

EXHIBIT 803-B

COMPILATION OF DISCLOSURES FROM ADDITIONAL PRIOR ART REFERENCES

The references listed below anticipate and/or render obvious (alone or in combination with other references), including at least under Complainant's apparent infringement theory claims 1-3, 7-10 of U.S. Patent No. 11,841,803 ("the '803 patent") under 35 U.S.C. §§ 102 and/or 103.

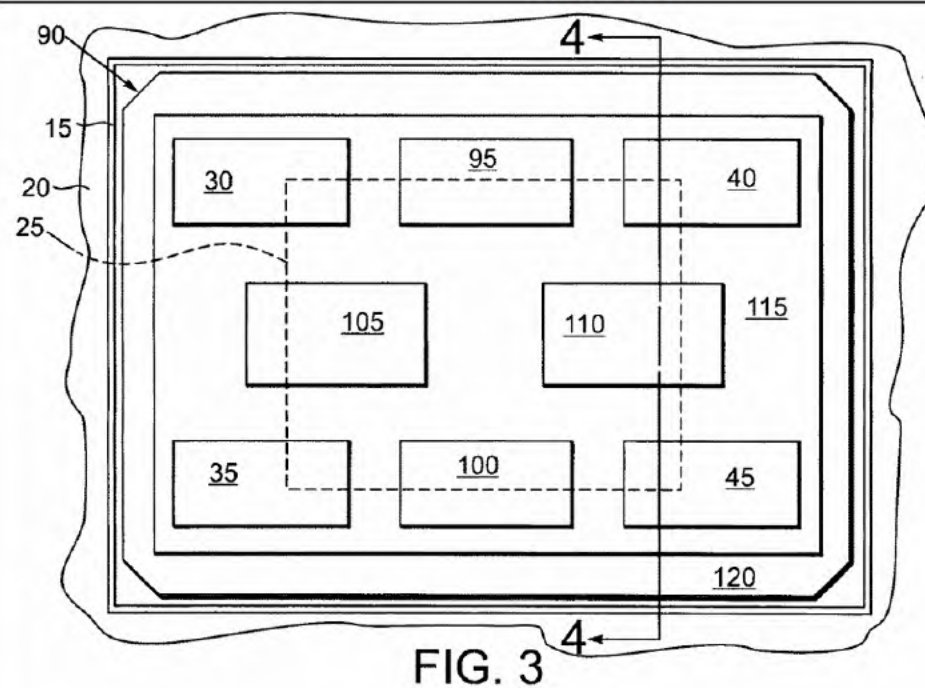
- U.S. Patent No. 9,170,948 to Loh ("Loh-948")
- U.S. Patent No. 10,714,462 to Bhagavat ("Bhagavat")
- U.S. Patent No. 11,342,320 to Elsherbini ("Elsherbini")
- EP Patent No. 2,936,554 to Loh ("Loh-554")
- U.S. Patent Application Publication No. 2017/0365580 to Shih ("Shih")
- U.S. Patent Application Publication No. 2018/0240778 to Liu ("Liu")
- U.S. Patent Application Publication No. 2020/0098724 to Liff ("Liff")
- U.S. Patent Application Publication No. 2009/0037658 to Sistla ("Sistla")
- U.S. Patent Application Publication No. 2019/0026228 to Jiang ("Jiang")
- Interconnect-Memory Challenges for Multi-chip, Silicon Interposer Systems ("Loh") (NVIDITC-000018921)
- Redistribution Layers (RDLs) for 2.5D/3D IC Integration ("Lau") (NVIDITC-000133153)
- Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency ("Saban") (NVIDITC-000134989)
- NoC Architectures for Silicon Interposer Systems ("Jerger") (NVIDITC-000019008)
- Enabling Interposer-Based Disintegration Of Multi-Core Processors ("Kannan") (NVIDITC-000018285)

NVIDIA has charted the portions of the references that correspond to each element, including at least under Complainant's apparent interpretation of the claims as reflected in Complainant's infringement contentions. Where the chart below states that a reference "discloses," "teaches," or "suggests" a claim limitation, such disclosure, teaching, or suggestion may be express, implied, or inherent. The cited evidence is merely illustrative, and NVIDIA reserves the right to cite alternative or additional evidence.

No part of this exemplary chart construes, or is intended to construe, the specification, file history, or claims of the asserted patents. In addition, nothing stated in this chart shall be treated as an admission or suggestion that NVIDIA agrees with Complainant regarding either the scope of any of the asserted claims or Complainant's apparent interpretation of the scope of the asserted claims, or that NVIDIA's accused products meet any limitations of the claims.

To the extent that Complainant contends that any of the references listed above does not disclose one or more limitations of the asserted claims, it would have been obvious to combine the teachings of the references with (1) the knowledge of one of ordinary skill in the art, and/or (2) the teachings of any of the prior art set forth in NVIDIA's other invalidity charts and cover pleading, to show all the limitations of the asserted claims, including for the reasons set forth in NVIDIA's Invalidity Contention Cover Pleading, which are incorporated herein.

U.S. Patent No. 11,841,803	References
1[Pre]: A system, comprising:	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <i>See evidence cited for '803 patent claims 1-3, 7-10.</i> This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
1[A]: a central processing unit (CPU) communicably coupled to a first graphics processing unit (GPU) chiplet of a GPU array,	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <u>Bhagavat</u> Bhagavat at Fig. 3



Bhagavat at Fig. 4

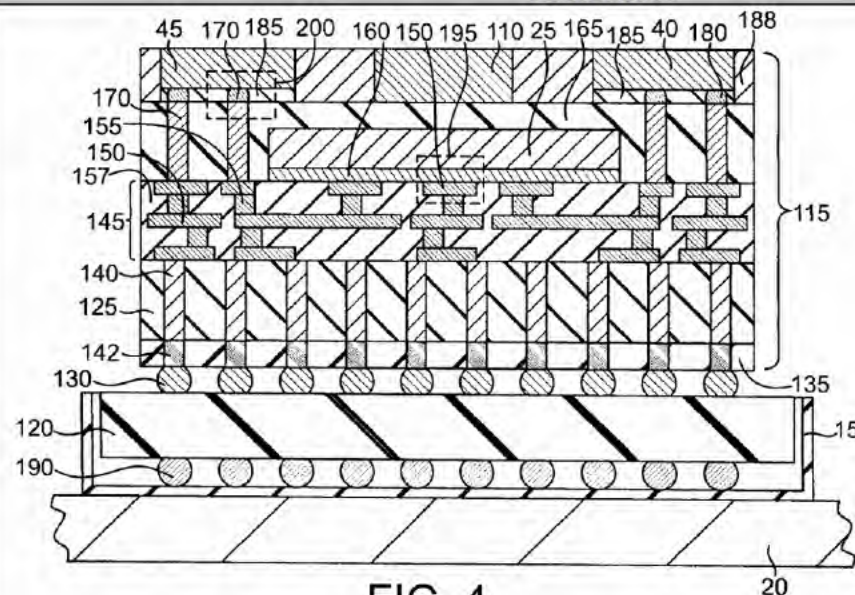
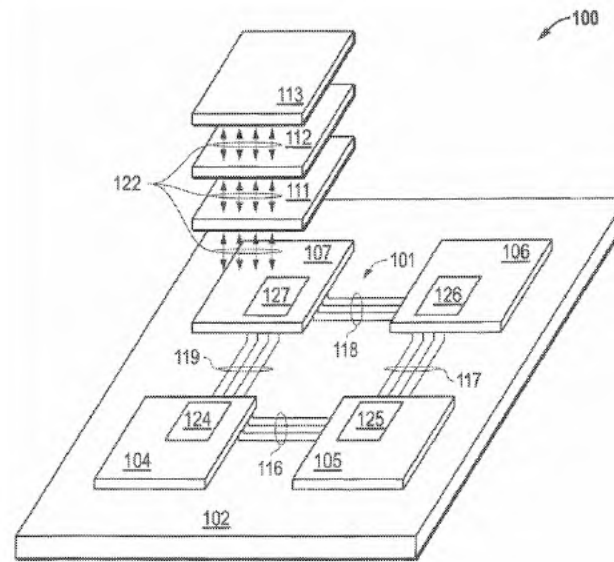


FIG. 4

Bhagavat at 2:28-58 (“One conventional multi-chip module variant includes side-by-side processor and memory chips in a 2.5D arrangement on a silicon interposer that is, in-turn, mounted on a package substrate. The conventional package substrate is manufactured with a footprint and pin out tailored for a particular type of BGA socket. In other words, the size and shape of the package substrate is, to a large extent, dictated by the mechanical properties, size, etc. of the socket. Performance of a given multi-chip module can be often increased by incorporating additional memory chips into the module that cooperate with processor(s) or system-on-chip chips. However, packing additional memory chips into a multi-chip module designed for a standard socket and attendant package substrate, is a technical challenge. One conventional solution is to simply increase the size of both the interposer and the package. Of course this technique almost always involves a redesign of the socket, which will require the redesign and configuration of the multitudes of different types of circuit boards that use the originally adopted standard socket.”)

U.S. Patent No. 11,841,803	References
	The disclosed embodiments solve the issue of incorporating additional chips in a multi-chip module by stacking the somewhat smaller chips over a larger processor or other type of integrated circuit and at least partially laterally overlapping the upper chips with the lower chip all without having to substantially change the physical footprints of the underlying interposer and the package substrate. In this way, additional memory devices or other types of integrated circuits can be incorporated into a multi chip module while preserving the ability to use existing socket sizes and designs.”). Bhagavat at 4:60-5:4 (“The semiconductor chips 25, 30, 35, 40, 45, 95 and 100 can be any of a variety of integrated circuits. A non-exhaustive list of examples includes microprocessors, graphics processing units, application processing units that combines aspects of both, memory devices, an application integrated specific circuit or other. In one arrangement, the semiconductor chip 25 can be a processor and the semiconductor chips 30, 35, 40, 45, 95 and 100 can be memory chips, such as DRAM, SRAM or other. The circuit board 120 can be organic or ceramic and single, or more commonly, multilayer. Variations include package substrates, system boards, daughter boards, circuit cards and others.”). <u>Loh-554</u> Loh-554 at Fig. 1

**FIG. 1**

Loh-554 at Fig. 3

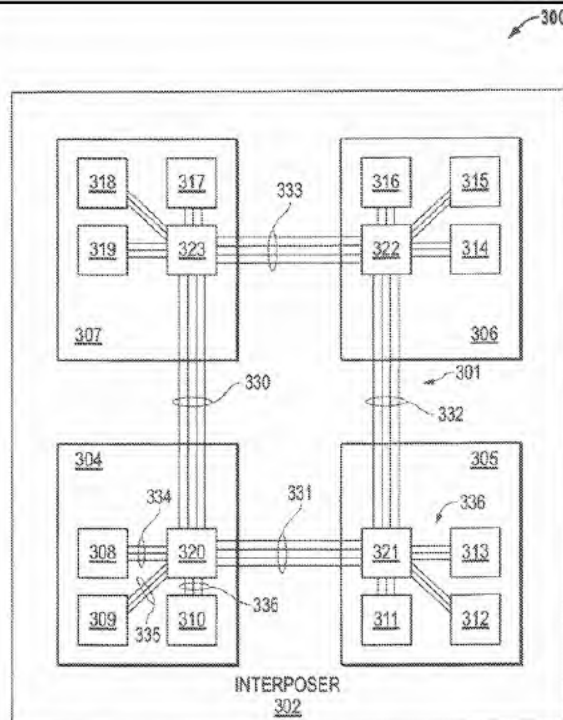


FIG. 3

Loh 554 at [0006]-[0007] (“FIG. 1 illustrates die-stacked processing system 100 employing a multi-hop network comprising a router partition and a link partition in accordance with some embodiments. The processing system 100 can comprise any of a variety of computing systems, including a notebook or tablet computer, a desktop computer, a server, a network router, switch, or hub, a computing-enabled cellular phone, a personal digital assistant, and the like. The example processing system 100 includes a plurality of horizontally-stacked die 104, 105, 106, and 107 disposed at a surface of an interposer 102. The die 107 is the lowest layer of a vertical die stack 110 further including die 111, 112, and 113.

U.S. Patent No. 11,841,803	References
	The illustrated die 104–107 and 111–113 may include any variety of processor cores and combinations thereof, such as central processing units (CPU), graphics processing units (GPU), digital signal processors (DSP), and the like. The illustrated die may also implement any variety of storage devices including, but not limited to, memory architectures such as dynamic random access memory (DRAM), static random access memory (SRAM), read-only memory (ROM), flash memory, ferroelectric RAM (F-RAM), magneto-resistive RAM (MRAM), and the like. The illustrated die 104–107 and 126–128 can also include any peripheral devices such as northbridge and southbridge functions, input/output controllers, network interfaces and the like. The processing system 100 also can include a variety of other components not illustrated in FIG. 1, such as one or more external interfaces to display components, storage devices, input devices (e.g., a mouse or keyboard), and the like.”). Loh-554 at [0028] (“FIG. 3 illustrates a plan view of another example processing system 300 implementing a partitioned multi-hop network 301 in accordance with some embodiments. In this example, the partitioned multi-hop network 301 implements a link partition comprising a combination of intra-die links together with inter-die links. As illustrated, the processing system 300 comprises horizontally stacked die 304, 305, 306, and 307 disposed at a surface of an interposer 302. Each die implements a plurality of devices, such as devices 308, 309, and 310 at die 304, devices 311, 312, and 313 at die 305, devices 314, 315, and 316 at die 306, and devices 317, 318, and 319 at die 307. These devices may include, but are not limited to CPUs, GPUs, DSPs, memory controllers, input/output controllers, storage devices, and the like.”). <u>Shih</u> Shih at Abstract (“A semiconductor package may include a first logic die and a second logic die located laterally adjacent to the first logic die. A bridge memory die may be coupled to both the first logic die and the second logic die on a first active face of the first logic die and a second active face of the second logic die. A redistribution layer (RDL) structure may be coupled to the first logic die, the second logic die, and the bridge memory die. The bridge memory die may be interposed between at least a portion of the first logic die and the RDL structure and between at least a portion of the second logic die and the RDL structure. A

U.S. Patent No. 11,841,803	References
	molding compound may at least partially encapsulate the first logic die, the second logic die, and the bridge memory die.”). Shih at [0006]-[0007] (“It is one object of the invention to provide an improved semiconductor package and fabrication methods thereof, which are capable of integrating multiple logic dies and dual-port RAM in one package. In one aspect, the present invention discloses a semiconductor package including a first logic die, a second logic die disposed in close proximity to the first logic die, a bridge memory die coupled to both the first logic die and the second logic die, a redistribution layer (RDL) structure coupled to the first logic die and the second logic die, and a molding compound at least partially encapsulating the first logic die, the second logic, and the bridge memory die.”) Shih at [0010]-[0011] (“According to one embodiment, the first logic die comprises central processing units, graphics processing units, or application processors. The second logic die comprises central processing units, graphics processing units, or application processors. According to one embodiment, the bridge memory die allows inter-processor communication between the first logic die and the second logic die.”). <u>Liu</u> Liu at Fig. 2

U.S. Patent No. 11,841,803	References
	MULTICHIP PACKAGE MAIN DIE (E.G., CPU, GPU, ASIC, PLD, ETC.) XCVR MEM ... FIG. 2 Liu at Fig. 3

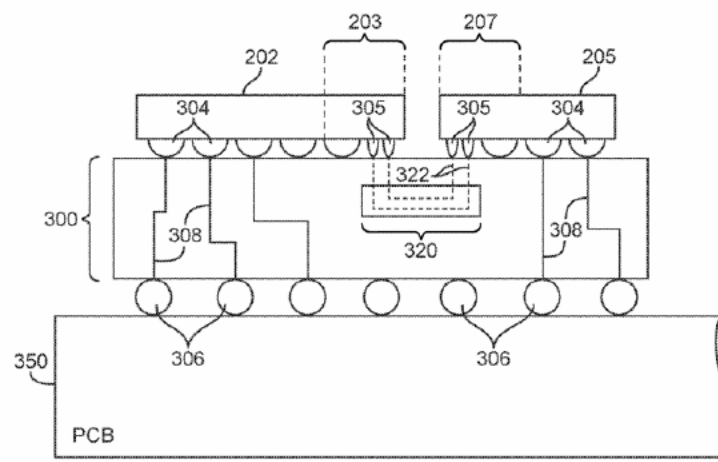
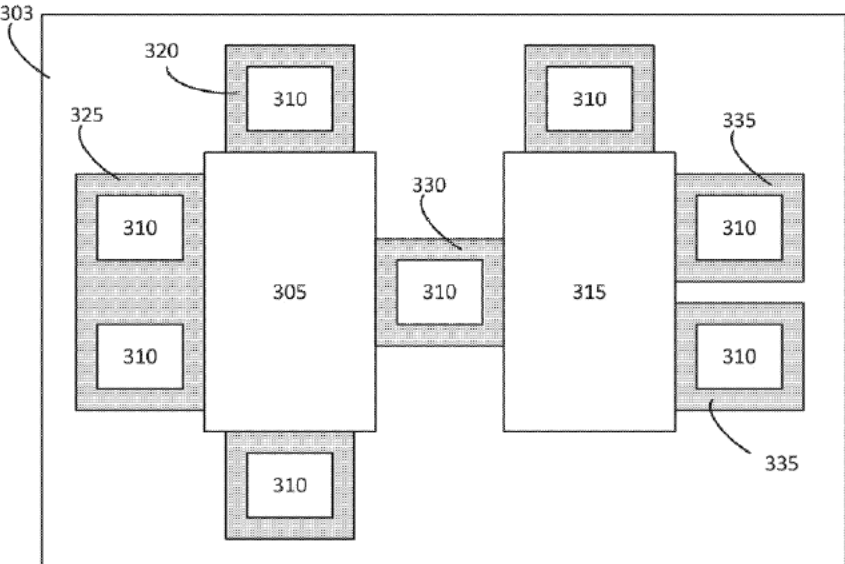


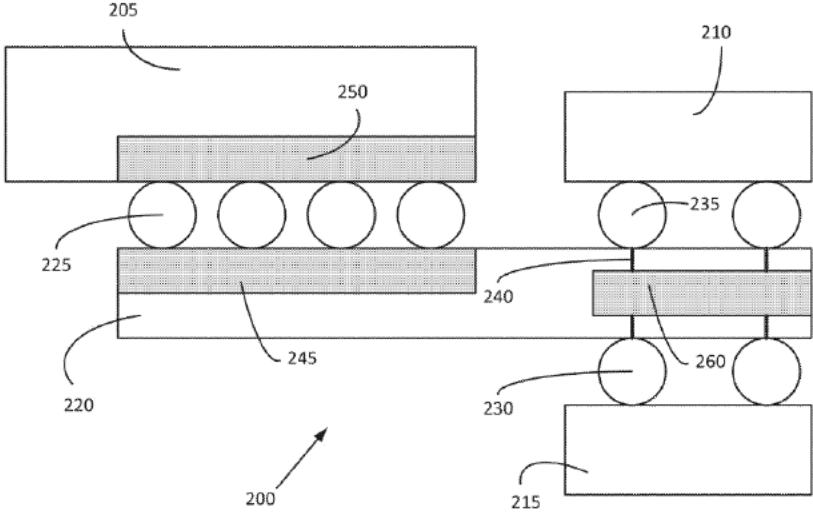
FIG. 3

Liu at [003] (“As demands on integrated circuit technology continue to outstrip even the gains afforded by ever decreasing device dimensions, more and more applications demand a packaged solution with more integration than possible in one silicon die. In an effort to meet this need, more than one die may be placed within a single integrated circuit package (i.e., a multichip package). As different types of devices cater to different types of applications, more dies may be required in some systems to meet the requirements of high performance applications. Accordingly, to obtain better performance and higher density, an integrated circuit package may include multiple dies arranged laterally along the same plane.”).

Liu at [0026]-[0027] (“As integrated circuit fabrication technology scales towards smaller process nodes, it becomes increasingly challenging to design an entire system on a single integrated circuit die (sometimes referred to as a system-on-chip). Designing analog and digital circuitry to support desired performance levels while minimizing leakage and power consumption can be extremely time consuming and costly.”).

U.S. Patent No. 11,841,803	References
	One alternative to single-die packages is an arrangement in which multiple dies are placed within a single package. Such types of packages that contain multiple interconnected dies may sometimes be referred to as systems-in-package (SiPs), multi-chip modules (MCM), or multichip packages. Placing multiple chips (dies) into a single package may allow each die to be implemented using the most appropriate technology process (e.g., a memory chip may be implemented using the 28 nm technology node, whereas the radio-frequency analog chip may be implemented using the 45 nm technology node), may increase the performance of die-to-die interface (e.g., driving signals from one die to another within a single package is substantially easier than driving signals from one package to another, thereby reducing power consumption of associated input-output buffers), may free up input-output pins (e.g., input-output pins associated with die-to-die connections are much smaller than pins associated with package-to-board connections), and may help simplify printed circuit board (PCB) design (i.e., the design of the PCB on which the multi-chip package is mounted during normal system operation.”). Liu at [0028] (“In order to facilitate communications between two chips on a multi-chip package, the package may include an embedded multi-die interconnect bridge (EMIB) that is designed and patented by INTEL Corporation. An EMIB is a small silicon die that is embedded in the underlying substrate of a multi-chip package and that offers dedicated ultra-high-density interconnection between dies within the package. EMIBs generally include wires of minimal length, which help to significantly reduce loading and directly boost performance.”). Liu at [0033] (“As shown in FIG. 2, a multi-chip package 200 may include a main die 202, a transceiver die 204, a memory die 206, and additional auxiliary dies 208. Main die 202, for example, may be a central processing unit (CPU), a graphics processing unit (GPU), an application-specific integrated circuit (ASIC), a programmable logic device (PLD), or any other desired processor or logic device. Secondary integrated circuit dies such as transceiver die 204, memory die 206, and auxiliary dies 208 may be coupled to main die 202 and may communicate with main die 202. Memory die 206, for example, may be an erasable-programmable read-only memory (EPROM) chip, a non-volatile memory (e.g., 3D XPoint) chip, a volatile memory (e.g., high bandwidth memory) chip, or any other suitable memory

U.S. Patent No. 11,841,803	References
	device. Auxiliary dies 208 may include additional memory dies, transceiver dies, programmable logic devices, and any other suitable integrated circuit devices.”). <u>Liff</u> Liff at Fig. 3  Figure 3 Liff at Fig. 2

U.S. Patent No. 11,841,803	References
	 Figure 2 Liff at Fig. 1

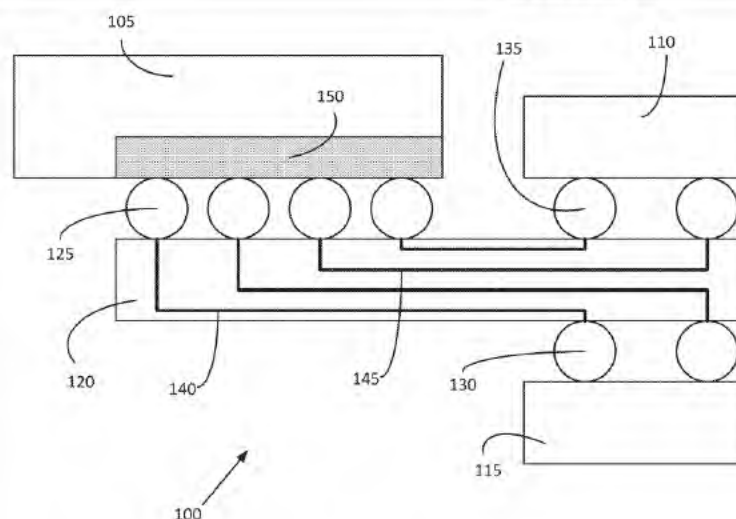


Figure 1

Liff at [0043] (“The user of an interposer such as interposers 120 or 220 may allow different possible structural arrangements to enable a network of interconnection such as is shown in FIG. 3 with respect to package 300 wherein the interposer may be used as a bridge between various structural elements to increase the computational performance of the package. Specifically, the package 300 may include a plurality of processing units 305 and 315, which may be similar to processing units 105 or 205. The package 300 may also include a plurality of interposers such as interposers 320/325/330/335, which may be similar to interposers 120 or 220. The package 300 may also include a plurality of memories 310, which may be similar to memories 115 or 215. In embodiments, the memories 310 may be positioned on a side of the interposers 320/325/330/335 that is opposite the side of the interposers to which the processing units 305/315 are coupled. In other embodiments, one or more of the memories 310 may be on the same side of the interposers 320/325/330/335 as the side to which the processing units 305/315 are coupled. In embodiments, one or more of the elements such as the interposer

U.S. Patent No. 11,841,803	References
	320/325/330/335, memories 310, or processing units 305/315 may be coupled with a substrate 303. The substrate may be, for example, a substrate of a printed circuit board (PCB), an interposer, or some other type of substrate.”). Liff at [0044] (“As can be seen, one of the interposers such as interposer 330 may be communicatively and physically coupled with both processing units 305 and 315. This coupling with two processing units may be referred to as “stitching” the processing units together. Some processing units, e.g., processing unit 305, may have interposers on each side of the processing unit. More specifically, the interposers 320/325/335 may extend from the processing units 305/315 in a cantilevered fashion. Other processing units, e.g., processing unit 315, may not have an interposer coupled with one or more sides of the processing units. In some embodiments, a processing unit such as processing unit 315 may have two interposers 335 coupled with a single side of the processing unit, and each of the interposers 335 may have a single memory 310 attached thereto. By contrast, a processing unit such as processing unit 305 may be coupled with an interposer such as interposer 325 that has two memories 310 attached thereto.”). Liff at Claim 1 (“A semiconductor package structure comprising: an interposer with a first side and a second side opposite the first side; a memory communicatively and physically coupled with the first side of the interposer; a processing unit communicatively and physically coupled with the second side of the interposer, wherein the processing unit is communicatively coupled, by the interposer, with the memory; and a third chip coupled with the second side of the interposer.”). <u>Jerger</u> Jerger at Fig. 1

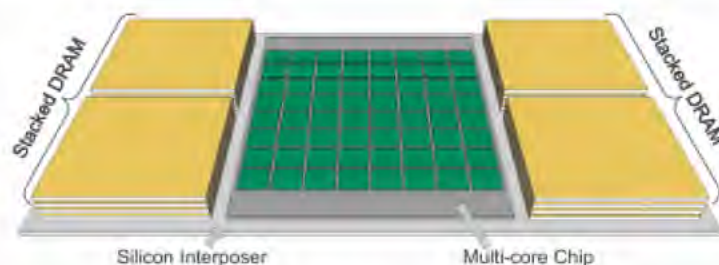
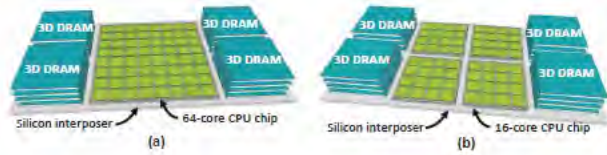


Fig. 1. Baseline system organization consisting of a multi-core chip horizontally stacked with four 3D DRAM stacks on a silicon interposer.

Jerger at § I (“Die-stacking technology enables the combination of multiple distinct silicon chips within a single package. In the past several years, the industrial adoption of die-stacking technologies has been accelerating [9]. Vertical or 3D stacking takes multiple silicon die and places one on top of the other with through-silicon vias (TSVs) providing inter-layer connectivity. Another die-stacking approach takes multiple silicon die, and “stacks” them side-by-side on a silicon interposer carrier, for example as shown in Fig. 1. Known as “2.5D stacking” [15], this technology is already supported by design tools [22], planned for future GPU designs [18], and is already in some commercially-available products [45], [46]. Section II provides more details and a comparison of the approaches.”).

Loh

Loh at § 3.1 (“While the previous examples have illustrated a single monolithic multi-core chip stacked on an interposer, another approach is to stack multiple processing chips on the interposer. For example, instead of one 64-core chip, four smaller 16-core chips could be mounted to provide the same compute capability as shown in Fig. 4(b). The smaller chips would likely be cheaper due to better yield (assuming adequate known-good-die testing prior to stacking on the interposer). This, however, further complicates the design of the NoC, as now even some core-to-core communications must navigate their way across the interposer. This can lead to asymmetric NoC topologies, such as that shown in Fig. 4(b), with more complex routing and greater susceptibility to network hotspots.

U.S. Patent No. 11,841,803	References
	While all four processing die in Fig. 4(b) may be implemented in the same process technology, die-to-die parametric variations may necessitate additional timing and/or voltage margins leading to less efficient or lower-performance operation.”). <u>Kannan</u> Kannan at Fig. 1  Figure 1: (a) An example interposer-based system integrating a 64-core processor chip with four 3D stacks of DRAM, (b) and a 64-core system composed of four 16-core processor chips. Kannan at Abstract (“Silicon interposers enable the integration of multiple stacks of in-package memory to provide higher bandwidth or lower energy for memory accesses. Once the interposer has been paid for, there are new opportunities to exploit the interposer. Recent work considers using the routing resources of the interposer to improve the network-on-chip's (NoC) capabilities. In this work, we exploit the interposer to "disintegrate" a multi-core CPU chip into smaller chips that individually and collectively cost less to manufacture than a single large chip. However, this fragments the overall NoC, which decreases performance as core-to-core messages between chips must now route through the interposer. We study the performance-cost trade-offs of interposer based, multi-chip, multi-core systems and propose new interposer NoC organizations to mitigate the performance challenges while preserving the cost benefits.”).

Sistla

Sistla at Fig. 1

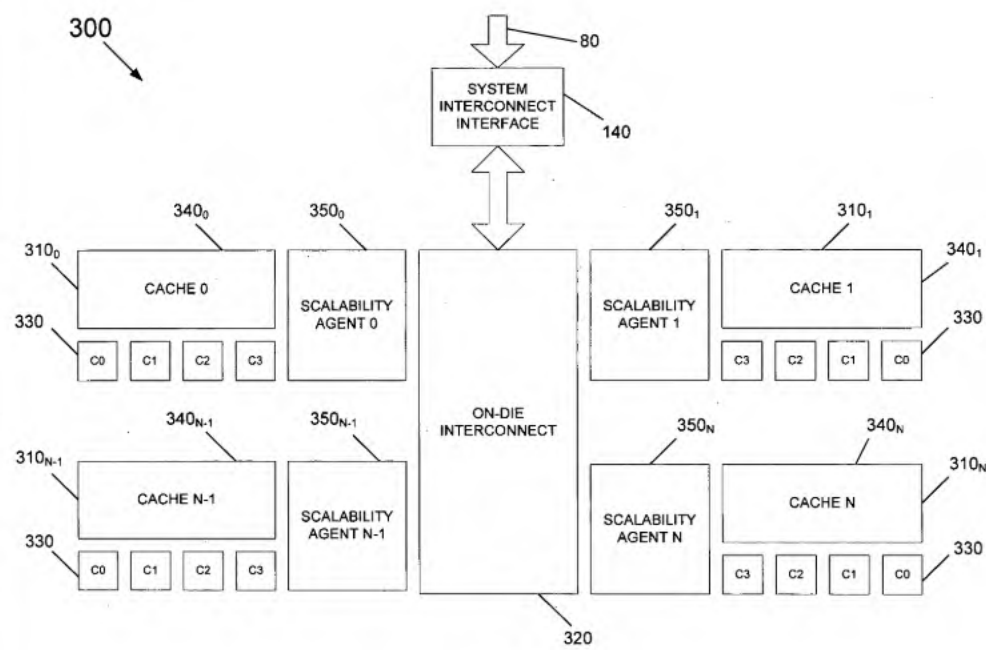


Fig. 1

Sistla at Fig. 2

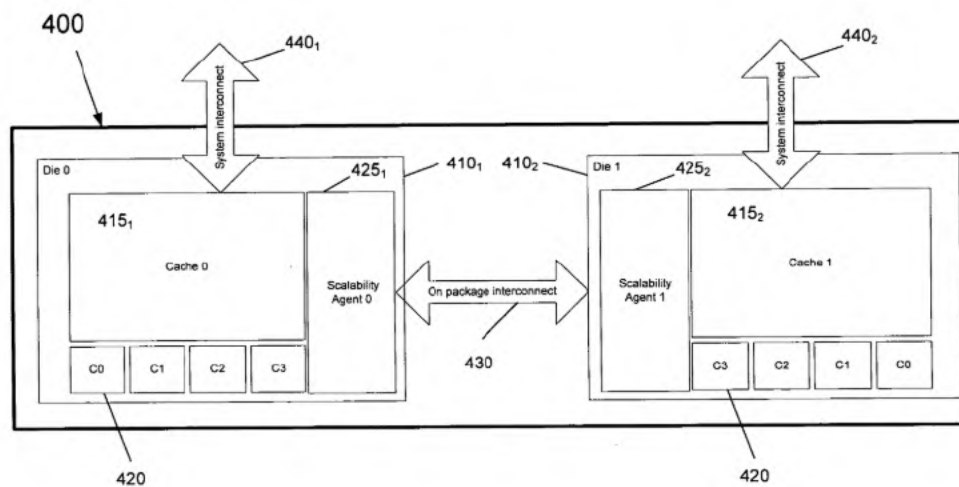


FIG. 2

Sistla at Fig. 5

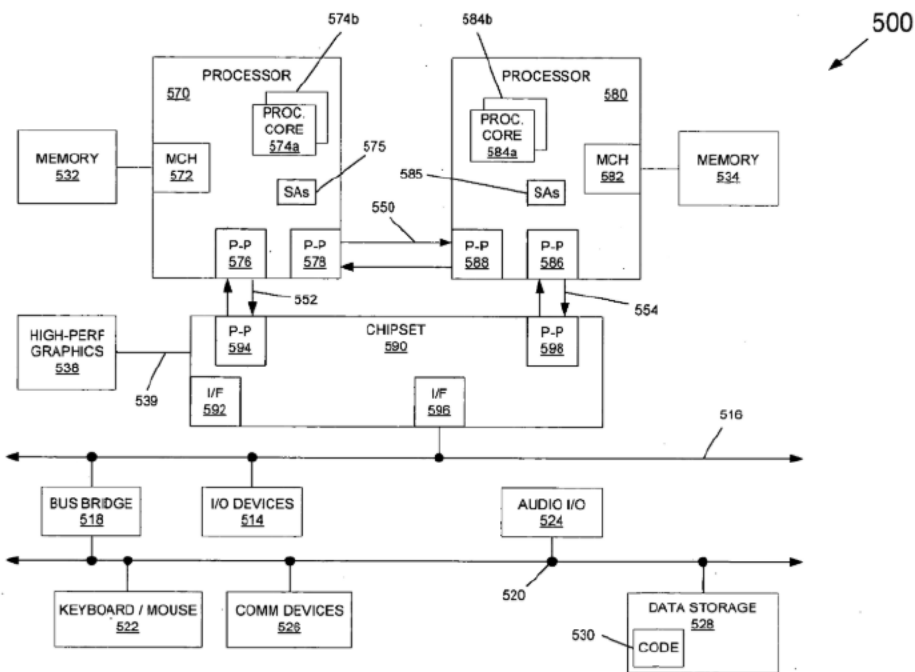


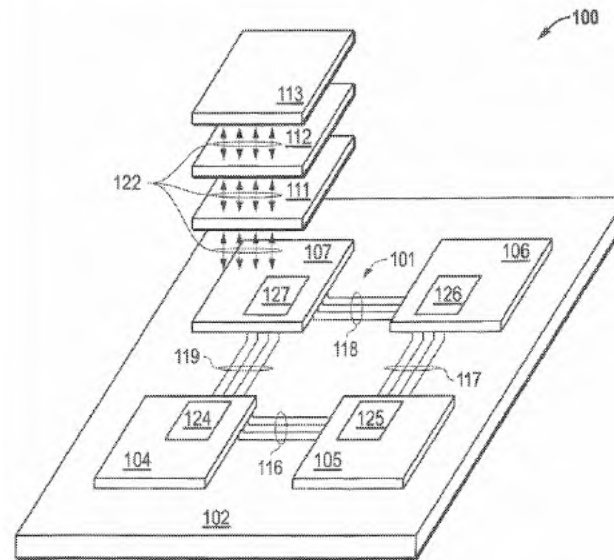
FIG. 5

Sistla at [0001] (“Microprocessors generally include a variety of logic circuits fabricated on a single semiconductor integrated circuit (IC). These logic circuits typically include a processor core, memory, and other components. Today’s high-end processors include multiple processor cores on the same IC. For instance, multi-core processors such as chip multi-processors (CMPs) feature a multi-core structure that implements multiple processor cores within an IC.”) Sistla at [0002] (“Increased silicon efficiencies are now providing new opportunities for adding additional functionality into the processor silicon. As an example, applications are taking advantage of increased multi-threading capabilities realized from an increased number of processing cores in the same processor. Performance of a multi-core processor may be optimized by mitigating system interconnect latency issues and ensuring that aggregated processor cores appear as one caching agent in order to avoid scalability issues and interconnect saturation. Systems that include caches each associated with multiple cores

U.S. Patent No. 11,841,803	References
	typically are structured as split caches in which each cache acts as an independent cache domain. This increases the total number of coherence misses, thus increasing the misses per instruction. Split-caches also lose capacity due to replication of shared code/data in all the independent caches.”) This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA’s other invalidity charts and cover pleading.
1[B]: wherein the GPU chiplet array includes: the first GPU chiplet communicably coupled to the CPU via a bus; and	Including at least under Complainant’s apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <u>Liff</u> Liff at [0045] (“Package 300 may enable a direct bus to the memories 310 from the processing units 305/315, and may be configurable to accommodate various memory heights that range from a single piece of memory to a memory stack. In this way when there is non-uniform memory access a hub of data may be stored between the two processing units 305 and 315.”). <u>Sistla</u> Sistla at [0043] (“Embodiments may be suited for CMP platforms to reduce traffic between the CMPs. As shown in FIG. 5, multiprocessor system 500 is a point-to-point interconnect system, and includes a first processor 570 and a second processor 580 coupled via a point-to-point interconnect 550, although the system may be of another bus architecture. As shown in FIG. 5, each of processors 570 and 580 may be multi-core processors including first and second processor cores (i.e., processor cores 574 a and 574 b and processor cores 584 a and 584 b), although other cores may be present. Furthermore, as shown in FIG. 5 SA 575 and 585, which may be an aggregation of individual SAs for each processor, may be coupled to each pair of

U.S. Patent No. 11,841,803	References
	processor cores 574 a and 574 b and 584 a and 584 b, respectively, to handle memory traffic in accordance with an embodiment of the present invention. Still referring to FIG. 5, first processor 570 further includes a memory controller hub (MCH) 572 and point-to-point (P-P) interfaces 576 and 578. Similarly, second processor 580 includes a MCH 582 and P-P interfaces 586 and 588. As shown in FIG. 5, MCH's 572 and 582 couple the processors to respective memories, namely a memory 532 and a memory 534, which may be portions of main memory (e.g., dynamic random access memory (DRAM)).") Sistla at [0044] ("First processor 570 and second processor 580 may be coupled to a chipset 590 via P-P interconnects 552 and 554, respectively. As shown in FIG. 5, chipset 590 includes P-P interfaces 594 and 598. Furthermore, chipset 590 includes an interface 592 to couple chipset 590 with a high performance graphics engine 538. In one embodiment, an Advanced Graphics Port (AGP) bus 539 or a point-to-point interconnect may be used to couple graphics engine 538 to chipset 590. As shown in FIG. 5, various I/O devices 514 may be coupled to first bus 516, along with a bus bridge 518 which couples first bus 516 to a second bus 520. Various devices may be coupled to second bus 520 including, for example, a keyboard/mouse 522, communication devices 526 and a data storage unit 528 which may include code 530, in one embodiment. Further, an audio I/O 524 may be coupled to second bus 520.") <i>See also</i> evidence cited for '803 patent limitation 1[A]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
1[C]: a second GPU chiplet communicably coupled to the first GPU chiplet via a passive crosslink,	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <u>Loh-554</u>

Loh-554 at Fig. 1

**FIG. 1**

Loh-554 at Fig. 3

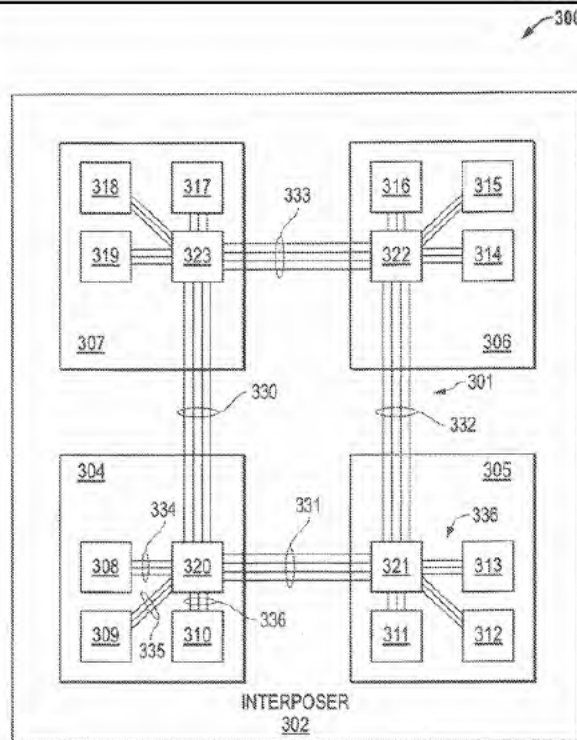


FIG. 3

Loh-554 at [0011]-[0012] (“During operation, the inter-die communications between the horizontally-stacked die 104–107 are conducted using the traces, vias, and other interconnects formed from one or more metal layers of the interposer 102. In a conventional system, a point-to-point link formed from interconnects in the interposer 102 between two die would be needed in order for the two die to communicate with each other. As noted above, this approach results in one or both of an excessive number of metal layers in the interposer or excessively long interconnects in order to accomplish the routing. To reduce or eliminate such issues, in some embodiments the processing system 100 implements a multi-hop network 101 composed

U.S. Patent No. 11,841,803	References
	of a router partition formed by routing logic at one or more of the horizontally-stacked die 104–107 and a link partition formed by the inter-device interconnects of the interposer 102 connecting the horizontally-stacked die 104–107 and the inter-die interconnects connecting the vertically-stacked die 107 and 111–113. The router partition comprises logic and other circuitry of the die that is used to make routing decisions to route packets on one or more hops over the inter-die interconnects and the intra-die interconnects forming the link partition. The link partition includes conductors coupling the transmit/receive circuitry of one die to the transmit/receive circuitry of another die ("inter-die interconnects"). These inter-die interconnects can include electrical conductors such as pads, pins, pin interfaces, metal layers, plated through holes, and vias on the interposer 102 or TSVs between vertically-stacked die. Such inter-die interconnects also can include optical conductors or a combination of both electrical and optical conductors. The link partition further can include the conductors coupling sets of transmit/receive circuitry on the same die ("intra-die interconnects"), such conductors including, for example, traces, vias, throughholes, pads, solder bumps, and the like. To illustrate, in the example processing system 100 the horizontally-stacked die 104-107 are arranged on the interposer 102 in a ring network whereby die 104 is connected to die 105 via link 116, die 105 is connected to die 106 via link 117, die 106 is connected to die 107 via link 118, and die 107 is connected to die 104 via link 119. The links 116-119 are implemented in the one or more metal layers of the interposer 102. Moreover, the illustrated link partition of processing system 100 also includes a link 120 formed by a plurality of TSVs 122 or other conductors that interconnect the vertically-stacked die 107 and 111-113. In other embodiments, the link partition also may include on-die links that interconnect devices located on a particular die, as illustrated in greater detail below with reference to FIG. 3.”). <u>Shih</u> Shih at [0010]-[0011] (“According to one embodiment, the first logic die comprises central processing units, graphics processing units, or application processors. The second logic die comprises central processing units, graphics processing units, or application processors.

According to one embodiment, the bridge memory die allows inter-processor communication between the first logic die and the second logic die.”).

Liu

Liu at Fig. 2

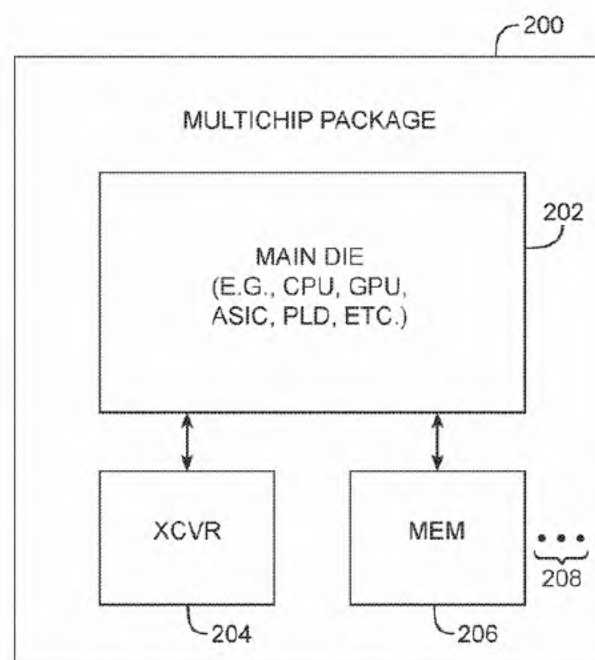


FIG. 2

Liu at Fig. 3

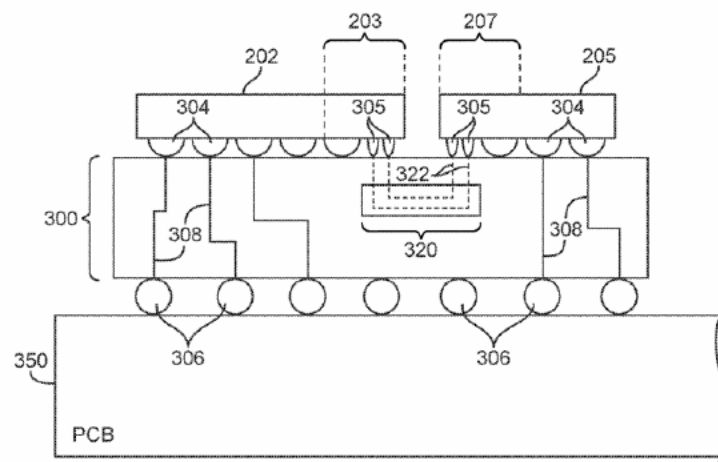


FIG. 3

Liu at [0004] (“EMIBs are small silicon dies that are sometimes embedded in the substrate of a multichip package and are used to interconnect integrated circuit dies within that multichip package. Traditionally, these EMIBs have limited power delivery capability compared to other interposer technologies such as silicon interposers.”).

Liu at [0006] (“An integrated circuit package may include a package substrate and one or more integrated circuit dies mounted on the package substrate. The package substrate may include an embedded multi-die interconnect bridge (EMIB) embedded within the package substrate. An EMIB is a silicon die that may be used to interconnect two integrated circuits in a multi-chip package. The integrated circuit dies mounted on the package substrate may communicate with one another through the EMIB. The EMIB may have a front side that faces the integrated circuit dies and a back side that opposes the front side. The package substrate may include a conductive path that is electrically coupled to the EMIB from the back side of the EMIB and

U.S. Patent No. 11,841,803	References
	that supplies power to the EMIB. The package substrate may be mounted on a printed circuit board that provides power to the EMIB through the conductive path.”). Liu at [0009] (“The EMIB may include a conductive routing trace (e.g., interconnect) that is coupled to the integrated circuit dies. A microvia formed in the EMIB may be coupled between one of the conductive pads formed at the back side of the EMIB and the conductive routing trace. Power supply voltage signals or data signals may be provided to the conductive routing trace through the microvia.”). Liu at [0028] (“In order to facilitate communications between two chips on a multi-chip package, the package may include an embedded multi-die interconnect bridge (EMIB) that is designed and patented by INTEL Corporation. An EMIB is a small silicon die that is embedded in the underlying substrate of a multi-chip package and that offers dedicated ultra-high-density interconnection between dies within the package. EMIBs generally include wires of minimal length, which help to significantly reduce loading and directly boost performance.”). Liu at [0035] (“Main die 202 may be coupled to a secondary die 205 using EMIB 320 that is embedded in package substrate 300. Signals being passed between main die 202 and secondary die 205 may pass through interconnects (e.g., conductive paths) 322 and microbumps 305. EMIB 320 may have a front side that faces main die 202 and secondary die 205 and may have a back side that faces package substrate 300. An EMIB is traditionally formed on a solid, electrically floating conductive plate for structural support. It is therefore difficult to provide power to microbumps 305 that overlap with regions 203 and 207 of main die 202 and secondary die 205, as power cannot be delivered vertically from the PCB through the EMIB to regions 203 and 207 because back side routing is blocked by the conductive plate.”). Liu at Claim 1 (“An integrated circuit package, comprising: a package substrate; and first and second integrated circuit dies mounted on the package substrate, wherein the package substrate comprises: a silicon die embedded within the package substrate, wherein the first integrated circuit die is configured to communicate with the second integrated circuit die through the silicon die,

wherein the silicon die has a front side that faces the first and second integrated circuit dies and a back side opposing the front side;
 a conductive path that is electrically coupled to the silicon die from the back side of the silicon die; and
 a backside conductor on which the silicon die is mounted, wherein the conductive path is connected to the backside conductor, and wherein the backside conductor provides a power supply voltage to the first and second integrated circuit dies.”).

Liff

Liff at Fig. 1

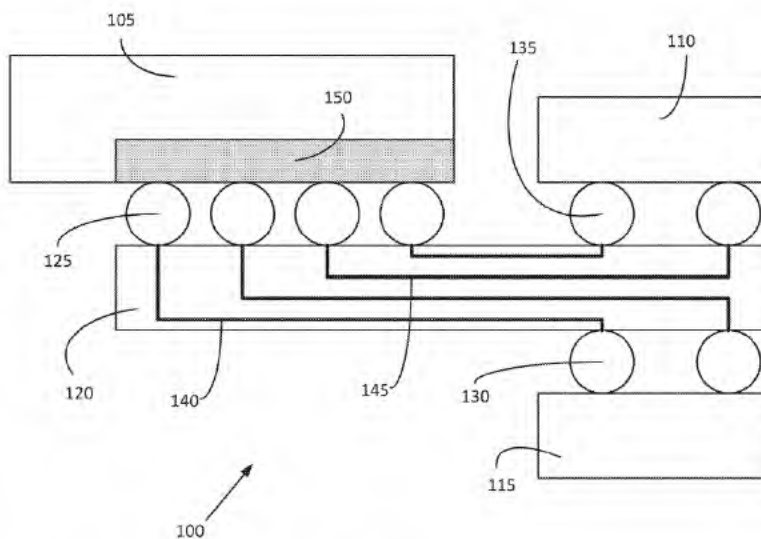


Figure 1

Liff at [0024] (“Embodiments herein relate to use of an integrated circuit (IC) such as an interposer with interconnects on both top and bottom. The interposer may also be referred to as

U.S. Patent No. 11,841,803	References
	a “bridge with direct electrical interconnect on both top and bottom,” however for the sake of consistency herein the term “interposer” will be generally used. The interconnects may be based on, for example, through - silicon - vias (TSVs) or IC stacking. The interposer itself may be passive or active. As used herein, a “passive” interposer may refer to an interposer that is configured to convey signals through the interposer without altering the signals with active circuitry. An “active” interposer may refer to an interposer that includes transistors, logic, or a processor, and is configured to alter the signals in some manner. The interposer may be used to stitch two or more one - sided ICs or pre - assembled stacks together. Specifically, in some embodiments memory or memory stacks may be embedded within the substrate of the package. This configuration may enable the package to satisfy various workloads such as deep learning, machine learning, artificial intelligence, virtual reality, or cloud workloads by expanding the available memory nearby.”). Liff at [0028] (“The processing unit 105, additional element 110, and memory 115 may all be coupled with an interposer 120 by electrical interconnects such as solder bumps 125, 130, and 135. In embodiments, the solder bumps may include tin, tin silver, copper, lead, bismuth, nickel, aluminum combinations thereof, or some other solder material. Although each of solder bumps 125, 130, and 135 are depicted in FIG.1 as having generally the same size as one another, in other embodiments respective ones of solder bumps 125, 130, and 135 may be a different size than another one of solder bumps 125, 130, and 135; or may be plated directly with copper pending the order of fabrication.”). Liff at [0031] (“The conductive elements may form a variety of conductive pathways 140 and 145 between the processing unit 105. Specifically, the conductive pathways 140/145 may include one or more traces or vias. As used herein, a trace may refer to an element that is generally parallel to the face of the interposer 120 to which the memory 115 is coupled, and a “via” may refer to an element that is generally perpendicular to a trace and may provide a communicative path from one layer of the interposer 120 to another layer of the interposer. As can be seen in FIG. 1, conductive pathway 145 may provide a communicative pathway between the processing unit 105 and the additional element 110. Conductive pathway 140 may provide a communicative pathway between the processing unit 105 and the memory 115.”).

U.S. Patent No. 11,841,803	References
	Liff at Claim 1 (“A semiconductor package structure comprising: an interposer with a first side and a second side opposite the first side; a memory communicatively and physically coupled with the first side of the interposer; a processing unit communicatively and physically coupled with the second side of the interposer, wherein the processing unit is communicatively coupled, by the interposer, with the memory; and a third chip coupled with the second side of the interposer.”). <u>Loh</u> Loh at § 3 (“A scalable interconnect is a necessary component for an interposer-based system to fully take advantage of the high-bandwidth of multiple in-package memory stacks. The challenges span both technology-related and architecture issues, of which both types are discussed in this section.”). <u>Lau</u> Lau at Fig. 1

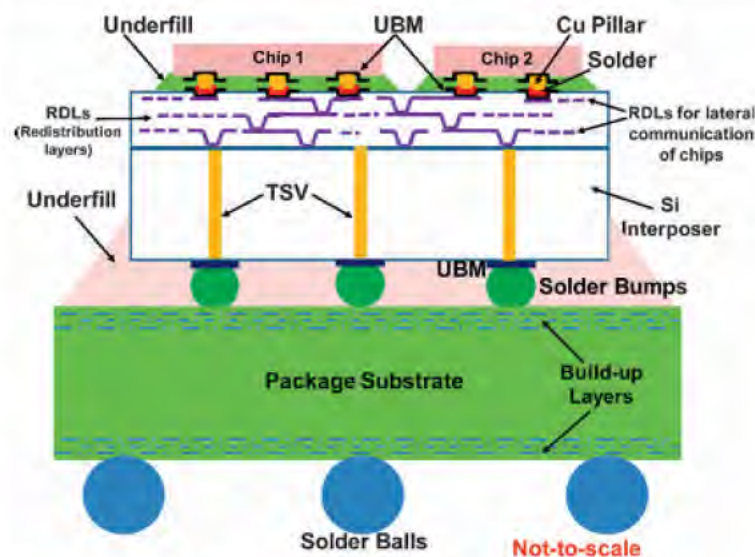


Fig. 1. TSV/RDL passive interposer on substrate.

Lau at 16 (“Abstract—Redistribution layer (RDL) is an integral part of 3D IC integration, especially for 2.5D IC integration with a passive interposer. The RDL allows for circuitry fan-outs of and allows for lateral communication between the chips attached to the interposer. There are at least two ways to fabricate the RDL namely (1) polymers to make the passivation and Cu-plating to make the metal layer, and (2) semiconductor back-end-of-line Cu damascene. In this study, the materials and processes of these methods are presented. Emphasis is placed on the Cu damascene method.”).

Lau at 16 (“One of the potential applications of 3D IC integration is wide I/O interfaces [1], which consists of a piece of device-less silicon with through-silicon vias (TSVs) and high performance, high-density IC chips without TSV. This piece of device-less silicon (also called a passive interposer) is used to support the chips and has RDLs (mainly) for lateral communication between the chips (Fig. 1).

As mentioned in [2-5], one of the key reasons to have the intermediate substrate (passive interposer) is created by the ever increasing IC density and pin-out, and shrinking IC pad-pitch and size, the conventional package substrate cannot support these IC requirements. A couple of examples are shown in Fig. 2 (Xilinx/TSMC) [6-8] and Fig. 3 (Altera/TSMC) [9, 10] (where CoWoS stands for chip on (interposer) wafer on (package) substrate). In these cases, it can be seen that even with 12 buildup layers (6-2-6) on the package substrate, it is still not enough to support the FPGA (field-programmable gate array) chips. In addition, a passive TSV interposer with four top RDLs (three Cu damascene layers and one aluminum layer) at a 0.4 mm pitch is needed. The minimum thickness of the RDLs and passivation is <1 mm. Also, it can be seen that, except for niche applications, passive TSV/RDL interposers are for very high-I/O, high-performance, and high density applications.”).

Saban

Saban at Fig. 1

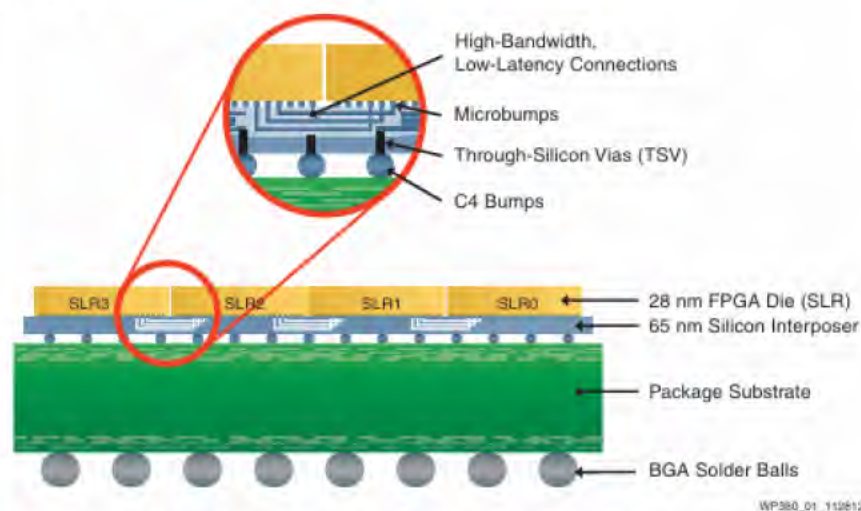


Figure 1: Virtex®-7 2000T FPGA Enabled by SSI Technology

Saban at Fig. 6

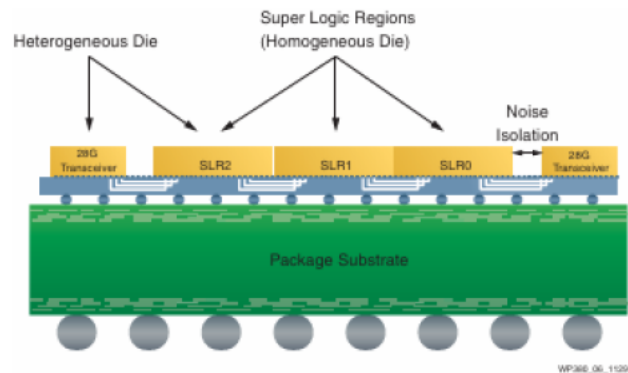


Figure 6: Heterogeneous 3D FPGA with Integrated 28G Transceivers

Saban at 4 (“To overcome these limitations, Xilinx has developed a new approach for building production volumes of high-capacity and high-performance FPGAs. The new solution enables high-bandwidth connectivity between multiple die by providing a much greater number of connections. It also imposes much lower latency and consumes dramatically lower power than either the multiple FPGA or MCM approach, while enabling the integration of massive quantities of interconnect logic, transceivers, and on-chip resources within a single package.

Within the density range of an FPGA family, the medium-density devices represent the “sweet spot.” That is, compared to the previous generation, they offer significantly greater capacity and bandwidth—on a die size that can be delivered earlier in the FPGA product life cycle than the largest devices in the same family. Thus, by combining several of these die in a single device, it is feasible to match or exceed the capacity and bandwidth offered by the largest monolithic devices, but with the manufacturing and time-to-volume advantages of smaller die.

Xilinx arrived at such a solution by applying several proven technologies in an innovative way. By combining TSV and microbump technology with its innovative ASMBL™ architecture, Xilinx is building a new class of FPGAs that delivers the capacity, performance, capabilities, and power characteristics required to address the programmable imperative. Via a

U.S. Patent No. 11,841,803	References
	passive interposer, Xilinx SSI technology combines multiple FPGA SLRs. The interposer provides tens of thousands of die-to-die connections to enable ultra-high interconnect bandwidth with far lower power consumption and one fifth the latency of standard I/Os. Figure 1 shows the side view of the die stack-up with four FPGA SLRs, silicon interposer, and package substrate.”). Saban at 4-5 (“Originally developed for use in a variety of die-stacking design methodologies, silicon interposers provide modular design flexibility and high-performance integration suitable for a wide range of applications. The silicon interposer acts as an interconnect vehicle based on a silicon manufacturing process (e.g., a 65 nm or 45 nm process) on which multiple die are set side by side and interconnected. SSI technology avoids the power and reliability issues that can result from stacking multiple FPGA dies on top of each other or an MCM. Compared to organic or ceramic substrates, which are often used in MCMs, silicon interposers offer far finer interconnect geometries (approximately 20X denser wire pitch) to provide device-scale interconnect hierarchy that enables more than 10,000 die-to-die connections.”). Saban at 7 (“In addition to integrating homogeneous SLRs on a silicon interposer, SSI technology also enables the integration of different types of die. In Figure 6, the Virtex-7 H870T FPGA ties together three SLRs as well as separate 28G capable transceiver circuits via the silicon interposer. Because the SLRs and 28 Gb/s transceiver circuits represent different silicon processes and functions, the Virtex-7 HT FPGA is the world's first heterogeneous architecture—an FPGA consisting of heterogeneous die placed side-by-side to operate as one integrated device.”). <u>Jerger</u> Jerger at Fig. 1

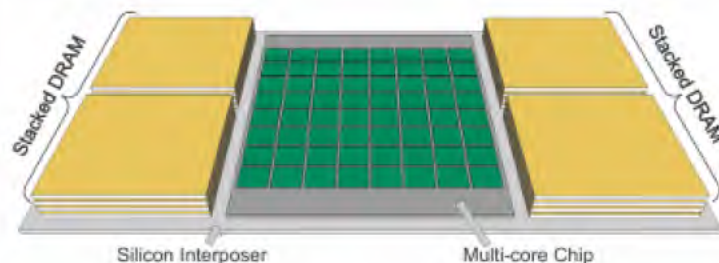


Fig. 1. Baseline system organization consisting of a multi-core chip horizontally stacked with four 3D DRAM stacks on a silicon interposer.

Jerger at Fig. 2

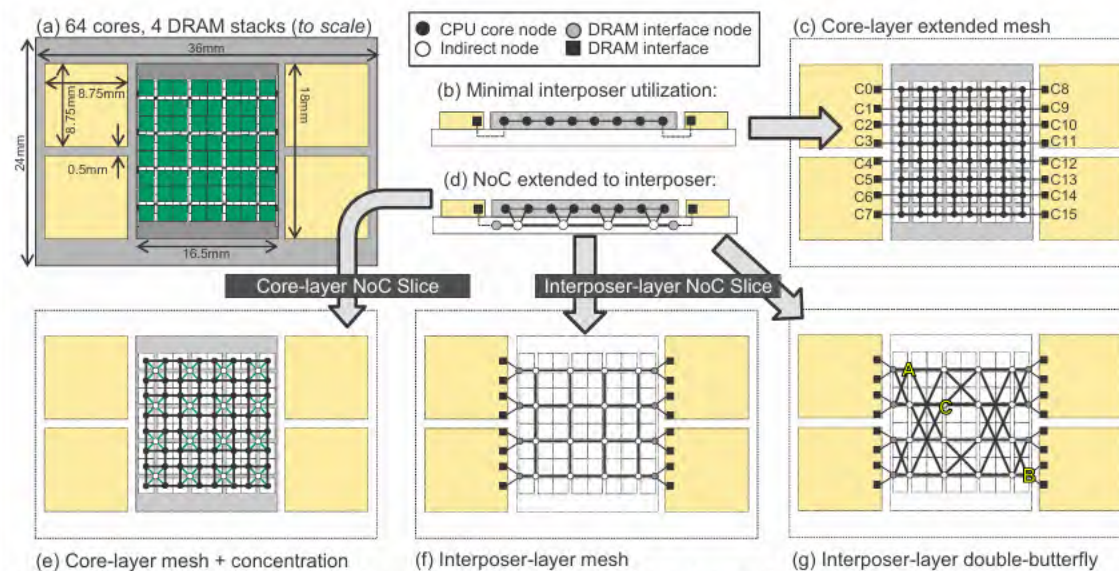
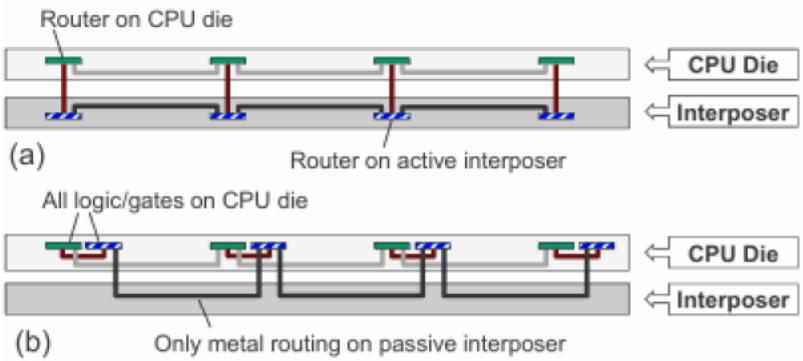


Fig. 2. (a) Top view of the evaluated 2.5D multi-core system with a 64-core CPU chip in the center of the interposer with four DRAM stacks placed on either side of the multi-core die. (b) Side view of a simple interconnect implementation minimizing usage of the interposer. (c) The multi-core NoC slice uses a mesh topology. (d) Side view of a NoC logically partitioned across both the multi-core die and an interposer. (e) Core-layer mesh with concentrated connections to interposer. (f,g) Two different topologies for the interposer NoC slice: concentrated mesh and double butterfly.

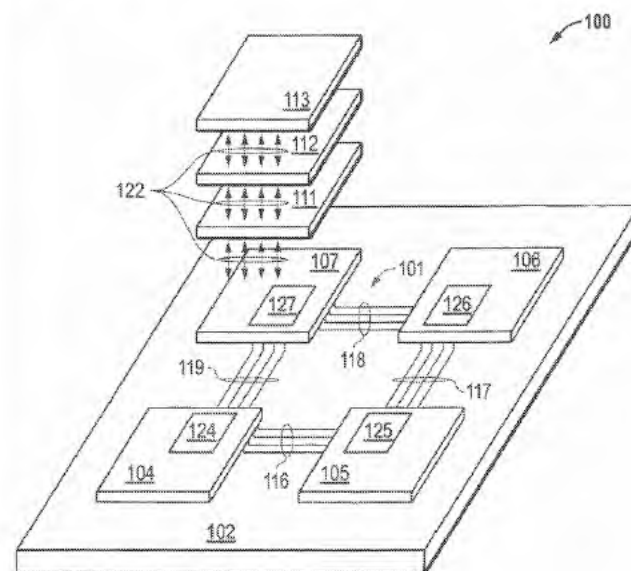
U.S. Patent No. 11,841,803	References
	Jerger at § II (“Horizontal or 2.5D stacking [15] enables the integration of multiple chips similar to vertical or 3D stacking [50]. Using Fig. 1 as a reference, a 2.5D system consists of a base silicon interposer with multiple other chips stacked on top. The interposer consists of a regular (but larger) silicon chip, with conventional metal layers facing upward. Current interposer implementations are “passive” and do not provide any transistors on the interposer silicon layer: only metal routing between chips and TSVs for signals entering/leaving the chip [45]. Future generations of “active” interposers could potentially integrate some devices (possibly in an older technology); the approach espoused in this paper is compatible with either type of interposer.”). Jerger at § II(C) (“The interposer simply provides a mechanical and electrical substrate for the integration of multiple disparate chips. Current 2.5D stacking primarily uses the interposer for edge-to-edge communication between adjacent chips (e.g., processor to stacked DRAM). Apart from this limited routing, the vast majority of the interposer’s area and routing resources are unutilized. Given the assumption that high performance systems will use interposers to integrate memory and processors, we consider what else can we do with the interposer? In this work, we propose a general approach to interposer-based NoC architectures that spans both the multi-core processor and interposer layers, exploiting the otherwise wasted routing resources of the interposer.”). <u>Kannan</u> Kannan at § 2.2 (“Silicon Interposers: A silicon interposer is effectively a large chip upon which other smaller die can be stacked. The micro-bumps (μbumps) used to connect the individual die to the interposer have greater density than C4 bumps (e.g., ~9× better assuming 150μm and 50μm pitches for C4 and μbumps), and the impedance across the interposer is identical to conventional on-chip interconnects (both are made with the same process). The main disadvantage is the cost of the additional interposer.”). This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or

U.S. Patent No. 11,841,803	References
	(3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
1[D]: wherein the passive crosslink is dedicated for inter-chiplet communications.	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <i>See evidence cited for '803 patent limitations 1[A]-1[C].</i> This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
2. The system of claim 1, wherein the passive crosslink comprises a passive interposer die.	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <u>Liff</u> Liff at [0030] ("The interposer 120 may be a passive interposer that may include silicon, glass, an alternative ceramic, or some other similar material. Alternatively, the interposer 120 may be an active interposer which may include silicon, silicon-on-insulator, silicon-germanium, or some other IC process technology. Specifically, the interposer 120 may be formed with a multi-layered dielectric material such as silicon dioxide, a carbon-doped silicon oxide, polyimide or other photo-imagable dielectric, or epoxy and silicon dioxide composite dielectric that is not photo-imagable and which may be referred to as a build-up film, or some other dielectric material. The interposer 120 may include a plurality of conductive elements such as traces, vias, or other elements such as inductors or capacitors. The traces or vias may be formed of one or more conductive materials such as copper, aluminum, gold, or some other conductive material. In some embodiments, piezoelectric or magnetic material may be added

U.S. Patent No. 11,841,803	References
	to the interposer 120, and particularly the conductive elements, to increase inductance and capacitance. In some embodiments, these elements may be found on either side of the interposer 120, or on both sides of the interposer 120.”). <u>Jerger</u> Jerger at Fig. 3  Figure 3 consists of two cross-sectional diagrams, (a) and (b), illustrating different NoC implementations. Diagram (a) shows a CPU die with routers on its surface, connected to an interposer layer. Diagram (b) shows a CPU die with all logic/gates on its surface, connected to an interposer layer that provides only metal routing. Labels include 'Router on CPU die', 'CPU Die', 'Interposer', 'Router on active interposer', 'All logic/gates on CPU die', and 'Only metal routing on passive interposer'. Jerger at § III(B) (“Dealing with a Passive Interposer: To implement the NoC on an active interposer, we simply place both the NoC links (wires) and the routers (transistors) on the interposer layer. Fig. 3(a) shows a small example NoC with the interposer layer’s partition of the NoC completely implemented on the interposer. For the near future, however, it is expected that only passive, device-less interposers will be commonly used. Fig. 3(b) shows an implementation where the active components of the router (e.g., buffers, arbiters) are placed on the CPU die, but the wide NoC links (e.g., 128bits/direction) still utilize the interposer’s routing resources. This approach enables the utilization of the interposer’s metal layers for NoC routing at the cost of some area on the CPU die to implement the NoC’s logic components.¹ Both NoCs in Fig. 3 are topologically and functionally identical, but have different physical organizations to match the capabilities (or lack thereof) of their respective interposers.”).

U.S. Patent No. 11,841,803	References
	<u>Loh</u> Loh at § 3.1 (“In the near term, silicon interposers will likely all be passive”, meaning that they provide metal interconnects (wires) to connect the processor to the memory stacks, but the interposer itself does not provide any “active” devices (i.e., transistors). In the longer term, the interposer layer could conceivably also incorporate active devices⁴, especially if the interposer is implemented in an older/cheaper technology and perhaps adequate redundancy is incorporated to address potential yield challenges from building such a relatively large piece of silicon. Implementing the NoC with a passive interposer enables the NoC to make use of the additional wiring resources of the interposer layer to either relieve routing congestion on the processor die, or to provide more total routing bandwidth without adding more metal layers to the processor. However, active components (i.e., network routers, buffers, repeaters) still must reside on the processor die, and so there are new challenges in organizing the network topology in a meaningful way that takes advantage of these additional wires on the interposer while working with the constraint that all logic remains on the processor die. The additional active components also consume more area and energy on the processor die.”). <i>See also</i> evidence cited for ’803 patent limitation 1[D]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA’s other invalidity charts and cover pleading.
3[A]: The system of claim 1, wherein the first GPU chiplet comprises a first PHY region including conductor structures for chiplet-to-chiplet communications,	Including at least under Complainant’s apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <u>Loh-554</u>

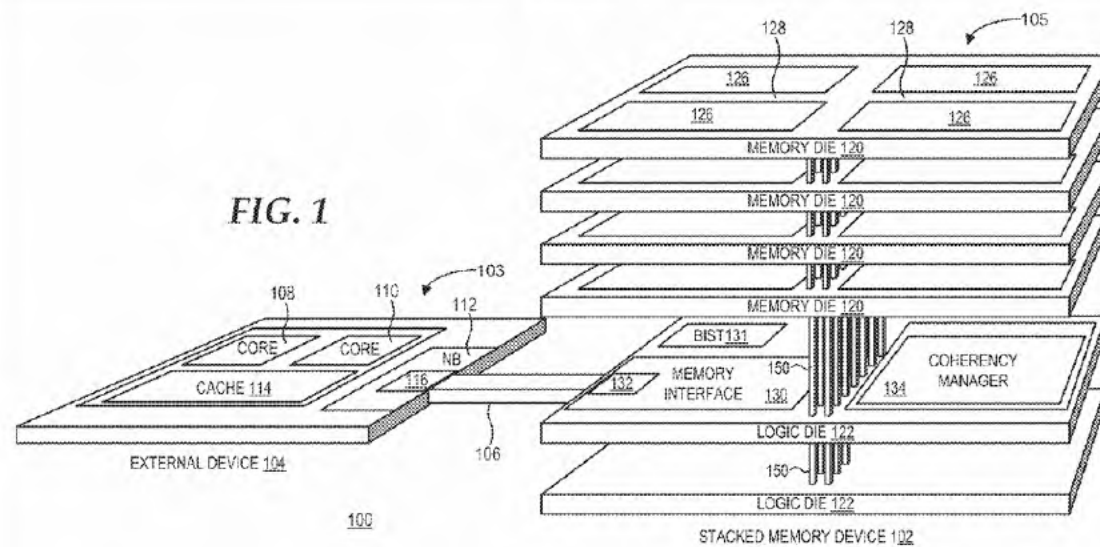
Loh 554 at Fig. 1

**FIG. 1**

Loh-554 at [0019]-[0020] (“The PHY on die 104 receives the signaling and buffers the memory write request information represented by the signaling. The routing logic 124 on die 104 inspects the buffered packet header and extracts the destination node. Since the final destination is not the local device on die 104, the routing logic 124 next performs a table lookup to determine the next node in the route to the memory controller. In this example, the next node in the path is the router on die 107. The routing logic 124 on die 104 therefore causes the write request packet to be placed in the output buffers corresponding to the link 119 connecting die 104 to die 107. The interface logic on die 104 then issues the packet by

U.S. Patent No. 11,841,803	References
	manipulating the physical interface (PHY) connected to the conductors that comprise link 119 connected to die 107 to transmit signaling representative of the memory write request. The PHY on die 107 receives the signaling and buffers the memory request packet. The routing logic 127 on die 107 inspects the packet header and extracts the destination node. In this example the destination node matches the memory controller's node identification. The routing logic 127 therefore places the write request into input buffer of the memory controller device on die 107. The memory controller accesses the appropriate DRAM cells on die 111-113, storing the write data to the location of memory indicated by the signaled address, thereby completing the requested memory write operation which was initiated by the CPU on die 105."). <i>See also</i> evidence cited for '803 patent claims 1-2. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
3[B]: and further wherein the second GPU chiplet comprises a second PHY region including conductor structures for chiplet-to-chiplet communications.	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <i>See</i> evidence cited for '803 patent limitation 3[A]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
7. The system of claim 1, wherein the passive crosslink	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation.

U.S. Patent No. 11,841,803	References
communicably couples all GPU chiplets in the GPU chiplet array.	<i>See, e.g.,</i> <i>See evidence cited for '803 patent limitations 1[C] and 1[D].</i> This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
8[A]: The system of claim 1, further comprising: a first cache memory hierarchy at the first GPU chiplet,	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <u>Loh-948</u> Loh-948 at Fig. 1



Loh-948 at Fig. 3

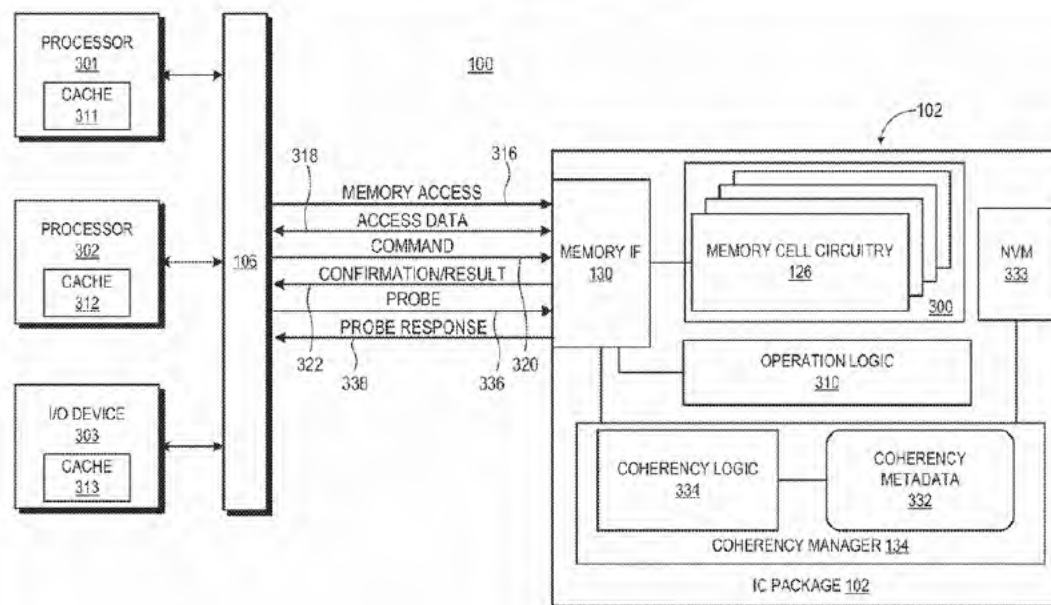


FIG. 3

Loh-948 at 2:9-36 (“FIGS. 1-7 illustrate example techniques for improved processing efficiency and decreased power consumption in a processing system through the use of a die-stacked memory device implementing an integrated coherency manager to offload some or all cache coherency protocol operations for memory addresses or memory address ranges mapped to the memory cell circuitry of the die-stacked memory device. The die-stacked memory device includes a set of one or more stacked memory dies and a set of one or more logic dies. The one or more logic dies implement hardware logic for a memory interface and the coherency manager. The memory interface is coupled to the memory cell circuitry and is coupleable to one or more devices external to the die-stacked memory device. The memory interface operates to perform memory accesses in response to memory access requests from

U.S. Patent No. 11,841,803	References
	both the coherency manager and the one or more external devices. The coherency manager comprises logic to perform one or more coherency operations for shared data stored at the one or more stacked memory dies. Due to the coherency manager's tight integration with the memory dies, the coherency manager can access shared data stored in the memory dies and perform related coherency operations with higher bandwidth and lower latency and power consumption compared to the external devices. Moreover, the offloading of coherency operations to the die-stacked memory device permits the external devices to perform other tasks focusing on program execution, thereby increasing the overall processing throughput of the system.”). Loh-948 at 3:12-28 (“In the illustrated example, the external device 104 is depicted as a processor, and is thus also referred to herein as "processor 104." External devices can include other types of devices, such as input/output (I/O) controllers. In this example, the processor 104 comprises one or more processor cores, such as processor cores 108 and 110, a northbridge 112, one or more caches 114 (e.g., a L1 cache, a L2 cache, etc.), and various peripheral components (not shown). The processor cores 108 and 110 can include any of a variety of processor cores and combinations thereof, such as a central processing unit (CPU) core a graphics processing unit (GPU), a digital signal processor (DSP), and the like. The peripheral components can include, for example, an integrated southbridge or input/output controller, and the like. The northbridge 112 includes, or is associated with, a memory controller interface 116 comprising a physical interface (PHY) connected to the conductors of the interconnect 106.”). Loh-948 at 4:25-61 (“In addition to implementing logic to facilitate access to the memory implemented by the memory dies 120, one or more logic dies 122 implement a coherency manager 134 to partially or fully implement a cache coherency protocol for memory addresses or memory address ranges mapped to the memory cell circuitry 126 for the benefit of the external devices of the processing system 100. The coherency manager 134 is coupled to the memory interface 130 and comprises logic to perform one or more coherency operations. The coherency manager 134 may include storage elements (e.g., registers, caches, or content addressable memories) located at one or more of the logic dies 122 to store cache coherency-related information (hereinafter, “coherency information”), the memory cell circuitry 126 may store the coherency information, or some portions of the coherency information may be stored

U.S. Patent No. 11,841,803	References
	in the storage elements of the logic dies 122 while other portions are stored in the memory cell circuitry 126. Further, in some embodiments, the coherency manager 134 can employ a non-volatile memory (NVM), such as flash memory, at a logic die 122 or in a memory die 120, to retain certain coherency information after a power down event. In the illustrated example, the coherency manager 134 and the memory interface 130 are implemented on the same logic die 122. In some embodiments, the memory interface 130 and the coherency manager 134 may be implemented on different logic dies. For example, the memory interface 130 may be implemented at one logic die 122 and the coherency manager 134 may be implemented at another logic die 122. In some embodiments, one or both of the memory interface 130 and the coherency manager 134 may be implemented across multiple logic dies. To illustrate, the memory interface 130 and the logic circuitry of the coherency manager 134 may be implemented at one logic die 122 and certain storage elements of the coherency manager 134 (e.g., a cache or content addressable memory) may be implemented at another logic die 122.”). <u>Shih</u> Shih at Fig. 4

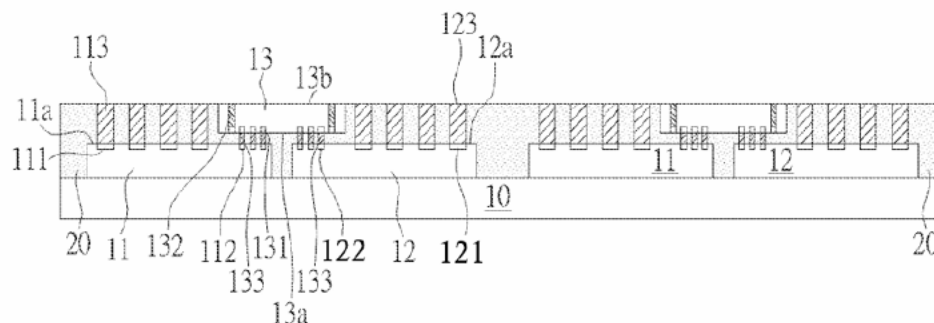


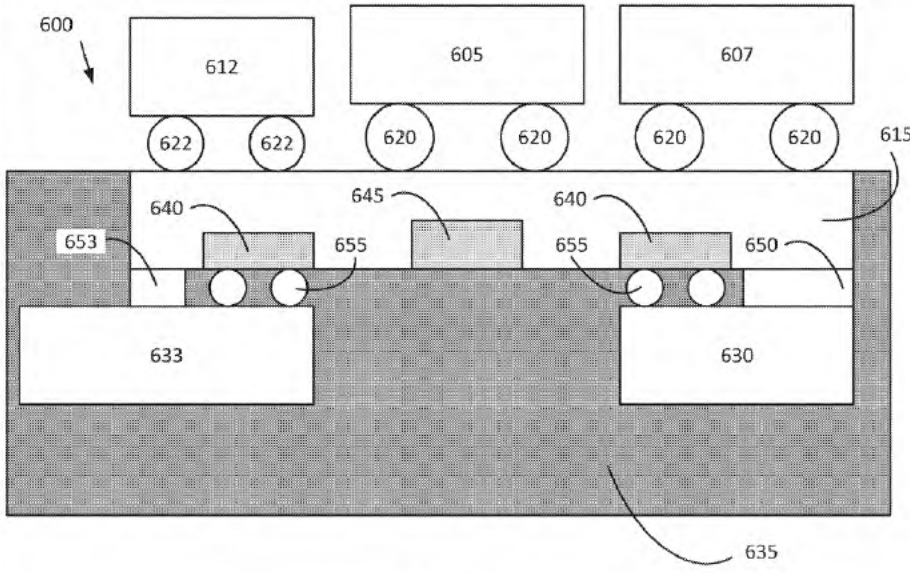
FIG. 4

Shih at [0031]-[0032] (“According to one embodiment of the invention, the bridge memory die 13 may be coupled to both of the logic dies 11 and 12 using, for example, a flip-chip bonding process. By providing such face-to-face configuration, the bridge memory die 13 is disposed in close proximity to the two logic dies 11 and 12. The distance between the memory die and the logic die is reduced and therefore the memory operating speeds and the signal integrity can be improved.

As shown in FIG. 4, a molding compound 20 is applied to at least partially encapsulate the two logic dies 11 and 12, the bridge memory die 13, and the connecting elements 113 and 123. The molding compound 20 may fill into a gap between the bridge memory die 13 and the two logic dies 11 and 12, and into a gap between the two logic dies 11 and 12. According to one embodiment of the invention, the molding compound 20 may be subjected to a curing process.”).

Liff

Liff at Fig. 6

U.S. Patent No. 11,841,803	References
	 Figure 6 Liff at Fig. 8

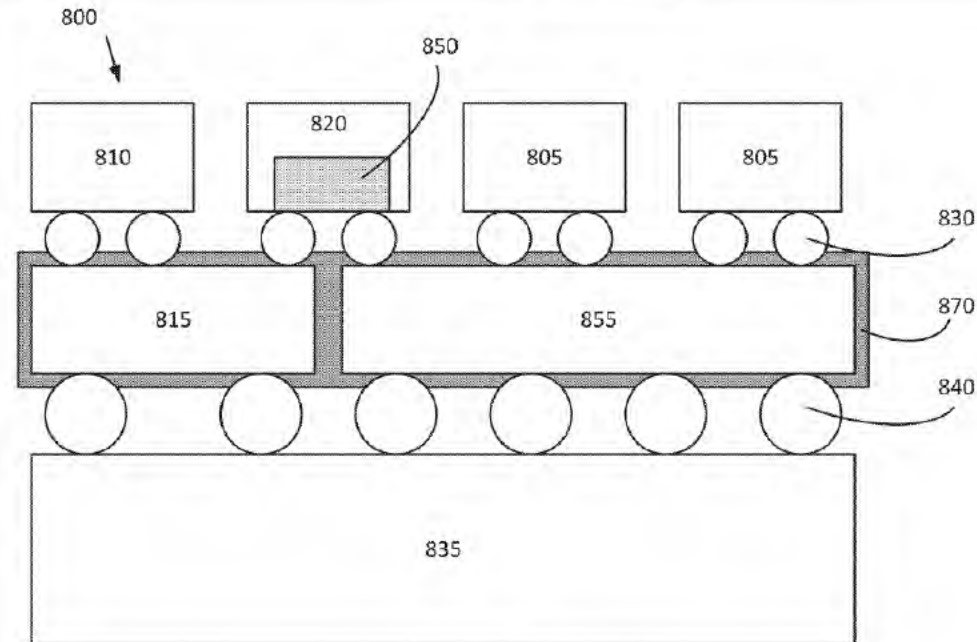
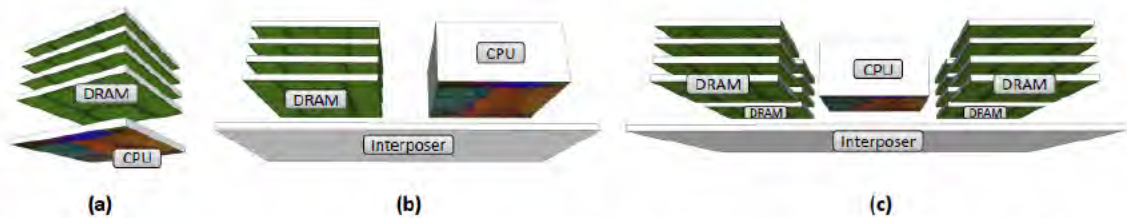


Figure 8

Liff at [0024] (“Embodiments herein relate to use of an integrated circuit (IC) such as an interposer with interconnects on both top and bottom. The interposer may also be referred to as a “bridge with direct electrical interconnect on both top and bottom,” however for the sake of consistency herein the term “interposer” will be generally used. The interconnects may be based on, for example, through - silicon - vias (TSVs) or IC stacking. The interposer itself may be passive or active. As used herein, a “passive” interposer may refer to an interposer that is configured to convey signals through the interposer without altering the signals with active circuitry. An “active” interposer may refer to an interposer that includes transistors, logic, or a processor, and is configured to alter the signals in some manner. The interposer may be used to stitch two or more one - sided ICs or pre - assembled stacks together. Specifically, in some embodiments memory or memory stacks may be embedded within the substrate of the

U.S. Patent No. 11,841,803	References
	package. This configuration may enable the package to satisfy various workloads such as deep learning, machine learning, artificial intelligence, virtual reality, or cloud workloads by expanding the available memory nearby.”). Liff at [0057] (“The package 600 may further include power routing 650 and 653 across the “top” of the memory 630 and 633 coming from the lower layers in the package via traces and via through the surrounding dielectric or from other devices on the interposer 615 or the substrate 635. As used herein, the “top” of the memory 630 and 633 may refer to the side of the memory that is closest to the interposer 615. More specifically, the power routing 653 and 650 may be coupled with the interposer 615, and configured to provide power from the interposer 615 to the memory 630 and 633. It will be understood that although the power routing 650 and 653 is shown as separate from, and coupled with, the interposer 615 and the memories 630 and 633, in some embodiments, the power routing 650 or 653 may be at least partially element of one or both of the interposer, the package, or the memory. For example, the power routing 650 or 653, may be positioned in a top layer of the memories 630 or 633 during the embedding process, and communicatively coupled with the interposer by solder bumps such as solder bumps 655 or further die embedding processes. In other embodiments, the power routing 650 or 653 may be positioned within or atop the interposer 615 and communicatively coupled with memories 630 or 633 by solder bumps such as solder bumps 655, or by one or more conductive elements such as vias, traces, or pads through copper plating. In some embodiments, power routing 650 or 653 may not be present, and rather the memory stack 633 or 630 may terminate with TSVs that can access power from the base of the substrate 635 and draw the power directly into the memory 633 and 630.”). Liff at [0068] (“As can be seen, the die 820 may connect memory 815 and base die 855. As a result, a processing unit such as processing unit 805 may be communicatively coupled with the die 820 through base die 855. The die 820 may in turn be communicatively coupled with memory 815 and, through memory 815, additional element 810. In operation, the processing unit 805 may transmit a memory access request through the base die 855 to die 820, and more specifically to memory control logic 850. The memory control logic 850 may then access memory 815 or the additional element 810 through memory 815 or along redistribution layers atop 815 if additional element 810 is a memory. If the memory access request was a read command, the memory controller 850 may read data from the memory 815 or the additional

U.S. Patent No. 11,841,803	References
	element 810 and return that data to the processing unit 805 through the base die 855 or across redistribution layers atop 855 such as those within substrate 870.”). Liff at Claim 1 (“A semiconductor package structure comprising: an interposer with a first side and a second side opposite the first side; a memory communicatively and physically coupled with the first side of the interposer; a processing unit communicatively and physically coupled with the second side of the interposer, wherein the processing unit is communicatively coupled, by the interposer, with the memory; and a third chip coupled with the second side of the interposer.”). <u>Loh</u> Loh at Fig. 1  Figure 1 consists of three 3D perspective diagrams labeled (a), (b), and (c). Diagram (a) shows a vertical stack of four green rectangular blocks labeled 'DRAM' on top of a single white rectangular block labeled 'CPU'. Diagram (b) shows a stack of three green rectangular blocks labeled 'DRAM' on top of a white rectangular block labeled 'CPU', which is mounted on a larger, light gray rectangular base labeled 'Interposer'. Diagram (c) shows a stack of three green rectangular blocks labeled 'DRAM' on top of a white rectangular block labeled 'CPU', which is mounted on a larger, light gray rectangular base labeled 'Interposer'. To the right of the CPU block in (c), there is another stack of three green rectangular blocks labeled 'DRAM'. Figure 1: Processor-and-memory systems using (a) vertical 3D stacking, (b) 2.5D stacking on a silicon interposer, and (c) 2.5D stacking incorporating multiple memory stacks. Loh at Fig. 3

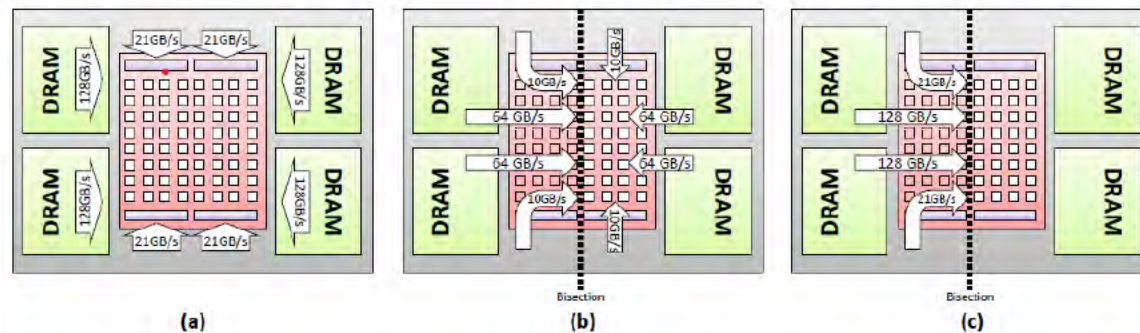


Figure 3: (a) An example processor-and-memory system with four 128 GB/s memory stacks, and four channels of 21 GB/s external memory. (b) Memory usage scenario where all cores access all memory sources uniformly. (c) Memory usage scenario where all cores on the right of the bisection line access only memory from the other side of the bisection.

Loh at Fig. 4

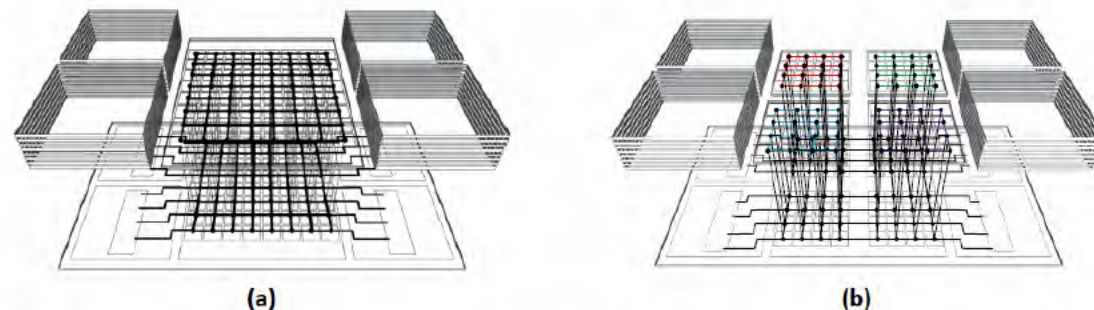


Figure 4: A system with four memory stacks on an interposer along with 64 CPU cores implemented as (a) a monolithic 64-core chip, and (b) four separate 16-core chips. The figure includes example NoC topologies.

Loh at 2.2 (In our first scenario, we consider a well-balanced application where each processor core's memory traffic is uniformly sourced from each stack of memory and each external memory channel. More specifically, each stack's 128 GB/s bandwidth would be evenly divided among each of the cores (i.e., 2 GB/s per core per stack). Likewise, each external memory

U.S. Patent No. 11,841,803	References
	channel's 21 GB/s is divided evenly 64 ways for 0.32 GB/s per core per channel. Fig. 3(b) illustrates how the 32 cores on the right would consume 64 GB/s of bandwidth from each of the two memory stacks on the left, plus another ~10 GB/s from each memory channel. This implies that the underlying network on chip (NoC) must support 148 GB/s across this bisection line to not become a bottleneck. Furthermore, the traffic from the cores on the left will be symmetric, so the interconnect must independently support ~148 GB/s <i>in each direction</i> across the bisection. Making matters worse, the estimated bandwidth numbers here are only for raw memory traffic rates; additional bandwidth is needed for cache coherence traffic among the cores. Real applications are generally not going to demonstrate a perfectly uniform distribution of memory accesses. Fig. 3(c) provides an example at the other extreme. In this case, all of the right-side cores receive all of their traffic from the left-side memory resources. As such, the memory traffic across the bisection doubles to 298 GB/s (and again, more bandwidth may need to be supported for cache coherence traffic). In practice, a real application's needs will probably fall somewhere between these cases, but the examples serve to illustrate the likely range of bandwidths that the interconnect must support to avoid becoming the bottleneck to memory.”). <u>Jerger</u> Jerger at § III(C) (“Differentiating Coherence and Memory Traffic: Differentiating between traffic classes in the network can have numerous benefits. Coherence traffic must be divided across multiple virtual or physical networks [49], [52] in order to avoid protocol-level deadlock. This separation also provides opportunities to tailor the network to particular aspects of the coherence traffic [48]. The core-to-core cache coherence traffic and the core-to-memory traffic exhibit different characteristics [4]. Coherence traffic is characterized by any-to-any communication patterns that over long intervals can resemble uniform random traffic. However, memory traffic produces a many-to-few traffic pattern [1], as memory requests originate from cores and always target memory nodes, never other cores. With memory positioned at the edge of the system, the average hop count for memory traffic on a conventional mesh is substantially larger than the average hop count for cache traffic. Furthermore, cache coherence traffic can often interfere with main memory traffic, and likewise main memory traffic can get in the way of coherence traffic. Given a NoC topology

that is distributed across both the multi-core die and the interposer, we propose to functionally partition the NoC so that core-to-core coherence traffic is routed on the multi-core die's portion of the NoC, and the main memory traffic is transported on the interposer's portion of the NoC.”).

Kannan

Kannan at Fig. 5

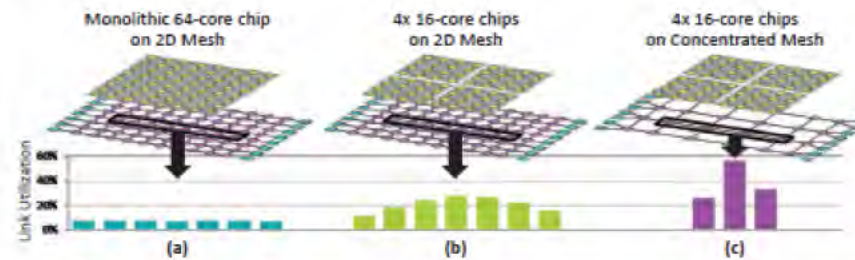


Figure 5: Link utilization from a single “horizontal” row of routers on the interposer for (a) a monolithic 64-core chip stacked on a interposer with a 2D mesh, (b) four 16-core chips stacked on a 2D mesh, and (c) four 16-core chips stacked on a concentrated mesh.

Kannan at § 3.1 (“In an interposer-based system employing a monolithic multi-core chip, the NoC traffic can be cleanly separated into cache coherence traffic routed on the CPU layer, and main memory traffic on the interposer layer [13]. This minimizes interference and contention between the traffic types and more easily allows for per-layer customized topologies that best match the respective traffic patterns. When the multi-core chip has been broken down into smaller pieces, coherence traffic between cores on different chips must now venture off onto the interposer. As a result, this mixes some amount of coherence traffic in with the main memory traffic, which in turn disturbs the traffic patterns observed on the interposer.”).

U.S. Patent No. 11,841,803	References
	Kannan at § 3.1 (“Figure 5 shows per-link traffic for a horizontal set of routers across the interposer for several topologies. The first is the baseline case for a monolithic 64-core chip stacked on an interposer that also implements a 2D mesh. All coherence stays on the CPU die, and memory traffic is routed across the interposer, which results in relatively low and even utilization across all links. Figure 5(b) shows four 16-core chips on top of the same interposer mesh. Traffic between chips must now route through the interposer, which is reflected by an increase particularly in the middle link right between the two chips. Figure 5(c) shows the same four chips, but now stacked on an interposer with a concentrated-mesh network. Any traffic from the left side chips to the right must cross the middle links, causing further contention with memory traffic. The utilization of the middle link clearly shows how the bisection-crossing links can easily become bottlenecks in multi-chip interposer systems.”). <u>Sistla</u> Sistla at Fig. 1

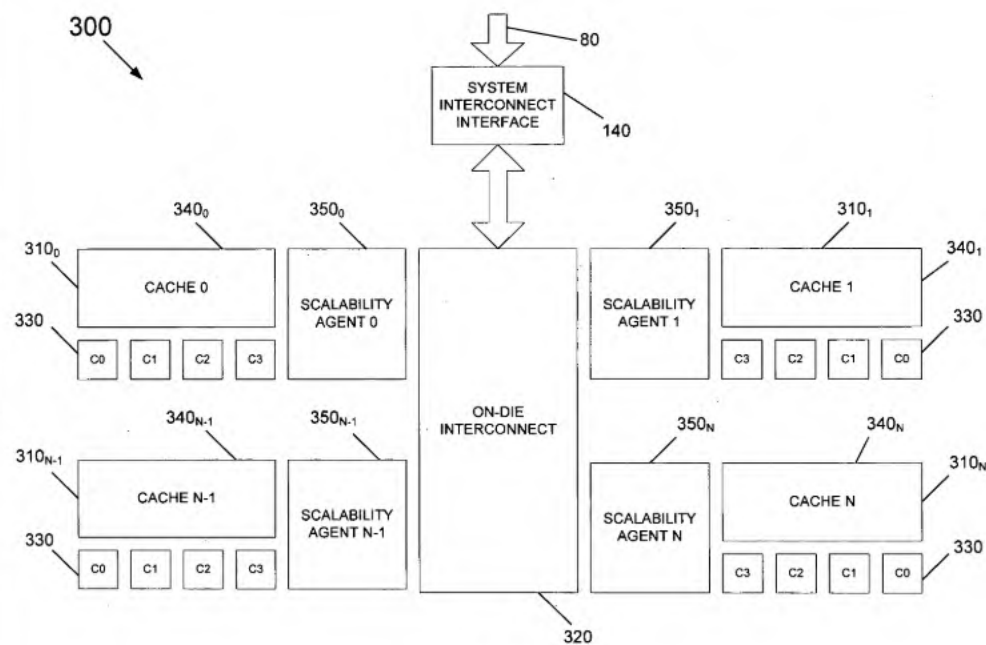


Fig. 1

Sistla at Fig. 2

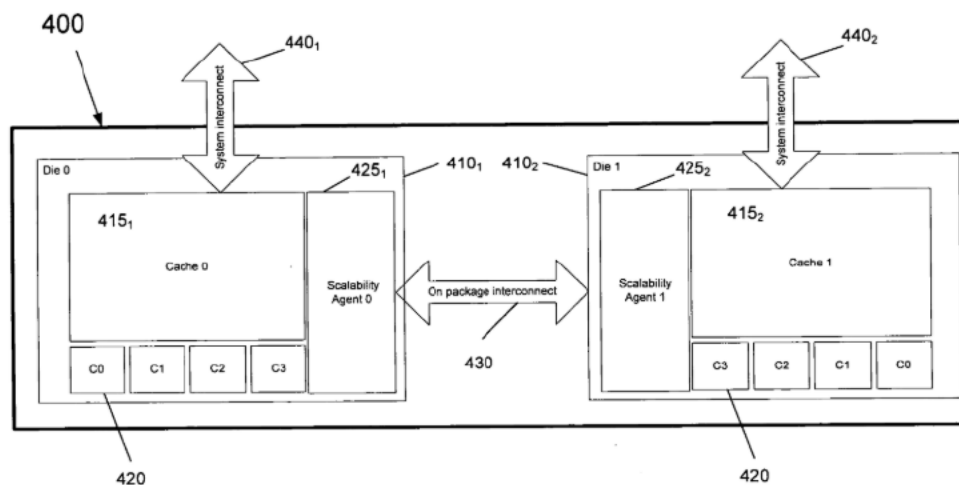


FIG. 2

Sistla at [0002] (“Increased silicon efficiencies are now providing new opportunities for adding additional functionality into the processor silicon. As an example, applications are taking advantage of increased multi-threading capabilities realized from an increased number of processing cores in the same processor. Performance of a multi-core processor may be optimized by mitigating system interconnect latency issues and ensuring that aggregated processor cores appear as one caching agent in order to avoid scalability issues and interconnect saturation. Systems that include caches each associated with multiple cores typically are structured as split caches in which each cache acts as an independent cache domain. This increases the total number of coherence misses, thus increasing the misses per instruction. Split-caches also lose capacity due to replication of shared code/data in all the independent caches.”)

U.S. Patent No. 11,841,803	References
	Sistla at [0008] (“A scalability agent protocol enables a clustered method of scaling the number of cores in a processor. In a clustered scaling method, a core-cache unit is used as a building block and multiple such units are joined to increase the number of cores in the processor. The core-cache unit can contain two or more cores with an appropriately sized shared cache. In various embodiments, multiple core-cache clusters (referred to herein as “clusters”) of a multi-core processor, such as coupled via a system interconnect such as a Common System Interconnect (CSI), may be adapted to provide a socket-wide inclusive cache structure, while individual caches of the clusters may be non-inclusive. The term “core-cache cluster” is generally considered to be a modular unit that comprises one or more cores and a shared cache. Core-cache clusters are used as building blocks for a scalable, multi-core processor.”) Sistla at [0009] (“According to one embodiment of the invention, the “scalability agent protocol” is a communication scheme that enables the aggregating of core-cache clusters operating independently from each other, and provides “graceful” scalability where the responsibility for maintaining memory coherency is approximately shared equally among the core-cache clusters despite any increase or decrease in number, as the scalability agent protocol is partitionable among address regions in the system memory address space. The latencies to the cache in the core-cache cluster substantially remain unchanged as the number of cores increases.”) Sistla at [0010] (“A “scalability agent” is hardware and/or software that manages the flow of outgoing and incoming transactions into a socket associated with the core-cache cluster and supports the scalability agent protocol. According to one embodiment of the invention, the scalability agent (i) aggregates core-cache clusters to appear as one caching agent, (ii) handles local cache coherence between core-cache clusters on the same IC, and (iii) support scalability so that the operations of a core-cache cluster are not significantly effected if other core-cache clusters are added.”) Sistla at [0011] (“Referring now to FIG. 1, an exemplary embodiment of a multi-core processor 300 such as a clustered CMP having a scalability agent is shown. Multi-core processor 300 includes a plurality of core-cache clusters 310 1-310 n (generically cluster 310)

U.S. Patent No. 11,841,803	References
	in communication with each other over an on-die interconnect 320. Multi-core processor 300 is in communication with externally located devices over a system interconnect interface 140 and via an interconnect 80. According to one embodiment of the invention, on-die interconnect 320 is configured as a ring interconnect, but may be configured as an interconnect mesh (e.g., 2D mesh). Each core-cache cluster 310 includes one or more cores 330 that share one of a plurality of caches 340 1-340 N (generically cache 340). The architecture of core-cache clusters 310 may be in accordance with a caching bridge architecture or a distributed shared cache architecture. The transactions involving core-cache clusters 310 are controlled by corresponding scalability agents (SAs) 350 1-350 n (generically scalability agent 350) as described below. According to this architecture, multi-core processor 300 enables the latency of a first shared cache 340 1 to remain substantially constant despite increases in the number of cores in processor 300, ensuring that the scalar performance of threads with no or limited sharing remains constant.”) Sistla at [0012] (“A processor in accordance with one embodiment of the present invention can be aggregated to increase its overall performance and support next generation processor designs. For example, if the core-cache cluster is using the caching bridge style architecture, better performance may be realized by aggregating two (4-core) core-cache clusters to produce an eight core (8-core) multi-core processor. Also, two 4-core clusters can be used to build an 8-core processor in one generation, a 12-core processor in the next generation and 16-core processor in a subsequent generation. The appropriate number “N” of core-cache clusters 310 and the number of cores in each core-cache cluster may be determined to achieve optimum performance. This offers flexibility and the option to choose a simpler implementation. In addition to the above mentioned advantages, SAs enable the individual caches to act together as a shared last level cache (LLC), and allows data that is private to each core-cache cluster to stay in its own cache, resulting in lower latency to private data. Thus further shown in FIG. 1, SA units 350 1-350 4 (generically SA units 350) (N=4) uniquely correspond to core-cache clusters 310 and may be address partitioned into independent SA units 350, each of which is responsible for a subset of an address space.”) Sistla at [0013] (“While shown in the embodiment of FIG. 1 as including a processor formed on a single die, in other embodiments a multi-chip package may be implemented. Referring

U.S. Patent No. 11,841,803	References
	now to FIG. 2, shown is a block diagram of a processor in accordance with another embodiment of the present invention. As shown in FIG. 2, processor 400, which may be a multi-chip processor package, includes a first die 410 1 and a second die 410 2 connected by an on-package interconnect 430, which may be a point-to-point (PTP) interconnect. Each die 410 (generically) may include at least one core-cache cluster including a cache 415 1 and 415 2 and a plurality of cores 420. Each die 410 further includes a scalability agent 425 1 and 425 2. Processor 400 may communicate with other components of a system using a pair of system interconnects 440 1 and 440 2 both of which may be PTP links, in some embodiments. Of course, other implementations are possible. For example, each die may include multiple core-cache clusters and so forth. By implementing the SA on each of the dies, the dies can be joined at the interconnect, thus doubling the number of cores. In addition the caches 415 1 and 415 2 will now behave like a shared cache of the size of both, providing a performance improvement. For example, if each die were a 4-core die with 8 megabytes (M) cache, then the combined 8-core processor may behave as a 16M cache.”) Sistla at [0014] (“Based on such configurations, a scalability agent is adapted to support the scalability agent protocol for aggregating core-cache clusters but the aggregation appears as a single caching agent to devices communicatively coupled to a system interconnect. The purpose for shielding the number of core-cache clusters is two-fold. First, the shielding mitigates interconnect saturation issues caused by repetitive traffic over an on-die (or package) interconnect. Second, the shielding avoids repeated reconfiguration of a home agent. More specifically, if each processor core constituted a caching agent, an “N” cluster, “M” socket system would be perceived as a N*M caching agent system to the home agents in the system. The SA essentially functions by assuming the burden of local coherency and distributing coherency responsibilities to the core-cache clusters themselves. Otherwise, each time a core-cache cluster is modified or added, the home agent would need to be re-designed. Besides the shielded aggregation functionality, the scalability agent protocol is designed for handling local coherence between core-cache clusters on the same IC.”) Sistla at [0021] (“As will be described below, read/write/snoop flows may be modified to preserve coherence while conforming to the principles described before. The interface between SA and core-cache clusters is designed to assist in the implementation of the above-described

U.S. Patent No. 11,841,803	References
	policies. In addition, LLC flows are modified to deal with hits which require snoops to other clusters, misses and write backs (WBs) which miss the LLC (they can since the line could be located in a LLC in another cluster). The policies are designed to avoid replication and maintain a single copy of each line in the socket.”) Sistla at [0022] (“Referring now to FIG. 4, shown is a flow diagram of a method in accordance with one embodiment of the present invention. More specifically, as shown in FIG. 4 method 100 may be used to illustrate cache management policies for requests that hit in a local cache associated with a requesting core and a miss in such a cache. Referring now to FIG. 4, method 100 may begin by determining whether a core request hits in the local cache (diamond 120). If a request hits the local cache and only the local cluster bits are set (as determined at diamond 125), a local cross snoop could be launched or the line may be returned to the requesting core without cross snoops (block 130). If a local cluster bit is set along with some other cluster bits (or just other bits are set) control passes to diamond 135. If the current request is a read, then the current request is a hit, no request to SA is needed. Thus, appropriate local core and local cluster bits are set before data is delivered to the core (block 140). Otherwise, if the current request is a request for ownership (RFO), then a miss request with xsnp_only bit set is sent to the appropriate SA, which may be based on an address hash (block 145). It is possible that new data may arrive because of the xsnp launched to the other cluster through the SA. Once the xsnp completes, cached data or new data is delivered to the requesting core. Then it is filled into the local cache with just the local cluster bit and the requesting core's local core bit set (again at block 140).”) Sistla at [0023] (“If instead at diamond 120 a miss is determined, the following cache miss policies are followed. A miss request is launched to the SA with the xsnp_only bit reset (block 150). When the SA completes the transaction, it will send a bit called local_data. If local_data bit is set (as determined at diamond 155), then data is delivered to the core and is not filled into the local cache (block 160). If the local_data bit is not set then data is delivered to the core and the line is filled into the local cache, and the local cluster and requesting core's local core bit are set (block 165). Note that the SA uses local_data bit to control replication of data in other caches. While shown with this particular implementation in the embodiment of FIG. 4, the scope of the present invention is not limited in this regard.”)

U.S. Patent No. 11,841,803	References
	Sistla at [0024] (“From the socket perspective, transactions can be classified into two broad classes: outgoing transactions and incoming transactions. For both classes of transactions, the scalability agent plays the role of an on-die snooping agent and snoop response aggregator. The scalable core-cache cluster aggregation architecture presented by embodiments of the present invention can be used by processors with larger numbers of cores by aggregating core-cache clusters. The on-die snooping functionality of the scalability agents ensures a low latency cache-to-cache transfer between core-cache clusters on the same die. Note that the on-die interconnect latency and bandwidth are non-critical since the traffic from processor cores is filtered through a shared cache.”) Sistla at [0025] (“The following eviction policies may be followed in one embodiment. On a local cache eviction, cluster bits are sent to the SA along with the eviction request (which is a WB message type). In addition the wr_miss bit is set to zero. The following write miss policies are followed. It is possible that a WB from the core will miss the local cache. This is because the line could be residing in another core-cache cluster. When this happens the write is issued to the SA just as an eviction, but the wr_miss bit is set. The following snoop processing policies are followed. When an incoming snoop is received, the snp_all, local_snp bits are examined. When the snp_all bit is set, it indicates that the snoop must be sent to all cores independent of the shared cache results. Note that an owner cache will never receive a snoop with snp_all bit set. If local_snp bit is set and the current snoop hits in the local cache, then appropriate local cores are snooped and respond to the SA with appropriate responses, cluster bits and the my_own bit set. Cluster bits and core bits may be updated appropriately. If local_snp bit is set and the current snoop is a miss, responses may be returned with my_own bit reset. The SA is responsible for performing on-die snooping. In addition to snooping its core-cache cluster as a result of outgoing and incoming requests, it works with the clusters to ensure that there is only a single copy of each line in the socket. It also resolves conflicts between clusters, although the conflict resolution is beyond the scope here, it is assumed that all transactions are conflict free.”) Sistla at [0026] (“Various outgoing transactions having a cluster hit are shown in Table 1, in which SA flows are demonstrated by using two SA units SA0 and SA1 and two core-cache

U.S. Patent No. 11,841,803	References
	clusters CC0 and CC1, using a modified, exclusive, shared, invalid (MESI) cache coherency policy. Local cross snoops are not shown, only the flows through the SA are described.”) Sistla at [0027] (“Referring to Table 1, a read/invalidate/own transaction (RdInvlOwn) Hit M, E in CC0, some local cores bits and some other cluster bits are set. This implies that the line can either be S or I in other cores and clusters. The same flow applies when one or more other cluster bits are set, no local cores bits are set and the other cluster cache line state is E/S/I. CC0 will send a miss request to SA0 with the xnp_only bit set to 1 and cluster bits set to 11. CC0 will send a miss request to SA0 with the xnp_only bit set to 1 and cluster bits set to 11. CC1 will miss its cache, but will snoop all its cores and respond with RspI and my_copy set to zero. SA0 receives an invalidate response (RspI) from CC1 with my_copy set to 0. It will send a complete GntE_Cmp to CC0 with local_data set to zero. Since local_data is set and this is a hit request, data is delivered to core and the cluster bit is reset for CC1 in cache.”) Sistla at [0028] (“Referring still to Table 1, assume the RdInvlOwn transaction hits S. Some or none of the local core bits are set, one or more other cluster bits are set. This is essentially a socket miss and the socket miss request is sent after cleaning the socket state. CC0 will send a miss request to SA0 with the xnp_only bit set to 0 and cluster bits set to 01. SA0 examines cluster bits and sends a snoop to CC1 with local_snp set and snp_all set to 1. CC1 will miss its cache, but will snoop all its cores and respond with RspI and my_copy set to zero. SA0 receives the RspI from CC1 with my_copy set to 0. It will then send a source broadcast on a system interconnect. Eventually home logic will send a completion to the requesting cluster. Since local_data is set to zero and this is a miss request, data is delivered to core and the line is filled into Cache0. Cluster bits are initialized to 10 and appropriate cores bits are set.”) Sistla at [0029] (“Referring to Table 1, a read code/data transaction (RdCode/RdData) hits M, E in CC0. Only just one other cluster bit is set. The line state is I in the other cluster. CC0 will send a miss request to SA0 with the xnp_only bit set to 1 and cluster bits set to 01. SA0 examines cluster bits and sends a snoop to CC1 with local_snp set and snp_all set to 1. CC1 will miss its cache, but will snoop all its cores and respond with RspI and my_copy set to zero. SA0 receives the RspI from CC1 with my_copy set to 0. It will send a complete GntE_Cmp to CC0 with local_data set to zero. Since local_data is set and this is a hit request, data is delivered to core and the cluster bit is reset for CC1 in cache.”)

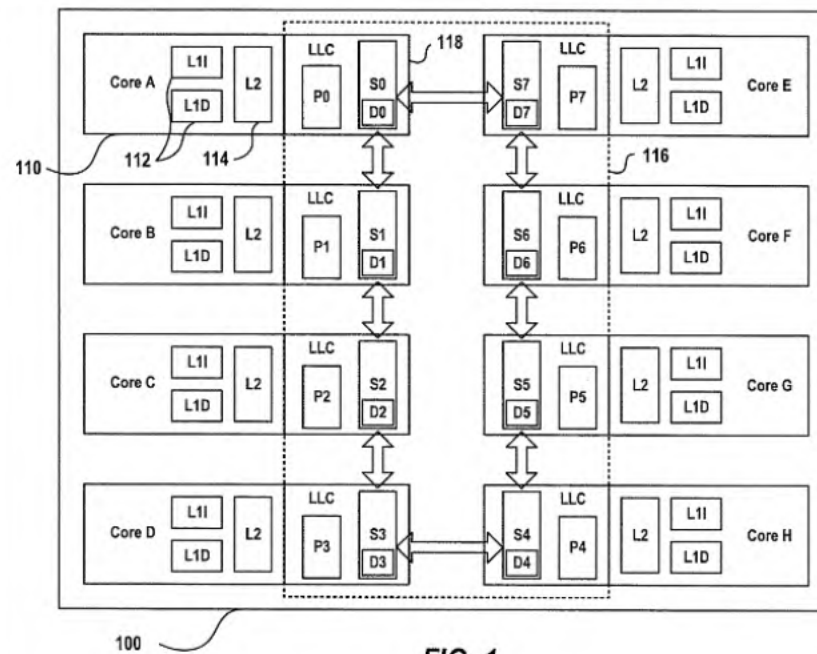
U.S. Patent No. 11,841,803	References
	Sistla at [0030] (“Referring to Table 1, a RdCode/RdData transaction hits M, E in CC0. Only just one other cluster bit is set. The line state is E,S in the other cluster. CC0 will send a miss request to SA0 with the xnp_only bit set to 1 and cluster bits set to 11. SA0 examines cluster bits and sends a snoop to CC1 with local_snp set and snp_all set to 1. CC1 will miss its cache, but will snoop all its cores and respond with RspS and my_copy set to zero. SA0 receives the RspS from CC1 with my_copy set to 0. It will send a complete GntS_Cmp to CC0 with local_data set to zero. Since local_data and this is a hit request, data is delivered to core and the local cluster and local core bits are set for the line.”) Sistla at [0031] (“Referring to Table 1, a RdCode/RdData/RdInvlOwn transaction hits M, E in CC0. Only just one other cluster bit is set. The line state is M in the other cluster. CC0 will send a miss request to SA0 with the xnp_only bit set to 1 and cluster bits set to 01. SA0 examines cluster bits and sends a snoop to CC1 with local_snp set and snp_all set to 1. CC1 will miss its cache, but will snoop all its cores and respond with RspFwdI and my_copy set to zero. SA0 receives the RspFwdI from CC1 with my_copy set to 0. It will send a complete GntE_Cmp to CC0 with local_data set to zero. Since local_data and this is a hit request, data is delivered to core and the local cluster bit and appropriate local core bit set for the line.”) Sistla at [0032] (“Shown in Table 2 below are examples of outgoing transactions on a cluster miss in accordance with one embodiment of the present invention.”) Sistla at [0033] (“Referring to Table 2, assume a Local Cluster Miss and a Remote Cluster Miss. In this case an external transaction is required. CC0 will send a miss request to SA0 with the xnp_only bit set to 0 and the rd_miss bit set. When rd_miss is set cluster bits are irrelevant, SA0 examines cluster bits and sends a snoop to CC1 with local_snp set and snp_all set to 0. CC1 will miss its cache, since snp_all is set 0. It will not snoop its core but return a RspI to the SA with my_copy set to zero. SA0 receives the RspI from CC1 with my_copy set to 0. This implies that CC1 did not have the line. It will then send a source broadcast on the system interconnect. Eventually home logic will send a complete to the requesting cluster. Since local_data is set to zero and this is a miss request, data is delivered to core and the line is filled into Cache0. Cluster bits are initialized to 10 and appropriate cores bits are set.”)

U.S. Patent No. 11,841,803	References
	Sistla at [0034] (“Referring to Table 2, a RdCode/Rd Data Local Cluster has a Miss and the Remote Cluster M/E/S with only its local core bits set and no other cluster bits set. CC0 will send a miss request to SA0 with the xnp_only bit set to 0 and the rd_miss bit set. SA0 examines cluster bits and sends a snoop to CC1 with local_snp set and snp_all set to 0. CC1 will hit in its cache. It will examine its local core bits and complete appropriate processing. Then it will respond with RspFwdS to SA0 and send data directly to requester. The cluster bit of CC0 is set in CC1. Along with the RspFwdS, my_copy bit is set and cluster bits are shipped to SA0. SA0 receives the RspFwdS along with my_copy set. It examines cluster bits from CC1 and decides if it needs to snoop the requesting/other clusters. In this case, no further snoops are needed. It sends a Cmp to CC0 with local_data set to 1. CC0 will examine local_data is set, it will deliver data to the requesting core and will not fill it into its cache.”) Sistla at [0035] (“Referring still to Table 2, a RdInvlOwn transaction misses in the Local Cluster, Remote Cluster M/E with only its local core bits set and no other cluster bits set. CC0 will send a miss request to SA0 with the xnp_only bit set to 0 and the rd_miss bit set. SA0 examines cluster bits and sends a snoop to CC1 with local_snp set and snp_all set to 0. CC1 will hit in its cache, it will examine its local core bits and complete appropriate processing. Then it will respond with RspFwdI to SA0 and send data directly to requester. The cluster bit of CC0 is set in CC1. Along with the RspFwdI, my_copy bit is set and cluster bits are shipped to SA0. SA0 receives the RspFwdI along with my_copy set. It examines cluster bits from CC1 and decides if it needs to snoop the requesting/other clusters. In this case, no further snoops are needed. It sends a Cmp to CC0 with local_data set to 1. CC0 will examine local_data is set, it will deliver data to the requesting core and will not fill it into its cache.”) Sistla at [0036] (“Referring still to Table 2, a RdInvlOwn transaction has a Local Cluster Miss, Remote Cluster S with only its local core bits set and no other cluster bits set. CC0 will send a miss request to SA0 with the xnp_only bit set to 0 and the rd_miss bit is set. SA0 examines cluster bits and sends a snoop to CC1 with local_snp set and snp_all set to 0. CC1 will hit in its cache, it will examine its local core bits and complete appropriate processing. In this case the line is invalidated. Then it will respond with RspI to SA0. Along with the RspI, my_copy bit is set and cluster bits are shipped to SA0. SA0 receives the RspI along with my_copy set. It

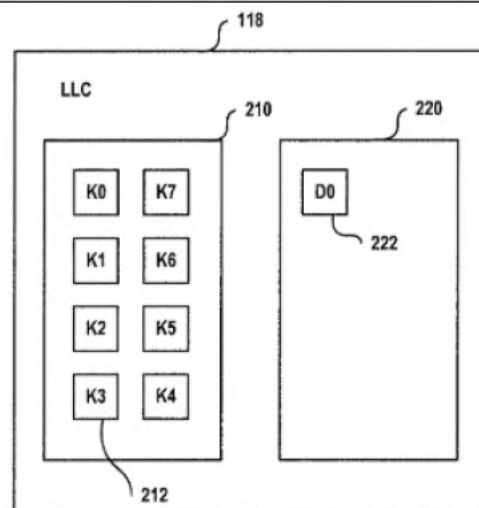
U.S. Patent No. 11,841,803	References
	examines cluster bits from CC1 and decides if it needs to snoop the requesting/other clusters. In this case, no further snoops are needed. Since the line is invalid in the socket, it sends a miss for a source broadcast. Eventually the system domain will deliver data to CC0. local_data is not set. CC0 will examine local_data is set, it will deliver data to the requesting core and fill the line into its cache. Appropriate core bit and local cluster bits are set.”) Sistla at [0037] (“Referring still to Table 2, on RdCode/RdData Local Cluster Miss, Remote Cluster M/E with no local core bits set occurs. Only requesting cluster's cluster bit is set. RspI/RspS. This is a special case which requires what can be described as cleanup snoop. In the first phase, since SA does not know which cluster is the owner, it sends a snoop to all clusters with the snp_all bit reset. Only one cluster will hit the snoop (the owner cluster) and rest will all miss. Owner cluster will send back a snoop response with cluster bits for the line and the my_own bit is set on response. When the SA receives cluster bits, it issues snoops again with the snp_all bit set for all the cluster for which the cluster bit is set. This snoop is the cleanup snoops which cleans all cluster which may contain the line. One the cleanup snoop is completed. a complete is indicated to the requesting cluster. CC0 will send a miss request to SA0 with the xnp_only bit set to 0 and. The rd_miss bit is set. When rd_miss is set cluster bits are irrelevant. SA0 examines cluster bits and sends a snoop to CC1 with local_snp set and snp_all set to 0. CC1 will hit in its cache it will examine its local core bits and complete appropriate processing. It will respond with RspFwdI to SA0. Data is shipped to the requester with a Data_E. In addition cluster bit of the requester is set (in this case it is already set). Along with the response, my_copy bit is set and the original cluster bits are shipped to SA0. SA0 receives the RspFwdI along with my_copy set. It examines cluster bits from CC1 and decides if it needs to snoop the requesting cluster. A snoop is issued to requesting cluster with the snoop all bit set. CC0 will complete the snoop and will send a RspI/RspS SA0. SA0 then sends a Cmp to CC0. CC0 now can use the data without filling into its local cache.”) Sistla at [0038] (“Referring still to Table 2, assume RdData/RdInvlOwn Local Cluster Miss, Remote Cluster M/E with no local core bits set. Only requesting cluster's cluster bit is set. Another core in the requesting cluster contains the line in M state. RspFwdI. The forwarding core sends the data directly to the requesting cluster (its own cluster). Its local cluster sends a RspFwdI to SA. In this case SA receives two RspFwdI's. The requesting cluster will receive data twice and it must discard the data from the home. Modified data is delivered to the

U.S. Patent No. 11,841,803	References
	requesting core if the request is RdInvlOwn type. If the original request is a RdData, then local cluster delivers the data to the requesting core in E state and initiates a Wr_Miss flow for the current line. CC0 will send a miss request to SA0 with the xnp_only bit set to 0 and the rd_miss bit set. SA0 examines cluster bits and sends a snoop to CC1 with local_snp set and snp_all set to 0. CC1 will hit in its cache, it will examine its local core bits and complete appropriate processing. In this case the line is demoted to S state. Then it will respond with RspFwdI to SA0. Data is shipped to the requester with a Data_E. In addition cluster bit of the requester is set. Along with the response, my_own bit is set and the original cluster bits are shipped to SA0. SA0 receives the RspFwdI along with my_own set. It examines cluster bits from CC1 and decides if it needs to snoop the requesting cluster. A snoop is issued to requesting cluster with the snoop all bit set. The snoop will complete and send a RspFwdI to SA0. Data is sent from the M core to core-cache cluster logic. This data will override the data returned from CC1. Data is now M state. CC0 initiates wr miss flow and return the data to the requester in E state. It will not fill it in its cache.”) Sistla at [0039] (“Referring still to Table 2, assume RdInvlOwn whose socket state is S. It is a miss in the local cluster and the owner cache state indicates shared state. Basically the line is cleaned up in the socket and an external request is sent. CC0 will send a miss request to SA0 with the xnp_only bit set to 0 and the rd_miss bit set. SA0 examines cluster bits and sends a snoop to CC1 with local_snp set and snp_all set to 0. CC1 will hit in its cache, it will examine its local core bits and complete appropriate processing. It will respond with RspI to SA0 and invalidate its local copy. Along with the response, my_own bit is set and the original cluster bits are shipped to SA0. SA0 receives the RspI along with my_own set. It examines cluster bits from CC1 and decides if it needs to snoop the requesting cluster. A snoop is issued to requesting cluster with the snoop all bit set. CC0 will complete the snoop and will send a RspI to SA0. SA0 then sends a source broadcast on CSI with RdInvlOwn. Eventually the system interconnect will send a DataC_E_Cmp to CC0. This line is filled into CC0 cache and delivered to the requesting core.”) Sistla at [0040] (“Referring still to Table 2, assume a write back from core which misses the local cache. SA will send a no operation (nop) snoop first. This snoop will not have any effect on the cache state. Based on responses SA can determine which cluster is the owner cluster.

U.S. Patent No. 11,841,803	References
	CC0 will send a write miss request to SA0 with the wr_miss bit set to 1 and SA0 sends cluster bits and sends a Nop snoop to CC1 with local_snp set and snp_all set to 0. CC1 will hit in its cache. Since it is a hit but is a nop, no changes are made to cache state. However a RspI is sent with my_own bit set. SA0 will now send a data_pull snoop to the requesting cluster with the id for the owner cluster. CC0 will then send a writeback to the owner cluster. Once receiving a complete from the owner cluster, it sends a de-allocate to SA.”) Sistla at [0041] (“Referring still to Table 2, an eviction from a cluster cache can occur. Since evictions can only happen from owner cache, the cluster bits are sent to SA. SA performs appropriate cleanup of other cluster caches and the eviction is then sent out to the system memory. CC0 will send a write request to SA0 with the wr_miss bit set to 0 and cluster bits are indicated. SA0 sends snoops to all clusters for which cluster bits are indicated. Local_snp is set and snp_all is set. CC1 responds to the snoop with a RspI and my_own bit is reset. Once all responses are received, data_pull is indicated to the requesting cluster. Requesting cluster can now send the WbMtoI directly to the external system. Eventually transaction is completed in the CSI domain and deallocate is sent to SA.”) Sistla at [0042] (“Referring still to Table 2, incoming snoop transactions end up performing the two snoop steps. First the request snoop is performed then the cleanup snoop is performed. Incoming snoops gets allocated into SA0. SA0 sends snoops to all clusters with local_snp set to zero and snp_all is set to zero. Owner cluster will respond with a RspFwdI and my_own bit is set. In this case it is CC0. CC1 sends a RspI. SA0 sends a combined RspFwdI to home for the incoming snoops. It then sends a data pull message to CC0 (owner cluster). CC0 on receipt of the data pull message sends the data message to the requester of the original snoop.”) <u>Jiang</u> Jiang at Fig. 1



Jiang at Fig. 2

**FIG. 2**

Jiang at Fig. 8

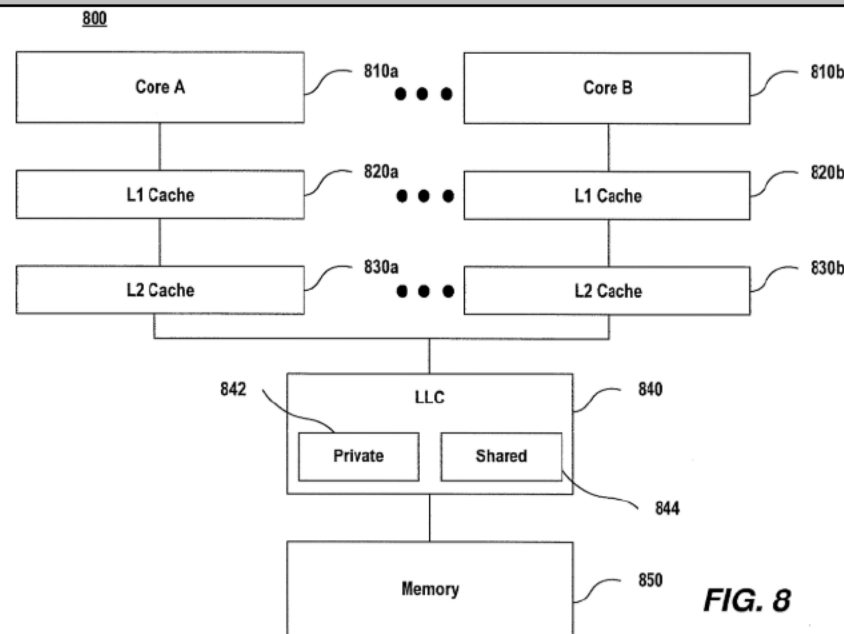


FIG. 8

Jiang at Abstract (“A multi-core CPU includes a Last-Level Cache (LLC) interconnected with a plurality of cores. The LLC may include a shared portion and a private portion. The shared portion may be shared by the plurality of cores. The private portion may be connected to a first core of the plurality of cores and may be exclusively assigned to the first core. The first core may be configured to initiate a data access request to access data stored in the LLC and initiate a data access request to access data stored in the LLC. The first core may route the data access request to the private portion based on the determination that the data access request is the TLS type of access request and route the data access request to the shared portion based on the determination that the data access request is not the TLS type of access request.”)

Jiang at [0001] (“The present application relates to computer architecture, and more particularly, to central processing units (CPUs) having private caches for thread local storage (TLS) type of data access and data accessing methods performed by such CPUs utilizing the private caches.”)

U.S. Patent No. 11,841,803	References
	Jiang at [0002] (“Commercial CPUs, such as a CPU designed based on the x86 architecture, usually adopts a hierarchical caching structure between CPU core(s) and the main memory (e.g., a dynamic random-access memory (DRAM)). The caching structure may include multiple levels of caches, ranging from fast but small (i.e., low storage capacity) lower level caches to slow but large (i.e., high storage capacity) higher level caches. Data accessing (e.g., loading/reading data from the memory or storing/writing data to the memory) is performed by the CPU through the multi-level caches to reduce latency and improve data accessing speed. For example, when the CPU performs a read/load operation, the CPU first seeks a copy of the data in the lowest level cache. If the copy is found (also referred to as a “hit”), then the CPU fetches the copy from the lowest level cache without reaching out to the higher level caches or the main memory. If the copy is not found in the lowest level cache (also referred to as a “miss”), then the CPU seeks the copy in the next, higher level cache, and repeats the process through all levels of caches. If the data is not found in all levels of caches, the CPU then fetches the data from the main memory.”) Jiang at [0003] (“Among the multiple level of caches, the last-level cache (LLC) is usually at the third level in the x86 architecture so it may also be referred to as an L3 cache. An LLC usually has a few tens of megabytes (MBs) in size, and causes relatively long access latency compared to the lower level caches (e.g., level-one (L1) and level-two (L2) caches). In existing commercial (e.g., off-the-shelf) multi-core CPUs, the LLC is usually split into smaller slices and distributed and interconnected among multiple CPU cores. For example, each slice is attached to one core. A single piece of data is stored in a distributed manner across multiple slices, and thus across multiple cores. This design allows the entire LLC capacity to be shared by multiple cores. But it also increases the LLC hit latency and increases power consumption. This is because a CPU core has to undergo multiple hops to fetch portions of the same piece of data from LLC slices attached to other cores, regardless of whether the data is private to a single core or not. The multiple hops cause latency and consume power, which are significant considering the accumulating effect in a data center environment where hundreds of thousand CPUs operating all of the time.”)

U.S. Patent No. 11,841,803	References
	Jiang at [0004] (“In some aspects, the present disclosure is directed to a CPU. The CPU may include a plurality of cores. The CPU may also include an LLC interconnected with the plurality of cores. The LLC may include a shared portion and a private portion. The shared portion may be shared by the plurality of cores. The private portion may be connected to a first core of the plurality of cores and may be exclusively assigned to the first core. The first core may be configured to initiate a data access request to access data stored in the LLC. The first core may also be configured to determine whether the data access request is a Thread Local Storage (TLS) type of access request based on an annotation associated with the data access request. In addition, the first core may be configured to route the data access request to the private portion based on the determination that the data access request is the TLS type of access request, and route the data access request to the shared portion based on the determination that the data access request is not the TLS type of access request.”) Jiang at [0005] (“In some other aspects, the present disclosure is directed to a method, performed by a CPU including a plurality of cores, of accessing data stored in an LLC interconnected with the plurality of cores. The method may comprise initiating a data access request and determining whether the data access request is a TLS type of access request based on an annotation associated with the data access request. The method may also include routing the data access request to a private portion of the LLC in response to the determination that the data access request is the TLS type of access request. The private portion is connected to a first core and exclusively assigned to the first core. The method may further include routing the data access request to a shared portion of the LLC when in response to the determination that the data access request is not the TLS type of access request. The shared portion is shared by the plurality of cores.”) Jiang at [0006] (“In further aspects, the present disclosure is directed to a data processing system. The data processing system may include a memory storing data and a processor including a plurality of cores. The processor may be coupled to the memory through multiple levels of caches. The multiple levels of caches may include an LLC. The LLC may include a shared portion and a private portion. The shared portion may be shared by the plurality of cores. The private portion may be connected to a first core of the plurality of cores and may be exclusively assigned to the first core. The first core may be configured to initiate a data access

U.S. Patent No. 11,841,803	References
	request to access data stored in the LLC. The first core may also be configured to determine whether the data access request is a TLS type of access request based on an annotation associated with the data access request. In addition, the first core may be configured to route the data access request to the private portion based on the determination that the data access request is the TLS type of access request, and route the data access request to the shared portion based on the determination that the data access request is not the TLS type of access request.”) Jiang at [0010] (“FIG. 1 is a schematic diagram of an exemplary CPU having multiple cores, according to some embodiments of the present application.”) Jiang at [0011] (“FIG. 2 is a schematic diagram of an exemplary LLC slice including a private portion and a shared portion, according to some embodiments of the present application.”) Jiang at [0017] (“FIG. 8 shows an exemplary data processing system, according to some embodiments of the present application.”) Jiang at [0019] (“As described in the Background section, in commercial x86 CPUs the entire storage capacity of the LLC is shared among multiple cores. Accessing data private to a single core in such a design is unnecessarily slow and also consumes extra power due to the multiple hops the accessing signal has to undergo. The solution disclosed in the present application provides local LLC caching for TLS data, which are most likely or even guaranteed to be private to a single core. TLS type of data accesses also represent the majority of private data accesses. Therefore, by reducing interconnect traffic among the cores, the disclosed solution can improve CPU performance by reducing data access latency, and reduce power consumption.”) Jiang at [0020] (“FIG. 1 is a schematic diagram of an exemplary CPU 100, in accordance with some embodiments disclosed in this application. CPU 100 may include a plurality of cores, such as cores A, B, . . . , H. Other embodiments may include less than or more than eight cores. CPU 100 may also be referred to as a multi-core CPU. A multi-core CPU refers to a single computing component with two or more independent actual processing units, each of the independent actual processing unit is called a core. A CPU core can read and execute

U.S. Patent No. 11,841,803	References
	program instructions, such as add, move data, branch, etc., similar to a single-core CPU. One benefit of the multi-core CPU is that it can run multiple instructions on separate cores at the same time, increasing overall speed for programs amenable to parallel computing. Such programs may implement parallel computing through multithreading techniques, in which small sequence of programmed instructions (called “threads”) can be executed concurrently on multiple core.”) Jiang at [0021] (“Program instructions and data can be stored in a memory, such as the main memory of a computer system including a dynamic random-access memory (DRAM). A CPU, such as CPU 100, may read instructions and data from the memory, execute the instructions and/or process the data based on the instructions, and/or write the execution results or processed data back to the memory. However, the operation speed of the memory is usually much slower than the operation speed of the CPU. Therefore, if every data accessing operation such as read/load and write/store is from the memory, the CPU has to wait for the completion of the data accessing operation in order to process the data or proceed to the next instruction, causing a huge latency. To alleviate this program, multiple levels of caches are added between the CPU and the memory, ranging from small (in terms of storage capacity) but fast lower level caches that are closer to the CPU side to the large (in terms of storage capacity) but slow (but sill faster than the speed of the main memory) higher level caches that are closer to the memory side.”) Jiang at [0022] (“FIG. 1 shows an example of this hierarchy of caches. Referring to FIG. 1, three levels of caches are provided to CPU 100. The lowest level cache, referred to as L1 (reference number 112), may be split into two parts: an instruction cache L1I for caching instructions and a data cache L1D for caching data. In other embodiments, an L1 cache may be used for caching both instructions and data without separating the cache into two parts. Higher level caches are usually not split, although such a split design may be used in other embodiments. The next level is referred to as a level 2 cache, or L2 cache, denoted as reference number 114 in FIG. 1. As shown in FIG. 1, both L1 and L2 caches are dedicated to their corresponding core 110 and not shared among the cores. In hardware implementation, L1 cache 112 may be built onto the CPU chip itself, while L2 cache may be built onto the CPU chip or provided as a separate chip. Accordingly, the schematic shown in FIG. 1 should be

U.S. Patent No. 11,841,803	References
	understood as showing a functional or logical relationship among the components, rather than a physical arrangement or hardware implementation of these components.”) Jiang at [0023] (“The next level cache is referred to as a level 3 cache or L3 cache. In FIG. 1, the L3 cache is the last-level cache (LLC). In some embodiments, LLC may be at an even higher level, such as level 4 (“L4”). The solution disclosed in this application is also applicable to a level 4 LLC or LLC at an even higher level.”) Jiang at [0024] (“The LLC may have a few tens of megabytes (MBs) in size, much larger than that of the L2 cache. Referring to FIG. 1, LLC 116 may be split into smaller slices, each attached to a single core. For example, an LLC slice 118 is attached to core A 110. For simplicity, hereinafter a CPU core is generally referred to as core 110, which may be any core of CPU 100, not necessarily core A. Similarly, L1 and L2 caches are generally referred to as 112 and 114, respectively. For the LLC, reference number 116 refers to the entire LLC while reference 118 refers to an LLC slice, which may be any one slice of the LLC 116, not necessarily the slice attached to core A. As used herein, the term “attach” means logically assigned to. For example, in FIG. 1, LLC slice 118 is logically assigned to core A, which may include connecting to core A with signal lines. While the LLC slices in FIG. 1 are separately drawn to attach to different cores, the entire LLC may be implemented as a single chip with its storage capacity logically divided into slices.”) Jiang at [0025] (“LLC 116 may include a private portion and a shared portion. Referring to FIG. 1, the private portion, as shown in each LLC slice, is denoted as P0, P1, P2, . . . , P7, while the shared portion is denoted as S0, S1, S2, . . . , S7. The shared portion is shared among multiple cores. For example, the shared portion may be distributed and interconnected among the multiple cores. The shared portion may be cache-line interleaved. Therefore, when a data structure D is stored in the shared portion of LLC 116, D is stored in 8 cache lines (D0 through D7), and are distributed and interleaved among all 8 LLC slices. If core A needs to read D4, which is stored in the shared portion S4, the reading operation has to undergo multiple hops (e.g., the round trip of S0-S1-S2-S3-S4-S3-S2-S1-S0) to fetch D4. If D is private to core A, e.g., D is only needed by the thread executed by core A but not other cores, storing D in the shared portion is not efficient.”)

U.S. Patent No. 11,841,803	References
	Jiang at [0026] (“On the other hand, a private portion of the LLC is connected to a single core and is exclusively assigned to that core. For example, in FIG. 1, private portion P0 is connected to core A and stores data only for core A. The private portion may also be referred to as a local LLC slice. While private caching is efficient in terms of accessing private data, commercial CPUs such as x86 CPUs avoid using private caches due to several reasons: first, caching data in private portions or local LLC slices is in general an inefficient way of utilizing the LLC storage capacity because some threads may demand large cache capacity while others may not utilize the cache space that much; second, private caching also creates a data coherence issue, as the same data may have multiple version separately cached in multiple local LLC slices. Therefore, because commercial CPUs are designed to handle a wide variety of tasks, the advantages of private caching are outweighed by its drawbacks.”) Jiang at [0027] (“In a data center environment, however, commercial CPUs aimed to achieve general applicability are not optimal choices because of the tradeoffs such as omitting the private caches. This is because the minor disadvantage of increased latency and power consumption caused by the multi-hop data accessing operations is amplified by hundreds of thousands or millions of CPUs operating non-stop around the clock. Therefore, to meet the performance and the power consumption requirements, customized CPUs, rather than off-the-shelf CPUs, are preferred. One customization is to include private caching techniques to accelerate the data accessing speed of private data, while maintaining the advantages provided by the shared caches.”) Jiang at [0028] (“FIG. 2 shows the exemplary LLC slice 118 that includes a private portion 210 and a shared portion 220. Private portion 210 is exclusively assigned to the connected core (core A). Instead of using the cache-line interleaved design as in the shared portion, all cache lines 212 (K0, K1, K2, . . . , K7) are local to private portion 210, whereas only cache line D0 is local to shared portion 220. Accordingly, a data structure private to core A can be stored entirely in the private portion 210 and accessed by core A without traversing the multiple hops as described above in the context of the shared LLC portion(s).”)

U.S. Patent No. 11,841,803	References
	Jiang at [0029] (“While it is difficult to determine, in general, what kind of data is private to a CPU core, there are certain types of data that are most likely private. For example, a thread local storage (TLS) type of data is most likely to be private to each thread running on a CPU core and is also known to represent the majority of private data. Therefore, embodiments of the present application disclose a CPU cache accessing solution in which a CPU core routes all TLS type of data access requests that miss the L2 cache to the private portion or local slice of the LLC, while routing non-TLS accesses to the shared portion of the LLC. By routing TLS type of data access requests, which are most likely or even guaranteed private to the thread running on a single CPU core, to the private portion of the LLC that is connected to and exclusively assigned to that CPU core, higher performance and lower power consumption can be achieved due to reduced LLC hit latency and reduced interconnect traffic.”) Jiang at [0039] (“FIG. 8 shows an exemplary data processing system 800. System 800 may include a processor having a plurality of cores (e.g., core A 810 a, core B 810 b, etc.). The processor may be coupled with a memory 850 through multiple levels of caches (e.g., L1 cache 820 a/820 b, L2 cache 830 a/830 b, LLC 840, etc.). LLC 840 may include a private portion 842 and a shared portion 844. As discussed above, private portion 842 may include individual slices that are each connected to a corresponding core and exclusively assigned to that core, while the shared portion may be shared by the plurality of cores. Each core may perform data accessing operations utilizing LLC 840 according to, for example, method 700.”) Jiang at [0040] (“The specification has described multi-core CPUs having private caches for thread local storage (TLS) type of data access and data accessing methods performed by such CPUs utilizing the private caches. The illustrated steps are set out to explain the exemplary embodiments shown, and it should be anticipated that ongoing technological development will change the manner in which particular functions are performed. Thus, these examples are presented herein for purposes of illustration, and not limitation. For example, steps or processes disclosed herein are not limited to being performed in the order described, but may be performed in any order, and some steps may be omitted, consistent with disclosed embodiments. Further, the boundaries of the functional building blocks have been arbitrarily defined herein for the convenience of the description. Alternative boundaries can be defined as long as the specified functions and relationships thereof are appropriately performed. Alternatives (including equivalents, extensions, variations, deviations, etc., of those described

U.S. Patent No. 11,841,803	References
	herein) will be apparent to persons skilled in the relevant art(s) based on the teachings contained herein. Such alternatives fall within the scope and spirit of the disclosed embodiments.”) Jiang at [0041] (“While examples and features of disclosed principles are described herein, modifications, adaptations, and other implementations are possible without departing from the spirit and scope of the disclosed embodiments. Also, the words “comprising,” “having,” “containing,” and “including,” and other similar forms are intended to be equivalent in meaning and be open ended in that an item or items following any one of these words is not meant to be an exhaustive listing of such item or items, or meant to be limited to only the listed item or items. It must also be noted that as used herein and in the appended claims, the singular forms “a,” “an,” and “the” include plural references unless the context clearly dictates otherwise.”) Jiang at [0042] (“Embodiments of the present application may be implemented using hardware, software, firmware, or any combination thereof for allowing a specialized device to perform the functions described above. One or more steps, operations, functions, and modules described herein may be implemented by firmware instructions or codes stored in one or more memory devices and executed by one or more hardware processor devices. Exemplary hardware processor devices include logic gate circuitry configured or made to perform data processing and/or logic operations, integrated circuits (ICs) designed and made to perform specific functions, programmable gate arrays (PGAs), field-programmable gate arrays (FPGAs), etc.”) Jiang at [0043] (“Multiple function modules may be integrated into a single physical device, or may be provided in separate physical devices.”) Jiang at [0044] (“It is intended that the disclosure and examples be considered as exemplary only, with a true scope and spirit of disclosed embodiments being indicated by the following claims.”) This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or

U.S. Patent No. 11,841,803	References
	(3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
8[B]: wherein a first level of the first cache memory hierarchy is coherent within the first GPU chiplet; and	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <i>See</i> evidence cited for '803 patent limitation 8[A]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
8[C]: a second cache memory hierarchy at the second GPU chiplet,	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <i>See</i> evidence cited for '803 patent limitation 8[A]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
8[D]: wherein a first level of the second cache memory hierarchy is coherent within the second GPU chiplet.	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <i>See</i> evidence cited for '803 patent limitations 8[A] and 8[B]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or

U.S. Patent No. 11,841,803	References
	(3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
9[A]: 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,	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <i>See evidence cited for '803 patent Claim 8.</i> This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
9[B]: wherein the unified cache memory is coherent across all GPU chiplets of the GPU chiplet array.	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <i>See evidence cited for '803 patent Claim 8.</i> This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA's other invalidity charts and cover pleading.
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.	Including at least under Complainant's apparent infringement theory, the prior art discloses and/or renders obvious this limitation. <i>See, e.g.,</i> <u>Elsherbini</u>

Elsherbini at Fig. 20

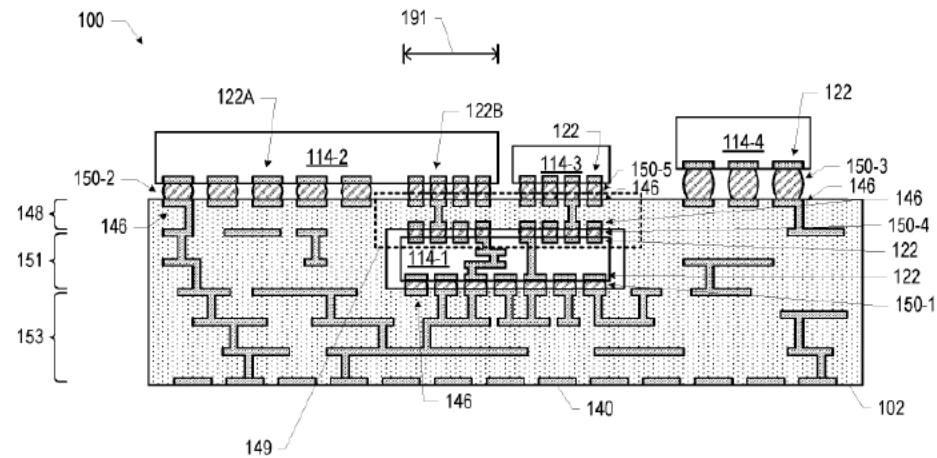


FIG. 20

Elsherbini at Fig. 24B

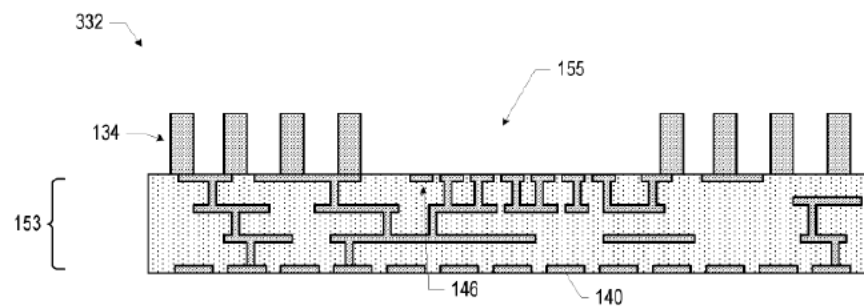


FIG. 24B

Elsherbini at Fig. 24D

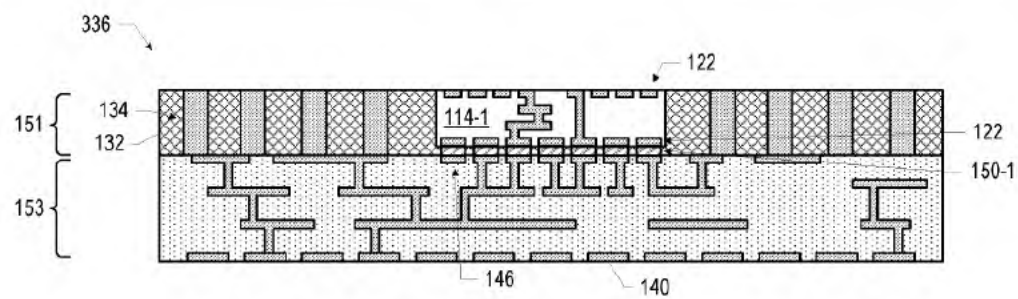


FIG. 24D

Elsherbini at Fig. 25A

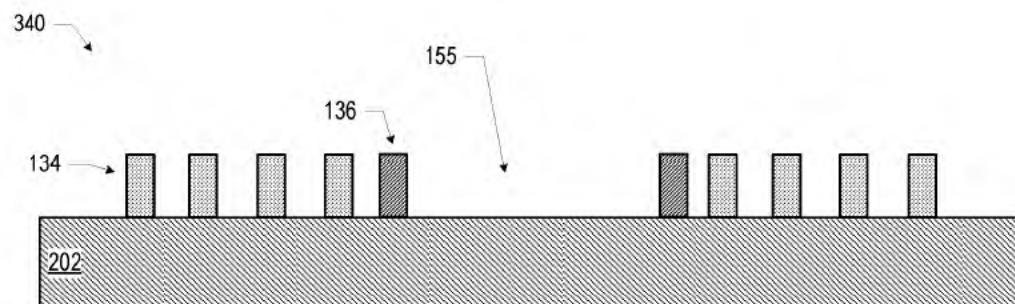


FIG. 25A

Elsherbini at Fig. 25C

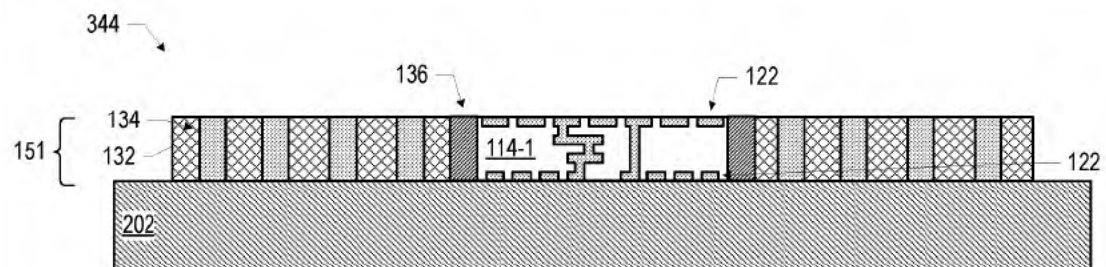


FIG. 25C

Elsherbini at Fig. 26A

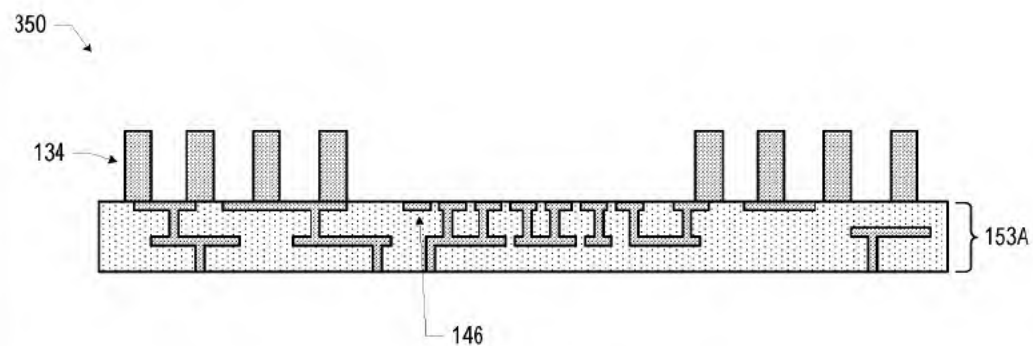


FIG. 26A

Elsherbini at Fig. 26B

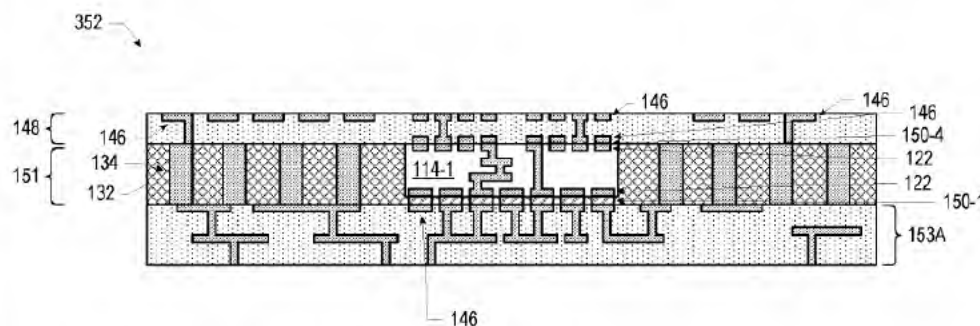


FIG. 26B

Elsherbini at [0096]-[0098] (“FIG. 24B illustrates an assembly 332 subsequent to forming conductive pillars 134 on the top surface of the package substrate portion 153 of the assembly 330 (FIG. 24A). The conductive pillars 134 may be disposed around a de-population region 155 in which no conductive pillars 134 are present. The conductive pillars 134 may take the form of any of the embodiments disclosed herein, and may be formed using any suitable technique (e.g., plating). For example, the conductive pillars 134 may include copper.

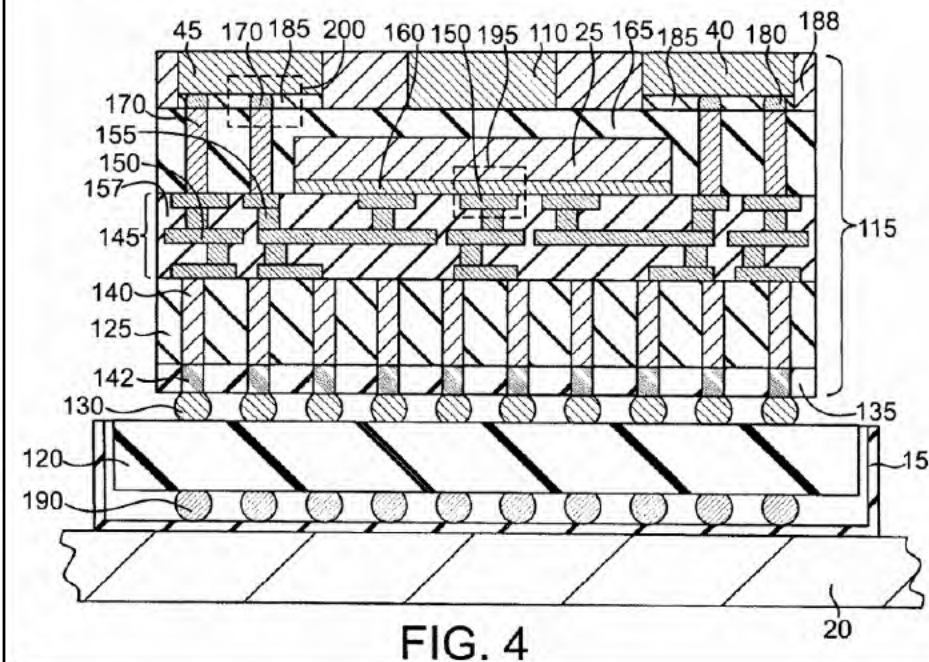
FIG. 24C illustrates an assembly 334 subsequent to placing the die 114-1 in the depopulation region 155 of the assembly 332 (FIG. 24B) and coupling the die 114-1 to the package substrate portion 153. In particular, the conductive contacts 122 at the bottom surface of the die 114-1 may be coupled to the conductive contacts 146 at the top surface of the package substrate portion 153 via DTPS interconnects 150-1. The DTPS interconnects 150-1 may take any of the forms disclosed herein, such as solder interconnects or anisotropic conductive material interconnects.

U.S. Patent No. 11,841,803	References
	FIG. 24D illustrates an assembly 336 subsequent to providing a mold material 132 around the die 114-1 and the conductive pillars 134 of the assembly 334 (FIG. 24C) to complete the package substrate portion 151. In some embodiments, the mold material 132 may be initially deposited on and over the tops of the conductive pillars 134 and the die 114-1, then polished back to expose the conductive contacts 122 at the top surface of the die 114-1, and the top surfaces of the conductive pillars 134.”). Elsherbini at [0101] (“FIG. 25A illustrates an assembly 340 subsequent to forming a plurality of conductive pillars 134 and a placement ring 136 on a carrier 202. The conductive pillars 134 may take any of the forms disclosed herein, and may be formed using any suitable technique (e.g., the techniques discussed above with reference to FIG. 24B). The placement ring 136 may take any of the forms disclosed herein, and may be formed using any suitable technique (e.g., any of the techniques disclosed herein). The placement ring 136 may surround a de-population region 155 in which no conductive pillars 134 are present.”). Elsherbini at [0103] (“FIG. 25C illustrates an assembly 344 subsequent to providing a mold material 132 around the conductive pillars 134 and placement ring 136 of the assembly 342 (FIG. 25B) to complete the package substrate portion 151. In some embodiments, the mold material 132 may be initially deposited on and over the tops of the conductive pillars 134 and the die 114-1, then polished back to expose the conductive contacts 122 at the surface of the die 114-1, and the surfaces of the conductive pillars 134.”). Elsherbini at [0108-0109] (“FIG. 26A illustrates an assembly 350 subsequent to forming a package substrate sub-portion 153A and forming conductive pillars 134 thereon. The conductive pillars 134 may take the form of any of the embodiments disclosed herein, and the package substrate sub-portion 153A may represent the top half of the package substrate portion 153, as discussed further below. FIG. 26B illustrates an assembly 352 subsequent to attaching a die 114-1 to the assembly 350 (FIG. 26A), providing a mold material 132 around the conductive pillars 134 and the die 114-1 to complete the package substrate portion 151, and forming a package substrate portion 148 on

the package substrate portion 151. These operations may take any of the forms discussed above.”).

Bhagavat

Bhagavat at Fig. 4



Bhagavat at Fig. 6

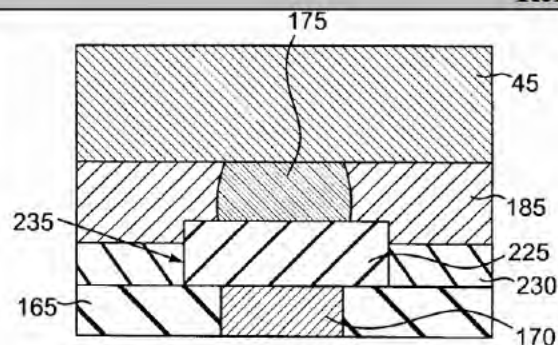


FIG. 6

Bhagavat at 6:17-39("Additional details of the electrical connections between the TDVs 170 and the chips 40 and 45 can be understood by referring now to FIG. 6, which as noted above, is the portion of FIG. 4 circumscribed by the dashed rectangle 200. Note that a portion of one of the TDVs 170 as well as the dielectric layer 165 are depicted. A conductive pillar 225 is formed on and in ohmic contact with one of the TDVs 170 and projects vertically upward beyond a dielectric film 230 composed of silicon oxide or other materials. The dielectric film 230 includes a suitable opening 235 formed therein to accommodate the conductive pillar 225. The conductive pillar 225 is advantageously formed by plating material into the opening 235 through a suitable mask (not shown) or by way of material deposition and lithographic patterning as desired. The I/O 175 of the chip 45 is preferably a solder bump or micro bump and metallurgically connected to the conductive pillar 225 by way of contact and solder reflow. Optionally, the I/O 175 can be another conductive pillar that is joined to the conductive pillar 225 by thermal bonding or by a solder cap as desired. As noted above, the underfill 185 is deposited between the chip 45 and the dielectric layer 165 using capillary techniques and to alleviate issues of CTE mismatch. Optionally a molded underfill could be used.").

Shih

U.S. Patent No. 11,841,803	References
	Shih at [0027] (“On the active face 11a of the logic die 11, input/output (I/O) pads 111 and 112 are provided. The I/O pads 112 are disposed near the edge of the logic die 11 that is close to the logic die 12. Connecting elements 113 such as metal bumps or metal pillars are formed on the I/O pads 111, respectively. According to one embodiment, the connecting elements 113 may comprise copper, aluminum, gold, or the like, but is not limited thereto. On the active face 12a of the logic die 12, likewise, I/O pads 121 and 122 are provided. The I/O pads 122 are disposed near the edge of the logic die 12 that is close to the logic die 11. Connecting elements 123 such as metal bumps or metal pillars are formed on the I/O pads 121, respectively. According to one embodiment, the connecting elements 123 may comprise copper, aluminum, ld, or the like, but is not limited thereto.”). This limitation also would have been obvious to a person of ordinary skill in the art based on (1) the prior art alone; (2) the general knowledge of a person of ordinary skill in the art; and/or (3) the prior art in combination with the teachings with respect to this claim element as detailed in NVIDIA’s other invalidity charts and cover pleading.