



(19) **United States**

(12) **Patent Application Publication**
SALEH et al.

(10) **Pub. No.: US 2020/0409859 A1**

(43) **Pub. Date: Dec. 31, 2020**

(54) **GPU CHIPLETS USING HIGH BANDWIDTH
CROSSLINKS**

Publication Classification

(71) Applicant: **ADVANCED MICRO DEVICES,
INC.**, Santa Clara, CA (US)

(51) **Int. Cl.**
G06F 12/0897 (2006.01)
G06F 13/16 (2006.01)
G06F 13/40 (2006.01)
(52) **U.S. Cl.**
CPC .. *G06F 12/0897* (2013.01); *G06F 2212/1024*
(2013.01); *G06F 13/4027* (2013.01); *G06F*
13/1668 (2013.01)

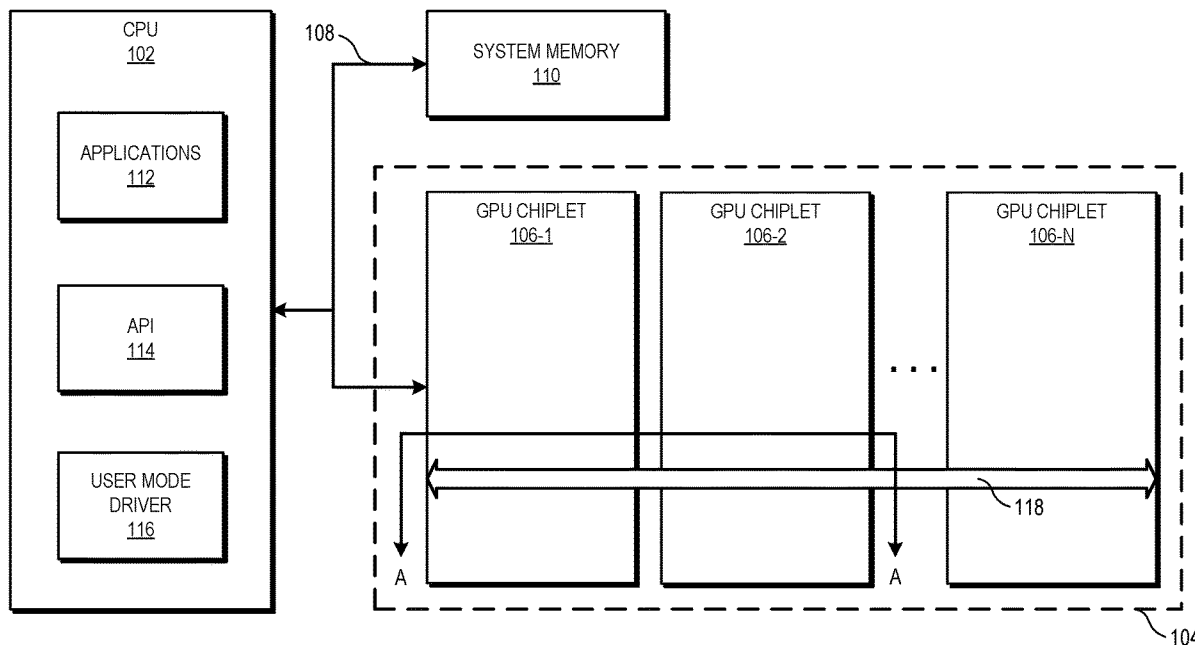
(72) Inventors: **Skyler J. SALEH**, San Diego, CA
(US); **Samuel NAFFZIGER**, Fort
Collins, CO (US); **Milind S.**
BHAGAVAT, Santa Clara, CA (US);
Rahul AGARWAL, Santa Clara, CA
(US)

(57) **ABSTRACT**

A chiplet system includes a central processing unit (CPU) communicably coupled to a first GPU chiplet of a GPU chiplet array. The GPU chiplet array includes the first GPU chiplet communicably coupled to the CPU via a bus and a second GPU chiplet communicably coupled to the first GPU chiplet via a passive crosslink. The passive crosslink is a passive interposer die dedicated for inter-chiplet communications and partitions systems-on-a-chip (SoC) functionality into smaller functional chiplet groupings.

(21) Appl. No.: **16/456,287**

(22) Filed: **Jun. 28, 2019**



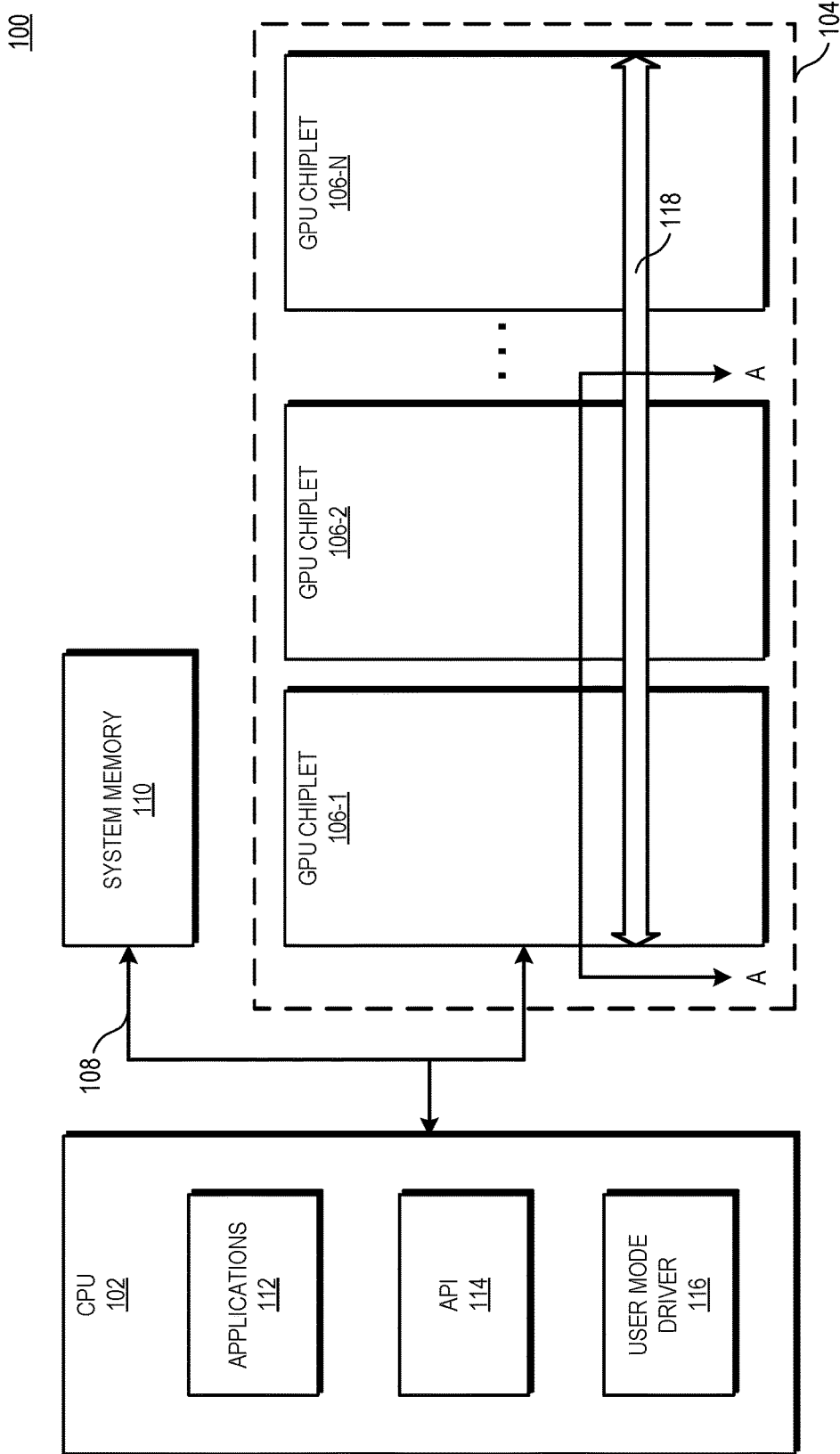


FIG. 1

200

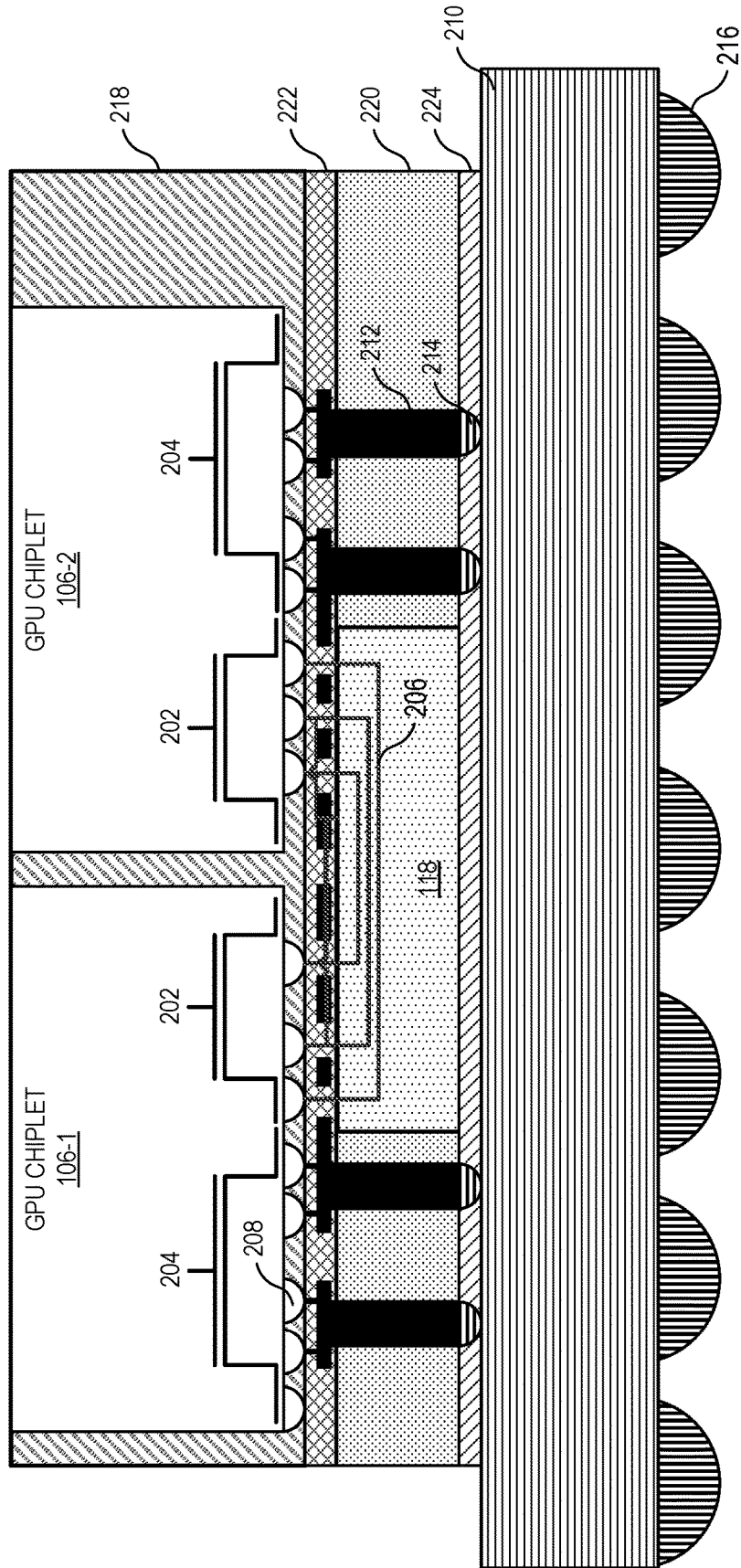


FIG. 2

300

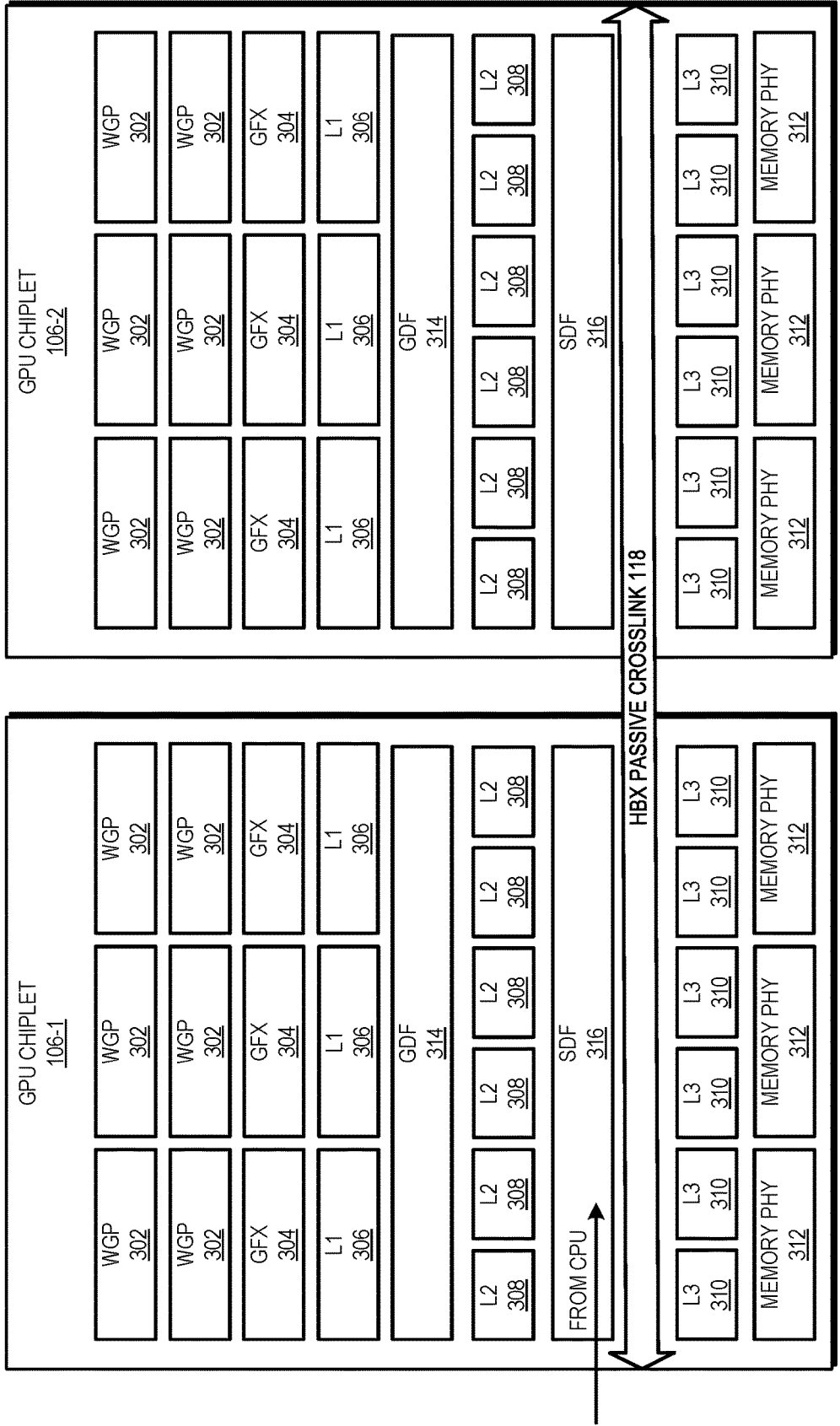


FIG. 3

400



FIG. 4

500

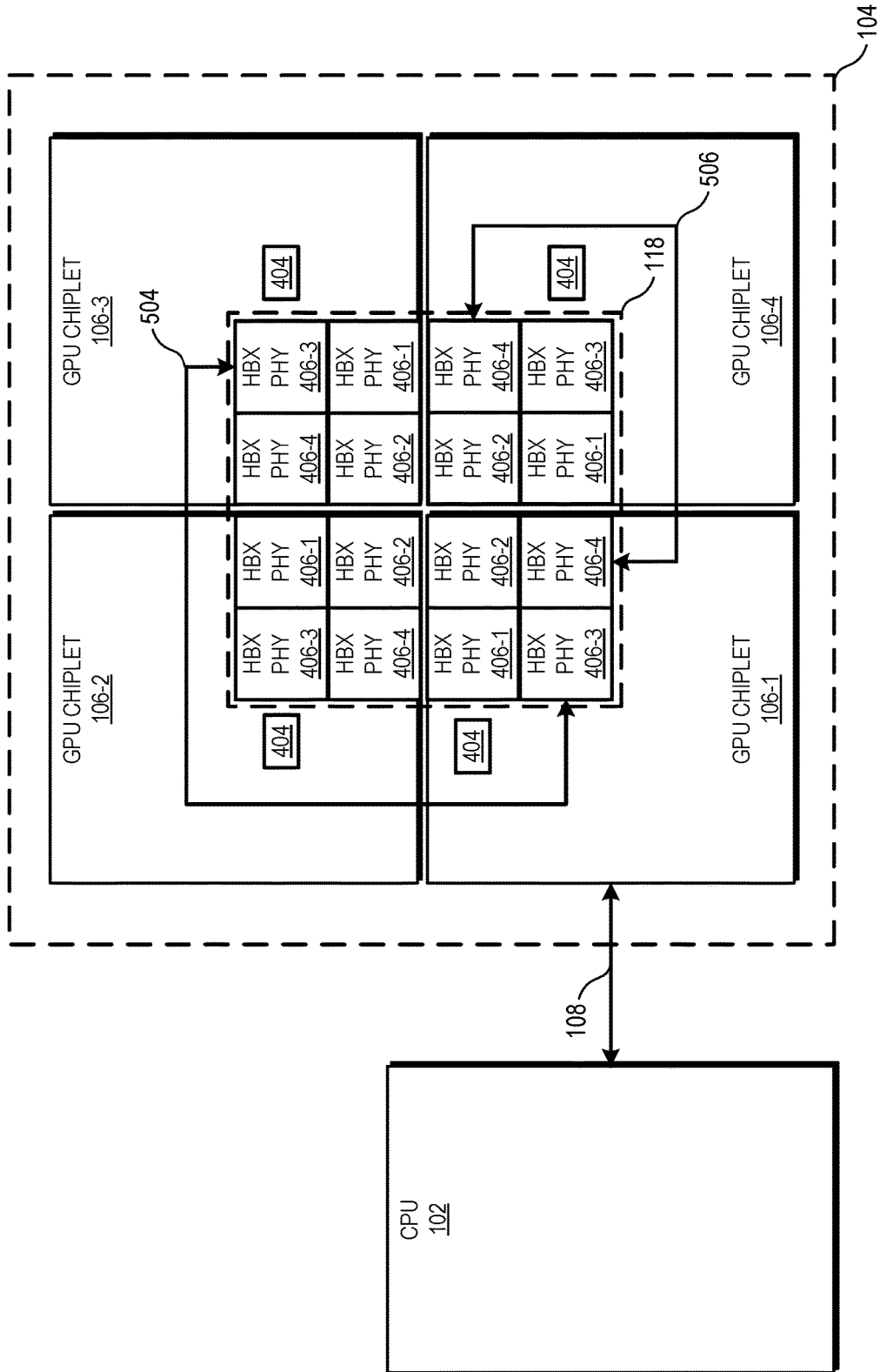


FIG. 5

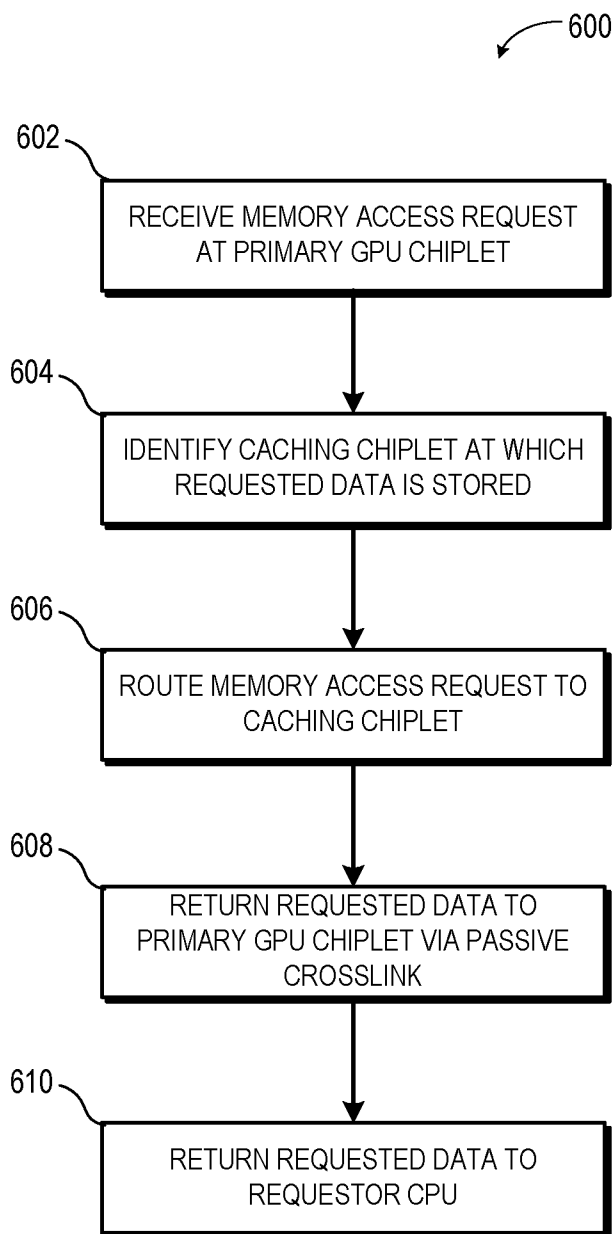


FIG. 6

GPU CHIPLETS USING HIGH BANDWIDTH CROSSLINKS

BACKGROUND

[0001] Computing devices such as mobile phones, personal digital assistants (PDAs), digital cameras, portable players, gaming, and other devices requires the integration of more performance and features into increasingly smaller spaces. As a result, the density of processor dies and number of dies integrated within a single integrated circuit (IC) package have increased. Some conventional multi-chip modules include two or more semiconductor chips mounted side-by-side on a carrier substrate or in some cases on an interposer (so-called “2.5D”) that is, in-turn, mounted on a carrier substrate.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The present disclosure may be better understood, and its numerous features and advantages made apparent to those skilled in the art by referencing the accompanying drawings. The use of the same reference symbols in different drawings indicates similar or identical items.

[0003] FIG. 1 is a block diagram illustrating a processing system employing high bandwidth passive crosslinks for coupling GPU chiplets in accordance with some embodiments.

[0004] FIG. 2 is a block diagram illustrating a sectional view of GPU chiplets and passive crosslinks in accordance with some embodiments.

[0005] FIG. 3 is a block diagram illustrating a cache hierarchy of GPU chiplets coupled by a passive crosslink in accordance with some embodiments.

[0006] FIG. 4 is a block diagram illustrating a floor plan view of a GPU chiplet in accordance with some embodiments.

[0007] FIG. 5 is a block diagram illustrating a processing system utilizing a four-chiplet configuration in accordance with some embodiments.

[0008] FIG. 6 is a flow diagram illustrating a method of performing inter-chiplet communications in accordance with some embodiments.

DETAILED DESCRIPTION

[0009] Conventional monolithic die designs are becoming increasingly expensive to manufacture. Chiplets have been used successfully in CPU architectures to reduce cost of manufacture and improve yields, as its heterogeneous computational nature is more naturally suited to separate CPU cores into distinct units that do not require much inter-communication. GPU work by its nature includes parallel work. However, the geometry that a GPU processes includes not only sections of fully parallel work but also work that requires synchronous ordering between different sections. Accordingly, a GPU programming model that spreads sections of work on different threads is often inefficient because the parallelism is difficult to distribute across multiple different working groups and chiplets as it is difficult and expensive computationally to synchronize the memory contents of shared resources throughout the entire system to provide a coherent view of the memory to applications. Additionally, from a logical point of view, applications are written with the view that the system only has a single GPU. That is, even though a conventional GPU includes many

GPU cores, applications are programmed as addressing a single device. So, it has been historically challenging to bring chiplet design methodology to GPU architectures.

[0010] To improve system performance by using GPU chiplets while preserving the current programming model, FIGS. 1-6 illustrate systems and methods that utilize high bandwidth passive crosslinks for coupling GPU chiplets. In various embodiments, a system includes a central processing unit (CPU) communicably coupled to a first graphics processing unit (GPU) chiplet of a GPU chiplet array. The GPU chiplet array includes the first GPU chiplet communicably coupled to the CPU via a bus and a second GPU chiplet communicably coupled to the first GPU chiplet via a passive crosslink. In various embodiments, the passive crosslink is a passive interposer die dedicated for inter-chiplet communications. The GPU chiplet breaks up systems-on-a-chip (SoC) into smaller functional groupings referred to as “chiplets” that perform functions of the various cores of the SoC (e.g., a GPU).

[0011] Currently, various architectures already have at least one level of cache (e.g., L3 or other last level cache (LLC)) that is coherent across the entire conventional GPU die. Here, the chiplet-based GPU architecture positions those physical resources (e.g., LLC) on different dies and communicably couples those physical resources such that the LLC level is unified and remains cache coherent across all GPU chiplets. Thus, although operating in a massively parallel environment, the L3 cache level to be coherent. During operations, a memory address request from the CPU to the GPU is transmitted only to a single GPU chiplet, which then communicates with a high bandwidth passive crosslink to locate the requested data. From the CPU’s point of view, it appears to be addressing a single die, monolithic GPU. This allows for use of a large-capacity, multi-chiplet GPU that appears as a single device to an application.

[0012] FIG. 1 is a block diagram illustrating a processing system 100 employing high bandwidth passive crosslinks for coupling GPU chiplets in accordance with some embodiments. In the depicted example, the system 100 includes a central processing unit (CPU) 102 for executing instructions and an array 104 of one or more GPU chiplets, such as the three illustrated GPU chiplets 106-1, 106-2, and through 106-N (collectively, GPU chiplets 106). In various embodiments, and as used herein, the term “chiplet” refers to any device including, but is not limited to, the following characteristics: 1) a chiplet includes an active silicon die containing part of the computational logic used to solve a full problem (i.e., the computational workload is distributed across multiples of these active silicon dies); 2) chiplets are packaged together as a monolithic unit on the same substrate; and 3) the programming model preserves the concept that these separate computational dies are a single monolithic unit (i.e., each chiplet is not exposed as a separate device to an application that uses the chiplets for processing computational workloads).

[0013] In various embodiments, the CPU 102 is connected via a bus 108 to a system memory 110, such as a dynamic random access memory (DRAM). In various embodiments, the system memory 110 can also be implemented using other types of memory including static random access memory (SRAM), nonvolatile RAM, and the like. In the illustrated embodiment, the CPU 102 communicates with the system memory 110 and also the GPU chiplet 106-1 over bus 108 that is implemented as a peripheral component interconnect

(PCI) bus, PCI-E bus, or other type of bus. However, some embodiments of the system **100** includes the GPU chiplet **106-1** communicating with the CPU **102** over a direct connection or via other buses, bridges, switches, routers, and the like.

[0014] As illustrated, the CPU **102** includes a number of processes, such as executing one or more application(s) **112** to generate graphic commands and a user mode driver **116** (or other drivers, such as a kernel mode driver). In various embodiments, the one or more applications **112** include applications that utilizes the functionality of the GPU chiplets **106**, such as applications that generate work in the system **100** or an operating system (OS). An application **112** may include one or more graphics instructions that instruct the GPU chiplets **106** to render a graphical user interface (GUI) and/or a graphics scene. For example, the graphics instructions may include instructions that define a set of one or more graphics primitives to be rendered by the GPU chiplets **106**.

[0015] In some embodiments, the application **112** utilizes a graphics application programming interface (API) **114** to invoke a user mode driver **116** (or a similar GPU driver). User mode driver **116** issues one or more commands to the array **104** of one or more GPU chiplets for rendering one or more graphics primitives into displayable graphics images. Based on the graphics instructions issued by application **112** to the user mode driver **116**, the user mode driver **116** formulates one or more graphics commands that specify one or more operations for GPU chiplets to perform for rendering graphics. In some embodiments, the user mode driver **116** is a part of the application **112** running on the CPU **102**. For example, the user mode driver **116** may be part of a gaming application running on the CPU **102**. Similarly, a kernel mode driver (not shown) may be part of an operating system running on the CPU **102**.

[0016] In the depicted embodiment of FIG. 1, a passive crosslink **118** communicably couples the GPU chiplets **106** (i.e., GPU chiplets **106-1** through **106-N**) to each other. Although three GPU chiplets **106** are shown in FIG. 1, the number of GPU chiplets in the chiplet array **104** is a matter of design choice and may vary in other embodiments, such as described in more detail below. In various embodiments, the passive crosslink **118** includes an interconnect chip such as a high density crosslink (HDCL) die interposer or other similar technology for inter-chiplet communications. As a general operational overview, the CPU **102** is communicably coupled to a single GPU chiplet (i.e., GPU chiplet **106-1**) through the bus **108**. CPU-to-GPU transactions or communications from the CPU **102** to the array **104** of chiplets **106** is received at the GPU chiplet **106-1**. Subsequently, any inter-chiplet communications is routed through the passive crosslink **118** as appropriate to access memory channels on other GPU chiplets **106**. In this manner, the GPU chiplet-based system **100** includes GPU chiplets **106** that are addressable as a single, monolithic GPU from a software developer's perspective (e.g., the CPU **102** and any associated applications/drivers are unaware of the chiplet-based architecture), and therefore avoids requiring any chiplet-specific considerations on the part of a programmer or developer.

[0017] Additional details of the chiplet-based architecture may be understood with reference to FIG. 2, which is a block diagram illustrating a sectional view of GPU chiplets and passive crosslink in accordance with some embodiments.

The view **200** provides a sectional view of GPU chiplets **106-1**, **106-2**, and passive crosslink **118** of FIG. 1 taken at section A-A. In various embodiments, each GPU chiplet **106** is constructed with a physical device (PHY) region **202**, which has various internal and external conductor structures dedicated to the transmission of chiplet-to-chiplet signals, and a non-PHY region **204**, which has conductor structures that are tailored more to the conveyance of power and ground and/or chiplet-to-circuit board signals.

[0018] As previously noted, the GPU chiplets **106** are communicably coupled by way of the passive crosslink **118**. In various embodiments, the passive crosslink **118** is an interconnect chip constructed of silicon, germanium or other semiconductor materials and may be bulk semiconductor, semiconductor on insulator or other designs. The passive crosslink **118** includes a plurality of internal conductor traces, which may be on a single level or multiple levels as desired. Three of the traces are illustrated in FIG. 2 and labeled collectively as traces **206**. The traces **206** interface electrically with conductor structures of the PHY regions **202** of the GPU chiplets **106** by way of conducting pathways. It is noted that the passive crosslink **118** does not contain any through silicon vias (TSVs). In this manner, the passive crosslink **118** is a passive interposer die that communicably couples and routes communications between the GPU chiplets **106**, thereby forming a passive routing network.

[0019] The non-PHY regions **204** of the GPU chiplets **106** interface electrically with a circuit board **210** (or any other substrate) by way of a plurality of conductive pillars **212**. Each conductive pillar **212** is connected electrically to the GPU chiplets **106** by way of a solder interconnect **208**, which may include a solder bump, micro bump, and the like. In various embodiments, the circuit board **210** interfaces electrically with other electrical structure, such as another circuit board or other structures, by way of a plurality of interconnect structures **214** (e.g., solder balls and the like). However, those skilled in the art will appreciate that various types of interconnect structures such as pins, land grid array structures, other interconnects, and the like may be used without departing from the scope of this disclosure.

[0020] The conductive pillars **212** connect signals between the GPU chiplets **106** and the substrate in areas where the HDCL die is not present (e.g., areas with vertical discrepancy between the GPU chiplets **106** and the circuit board **210**), with empty spaces filled with epoxy or other gap fill materials. In this manner, power and input/output (I/O) lines of the non-PHY regions **204** are routed around the interposer die (i.e., passive crosslink **118**) using redistribution (RDL) technology, thereby replacing usage of conventional TSVs. For example, as illustrated in the embodiment of FIG. 2, the GPU chiplets **106** and passive crosslink **118** are at least partially encapsulated in a molding material, such as the two molding layers **218** and **220** of FIG. 2. The conductive pillars **212** traverse a plurality of insulating layers, such as the molding layer **220** and a polymer layer **222**. In various embodiments, the polymer layer **222** is a RDL layer designed to act as a stress buffer and/or an isolation film for enabling redistribution layer routing. The conductive pillars **212** include various conductor materials, such as copper and the like. Similarly, the solder interconnects **208** and interconnect structures **214** include materials using various solder compositions, such as tin-silver, tin-silver-copper, and the like.

[0021] The circuit board **210** may include organic or ceramic and single, or more commonly, multilayer materials. To cushion against the effects of mismatched coefficients of thermal expansion, an underfill material **224** (e.g., polymeric underfill) may be positioned between the molding layer **220** and an upper surface of the circuit board **210**. The underfill material **224** may extend laterally beyond the left and right edges (and other edges not visible in FIG. 2) of the molding layer **220** as desired.

[0022] FIG. 3 is a block diagram illustrating a cache hierarchy of GPU chiplets coupled by a passive crosslink in accordance with some embodiments. The view **300** provides a hierarchical view of GPU chiplets **106-1**, **106-2**, and passive crosslink **118** of FIG. 1. Each of the GPU chiplets **106-1**, **106-2** include a plurality of workgroup processors **302** (WGP) and a plurality of fixed function blocks **304** (GFX) that communicate with a given channel's L1 cache memory **306**. Each GPU chiplet **106** also includes a plurality of L2 cache memory **308** banks that are individually accessible, a plurality of L3 cache memory **310** channels, and a plurality of memory PHY **312** (denoted as graphics double data rate (GDDR) in FIG. 3 for indicating connection to GDDR memory) channels mapped to the L3 channels. The L2 level of cache is coherent within a single chiplet and the L3 level (or other last level) of cache is unified and coherent across all of the GPU chiplets **106**.

[0023] A graphics data fabric **314** (GDF) of each GPU chiplet **106** connects all of the L1 cache memories **306** to each of the channels of the L2 cache memory **308**, thereby allowing each of the workgroup processors **302** and fixed function blocks **304** to access data stored in any bank of the L2 cache memory **308**. Each GPU chiplet **106** also includes a scalable data fabric **316** (SDF) (also known as a SOC memory fabric) that routes across the graphics core (GC) and system on chip (SOC) IP cores to the passive crosslink **118**. The GC includes CUs/WGPs, fixed function graphics blocks, caches above L3, and the like. Portions of the GPU used for traditional graphics and compute (i.e., the GC) are differentiable from other portions of the GPU used for handling auxiliary GPU functionality such as video decode, display output, and various system supporting structures that are contained on the same die. The passive crosslink **118** routes to both local L3 cache memory **310** of a chiplet (e.g., of GPU chiplet **106-1**) and the L3 cache memory **310** of all other external GPU chiplets (e.g., GPU chiplet **106-2** in FIG. 3). In this manner, a memory address request is routed to the appropriate lanes on the passive crosslink **118** to access L3 cache memory **310** locally or at L3 cache memory **310** of a different GPU chiplet **106** (as discussed in more detail with respect to FIG. 5).

[0024] FIG. 4 is a block diagram illustrating a floor plan view of a GPU chiplet in accordance with some embodiments. The view **400** provides a floor plan view of GPU chiplet **106-1** of FIGS. 1 and 2. As previously described in more detail with respect to FIG. 3, the GPU chiplet **106-1** includes a plurality of workgroup processors **302** (WGP) and a plurality of fixed function blocks **304** (GFX). The GPU chiplet **106-1** also includes a hierarchical cache memory **402** (e.g., L1 cache memory **306**, L2 cache memory **308**, and L3 cache memory **310** of FIG. 3) and memory PHY **312**. The GPU chiplet **106-1** further includes a passive crosslink controller **404** and one or more passive crosslink PHY tiles, such as the four illustrated passive crosslink PHYs **406-1**,

406-2, **406-3**, and **406-4** (collectively, passive crosslink PHYs **406**), at a first corner (e.g., top-right corner of GPU chiplet **106-1** in FIG. 4).

[0025] The passive crosslink controller **404** connects to the last level cache (LLC) of the GPU chiplet **106-1** (e.g., L3 cache memory as discussed herein), and handles routing between the LLC and electrically-active portions of the logic of data fabric crossbars (e.g., SDF **316** of FIG. 3). The passive crosslink PHYs **406** (e.g., passive crosslink PHYs **406-1**, **406-2**, **406-3**, and **406-4**) include the wire transport of data across various GPU chiplets **106**. In particular, the passive crosslink PHYs **406** correspond to traces **206** of FIG. 2 that form dedicated communication channels between GPU chiplets **106** that the GPU chiplet **106-1** interconnects with.

[0026] In various embodiments, passive crosslink PHY **406-1** corresponds to a dedicated communications channel that routes to the local L3 cache memory **310** of GPU chiplet **106-1**. In contrast, passive crosslink PHY **406-2** corresponds to a dedicated communications channel that routes to the L3 cache memory **310** of an external GPU chiplet on a different chiplet die (e.g., GPU chiplet **106-2** of FIG. 1). That is, the dedicated communications channel of passive crosslink PHY **406-2** do not communicate with any chiplet other than GPU chiplet **106-2**. Similarly, passive crosslink PHY **406-3**, **406-4** correspond to dedicated communications channels that route to the L3 cache memory **310** of GPU chiplets **106-3** and **106-4**, respectively. That is, the dedicated communications channels of passive crosslink PHY **406-3**, **406-4** do not communicate with any chiplet other than GPU chiplets **106-3** and **106-4**, respectively.

[0027] In some embodiments, the GPU chiplet **106-1** further includes an optional (as indicated by the dotted lines) second set of crosslink PHYs **408** at a second corner of the GPU chiplet **106-1** for communicating with additional GPU chiplets **106**. The passive crosslink **118** thus operates as an extension cord between routing fabric of two or more dies and provides coherent L3 memory access with uniform memory access behavior (or mostly uniform memory access behavior). Those skilled in the art will recognize that the performance of a processing system generally scales linearly based on the number of GPU chiplets utilized by nature of physical duplication (e.g., as the number of GPU chiplets increases, so does the number of memory PHYs **312**, WGPs **302**, and the like).

[0028] Referring now to FIG. 5, illustrated is a block diagram of a processing system utilizing a four-chiplet configuration in accordance with some embodiments. The processing system **500** is similar to the processing system **100** of FIG. 1 but omits certain elements for ease of illustration. As illustrated, the system **500** includes the CPU **102** and four GPU chiplets, such as the illustrated GPU chiplets **106-1**, **106-2**, **106-3**, and **106-4**. The CPU **102** communicates with GPU chiplet **106-1** via the bus **108**. With reference back to FIG. 4, the passive crosslink PHY **406-1** corresponds to a dedicated communications channel that routes to the local L3 cache memory **310** of GPU chiplet **106-1** (not shown). Passive crosslink PHY **406-2** corresponds to a dedicated communications channel that routes to the L3 cache memory **310** of GPU chiplet **106-2** (labeled as signal route **502** in FIG. 5). Passive crosslink PHY **406-3** corresponds to a dedicated communications channel that routes to the L3 cache memory **310** of GPU chiplet **106-3** (labeled as signal route **504** in FIG. 5). Passive crosslink

PHY 406-4 corresponds to a dedicated communications channel that routes to the L3 cache memory 310 of GPU chiplet 106-4 (labeled as signal route 506 in FIG. 5).

[0029] As a general operational overview, the processing system 500 utilizes a master-slave topology in which a single GPU chiplet in direct communication with the CPU 102 (i.e., GPU chiplet 106-1) is designated as the master chiplet (hereinafter, primary GPU chiplet or host GPU chiplet). The other GPU chiplets communicate with the CPU 102 indirectly via the passive crosslink 118 and are designated as the slave chiplets (hereinafter, secondary GPU chiplet(s)). Accordingly, the primary GPU chiplet 106-1 serves as the singular entry point from the CPU 102 to the entire GPU chiplet array.

[0030] As illustrated in FIG. 5, in one example, the CPU 102 transmits an access request (e.g., read request) for memory address XYZ to the primary GPU chiplet 106-1. The passive crosslink controller 404 determines that data associated with memory address XYZ is cached in L3 cache memory 310 of secondary GPU chiplet 106-4. Based on that determination, the access request is routed via signal route 506 of the passive crosslink 118 to secondary GPU chiplet 106-4. The secondary GPU chiplet 106-4 returns the result back to primary GPU chiplet 106-1, which subsequently returns the requested data to the originating requestor (i.e., CPU 102). In this manner, the CPU 102 only has a single external view and does not require direct communications to two or more GPU chiplets 106 via bus 108.

[0031] Those skilled in the art will recognize that although FIG. 5 is described here in the specific context of square GPU chiplet dies and a corner positions for the passive crosslink, various other configurations, die shapes, and geometries may be utilized in various embodiments without departing from the scope of this disclosure. For example, in some embodiments, the GPU chiplets may be constructed as pentagon-shaped dies such that five GPU chiplets may be coupled together in a chiplet array. In other embodiments, GPU chiplets may include passive crosslinks at two or more corners of a square GPU chiplet (e.g., optional second set of crosslink PHYs 408 of FIG. 4) such that multiple GPU chiplets may be tiled together in a chiplet array. Similarly, in other embodiments, GPU chiplets may include passive crosslinks spanning an entire side of a square GPU chiplet such that multiple GPU chiplets may be strung together in a long row/column configuration to an intervening passive crosslink.

[0032] FIG. 6 is a flow diagram illustrating a method 600 of performing inter-chiplet communications in accordance with some embodiments. At block 602, a primary GPU chiplet of a GPU chiplet array receives a memory access request from a requesting CPU. For example, with reference to FIG. 5, the primary GPU chiplet 106-1 receives an access request for memory address XYZ from the CPU 102. In some embodiments, the primary GPU chiplet 106-1 receives the access request at its scalable data fabric 316 via the bus 108.

[0033] At block 604, the primary GPU chiplet 106-1 identifies a caching chiplet corresponding to the GPU chiplet at which the requested data is cached (also referred to interchangeably as the “secondary chiplet”). For example, with reference to FIG. 5, a passive crosslink controller 404 of the primary GPU chiplet 106-1 determines that data associated with memory address XYZ is cached in L3 cache memory 310 of secondary GPU chiplet 106-4. In some

embodiments, the memory address range is address-sliced across the multiple GPU chiplets 106. In other embodiments, the processing system 100 utilizes other addressing topologies (e.g., flat address partitioning, addressing based on page settings from virtual to physical address translation, and the like). If the requesting data is not cached in the L3 of the secondary chiplet (i.e., the caching chiplet responsible for caching data associated with memory address XYZ), the memory access request is treated as an L3 miss and the secondary chiplet fetches the requested data from GDDR memory attached to that secondary chiplet.

[0034] At block 606, the primary GPU chiplet 106-1 routes the memory access request to the caching chiplet corresponding to the GPU chiplet at which the requested data is cached based on determination of block 604 and via the passive crosslink 118. For example, with reference to FIG. 5, the access request is routed via signal route 506 of the passive crosslink 118 to secondary GPU chiplet 106-4. In some embodiments, routing the memory access request includes the scalable data fabric 316 communicating with the passive crosslink 118, the scalable data fabric 316 requesting a data associated with the memory access request from the caching chiplet (e.g., secondary GPU chiplet 106-4).

[0035] In other embodiments, after determining that the requested data is cached locally at the L3 cache 310 of the primary GPU chiplet 106-1, the scalable data fabric 316 routes the access request via the passive crosslink 118 to the L3 cache 310 of the primary GPU chiplet 106-1. For example, with reference to FIG. 4, the scalable data fabric 316 routes the memory access request via passive crosslink PHY 406-1 corresponding to a dedicated communications channel between the local L3 cache memory 310 of GPU chiplet 106-1 and the passive crosslink 118.

[0036] At block 608, the caching chiplet returns data corresponding to the memory access request back to the primary GPU chiplet via the passive crosslink 118. For example, with reference to FIG. 5, the secondary GPU chiplet 106-4 returns the result back to primary GPU chiplet 106-1. In particular, the return communication is routed via the same signal route 506 of the passive crosslink 118 from which the memory access request was routed in block 606. Similarly, with reference to FIG. 4, the return communication is routed via passive crosslink PHY 406-1 corresponding to a dedicated communications channel between the local L3 cache memory 310 of GPU chiplet 106-1 and the passive crosslink 118 when the caching chiplet is the same as the primary GPU chiplet. In other embodiments, the request data port and the return data port do not share the same physical route.

[0037] At block 610, the primary GPU chiplet returns the requested data to the originating requestor (i.e., CPU 102) via the bus 108. In some embodiments, returning the requested data to the CPU 102 includes receiving the requested data from the caching chiplet at the scalable data fabric 316 of the primary GPU chiplet (i.e., GPU chiplet 106-1) and transmitting the requested data to the CPU 102 via bus 108.

[0038] Accordingly, as discussed herein, a passive die interposer deploys monolithic GPU functionality using a set of interconnected GPU chiplets in a manner that makes the chiplet implementation appear as a traditional monolithic GPU from a programmer model/developer perspective. The scalable data fabric of one GPU chiplet is able to access the

lower level cache(s) on other GPU chiplets in nearly the same time as to access the lower level cache on its same chiplet, and thus allows the GPU chiplets to maintain cache coherency without requiring additional inter-chiplet coherency protocols. This low-latency, inter-chiplet cache coherency in turn enables the chiplet-based system to operate as a monolithic GPU from the software developer's perspective, and thus avoids chiplet-specific considerations on the part of a programmer or developer.

[0039] A computer readable storage medium may include any non-transitory storage medium, or combination of non-transitory storage media, accessible by a computer system during use to provide instructions and/or data to the computer system. Such storage media can include, but is not limited to, optical media (e.g., compact disc (CD), digital versatile disc (DVD), Blu-Ray disc), magnetic media (e.g., floppy disc, magnetic tape, or magnetic hard drive), volatile memory (e.g., random access memory (RAM) or cache), non-volatile memory (e.g., read-only memory (ROM) or Flash memory), or microelectromechanical systems (MEMS)-based storage media. The computer readable storage medium may be embedded in the computing system (e.g., system RAM or ROM), fixedly attached to the computing system (e.g., a magnetic hard drive), removably attached to the computing system (e.g., an optical disc or Universal Serial Bus (USB)-based Flash memory), or coupled to the computer system via a wired or wireless network (e.g., network accessible storage (NAS)).

[0040] In some embodiments, certain aspects of the techniques described above may implemented by one or more processors of a processing system executing software. The software includes one or more sets of executable instructions stored or otherwise tangibly embodied on a non-transitory computer readable storage medium. The software can include the instructions and certain data that, when executed by the one or more processors, manipulate the one or more processors to perform one or more aspects of the techniques described above. The non-transitory computer readable storage medium can include, for example, a magnetic or optical disk storage device, solid state storage devices such as Flash memory, a cache, random access memory (RAM) or other non-volatile memory device or devices, and the like. The executable instructions stored on the non-transitory computer readable storage medium may be in source code, assembly language code, object code, or other instruction format that is interpreted or otherwise executable by one or more processors.

[0041] Note that not all of the activities or elements described above in the general description are required, that a portion of a specific activity or device may not be required, and that one or more further activities may be performed, or elements included, in addition to those described. Still further, the order in which activities are listed are not necessarily the order in which they are performed. Also, the concepts have been described with reference to specific embodiments. However, one of ordinary skill in the art appreciates that various modifications and changes can be made without departing from the scope of the present disclosure as set forth in the claims below. Accordingly, the specification and figures are to be regarded in an illustrative rather than a restrictive sense, and all such modifications are intended to be included within the scope of the present disclosure.

[0042] Benefits, other advantages, and solutions to problems have been described above with regard to specific embodiments. However, the benefits, advantages, solutions to problems, and any feature(s) that may cause any benefit, advantage, or solution to occur or become more pronounced are not to be construed as a critical, required, or essential feature of any or all the claims. Moreover, the particular embodiments disclosed above are illustrative only, as the disclosed subject matter may be modified and practiced in different but equivalent manners apparent to those skilled in the art having the benefit of the teachings herein. No limitations are intended to the details of construction or design herein shown, other than as described in the claims below. It is therefore evident that the particular embodiments disclosed above may be altered or modified and all such variations are considered within the scope of the disclosed subject matter. Accordingly, the protection sought herein is as set forth in the claims below.

What is claimed is:

1. A system, comprising:

a central processing unit (CPU) communicably coupled to a first graphics processing unit (GPU) chiplet of a GPU chiplet array, wherein the GPU chiplet array includes: the first GPU chiplet communicably coupled to the CPU via a bus; and

a second GPU chiplet communicably coupled to the first GPU chiplet via a passive crosslink, wherein the passive crosslink is dedicated for inter-chiplet communications.

2. The system of claim 1, wherein the passive crosslink comprises a passive interposer die.

3. The system of claim 1, wherein the first GPU chiplet comprises a first PHY region including conductor structures for chiplet-to-chiplet communications, and further wherein the second GPU chiplet comprises a second PHY region including conductor structures for chiplet-to-chiplet communications.

4. The system of claim 3, further comprising:

a third GPU chiplet communicably coupled to the first GPU chiplet via the passive crosslink, wherein the passive crosslink is dedicated for inter-chiplet communications, wherein the third GPU chiplet comprises a third PHY region including conductor structures for chiplet-to-chiplet communications.

5. The system of claim 4, wherein the first PHY region of the first GPU chiplet comprises a first passive crosslink PHY including conductor traces solely for communications between the passive crosslink and a last level cache of the first GPU chiplet.

6. The system of claim 4, wherein:

the second PHY region of the second GPU chiplet comprises a second passive crosslink PHY including conductor traces solely for communications between the passive crosslink and a last level cache of the second GPU chiplet; and

the third PHY region of the third GPU chiplet comprises a third passive crosslink PHY including conductor traces solely for communications between the passive crosslink and a last level cache of the third GPU chiplet.

7. The system of claim 1, wherein the passive crosslink communicably couples all GPU chiplets in the GPU chiplet array.

- 8. The system of claim 1, further comprising:
a first cache memory hierarchy at the first GPU chiplet, wherein a first level of the first cache memory hierarchy is coherent within the first GPU chiplet; and
a second cache memory hierarchy at the second GPU chiplet, wherein a first level of the second cache memory hierarchy is coherent within the second GPU chiplet.
- 9. The system of claim 8, further comprising:
a unified cache memory including both a last level of the first cache memory hierarchy and a last level of the second cache memory hierarchy, wherein the unified cache memory is coherent across all chiplets of the GPU chiplet array.
- 10. The system of claim 1, further comprising:
a plurality of conductive pillars coupling a circuit board substrate to a first non-PHY region of the first GPU chiplet and a second non-PHY region of the second GPU chiplet.
- 11. A method, comprising:
receiving, at a first GPU chiplet of a GPU chiplet array, a memory access request from a central processing unit (CPU);
determining, at a passive crosslink controller of the first GPU chiplet, a caching GPU chiplet corresponding to a location at which data associated with the memory access request is stored;
routing, via a passive crosslink dedicated for inter-chiplet communications in the GPU chiplet array, the memory access request to a last level cache of the caching GPU chiplet; and
returning data associated with the memory access request to the CPU.
- 12. The method of claim 11, wherein routing the memory access request further includes a scalable data fabric requesting data associated with the memory access request from the caching GPU chiplet.
- 13. The method of claim 11, wherein routing the memory access request to the last level cache of the caching GPU chiplet further comprises:
routing, based on determining the first GPU chiplet is the caching GPU chiplet, the memory access request via a first passive crosslink PHY including conductor traces solely for communications between the passive crosslink and the last level cache of the first GPU chiplet.
- 14. The method of claim 11, wherein routing the memory access request to the last level cache of the caching GPU chiplet further comprises:
routing, based on determining a second GPU chiplet is the caching GPU chiplet, the memory access request via a second passive crosslink PHY including conductor traces solely for communications between the passive crosslink and the last level cache of the second GPU chiplet.

- 15. The method of claim 11, further comprising:
returning data associated with the memory access request to the first GPU chiplet via a passive crosslink PHY including conductor traces solely for communications between the passive crosslink and the caching GPU chiplet.
- 16. A non-transitory computer readable medium embodying a set of executable instructions, the set of executable instructions to manipulate at least one processor to:
receive, at a first GPU chiplet of a GPU chiplet array, a memory access request from a central processing unit (CPU);
determine, at a passive crosslink controller of the first GPU chiplet, a caching GPU chiplet corresponding to a location at which data associated with the memory access request is stored;
route, via a passive crosslink dedicated for inter-chiplet communications in the GPU chiplet array, the memory access request to a last level cache of the caching GPU chiplet; and
return data associated with the memory access request to the CPU.
- 17. The non-transitory computer readable medium of claim 16, the set of executable instructions further to manipulate at least one processor to: request, via a scalable data fabric, data associated with the memory access request from the caching GPU chiplet.
- 18. The non-transitory computer readable medium of claim 16, the set of executable instructions further to manipulate at least one processor to:
route, based on determining the first GPU chiplet is the caching GPU chiplet, the memory access request via a first passive crosslink PHY including conductor traces solely for communications between the passive crosslink and the last level cache of the first GPU chiplet.
- 19. The non-transitory computer readable medium of claim 16, the set of executable instructions further to manipulate at least one processor to:
route, based on determining a second GPU chiplet is the caching GPU chiplet, the memory access request via a second passive crosslink PHY including conductor traces solely for communications between the passive crosslink and the last level cache of the second GPU chiplet.
- 20. The non-transitory computer readable medium of claim 16, the set of executable instructions further to manipulate at least one processor to:
return data associated with the memory access request to the first GPU chiplet via a passive crosslink PHY including conductor traces solely for communications between the passive crosslink and the caching GPU chiplet.

* * * * *