such as a shared last level cache memory, system RAM, and/or embedded on-chip or on-package DRAM. In at least one embodiment, SoC interface 1637 can also enable communication with fixed function devices within an SoC, such as camera imaging pipelines, and enables use of and/or implements global memory atomics that may be shared between graphics core 1600C and CPUs within an SoC. In at least one embodiment, SoC interface 1637 can also implement power management controls for graphics core 1600C and enable an interface between a clock domain of graphic core 1600C and other clock domains within an SoC. In at least one embodiment, SoC interface 1637 enables receipt of command buffers from a command streamer and global thread dispatcher that are configured to provide commands and instructions to each of one or more graphics cores within a graphics processor. In at least one embodiment, commands and instructions can be dispatched to media pipeline 1639, when media operations are to be performed, or a geometry and fixed function pipeline (such as geometry and fixed function pipeline 1636, geometry and fixed function pipeline 1614) when graphics processing operations are to be performed.

[0279] In at least one embodiment, graphics microcontroller 1638 can be configured to perform various scheduling and management tasks for graphics core 1600C. In at least one embodiment, graphics microcontroller 1638 can perform graphics and/or compute workload scheduling on various graphics parallel engines within execution unit (EU) arrays 1602A-1602F, 1604A-1604F within sub-cores 1601A-1601F. In at least one embodiment, host software executing on a CPU core of an SoC including graphics core 1600C can submit workloads one of multiple graphic processor doorbells, which invokes a scheduling operation on an appropriate graphics engine. In at least one embodiment, scheduling operations include determining which workload to run next, submitting a workload to a command streamer, pre-empting existing workloads running on an engine, monitoring progress of a workload, and notifying host software when a workload is complete. In at least one embodiment, graphics microcontroller 1638 can also facilitate low-power or idle states for graphics core 1600C, providing graphics core 1600C

with an ability to save and restore registers within graphics core 1600C across low-power state transitions independently from an operating system and/or graphics driver software on a system.

[0280] In at least one embodiment, graphics core 1600C may have greater than or fewer than illustrated sub-cores 1601A-1601F, up to N modular sub-cores. For each set of N sub-cores, in at least one embodiment, graphics core 1600C can also include shared function logic 1610, shared and/or cache memory 1612, a geometry/fixed function pipeline 1614, as well as additional fixed function logic 1616 to accelerate various graphics and compute processing operations. In at least one embodiment, shared function logic 1610 can include logic units (such as sampler, math, and/or inter-thread communication logic) that can be shared by each N sub-cores within graphics core 1600C. In at least one embodiment fixed, shared and/or cache memory 1612 can be a last-level cache for N sub-cores 1601A-1601F within graphics core 1600C and can also serve as shared memory that is accessible by multiple sub-cores. In at least one embodiment, geometry/fixed function pipeline 1614 can be included instead of geometry/fixed function pipeline 1636 within fixed function block 1630 and can include same or similar logic units.

[0281] In at least one embodiment, graphics core 1600C includes additional fixed function logic 1616 that can include various fixed function acceleration logic for use by graphics core 1600C. In at least one embodiment, additional fixed function logic 1616 includes an additional geometry pipeline for use in position only shading. In position-only shading, at least two geometry pipelines exist, whereas in a full geometry pipeline within geometry/fixed function pipeline 1616, 1636, and a cull pipeline, which is an additional geometry pipeline which may be included within additional fixed function logic 1616. In at least one embodiment, cull pipeline is a trimmed down version of a full geometry pipeline. In at least one embodiment, a full pipeline and a cull pipeline can execute different instances of an application, each instance having a separate context. In at least one embodiment, position only shading can hide long cull runs of discarded triangles, enabling shading to be completed earlier in some instances. In at least one embodiment, in at least one embodiment, cull pipeline logic within additional fixed function logic 1616 can execute position shaders in parallel with a main application and generates critical results faster than a full pipeline, as cull pipeline fetches and shades position attribute of vertices, without performing rasterization and rendering of pixels to a frame buffer. In at least one

embodiment, cull pipeline can use generated critical results to compute visibility information for all triangles without regard to whether those triangles are culled. In at least one embodiment, full pipeline (which in this instance may be referred to as a replay pipeline) can consume visibility information to skip culled triangles to shade only visible triangles that are finally passed to a rasterization phase.

[0282] In at least one embodiment, additional fixed function logic 1616 can also include machine-learning acceleration logic, such as fixed function matrix multiplication logic, for implementations including optimizations for machine learning training or inferencing.

[0283] In at least one embodiment, within each graphics sub-core 1601A-1601F includes a set of execution resources that may be used to perform graphics, media, and compute operations in response to requests by graphics pipeline, media pipeline, or shader programs. In at least one embodiment, graphics sub-cores 1601A-1601F include multiple EU arrays 1602A-1602F, 1604A-1604F, thread dispatch and inter-thread communication (TD/IC) logic 1603A-1603F, a 3D (such as texture) sampler 1605A-1605F, a media sampler 1606A-1606F, a shader processor 1607A-1607F, and shared local memory (SLM) 1608A-1608F. EU arrays 1602A-1602F, 1604A-1604F each include multiple execution units, which are general-purpose graphics processing units capable of performing floating-point and integer/fixed-point logic operations in service of a graphics, media, or compute operation, including graphics, media, or compute shader programs. In at least one embodiment, TD/IC logic 1603A-1603F performs local thread dispatch and thread control operations for execution units within a sub-core and facilitate communication between threads executing on execution units of a sub-core. In at least one embodiment, 3D sampler 1605A-1605F can read texture or other 3D graphics related data into memory. In at least one embodiment, 3D sampler can read texture data differently based on a configured sample state and texture format associated with a given texture. In at least one embodiment, media sampler 1606A-1606F can perform similar read operations based on a type and format associated with media data. In at least one embodiment, each graphics sub-core 1601A-1601F can alternately include a unified 3D and media sampler. In at least one embodiment, threads executing on execution units within each of sub-cores 1601A-1601F can make use of shared local memory 1608A-1608F within each sub-core, to enable threads executing within a thread group to execute using a common pool of on-chip memory.

[0284] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, portions or all of inference and/or training logic 615 may be incorporated into graphics processor 1610. In at least one embodiment, in at least one embodiment, training and/or inferencing techniques described herein may use one or more of ALUs embodied in graphics processor 1612, graphics microcontroller 1638, geometry & fixed function pipeline 1614 and 1636, or other logic in FIG. 16B. Moreover, in at least one embodiment, inferencing and/or training operations described herein may be done using logic other than logic illustrated in FIGS. 6B or 6C. In at least one embodiment, weight parameters may be stored in on-chip or off-chip memory and/or registers (shown or not shown) that configure ALUs of graphics processor 1600C to perform one or more machine learning algorithms, neural network architectures, use cases, or training techniques described herein.

[0285] FIGS. 16D-16E illustrate thread execution logic 1600D including an array of processing elements of a graphics processor core according to at least one embodiment. FIG. 16D illustrates at least one embodiment, in which thread execution logic 1600D is used. FIG. 16E illustrates exemplary internal details of an execution unit, according to at least one embodiment.

[0286] As illustrated in FIG. 16D, in at least one embodiment, thread execution logic 1600D includes a shader processor 1602, a thread dispatcher 1604, instruction cache 1606, a scalable execution unit array including a plurality of execution units 1608A-1608N, sampler(s) 1610, a data cache 1612, and a data port 1614. In at least one embodiment a scalable execution unit array can dynamically scale by enabling or disabling one or more execution units (such as any of execution unit 1608A, 1608B, 1608C, 1608D, through 1608N-1 and 1608N) based on computational requirements of a workload, for example. In at least one embodiment, scalable execution units are interconnected via an interconnect fabric that links to each of execution unit. In at least one embodiment, thread execution logic 1600D includes one or more connections to memory, such as system memory or cache memory, through one or more of instruction cache 1606, data port 1614, sampler 1610, and execution units 1608A-1608N. In at least one embodiment, each execution unit (such as 1608A) is a stand-alone programmable general-

purpose computational unit that is capable of executing multiple simultaneous hardware threads while processing multiple data elements in parallel for each thread. In at least one embodiment, array of execution units 1608A-1608N is scalable to include any number individual execution units.

[0287] In at least one embodiment, execution units 1608A-1608N are primarily used to execute shader programs. In at least one embodiment, shader processor 1602 can process various shader programs and dispatch execution threads associated with shader programs via a thread dispatcher 1604. In at least one embodiment, thread dispatcher 1604 includes logic to arbitrate thread initiation requests from graphics and media pipelines and instantiate requested threads on one or more execution units in execution units 1608A-1608N. In at least one embodiment, in at least one embodiment, a geometry pipeline can dispatch vertex, tessellation, or geometry shaders to thread execution logic for processing. In at least one embodiment, thread dispatcher 1604 can also process runtime thread spawning requests from executing shader programs.

[0288] In at least one embodiment, execution units 1608A-1608N support an instruction set that includes native support for many standard 3D graphics shader instructions, such that shader programs from graphics libraries (such as Direct 3D and OpenGL) are executed with a minimal translation. In at least one embodiment, execution units support vertex and geometry processing (such as vertex programs, geometry programs, vertex shaders), pixel processing (such as pixel shaders, fragment shaders) and general-purpose processing (such as compute and media shaders). In at least one embodiment, each of execution units 1608A-1608N, which include one or more arithmetic logic units (ALUs), is capable of multi-issue single instruction multiple data (SIMD) execution and multi-threaded operation enables an efficient execution environment despite higher latency memory accesses. In at least one embodiment, each hardware thread within each execution unit has a dedicated high-bandwidth register file and associated independent thread-state. In at least one embodiment, execution is multi-issue per clock to pipelines capable of integer, single and double precision floating point operations, SIMD branch capability, logical operations, transcendental operations, and other miscellaneous operations. In at least one embodiment, while waiting for data from memory or one of shared functions, dependency logic within execution units 1608A-1608N causes a waiting thread to sleep until requested data has

been returned. In at least one embodiment, while a waiting thread is sleeping, hardware resources may be devoted to processing other threads. In at least one embodiment, in at least one embodiment, during a delay associated with a vertex shader operation, an execution unit can perform operations for a pixel shader, fragment shader, or another type of shader program, including a different vertex shader.

[0289] In at least one embodiment, each execution unit in execution units 1608A-1608N operates on arrays of data elements. In at least one embodiment, a number of data elements is "execution size," or number of channels for an instruction. In at least one embodiment, an execution channel is a logical unit of execution for data element access, masking, and flow control within instructions. In at least one embodiment, a number of channels may be independent of a number of physical Arithmetic Logic Units (ALUs) or Floating Point Units (FPUs) for a particular graphics processor. In at least one embodiment, execution units 1608A-1608N support integer and floating-point data types.

[0290] In at least one embodiment, an execution unit instruction set includes SIMD instructions. In at least one embodiment, various data elements can be stored as a packed data type in a register and an execution unit will process various elements based on data size of elements. In at least one embodiment, in at least one embodiment, when operating on a 256-bit wide vector, 256 bits of a vector are stored in a register and an execution unit operates on a vector as four separate 64-bit packed data elements (Quad-Word (QW) size data elements), eight separate 32-bit packed data elements (Double Word (DW) size data elements), sixteen separate 16-bit packed data elements (Word (W) size data elements), or thirty-two separate 8-bit data elements (byte (B) size data elements). However, in at least one embodiment, different vector widths and register sizes are possible.

[0291] In at least one embodiment, one or more execution units can be combined into a fused execution unit 1609A-1609N having thread control logic (1607A-1607N) that is common to fused EUs. In at least one embodiment, multiple EUs can be fused into an EU group. In at least one embodiment, each EU in fused EU group can be configured to execute a separate SIMD hardware thread. Number of EUs in a fused EU group can vary according to various embodiments. In at least one embodiment, various SIMD widths can be performed per-EU, including but not limited to SIMD8, SIMD16, and SIMD32. In at least one embodiment, each

fused graphics execution unit 1609A-1609N includes at least two execution units. In at least one embodiment, in at least one embodiment, fused execution unit 1609A includes a first EU 1608A, second EU 1608B, and thread control logic 1607A that is common to first EU 1608A and second EU 1608B. In at least one embodiment, thread control logic 1607A controls threads executed on fused graphics execution unit 1609A, allowing each EU within fused execution units 1609A-1609N to execute using a common instruction pointer register.

[0292] In at least one embodiment, one or more internal instruction caches (such as 1606) are included in thread execution logic 1600D to cache thread instructions for execution units. In at least one embodiment, one or more data caches (such as 1612) are included to cache thread data during thread execution. In at least one embodiment, a sampler 1610 is included to provide texture sampling for 3D operations and media sampling for media operations. In at least one embodiment, sampler 1610 includes specialized texture or media sampling functionality to process texture or media data during a sampling process before providing sampled data to an execution unit.

[0293] During execution, in at least one embodiment, graphics and media pipelines send thread initiation requests to thread execution logic 1600D via thread spawning and dispatch logic. In at least one embodiment, once a group of geometric objects has been processed and rasterized into pixel data, pixel processor logic (such as pixel shader logic, fragment shader logic, etc.) within shader processor 1602 is invoked to further compute output information and cause results to be written to output surfaces (such as color buffers, depth buffers, stencil buffers, etc.). In at least one embodiment, a pixel shader or fragment shader calculates values of various vertex attributes that are to be interpolated across a rasterized object. In at least one embodiment, pixel processor logic within shader processor 1602 then executes an application programming interface (API)-supplied pixel or fragment shader program. In at least one embodiment, to execute a shader program, shader processor 1602 dispatches threads to an execution unit (such as 1608A) via thread dispatcher 1604. In at least one embodiment, shader processor 1602 uses texture sampling logic in sampler 1610 to access texture data in texture maps stored in memory. In at least one embodiment, arithmetic operations on texture data and input geometry data compute pixel color data for each geometric fragment, or discards one or more pixels from further processing.

[0294] In at least one embodiment, data port 1614 provides a memory access mechanism for thread execution logic 1600D to output processed data to memory for further processing on a graphics processor output pipeline. In at least one embodiment, data port 1614 includes or couples to one or more cache memories (such as data cache 1612) to cache data for memory access via a data port.

[0295] As illustrated in FIG. 16E, in at least one embodiment, a graphics execution unit 1608 can include an instruction fetch unit 1637, a general register file array (GRF) 1624, an architectural register file array (ARF) 1626, a thread arbiter 1622, a send unit 1630, a branch unit 1632, a set of SIMD floating point units (FPUs) 1634, and, in at least one embodiment, a set of dedicated integer SIMD ALUs 1635. In at least one embodiment, GRF 1624 and ARF 1626 includes a set of general register files and architecture register files associated with each simultaneous hardware thread that may be active in graphics execution unit 1608. In at least one embodiment, per thread architectural state is maintained in ARF 1626, while data used during thread execution is stored in GRF 1624. In at least one embodiment, execution state of each thread, including instruction pointers for each thread, can be held in thread-specific registers in ARF 1626.

[0296] In at least one embodiment, graphics execution unit 1608 has an architecture that is a combination of Simultaneous Multi-Threading (SMT) and fine-grained Interleaved Multi-Threading (IMT). In at least one embodiment, architecture has a modular configuration that can be fine-tuned at design time based on a target number of simultaneous threads and number of registers per execution unit, where execution unit resources are divided across logic used to execute multiple simultaneous threads.

[0297] In at least one embodiment, graphics execution unit 1608 can co-issue multiple instructions, which may each be different instructions. In at least one embodiment, thread arbiter 1622 of graphics execution unit thread 1608 can dispatch instructions to one of send unit 1630, branch unit 1642, or SIMD FPU(s) 1634 for execution. In at least one embodiment, each execution thread can access 128 general-purpose registers within GRF 1624, where each register can store 32 bytes, accessible as a SIMD 8-element vector of 32-bit data elements. In at least one embodiment, each execution unit thread has access to 4 Kbytes within GRF 1624, although embodiments are not so limited, and greater or fewer register resources may be provided in other

embodiments. In at least one embodiment, up to seven threads can execute simultaneously, although a number of threads per execution unit can also vary according to embodiments. In at least one embodiment, in which seven threads may access 4 Kbytes, GRF 1624 can store a total of 28 Kbytes. In at least one embodiment, flexible addressing modes can permit registers to be addressed together to build effectively wider registers or to represent strided rectangular block data structures.

[0298] In at least one embodiment, memory operations, sampler operations, and other longer-latency system communications are dispatched via "send" instructions that are executed by message passing send unit 1630. In at least one embodiment, branch instructions are dispatched to a dedicated branch unit 1632 to facilitate SIMD divergence and eventual convergence.

[0299] In at least one embodiment graphics execution unit 1608 includes one or more SIMD floating point units (FPU(s)) 1634 to perform floating-point operations. In at least one embodiment, FPU(s) 1634 also support integer computation. In at least one embodiment FPU(s) 1634 can SIMD execute up to M number of 32-bit floating-point (or integer) operations, or SIMD execute up to 2M 16-bit integer or 16-bit floating-point operations. In at least one embodiment, at least one of FPU(s) provides extended math capability to support high-throughput transcendental math functions and double precision 64-bit floating-point. In at least one embodiment, a set of 8-bit integer SIMD ALUs 1635 are also present, and may be specifically optimized to perform operations associated with machine learning computations.

[0300] In at least one embodiment, arrays of multiple instances of graphics execution unit 1608 can be instantiated in a graphics sub-core grouping (such as a sub-slice). In at least one embodiment, execution unit 1608 can execute instructions across a plurality of execution channels. In at least one embodiment, each thread executed on graphics execution unit 1608 is executed on a different channel.

[0301] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, portions or all of inference and/or training logic 615 may be incorporated into execution logic 1600D. Moreover, in at least one embodiment, inferencing and/or training operations described herein may be done using logic other than logic illustrated in FIGS. 6B or

6C. In at least one embodiment, weight parameters may be stored in on-chip or off-chip memory and/or registers (shown or not shown) that configure ALUs of execution logic 1600D to perform one or more machine learning algorithms, neural network architectures, use cases, or training techniques described herein.

[0302] FIG. 17A illustrates a parallel processing unit ("PPU") 1700A, according to at least one embodiment. In at least one embodiment, PPU 1700A is configured with machine-readable code that, if executed by PPU 1700A, causes PPU 1700A to perform some or all of processes and techniques described throughout this disclosure. In at least one embodiment, PPU 1700A is a multi-threaded processor that is implemented on one or more integrated circuit devices and that utilizes multithreading as a latency-hiding technique designed to process computer-readable instructions (also referred to as machine-readable instructions or simply instructions) on multiple threads in parallel. In at least one embodiment, a thread refers to a thread of execution and is an instantiation of a set of instructions configured to be executed by PPU 1700A. In at least one embodiment, PPU 1700A is a graphics processing unit ("GPU") configured to implement a graphics rendering pipeline for processing three-dimensional ("3D") graphics data in order to generate two-dimensional ("2D") image data for display on a display device such as a liquid crystal display ("LCD") device. In at least one embodiment, PPU 1700A is utilized to perform computations such as linear algebra operations and machine-learning operations. FIG. 17A illustrates an example parallel processor for illustrative purposes only and should be construed as a non-limiting example of processor architectures contemplated within scope of this disclosure and that any suitable processor may be employed to supplement and/or substitute for same.

[0303] In at least one embodiment, one or more PPUs 1700A are configured to accelerate High Performance Computing ("HPC"), datacenter, and machine learning applications. In at least one embodiment, PPU 1700A is configured to accelerate deep learning systems and applications including following non-limiting examples: autonomous vehicle platforms, deep learning, high-accuracy speech, image, text recognition systems, intelligent video analytics, molecular simulations, drug discovery, disease diagnosis, weather forecasting, big data analytics, astronomy, molecular dynamics simulation, financial modeling, robotics, factory automation, real-time language translation, online search optimizations, and personalized user recommendations, and more.

[0304] In at least one embodiment, PPU 1700A includes, without limitation, an Input/Output ("I/O") unit 1706, a front-end unit 1710, a scheduler unit 1712, a work distribution unit 1714, a hub 1716, a crossbar ("Xbar") 1720, one or more general processing clusters ("GPCs") 1718, and one or more partition units ("memory partition units") 1722. In at least one embodiment, PPU 1700A is connected to a host processor or other PPUs 1700A via one or more high-speed GPU interconnects ("GPU interconnects") 1708. In at least one embodiment, PPU 1700A is connected to a host processor or other peripheral devices via an interconnect 1702. In at least one embodiment, PPU 1700A is connected to a local memory comprising one or more memory devices ("memory") 1704. In at least one embodiment, memory devices 1704 include, without limitation, one or more dynamic random access memory ("DRAM") devices. In at least one embodiment, one or more DRAM devices are configured and/or configurable as high-bandwidth memory ("HBM") subsystems, with multiple DRAM dies stacked within each device.

[0305] In at least one embodiment, high-speed GPU interconnect 1708 may refer to a wire-based multi-lane communications link that is used by systems to scale and include one or more PPUs 1700A combined with one or more central processing units ("CPUs"), supports cache coherence between PPUs 1700A and CPUs, and CPU mastering. In at least one embodiment, data and/or commands are transmitted by high-speed GPU interconnect 1708 through hub 1716 to/from other units of PPU 1700A such as one or more copy engines, video encoders, video decoders, power management units, and other components which may not be explicitly illustrated in FIG. 17A.

[0306] In at least one embodiment, I/O unit 1706 is configured to transmit and receive communications (such as commands, data) from a host processor (not illustrated in FIG. 17A) over system bus 1702. In at least one embodiment, I/O unit 1706 communicates with host processor directly via system bus 1702 or through one or more intermediate devices such as a memory bridge. In at least one embodiment, I/O unit 1706 may communicate with one or more other processors, such as one or more of PPUs 1700A via system bus 1702. In at least one embodiment, I/O unit 1706 implements a Peripheral Component Interconnect Express ("PCIe") interface for communications over a PCIe bus. In at least one embodiment, I/O unit 1706 implements interfaces for communicating with external devices.

[0307] In at least one embodiment, I/O unit 1706 decodes packets received via system bus 1702. In at least one embodiment, at least some packets represent commands configured to cause PPU 1700A to perform various operations. In at least one embodiment, I/O unit 1706 transmits decoded commands to various other units of PPU 1700A as specified by commands. In at least one embodiment, commands are transmitted to front-end unit 1710 and/or transmitted to hub 1716 or other units of PPU 1700A such as one or more copy engines, a video encoder, a video decoder, a power management unit, etc. (not explicitly illustrated in FIG. 17A). In at least one embodiment, I/O unit 1706 is configured to route communications between and among various logical units of PPU 1700A.

[0308] In at least one embodiment, a program executed by host processor encodes a command stream in a buffer that provides workloads to PPU 1700A for processing. In at least one embodiment, a workload comprises instructions and data to be processed by those instructions. In at least one embodiment, buffer is a region in a memory that is accessible (such as read/write) by both host processor and PPU 1700A — a host interface unit may be configured to access buffer in a system memory connected to system bus 1702 via memory requests transmitted over system bus 1702 by I/O unit 1706. In at least one embodiment, host processor writes command stream to buffer and then transmits a pointer to start of command stream to PPU 1700A such that front-end unit 1710 receives pointers to one or more command streams and manages one or more command streams, reading commands from command streams and forwarding commands to various units of PPU 1700A.

[0309] In at least one embodiment, front-end unit 1710 is coupled to scheduler unit 1712 that configures various GPCs 1718 to process tasks defined by one or more command streams. In at least one embodiment, scheduler unit 1712 is configured to track state information related to various tasks managed by scheduler unit 1712 where state information may indicate which of GPCs 1718 a task is assigned to, whether task is active or inactive, a priority level associated with task, and so forth. In at least one embodiment, scheduler unit 1712 manages execution of a plurality of tasks on one or more of GPCs 1718.

[0310] In at least one embodiment, scheduler unit 1712 is coupled to work distribution unit 1714 that is configured to dispatch tasks for execution on GPCs 1718. In at least one embodiment, work distribution unit 1714 tracks a number of scheduled tasks received from

scheduler unit 1712 and work distribution unit 1714 manages a pending task pool and an active task pool for each of GPCs 1718. In at least one embodiment, pending task pool comprises a number of slots (such as 32 slots) that contain tasks assigned to be processed by a particular GPC 1718; active task pool may comprise a number of slots (such as 4 slots) for tasks that are actively being processed by GPCs 1718 such that as one of GPCs 1718 completes execution of a task, that task is evicted from active task pool for GPC 1718 and one of other tasks from pending task pool is selected and scheduled for execution on GPC 1718. In at least one embodiment, if an active task is idle on GPC 1718, such as while waiting for a data dependency to be resolved, then active task is evicted from GPC 1718 and returned to pending task pool while another task in pending task pool is selected and scheduled for execution on GPC 1718.

[0311] In at least one embodiment, work distribution unit 1714 communicates with one or more GPCs 1718 via XBar 1720. In at least one embodiment, XBar 1720 is an interconnect network that couples many of units of PPU 1700A to other units of PPU 1700A and can be configured to couple work distribution unit 1714 to a particular GPC 1718. In at least one embodiment, one or more other units of PPU 1700A may also be connected to XBar 1720 via hub 1716.

[0312] In at least one embodiment, tasks are managed by scheduler unit 1712 and dispatched to one of GPCs 1718 by work distribution unit 1714. GPC 1718 is configured to process task and generate results. In at least one embodiment, results may be consumed by other tasks within GPC 1718, routed to a different GPC 1718 via XBar 1720, or stored in memory 1704. In at least one embodiment, results can be written to memory 1704 via partition units 1722, which implement a memory interface for reading and writing data to/from memory 1704. In at least one embodiment, results can be transmitted to another PPU 1704 or CPU via high-speed GPU interconnect 1708. In at least one embodiment, PPU 1700A includes, without limitation, a number U of partition units 1722 that is equal to number of separate and distinct memory devices 1704 coupled to PPU 1700A. In at least one embodiment, partition unit 1722 will be described in more detail below in conjunction with FIG. 17C.

[0313] In at least one embodiment, a host processor executes a driver kernel that implements an application programming interface ("API") that enables one or more applications executing on host processor to schedule operations for execution on PPU 1700A. In at least one

embodiment, multiple compute applications are simultaneously executed by PPU 1700A and PPU 1700A provides isolation, quality of service ("QoS"), and independent address spaces for multiple compute applications. In at least one embodiment, an application generates instructions (such as in form of API calls) that cause driver kernel to generate one or more tasks for execution by PPU 1700A and driver kernel outputs tasks to one or more streams being processed by PPU 1700A. In at least one embodiment, each task comprises one or more groups of related threads, which may be referred to as a warp. In at least one embodiment, a warp comprises a plurality of related threads (such as 32 threads) that can be executed in parallel. In at least one embodiment, cooperating threads can refer to a plurality of threads including instructions to perform task and that exchange data through shared memory. In at least one embodiment, threads and cooperating threads are described in more detail, in accordance with at least one embodiment, in conjunction with FIG. 17C.

[0314] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, deep learning application processor is used to train a machine learning model, such as a neural network, to predict or infer information provided to PPU 1700A. In at least one embodiment, PPU 1700A is used to infer or predict information based on a trained machine learning model (such as neural network) that has been trained by another processor or system or by PPU 1700A. In at least one embodiment, PPU 1700A may be used to perform one or more neural network use cases described herein.

[0315] FIG. 17B illustrates a general processing cluster ("GPC") 1700B, according to at least one embodiment. In at least one embodiment, GPC 1700B is GPC 1718 of FIG. 17A. In at least one embodiment, each GPC 1700B includes, without limitation, a number of hardware units for processing tasks and each GPC 1700B includes, without limitation, a pipeline manager 1702, a pre-raster operations unit ("PROP") 1704, a raster engine 1708, a work distribution crossbar ("WDX") 1716, a memory management unit ("MMU") 1718, one or more Data Processing Clusters ("DPCs") 1706, and any suitable combination of parts.

[0316] In at least one embodiment, operation of GPC 1700B is controlled by pipeline manager 1702. In at least one embodiment, pipeline manager 1702 manages configuration of one or more

DPCs 1706 for processing tasks allocated to GPC 1700B. In at least one embodiment, pipeline manager 1702 configures at least one of one or more DPCs 1706 to implement at least a portion of a graphics rendering pipeline. In at least one embodiment, DPC 1706 is configured to execute a vertex shader program on a programmable streaming multi-processor ("SM") 1714. In at least one embodiment, pipeline manager 1702 is configured to route packets received from a work distribution unit to appropriate logical units within GPC 1700B, in at least one embodiment, and some packets may be routed to fixed function hardware units in PROP 1704 and/or raster engine 1708 while other packets may be routed to DPCs 1706 for processing by a primitive engine 1712 or SM 1714. In at least one embodiment, pipeline manager 1702 configures at least one of DPCs 1706 to implement a neural network model and/or a computing pipeline.

[0317] In at least one embodiment, PROP unit 1704 is configured, in at least one embodiment, to route data generated by raster engine 1708 and DPCs 1706 to a Raster Operations ("ROP") unit in partition unit 1722, described in more detail above in conjunction with FIG. 17A. In at least one embodiment, PROP unit 1704 is configured to perform optimizations for color blending, organize pixel data, perform address translations, and more. In at least one embodiment, raster engine 1708 includes, without limitation, a number of fixed function hardware units configured to perform various raster operations, in at least one embodiment, and raster engine 1708 includes, without limitation, a setup engine, a coarse raster engine, a culling engine, a clipping engine, a fine raster engine, a tile coalescing engine, and any suitable combination thereof. In at least one embodiment, setup engine receives transformed vertices and generates plane equations associated with geometric primitive defined by vertices; plane equations are transmitted to coarse raster engine to generate coverage information (such as an x, y coverage mask for a tile) for primitive; output of coarse raster engine is transmitted to culling engine where fragments associated with primitive that fail a z-test are culled, and transmitted to a clipping engine where fragments lying outside a viewing frustum are clipped. In at least one embodiment, fragments that survive clipping and culling are passed to fine raster engine to generate attributes for pixel fragments based on plane equations generated by setup engine. In at least one embodiment, output of raster engine 1708 comprises fragments to be processed by any suitable entity such as by a fragment shader implemented within DPC 1706.

[0318] In at least one embodiment, each DPC 1706 included in GPC 1700B comprise, without limitation, an M-Pipe Controller ("MPC") 1710; primitive engine 1712; one or more SMs 1714; and any suitable combination thereof. In at least one embodiment, MPC 1710 controls operation of DPC 1706, routing packets received from pipeline manager 1702 to appropriate units in DPC 1706. In at least one embodiment, packets associated with a vertex are routed to primitive engine 1712, which is configured to fetch vertex attributes associated with vertex from memory; in contrast, packets associated with a shader program may be transmitted to SM 1714.

[0319] In at least one embodiment, SM 1714 comprises, without limitation, a programmable streaming processor that is configured to process tasks represented by a number of threads. In at least one embodiment, SM 1714 is multi-threaded and configured to execute a plurality of threads (such as 32 threads) from a particular group of threads concurrently and implements a Single-Instruction, Multiple-Data ("SIMD") architecture where each thread in a group of threads (such as a warp) is configured to process a different set of data based on same set of instructions. In at least one embodiment, all threads in group of threads execute same instructions. In at least one embodiment, SM 1714 implements a Single-Instruction, Multiple Thread ("SIMT") architecture wherein each thread in a group of threads is configured to process a different set of data based on same set of instructions, but where individual threads in group of threads are allowed to diverge during execution. In at least one embodiment, a program counter, call stack, and execution state is maintained for each warp, enabling concurrency between warps and serial execution within warps when threads within warp diverge. In another embodiment, a program counter, call stack, and execution state is maintained for each individual thread, enabling equal concurrency between all threads, within and between warps. In at least one embodiment, execution state is maintained for each individual thread and threads executing same instructions may be converged and executed in parallel for better efficiency. At least one embodiment of SM 1714 are described in more detail below.

[0320] In at least one embodiment, MMU 1718 provides an interface between GPC 1700B and memory partition unit (such as partition unit 1722 of FIG. 17A) and MMU 1718 provides translation of virtual addresses into physical addresses, memory protection, and arbitration of memory requests. In at least one embodiment, MMU 1718 provides one or more translation

lookaside buffers ("TLBs") for performing translation of virtual addresses into physical addresses in memory.

[0321] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, deep learning application processor is used to train a machine learning model, such as a neural network, to predict or infer information provided to GPC 1700B. In at least one embodiment, GPC 1700B is used to infer or predict information based on a trained machine learning model (such as neural network) that has been trained by another processor or system or by GPC 1700B. In at least one embodiment, GPC 1700B may be used to perform one or more neural network use cases described herein.

[0322] FIG. 17C illustrates a memory partition unit 1700C of a parallel processing unit ("PPU"), in accordance with at least one embodiment. In at least one embodiment, memory partition unit 1700C includes, without limitation, a Raster Operations ("ROP") unit 1702; a level two ("L2") cache 1704; a memory interface 1706; and any suitable combination thereof. In at least one embodiment, memory interface 1706 is coupled to memory. In at least one embodiment, memory interface 1706 may implement 32, 64, 128, 1024-bit data buses, or similar implementations, for high-speed data transfer. In at least one embodiment, PPU incorporates U memory interfaces 1706, one memory interface 1706 per pair of partition units 1700C, where each pair of partition units 1700C is connected to a corresponding memory device. In at least one embodiment, in at least one embodiment, PPU may be connected to up to Y memory devices, such as high bandwidth memory stacks or graphics double-data-rate, version 5, synchronous dynamic random a17ess memory ("GDDR5 SDRAM").

[0323] In at least one embodiment, memory interface 1706 implements a high bandwidth memory second generation ("HBM2") memory interface and Y equals half U. In at least one embodiment, HBM2 memory stacks are located on same physical package as PPU, providing substantial power and area savings compared with GDDR5 SDRAM systems. In at least one embodiment, each HBM2 stack includes, without limitation, four memory dies and Y equals 4, with each HBM2 stack including two 128-bit channels per die for a total of 8 channels and a data bus width of 1024 bits. In at least one embodiment, memory supports Single-Error Correcting

Double-Error Detecting ("SECDED") Error Correction Code ("ECC") to protect data. In at least one embodiment, ECC provides higher reliability for compute applications that are sensitive to data corruption.

[0324] In at least one embodiment, PPU implements a multi-level memory hierarchy. In at least one embodiment, memory partition unit 1700C supports a unified memory to provide a single unified virtual address space for central processing unit ("CPU") and PPU memory, enabling data sharing between virtual memory systems. In at least one embodiment, frequency of accesses by a PPU to memory located on other processors is traced to ensure that memory pages are moved to physical memory of PPU that is accessing pages more frequently. In at least one embodiment, high-speed GPU interconnect 1708 supports address translation services allowing PPU to directly access a CPU's page tables and providing full access to CPU memory by PPU.

[0325] In at least one embodiment, copy engines transfer data between multiple PPUs or between PPUs and CPUs. In at least one embodiment, copy engines can generate page faults for addresses that are not mapped into page tables and memory partition unit 1700C then services page faults, mapping addresses into page table, after which copy engine performs transfer. In at least one embodiment, memory is pinned (in at least one embodiment, non-pageable) for multiple copy engine operations between multiple processors, substantially reducing available memory. In at least one embodiment, with hardware page faulting, addresses can be passed to copy engines without regard as to whether memory pages are resident, and copy process is transparent.

[0326] Data from memory 1704 of FIG. 17A or other system memory is fetched by memory partition unit 1700C and stored in L2 cache 1704, which is located on-chip and is shared between various GPCs, in accordance with at least one embodiment. Each memory partition unit 1700C, in at least one embodiment, includes, without limitation, at least a portion of L2 cache associated with a corresponding memory device. In at least one embodiment, lower level caches are implemented in various units within GPCs. In at least one embodiment, each of SMs 1714 may implement a level one ("L1") cache wherein L1 cache is private memory that is dedicated to a particular SM 1714 and data from L2 cache 1704 is fetched and stored in each of L1 caches for

processing in functional units of SMs 1714. In at least one embodiment, L2 cache 1704 is coupled to memory interface 1706 and XBar 1720.

[0327] ROP unit 1702 performs graphics raster operations related to pixel color, such as color compression, pixel blending, and more, in at least one embodiment. ROP unit 1702, in at least one embodiment, implements depth testing in conjunction with raster engine 1708, receiving a depth for a sample location associated with a pixel fragment from culling engine of raster engine 1708. In at least one embodiment, depth is tested against a corresponding depth in a depth buffer for a sample location associated with fragment. In at least one embodiment, if fragment passes depth test for sample location, then ROP unit 1702 updates depth buffer and transmits a result of depth test to raster engine 1708. It will be appreciated that number of partition units 1700C may be different than number of GPCs and, therefore, each ROP unit 1702 can, in at least one embodiment, be coupled to each of GPCs. In at least one embodiment, ROP unit 1702 tracks packets received from different GPCs and determines which that a result generated by ROP unit 1702 is routed to through XBar1720.

[0328] FIG. 17D illustrates a streaming multi-processor ("SM") 1700D, according to at least one embodiment. In at least one embodiment, SM 1700D is SM 1714 of FIG. 17B. In at least one embodiment, SM 1700D includes, without limitation, an instruction cache 1702; one or more scheduler units 1704; a register file 1708; one or more processing cores ("cores") 1710; one or more special function units ("SFUs") 1712; one or more load/store units ("LSUs") 1714; an interconnect network 1716; a shared memory/level one ("L1") cache 1718; and any suitable combination thereof. In at least one embodiment, a work distribution unit dispatches tasks for execution on general processing clusters ("GPCs") of parallel processing units ("PPUs") and each task is allocated to a particular Data Processing Cluster ("DPC") within a GPC and, if task is associated with a shader program, task is allocated to one of SMs 1700D. In at least one embodiment, scheduler unit 1704 receives tasks from work distribution unit and manages instruction scheduling for one or more thread blocks assigned to SM 1700D. In at least one embodiment, scheduler unit 1704 schedules thread blocks for execution as warps of parallel threads, wherein each thread block is allocated at least one warp. In at least one embodiment, each warp executes threads. In at least one embodiment, scheduler unit 1704 manages a plurality of different thread blocks, allocating warps to different thread blocks and then dispatching

instructions from plurality of different cooperative groups to various functional units (such as processing cores 1710, SFUs 1712, and LSUs 1714) during each clock cycle.

[0329] In at least one embodiment, Cooperative Groups may refer to a programming model for organizing groups of communicating threads that allows developers to express granularity at which threads are communicating, enabling expression of richer, more efficient parallel decompositions. In at least one embodiment, cooperative launch APIs support synchronization amongst thread blocks for execution of parallel algorithms. In at least one embodiment, applications of programming models provide a single, simple construct for synchronizing cooperating threads: a barrier across all threads of a thread block (such as syncthreads() function). However, in at least one embodiment, programmers may define groups of threads at smaller than thread block granularities and synchronize within defined groups to enable greater performance, design flexibility, and software reuse in form of collective group-wide function interfaces. In at least one embodiment, Cooperative Groups enables programmers to define groups of threads explicitly at sub-block (in at least one embodiment, as small as a single thread) and multi-block granularities, and to perform collective operations such as synchronization on threads in a cooperative group. In at least one embodiment, programming model supports clean composition across software boundaries, so that libraries and utility functions can synchronize safely within their local context without having to make assumptions about convergence. In at least one embodiment, Cooperative Groups primitives enable new patterns of cooperative parallelism, including, without limitation, producer-consumer parallelism, opportunistic parallelism, and global synchronization across an entire grid of thread blocks.

[0330] In at least one embodiment, a dispatch unit 1706 is configured to transmit instructions to one or more of functional units and scheduler unit 1704 includes, without limitation, two dispatch units 1706 that enable two different instructions from same warp to be dispatched during each clock cycle. In at least one embodiment, each scheduler unit 1704 includes a single dispatch unit 1706 or additional dispatch units 1706.

[0331] In at least one embodiment, each SM 1700D, in at least one embodiment, includes, without limitation, register file 1708 that provides a set of registers for functional units of SM 1700D. In at least one embodiment, register file 1708 is divided between each of functional units such that each functional unit is allocated a dedicated portion of register file 1708. In at

least one embodiment, register file 1708 is divided between different warps being executed by SM 1700D and register file 1708 provides temporary storage for operands connected to data paths of functional units. In at least one embodiment, each SM 1700D comprises, without limitation, a plurality of L processing cores 1710. In at least one embodiment, SM 1700D includes, without limitation, a large number (such as 128 or more) of distinct processing cores 1710. In at least one embodiment, each processing core 1710, in at least one embodiment, includes, without limitation, a fully-pipelined, single-precision, double-precision, and/or mixed precision processing unit that includes, without limitation, a floating point arithmetic logic unit and an integer arithmetic logic unit. In at least one embodiment, floating point arithmetic logic units implement IEEE 754-2008 standard for floating point arithmetic. In at least one embodiment, processing cores 1710 include, without limitation, 64 single-precision (32-bit) floating point cores, 64 integer cores, 32 double-precision (64-bit) floating point cores, and 8 tensor cores.

[0332] Tensor cores are configured to perform matrix operations in accordance with at least one embodiment. In at least one embodiment, one or more tensor cores are included in processing cores 1710. In at least one embodiment, tensor cores are configured to perform deep learning matrix arithmetic, such as convolution operations for neural network training and inferencing. In at least one embodiment, each tensor core operates on a 4x4 matrix and performs a matrix multiply and accumulate operation D = A X B + C, where A, B, C, and D are 4x4 matrices.

[0333] In at least one embodiment, matrix multiply inputs A and B are 16-bit floating point matrices and accumulation matrices C and D are16-bit floating point or 32-bit floating point matrices. In at least one embodiment, tensor cores operate on 16-bit floating point input data with 32-bit floating point accumulation. In at least one embodiment, 16-bit floating point multiply uses 64 operations and results in a full precision product that is then accumulated using 32-bit floating point addition with other intermediate products for a 4x4x4 matrix multiply. Tensor cores are used to perform much larger two-dimensional or higher dimensional matrix operations, built up from these smaller elements, in at least one embodiment. In at least one embodiment, an API, such as CUDA 9 C++ API, exposes specialized matrix load, matrix multiply and accumulate, and matrix store operations to efficiently use tensor cores from a

CUDA-C++ program. In at least one embodiment, at CUDA level, warp-level interface assumes 16x16 size matrices spanning all 32 threads of warp.

[0334] In at least one embodiment, each SM 1700D comprises, without limitation, M SFUs 1712 that perform special functions (such as attribute evaluation, reciprocal square root, etc.). In at least one embodiment, SFUs 1712 include, without limitation, a tree traversal unit configured to traverse a hierarchical tree data structure. In at least one embodiment, SFUs 1712 include, without limitation, a texture unit configured to perform texture map filtering operations. In at least one embodiment, texture units are configured to load texture maps (such as a 2D array of texels) from memory and sample texture maps to produce sampled texture values for use in shader programs executed by SM 1700D. In at least one embodiment, texture maps are stored in shared memory/L1 cache 1718. In at least one embodiment, texture units implement texture operations such as filtering operations using mip-maps (such as texture maps of varying levels of detail), in accordance with at least one embodiment. In at least one embodiment, each SM 1700D includes, without limitation, two texture units.

[0335] Each SM 1700D comprises, without limitation, N LSUs 1714 that implement load and store operations between shared memory/L1 cache 1718 and register file 1708, in at least one embodiment. Each SM 1700D includes, without limitation, interconnect network 1716 that connects each of functional units to register file 1708 and LSU 1714 to register file 1708 and shared memory/L1 cache 1718 in at least one embodiment. In at least one embodiment, interconnect network 1716 is a crossbar that can be configured to connect any of functional units to any of registers in register file 1708 and connect LSUs 1714 to register file 1708 and memory locations in shared memory/L1 cache 1718.

[0336] In at least one embodiment, shared memory/L1 cache 1718 is an array of on-chip memory that allows for data storage and communication between SM 1700D and primitive engine and between threads in SM 1700D, in at least one embodiment. In at least one embodiment, shared memory/L1 cache 1718 comprises, without limitation, 128KB of storage capacity and is in path from SM 1700D to partition unit. In at least one embodiment, shared memory/L1 cache 1718, in at least one embodiment, is used to cache reads and writes. In at least one embodiment, one or more of shared memory/L1 cache 1718, L2 cache, and memory are backing stores.

[0337] Combining data cache and shared memory functionality into a single memory block provides improved performance for both types of memory accesses, in at least one embodiment. In at least one embodiment, capacity is used or is usable as a cache by programs that do not use shared memory, such as if shared memory is configured to use half of capacity, texture and load/store operations can use remaining capacity. Integration within shared memory/L1 cache 1718 enables shared memory/L1 cache 1718 to function as a high-throughput conduit for streaming data while simultaneously providing high-bandwidth and low-latency access to frequently reused data, in accordance with at least one embodiment. In at least one embodiment, when configured for general purpose parallel computation, a simpler configuration can be used compared with graphics processing. In at least one embodiment, fixed function graphics processing units are bypassed, creating a much simpler programming model. In general purpose parallel computation configuration, work distribution unit assigns and distributes blocks of threads directly to DPCs, in at least one embodiment. In at least one embodiment, threads in a block execute same program, using a unique thread ID in calculation to ensure each thread generates unique results, using SM 1700D to execute program and perform calculations, shared memory/L1 cache 1718 to communicate between threads, and LSU 1714 to read and write global memory through shared memory/L1 cache 1718 and memory partition unit. In at least one embodiment, when configured for general purpose parallel computation, SM 1700D writes commands that scheduler unit 1704 can use to launch new work on DPCs.

[0338] In at least one embodiment, PPU is included in or coupled to a desktop computer, a laptop computer, a tablet computer, servers, supercomputers, a smart-phone (such as a wireless, hand-held device), personal digital assistant ("PDA"), a digital camera, a vehicle, a head mounted display, a hand-held electronic device, and more. In at least one embodiment, PPU is embodied on a single semiconductor substrate. In at least one embodiment, PPU is included in a system-on-a-chip ("SoC") along with one or more other devices such as additional PPUs, memory, a reduced instruction set computer ("RISC") CPU, a memory management unit ("MMU"), a digital-to-analog converter ("DAC"), and like.

[0339] In at least one embodiment, PPU may be included on a graphics card that includes one or more memory devices. A graphics card may be configured to interface with a PCIe slot on a

motherboard of a desktop computer. In at least one embodiment, PPU may be an integrated graphics processing unit ("iGPU") included in chipset of motherboard.

[0340] Inference and/or training logic 615 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 615 are provided below in conjunction with FIGS. 6B and/or 6C. In at least one embodiment, deep learning application processor is used to train a machine learning model, such as a neural network, to predict or infer information provided to SM 1700D. In at least one embodiment, SM 1700D is used to infer or predict information based on a trained machine learning model (such as neural network) that has been trained by another processor or system or by SM 1700D. In at least one embodiment, SM 1700D may be used to perform one or more neural network use cases described herein.

[0341] In at least one embodiment, a single semiconductor platform may refer to a sole unitary semiconductor-based integrated circuit or chip. In at least one embodiment, multi-chip modules may be used with increased connectivity which simulate on-chip operation, and make substantial improvements over utilizing a central processing unit ("CPU") and bus implementation. In at least one embodiment, various modules may also be situated separately or in various combinations of semiconductor platforms per desires of user.

[0342] In at least one embodiment, computer programs in form of machine-readable executable code or computer control logic algorithms are stored in main memory 804 and/or secondary storage. Computer programs, if executed by one or more processors, enable system 800 to perform various functions in accordance with at least one embodiment. In at least one embodiment, memory 804, storage, and/or any other storage are possible examples of computer-readable media. In at least one embodiment, secondary storage may refer to any suitable storage device or system such as a hard disk drive and/or a removable storage drive, representing a floppy disk drive, a magnetic tape drive, a compact disk drive, digital versatile disk ("DVD") drive, recording device, universal serial bus ("USB") flash memory, etc. In at least one embodiment, architecture and/or functionality of various previous figures are implemented in context of CPU 802; parallel processing system 812; an integrated circuit capable of at least a portion of capabilities of both CPU 802; parallel processing system 812; a chipset (such as a

group of integrated circuits designed to work and sold as a unit for performing related functions, etc.); and any suitable combination of integrated circuit(s).

[0343] In at least one embodiment, architecture and/or functionality of various previous figures are implemented in context of a general computer system, a circuit board system, a game console system dedicated for entertainment purposes, an application-specific system, and more. In at least one embodiment, computer system 800 may take form of a desktop computer, a laptop computer, a tablet computer, servers, supercomputers, a smart-phone (such as a wireless, hand-held device), personal digital assistant ("PDA"), a digital camera, a vehicle, a head mounted display, a hand-held electronic device, a mobile phone device, a television, workstation, game consoles, embedded system, and/or any other type of logic.

[0344] In at least one embodiment, parallel processing system 812 includes, without limitation, a plurality of parallel processing units ("PPUs") 814 and associated memories 816. In at least one embodiment, PPUs 814 are connected to a host processor or other peripheral devices via an interconnect 818 and a switch 820 or multiplexer. In at least one embodiment, parallel processing system 812 distributes computational tasks across PPUs 814 which can be parallelizable — for example, as part of distribution of computational tasks across multiple graphics processing unit ("GPU") thread blocks. In at least one embodiment, memory is shared and accessible (such as for read and/or write access) across some or all of PPUs 814, although such shared memory may incur performance penalties relative to use of local memory and registers resident to a PPU 814. In at least one embodiment, operation of PPUs 814 is synchronized through use of a command such as __syncthreads(), wherein all threads in a block (such as executed across multiple PPUs 814) to reach a certain point of execution of code before proceeding.

[0345] Other variations are within spirit of present disclosure. Thus, while disclosed techniques are susceptible to various modifications and alternative constructions, certain illustrated embodiments thereof are shown in drawings and have been described above in detail. It should be understood, however, that there is no intention to limit disclosure to specific form or forms disclosed, but on contrary, intention is to cover all modifications, alternative constructions, and equivalents falling within spirit and scope of disclosure, as defined in appended claims.

[0346] Use of terms "a" and "an" and "the" and similar referents in context of describing

disclosed embodiments (especially in context of following claims) are to be construed to cover both singular and plural, unless otherwise indicated herein or clearly contradicted by context, and not as a definition of a term. Terms "including," "having," "including," and "containing" are to be construed as open-ended terms (meaning "including, but not limited to,") unless otherwise noted. Term "connected," when unmodified and referring to physical connections, is to be construed as partly or wholly contained within, attached to, or joined together, even if there is something intervening. Recitation of ranges of values herein are merely intended to serve as a shorthand method of referring individually to each separate value falling within range, unless otherwise indicated herein and each separate value is incorporated into specification as if it were individually recited herein. Use of a set (such as a set of items) or subset, unless otherwise noted or contradicted by context, is to be construed as a nonempty collection including one or more members. Further, unless otherwise noted or contradicted by context, a subset of a corresponding set does not necessarily denote a proper subset of corresponding set, but subset and corresponding set may be equal.

[0347] Conjunctive language, such as phrases of form "at least one of A, B, and C," or "at least one of A, B and C," unless specifically stated otherwise or otherwise clearly contradicted by context, is otherwise understood with context as used in general to present that an item, term, etc., may be either A or B or C, or any nonempty subset of set of A and B and C. For instance, in illustrative example of a set having three members, conjunctive phrases "at least one of A, B, and C" and "at least one of A, B and C" refer to any of following sets: {A}, {B}, {C}, {A, B}, {A, C}, {B, C}, {A, B, C}. Thus, such conjunctive language may not be intended to imply that certain embodiments require at least one of A, at least one of B, and at least one of C each to be present. In addition, unless otherwise noted or contradicted by context, a plurality indicates a state of being plural (such as a plurality of items indicates multiple items). A plurality is at least two items, but can be more when so indicated either explicitly or by context. Further, unless stated otherwise or otherwise clear from context, based on means based at least in part on and not based solely on.

[0348] Operations of processes described herein can be performed in any suitable order unless otherwise indicated herein or otherwise clearly contradicted by context. In at least one embodiment, a process such as those processes described herein (or variations and/or

combinations thereof) is performed under control of one or more computer systems configured with executable instructions and is implemented as code (such as executable instructions, one or more computer programs or one or more applications) executing collectively on one or more processors, by hardware or combinations thereof. In at least one embodiment, code is stored on a computer-readable storage medium, for example, in form of a computer program including a plurality of instructions executable by one or more processors. In at least one embodiment, a computer-readable storage medium is a non-transitory computer-readable storage medium that excludes transitory signals (such as a propagating transient electric or electromagnetic transmission) but includes non-transitory data storage circuitry (such as buffers, cache, and queues) within transceivers of transitory signals. In at least one embodiment, code (such as executable code or source code) is stored on a set of one or more non-transitory computer-readable storage media having stored thereon executable instructions (or other memory to store executable instructions) that, when executed (in at least one embodiment, as a result of being executed) by one or more processors of a computer system, cause computer system to perform operations described herein. A set of non-transitory computer-readable storage media, in at least one embodiment, includes multiple non-transitory computer-readable storage media and one or more of individual non-transitory storage media of multiple non-transitory computer-readable storage media lack all of code while multiple non-transitory computer-readable storage media collectively store all of code. In at least one embodiment, executable instructions are executed such that different instructions are executed by different processors — for example, a non-transitory computer-readable storage medium store instructions and a main central processing unit ("CPU") executes some of instructions while a graphics processing unit ("GPU") executes other instructions. In at least one embodiment, different components of a computer system have separate processors and different processors execute different subsets of instructions.

[0349] Accordingly, in at least one embodiment, computer systems are configured to implement one or more services that singly or collectively perform operations of processes described herein and such computer systems are configured with applicable hardware and/or software that enable performance of operations. Further, a computer system that implements at least one embodiment of present disclosure is a single device and, in another embodiment, is a distributed computer system including multiple devices that operate differently such that distributed computer system performs operations described herein and such that a single device

does not perform all operations.

[0350] Use of any and all examples, or exemplary language (such as "such as") provided herein, is intended merely to better illuminate embodiments of disclosure and does not pose a limitation on scope of disclosure unless otherwise claimed. No language in specification should be construed as indicating any non-claimed element as essential to practice of disclosure.

[0351] All references, including publications, patent applications, and patents, cited herein are hereby incorporated by reference to same extent as if each reference were individually and specifically indicated to be incorporated by reference and were set forth in its entirety herein.

[0352] In description and claims, terms "coupled" and "connected," along with their derivatives, may be used. It should be understood that these terms may be not intended as synonyms for each other. Rather, in particular examples, "connected" or "coupled" may be used to indicate that two or more elements are in direct or indirect physical or electrical contact with each other. "Coupled" may also mean that two or more elements are not in direct contact with each other, but yet still co-operate or interact with each other.

[0353] Unless specifically stated otherwise, it may be appreciated that throughout specification, references to processing, computing, calculating, determining, or the like, refer to action and/or processes of a computer or computing system, or similar electronic computing device, that manipulate and/or transform data represented as physical, such as electronic, quantities within computing system's registers and/or memories into other data similarly represented as physical quantities within computing system's memories, registers or other such information storage, transmission or display devices.

[0354] In a similar manner, a processor may refer to any device or portion of a device that processes electronic data from registers and/or memory and transform that electronic data into other electronic data that may be stored in registers and/or memory. As non-limiting examples, "processor" may be a CPU or a GPU. A "computing platform" may include one or more processors. As used herein, "software" processes may include, for example, software and/or hardware entities that perform work over time, such as tasks, threads, and intelligent agents. Also, each process may refer to multiple processes, for carrying out instructions in sequence or in parallel, continuously or intermittently. Terms "system" and "method" are used herein

interchangeably insofar as system may embody one or more methods and methods may be considered a system.

[0355] In present document, references may be made to obtaining, acquiring, receiving, or inputting analog or digital data into a subsystem, computer system, or computer-implemented machine. Obtaining, acquiring, receiving, or inputting analog and digital data can be accomplished in a variety of ways such as by receiving data as a parameter of a function call or a call to an application programming interface. In some implementations, process of obtaining, acquiring, receiving, or inputting analog or digital data can be accomplished by transferring data via a serial or parallel interface. In another implementation, process of obtaining, acquiring, receiving, or inputting analog or digital data can be accomplished by transferring data via a computer network from providing entity to acquiring entity. References may also be made to providing, outputting, transmitting, sending, or presenting analog or digital data. In various examples, process of providing, outputting, transmitting, sending, or presenting analog or digital data can be accomplished by transferring data as an input or output parameter of a function call, a parameter of an application programming interface or interprocess communication mechanism.

[0356] Although discussion above sets forth example implementations of described techniques, other architectures may be used to implement described functionality, and are intended to be within scope of this disclosure. Furthermore, although specific distributions of responsibilities are defined above for purposes of discussion, various functions and responsibilities might be distributed and divided in different ways, depending on circumstances.

[0357] Furthermore, although subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that subject matter claimed in appended claims is not necessarily limited to specific features or acts described. Rather, specific features and acts are disclosed as exemplary forms of implementing the claims.

CLAIMS

WHAT IS CLAIMED IS:

1. A cooling system for a datacenter device, comprising:

a plurality of fins between a first plate and a second plate to dissipate a first amount of heat to an environment in a first configuration of the plurality of fins, the first plate movable relative to the second plate to expose a surface area of the plurality of fins to the environment in a second configuration of the plurality of fins and to dissipate a second amount of heat more than the first amount of heat.

2. The cooling system of claim 1, further comprising:

the surface area exposed in the second configuration more than a primary surface area of the plurality of fins in the first configuration.

3. The cooling system of claim 1, further comprising:

intermediate configurations of the plurality of fins between the first configuration and the second configuration to provide intermediate surface areas to enable dissipation of third amounts of heat that is relative with individual intermediate configurations of the plurality of fins.

4. The cooling system of claim 1, further comprising:

a gear subsystem, an electromagnetic subsystem, a thermoelectric generator subsystem, a thermo-reactive subsystem, or a pneumatic subsystem to move the first plate relative to the second plate in response to a sensed heat from an associated computing component or from the environment.

5. The cooling system of claim 1, further comprising:

at least one ribbon feature associated with the plurality of fins to enable individual fins to comprise an overlapping portion that is at least partially shielded from the environment in the first configuration and that is separated to expose the overlapping portion to the environment in the second configuration.

6. The cooling system of claim 1, further comprising:

at least one ribbon feature associated with the plurality of fins and formed in part from a bimorph material to enable the first plate movable relative to the second plate upon a sensed heat acting on the bimorph material.

7. The cooling system of claim 1, further comprising:

fluid or gas lines to receive a cooling fluid from a cooling loop of a datacenter hosting the datacenter device;

a pneumatic subsystem to use the cooling fluid to extend a piston and to cause the first plate to move relative to the second plate to expose the surface area of the plurality of fins in the second configuration of the plurality of fins.

8. The cooling system of claim 1, further comprising:

a learning subsystem comprising at least one processor to evaluate temperatures sensed from components, servers, or racks in the datacenter, with surface area of the plurality of fins associated with different amounts of heat dissipation, and to provide an output associated with at least a temperature to facilitate the movement of the first plate relative to the second plate to expose the surface area of the plurality of fins.

9. The cooling system of claim 8, further comprising:

at least one controller facilitating the movement of the first plate relative to the second plate; and

the learning subsystem executing a machine learning model to:

process the temperatures using multiple neuron levels of the machine learning model having the temperatures and having prior associated surface areas of the plurality of fins; and

provide the output associated with at least a temperature to the at least one controller, the output provided after an evaluation of the prior associated surface areas of the plurality of fins and prior associated temperatures.

10. At least one processor for a cooling system, comprising:

at least one logic unit to control movement associated with a plurality of fins between a first plate and a second plate, the plurality of fins to dissipate a first amount of heat to an environment in a first configuration of the plurality of fins, the first plate movable relative to

the second plate to expose a surface area of the plurality of fins to the environment in a second configuration of the plurality of fins.

11. The least one processor of claim 10, further comprising:

a learning subsystem to evaluate temperatures sensed from components, servers, or racks in the datacenter, with surface area of the plurality of fins associated with different amounts of heat dissipation, and to provide an output associated with at least a temperature to facilitate the movement of the first plate relative to the second plate to expose the surface area of the plurality of fins, the surface area more than a primary surface area of the plurality of fins.

12. The least one processor of claim 11, further comprising:

the learning subsystem executing a machine learning model to:

process the temperatures using multiple neuron levels of the machine learning model having the temperatures and having prior associated surface areas of the plurality of fins; and

provide the output associated with at least a temperature to at least one controller, the output provided after an evaluation of the prior associated surface areas of the plurality of fins and prior associated temperatures.

13. The at least one processor of claim 10, further comprising:

an instruction output to communicate the output associated with the controller to facilitate the movement of the first plate relative to the second plate.

14. The at least one processor of claim 10, further comprising:

the at least one logic unit adapted to receive a temperature value from a temperature sensor associated with a datacenter device, the temperature sensor adapted to facilitate movement of the first plate movable relative to the second plate.

15. At least one processor for a cooling system, comprising:

at least one logic unit to train one or more neural network having hidden layers of neurons to evaluate temperatures sensed from components, servers, or racks in the datacenter, with surface area of plurality of fins associated with different amounts of heat dissipation from the plurality of fins, and to provide an output associated with at least a temperature to facilitate

movement of a first plate relative to a second plate having the plurality of fins there between to expose a surface area of the plurality of fins.

16. The at least one processor of claim 15, further comprising:

the at least one logic unit to output at least one instruction associated with at least a temperature to facilitate the movement of the first plate relative to the second plate having the plurality of fins there between to expose the surface area of the plurality of fins.

17. The at least one processor of claim 15, further comprising:

at least an instruction output to communicate the output associated with a controller to facilitate the movement of the first plate relative to the second plate.

18. The at least one processor of claim 15, further comprising:

the at least one logic unit adapted to receive a temperature value from a temperature sensor associated with a datacenter device, the temperature sensor adapted to facilitate movement of the first plate movable relative to the second plate.

19. A datacenter cooling system, comprising:

at least one processor to train one or more neural network having hidden layers of neurons to evaluate temperatures sensed from components, servers, or racks in the datacenter, with surface area of a plurality of fins associated with different amounts of heat dissipation from the plurality of fins, and to provide an output associated with at least a temperature to facilitate movement of a first plate relative to a second plate having the plurality of fins there between to expose a surface area of the plurality of fins.

20. The datacenter cooling system of claim 19, further comprising:

the at least one processor to output at least one instruction associated with at least a temperature to facilitate the movement of the first plate relative to the second plate having the plurality of fins there between to expose the surface area of the plurality of fins.

21. The datacenter cooling system of claim 19, further comprising:

at least an instruction output of the at least one processor to communicate the output associated with a controller to facilitate the movement of the first plate relative to the second plate.

22. The datacenter cooling system of claim 19, further comprising:

the at least one processor adapted to receive a temperature value from a temperature sensor associated with a datacenter device, the temperature sensor adapted to facilitate movement of the first plate movable relative to the second plate.

23. A heat sink comprising plurality of fins that are movable to dissipate heat to an environment by exposing a surface area of the plurality of fins in a first configuration that is more than a primary surface area of the plurality of fins in a second configuration.

24. The heat sink of claim 23, further comprising:

a processorless subsystem to cause the plurality of fins to move.

25. The heat sink of claim 24, further comprising:

the processorless subsystem enabled by bimorph metal associated with individual fins, the bimorph metal to unfold a portion of the individual fins, thereby exposing the surface area of the plurality of fins.

26. The heat sink of claim 23, further comprising:

the plurality of fins adapted to expand in response to heat from an associated component being above a threshold and adapted to contract in response to the heat being below the threshold.

27. A method for cooling a datacenter device, comprising:

providing a plurality of fins between a first plate and a second plate to dissipate a first amount of heat to an environment in a first configuration of the plurality of fins, the first plate movable relative to the second plate to expose a surface area of the plurality of fins to the environment in a second configuration of the plurality of fins and to dissipate a second amount of heat more than the first amount of heat.

28. The method of claim 27, further comprising:

providing a gear subsystem, an electromagnetic subsystem, a thermoelectric generator subsystem, a thermo-reactive subsystem, or a pneumatic subsystem to move the first plate relative to the second plate in response to a sensed heat from an associated computing component or from the environment.

29. The method of claim 26, further comprising:

associating at least one ribbon feature with the plurality of fins;

enabling, by the at least one ribbon feature, individual fins to comprise an overlapping portion that is at least partially shielded from the environment in the first configuration; and

enabling a change in structure of the at least one ribbon feature to expose the overlapping portion to the environment in the second configuration.

30. The method of claim 26, further comprising:

receiving a cooling fluid from a cooling loop of a datacenter hosting the datacenter device using fluid or gas lines;

extending a piston associated with a pneumatic subsystem using the cooling fluid to cause the first plate to move relative to the second plate to expose the surface area of the plurality of fins in the second configuration of the plurality of fins.

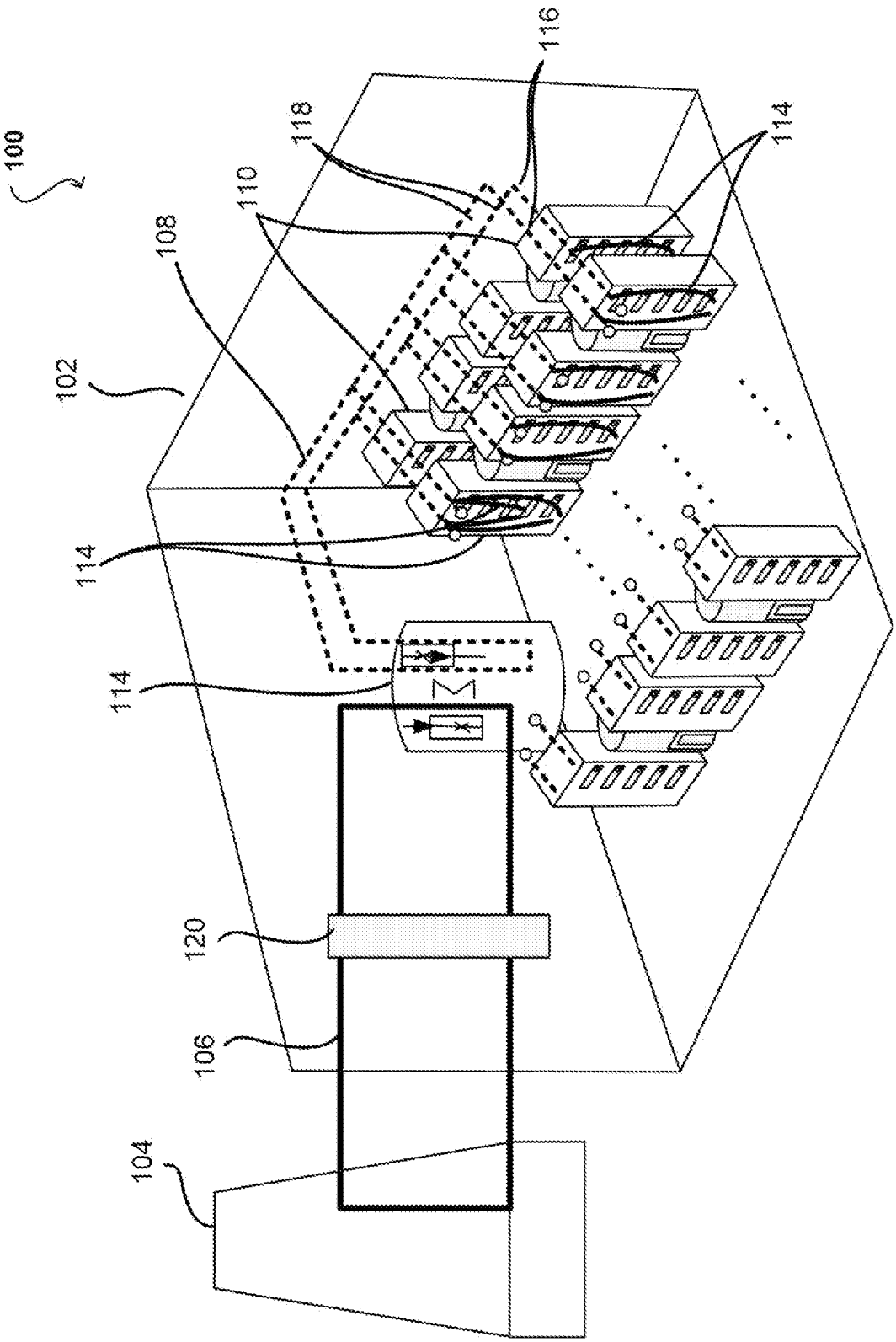

FIG. 1

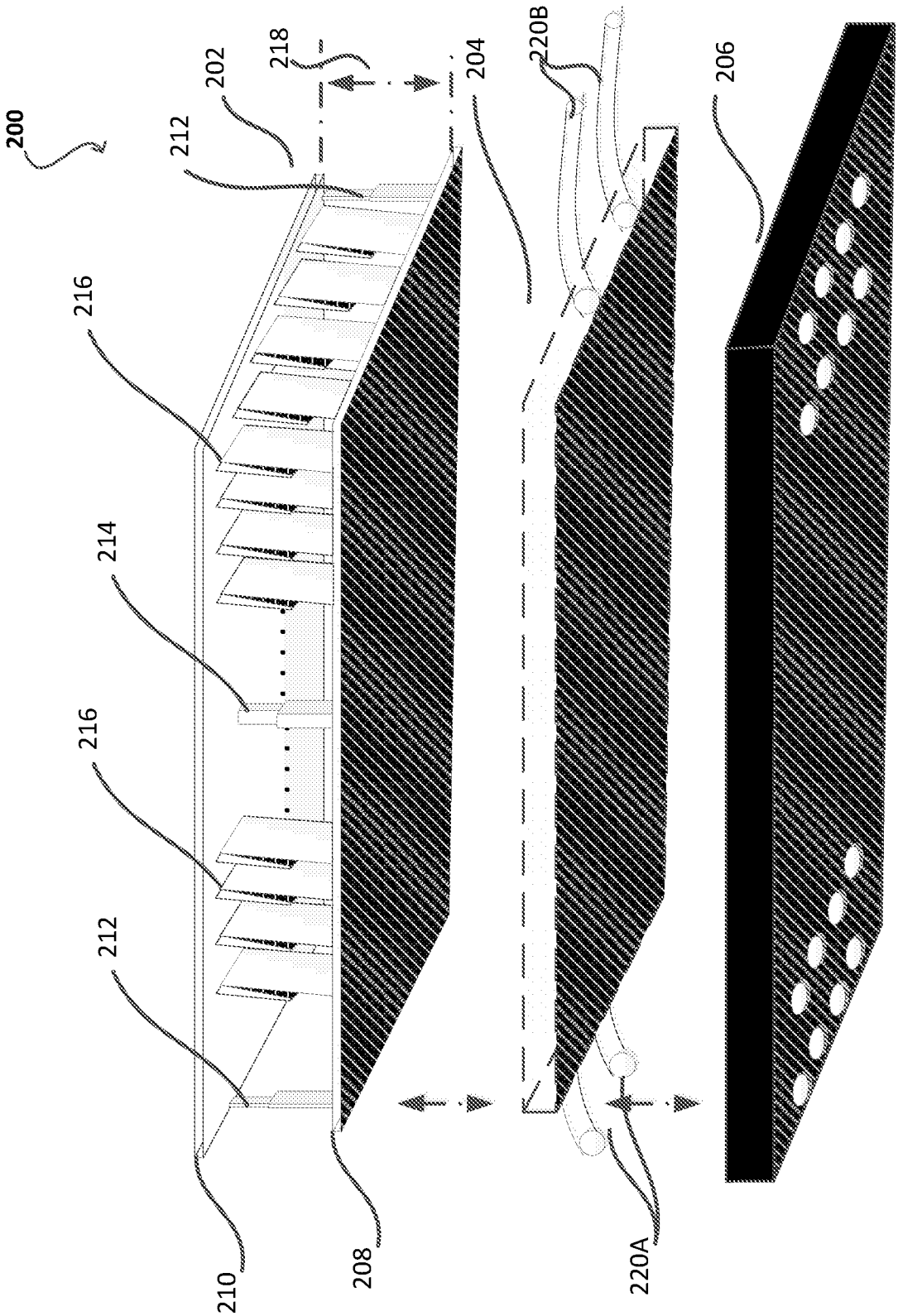

FIG. 2A

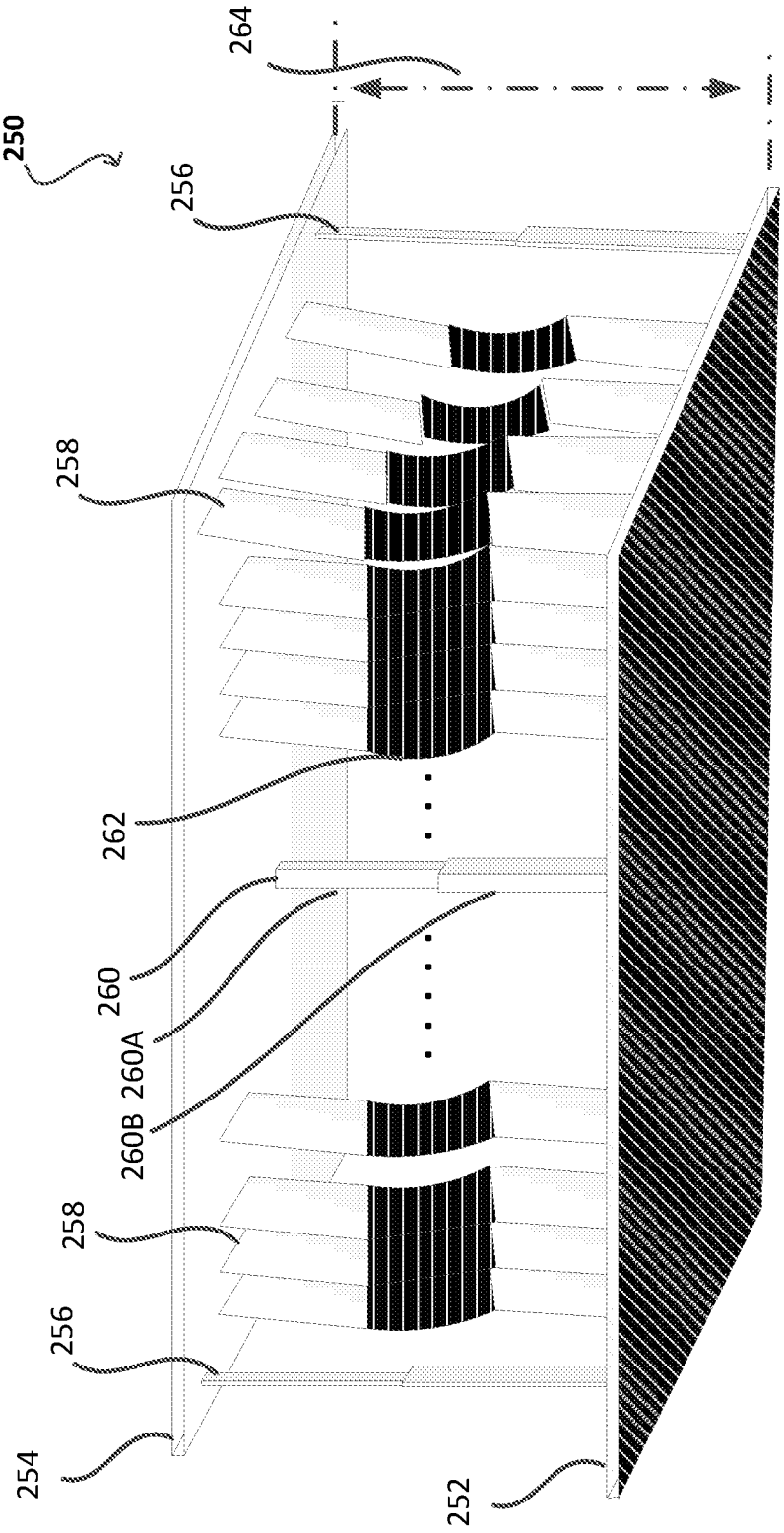

FIG. 2B

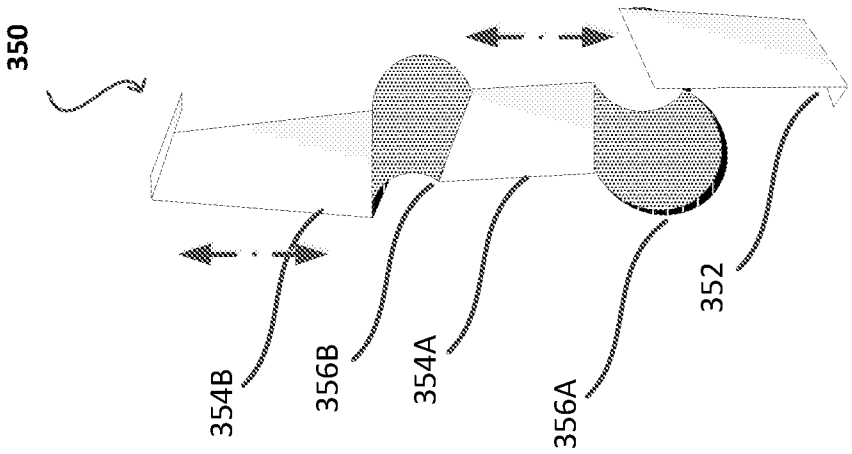

FIG. 3C

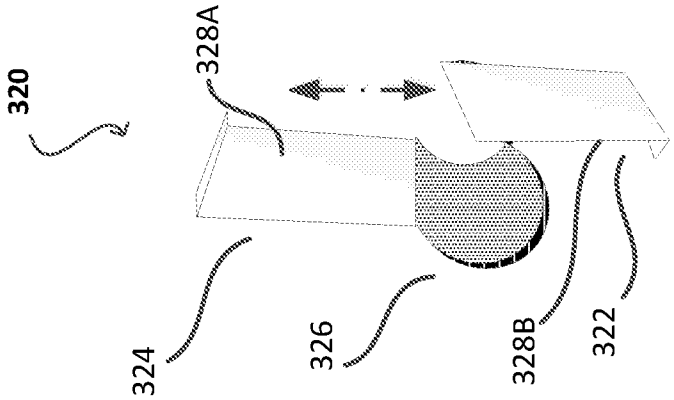

FIG. 3B

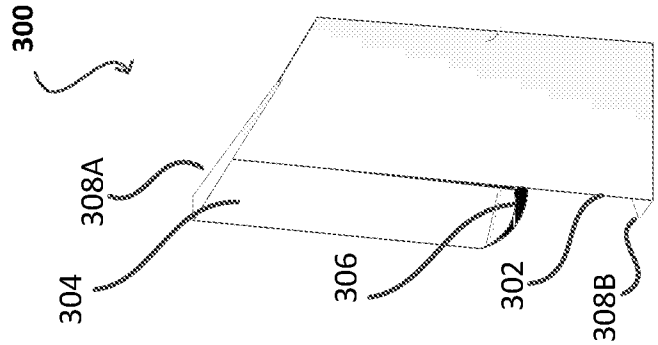

FIG. 3A

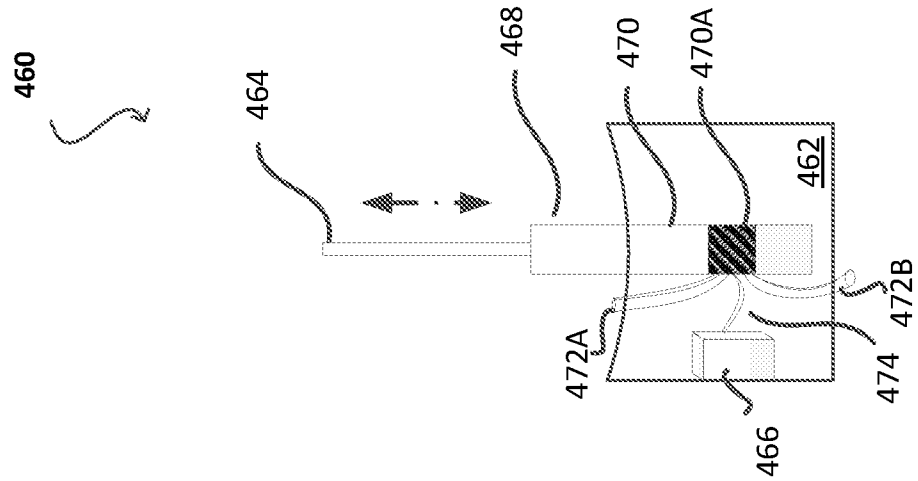

FIG. 4C

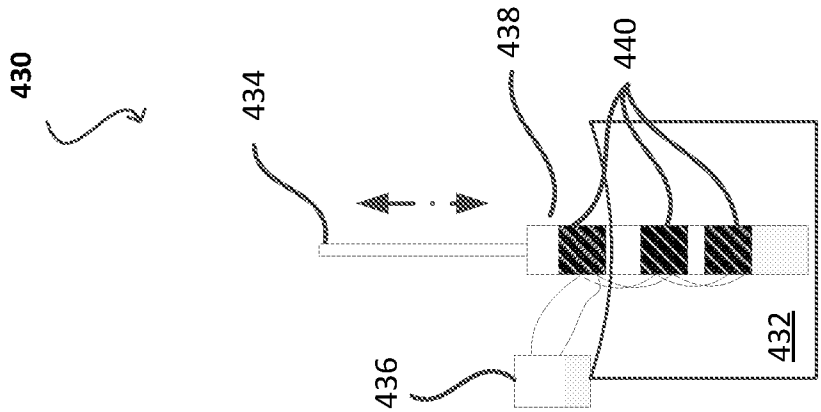

FIG. 4B

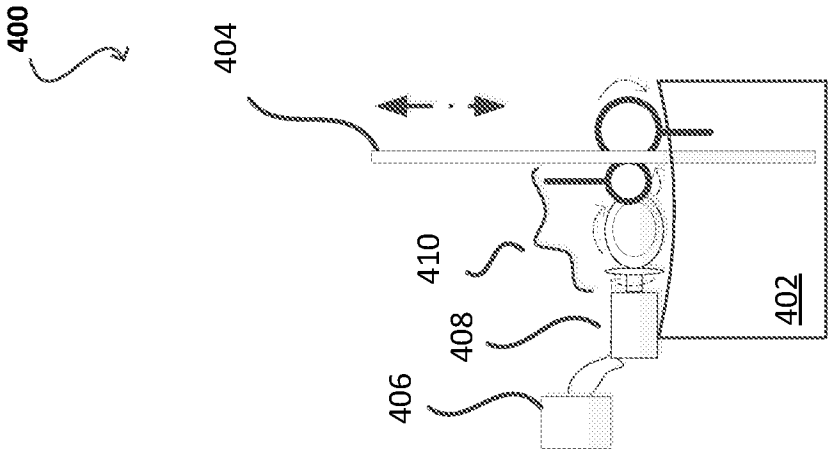

FIG. 4A

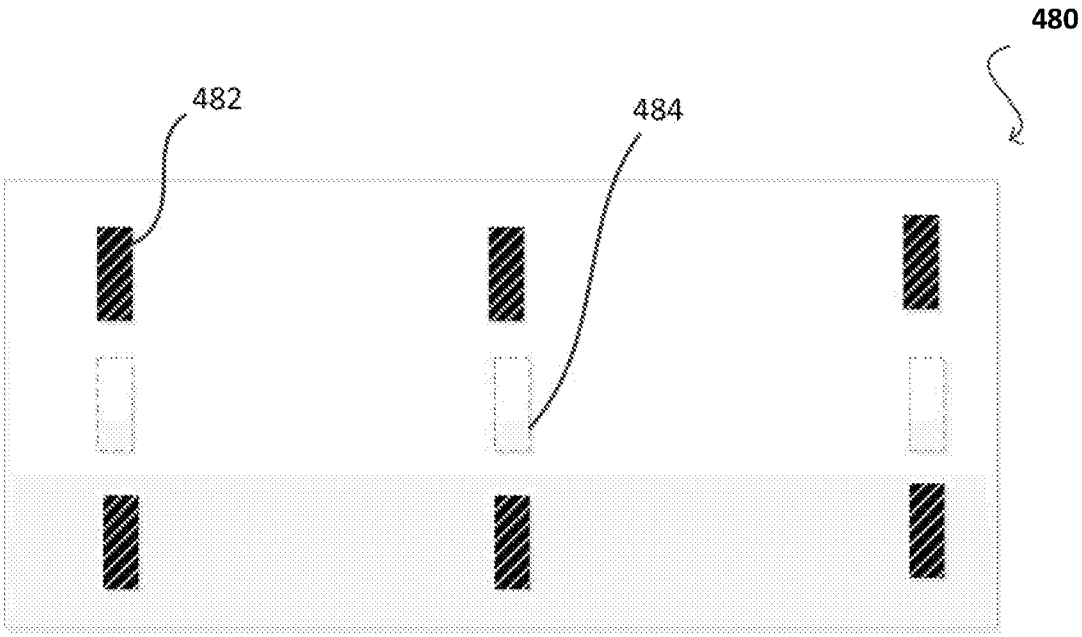

FIG. 4D

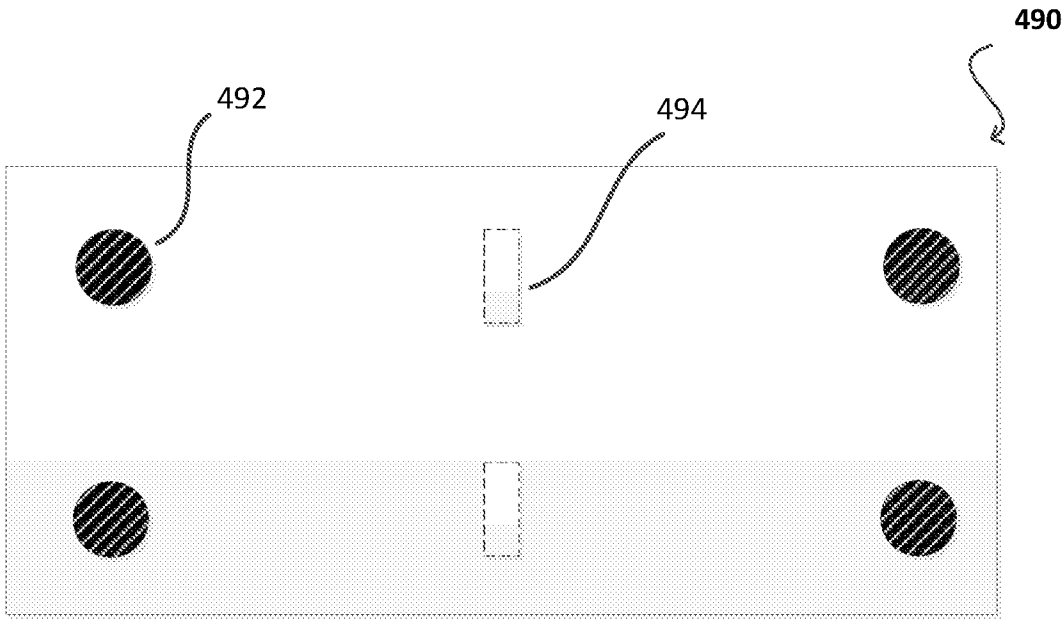

FIG. 4E

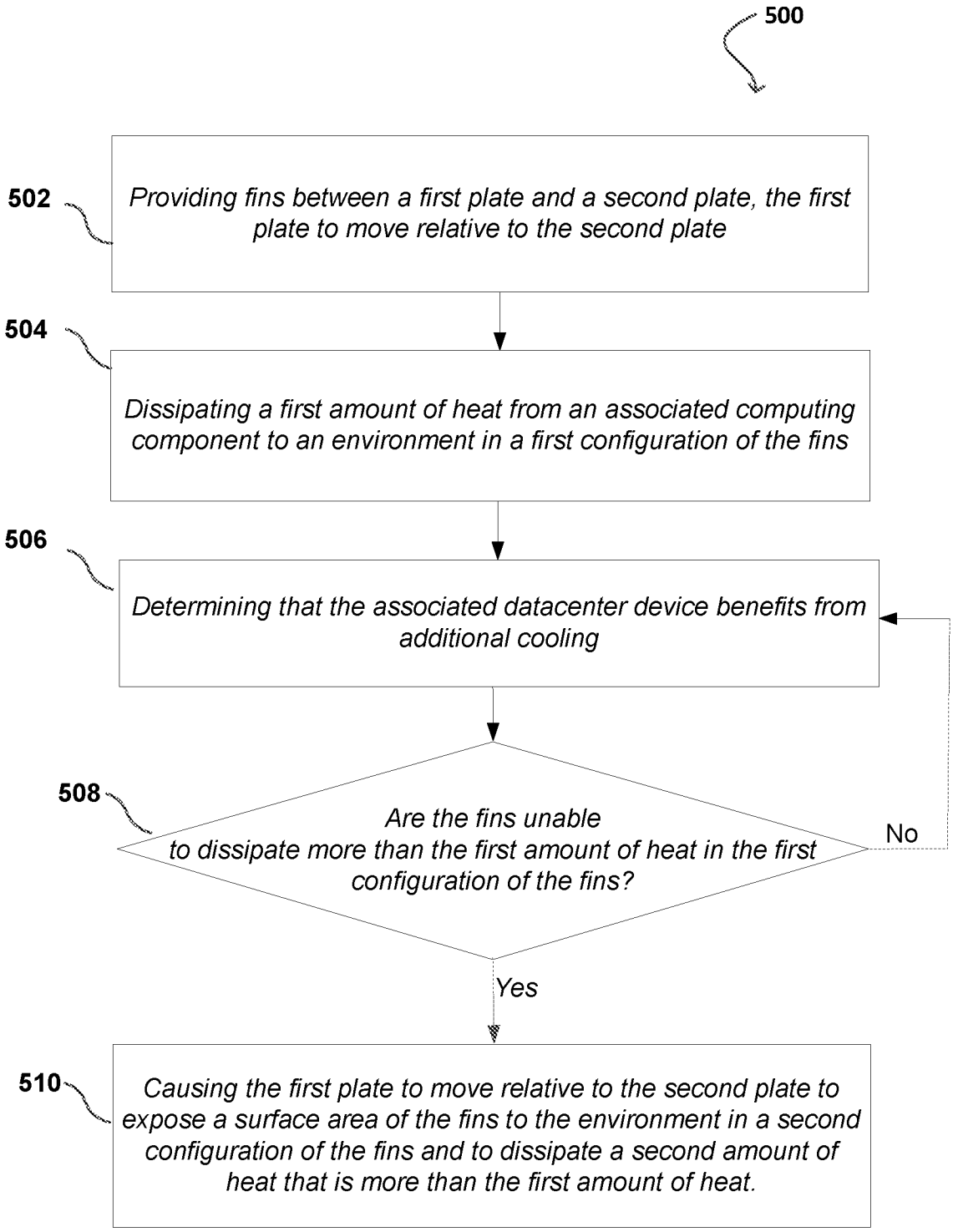
500
502
Providing fins between a first plate and a second plate, the first plate to move relative to the second plate
504
Dissipating a first amount of heat from an associated computing component to an environment in a first configuration of the fins
506
Determining that the associated datacenter device benefits from additional cooling
508
Are the fins unable to dissipate more than the first amount of heat in the first configuration of the fins?
No
Yes
510
Causing the first plate to move relative to the second plate to expose a surface area of the fins to the environment in a second configuration of the fins and to dissipate a second amount of heat that is more than the first amount of heat.

FIG. 5

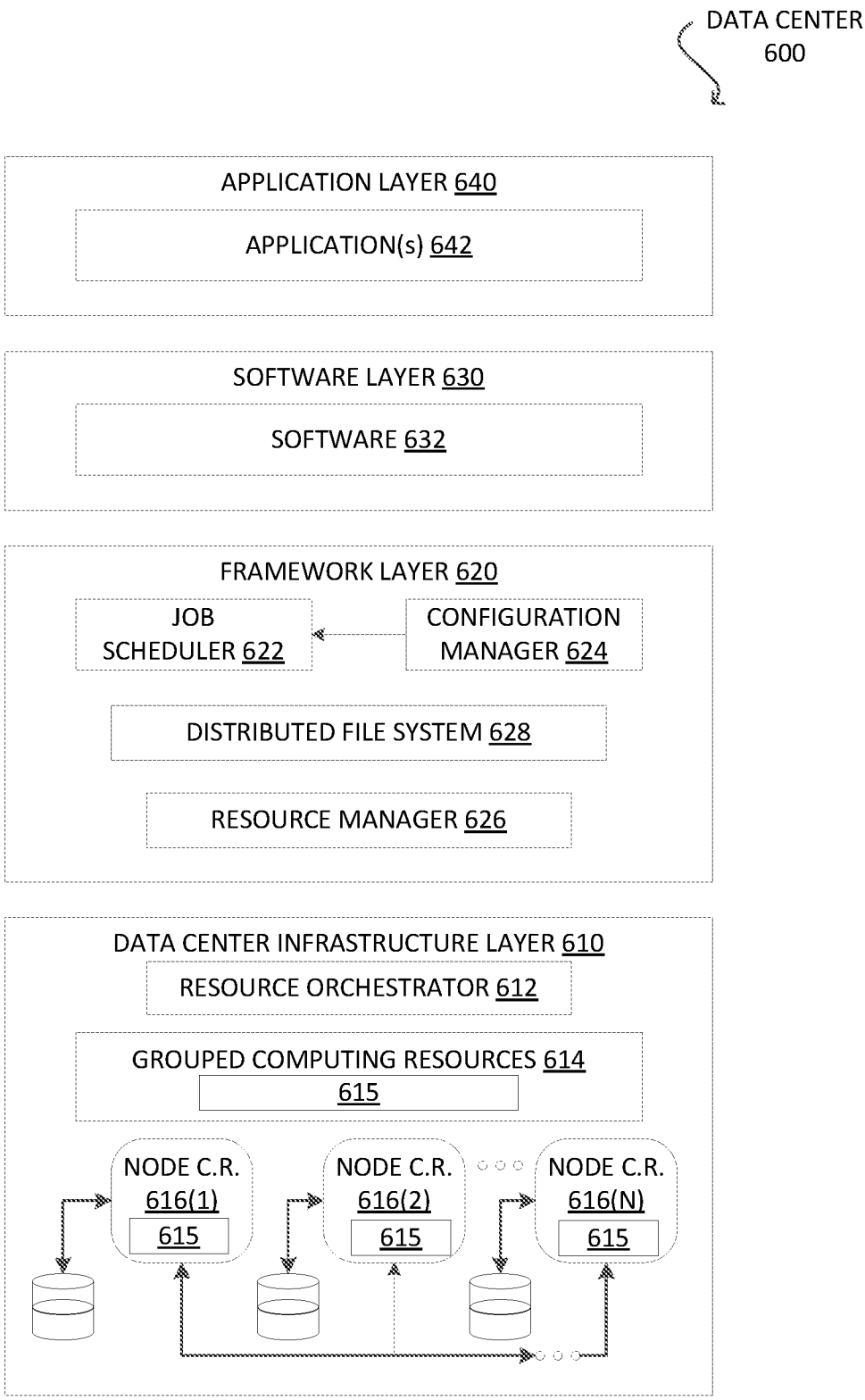
DATA CENTER
600
APPLICATION LAYER 640
APPLICATION(s) 642
SOFTWARE LAYER 630
SOFTWARE 632
FRAMEWORK LAYER 620
JOB SCHEDULER 622
CONFIGURATION MANAGER 624
DISTRIBUTED FILE SYSTEM 628
RESOURCE MANAGER 626
DATA CENTER INFRASTRUCTURE LAYER 610
RESOURCE ORCHESTRATOR 612
GROUPED COMPUTING RESOURCES 614
615
NODE C.R. 616(1)
615
NODE C.R. 616(2)
615
NODE C.R. 616(N)
615

FIG. 6A

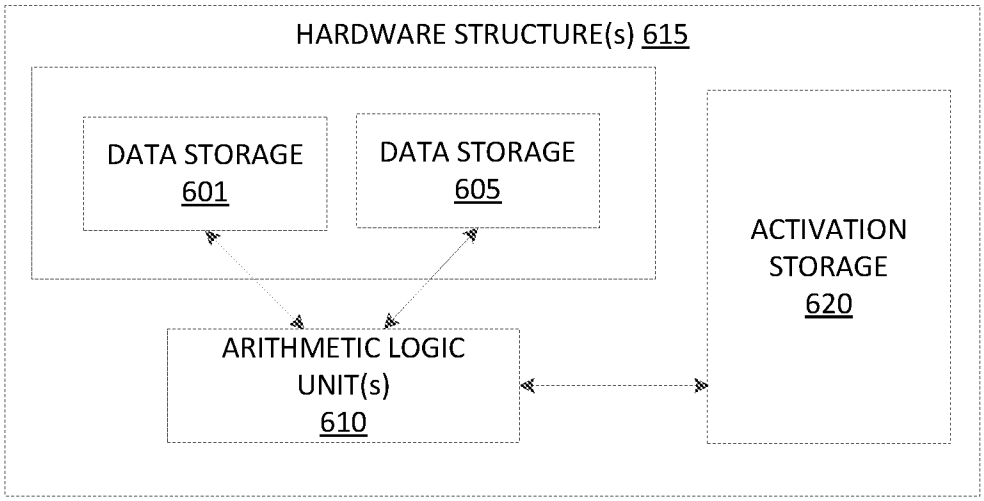
HARDWARE STRUCTURE(s) 615
DATA STORAGE
601
DATA STORAGE
605
ACTIVATION
STORAGE
620
ARITHMETIC LOGIC
UNIT(s)
610

FIG. 6B

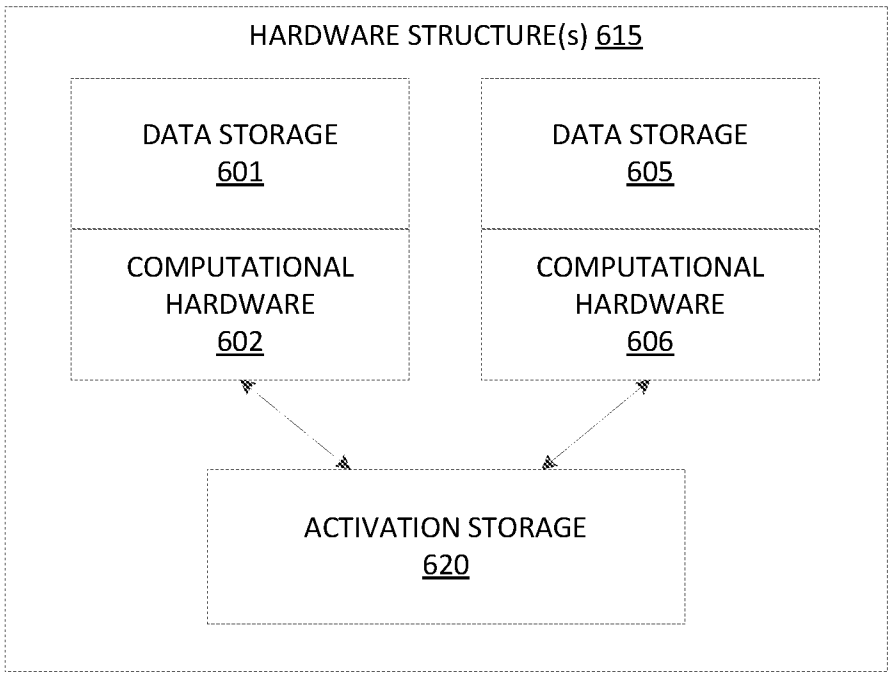
HARDWARE STRUCTURE(s) 615
DATA STORAGE
601
DATA STORAGE
605
COMPUTATIONAL
HARDWARE
602
COMPUTATIONAL
HARDWARE
606
ACTIVATION STORAGE
620

FIG. 6C

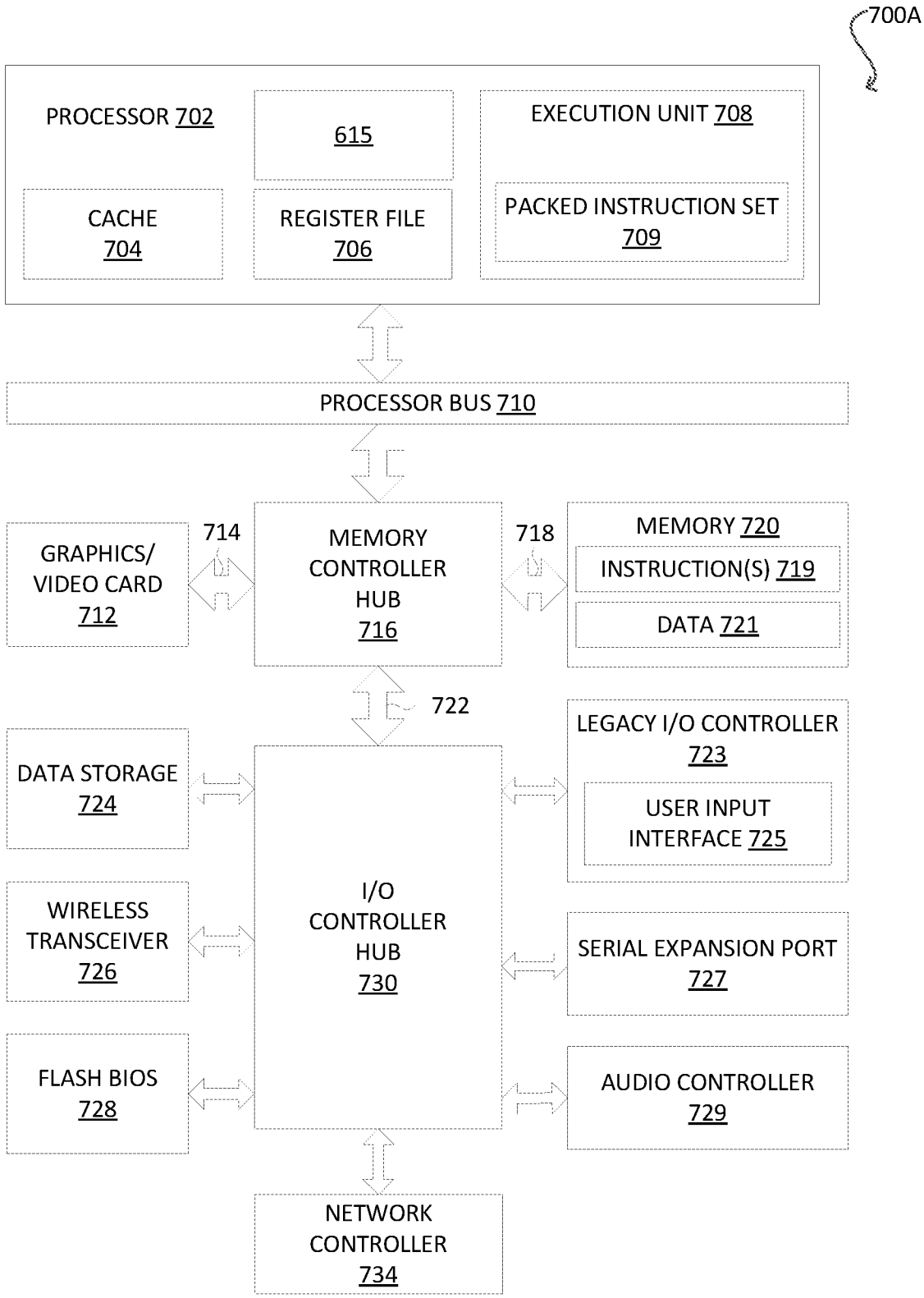
700A
PROCESSOR 702
615
EXECUTION UNIT 708
CACHE 704
REGISTER FILE 706
PACKED INSTRUCTION SET 709
PROCESSOR BUS 710
GRAPHICS/ VIDEO CARD 712
714
MEMORY CONTROLLER HUB 716
718
MEMORY 720
INSTRUCTION(S) 719
DATA 721
722
DATA STORAGE 724
LEGACY I/O CONTROLLER 723
USER INPUT INTERFACE 725
WIRELESS TRANSCEIVER 726
I/O CONTROLLER HUB 730
SERIAL EXPANSION PORT 727
FLASH BIOS 728
AUDIO CONTROLLER 729
NETWORK CONTROLLER 734

FIG. 7A

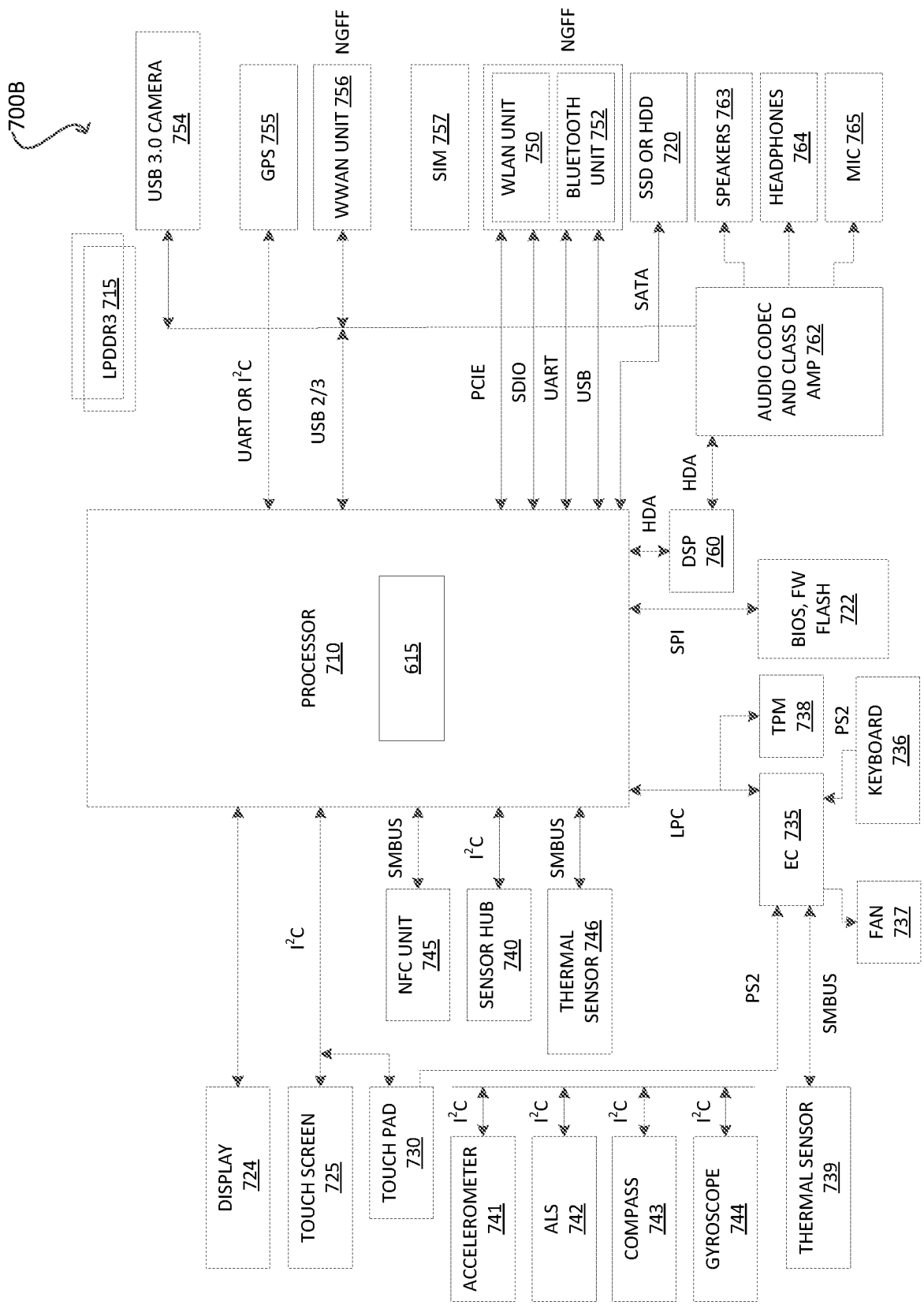
700B
USB 3.0 CAMERA 754
GPS 755
WWAN UNIT 756
NGFF
SIM 757
WLAN UNIT 750
BLUETOOTH UNIT 752
NGFF
SSD OR HDD 720
SPEAKERS 763
HEADPHONES 764
MIC 765
LPDDR3 715
UART OR I2C
USB 2/3
PCIE
SDIO
UART
USB
SATA
AUDIO CODEC AND CLASS D AMP 762
HDA
HDA
DSP 760
PROCESSOR 710
615
SPI
BIOS, FW FLASH 722
TPM 738
PS2
KEYBOARD 736
LPC
EC 735
SMBUS
I2C
SMBUS
I2C
SMBUS
NFC UNIT 745
SENSOR HUB 740
THERMAL SENSOR 746
FAN 737
PS2
SMBUS
DISPLAY 724
TOUCH SCREEN 725
TOUCH PAD 730
I2C
I2C
I2C
I2C
ACCELEROMETER 741
ALS 742
COMPASS 743
GYROSCOPE 744
THERMAL SENSOR 739

FIG. 7B

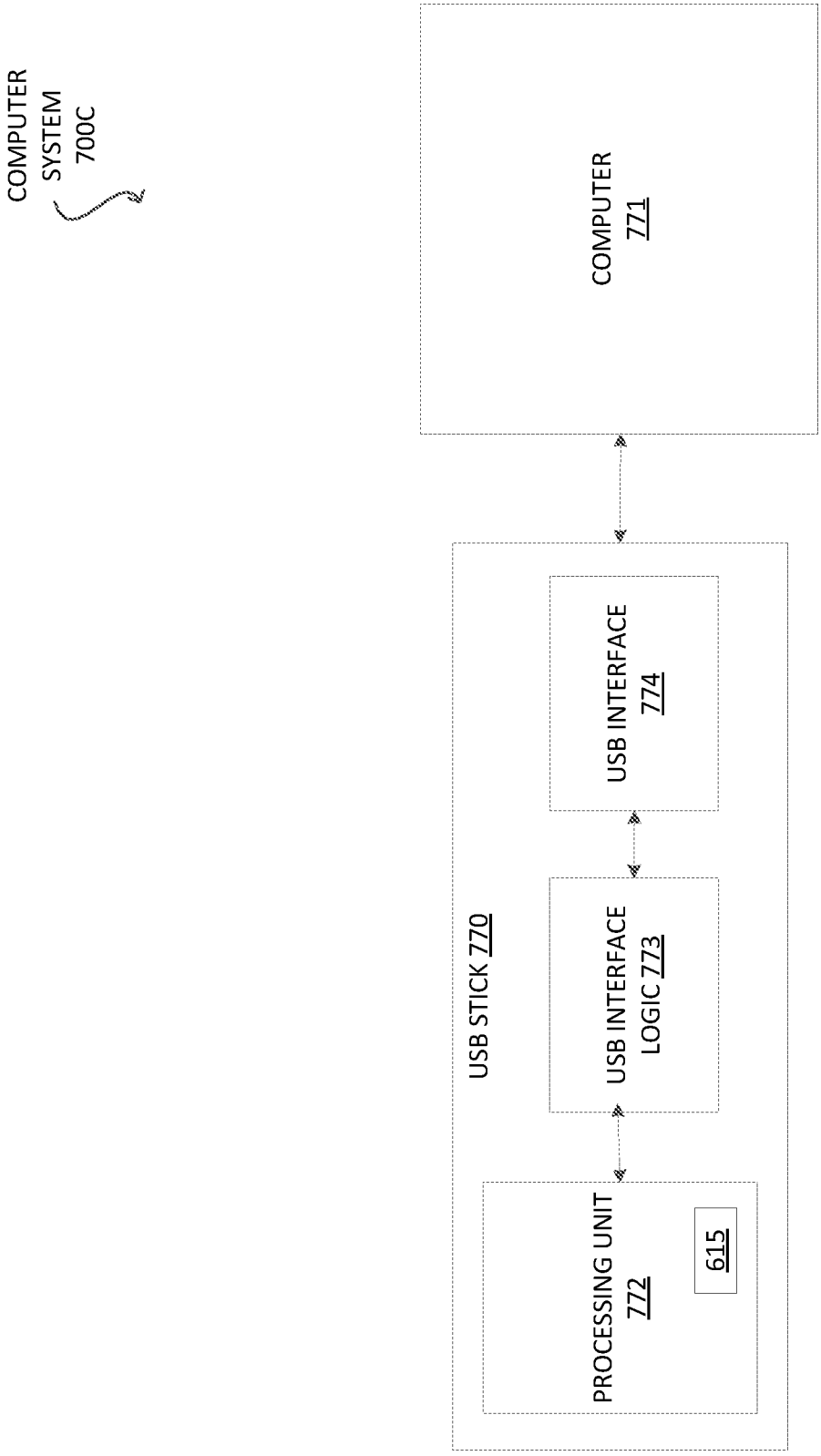
COMPUTER SYSTEM 700C
COMPUTER 771
USB STICK 770
USB INTERFACE 774
USB INTERFACE LOGIC 773
PROCESSING UNIT 772
615

FIG. 7C

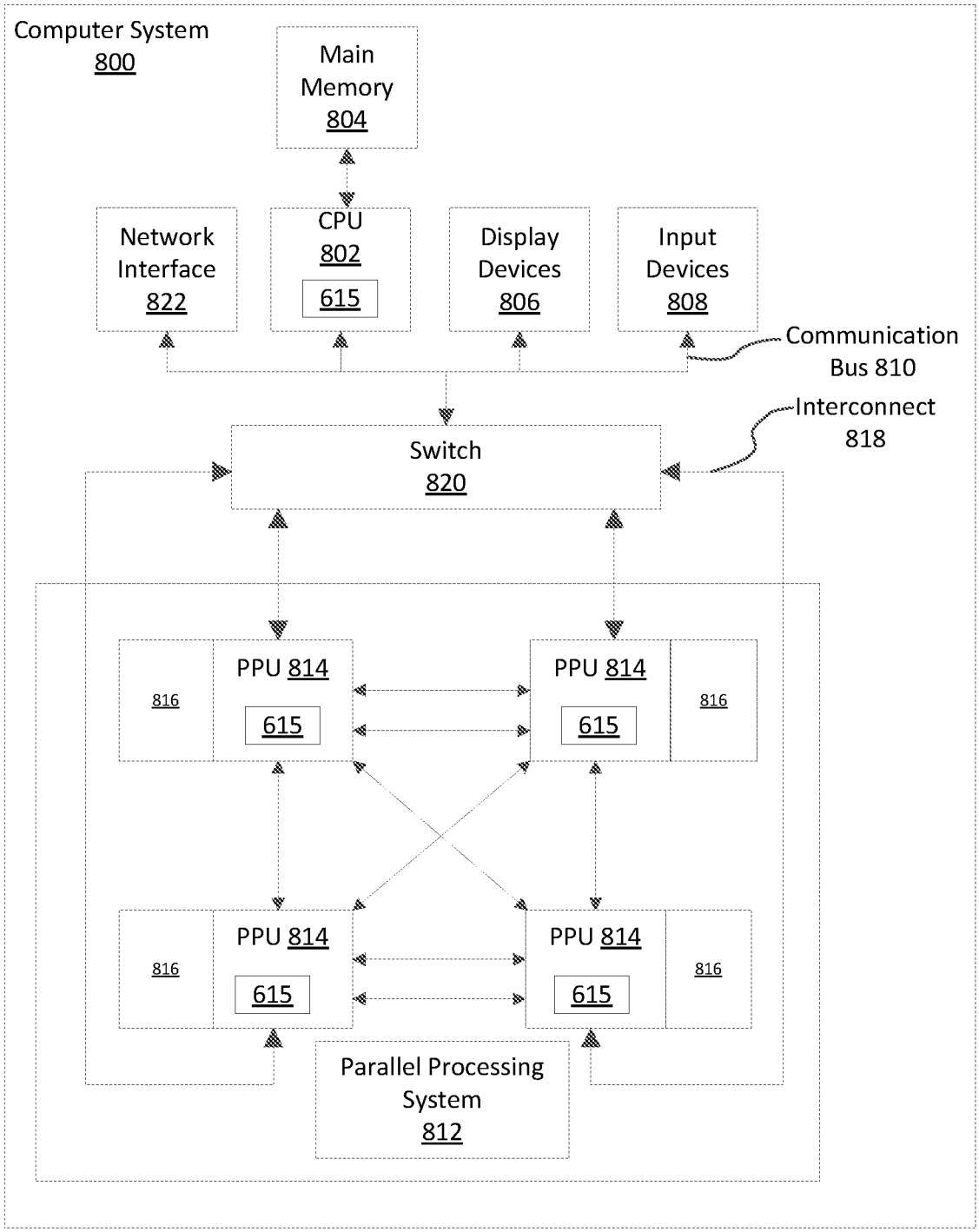
Computer System
800
Main
Memory
804
Network
Interface
822
CPU
802
615
Display
Devices
806
Input
Devices
808
Communication
Bus 810
Interconnect
818
Switch
820
816
PPU 814
615
PPU 814
615
816
816
PPU 814
615
PPU 814
615
816
Parallel Processing
System
812

FIG. 8

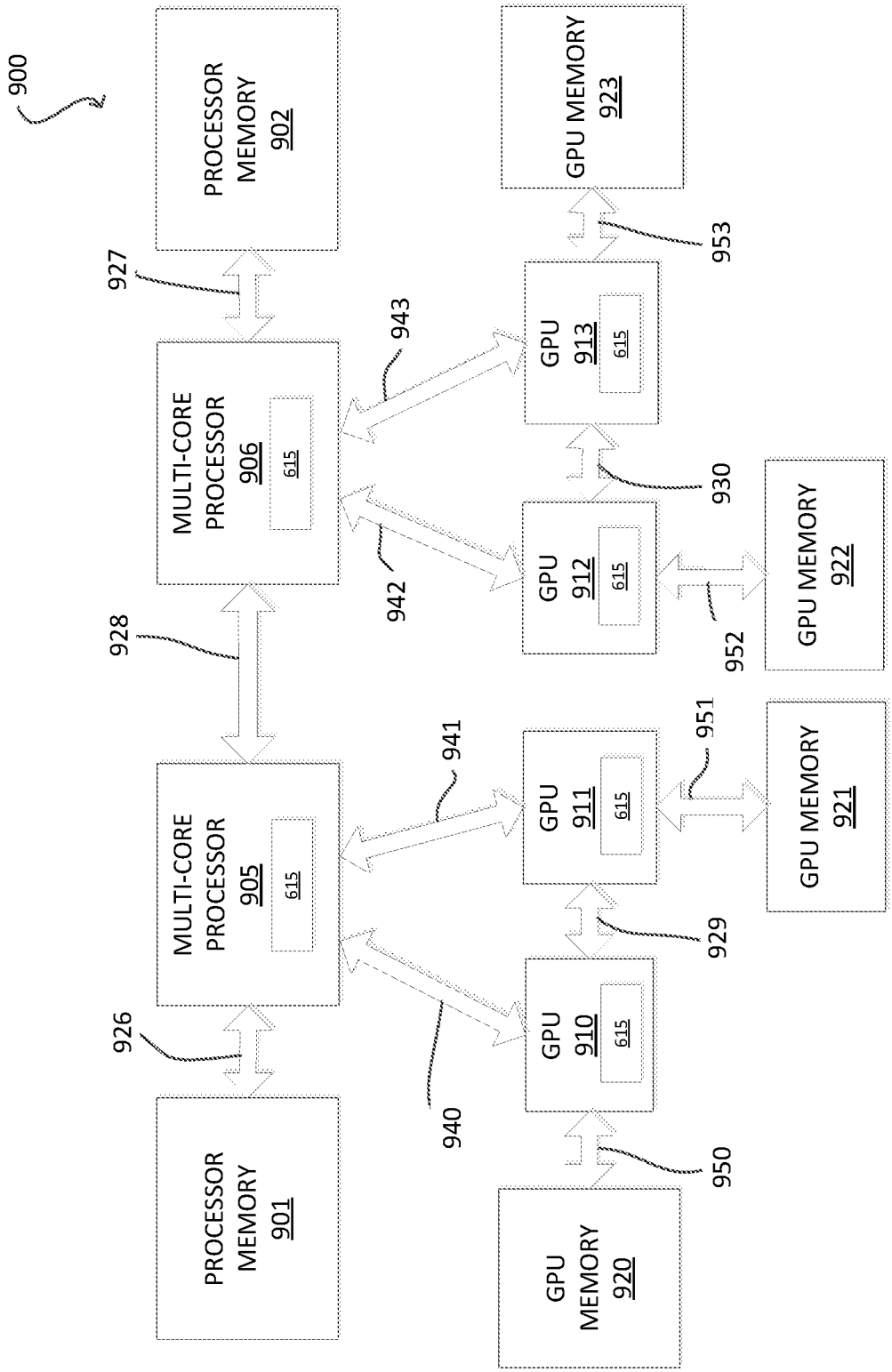
900
PROCESSOR MEMORY 902
GPU MEMORY 923
927
953
943
MULTI-CORE PROCESSOR 906
615
GPU 913
615
930
928
942
GPU 912
615
GPU MEMORY 922
952
941
951
MULTI-CORE PROCESSOR 905
615
GPU 911
615
GPU MEMORY 921
926
929
940
GPU 910
615
PROCESSOR MEMORY 901
950
GPU MEMORY 920

FIG. 9A

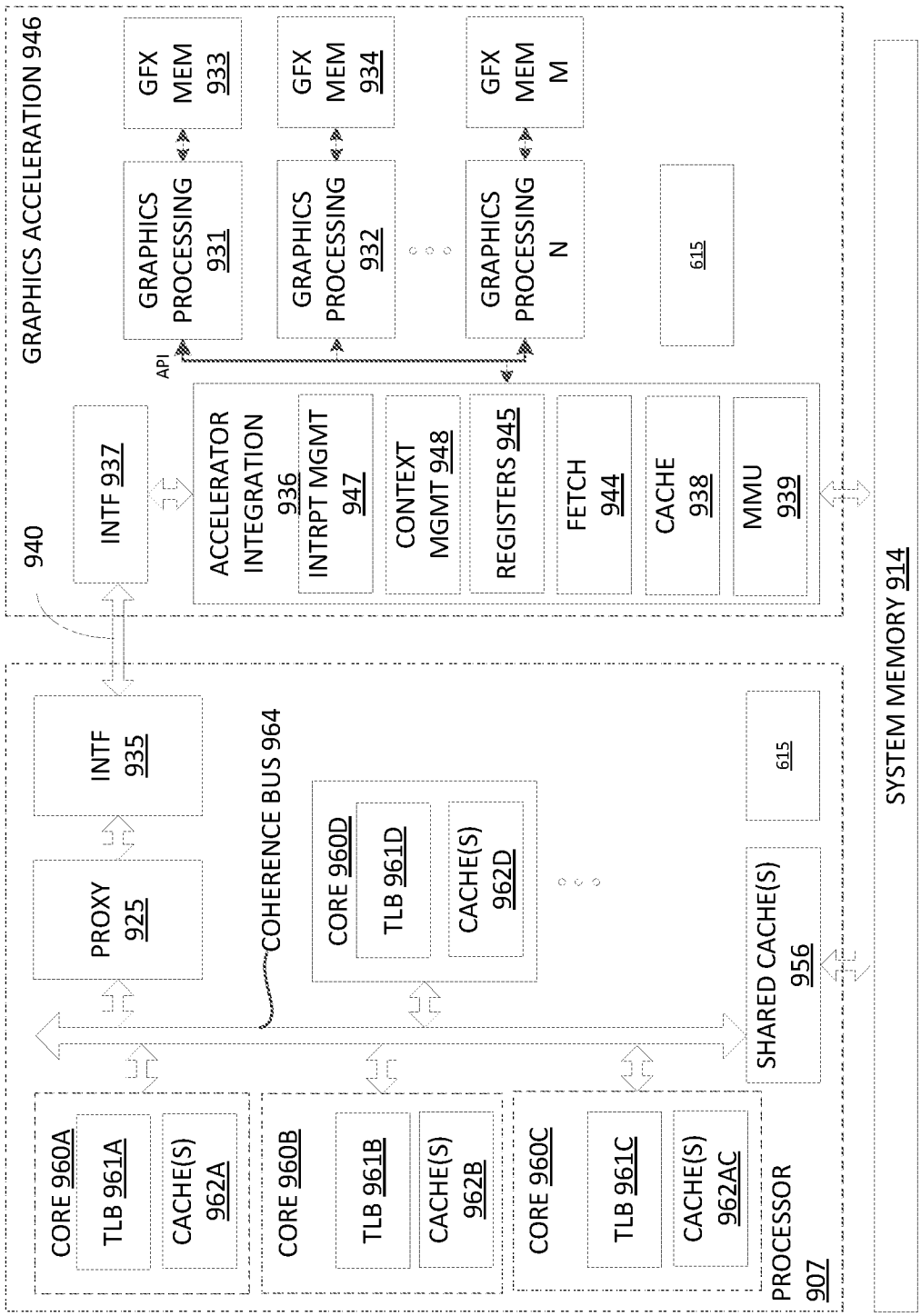
GRAPHICS ACCELERATION 946
940
INTF 937
ACCELERATOR INTEGRATION 936
INTRPT MGMT 947
CONTEXT MGMT 948
REGISTERS 945
FETCH 944
CACHE 938
MMU 939
API
GRAPHICS PROCESSING 931
GRAPHICS PROCESSING 932
GRAPHICS PROCESSING N
GFX MEM 933
GFX MEM 934
GFX MEM M
615
INTF 935
PROXY 925
COHERENCE BUS 964
CORE 960D
TLB 961D
CACHE(S) 962D
SHARED CACHE(S) 956
615
CORE 960A
TLB 961A
CACHE(S) 962A
CORE 960B
TLB 961B
CACHE(S) 962B
CORE 960C
TLB 961C
CACHE(S) 962AC
PROCESSOR 907
SYSTEM MEMORY 914

FIG. 9B

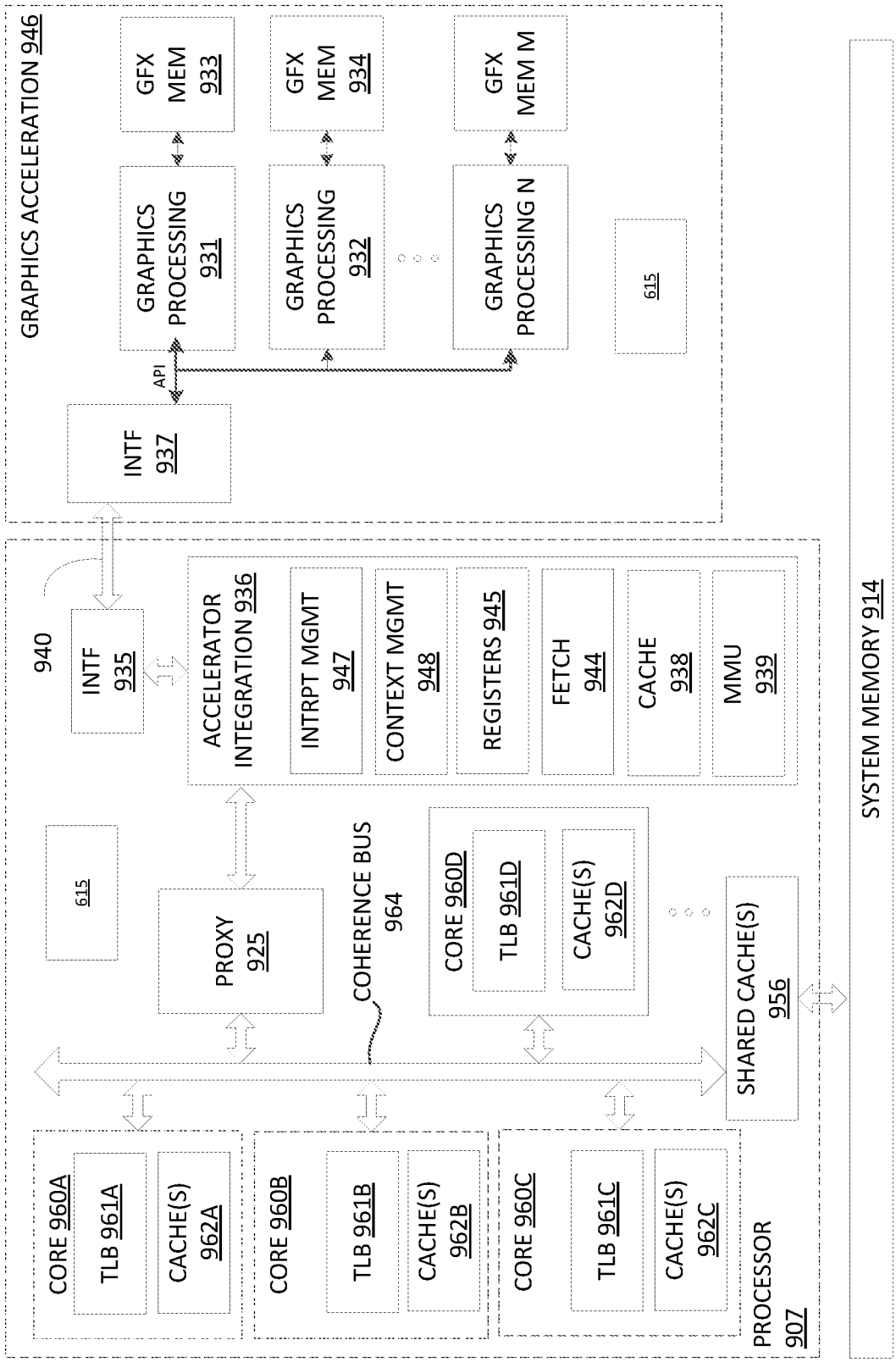
GRAPHICS ACCELERATION 946
GFX MEM 933
GFX MEM 934
GFX MEM M
GRAPHICS PROCESSING 931
GRAPHICS PROCESSING 932
GRAPHICS PROCESSING N
615
API
INTF 937
940
INTF 935
ACCELERATOR INTEGRATION 936
INTRPT MGMT 947
CONTEXT MGMT 948
REGISTERS 945
FETCH 944
CACHE 938
MMU 939
615
PROXY 925
COHERENCE BUS 964
CORE 960D
TLB 961D
CACHE(S) 962D
SHARED CACHE(S) 956
SYSTEM MEMORY 914
CORE 960A
TLB 961A
CACHE(S) 962A
CORE 960B
TLB 961B
CACHE(S) 962B
CORE 960C
TLB 961C
CACHE(S) 962C
PROCESSOR 907

FIG. 9C

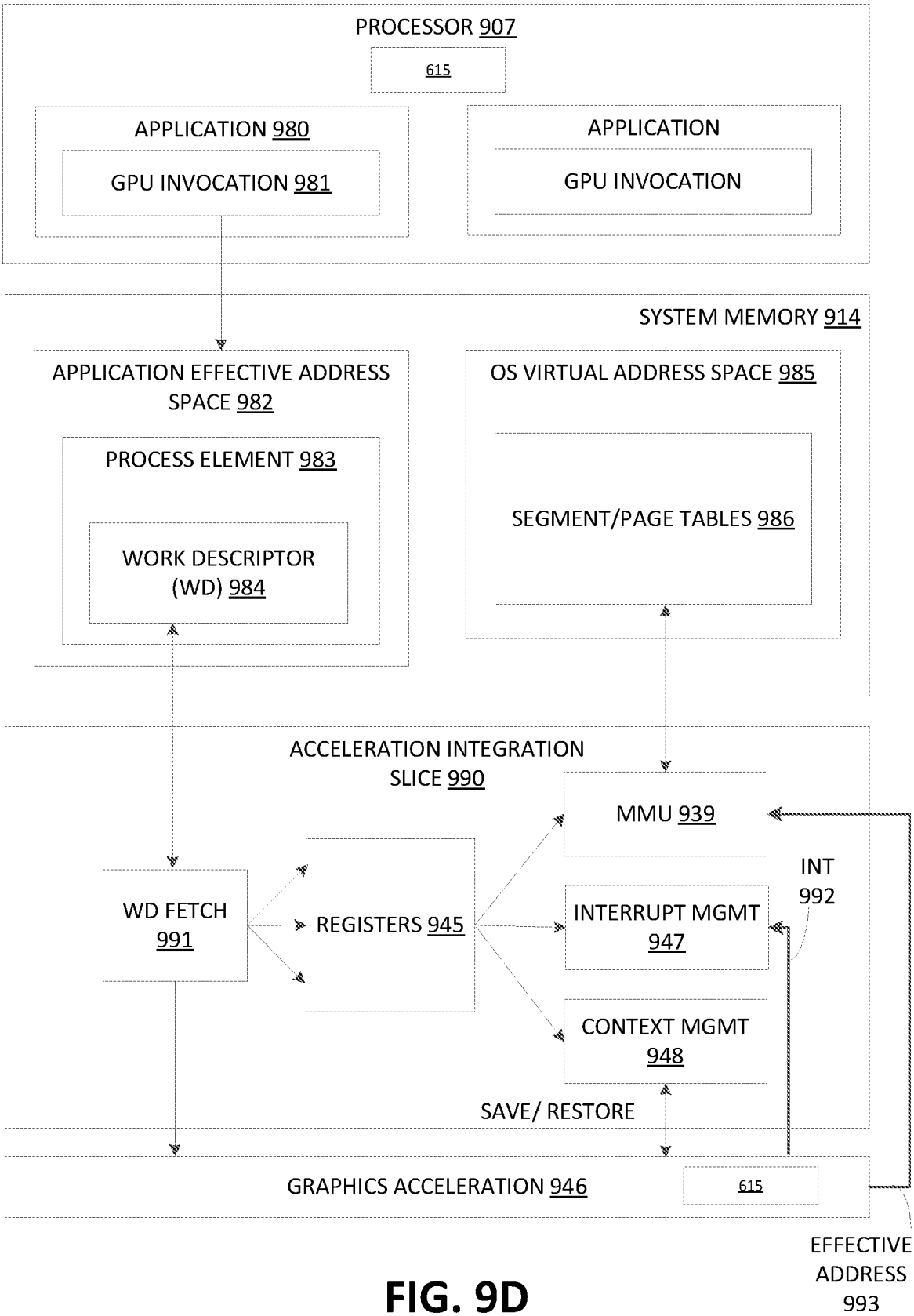
PROCESSOR 907
615
APPLICATION 980
GPU INVOCATION 981
APPLICATION
GPU INVOCATION
SYSTEM MEMORY 914
APPLICATION EFFECTIVE ADDRESS SPACE 982
PROCESS ELEMENT 983
WORK DESCRIPTOR (WD) 984
OS VIRTUAL ADDRESS SPACE 985
SEGMENT/PAGE TABLES 986
ACCELERATION INTEGRATION SLICE 990
MMU 939
INT 992
WD FETCH 991
REGISTERS 945
INTERRUPT MGMT 947
CONTEXT MGMT 948
SAVE/ RESTORE
GRAPHICS ACCELERATION 946
615
EFFECTIVE ADDRESS 993

FIG. 9D

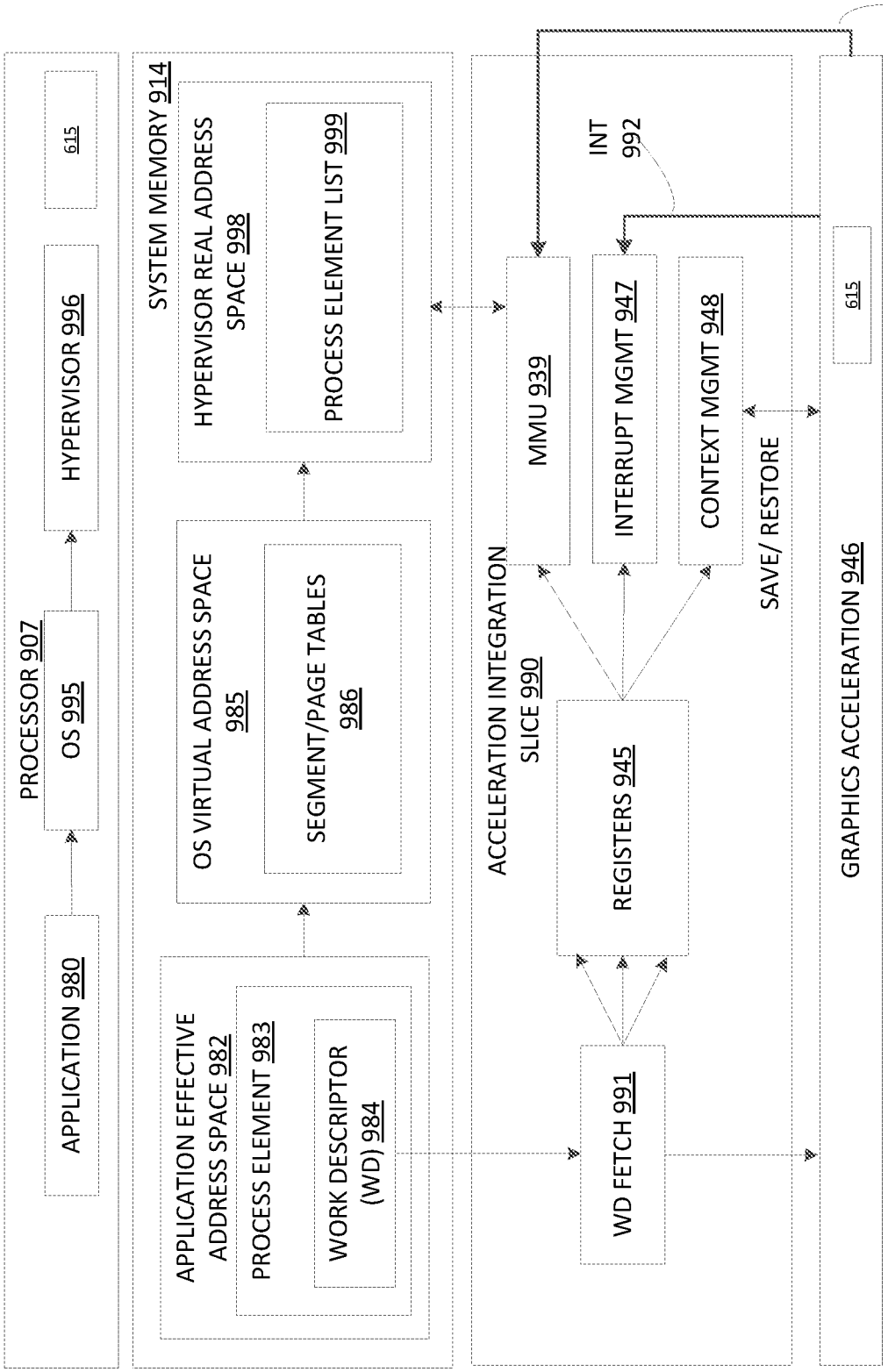
PROCESSOR 907
APPLICATION 980
OS 995
HYPERVISOR 996
615
SYSTEM MEMORY 914
APPLICATION EFFECTIVE ADDRESS SPACE 982
PROCESS ELEMENT 983
WORK DESCRIPTOR (WD) 984
OS VIRTUAL ADDRESS SPACE 985
SEGMENT/PAGE TABLES 986
HYPERVISOR REAL ADDRESS SPACE 998
PROCESS ELEMENT LIST 999
ACCELERATION INTEGRATION SLICE 990
WD FETCH 991
REGISTERS 945
MMU 939
INTERRUPT MGMT 947
CONTEXT MGMT 948
SAVE/ RESTORE
INT 992
EFFECTIVE ADDRESS 993
GRAPHICS ACCELERATION 946
615

FIG. 9E

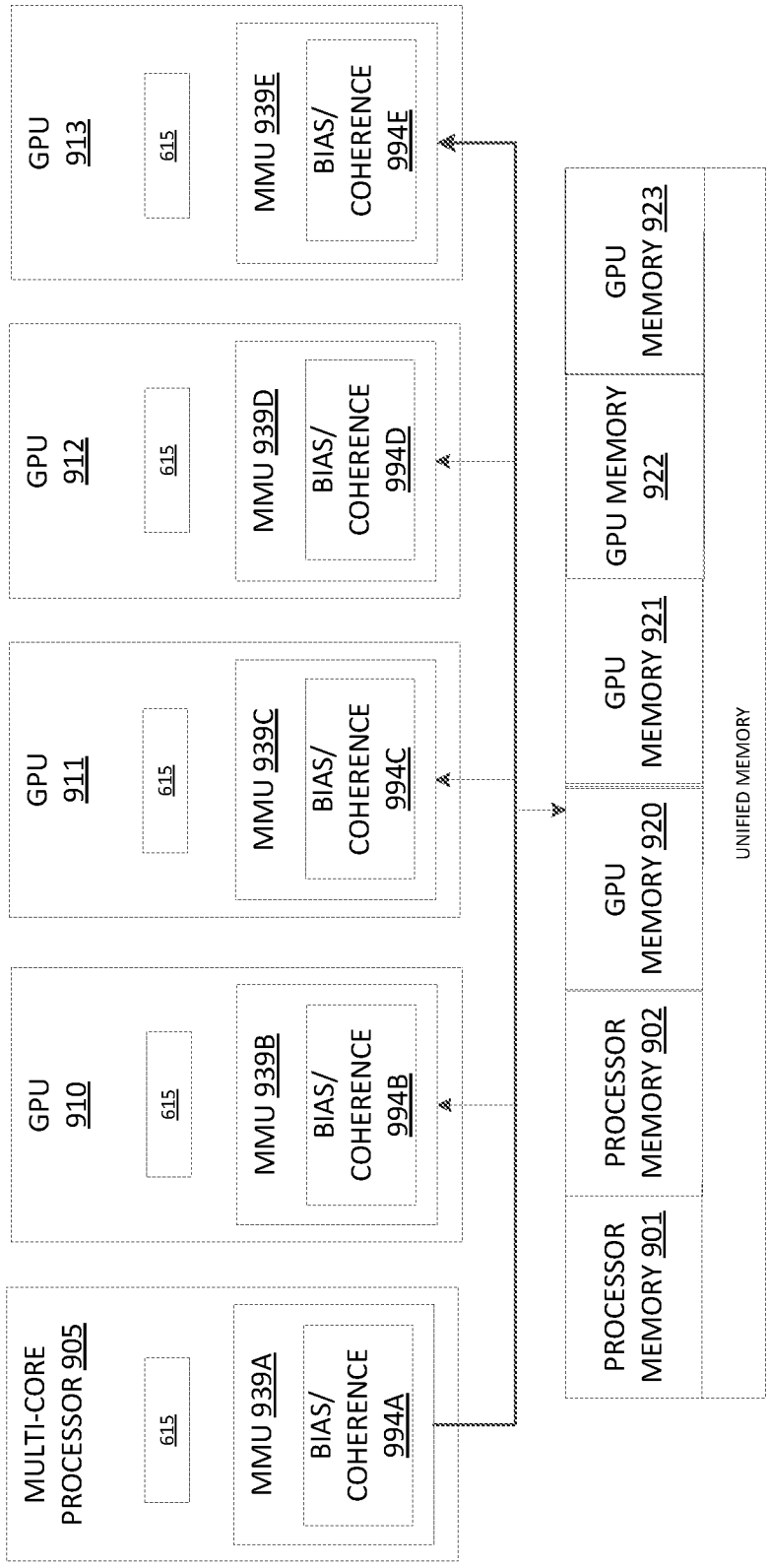
MULTI-CORE PROCESSOR 905
615
MMU 939A
BIAS/ COHERENCE 994A
GPU 910
615
MMU 939B
BIAS/ COHERENCE 994B
GPU 911
615
MMU 939C
BIAS/ COHERENCE 994C
GPU 912
615
MMU 939D
BIAS/ COHERENCE 994D
GPU 913
615
MMU 939E
BIAS/ COHERENCE 994E
PROCESSOR MEMORY 901
PROCESSOR MEMORY 902
GPU MEMORY 920
GPU MEMORY 921
GPU MEMORY 922
GPU MEMORY 923
UNIFIED MEMORY

FIG. 9F

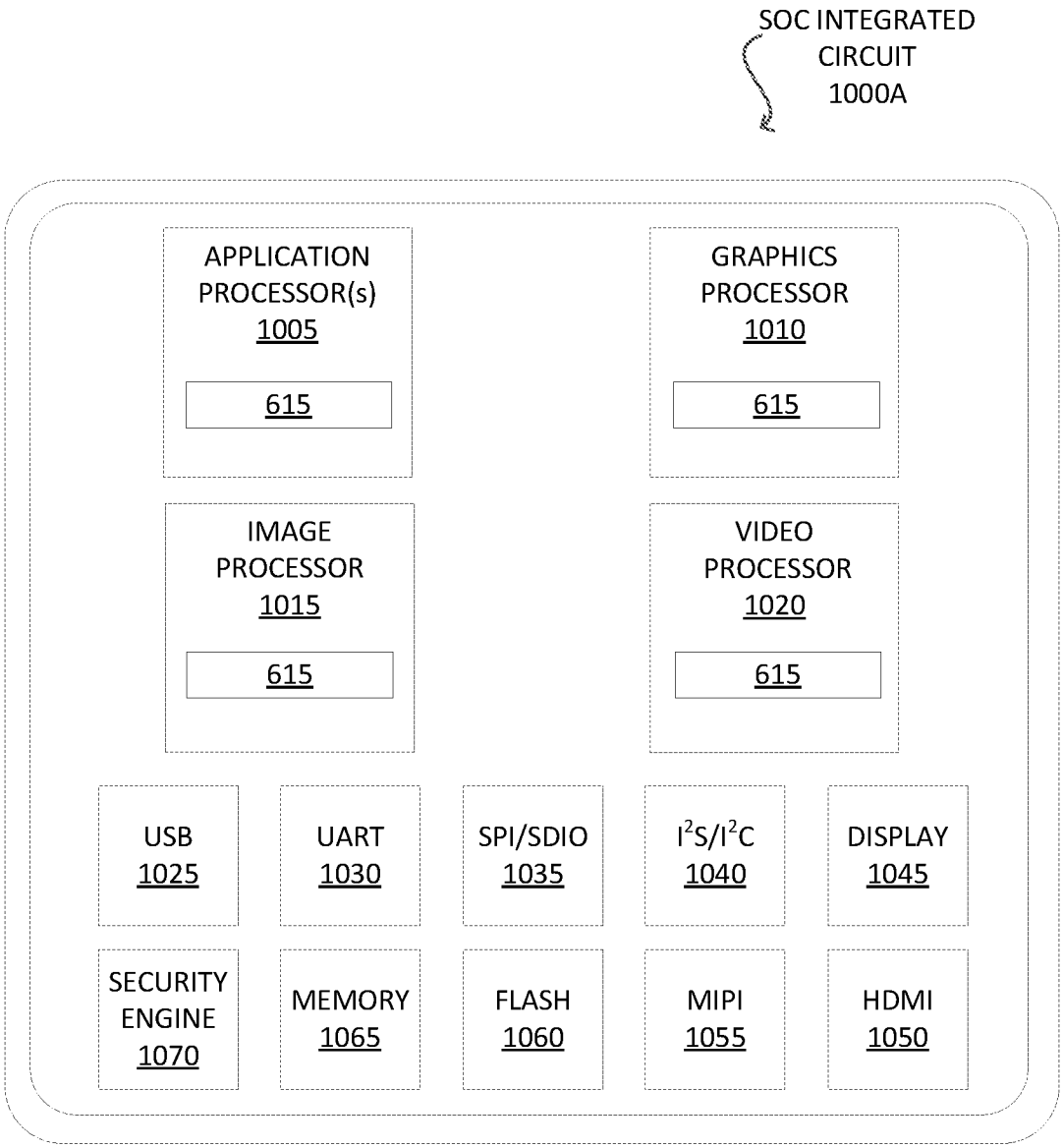
SOC INTEGRATED CIRCUIT 1000A
APPLICATION PROCESSOR(s) 1005
615
GRAPHICS PROCESSOR 1010
615
IMAGE PROCESSOR 1015
615
VIDEO PROCESSOR 1020
615
USB 1025
UART 1030
SPI/SDIO 1035
I2S/I2C 1040
DISPLAY 1045
SECURITY ENGINE 1070
MEMORY 1065
FLASH 1060
MIPI 1055
HDMI 1050

FIG. 10A

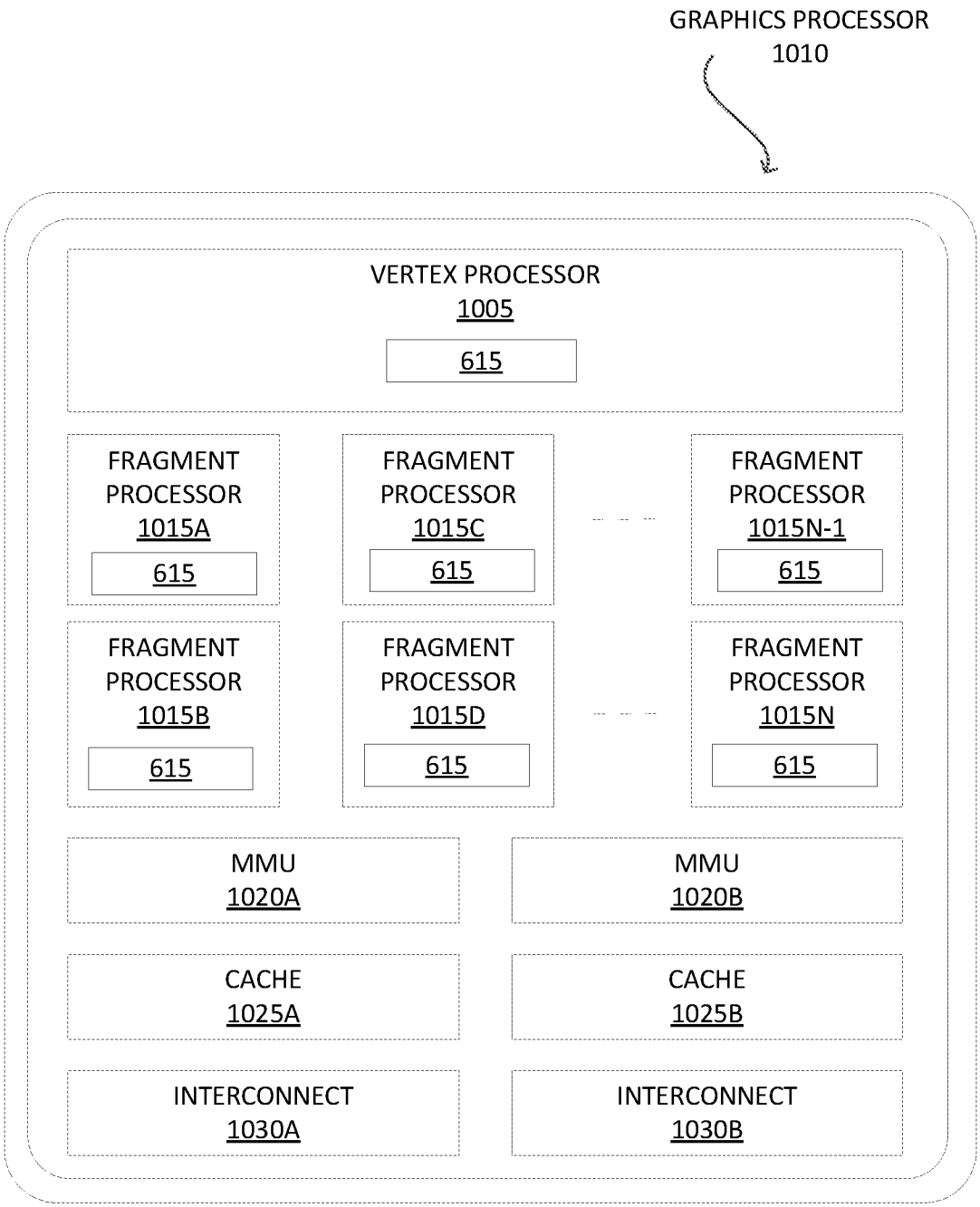
GRAPHICS PROCESSOR
1010
VERTEX PROCESSOR
1005
615
FRAGMENT PROCESSOR
1015A
615
FRAGMENT PROCESSOR
1015C
615
FRAGMENT PROCESSOR
1015N-1
615
FRAGMENT PROCESSOR
1015B
615
FRAGMENT PROCESSOR
1015D
615
FRAGMENT PROCESSOR
1015N
615
MMU
1020A
MMU
1020B
CACHE
1025A
CACHE
1025B
INTERCONNECT
1030A
INTERCONNECT
1030B

FIG. 10B

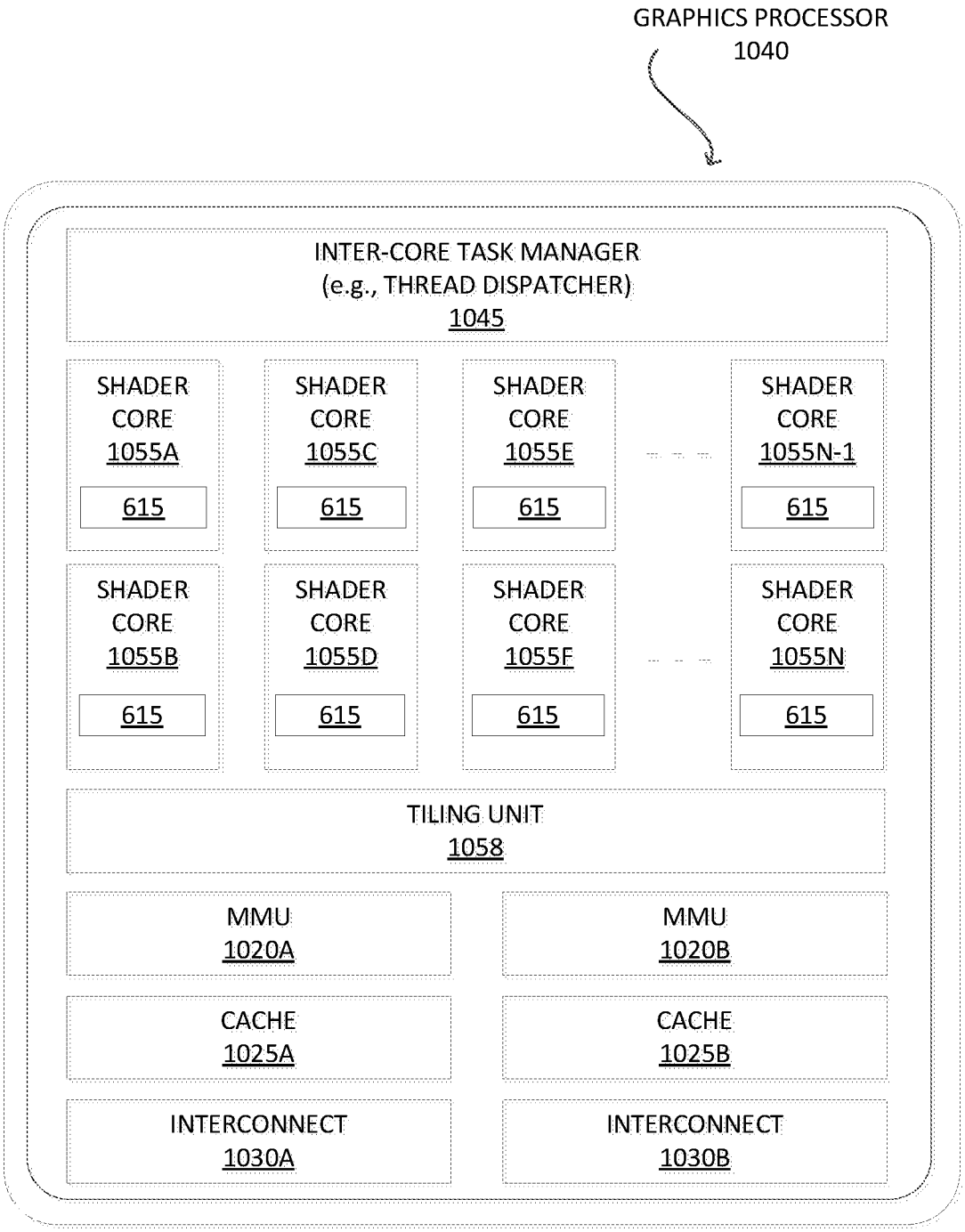
GRAPHICS PROCESSOR
1040
INTER-CORE TASK MANAGER
(e.g., THREAD DISPATCHER)
1045
SHADER CORE 1055A
615
SHADER CORE 1055C
615
SHADER CORE 1055E
615
SHADER CORE 1055N-1
615
SHADER CORE 1055B
615
SHADER CORE 1055D
615
SHADER CORE 1055F
615
SHADER CORE 1055N
615
TILING UNIT
1058
MMU
1020A
MMU
1020B
CACHE
1025A
CACHE
1025B
INTERCONNECT
1030A
INTERCONNECT
1030B

FIG. 10C

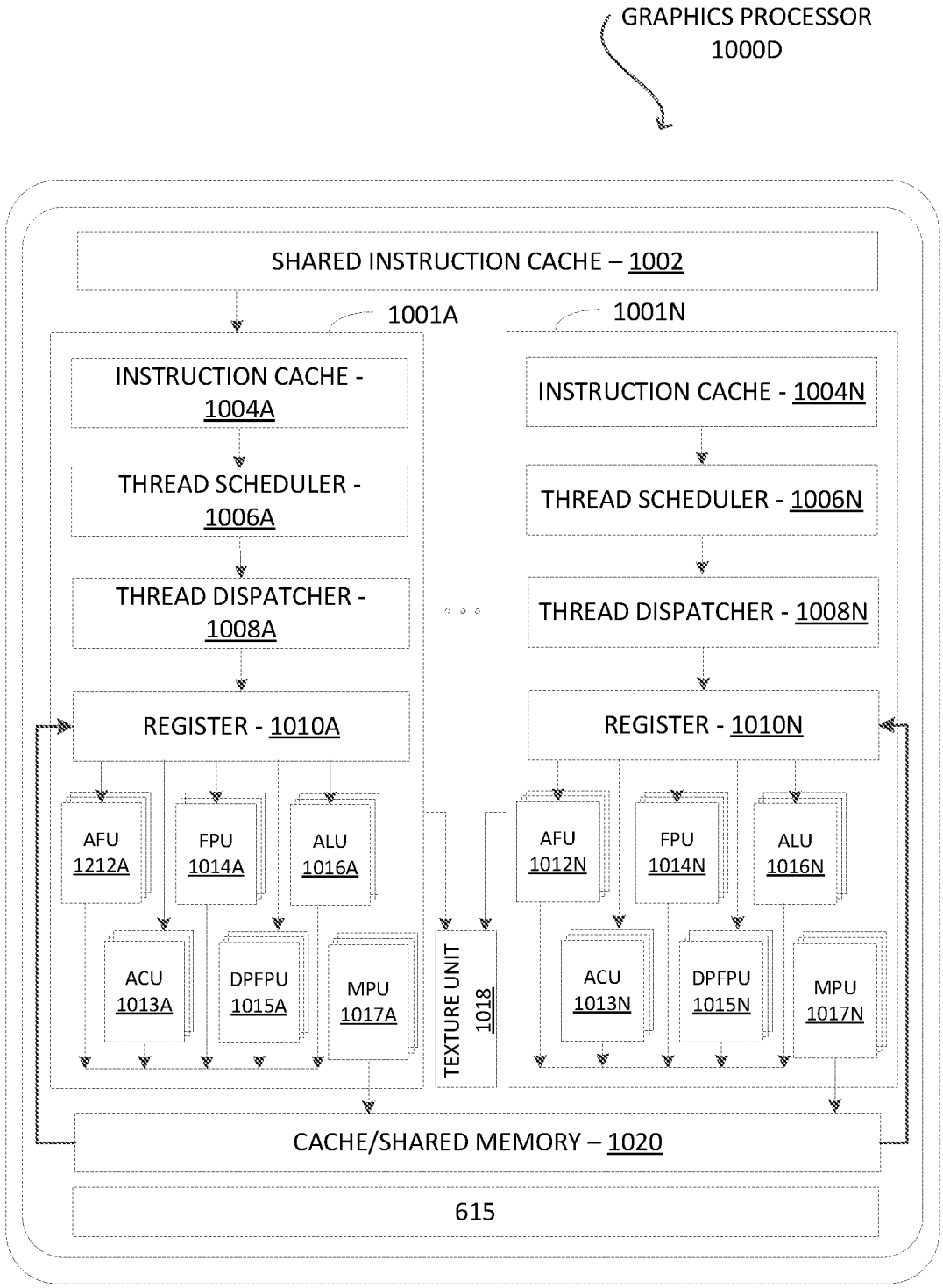
GRAPHICS PROCESSOR
1000D
SHARED INSTRUCTION CACHE – 1002
1001A
1001N
INSTRUCTION CACHE - 1004A
INSTRUCTION CACHE - 1004N
THREAD SCHEDULER - 1006A
THREAD SCHEDULER - 1006N
THREAD DISPATCHER - 1008A
THREAD DISPATCHER - 1008N
REGISTER - 1010A
REGISTER - 1010N
AFU 1212A
FPU 1014A
ALU 1016A
AFU 1012N
FPU 1014N
ALU 1016N
ACU 1013A
DPFPU 1015A
MPU 1017A
TEXTURE UNIT 1018
ACU 1013N
DPFPU 1015N
MPU 1017N
CACHE/SHARED MEMORY – 1020
615

FIG. 10D

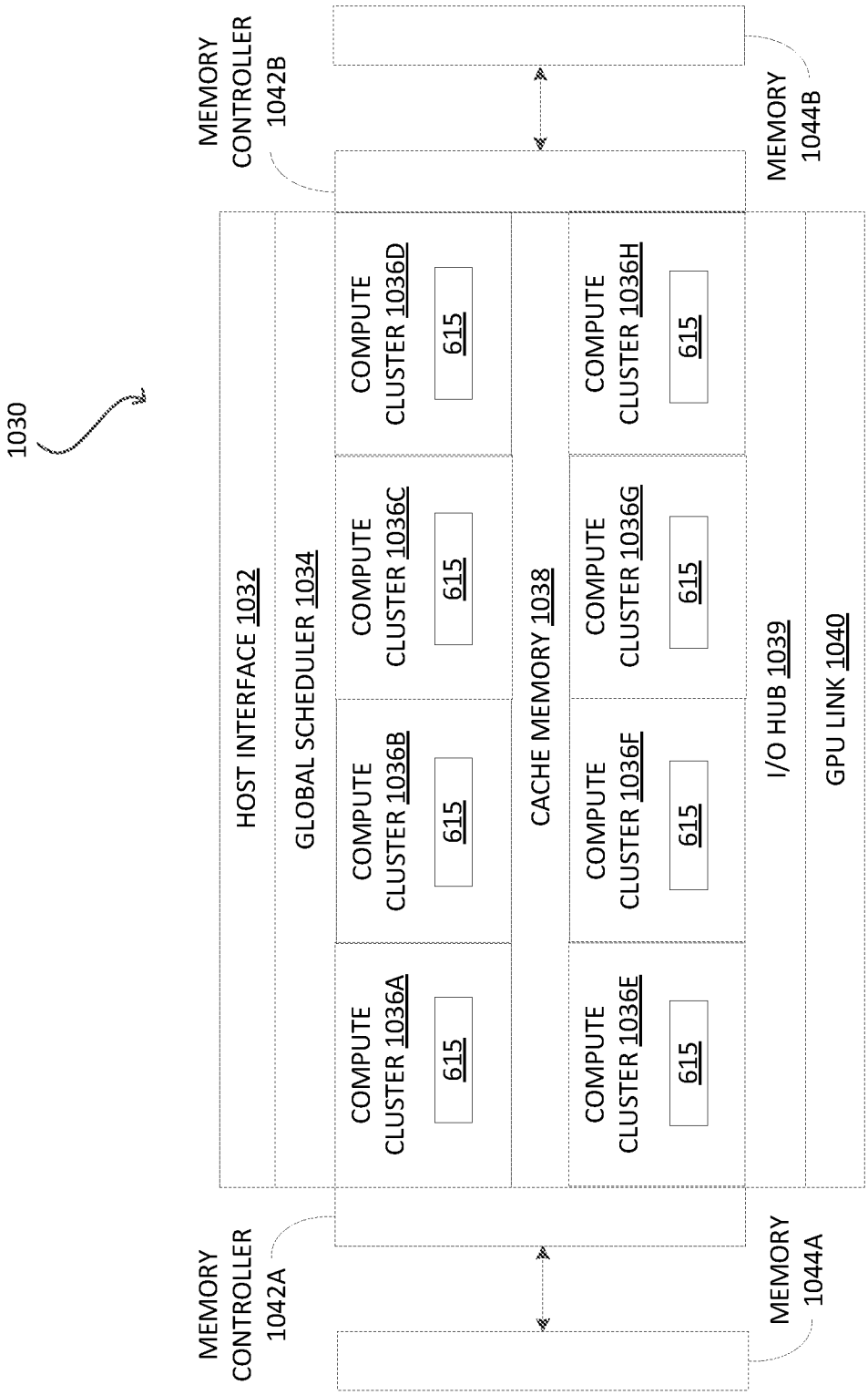
1030
MEMORY CONTROLLER 1042B
MEMORY 1044B
HOST INTERFACE 1032
GLOBAL SCHEDULER 1034
COMPUTE CLUSTER 1036A
615
COMPUTE CLUSTER 1036B
615
COMPUTE CLUSTER 1036C
615
COMPUTE CLUSTER 1036D
615
CACHE MEMORY 1038
COMPUTE CLUSTER 1036E
615
COMPUTE CLUSTER 1036F
615
COMPUTE CLUSTER 1036G
615
COMPUTE CLUSTER 1036H
615
I/O HUB 1039
GPU LINK 1040
MEMORY CONTROLLER 1042A
MEMORY 1044A

FIG. 10E

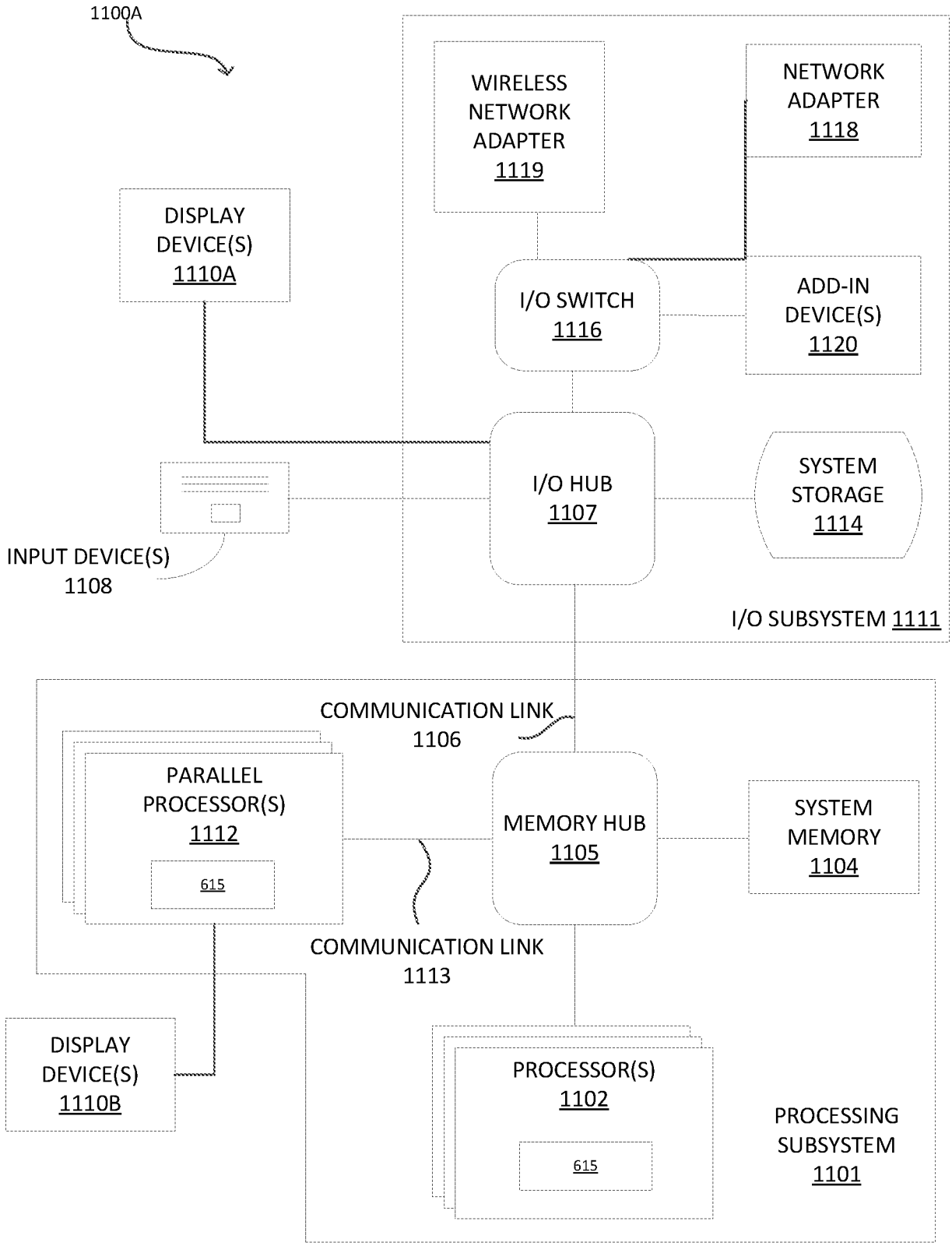
1100A
WIRELESS NETWORK ADAPTER 1119
NETWORK ADAPTER 1118
DISPLAY DEVICE(S) 1110A
I/O SWITCH 1116
ADD-IN DEVICE(S) 1120
I/O HUB 1107
SYSTEM STORAGE 1114
INPUT DEVICE(S) 1108
I/O SUBSYSTEM 1111
COMMUNICATION LINK 1106
PARALLEL PROCESSOR(S) 1112
615
MEMORY HUB 1105
SYSTEM MEMORY 1104
COMMUNICATION LINK 1113
DISPLAY DEVICE(S) 1110B
PROCESSOR(S) 1102
615
PROCESSING SUBSYSTEM 1101

FIG. 11A

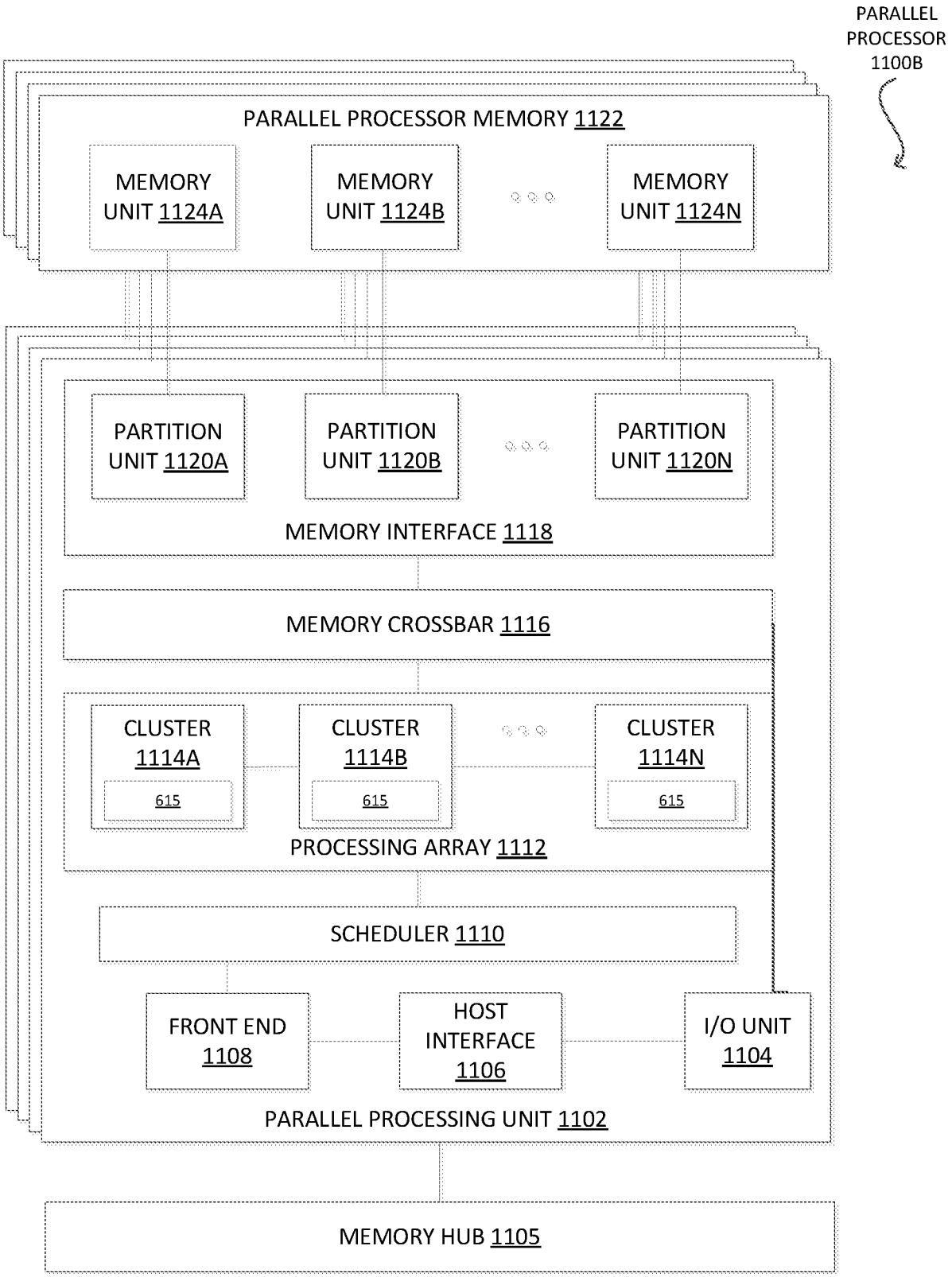
PARALLEL PROCESSOR 1100B
PARALLEL PROCESSOR MEMORY 1122
MEMORY UNIT 1124A
MEMORY UNIT 1124B
MEMORY UNIT 1124N
PARTITION UNIT 1120A
PARTITION UNIT 1120B
PARTITION UNIT 1120N
MEMORY INTERFACE 1118
MEMORY CROSSBAR 1116
CLUSTER 1114A
CLUSTER 1114B
CLUSTER 1114N
615
615
615
PROCESSING ARRAY 1112
SCHEDULER 1110
FRONT END 1108
HOST INTERFACE 1106
I/O UNIT 1104
PARALLEL PROCESSING UNIT 1102
MEMORY HUB 1105

FIG. 11B

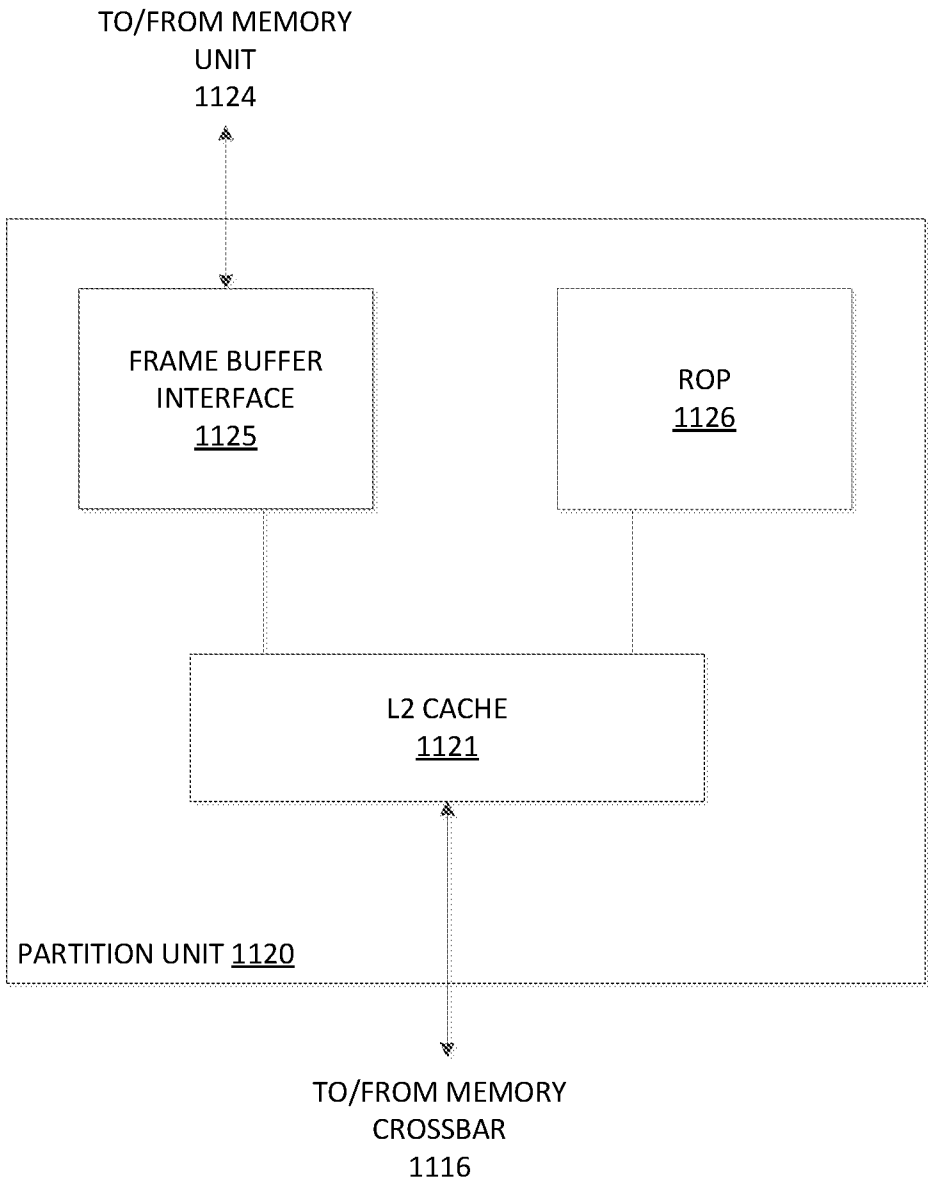
TO/FROM MEMORY
UNIT
1124
FRAME BUFFER
INTERFACE
1125
ROP
1126
L2 CACHE
1121
PARTITION UNIT 1120
TO/FROM MEMORY
CROSSBAR
1116

FIG. 11C

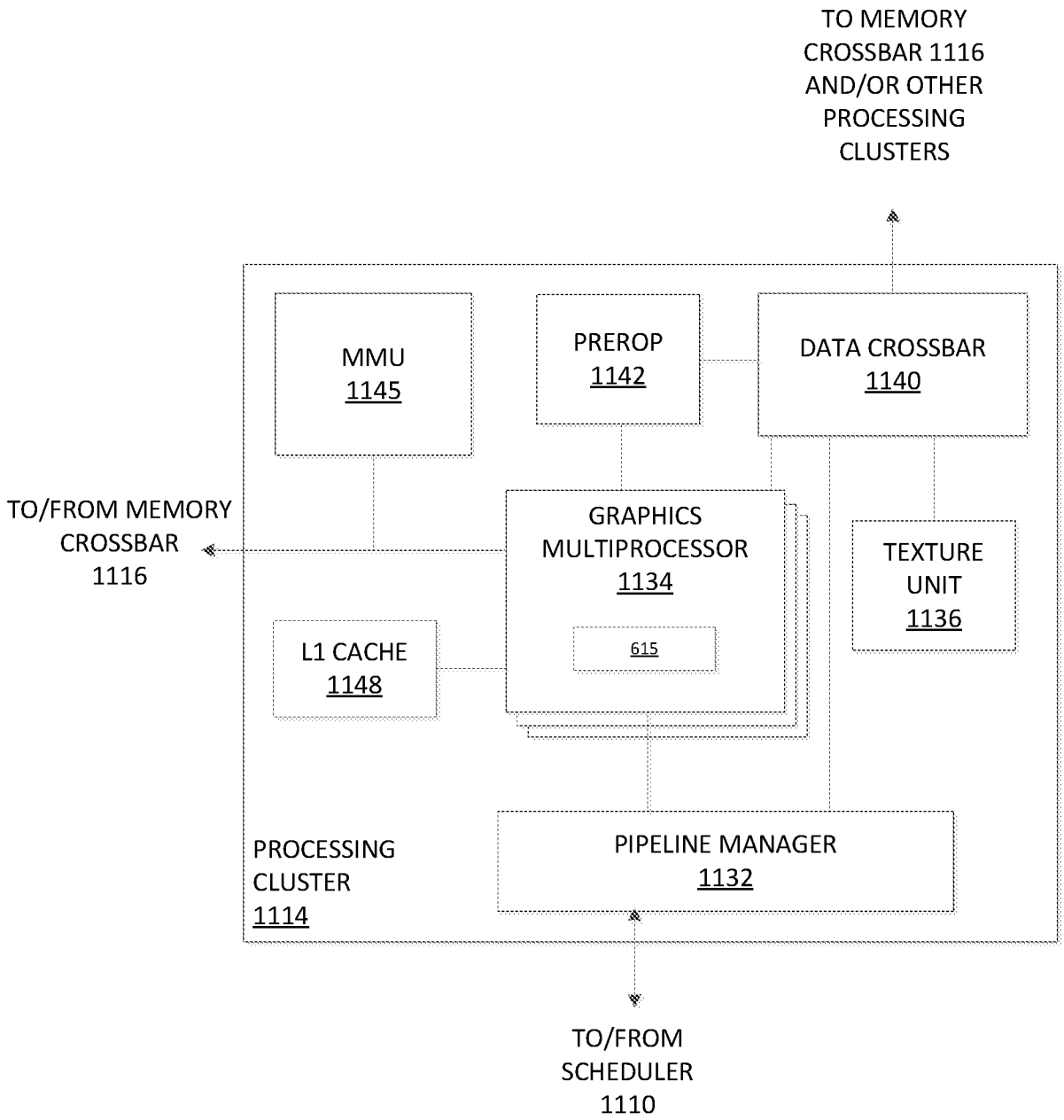
TO MEMORY CROSSBAR 1116 AND/OR OTHER PROCESSING CLUSTERS
MMU 1145
PREROP 1142
DATA CROSSBAR 1140
TO/FROM MEMORY CROSSBAR 1116
GRAPHICS MULTIPROCESSOR 1134
615
TEXTURE UNIT 1136
L1 CACHE 1148
PROCESSING CLUSTER 1114
PIPELINE MANAGER 1132
TO/FROM SCHEDULER 1110

FIG. 11D

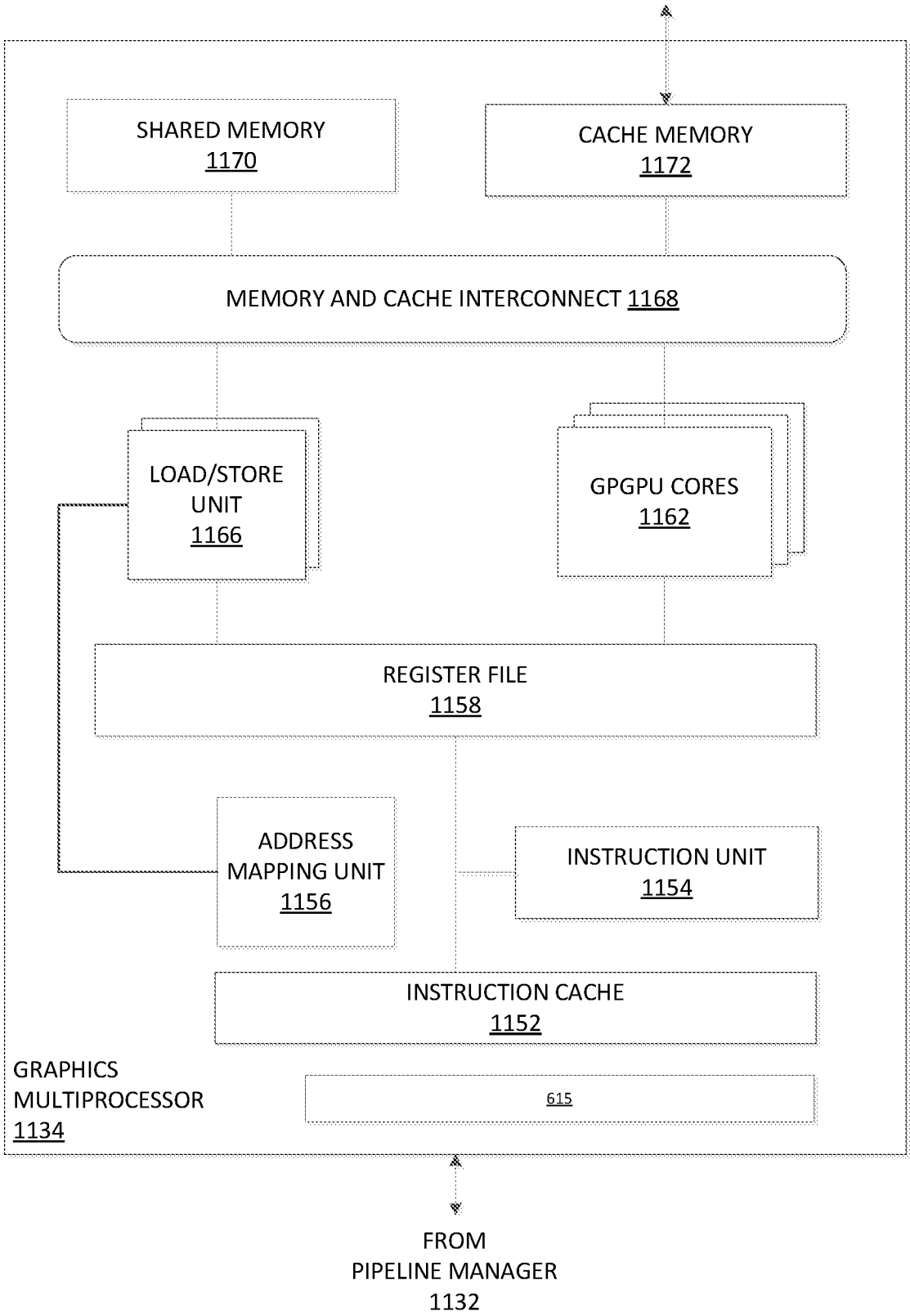
SHARED MEMORY
1170
CACHE MEMORY
1172
MEMORY AND CACHE INTERCONNECT 1168
LOAD/STORE
UNIT
1166
GPGPU CORES
1162
REGISTER FILE
1158
ADDRESS
MAPPING UNIT
1156
INSTRUCTION UNIT
1154
INSTRUCTION CACHE
1152
GRAPHICS
MULTIPROCESSOR
1134
615
FROM
PIPELINE MANAGER
1132

FIG. 11E

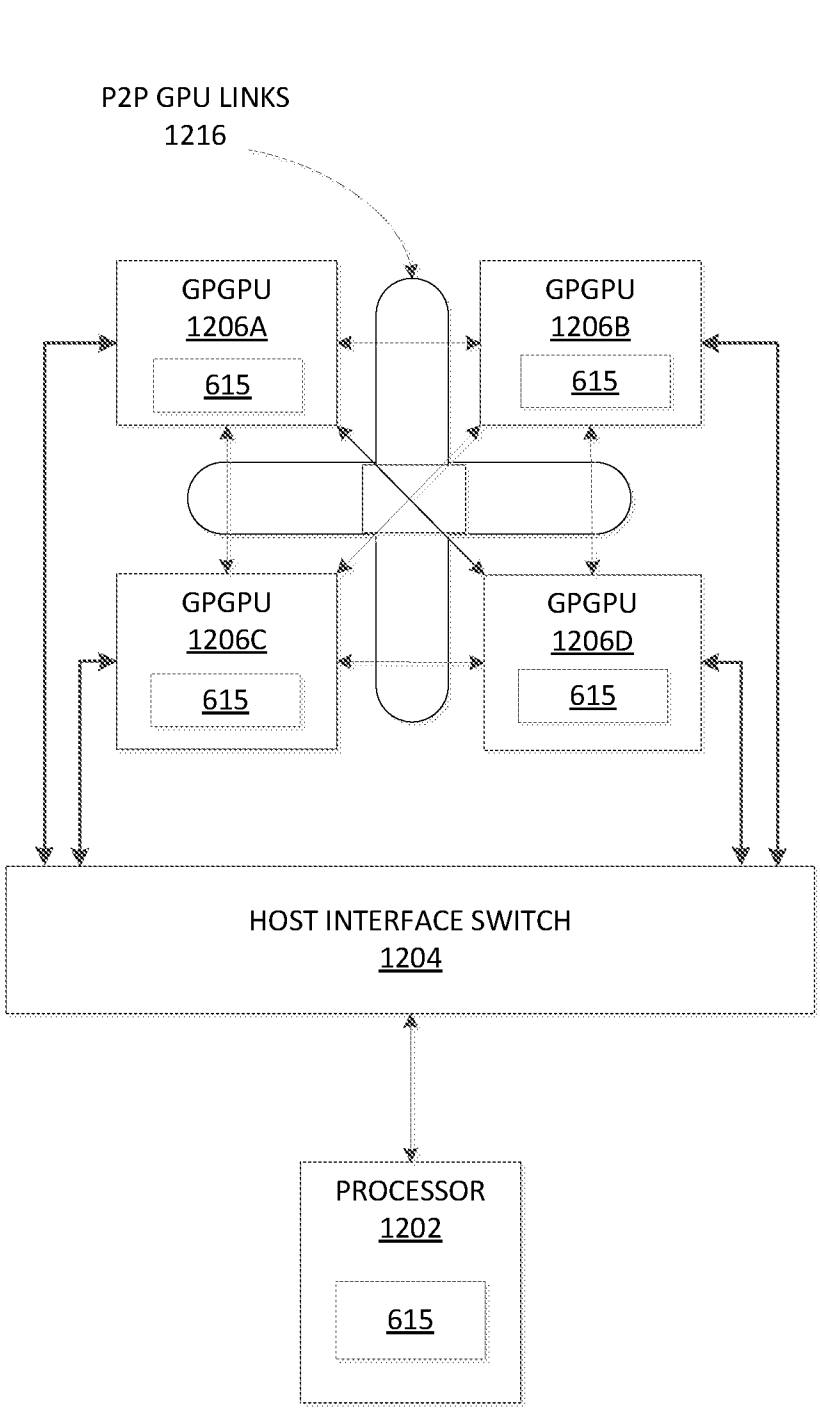
1200A
P2P GPU LINKS
1216
GPGPU
1206A
615
GPGPU
1206B
615
GPGPU
1206C
615
GPGPU
1206D
615
HOST INTERFACE SWITCH
1204
PROCESSOR
1202
615

FIG. 12A

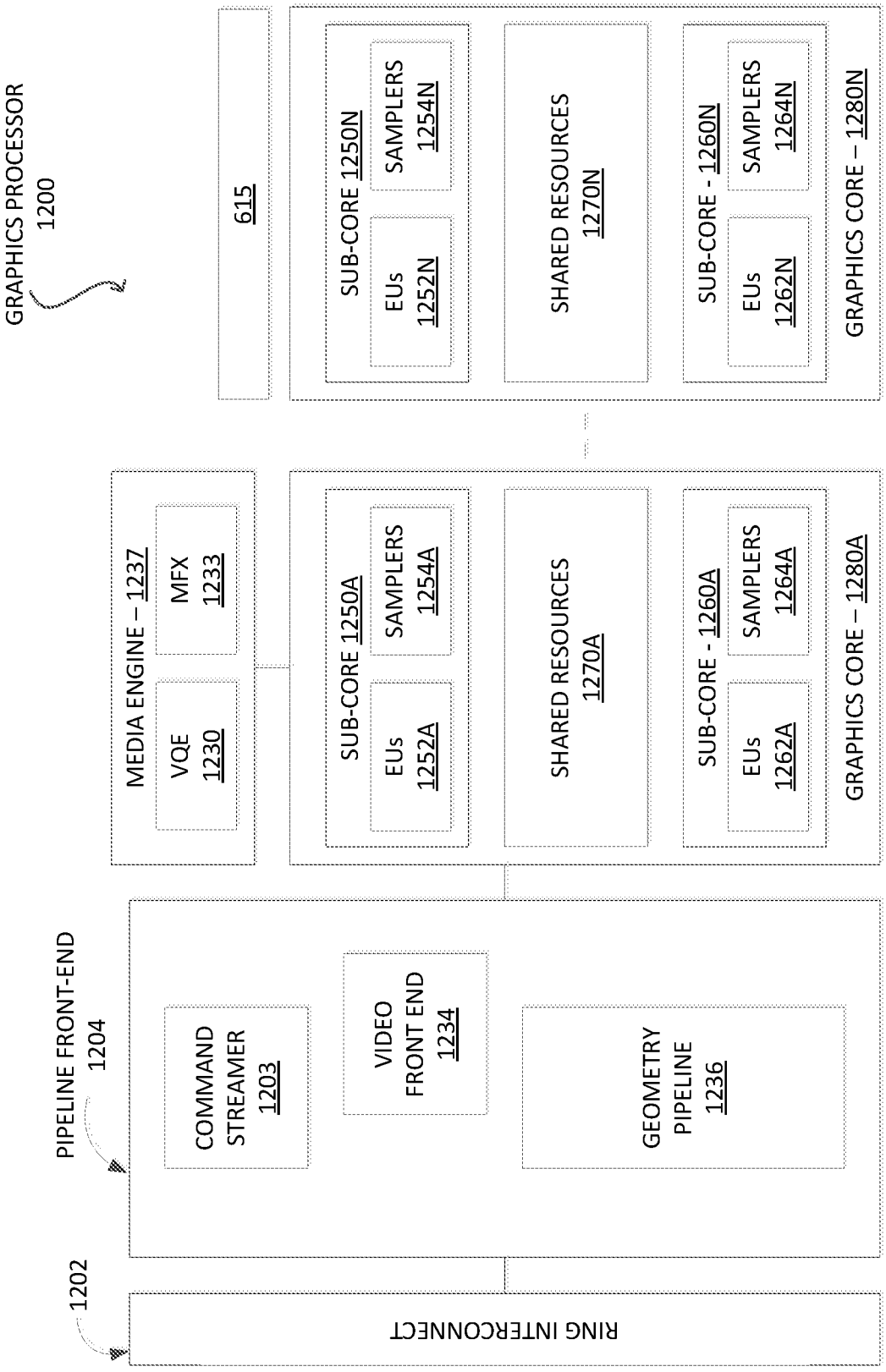
GRAPHICS PROCESSOR
1200
615
SUB-CORE 1250N
EUs
1252N
SAMPLERS
1254N
SHARED RESOURCES
1270N
SUB-CORE - 1260N
EUs
1262N
SAMPLERS
1264N
GRAPHICS CORE – 1280N
MEDIA ENGINE – 1237
VQE
1230
MFX
1233
SUB-CORE 1250A
EUs
1252A
SAMPLERS
1254A
SHARED RESOURCES
1270A
SUB-CORE - 1260A
EUs
1262A
SAMPLERS
1264A
GRAPHICS CORE – 1280A
PIPELINE FRONT-END
1204
COMMAND
STREAMER
1203
VIDEO
FRONT END
1234
GEOMETRY
PIPELINE
1236
1202
RING INTERCONNECT

FIG. 12B

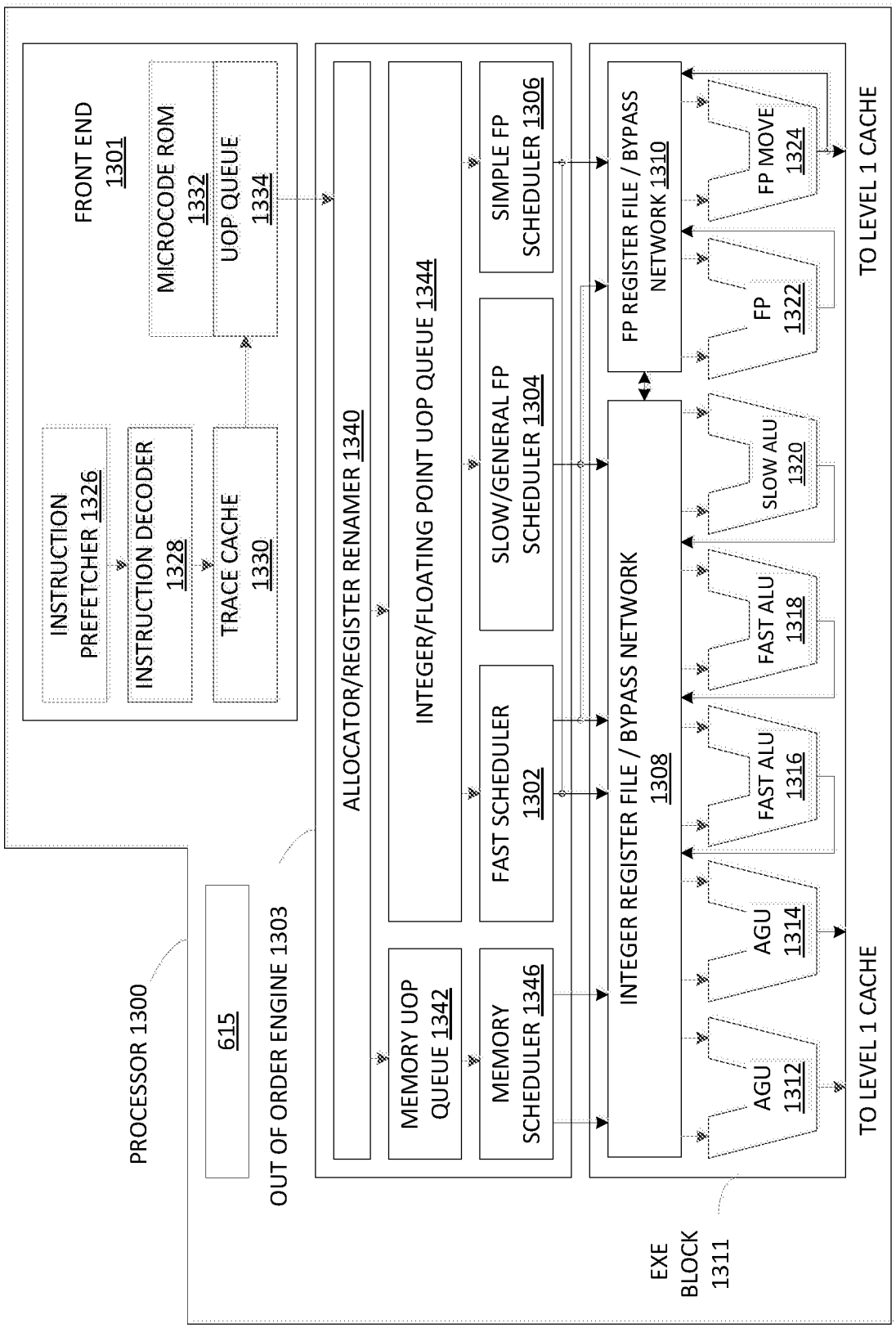
PROCESSOR 1300
615
OUT OF ORDER ENGINE 1303
FRONT END 1301
INSTRUCTION PREFETCHER 1326
INSTRUCTION DECODER 1328
TRACE CACHE 1330
MICROCODE ROM 1332
UOP QUEUE 1334
ALLOCATOR/REGISTER RENAMER 1340
MEMORY UOP QUEUE 1342
MEMORY SCHEDULER 1346
INTEGER/FLOATING POINT UOP QUEUE 1344
FAST SCHEDULER 1302
SLOW/GENERAL FP SCHEDULER 1304
SIMPLE FP SCHEDULER 1306
INTEGER REGISTER FILE / BYPASS NETWORK 1308
FP REGISTER FILE / BYPASS NETWORK 1310
AGU 1312
AGU 1314
FAST ALU 1316
FAST ALU 1318
SLOW ALU 1320
FP 1322
FP MOVE 1324
TO LEVEL 1 CACHE
TO LEVEL 1 CACHE
EXE BLOCK 1311

FIG. 13

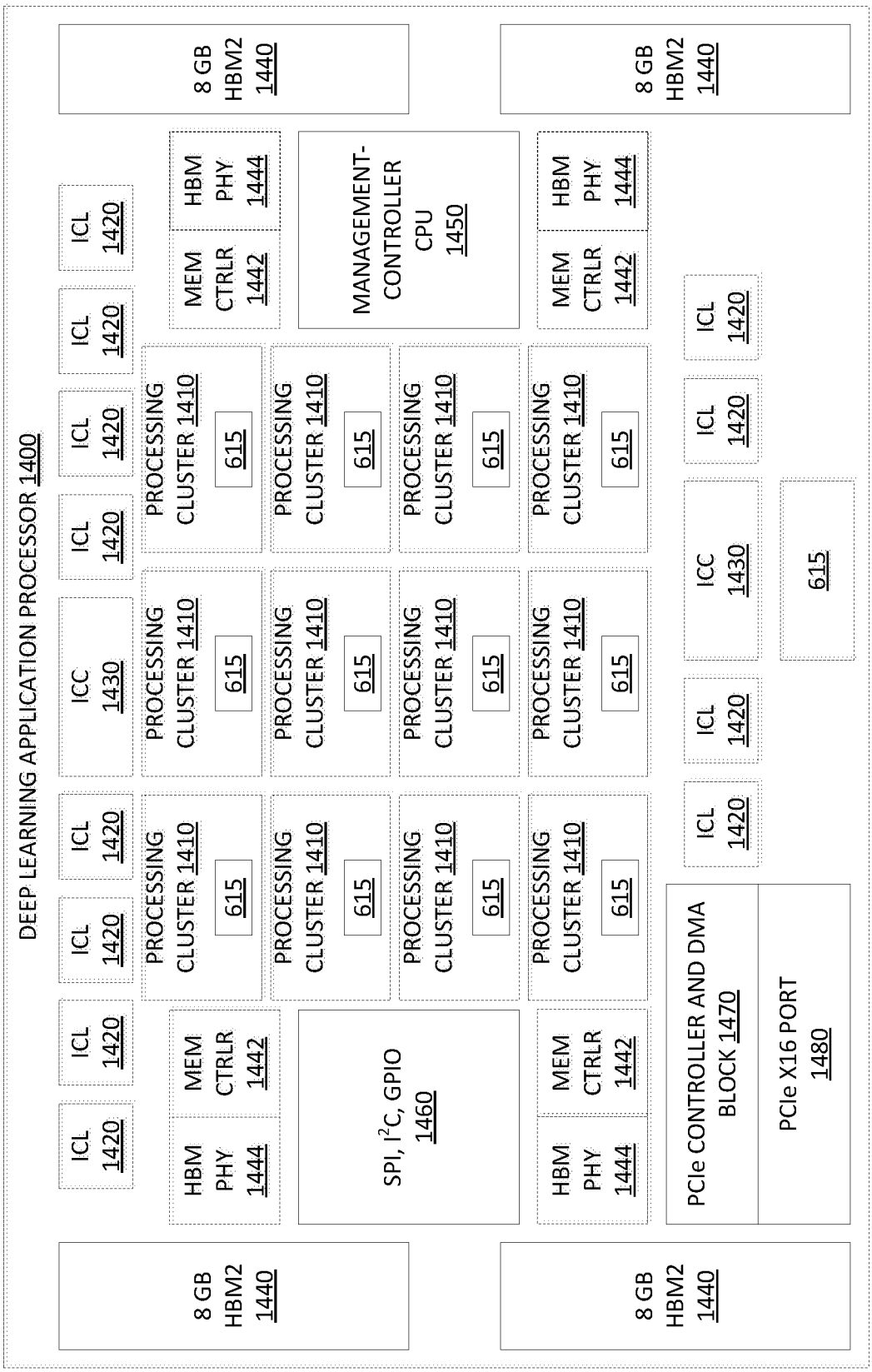
DEEP LEARNING APPLICATION PROCESSOR 1400
8 GB HBM2 1440
8 GB HBM2 1440
8 GB HBM2 1440
8 GB HBM2 1440
ICL 1420
ICC 1430
PROCESSING CLUSTER 1410
615
HBM PHY 1444
MEM CTRLR 1442
MANAGEMENT-CONTROLLER CPU 1450
SPI, I2C, GPIO 1460
PCIe CONTROLLER AND DMA BLOCK 1470
PCIe X16 PORT 1480

FIG. 14

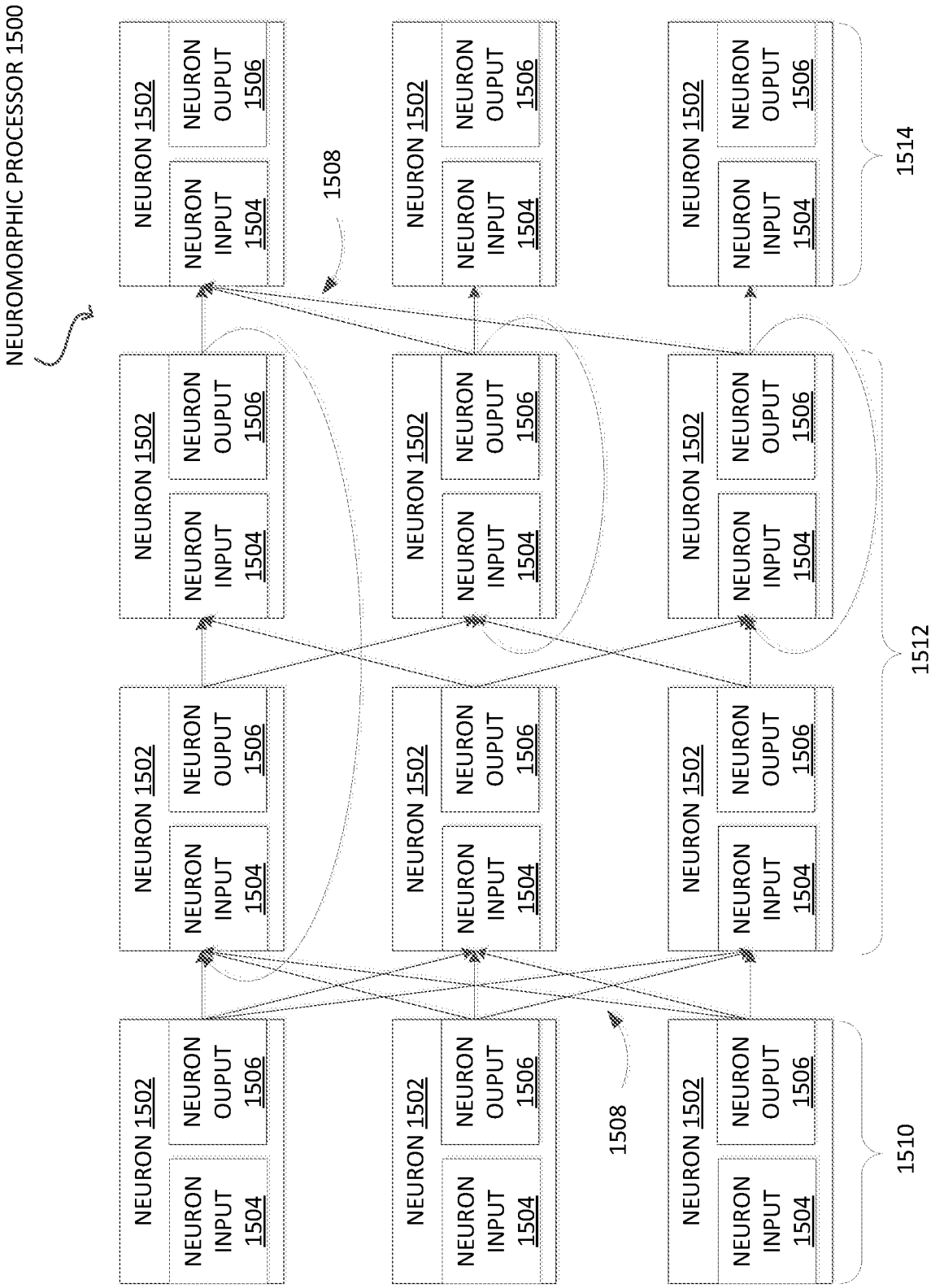
NEUROMORPHIC PROCESSOR 1500
NEURON 1502
NEURON INPUT 1504
NEURON OUPUT 1506
1508
1510
1512
1514

FIG. 15

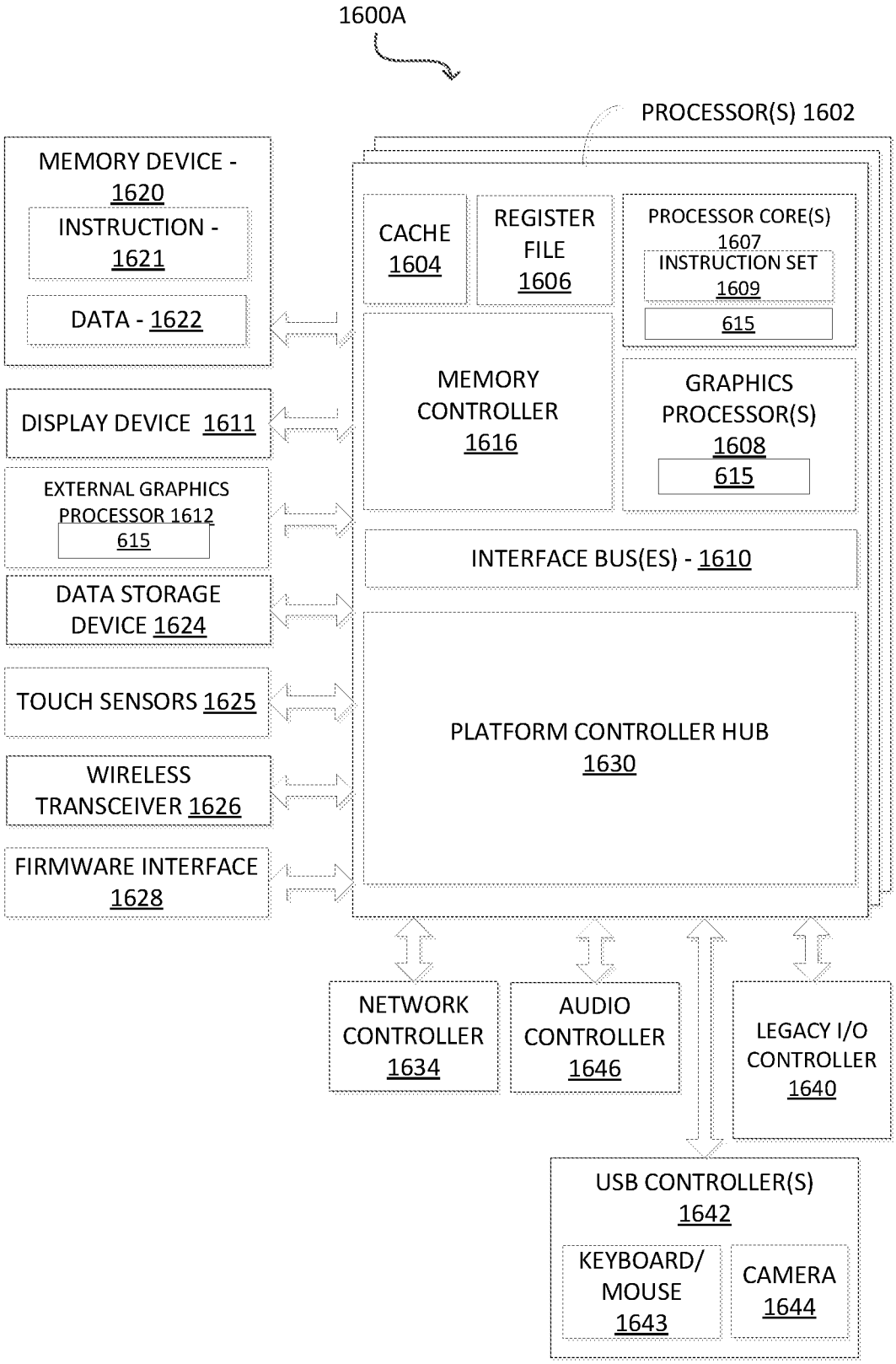
1600A
PROCESSOR(S) 1602
MEMORY DEVICE - 1620
INSTRUCTION - 1621
DATA - 1622
CACHE 1604
REGISTER FILE 1606
PROCESSOR CORE(S) 1607
INSTRUCTION SET 1609
615
MEMORY CONTROLLER 1616
GRAPHICS PROCESSOR(S) 1608
615
DISPLAY DEVICE 1611
EXTERNAL GRAPHICS PROCESSOR 1612
615
INTERFACE BUS(ES) - 1610
DATA STORAGE DEVICE 1624
TOUCH SENSORS 1625
PLATFORM CONTROLLER HUB 1630
WIRELESS TRANSCEIVER 1626
FIRMWARE INTERFACE 1628
NETWORK CONTROLLER 1634
AUDIO CONTROLLER 1646
LEGACY I/O CONTROLLER 1640
USB CONTROLLER(S) 1642
KEYBOARD/ MOUSE 1643
CAMERA 1644

FIG. 16A

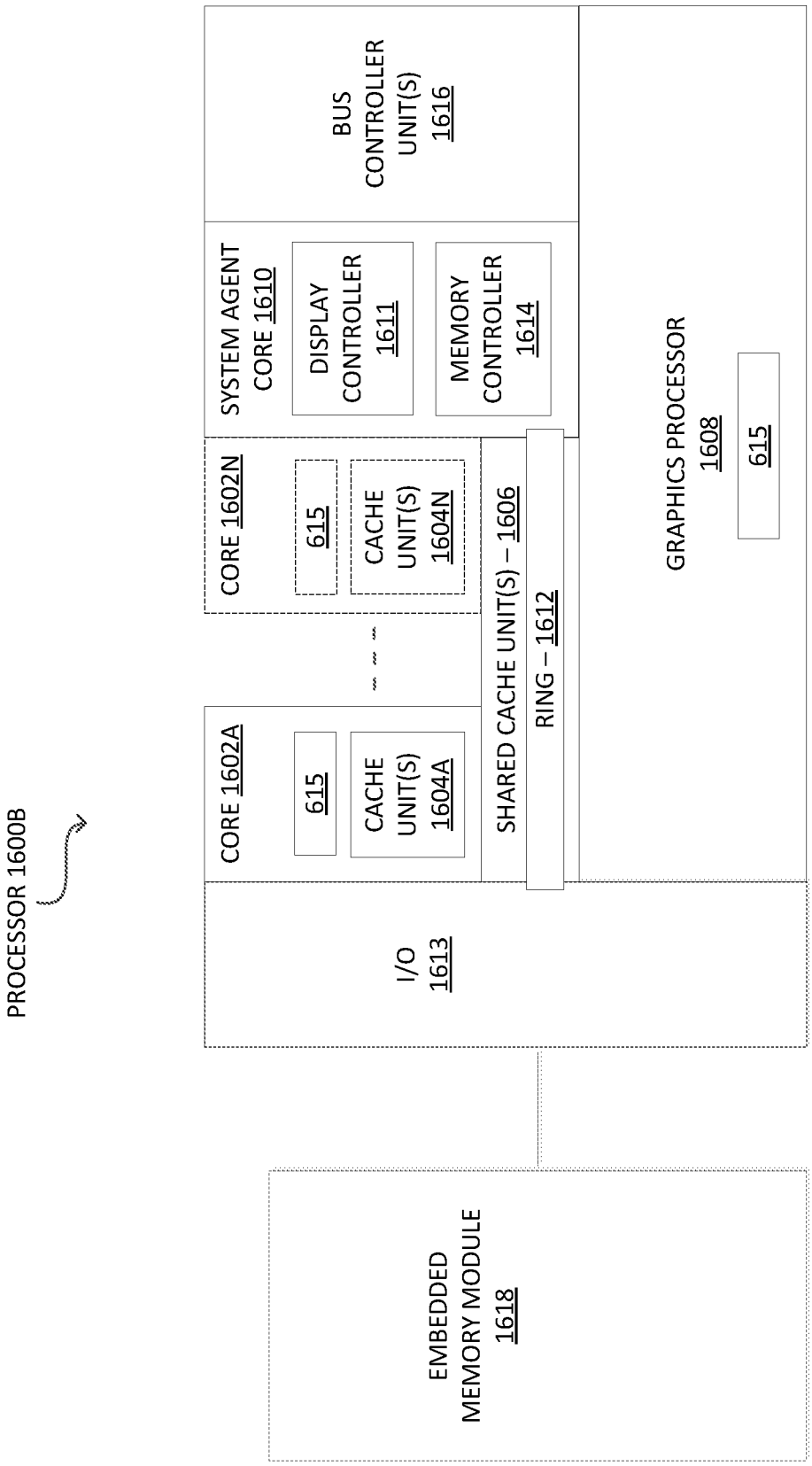
PROCESSOR 1600B
I/O 1613
CORE 1602A
615
CACHE UNIT(S) 1604A
CORE 1602N
615
CACHE UNIT(S) 1604N
SHARED CACHE UNIT(S) – 1606
RING – 1612
SYSTEM AGENT CORE 1610
DISPLAY CONTROLLER 1611
MEMORY CONTROLLER 1614
BUS CONTROLLER UNIT(S) 1616
GRAPHICS PROCESSOR 1608
615
EMBEDDED MEMORY MODULE 1618

FIG. 16B

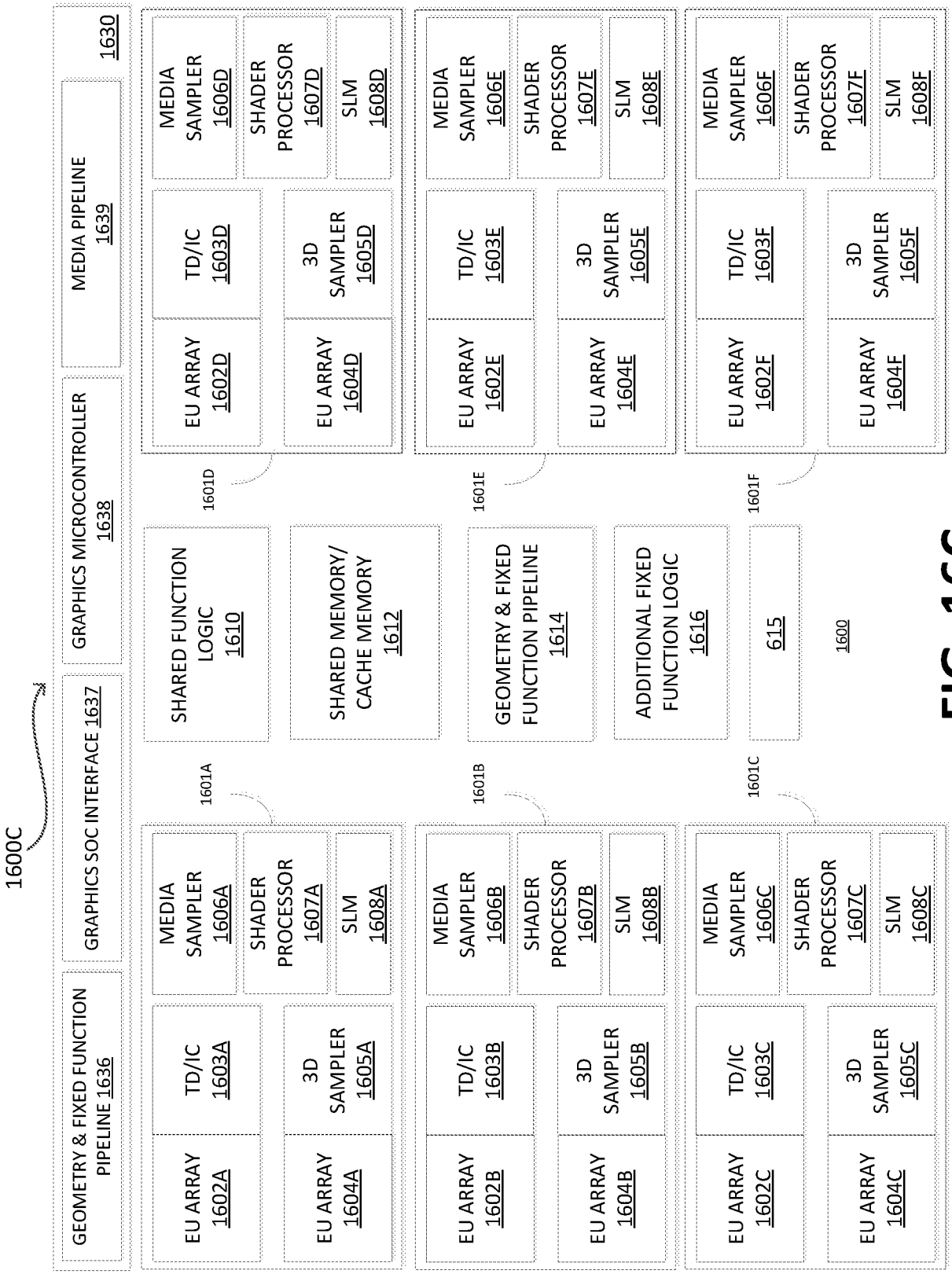
1600C
GEOMETRY & FIXED FUNCTION PIPELINE 1636
GRAPHICS SOC INTERFACE 1637
GRAPHICS MICROCONTROLLER 1638
MEDIA PIPELINE 1639
1630
SHARED FUNCTION LOGIC 1610
SHARED MEMORY/ CACHE MEMORY 1612
GEOMETRY & FIXED FUNCTION PIPELINE 1614
ADDITIONAL FIXED FUNCTION LOGIC 1616
615
1600
1601A
1601B
1601C
1601D
1601E
1601F
EU ARRAY 1602A
TD/IC 1603A
3D SAMPLER 1605A
EU ARRAY 1604A
MEDIA SAMPLER 1606A
SHADER PROCESSOR 1607A
SLM 1608A
EU ARRAY 1602B
TD/IC 1603B
3D SAMPLER 1605B
EU ARRAY 1604B
MEDIA SAMPLER 1606B
SHADER PROCESSOR 1607B
SLM 1608B
EU ARRAY 1602C
TD/IC 1603C
3D SAMPLER 1605C
EU ARRAY 1604C
MEDIA SAMPLER 1606C
SHADER PROCESSOR 1607C
SLM 1608C
EU ARRAY 1602D
TD/IC 1603D
3D SAMPLER 1605D
EU ARRAY 1604D
MEDIA SAMPLER 1606D
SHADER PROCESSOR 1607D
SLM 1608D
EU ARRAY 1602E
TD/IC 1603E
3D SAMPLER 1605E
EU ARRAY 1604E
MEDIA SAMPLER 1606E
SHADER PROCESSOR 1607E
SLM 1608E
EU ARRAY 1602F
TD/IC 1603F
3D SAMPLER 1605F
EU ARRAY 1604F
MEDIA SAMPLER 1606E
SHADER PROCESSOR 1607F
SLM 1608F

FIG. 16C

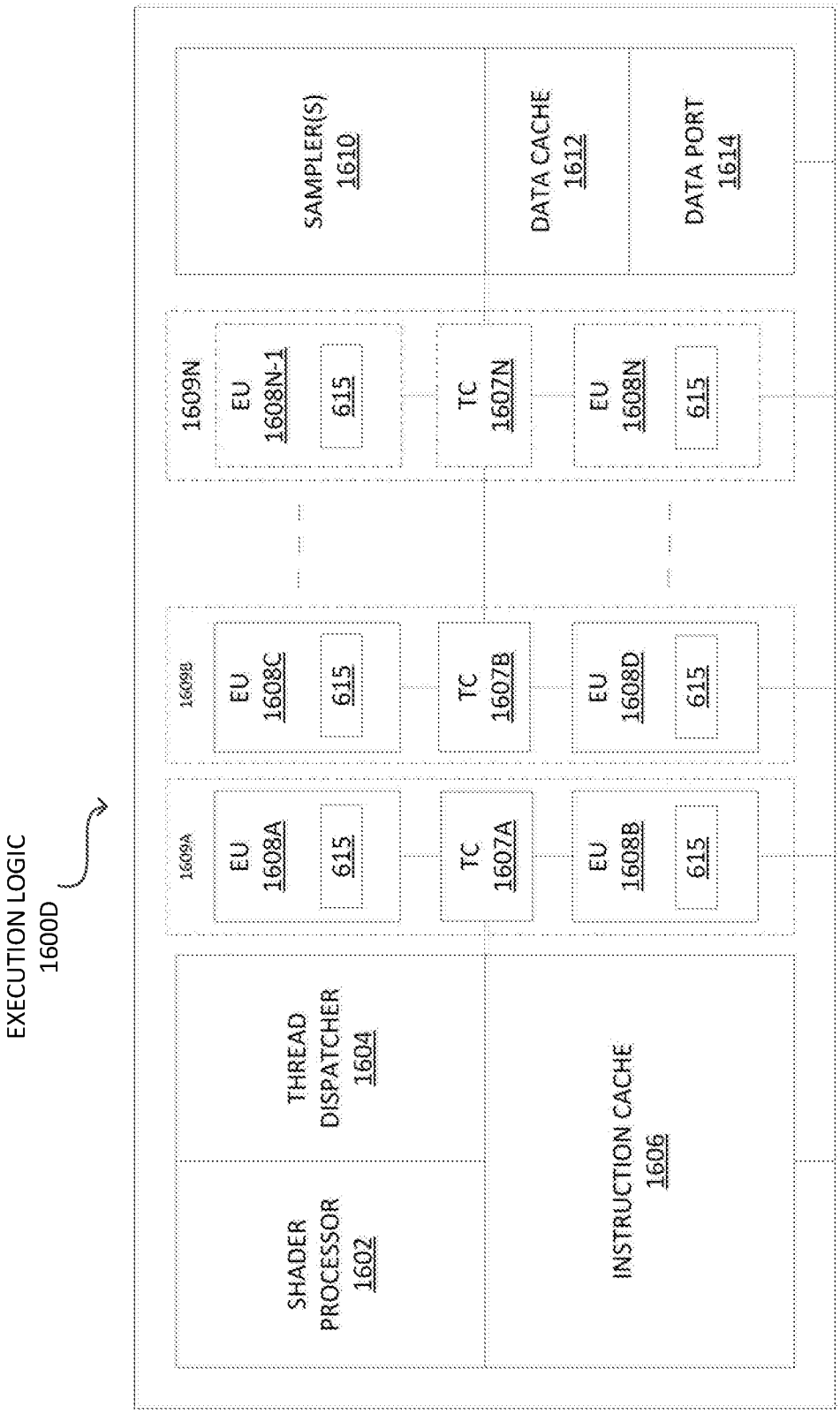
EXECUTION LOGIC
1600D
SHADER PROCESSOR 1602
THREAD DISPATCHER 1604
INSTRUCTION CACHE 1606
SAMPLER(S) 1610
DATA CACHE 1612
DATA PORT 1614
1609A
EU 1608A
615
TC 1607A
EU 1608B
615
1609B
EU 1608C
615
TC 1607B
EU 1608D
615
1609N
EU 1608N-1
615
TC 1607N
EU 1608N
615

FIG. 16D

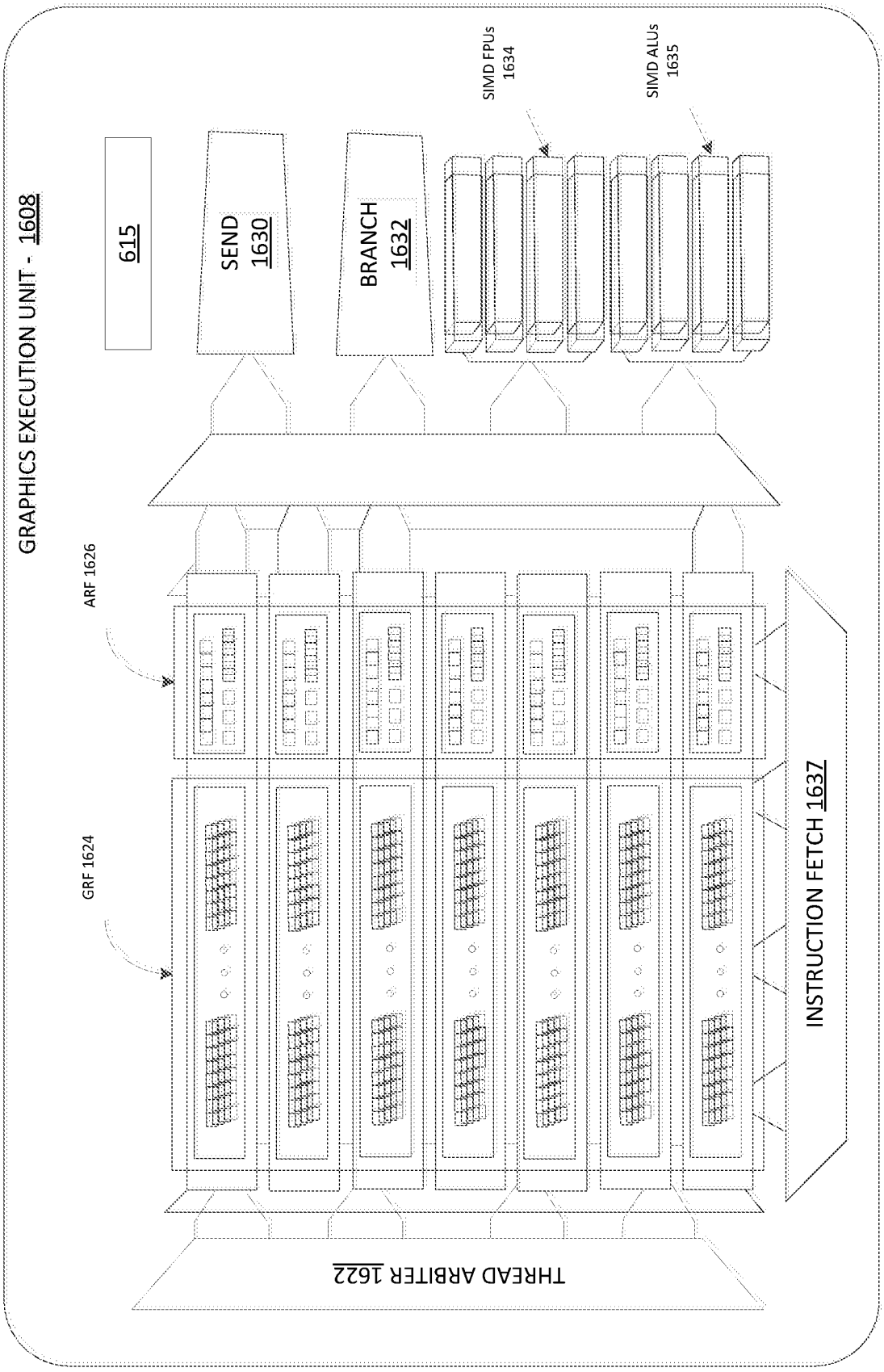
GRAPHICS EXECUTION UNIT - 1608
615
SEND 1630
BRANCH 1632
SIMD FPUs 1634
SIMD ALUs 1635
ARF 1626
GRF 1624
INSTRUCTION FETCH 1637
THREAD ARBITER 1622

FIG. 16E

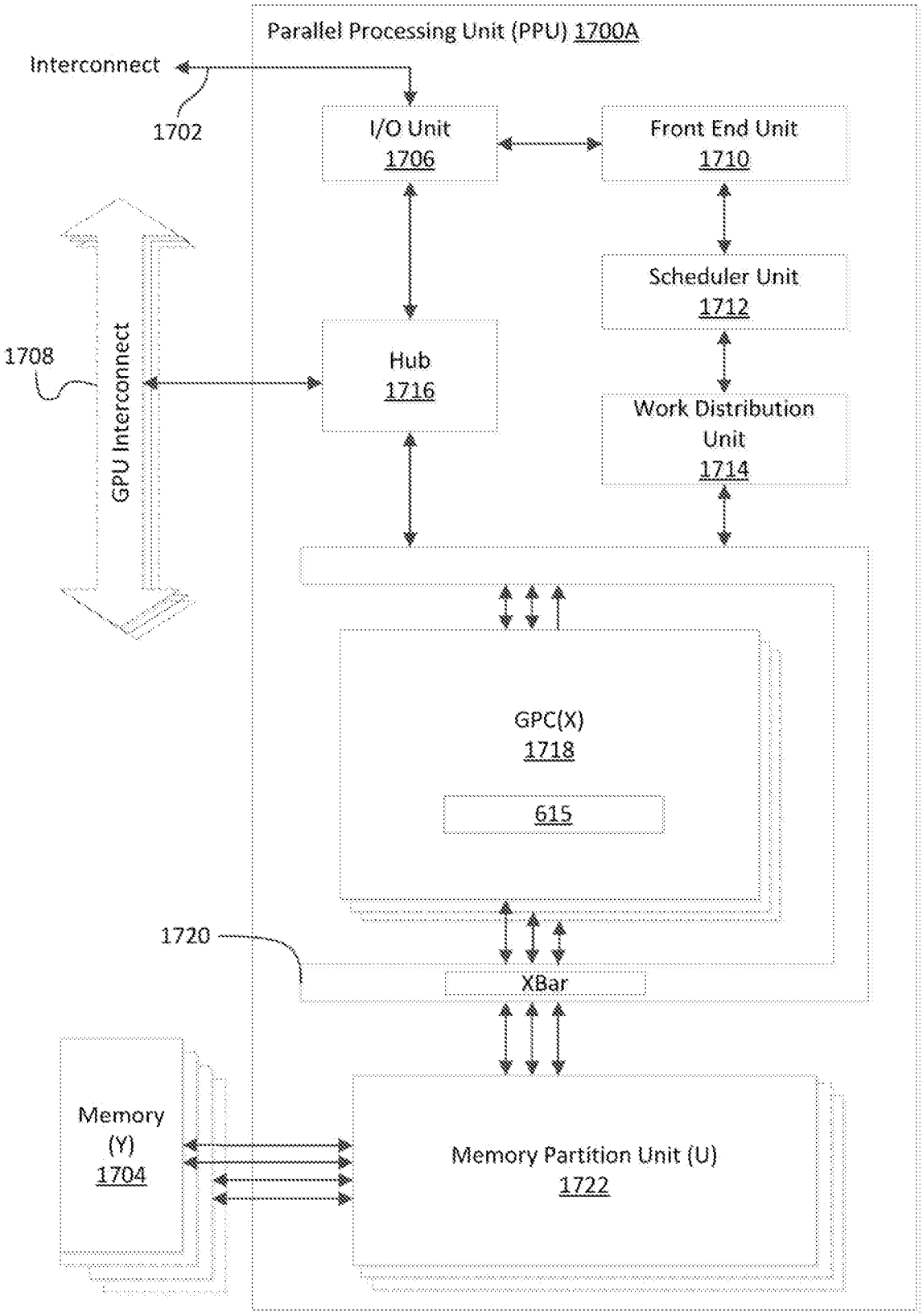
Parallel Processing Unit (PPU) 1700A
Interconnect
1702
I/O Unit
1706
Front End Unit
1710
Scheduler Unit
1712
1708
GPU Interconnect
Hub
1716
Work Distribution Unit
1714
GPC(X)
1718
615
1720
XBar
Memory
(Y)
1704
Memory Partition Unit (U)
1722

FIG. 17A

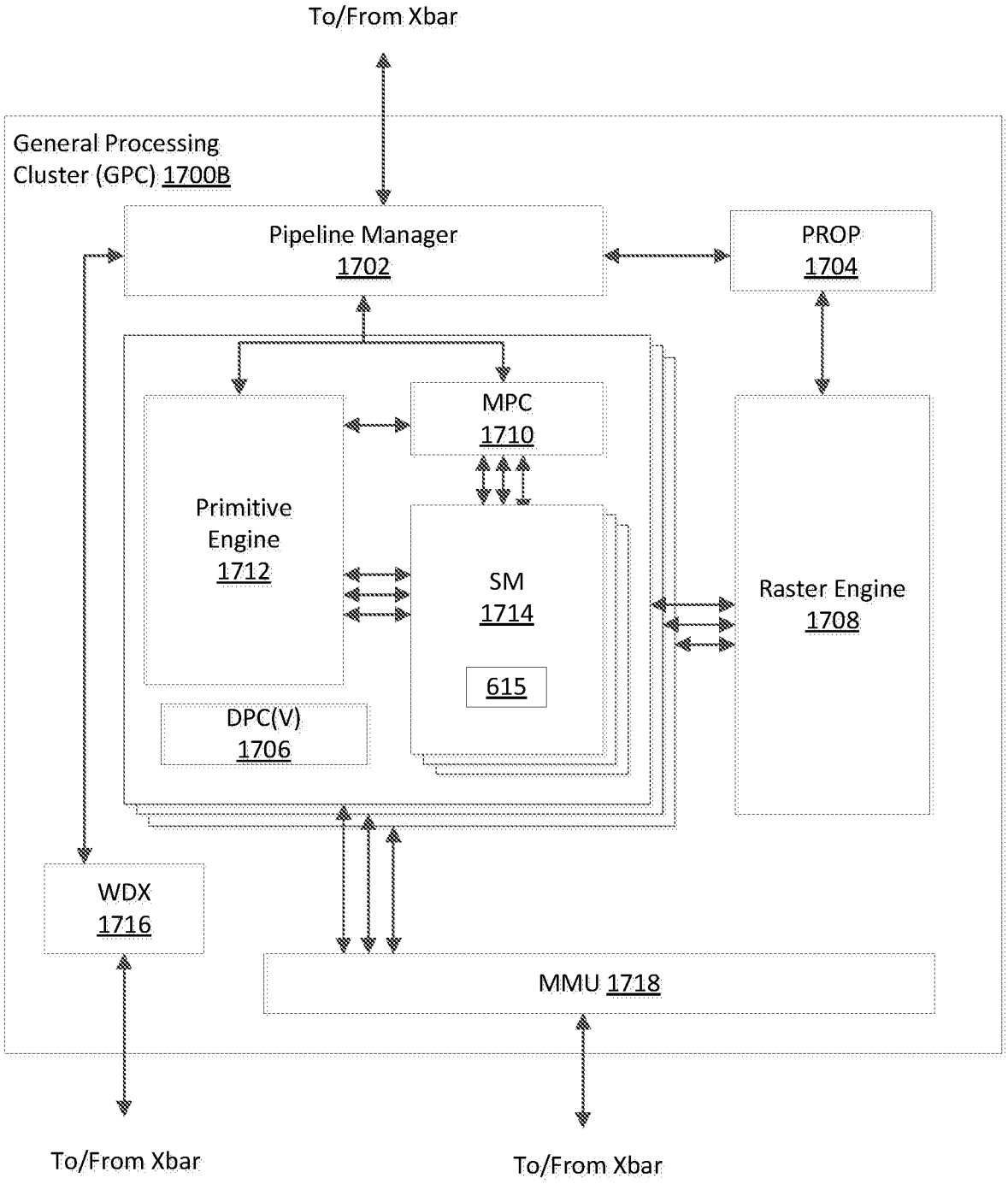
To/From Xbar
General Processing Cluster (GPC) 1700B
Pipeline Manager 1702
PROP 1704
MPC 1710
Primitive Engine 1712
SM 1714
615
Raster Engine 1708
DPC(V) 1706
WDX 1716
MMU 1718
To/From Xbar
To/From Xbar

FIG. 17B

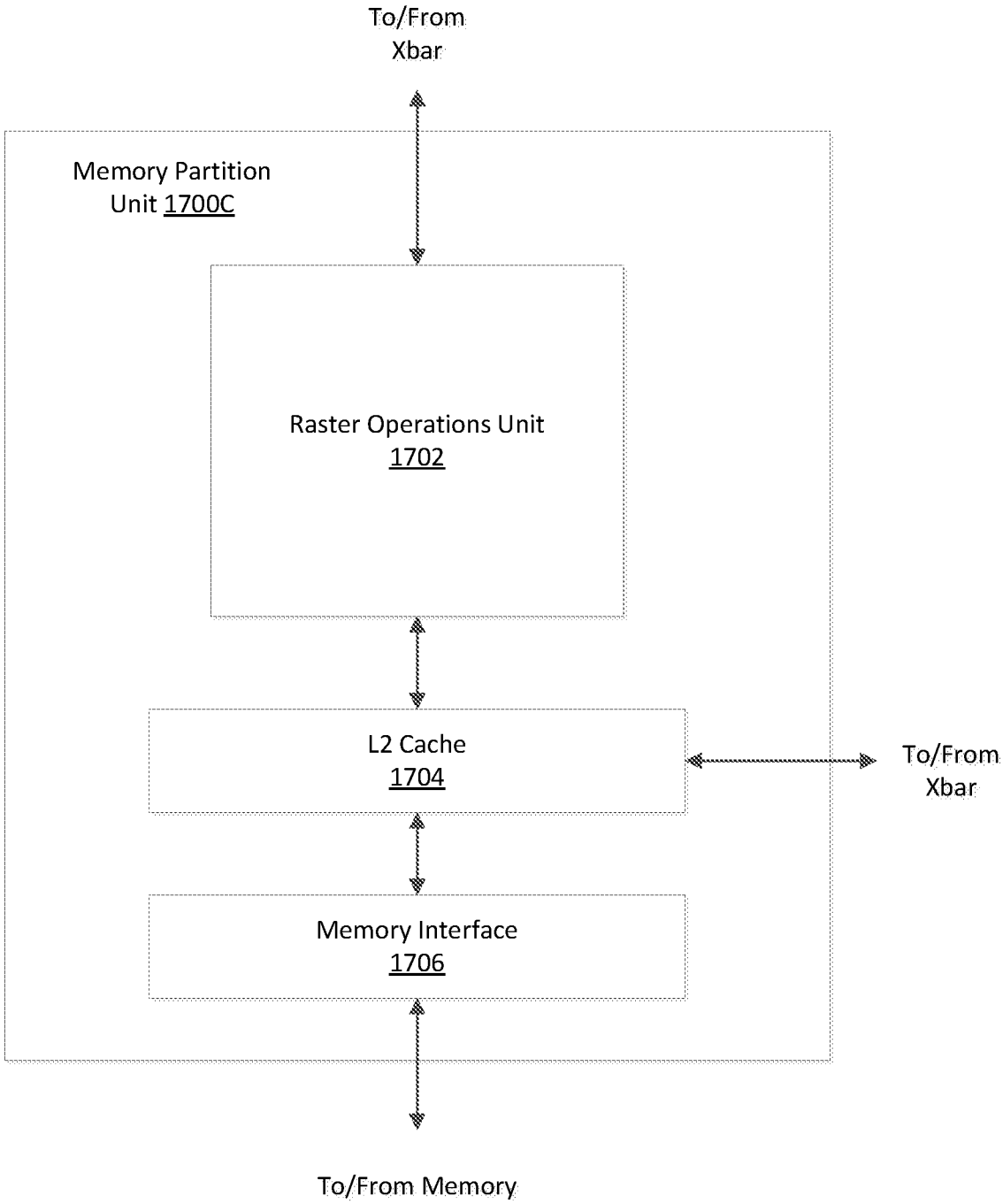
To/From
Xbar
Memory Partition
Unit 1700C
Raster Operations Unit
1702
L2 Cache
1704
To/From
Xbar
Memory Interface
1706
To/From Memory

FIG. 17C

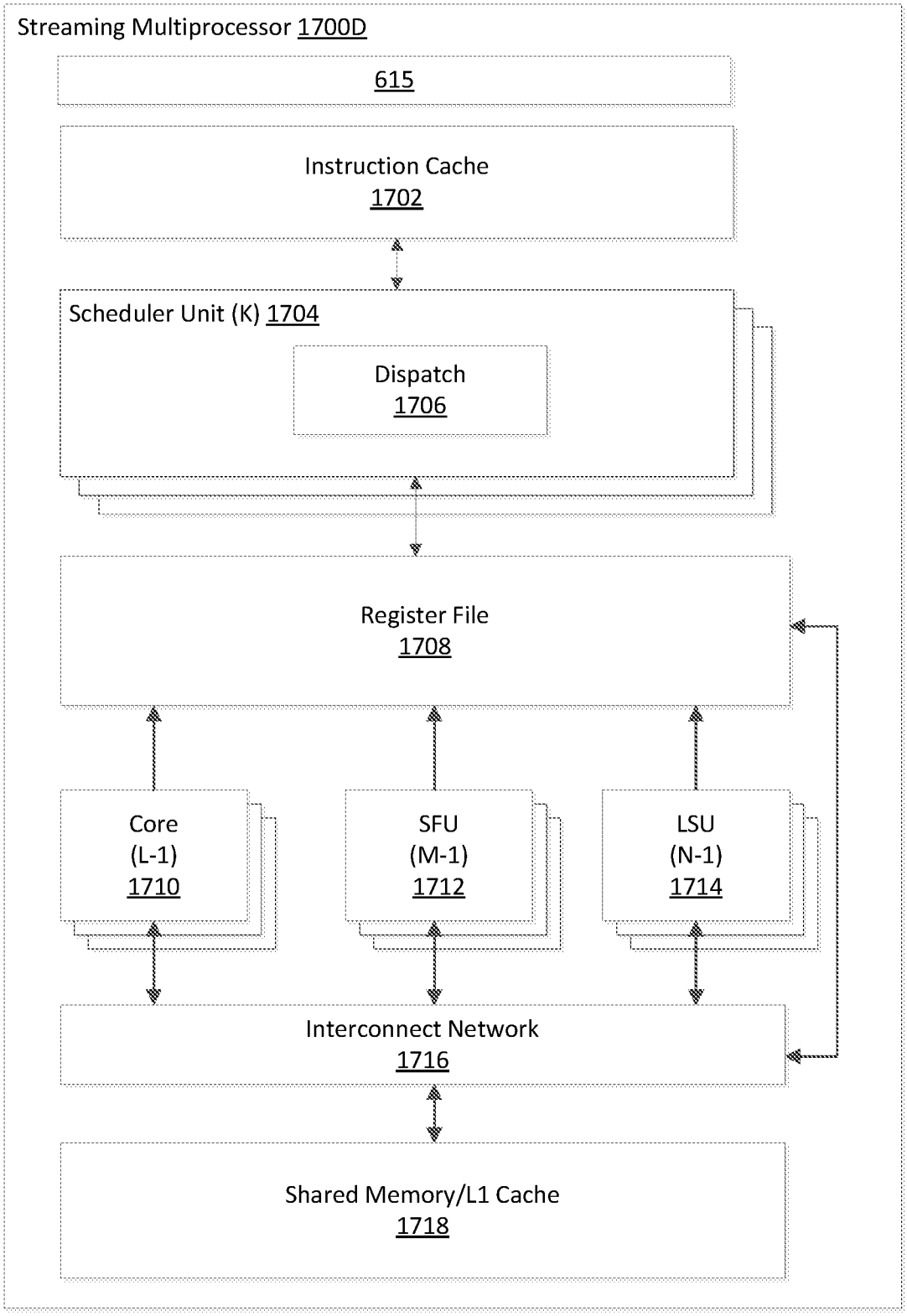
Streaming Multiprocessor 1700D
615
Instruction Cache
1702
Scheduler Unit (K) 1704
Dispatch
1706
Register File
1708
Core
(L-1)
1710
SFU
(M-1)
1712
LSU
(N-1)
1714
Interconnect Network
1716
Shared Memory/L1 Cache
1718

FIG. 17D

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2020/110811

A. CLASSIFICATION OF SUBJECT MATTER

H05K 7/20(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H05K7; H01L23

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

DWPI;CNABS;CNTXT:heat, sink, radiator, cooler, fin?, expansion, extension, elongation, float+, area

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 103841808 A (INST. ENG. THERMOPHYSICS CHINESE) 04 June 2014 (2014-06-04) paragraph 8, 33-50 and figure 3A-4	23-26
A	CN 109887894 A (BEIJING BITMAIN TECHNOLOGY CO., LTD.) 14 June 2019 (2019-06-14) the whole document	1-30
A	CN 110785063 A (BINZHOU SHIP PLA. NAVY et al.) 11 February 2020 (2020-02-11) the whole document	1-30
A	CN 106413343 A (HUAWEI TECHNOLOGIES CO., LTD.) 15 February 2017 (2017-02-15) the whole document	1-30
A	US 2013277439 A1 (HYUNDAI MOTOR CO., LTD. et al.) 24 October 2013 (2013-10-24) the whole document	1-30
A	US 2013180563 A1 (TEMPRONICS INC.et al.) 18 July 2013 (2013-07-18) the whole document	1-30

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
08 May 2021	**20 May 2021**
Name and mailing address of the ISA/CN **National Intellectual Property Administration, PRC** **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer **JIE,Qian** Telephone No. **010-53962952**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2020/110811

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 103841808 A	04 June 2014	CN 103841808 B	18 May 2016
CN 109887894 A	14 June 2019	CN 109887894 B	20 November 2020
CN 110785063 A	11 February 2020	None	
CN 106413343 A	15 February 2017	US 2019206763 A1	04 July 2019
		CN 106413343 B	26 April 2019
		WO 2018045933 A1	15 March 2018
		EP 3503701 A1	26 June 2019
US 2013277439 A1	24 October 2013	US 2017172013 A1	15 June 2017
		US 10375858 B2	06 August 2019
		KR 20130119076 A	31 October 2013
		KR 101349013 B1	10 January 2014
		US 9603288 B2	21 March 2017
US 2013180563 A1	18 July 2013	WO 2013103585 A1	11 July 2013