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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 August 2006 (31.08.2006)

PCT

(10) International Publication Number
WO 2006/089913 A1

(51) International Patent Classification:
G06F 9/455 (2006.01)

(21) International Application Number:
PCT/EP2006/060187

(22) International Filing Date:
22 February 2006 (22.02.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/066,201 25 February 2005 (25.02.2005) US

(71) Applicant *(for all designated States except US)*: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, New York 10504 (US).

(71) Applicant *(for MG only)*: **IBM UNITED KINGDOM LIMITED** [GB/GB]; Po Box 41, North Harbour, Portsmouth Hampshire PO6 3AU (GB).

(72) Inventors; and

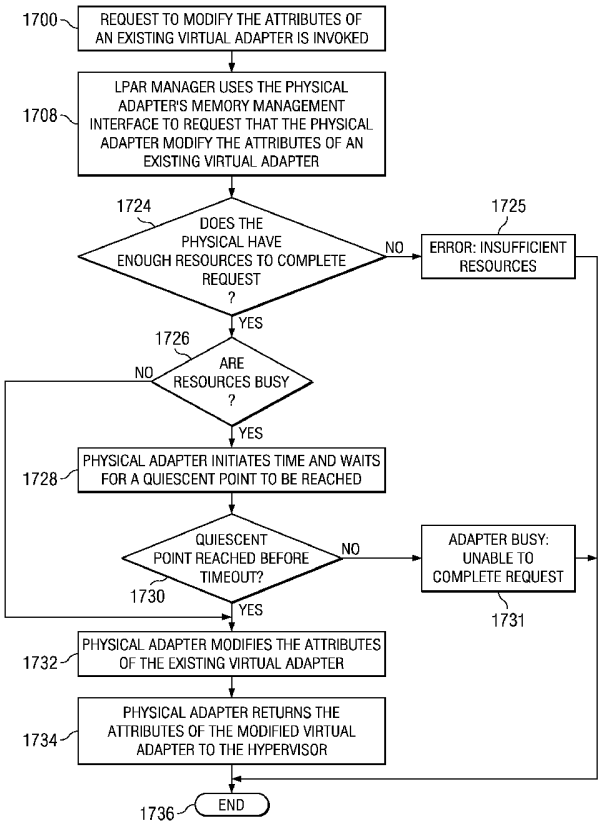
(75) Inventors/Applicants *(for US only)*: **ARNDT, Richard, Louis** [US/US]; 1607 Barn Swallow Drive, Austin, Texas 78746 (US). **BIRAN, Giora** [IL/IL]; 13 Inbar Street, 30900 Zichron-yaakov (IL). **BUCKLAND, Patrick, Allen** [US/US]; 2904 Cherry Lane, Austin, Texas 78703 (US). **KIEL, Harvey, Gene** [US/US]; 1268 Buckridge Drive Northeast, Rochester, Minnesota 55906 (US). **MAKHERVAKS, Vadim** [US/US]; 11509 Leon Grande, Austin, Texas 78759 (US). **RECIO, Renato, John** [US/US]; 6707 Winnepeg Cove, Austin, Texas 78759 (US). **SHALEV, Leah** [IL/IL]; Wingate Street, 16/b, 30900 Zichron-yaakov (IL). **SRIKRISHNAN, Jaya** [US/US]; 33 Sherwood Heights, Wappingers Falls, New York 12590 (US).

(74) Agent: **LING, Christopher, John**; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Winchester Hampshire SO21 2JN (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI,

[Continued on next page]

(54) Title: MODIFICATION OF VIRTUAL ADAPTER RESOURCES IN A LOGICALLY PARTITIONED DATA PROCESSING SYSTEM



(57) Abstract: A mechanism for modifying resources in a logically partitioned data processing system is provided. A request to modify resources associated with a virtual adapter allocated on a physical adapter is invoked. The resources associated with the virtual adapter comprise a subset of the physical adapter resources. The request to modify the physical adapter is conveyed to the physical adapter. Responsive to receipt of the request by the physical adapter, the physical adapter modifies the resources allocated to the virtual adapter.

WO 2006/089913 A1

GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

MODIFICATION 0F VIRTUAL ADAPTER RESOURCES IN A LOGICALLY PARTITIONED DATA PROCESSING SYSTEM

BACKGROUND OF THE INVENTION

Technical Field:

The present invention relates generally to communication protocols between a host computer and an input/output (I/O) adapter. In particular, the present invention provides a mechanism by which a single physical I/O adapter, such as a PCI, PCI-X, or PCI-E adapter, can modify the resources associated with one of more virtual adapters that reside within the physical adapter.

Description of Related Art:

Virtualization is the creation of substitutes for real resources. The substitutes have the same functions and external interfaces as their real counterparts, but differ in attributes such as size, performance, and cost. These substitutes are virtual resources and their users are usually unaware of the substitute's existence. Servers have used two basic approaches to virtualize system resources: partitioning and logically partitioning (LPAR) managers. Partitioning creates virtual servers as fractions of a physical server's resources, typically in coarse (e.g. physical) allocation units (e.g. a whole processor, along with its associated memory and I/O adapters). LPAR managers are software or firmware components that can virtualize all server resources with fine granularity (e.g. in small fractions of a single physical resource).

In conventional partitioned data processing systems, servers that support virtualization had two options for handling I/O. The first option was to not allow a single physical I/O adapter to be shared between virtual servers. The second option was to add functionality into the LPAR manager, or another intermediary, that provides the isolation necessary to permit multiple operating systems to share a single physical adapter.

The first option has several problems. One significant problem is that expensive adapters cannot be shared between virtual servers. If a virtual server only needs to use a fraction of an expensive adapter, an entire adapter would be dedicated to the server. As the number of virtual servers on the physical server increases, this leads to underutilization

of the adapters and a more expensive solution, because each virtual server needs at least one and potentially many physical adapters dedicated to it. For physical servers that support many virtual servers, another significant problem with this option is that it requires many adapter slots, with all the accompanying hardware (e.g. chips, connectors, cables, and the like) required to attach those adapters to the physical server and the downstream infrastructure (e.g. cables, switches, routers) to connect the additional host adapters to targets.

Though the second option provides a mechanism for sharing adapters between virtual servers, that mechanism must be invoked and executed on every I/O transaction. The invocation and execution of the sharing mechanism by the LPAR manager or other intermediary on every I/O transaction degrades performance. It also leads to a more expensive solution, because the customer must purchase more hardware, either to make up for the cycles used to perform the sharing mechanism or, if the sharing mechanism is offloaded to an intermediary, for the intermediary hardware.

It would be advantageous to have an improved method, apparatus, and computer instructions for directly modifying the resources associated with one of more virtual adapters that reside within a physical adapter, such as a PCI, PCI-X, or PCI-E adapter. It would also be advantageous to have the mechanism apply for adapters that support a memory mapped I/O interface, such as Ethernet NICs (Network Interface Controllers), FC (Fibre Channel) HBAs (Host Bus Adapters), pSCSI (parallel SCSI) HBAs, InfiniBand, TCP/IP Offload Engines, RDMA (Remote Direct Memory Access) enabled NICs (Network Interface Controllers), iSCSI adapters, iSER (iSCSI Extensions for RDMA) adapters, and the like.

SUMMARY OF THE INVENTION

The present invention provides a method, computer program product, and distributed data processing system for directly modifying the resources associated with one of more virtual adapters that reside within a physical adapter, such as a PCI, PCI-X, or PCI-E adapter. Specifically, the present invention is directed to a mechanism for sharing conventional PCI (Peripheral Component Interconnect) I/O adapters, PCI-X I/O adapters, PCI-Express I/O adapters, and, in general, any I/O adapter that uses a memory mapped I/O interface for host to adapter communications. A mechanism is provided for directly modifying the resources associated with one of more virtual adapters that reside within a physical adapter, such

as a PCI, PCI-X, or PCI-E adapter. Additionally, each virtual adapter has an associated set of host side resources, such as memory addresses and interrupt levels, and adapter side resources, such as adapter memory addresses and processing queues, and each virtual adapter is isolated from accessing the host side resources and adapter resources that belong to another virtual or physical adapter.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will now be described, by way of example only, with reference to the accompanying drawing, in which:

Figure 1 is a diagram of a distributed computer system illustrated in accordance with a preferred embodiment of the present invention;

Figure 2 is a functional block diagram of a small host processor node in accordance with a preferred embodiment of the present invention;

Figure 3 is a functional block diagram of a small integrated host processor node in accordance with a preferred embodiment of the present invention;

Figure 4 is a functional block diagram of a large host processor node in accordance with a preferred embodiment of the present invention;

Figure 5 is a diagram illustrating the elements of the parallel Peripheral Computer Interface (PCI) bus protocol in accordance with a preferred embodiment of the present;

Figure 6 is a diagram illustrating the elements of the serial PCI bus protocol (PCI-Express or PCI-E) in accordance with a preferred embodiment of the present;

Figure 7 is a diagram illustrating I/O virtualization functions provided in a host processor node in order to provide virtual host access isolation in accordance with a preferred embodiment of the present invention;

Figure 8 is a diagram illustrating the control fields used in a PCI bus transaction to identify a virtual adapter or system image in accordance with a preferred embodiment of the present invention;

Figure 9 is a diagram illustrating adapter resources that must be virtualized in order to allow: an adapter to directly access virtual host resources; allow a virtual host to directly access Adapter resources; and allow a non-PCI port on the adapter to access resources on the adapter or host in accordance with a preferred embodiment of the present invention;

Figure 10 is a diagram illustrating the creation of three access control levels used to manage a PCI family adapter that supports I/O virtualization in accordance with a preferred embodiment of the present invention;

Figure 11 is a diagram illustrating how host memory that is associated with a system image is made available to a virtual adapter that is associated with that system image through the logical partitioning manager in accordance with a preferred embodiment of the present invention;

Figure 12 is a diagram illustrating how a PCI family adapter allows a logical partitioning manager to associate memory in the PCI adapter to a system image and its associated virtual adapter in accordance with a preferred embodiment of the present invention;

Figure 13 is a diagram illustrating one of the options for determining the virtual adapter that is associated with an incoming memory address in accordance with a preferred embodiment of the present invention;

Figure 14 is a diagram illustrating one of the options for determining a virtual adapter that is associated with a PCI-X or PCI-E bus transaction in accordance with a preferred embodiment of the present invention;

Figure 15 is a diagram illustrating a virtual adapter management approach for virtualizing adapter resources in accordance with a preferred embodiment of the present invention; and

Figure 16 is a flowchart outlining an exemplary virtual adapter attribute modification routine in a data processing system implementing the virtual adapter management approach described in Figure 15 in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention applies to any general or special purpose host that uses a PCI family I/O adapter to directly attach a storage device or to attach to a network, where the network consists of endnodes, switches, routers and the links interconnecting these components. The network links can be, for example, Fibre Channel, Ethernet, InfiniBand, Advanced Switching Interconnect, or a proprietary link that uses proprietary or standard protocols. While embodiments of the present invention are shown and described as employing a peripheral component interconnect (PCI) family adapter, implementations of the invention are not limited to such a configuration as will be apparent to those skilled in the art. Teachings of the invention may be implemented on any physical adapter that support a memory mapped input/output (MMIO) interface, such as, but not limited to, HyperTransport, Rapid I/O, proprietary MMIO interfaces, or other adapters having a MMIO interface now know or later developed. Implementations of the present invention utilizing a PCI family adapter are provided for illustrative purposes to facilitate an understanding of the invention.

With reference now to the figures and in particular with reference to Figure 1, a diagram of a distributed computer system is illustrated in accordance with a preferred embodiment of the present invention. The distributed computer system represented in Figure 1 takes the form of a network, such as network 120, and is provided merely for illustrative purposes and the embodiments of the present invention described below can be implemented on computer systems of numerous other types and configurations. Two switches (or routers) are shown inside of network 120 - switch 116 and switch 140. Switch 116 connects to small host node 100 through port 112. Small host node 100 also contains a second type of port 104 which connects to a direct attached storage subsystem, such as direct attached storage 108.

Network 120 can also attach large host node 124 through port 136 which attaches to switch 140. Large host node 124 can also contain a second type of port 128, which connects to a direct attached storage subsystem, such as direct attached storage 132.

Network 120 can also attach a small integrated host node 144 which is connected to network 120 through port 148 which attaches to switch 140. Small integrated host node 144 can also contain a second type of port 152

which connects to a direct attached storage subsystem, such as direct attached storage 156.

Turning next to Figure 2, a functional block diagram of a small host node is depicted in accordance with a preferred embodiment of the present invention. Small host node 202 is an example of a host processor node, such as small host node 100 shown in Figure 1.

In this example, small host node 202, shown in Figure 2, includes two processor I/O hierarchies, such as processor I/O hierarchy 200 and 203, which are interconnected through link 201. In the illustrative example of Figure 2, processor I/O hierarchy 200 includes processor chip 207 which includes one or more processors and their associated caches. Processor chip 207 is connected to memory 212 through link 208. One of the links on processor chip, such as link 220, connects to PCI family I/O bridge 228. PCI family I/O bridge 228 has one or more PCI family (PCI, PCI-X, PCI-Express, or any future generation of PCI) links that is used to connect other PCI family I/O bridges or a PCI family I/O adapter, such as PCI family adapter 244 and PCI family adapter 245, through a PCI link, such as link 232, 236, and 240. PCI family adapter 245 can also be used to connect a network, such as network 264, through a link via either a switch or router, such as switch or router 260. PCI family adapter 244 can be used to connect direct attached storage, such as direct attached storage 252, through link 248. Processor I/O hierarchy 203 may be configured in a manner similar to that shown and described with reference to processor I/O hierarchy 200.

With reference now to Figure 3, a functional block diagram of a small integrated host node is depicted in accordance with a preferred embodiment of the present invention. Small integrated host node 302 is an example of a host processor node, such as small integrated host node 144 shown in Figure 1.

In this example, small integrated host node 302 includes two processor I/O hierarchies 300 and 303, which are interconnected through link 301. In the illustrative example, processor I/O hierarchy 300 includes processor chip 304, which is representative of one or more processors and associated caches. Processor chip 304 is connected to memory 312 through link 308. One of the links on the processor chip, such as link 330, connects to a PCI Family Adapter, such as PCI family adapter 345. Processor chip 304 has one or more PCI family (PCI, PCI-X, PCI-

Express, or any future generation of PCI) links that is used to connect either PCI family I/O bridges or a PCI family I/O adapter, such as PCI Family Adapter 344 and PCI Family Adapter 345 through a PCI link, such as link 316, 330, and 324. PCI family adapter 345 can also be used to connect with a network, such as network 364, through link 356 via either a switch or router, such as switch or router 360. PCI family adapter 344 can be used to connect with direct attached storage 352 through link 348.

Turning now to Figure 4, a functional block diagram of a large host node is depicted in accordance with a preferred embodiment of the present invention. Large host node 402 is an example of a host processor node, such as large host node 124 shown in Figure 1.

In this example, large host node 402 includes two processor I/O hierarchies 400 and 403 interconnected through link 401. In the illustrative example of Figure 4, processor I/O hierarchy 400 includes processor chip 404, which is representative of one or more processors and associated caches. Processor Chip 404 is connected to memory 412 through link 408. One of the links, such as link 440, on the processor chip connects to a PCI family I/O hub, such as PCI family I/O hub 441. The PCI family I/O hub uses a network 442 to attach to a PCI family I/O bridge 448. That is, PCI family I/O bridge 448 is connected to switch or router 436 through link 432 and switch or router 436 also attaches to PCI family I/O hub 441 through link 443. Network 442 allows the PCI family I/O hub and PCI family I/O bridge to be placed in different packages. PCI family I/O bridge 448 has one or more PCI family (PCI, PCI-X, PCI-Express, or any future generation of PCI) links that is used to connect with other PCI family I/O bridges or a PCI family I/O adapter, such as PCI family adapter 456 and PCI family adapter 457 through a PCI link, such as link 444, 446, and 452. PCI family adapter 456 can be used to connect direct attached storage 476 through link 460. PCI family adapter 457 can also be used to connect with network 464 through link 468 via, for example, either a switch or router 472.

Turning next to Figure 5, illustrations of the phases contained in a PCI bus transaction 500 and a PCI-X bus transaction 520 are depicted in accordance with a preferred embodiment of the present invention. PCI bus transaction 500 depicts the conventional PCI bus transaction that forms the unit of information which is transferred through a PCI fabric for conventional PCI. PCI-X bus transaction 520 depicts the PCI-X bus

transaction that forms the unit of information which is transferred through a PCI fabric for PCI-X.

PCI bus transaction 500 shows three phases: an address phase 508; a data phase 512; and a turnaround cycle 516. Also depicted is the arbitration for next transfer 504, which can occur simultaneously with the address, data, and turnaround cycle phases. For PCI, the address contained in the address phase is used to route a bus transaction from the adapter to the host and from the host to the adapter.

PCI-X transaction 520 shows five phases: an address phase 528; an attribute phase 532; a response phase 560; a data phase 564; and a turnaround cycle 566. Also depicted is the arbitration for next transfer 524 which can occur simultaneously with the address, attribute, response, data, and turnaround cycle phases. Similar to conventional PCI, PCI-X uses the address contained in the address phase to route a bus transaction from the adapter to the host and from the host to the adapter. However, PCI-X adds the attribute phase 532 which contains three fields that define the bus transaction requestor, namely: requestor bus number 544, requestor device number 548, and requestor function number 552 (collectively referred to herein as a BDF). The bus transaction also contains a Tag 540 that uniquely identifies the specific bus transaction in relation to other bus transactions that are outstanding between the requestor and a responder. The Byte Count 556 contains a count of the number of bytes being sent.

Turning now to Figure 6, an illustration of the phases contained in a PCI-Express bus transaction is depicted in accordance with a preferred embodiment of the present invention. PCI-E bus transaction 600 forms the unit of information which is transferred through a PCI fabric for PCI-E.

PCI-E bus transaction 600 shows six phases: frame phase 608; sequence number 612; header 664; data phase 668; cyclical redundancy check (CRC) 672; and frame phase 680. PCI-E header 664 contains a set of fields defined in the PCI-Express specification. The requestor identifier (ID) field 628 contains three fields that define the bus transaction requestor, namely: requestor bus number 684, requestor device number 688, and requestor function number 692. The PCI-E header also contains tag 652, which uniquely identifies the specific bus transaction in relation to other bus transactions that are outstanding between the requestor and a

responder. The length field 644 contains a count of the number of bytes being sent.

With reference now to Figure 7, a functional block diagram of a PCI adapter, such as PCI family adapter 736, and the firmware and software that run on host hardware (e.g. processor with possibly an I/O hub or I/O bridge), such as host hardware 700, is depicted in accordance with a preferred embodiment of the present invention.

Figure 7 also shows a logical partitioning (LPAR) manager 708 running on host hardware 700. LPAR manager 708 may be implemented as a Hypervisor manufactured by International Business Machines, Inc. of Armonk, New York. LPAR manager 708 can run in firmware, software, or a combination of the two. LPAR manager 708 hosts two system image (SI) partitions, such as system image 712 and system image 724 (illustratively designated system image 1 and system image 2). The system image partitions may be respective operating systems running in software, a special purpose image running in software, such as a storage block server or storage file server image, or a special purpose image running in firmware. Applications can run on these system images, such as applications 716, 720, 728, and 732 (illustratively designated application 1A, application 2, application 1B and application 3). Applications 716 and 728 are representative of separate instances of a common application program, and are thus illustratively designated with respective references of "1A" and "1B". In the illustrative example, application 716 and 720 run on system image 712 and applications 728 and 732 run on system image 724. As referred to herein, a virtual host comprises a system image, such as system image 712, or the combination of a system image and applications running within the system image. Thus, two virtual hosts are depicted in Figure 7.

PCI family adapter 736 contains a set of physical adapter configuration resources 740 and physical adapter memory resources 744. The physical adapter configuration resources 740 and physical adapter memory resources 744 contain information describing the number of virtual adapters that PCI family adapter 736 can support and the physical resources allocated to each virtual adapter. As referred to herein, a virtual adapter is an allocation of a subset of physical adapter resources, such as a subset of physical adapter resources and physical adapter memory, that is associated with a logical partition, such as system image 712 and applications 716 and 720 running on system image 712. LPAR manager 708 is provided a physical configuration resource interface

738, and physical memory configuration interface 742 to read and write into the physical adapter configuration resource and memory spaces during the adapter's initial configuration and reconfiguration. Through the physical configuration resource interface 738 and physical configuration memory interface 742, LPAR manager 708 creates virtual adapters and assigns physical resources to each virtual adapter. The LPAR manager 708 may use one of the system images, for example a special software or firmware partition, as a hosting partition that uses physical configuration resource interface 738 and physical configuration memory interface 742 to perform a portion, or even all, of the virtual adapter initial configuration and reconfiguration functions.

Figure 7 shows a configuration of PCI family adapter 736 configured with two virtual adapters. A first virtual adapter (designated virtual adapter 1) comprises virtual adapter resources 748 and virtual adapter memory 752 that were assigned by LPAR manager 708 that is associated with system image 712 (designated system image 1). Similarly, a second virtual adapter (designated virtual adapter 2) comprises virtual adapter resources 756 and virtual adapter memory 760 that were assigned by LPAR manager 708 to virtual adapter 2 and is associated with another system image 724 (designated system image 2). For an adapter used to connect to a direct attached storage, such as direct attached storage 108, 132, or 156 shown in Figure 1, examples of virtual adapter resources may include: the list of the associated physical disks, a list of the associated logical unit numbers, and a list of the associated adapter functions (e.g., redundant arrays of inexpensive disks (RAID) level). For an adapter used to connect to a network, such as network 120 of Figure 1, examples of virtual adapter resources may include: the list of the associated link level identifiers, a list of the associated network level identifiers, a list of the associated virtual fabric identifiers (e.g., virtual LAN IDs for Ethernet fabrics, N-port IDs for Fibre Channel fabrics, and partition keys for InfiniBand fabrics), and a list of the associated network layers functions (e.g., network offload services).

After LPAR manager 708 configures the PCI family adapter 736, each system image is allowed to only communicate with the virtual adapters that were associated with that system image by LPAR manager 708. As shown in Figure 7 (by solid lines), system image 712 is allowed to directly communicate with virtual adapter resources 748 and virtual adapter memory 752 of virtual adapter 1. System image 712 is not allowed to directly communicate with virtual adapter resources 756 and virtual adapter memory

760 of virtual adapter 2 as shown in Figure 7 by dashed lines. Similarly, system image 724 is allowed to directly communicate with virtual adapter resources 756 and virtual adapter memory 760 of virtual adapter 2, and is not allowed to directly communicate with virtual adapter resources 748 and virtual adapter memory 752 of virtual adapter 1.

With reference now to Figure 8, a depiction of a component, such as a processor, I/O hub, or I/O bridge 800, inside a host node, such as small host node 100, large host node 124, or small, integrated host node 144 shown in Figure 1, that attaches a PCI family adapter, such as PCI family adapter 804, through a PCI-X or PCI-E link, such as PCI-X or PCI-E Link 808, in accordance with a preferred embodiment of the present invention is shown.

Figure 8 shows that when a system image, such as system image 712 or 724, or LPAR manager 708, performs a PCI-X or PCI-E bus transaction, such as host to adapter PCI-X or PCI-E bus transaction 812, the processor, I/O hub, or I/O bridge 800 that connects to the PCI-X or PCI-E link 808 which issues the host to adapter PCI-X or PCI-E bus transaction 812 fills in the bus number, device number, and function number fields in the PCI-X or PCI-E bus transaction. The processor, I/O hub, or I/O bridge 800 has two choices for how to fill in these three fields: it can either use the same bus number, device number, and function number for all software components that use the processor, I/O hub, or I/O bridge 800; or it can use a different bus number, device number, and function number for each software component that uses the processor, I/O hub, or I/O bridge 800. The initiator of the transaction may be a software component, such as system image 712 or system image 724 (or an application running on a system image), or LPAR manager 708.

If the processor, I/O hub, or I/O bridge 800 uses the same bus number, device number, and function number for all transaction initiators, then when a software component initiates a PCI-X or PCI-E bus transaction, such as host to adapter PCI-X or PCI-E Bus Transaction 812, the processor, I/O hub, or I/O bridge 800 places the processor, I/O hub, or I/O bridge's bus number in the PCI-X or PCI-E bus transaction's requestor bus number field 820, such as requestor bus number 544 field of the PCI-X transaction shown in Figure 5 or requestor bus number 684 field of the PCI-E transaction shown in Figure 6. Similarly, the processor, I/O hub, or I/O bridge 800 places the processor, I/O hub, or I/O bridge's device number in the PCI-X or PCI-E bus transaction's requestor device number 824 field,

such as requestor device number 548 field shown in Figure 5 or requestor device number 688 field shown in Figure 6. Finally, the processor, I/O hub, or I/O bridge 800 places the processor, I/O hub, or I/O bridge's function number in the PCI-X or PCI-E bus transaction's requestor function number 828 field, such as requestor function number 552 field shown in Figure 5 or requestor function number 692 field shown in Figure 6. The processor, I/O hub, or I/O bridge 800 also places in the PCI-X or PCI-E bus transaction the physical or virtual adapter memory address to which the transaction is targeted as shown by adapter resource or address 816 field in Figure 8.

If the processor, I/O hub, or I/O bridge 800 uses a different bus number, device number, and function number for each transaction initiator, then the processor, I/O hub, or I/O bridge 800 assigns a bus number, device number, and function number to the transaction initiator. When a software component initiates a PCI-X or PCI-E bus transaction, such as host to adapter PCI-X or PCI-E bus transaction 812, the processor, I/O hub, or I/O bridge 800 places the software component's bus number in the PCI-X or PCI-E bus transaction's requestor bus number 820 field, such as requestor bus number 544 field shown in Figure 5 or requestor bus number 684 field shown in Figure 6. Similarly, the processor, I/O hub, or I/O bridge 800 places the software component's device number in the PCI-X or PCI-E bus transaction's requestor device number 824 field, such as requestor device number 548 field shown in Figure 5 or requestor device number 688 field shown in Figure 6. Finally, the processor, I/O hub, or I/O bridge 800 places the software component's function number in the PCI-X or PCI-E bus transaction's requestor function number 828 field, such as requestor function number 552 field shown in Figure 5 or requestor function number 692 field shown in Figure 6. The processor, I/O hub, or I/O bridge 800 also places in the PCI-X or PCI-E bus transaction the physical or virtual adapter memory address to which the transaction is targeted as shown by adapter resource or address field 816 in Figure 8.

Figure 8 also shows that when physical or virtual adapter 806 performs PCI-X or PCI-E bus transactions, such as adapter to host PCI-X or PCI-E bus transaction 832, the PCI family adapter, such as physical family adapter 804, that connects to PCI-X or PCI-E Link 808 which issues the adapter to host PCI-X or PCI-E bus transaction 832 places the bus number, device number, and function number associated with the physical or virtual adapter that initiated the bus transaction in the requestor bus number, device number, and function number 836, 840, and 844 fields. Notably, to

support more than one bus or device number, PCI family adapter 804 must support one or more internal busses (For a PCI-X Adapter, see the PCI-X Addendum to the PCI Local Bus Specification Revision 1.0 or 1.0a; for a PCI-E Adapter see PCI-Express Base Specification Revision 1.0 or 1.0a the details of which are herein incorporated by reference). To perform this function, LPAR manager 708 associates each physical or virtual adapter to a software component running by assigning a bus number, device number, and function number to the physical or virtual adapter. When the physical or virtual adapter initiates an adapter to host PCI-X or PCI-E bus transaction, PCI family adapter 804 places the physical or virtual adapter's bus number in the PCI-X or PCI-E bus transaction's requestor bus number 836 field, such as requestor bus number 544 field shown in Figure 5 or requestor bus number 684 field shown in Figure 6 (shown in Figure 8 as adapter bus number 836). Similarly, PCI family adapter 804 places the physical or virtual adapter's device number in the PCI-X or PCI-E bus transaction's requestor device number 840 field, such as requestor device number 548 field shown in Figure 5 or requestor device number 688 field shown in Figure 6 (shown in Figure 8 as adapter device number 840).PCI family adapter 804 places the physical or virtual adapter's function number in the PCI-X or PCI-E bus transaction's requestor function number 844 field, such as requestor function number 552 field shown in Figure 5 or requestor function number 692 field shown in Figure 6 (shown in Figure 8 as adapter function number 844). Finally, PCI family adapter 804 also places in the PCI-X or PCI-E bus transaction the memory address of the software component that is associated, and targeted by, the physical or virtual adapter in host resource or address 848 field.

With reference now to Figure 9, a functional block diagram of a PCI adapter with two virtual adapters depicted in accordance with a preferred embodiment of the present invention is shown. Exemplary PCI family adapter 900 is configured with two virtual adapters 916 and 920 (illustratively designated virtual adapter 1 and virtual adapter 2). PCI family adapter 900 may contain one (or more) PCI family adapter ports (also referred to herein as an upstream port), such as PCI-X or PCI-E adapter port 912. PCI family adapter 900 may also contain one (or more) device or network ports (also referred to herein as downstream ports), such as physical port 904 and physical port 908.

Figure 9 also shows the types of resources that can be virtualized on a PCI adapter. The resources of PCI family adapter 900 that may be virtualized include processing queues, address and configuration memory,

PCI ports, host memory management resources and device or network ports. In the illustrative example, virtualized resources of PCI family adapter 900 allocated to virtual adapter 916 include, for example, processing queues 924, address and configuration memory 928, PCI virtual port 936, host memory management resources 984 (such as memory region registration and memory window binding resources on InfiniBand or iWARP), and virtual device or network ports, such as virtual external port 932 and virtual external port 934 (more generally referred to as virtual ports). Similarly, virtualized resources of PCI family adapter 900 allocated to virtual adapter 920 include, for example, processing queues 940, address and configuration memory 944, PCI virtual port 952, host memory management resources 980, and virtual device or network ports, such as virtual external port 948 and virtual external port 950.

Turning next to Figure 10, a functional block diagram of the access control levels on a PCI family adapter, such as PCI family adapter 900 shown in Figure 9, is depicted in accordance with a preferred embodiment of the present invention. The three levels of access are a super-privileged physical resource allocation level 1000, a privileged virtual resource allocation level 1008, and a non-privileged level, 1016.

The functions performed at the super-privileged physical resource allocation level 1000 include but are not limited to: PCI family adapter queries, creation, modification and deletion of virtual adapters, submission and retrieval of work, reset and recovery of the physical adapter, and allocation of physical resources to a virtual adapter instance. The PCI family adapter queries are used to determine, for example, the physical adapter type (e.g. Fibre Channel, Ethernet, iSCSI, parallel SCSI), the functions supported on the physical adapter, and the number of virtual adapters supported by the PCI family adapter. The LPAR manager, such as LPAR manager 708 shown in Figure 7, performs the physical adapter resource management 1004 functions associated with super-privileged physical resource allocation level 1000. However, the LPAR manager may use a system image, for example an I/O hosting partition, to perform the physical adapter resource management 1004 functions.

The functions performed at the privileged virtual resource allocation level 1008 include, for example, virtual adapter queries, allocation and initialization of virtual adapter resources, reset and recovery of virtual adapter resources, submission and retrieval of work through virtual adapter resources, and, for virtual adapters that support

offload services, allocation and assignment of virtual adapter resources to a middleware process or thread instance. The virtual adapter queries are used to determine: the virtual adapter type (e.g. Fibre Channel, Ethernet, iSCSI, parallel SCSI) and the functions supported on the virtual adapter. A system image, such as system image 712 shown in Figure 7, performs the privileged virtual adapter resource management 1012 functions associated with virtual resource allocation level 1008.

Finally, the functions performed at the non-privileged level 1016 include, for example, query of virtual adapter resources that have been assigned to software running at the non-privileged level 1016 and submission and retrieval of work through virtual adapter resources that have been assigned to software running at the non-privileged level 1016. An application, such as application 716 shown in Figure 7, performs the virtual adapter access library 1020 functions associated with non-privileged level 1016.

Turning next to Figure 11, a functional block diagram of host memory addresses that are made accessible to a PCI family adapter is depicted in accordance with a preferred embodiment of the present invention. PCI family adapter 1101 is an example of PCI family adapter 900 that may have virtualized resources as described above in Figure 9.

Figure 11 depicts four different mechanisms by which a LPAR manager 708 can associate host memory to a system image and to a virtual adapter. Once host memory has been associated with a system image and a virtual adapter, the virtual adapter can then perform DMA write and read operations directly to the host memory. System images 1108 and 1116 are examples of system images, such as system images 712 and 724 described above with reference to Figure 7, that are respectively associated with virtual adapters 1104 and 1112. Virtual adapters 1104 and 1112 are examples of virtual adapters, such as virtual adapters 916 and 920 described above with reference to Figure 9, that comprise respective allocations of virtual adapter resources and virtual adapter memory.

The first exemplary mechanism that LPAR manager 708 can use to associate and make available host memory to a system image and to one or more virtual adapters is to write into the virtual adapter's resources a system image association list 1122. Virtual adapter resources 1120 contains a list of PCI bus addresses, where each PCI bus address in the list is associated by the platform hardware to the starting address of a

system image (SI) page, such as SI 1 page 1 1128 through SI 1 page N 1136 allocated to system image 1108. Virtual adapter resources 1120 also contains the page size, which is equal for all the pages in the list. At initial configuration, and during reconfigurations, LPAR manager 708 loads system image association list 1122 into virtual adapter resources 1120. The system image association list 1122 defines the set of addresses that virtual adapter 1104 can use in DMA write and read operations. After the system image association list 1122 has been created, virtual adapter 1104 must validate that each DMA write or DMA read requested by system image 1108 is contained within a page in the system image association list 1122. If the DMA write or DMA read requested by system image 1108 is contained within a page in the system image association list 1122, then virtual adapter 1104 may perform the operation. Otherwise virtual adapter 1104 is prohibited from performing the operation. Alternatively, the PCI family adapter 1101 may use a special, LPAR manager-style virtual adapter (rather than virtual adapter 1104) to perform the check that determines if a DMA write or DMA read requested by system image 1108 is contained within a page in the system image association list 1122. In a similar manner, virtual adapter 1112 associated with system image 1116 validates DMA write or read requests submitted by system image 1116. Particularly, virtual adapter 1112 provides validation for DMA read and write requests from system image 1116 by determining whether the DMA write or read request is in a page in system image association list (configured in a manner similarly to system image association list 1122) associated with system image pages of system image 1116.

The second mechanism that LPAR manager 708 can use to associate and make available host memory to a system image and to one or more virtual adapters is to write a starting page address and page size into system image association list 1122 in the virtual adapter's resources. For example, virtual adapter resources 1120 may contain a single PCI bus address that is associated by the platform hardware to the starting address of a system image page, such as SI 1 Page 1 1128. System image association list 1122 in virtual adapter resources 1120 also contains the size of the page. At initial configuration, and during reconfigurations, LPAR manager 708 loads the page size and starting page address into system image association list 1122 into the virtual adapter resources 1120. The system image association list 1122 defines the set of addresses that virtual adapter 1104 can use in DMA write and read operations. After the system image association list 1122 has been created, virtual adapter 1104 validates whether each DMA write or DMA read requested by system image

1108 is contained within a page in system image association list 1122. If the DMA write or DMA read requested by system image 1108 is contained within a page in the system image association list 1122, then virtual adapter 1104 may perform the operation. Otherwise, virtual adapter 1104 is prohibited from performing the operation. Alternatively, the PCI family adapter 1101 may use a special, LPAR manager-style virtual adapter (rather than virtual adapter 1104) to perform the check that determines if a DMA write or DMA read requested by system image 1108 is contained within a page in the system image association list 1122. In a similar manner, virtual adapter 1112 associated with system image 1116 may validate DMA write or read requests submitted by system image 1116. Particularly, a system image association list similar to system image association list 1122 may be associated with virtual adapter 1112. The system image association list associated with virtual adapter 1112 is loaded with a page size and starting page address of a system image page of system image 1116 associated with virtual adapter 1112. The system image association list associated with virtual adapter 1112 thus provides a mechanism for validation of DMA read and write requests from system image 1116 by determining whether the DMA write or read request is in a page in a system image association list associated with system image pages of system image 1116.

The third mechanism that LPAR manager 708 can use to associate and make available host memory to a system image and to one or more virtual adapters is to write into the virtual adapter's resources a system image buffer association list 1154. In Figure 11, virtual adapter resources 1150 contains a list of PCI bus address pairs (starting and ending address), where each pair of PCI bus addresses in the list is associated by the platform hardware to a pair (starting and ending) of addresses of a system image buffer, such as SI 2 Buffer 1 1166 through SI 1 Buffer N 1180 allocated to system image 1116. At initial configuration, and during reconfigurations, LPAR manager 708 loads system image buffer association list 1154 into the virtual adapter resources 1150. The system image buffer association list 1154 defines the set of addresses that virtual adapter 1112 can use in DMA write and read operations. After the system image buffer association list 1154 has been created, virtual adapter 1112 validates whether each DMA write or DMA read requested by system image 1116 is contained within a buffer in system image buffer association list 1154. If the DMA write or DMA read requested by system image 1116 is contained within a buffer in the system image buffer association list 1154, then virtual adapter 1112 may perform the operation. Otherwise,

virtual adapter 1112 is prohibited from performing the operation. Alternatively, the PCI family adapter 1101 may use a special, LPAR manager-style virtual adapter (rather than virtual adapter 1112) to perform the check that determines if DMA write or DMA read operations requested by system image 1116 is contained within a buffer in the system image buffer association list 1154. In a similar manner, virtual adapter 1104 associated with system image 1108 may validate DMA write or read requests submitted by system image 1108. Particularly, virtual adapter 1104 provides validation for DMA read and write requests from system image 1108 by determining whether the DMA write or read requested by system image 1108 is contained within a buffer in a buffer association list that contains PCI bus starting and ending address pairs in association with system image buffer starting and ending address pairs of buffers allocated to system image 1108 in a manner similar to that described above for system image 1116 and virtual adapter 1112.

The fourth mechanism that LPAR manager 708 can use to associate and make available host memory to a system image and to one or more virtual adapters is to write into the virtual adapter's resources a single starting and ending address in system image buffer association list 1154. In Figure 11, virtual adapter resources 1150 contains a single pair of PCI bus starting and ending address that is associated by the platform hardware to a pair (starting and ending) of addresses associated with a system image buffer, such as SI 2 buffer 1 1166. At initial configuration, and during reconfigurations, LPAR manager 708 loads the starting and ending addresses of SI 2 buffer 1166 into the system image buffer association list 1154 in virtual adapter resources 1150. The system image buffer association list 1154 then defines the set of addresses that virtual adapter 1112 can use in DMA write and read operations. After the system image buffer association list 1154 has been created, virtual adapter 1112 validates whether each DMA write or DMA read requested by system image 1116 is contained within the system image buffer association list 1154. If the DMA write or DMA read requested by system image 1116 is contained within system image buffer association list 1154, then virtual adapter 1112 may perform the operation. Otherwise, virtual adapter 1112 is prohibited from performing the operation. Alternatively, the PCI family adapter 1101 may use a special, LPAR manager-style virtual adapter (rather than virtual adapter 1150) to perform the check that determines if DMA write or DMA read requested by system image 1116 is contained within a page system image buffer association list 1154. In a similar manner, virtual adapter 1104 associated with system image 1108 may validate DMA

write or read requests submitted by system image 1108. Particularly, virtual adapter 1104 provides validation for DMA read and write requests from system image 1108 by determining whether the DMA write or read requested by system image 1108 is contained within a buffer in a buffer association list that contains a single PCI bus starting and ending address in association with a system image buffer starting and ending address allocated to system image 1108 in a manner similar to that described above for system image 1116 and virtual adapter 1112.

Turning next to Figure 12, a functional block diagram of a PCI family adapter configured with memory addresses that are made accessible to a system image is depicted in accordance with a preferred embodiment of the present invention.

Figure 12 depicts four different mechanisms by which a LPAR manager can associate PCI family adapter memory to a virtual adapter, such as virtual adapter 1204, and to a system image, such as system image 1208. Once PCI family adapter memory has been associated to a system image and a virtual adapter, the system image can then perform Memory Mapped I/O write and read (i.e., store and load) operations directly to the PCI family adapter memory.

A notable difference between the system image and virtual adapter configuration shown in Figures 11 and Figure 12 exists. In the configuration shown in Figure 11, PCI family adapter 1101 only holds a list of host addresses that do not have any local memory associated with them. If the PCI family adapter supports flow-through traffic, then data arriving on an external port can directly flow through the PCI family adapter and be transferred, through DMA writes, directly into these host addresses. Similarly, if the PCI family adapter supports flow-through traffic, then data from these host addresses can directly flow through the PCI family adapter and be transferred out of an external port. Accordingly, PCI family adapter 1101 shown in Figure 11 does not include local adapter memory and thus is unable to initiate a DMA operation. On the other hand, PCI family adapter 1201 shown in Figure 12 has local adapter memory that is associated with the list of host memory addresses. PCI family adapter 1201 can initiate, for example, DMA writes from its local memory to the host memory or DMA reads from the host memory to its local memory. Similarly, the host can initiate, for example, Memory Mapped I/O writes from its local memory to the PCI family adapter memory

or Memory Mapped I/O reads from the PCI family adapter memory to the host's local memory.

The first and second mechanisms that LPAR manager 708 can use to associate and make available PCI family adapter memory to a system image and to a virtual adapter is to write into the PCI family adapter's physical adapter memory translation table 1290 a page size and the starting address of one (first mechanism) or more (second mechanism) pages. In this case all pages have the same size. For example, Figure 12 depicts a set of pages that have been mapped between the system image 1208 and virtual adapter 1204. Particularly, SI 1 Page 1 1224 through SI 1 Page N 1242 of system image 1208 are mapped (illustratively shown by interconnected arrows) to virtual adapter memory pages 1224-1232 of physical adapter 1201 local memory. For system image 1208, all pages 1224-1242 in the list have the same size. At initial configuration, and during reconfigurations, LPAR manager 708 loads the PCI family adapter's physical adapter memory translation table 1290 with the page size and the starting address of one or more pages. The physical adapter memory translation table 1290 then defines the set of addresses that virtual adapter 1204 can use in DMA write and read operations. After physical adapter memory translation table 1290 has been created, PCI family adapter 1201 (or virtual adapter 1204) validates that each DMA write or DMA read requested by system image 1208 is contained in the physical adapter memory translation table 1290 and is associated with virtual adapter 1204. If the DMA write or DMA read requested by system image 1208 is contained in the physical adapter memory translation table 1290 and is associated with virtual adapter 1204, then virtual adapter 1204 may perform the operation. Otherwise, virtual adapter 1204 is prohibited from performing the operation. The physical adapter memory translation table 1290 also defines the set of addresses that system image 1208 can use in Memory Mapped I/O (MMIO) write and read operations. After physical adapter memory translation table 1290 has been created, PCI family adapter 1201 (or virtual adapter 1204) validates whether the Memory Mapped I/O write or read requested by system image 1208 is contained in the physical adapter memory translation table 1290 and is associated with virtual adapter 1204. If the MMIO write or MMIO read requested by system image 1208 is contained in the physical adapter memory translation table 1290 associated with virtual adapter 1204, then virtual adapter 1204 may perform the operation. Otherwise virtual adapter 1204 is prohibited from performing the operation. It should be understood that other system images and associated virtual adapters, e.g., system image 1216 and virtual adapter 1212, are

configured in a similar manner for PCI family adapter 1201 (or virtual adapter 1212) validation of DMA operations and MMIO operations requested by system image 1216.

The third and fourth mechanisms that LPAR manager 708 can use to associate and make available PCI family adapter memory to a system image and to a virtual adapter is to write into the PCI family adapter's physical adapter memory translation table 1290 one (third mechanism) or more (fourth mechanism) buffer starting and ending addresses (or starting address and length). In this case, the buffers may have different sizes. For example, Figure 12 depicts a set of varying sized buffers that have been mapped between system image 1216 and virtual adapter 1212. Particularly, SI 2 Buffer 1 1244 through SI 2 Buffer N 1248 of system image 1216 are mapped to virtual adapter buffers 1258-1274 of virtual adapter 1212. For system image 1216, the buffers in the list have different sizes. At initial configuration, and during reconfigurations, LPAR manager 708 loads the PCI family adapter's physical adapter memory translation table 1290 with the starting and ending address (or starting address and length) of one or more pages. The physical adapter memory translation table 1290 then defines the set of addresses that virtual adapter 1212 can use in DMA write and read operations. After physical adapter memory translation table 1290 has been created, PCI family adapter 1201 (or virtual adapter 1212) validates that each DMA write or DMA read requested by system image 1216 is contained in the physical adapter memory translation table 1290 and is associated with virtual adapter 1212. If the DMA write or DMA read requested by system image 1216 is contained in the physical adapter memory translation table 1290 and is associated with virtual adapter 1212, then virtual adapter 1212 may perform the operation. Otherwise, virtual adapter 1212 is prohibited from performing the operation. The physical adapter memory translation table 1290 also defines the set of addresses that system image 1216 can use in Memory Mapped I/O (MMIO) write and read operations. After physical adapter memory translation table 1290 has been created, PCI family adapter 1201 (or virtual adapter 1212) validates whether a MMIO write or read requested by system image 1216 is contained in the physical adapter memory translation table 1290 and is associated with virtual adapter 1212. If the MMIO write or MMIO read requested by system image 1216 is contained in the physical adapter memory translation table 1290 and is associated with virtual adapter 1212, then virtual adapter 1212 may perform the operation. Otherwise virtual adapter 1212 is prohibited from performing the operation. It should be understood that other system images and associated

virtual adapters, e.g., system image 1208 and associated virtual adapter 1204, are configured in a similar manner for PCI family adapter 1201 (or virtual adapter 1204) validation of DMA operations and MMIO operations requested by system image 1216.

With reference next to Figure 13, a functional block diagram of a PCI family adapter and a physical address memory translation table, such as a buffer table or a page table, is depicted in accordance with a preferred embodiment of the present invention.

Figure 13 also depicts four mechanisms for how an address referenced in an incoming PCI bus transaction 1304 can be used to look up the virtual adapter resources (including the local PCI family adapter memory address that has been mapped to the host address), such as virtual adapter resources 1398 or virtual adapter 1394 resources, associated with the memory address.

The first mechanism is to compare the memory address of incoming PCI bus transaction 1304 with each row of high address 1316 and low address 1320 in buffer table 1390. If incoming PCI bus transaction 1304 has an address that is lower than the contents of high address 1316 cell and that is higher than the contents of low address 1320 cell, then incoming PCI bus transaction 1304 is within the high address and low address cells that are associated with the corresponding virtual adapter. In such a scenario, the incoming PCI bus transaction 1304 is allowed to be performed on the matching virtual adapter. Alternatively, if incoming PCI bus transaction 1304 has an address that is not between the contents of high address 1316 cell and the contents of low address 1320 cell, then completion or processing of incoming PCI bus transaction 1304 is prohibited. The second mechanism is to simply allow a single entry in buffer table 1390 per virtual adapter.

The third mechanism is to compare the memory address of incoming PCI bus transaction 1304 with each row of page starting address 1322 and with each row of page starting Address 1322 plus the page size in the page table 1392. If incoming PCI bus transaction 1304 has an address that is higher than or equal to the contents of page starting address 1322 cell and lower than page starting address 1322 cell plus the page size, then incoming PCI bus transaction 1304 is within a page that is associated with a virtual adapter. Accordingly, incoming PCI bus transaction 1304 is allowed to be performed on the matching virtual adapter. Alternatively,

if incoming PCI bus transaction 1304 has an address that is not within the contents of page starting address 1322 cell and page starting address 1322 cell plus the page size, then completion of incoming PCI bus transaction 1304 is prohibited. The fourth mechanism is to simply allow a single entry in page table 1392 per virtual adapter.

With reference next to Figure 14, a functional block diagram of a PCI family adapter and a physical address memory translation table, such as a buffer table, a page table, or an indirect local address table, is depicted in accordance with a preferred embodiment of the present invention.

Figure 14 also depicts several mechanisms for how a requestor bus number, such as host bus number 1408, a requestor device number, such as host device number 1412, and a requestor function number, such as host function number 1416, referenced in incoming PCI bus transaction 1404 can be used to index into either buffer table 1498, page table 1494, or indirect local address table 1464. Buffer table 1498 is representative of buffer table 1390 shown in Figure 13. Page table 1490 is representative of page table 1392 shown in Figure 13. Local address table 1464 contains a local PCI family adapter memory address that references either a buffer table, such as buffer table 1438, or a page table, such as page table 1434, that only contains host memory addresses that are mapped to the same virtual adapter.

The requestor bus number, such as host bus number 1408, requestor device number, such as host device number 1412, and requestor function number, such as host function number 1416, referenced in incoming PCI bus transaction 1404 provides an additional check beyond the memory address mappings that were set up by a host LPAR manager.

Turning next to Figure 15, a virtual adapter level management approach is depicted in accordance with a preferred embodiment of the present invention. Under this approach, a physical or virtual host creates one or more virtual adapters, such as virtual adapter 1514, that each contain a set of resources within the scope of the physical adapter, such as PCI adapter 1532. Each virtual adapter is associated with a host side system image. A virtual adapter comprises a collection of resources (either virtualized or partitioned) of the physical adapter. By defining a virtual adapter entity, all virtual resources associated with a system image can be collectively manipulated by directing an action to the

corresponding virtual adapter. For example, a virtual adapter (and all included virtual resources) can be created, destroyed, or modified by performing a function targeting the corresponding virtual adapter. Additionally, the virtual adapter management approach allows all resources of a virtual adapter to be identified with a single identifier, e.g., a bus, device, and function number, that is associated with the virtual adapter. The set of resources associated with virtual adapter 1514 may include, for example: processing queues and associated resources 1504, adapter PCI port 1528 for one or more of adapter PCI port 1528 included on PCI physical adapter 1532, a PCI virtual port 1506 that is associated with one of the possible addresses on the adapter PCI port 1528, one or more downstream physical ports 1518 and 1522 for each downstream physical port, a downstream virtual port 1508 and 1510 that is associated with one of the possible addresses on physical port 1518 and 1522, and one or more address translation and protection tables (ATPTs) 1512. A virtual port, as referred to herein, comprises a software entity that facilitates receiving and sending of data from and to one or more resources of an input/output adapter. A virtual port is associated with, or mapped to, a port that is deployed on the input/output adapter. For example, a virtual port may be associated with an adapter PCI port with which the input/output adapter interfaces with a host or a physical port on the adapter that interfaces with a peripheral or network. A virtual port has an associated identifier, such as an address, index, or another suitable identifier for referencing the virtual adapter. A single port, such as a PCI port or a physical port on an input/output adapter, may have multiple virtual ports associated therewith. Additionally, a virtual port is preferably configured to exhibit one or more characteristics of a physical port to which it is mapped.

Turning next to Figure 16, a flowchart of a virtual adapter resource modification routine for modifying attributes of resources associated with a virtual adapter in a data processing system that uses the virtual adapter management approach described in Figure 15 is depicted in accordance with a preferred embodiment of the present invention.

The virtual adapter resource modification routine begins by invocation of a request to modify the attributes of an existing virtual adapter (step 1700). The request to modify the attributes of a virtual adapter may be invoked by, for example, a user management interface or an automated script/workflow. Table A contains examples of various virtual adapter attributes that may subjected to a modification request.

TABLE A

Attribute	Type	Description
New Downstream Virtual ID	Optional	The requested downstream network ID: - For Fibre Channel, N-port ID; - For Ethernet, MAC Address; - For Ethernet VLAN, VLAN ID; - For IP, IP Address; - For SCSI host; Initiator ID; - For SCSI target; Target ID.
Existing Adapter Processing Queue(s)	Optional	Use to modify the attributes of an existing processing queue, such as one or more of the following: - Number of work queue elements, - Number of scatter and/or gather elements per work queue element - The state of processing queue
Additional Adapter Processing Queue(s)	Optional	The requested: number of additional processing queues, the number of queue elements for each queue, and the number of scatter gather elements per work queue element. The types of processing queues requested may one or more of the following: - One or more Send/Receive Queue Pairs; zero, one or more Shared Receive Queues; one or more Completion Queues; and one or more Asynchronous Event Queues. - An IO Transaction Queue (that contains Command and Response elements in a single Queue); zero, one or more Completion Queues; and zero, one or more Asynchronous Event Queues. - A combination of these two types.
New Bus/Dev/Func Number for Virtual Adapter	Optional	Only used for PCI-X and PCI-E adapters. The requested PCI Bus Number, Device Number, and Function Number (Bus/Dev/Func #).
New Host address list	Optional	A page or buffer list of host memory addresses associated with the virtual adapter.
New Bus/Dev/Func Number of the Host that is associated with the Virtual Adapter	Optional	Only used for PCI-X and PCI-E adapters. The PCI Bus Number, Device Number, and Function Number (Bus/Dev/Func #) that are assigned to the Host, where the Host may be a Physical Host, a Partitioned Host, or a Virtual Host.

New size of Verb Memory Translation and Protection Table	Optional	The requested new number of Memory Translation and Protection Table entries that are to be assigned to the Virtual Adapter. This table is used for accesses through Memory Regions and Memory Windows.
New size of Host Address Translation and Protection Table	Optional	The requested new number of Host Address Translation and Protection Table entries that are to be assigned to the Virtual Adapter. This table is used to validate MMIOs and/or DMAs.
New MSI Level for the Virtual Adapter	Optional	For an adapter capable of supporting message signaled interrupts (MSI), the requested new message signaled interrupt level(s).
Virtual Adapter ID	Optional	An Identifier requested for the newly created Virtual Adapter.

The LPAR manager directly, or through another suitable intermediary, uses the physical adapter's memory management interface (i.e. the memory mapped I/O addresses that are used for virtual adapter configuration management) to request that the physical adapter modify the attributes of an existing virtual adapter (step 1708).

The physical adapter checks to see if the number of resources requested for the modified virtual adapter exceeds the resources available (step 1724). If the physical adapter does not have sufficient resources to complete the modify request, then it generates an error with a termination code that states it had insufficient resources (step 1725) and the virtual adapter resource modification routine exits (step 1736).

Alternatively, the LPAR manager, rather than the physical adapter, may check to determine if the physical adapter has sufficient resources to modify the virtual adapter resources prior to requesting the physical adapter to modify the virtual adapter resources.

Returning again to step 1724, if the physical adapter does have sufficient resources to complete the request, then it checks to see if the request is a request to modify resources that are currently busy (step 1726). If the request doesn't impact currently busy resources, then the physical adapter proceeds to modify resource attributes of the virtual adapter (step 1732). If it is determined that the request does impact busy resources at step 1726, then the PCI physical adapter initiates a timer to wait for a quiescent point to be reached (step 1728), that is a point where there are no more operations that utilize the resource

targeted by the virtual adapter resource modification request outstanding on the downstream and upstream interfaces.

The physical adapter then evaluates whether the quiescent point has been reached prior to the timeout (step 1730). If the physical adapter reaches a quiescent point before the timer times out, then it proceeds to modify the attribute of the virtual adapter according to step 1732. Otherwise, the virtual adapter resource modification routine generates an error indicating that the physical adapter was busy and is unable to complete the request (step 1731), and the routine proceeds to exit according to step 1736.

When the physical adapter modifies the attributes of the existing virtual adapter and completes the request, the physical adapter generates a return message that is conveyed to the LPAR manager (step 1734). TABLE B shows exemplary attribute information that may be conveyed to the LPAR manager upon successful modification of virtual adapter resource attributes.

TABLE B

Attribute	Type	Description
Downstream Virtual ID	Required	The assigned downstream network ID: - For Fibre Channel, N-port ID; - For Ethernet, MAC Address; - For Ethernet VLAN, VLAN ID; - For IP, IP Address; - For SCSI host; Initiator ID; - For SCSI target; Target ID.
Adapter Processing Queue(s)	Required	The assigned number of processing queues and the assigned number of queue elements for each queue. The types of processing queues requested may one or more of the following: - One or more Send/Receive Queue Pairs; zero, one or more Shared Receive Queues; one or more Completion Queues; and one or more Asynchronous Event Queues. - An IO Transaction Queue (that contains Command and Response elements in a single Queue); zero, one or more Completion Queues; and zero, one or more Asynchronous Event Queues. - A combination of these two types.

Bus/Dev/Func Number for Virtual Adapter	Required if Adapter supports Virt. Approach 1	Only used for PCI-X and PCI-E adapters. The assigned PCI Bus Number, Device Number, and Function Number (Bus/Dev/Func #).
Verb Memory Translation and Protection Table Entries	Required if Adapter supports Network Stack Offload	The number of Memory Translation and Protection Table entries that were assigned to the Virtual Adapter.
Host Address Translation and Protection Table Entries	Required if Adapter supports Virt. Approach 2 or 3	The number of Host Address Translation and Protection Table entries that were assigned to the Virtual Adapter.
MSI Level for the Virtual Adapter	Required if adapter supports MSI	For an adapter capable of supporting message signaled interrupts (MSI), the assigned message signaled interrupt level(s).
Virtual Adapter ID	Optional	An Identifier assigned for the newly created Virtual Adapter.

Upon conveying the return message to the LPAR manager, the virtual adapter resource modification routine exits according to step 1736.

CLAIMS

1. A method of modifying resources in a logically partitioned data processing system, the method comprising the steps of:

invoking a request to modify resources associated with a virtual adapter allocated on a physical adapter, wherein the resources comprise a subset of physical adapter resources;

conveying the request to the physical adapter; and

responsive to receipt of the request by the physical adapter, modifying the resources allocated to the virtual adapter on the physical adapter.

2. The method of claim 1, wherein the step of invoking is performed by a user management interface that interfaces with a logical partitioning manager.

3. The method of claim 1, wherein the step of conveying further includes:

requesting, by a logical partitioning manager interfacing with the physical adapter, the physical adapter to modify the resources of the virtual adapter through a memory management interface of the physical adapter.

4. The method of claim 1, wherein the physical adapter comprises a peripheral component interconnect family adapter.

5. The method of claim 1, further comprising:

evaluating whether the existing resources associated with the virtual adapter are sufficient to satisfy the request.

6. The method of claim 5, further comprising:

responsive to determining that the resources associated with the virtual adapter are containable within existing resources, initiating a timer.

7. The method of claim 6, further comprising:

evaluating whether a quiescent point is reached prior to the timer timing out.

8. The method of claim 7, wherein modifying the resources is performed responsive to the quiescent point being reached.

9. The method of claim 1, further comprising:

conveying a return message to a logical partitioning manager that indicates attributes of virtual adapter resources that have been modified.

10. The method of claim 1, wherein the virtual adapter has an associated identifier comprising a bus number, device number, and function number, and the request specifies the virtual adapter by referencing the identifier.

11. A computer program product for modifying resources in a logically partitioned data processing system, the computer program product comprising:

first instructions that invoke a request to modify resources associated with a virtual adapter allocated on a physical adapter, wherein the resources comprise a subset of physical adapter resources;

second instructions that convey the request to the physical adapter; and

third instructions that, responsive to receipt of the request by the physical adapter, modify the resources allocated to the virtual adapter on the physical adapter.

12. The computer program product of claim 11, further comprising:

fourth instructions that invoke the request by a user management interface that interfaces with a logical partitioning manager.

13. The computer program product of claim 11, further comprising:

fourth instructions that evaluate whether the resources associated with the virtual adapter are containable within existing resources.

14. The computer program product of claim 13, further comprising:

fifth instructions that, responsive to the fourth instructions determining that the resources associated with the virtual adapter are containable within existing resources, initiate a timer.

15. The computer program product of claim 14, further comprising:

sixth instructions that evaluate whether a quiescent point is reached prior to the timer timing out.

16. The computer program product of claim 15, wherein the third instructions modify the resources responsive to the sixth instructions determining that the quiescent point has been reached prior to the timer timing out.

17. The computer program product of claim 11, further comprising:

fourth instructions that convey a return message to a logical partitioning manager that indicates attributes of the resources that have been modified.

18. The computer program product of claim 11, wherein the virtual adapter has an associated identifier comprising a bus number, device number, and function number, and the request specifies the virtual adapter by referencing the identifier.

19. A logically partitioned data processing system adapted to modify virtual adapter resources, comprising:

a physical adapter having a plurality of allocated virtual adapters, wherein each virtual adapter has a respective subset of resources of the physical adapter allocated thereto;

a memory that contains a plurality of system images each respectively associated with a one of the plurality of virtual adapters;

a store containing a logical partitioning manager as a set of instructions; and

a processor that, responsive to execution of the instructions, generates a request to modify a subset of resources allocated to a virtual adapter of the plurality of virtual adapters and that conveys the request to the physical adapter, wherein the physical adapter modifies the subset of resources allocated to the virtual adapter responsive to receipt of the request.

20. The data processing system of claim 19, wherein the store comprises a system firmware.

21. The data processing system of claim 19, wherein the physical adapter comprises a peripheral component interconnect family adapter.

22. The data processing system of claim 19, wherein the physical adapter conveys a return message to the store that specifies attributes of the subset of resources that have been modified.

23. The data processing system of claim 19, wherein the virtual adapter has an associated identifier comprising a bus number, device number, and function number, and the request specifies the virtual adapter by referencing the identifier.

FIG. 1

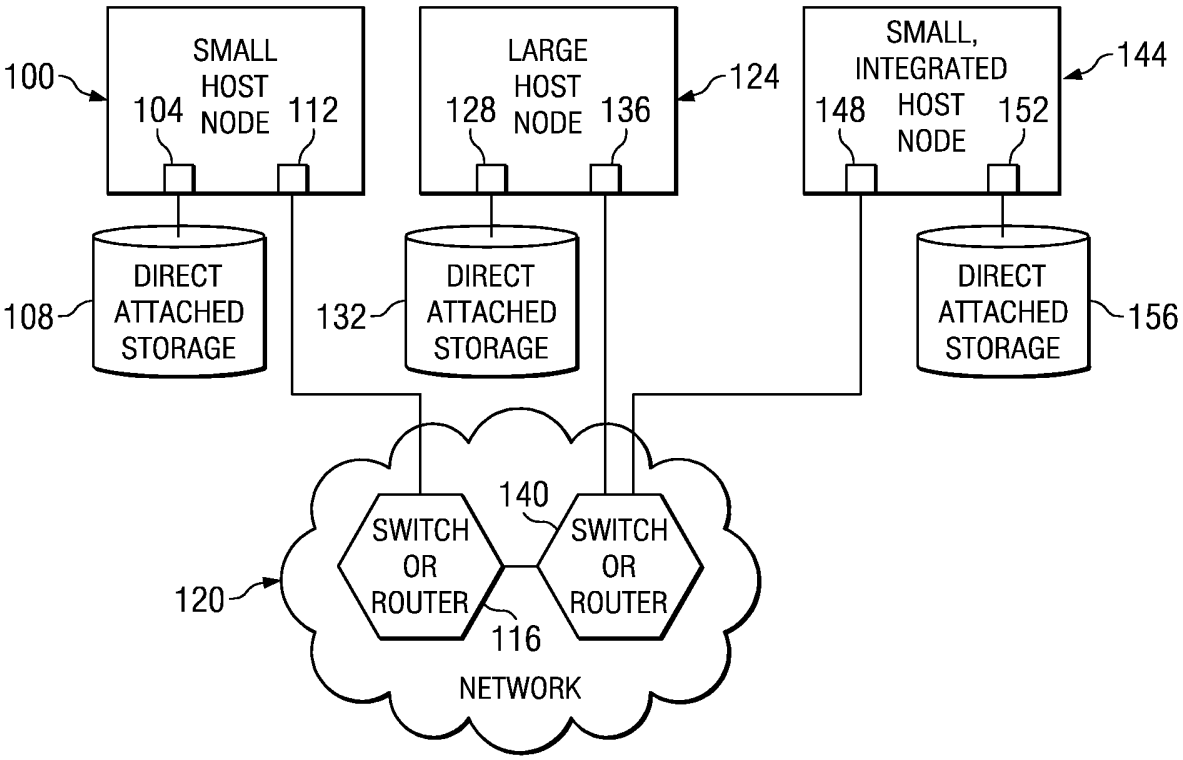

SMALL HOST NODE
104
112
100
LARGE HOST NODE
128
136
124
SMALL, INTEGRATED HOST NODE
148
152
144
DIRECT ATTACHED STORAGE
108
DIRECT ATTACHED STORAGE
132
DIRECT ATTACHED STORAGE
156
140
SWITCH OR ROUTER
SWITCH OR ROUTER
116
120
NETWORK

FIG. 2

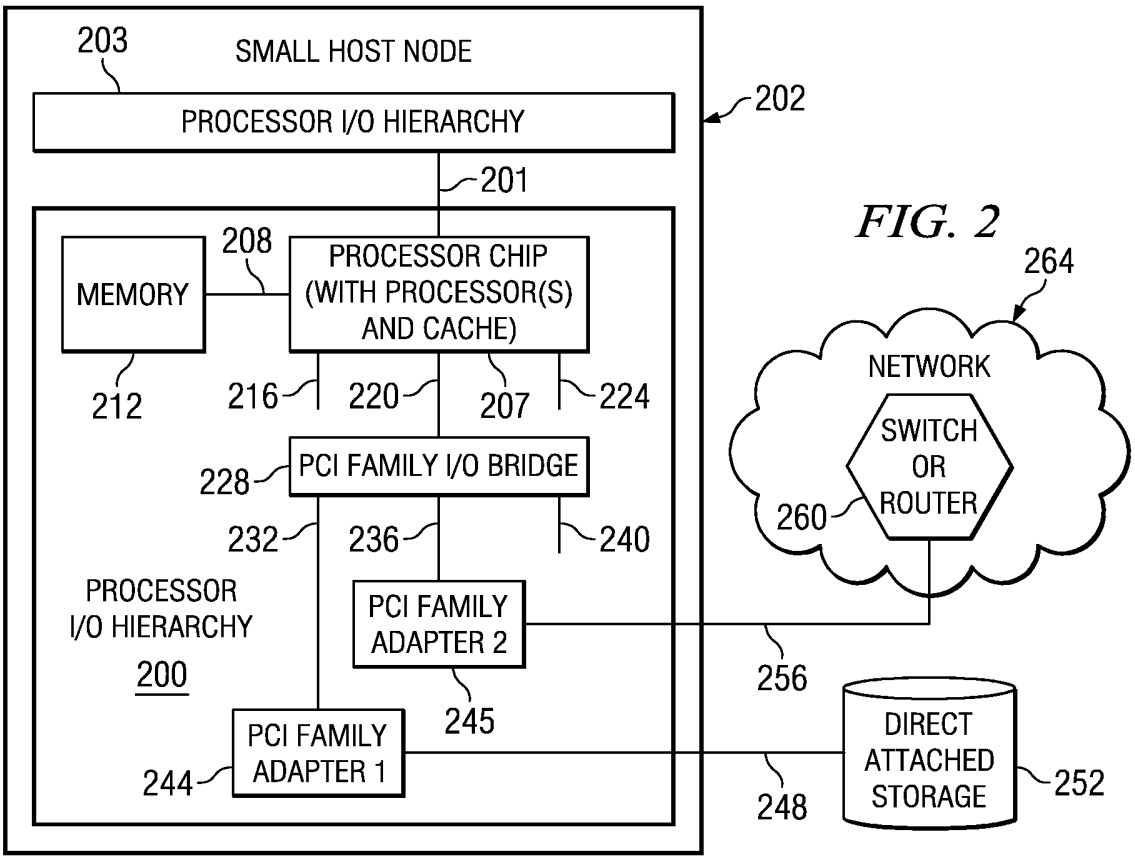

203
SMALL HOST NODE
PROCESSOR I/O HIERARCHY
202
201
208
MEMORY
PROCESSOR CHIP (WITH PROCESSOR(S) AND CACHE)
212
216
220
207
224
228
PCI FAMILY I/O BRIDGE
232
236
240
PROCESSOR I/O HIERARCHY
200
PCI FAMILY ADAPTER 2
245
PCI FAMILY ADAPTER 1
244
264
NETWORK
SWITCH OR ROUTER
260
256
DIRECT ATTACHED STORAGE
252
248

FIG. 3

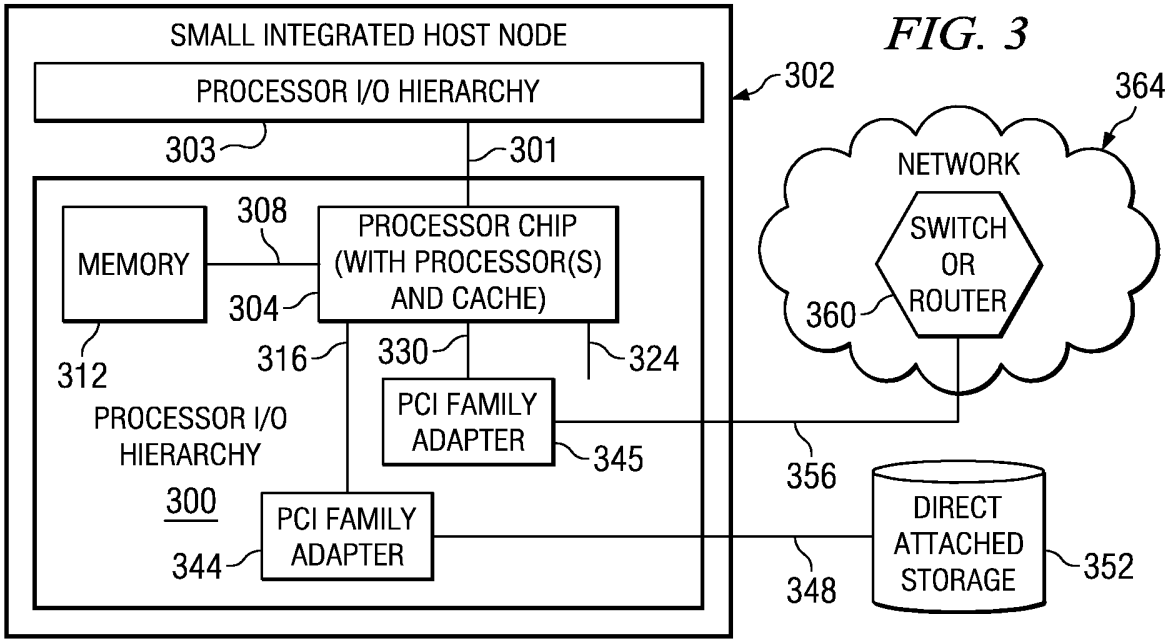

SMALL INTEGRATED HOST NODE
PROCESSOR I/O HIERARCHY
302
303
301
364
NETWORK
SWITCH OR ROUTER
360
MEMORY
308
304
PROCESSOR CHIP (WITH PROCESSOR(S) AND CACHE)
312
316
330
324
PROCESSOR I/O HIERARCHY
300
PCI FAMILY ADAPTER
345
356
PCI FAMILY ADAPTER
344
348
DIRECT ATTACHED STORAGE
352

FIG. 5

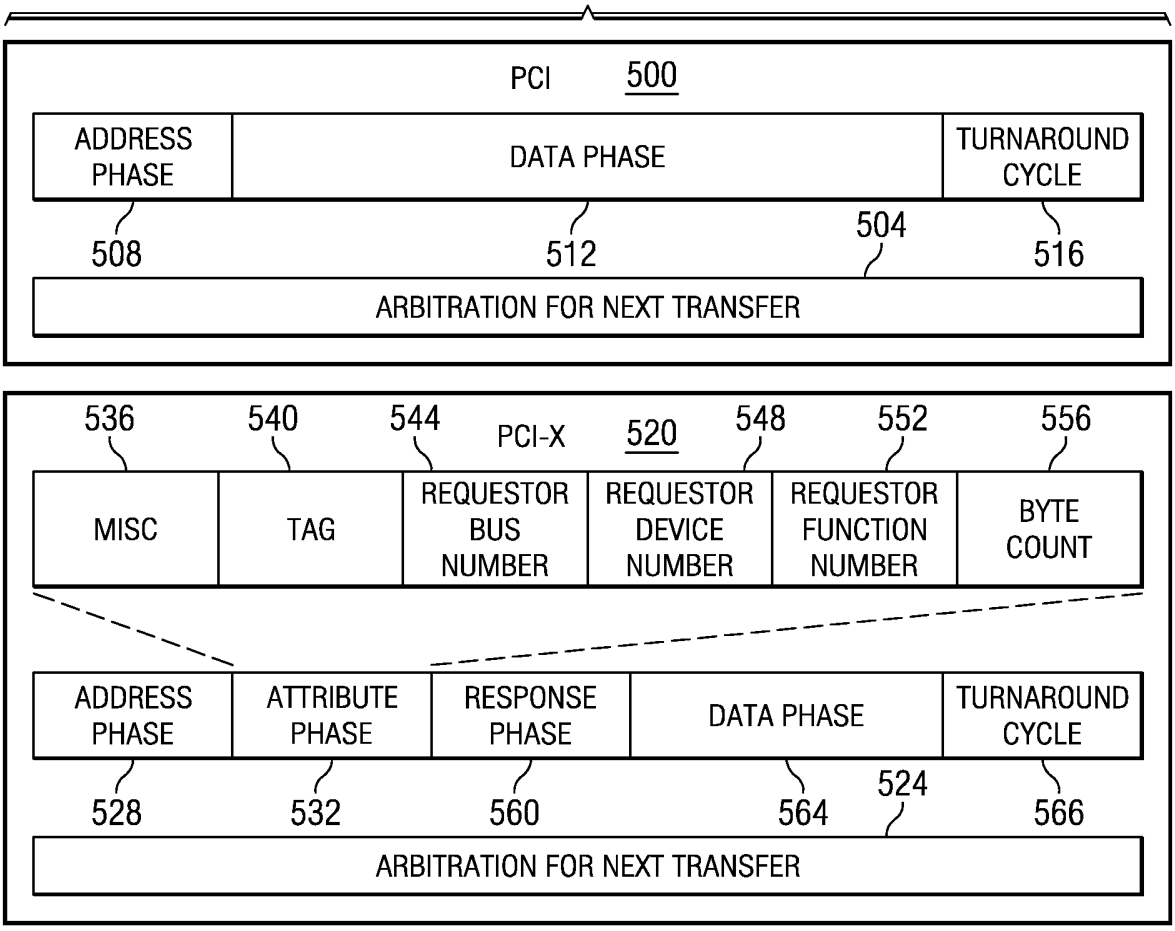

PCI 500
ADDRESS PHASE
DATA PHASE
TURNAROUND CYCLE
508
512
504
516
ARBITRATION FOR NEXT TRANSFER
536
540
544
PCI-X 520
548
552
556
MISC
TAG
REQUESTOR BUS NUMBER
REQUESTOR DEVICE NUMBER
REQUESTOR FUNCTION NUMBER
BYTE COUNT
ADDRESS PHASE
ATTRIBUTE PHASE
RESPONSE PHASE
DATA PHASE
TURNAROUND CYCLE
528
532
560
564
524
566
ARBITRATION FOR NEXT TRANSFER

FIG. 4

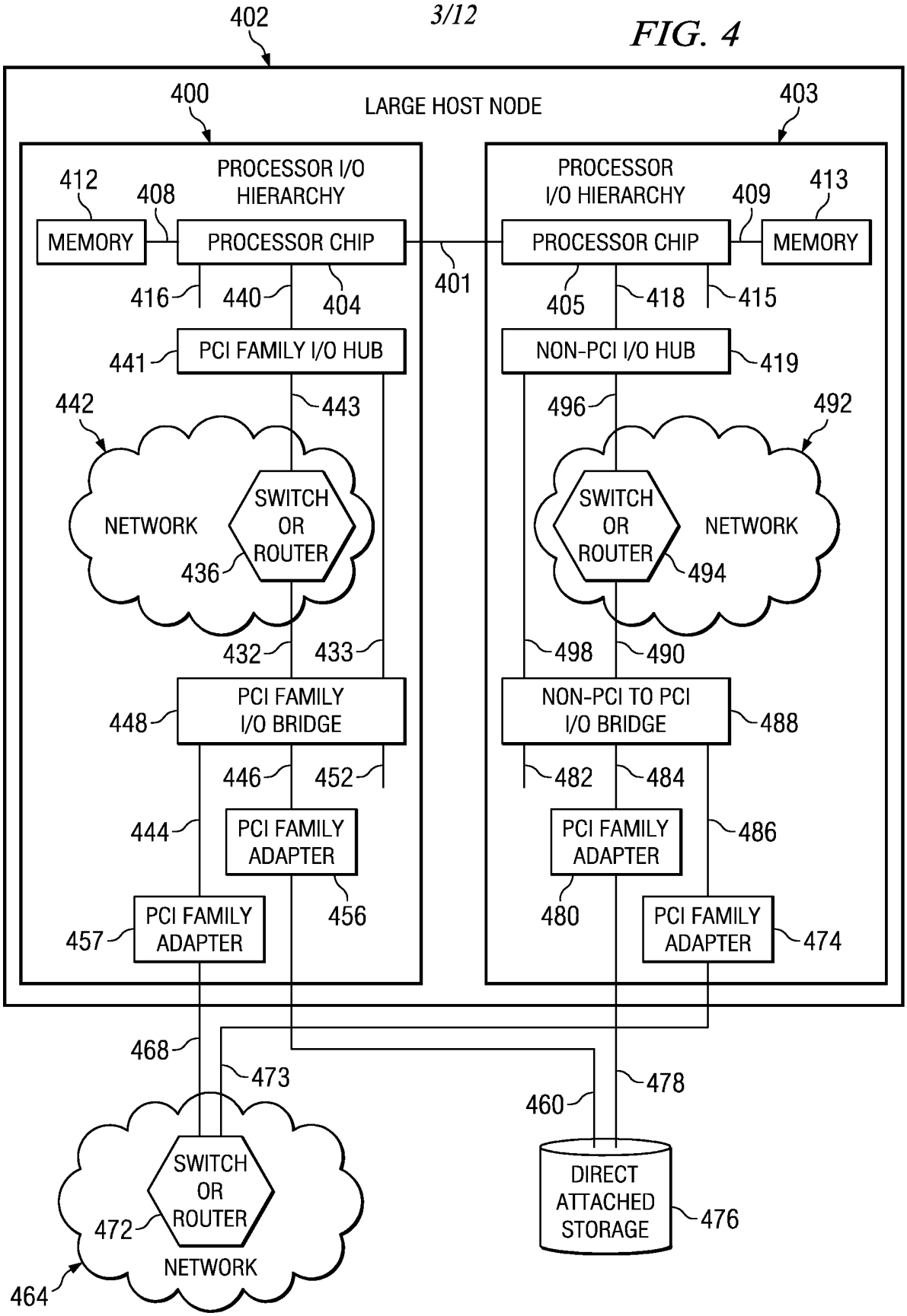
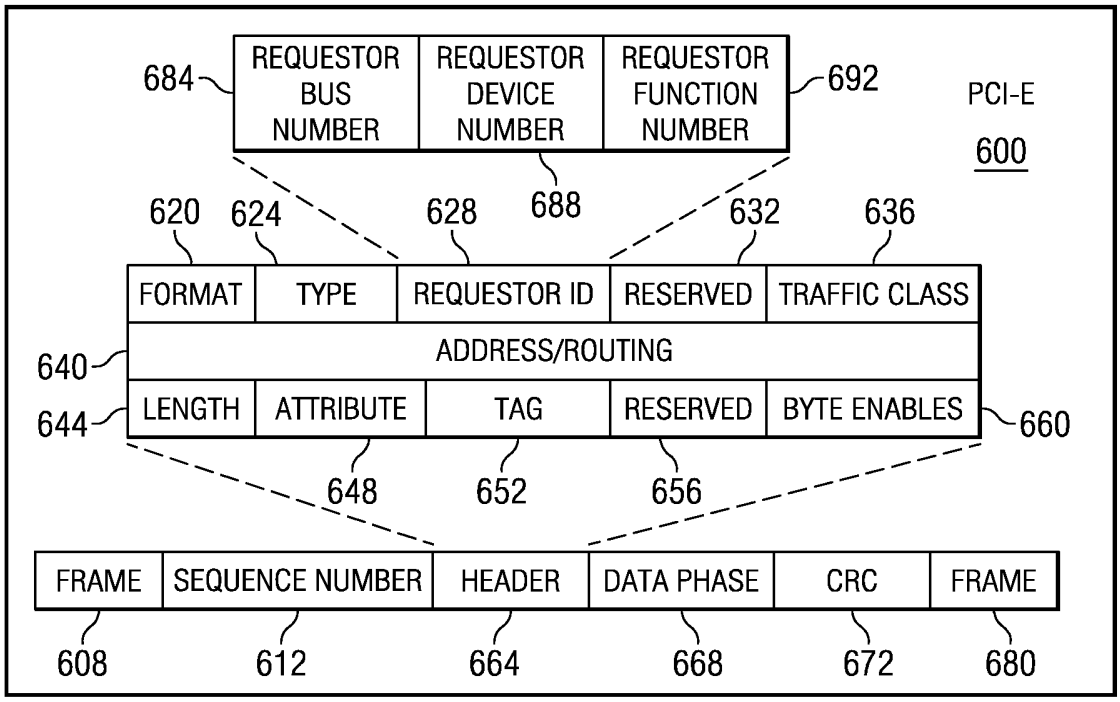

402
400
LARGE HOST NODE
403
PROCESSOR I/O HIERARCHY
PROCESSOR I/O HIERARCHY
412
408
409
413
MEMORY
PROCESSOR CHIP
PROCESSOR CHIP
MEMORY
401
416
440
404
405
418
415
441
PCI FAMILY I/O HUB
NON-PCI I/O HUB
419
442
443
496
492
NETWORK
SWITCH OR ROUTER
436
SWITCH OR ROUTER
NETWORK
494
432
433
498
490
448
PCI FAMILY I/O BRIDGE
NON-PCI TO PCI I/O BRIDGE
488
446
452
482
484
444
PCI FAMILY ADAPTER
PCI FAMILY ADAPTER
486
457
PCI FAMILY ADAPTER
456
480
PCI FAMILY ADAPTER
474
468
473
478
460
SWITCH OR ROUTER
472
NETWORK
464
DIRECT ATTACHED STORAGE
476

FIG. 6


684
REQUESTOR BUS NUMBER
REQUESTOR DEVICE NUMBER
REQUESTOR FUNCTION NUMBER
692
PCI-E
600
688
620
624
628
632
636
FORMAT
TYPE
REQUESTOR ID
RESERVED
TRAFFIC CLASS
640
ADDRESS/ROUTING
644
LENGTH
ATTRIBUTE
TAG
RESERVED
BYTE ENABLES
660
648
652
656
FRAME
SEQUENCE NUMBER
HEADER
DATA PHASE
CRC
FRAME
608
612
664
668
672
680

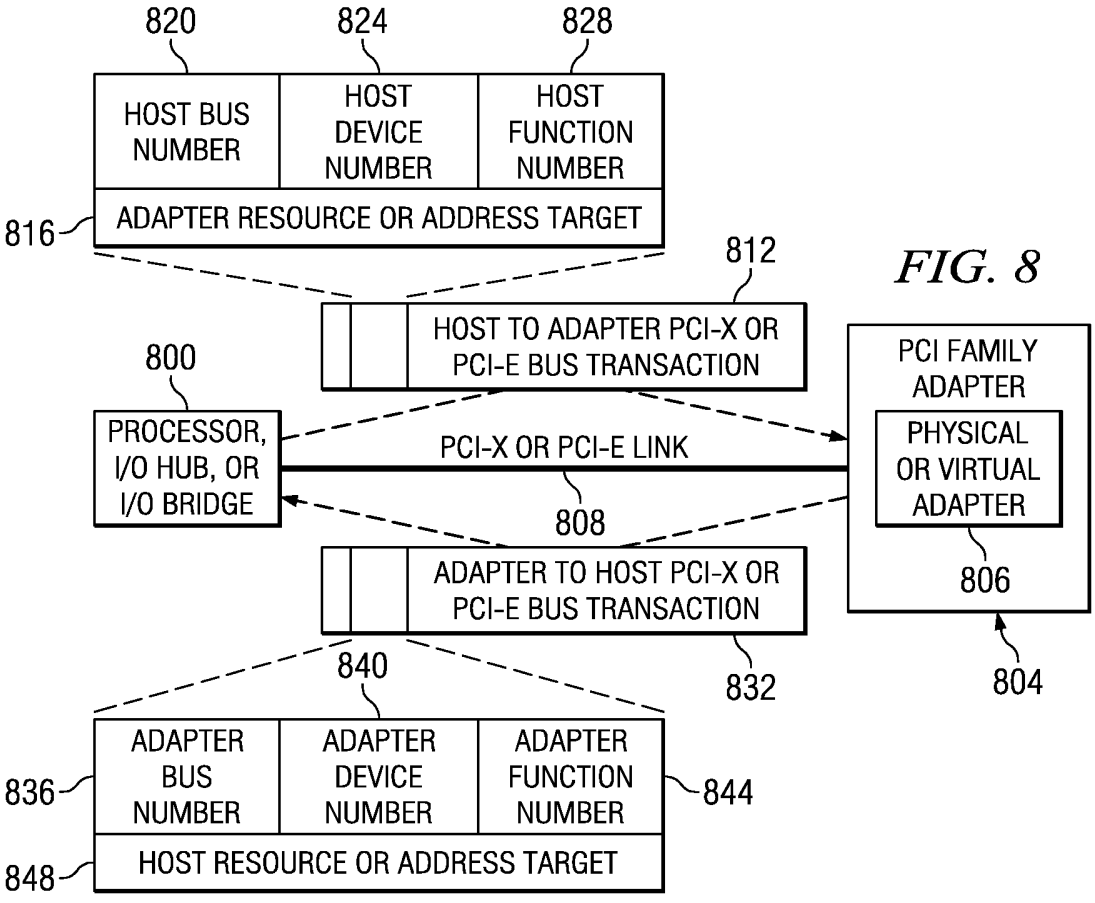

820
824
828
HOST BUS NUMBER
HOST DEVICE NUMBER
HOST FUNCTION NUMBER
816
ADAPTER RESOURCE OR ADDRESS TARGET
812
HOST TO ADAPTER PCI-X OR PCI-E BUS TRANSACTION
800
PROCESSOR, I/O HUB, OR I/O BRIDGE
PCI-X OR PCI-E LINK
808
ADAPTER TO HOST PCI-X OR PCI-E BUS TRANSACTION
832
PCI FAMILY ADAPTER
PHYSICAL OR VIRTUAL ADAPTER
806
804
840
ADAPTER BUS NUMBER
ADAPTER DEVICE NUMBER
ADAPTER FUNCTION NUMBER
836
844
848
HOST RESOURCE OR ADDRESS TARGET

FIG. 8

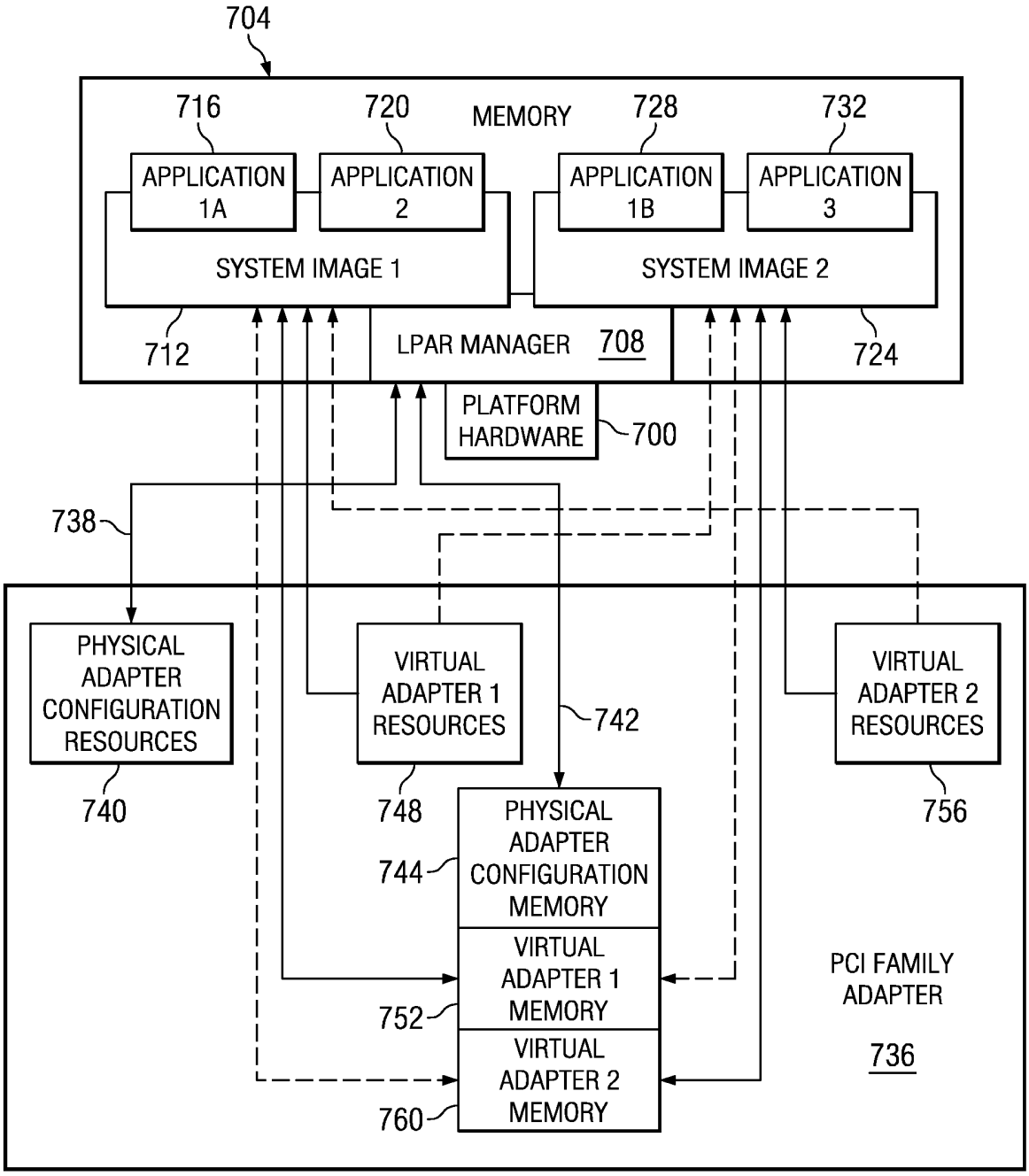

704
716
720
MEMORY
728
732
APPLICATION 1A
APPLICATION 2
APPLICATION 1B
APPLICATION 3
SYSTEM IMAGE 1
SYSTEM IMAGE 2
712
LPAR MANAGER 708
724
PLATFORM HARDWARE
700
738
PHYSICAL ADAPTER CONFIGURATION RESOURCES
740
VIRTUAL ADAPTER 1 RESOURCES
748
742
VIRTUAL ADAPTER 2 RESOURCES
756
744
PHYSICAL ADAPTER CONFIGURATION MEMORY
VIRTUAL ADAPTER 1 MEMORY
752
VIRTUAL ADAPTER 2 MEMORY
760
PCI FAMILY ADAPTER
736

FIG. 7

FIG. 9

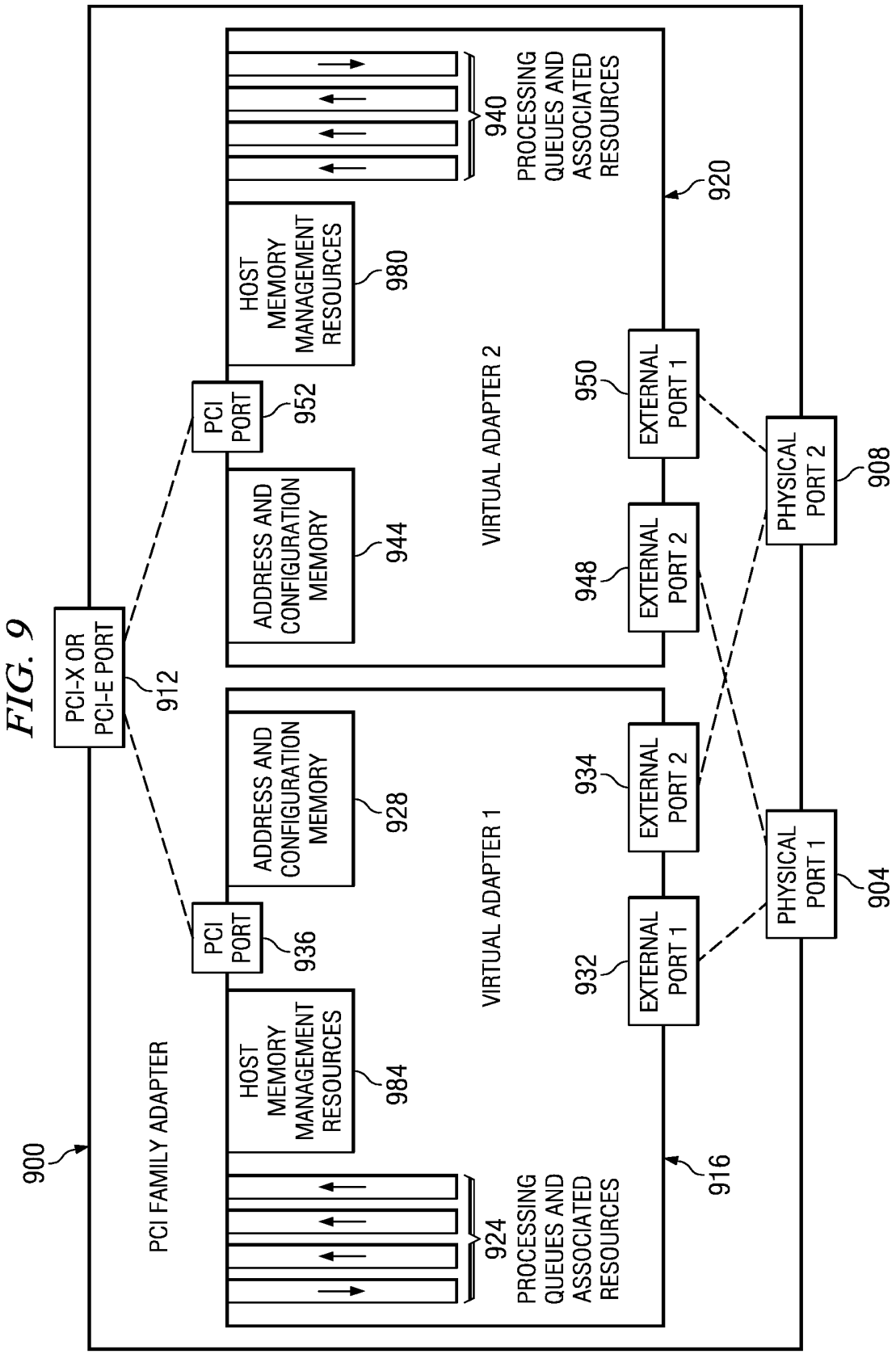

900
PCI FAMILY ADAPTER
PCI-X OR PCI-E PORT
912
PCI PORT
936
PCI PORT
952
HOST MEMORY MANAGEMENT RESOURCES
984
HOST MEMORY MANAGEMENT RESOURCES
980
ADDRESS AND CONFIGURATION MEMORY
928
ADDRESS AND CONFIGURATION MEMORY
944
924
PROCESSING QUEUES AND ASSOCIATED RESOURCES
940
PROCESSING QUEUES AND ASSOCIATED RESOURCES
916
920
VIRTUAL ADAPTER 1
VIRTUAL ADAPTER 2
932
EXTERNAL PORT 1
934
EXTERNAL PORT 2
948
EXTERNAL PORT 2
950
EXTERNAL PORT 1
PHYSICAL PORT 1
904
PHYSICAL PORT 2
908

FIG. 10

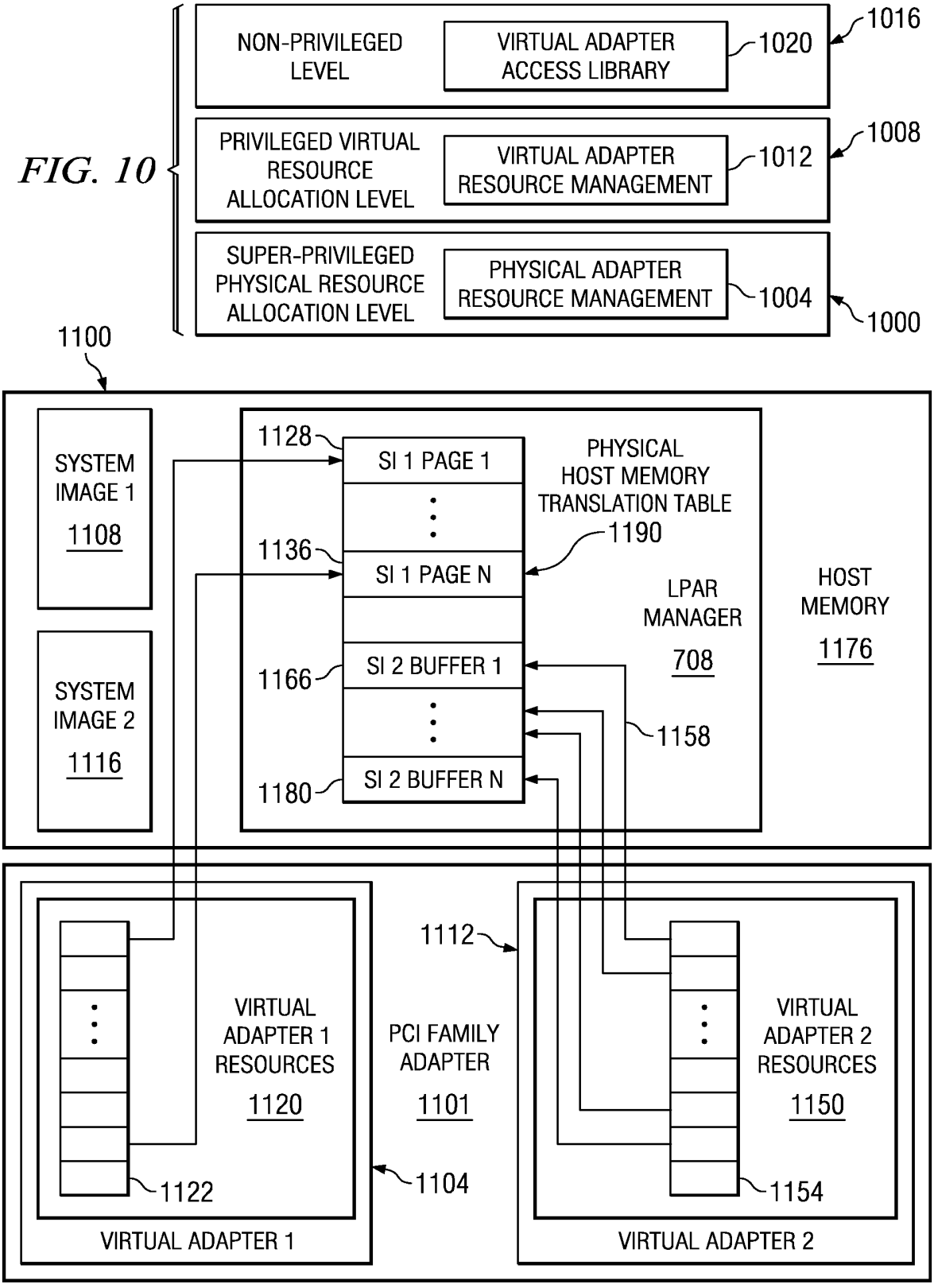

NON-PRIVILEGED LEVEL
VIRTUAL ADAPTER ACCESS LIBRARY
1020
1016
PRIVILEGED VIRTUAL RESOURCE ALLOCATION LEVEL
VIRTUAL ADAPTER RESOURCE MANAGEMENT
1012
1008
SUPER-PRIVILEGED PHYSICAL RESOURCE ALLOCATION LEVEL
PHYSICAL ADAPTER RESOURCE MANAGEMENT
1004
1000
1100
SYSTEM IMAGE 1
1108
SYSTEM IMAGE 2
1116
1128
SI 1 PAGE 1
1136
SI 1 PAGE N
1166
SI 2 BUFFER 1
1180
SI 2 BUFFER N
PHYSICAL HOST MEMORY TRANSLATION TABLE
1190
LPAR MANAGER
708
HOST MEMORY
1176
1158
1112
VIRTUAL ADAPTER 1 RESOURCES
1120
PCI FAMILY ADAPTER
1101
1104
VIRTUAL ADAPTER 2 RESOURCES
1150
1122
1154
VIRTUAL ADAPTER 1
VIRTUAL ADAPTER 2

FIG. 11

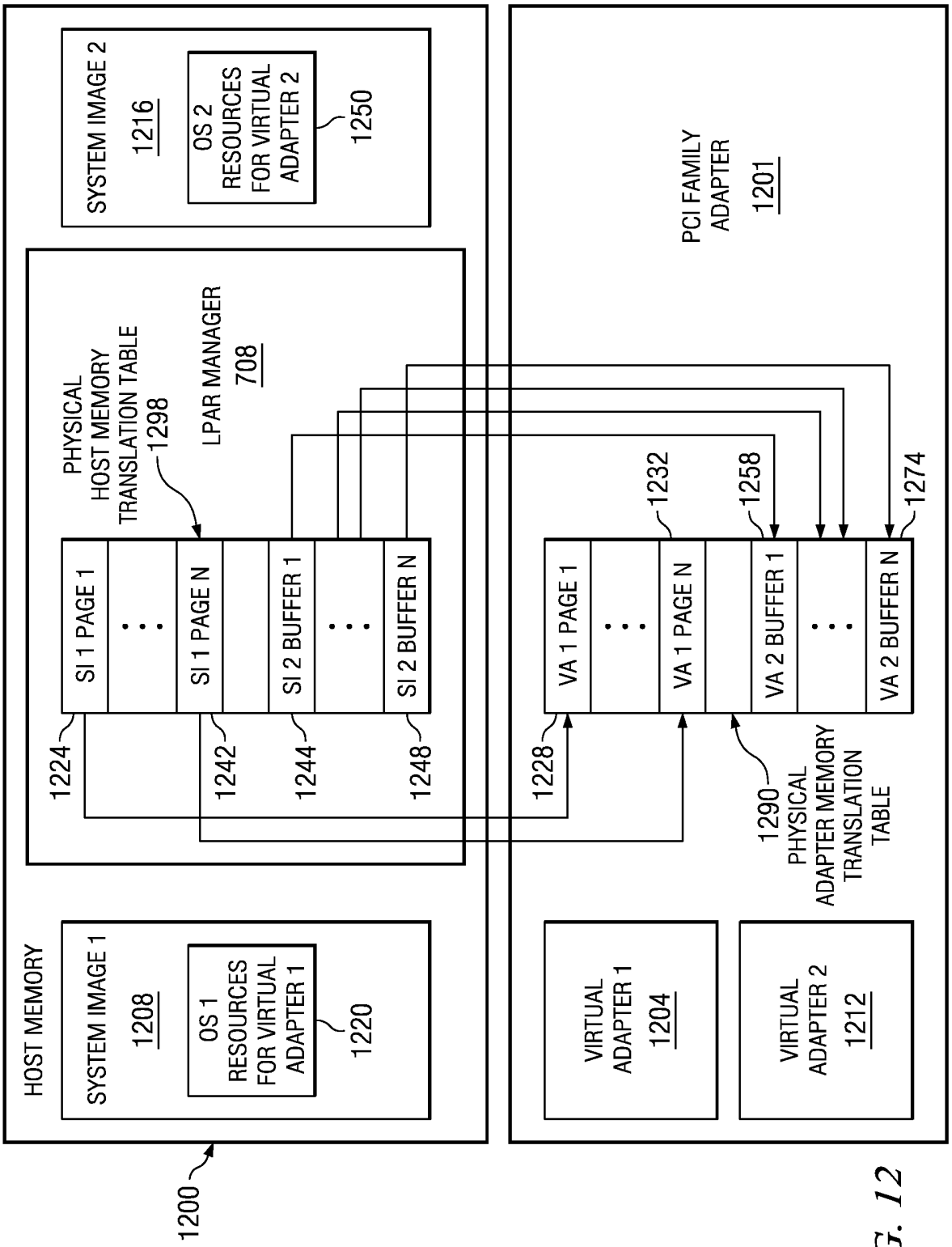

HOST MEMORY
SYSTEM IMAGE 1
1208
OS 1 RESOURCES FOR VIRTUAL ADAPTER 1
1220
SYSTEM IMAGE 2
1216
OS 2 RESOURCES FOR VIRTUAL ADAPTER 2
1250
LPAR MANAGER
708
PHYSICAL HOST MEMORY TRANSLATION TABLE
1298
SI 1 PAGE 1
SI 1 PAGE N
SI 2 BUFFER 1
SI 2 BUFFER N
1224
1242
1244
1248
PCI FAMILY ADAPTER
1201
VA 1 PAGE 1
VA 1 PAGE N
VA 2 BUFFER 1
VA 2 BUFFER N
1228
1232
1258
1274
1290
PHYSICAL ADAPTER MEMORY TRANSLATION TABLE
VIRTUAL ADAPTER 1
1204
VIRTUAL ADAPTER 2
1212
1200

FIG. 12

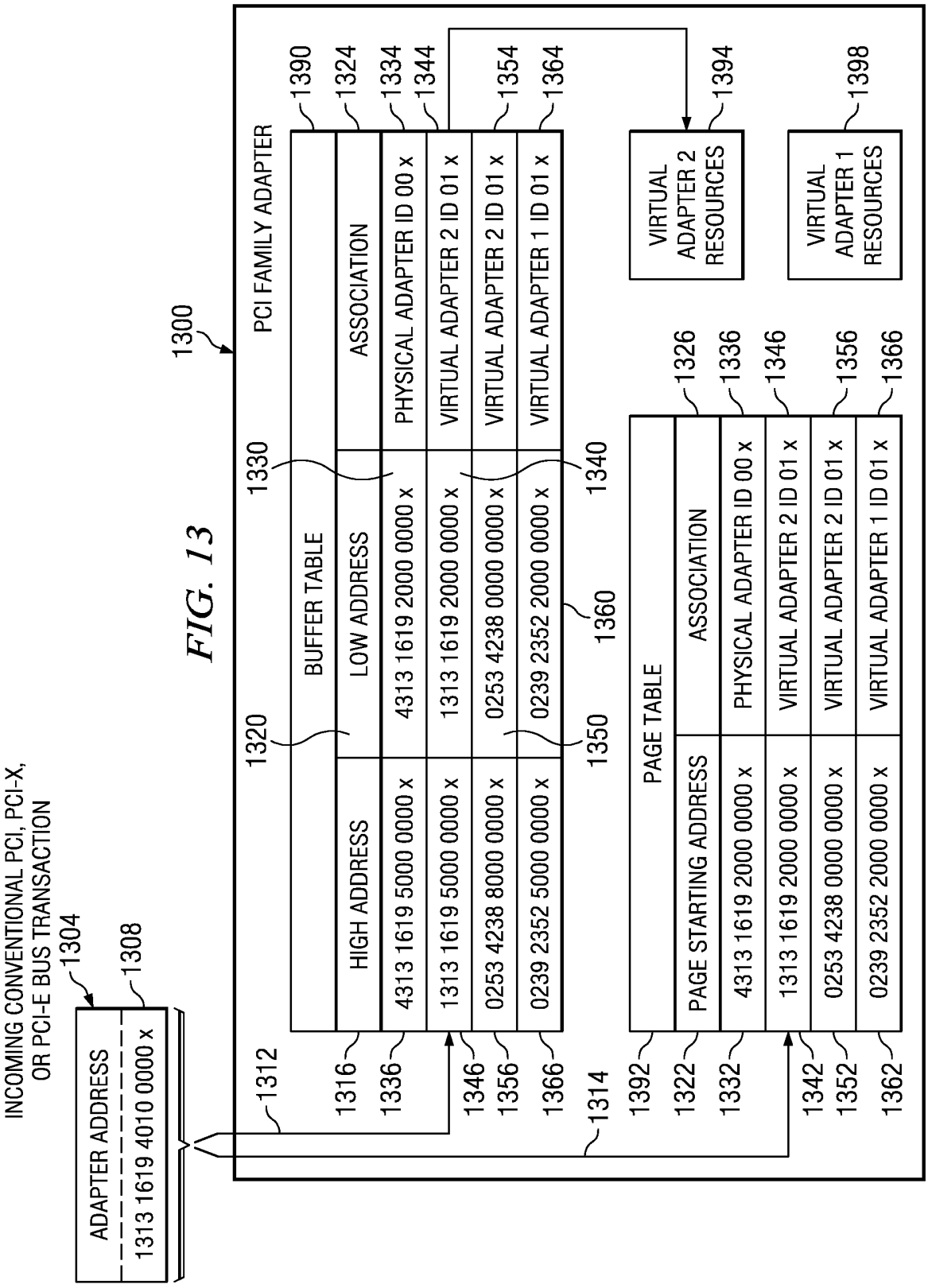

INCOMING CONVENTIONAL PCI, PCI-X, OR PCI-E BUS TRANSACTION
ADAPTER ADDRESS
1313 1619 4010 0000 x
1304
1308
1300
PCI FAMILY ADAPTER
1312
1314
1320
BUFFER TABLE
HIGH ADDRESS
LOW ADDRESS
ASSOCIATION
1330
1340
4313 1619 5000 0000 x
4313 1619 2000 0000 x
PHYSICAL ADAPTER ID 00 x
1313 1619 5000 0000 x
1313 1619 2000 0000 x
VIRTUAL ADAPTER 2 ID 01 x
0253 4238 8000 0000 x
0253 4238 0000 0000 x
VIRTUAL ADAPTER 2 ID 01 x
0239 2352 5000 0000 x
0239 2352 2000 0000 x
VIRTUAL ADAPTER 1 ID 01 x
1316
1336
1346
1356
1366
1324
1334
1344
1354
1364
1390
1360
1350
PAGE TABLE
PAGE STARTING ADDRESS
ASSOCIATION
4313 1619 2000 0000 x
PHYSICAL ADAPTER ID 00 x
1313 1619 2000 0000 x
VIRTUAL ADAPTER 2 ID 01 x
0253 4238 0000 0000 x
VIRTUAL ADAPTER 2 ID 01 x
0239 2352 2000 0000 x
VIRTUAL ADAPTER 1 ID 01 x
1392
1322
1332
1342
1352
1362
1326
1336
1346
1356
1366
VIRTUAL ADAPTER 2 RESOURCES
1394
VIRTUAL ADAPTER 1 RESOURCES
1398

FIG. 13

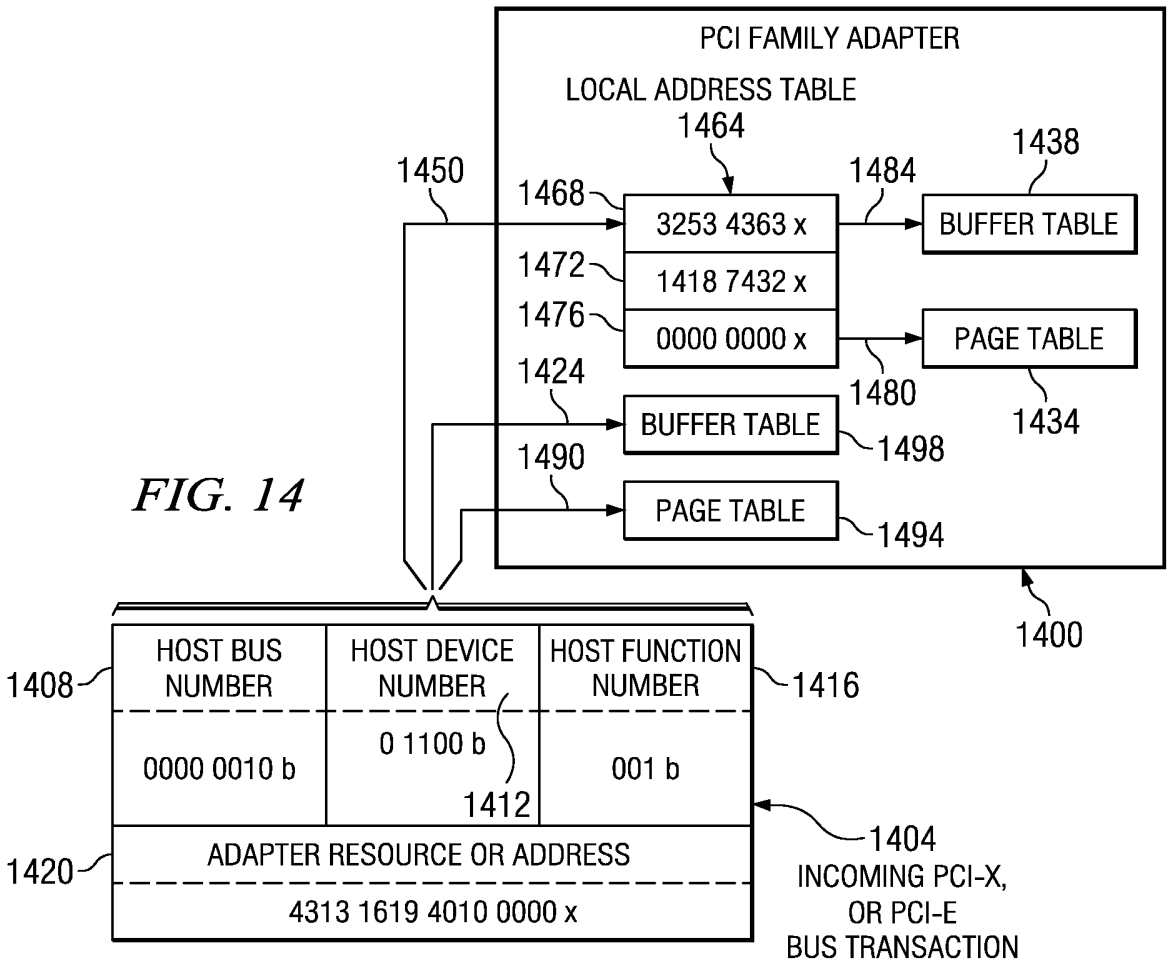

PCI FAMILY ADAPTER
LOCAL ADDRESS TABLE
1464
1450
1468
1484
1438
3253 4363 x
BUFFER TABLE
1472
1418 7432 x
1476
0000 0000 x
PAGE TABLE
1424
1480
1434
BUFFER TABLE
1498
1490
PAGE TABLE
1494
1400
HOST BUS NUMBER
HOST DEVICE NUMBER
HOST FUNCTION NUMBER
1408
1416
0000 0010 b
0 1100 b
1412
001 b
1404
ADAPTER RESOURCE OR ADDRESS
1420
4313 1619 4010 0000 x
INCOMING PCI-X, OR PCI-E BUS TRANSACTION

FIG. 14

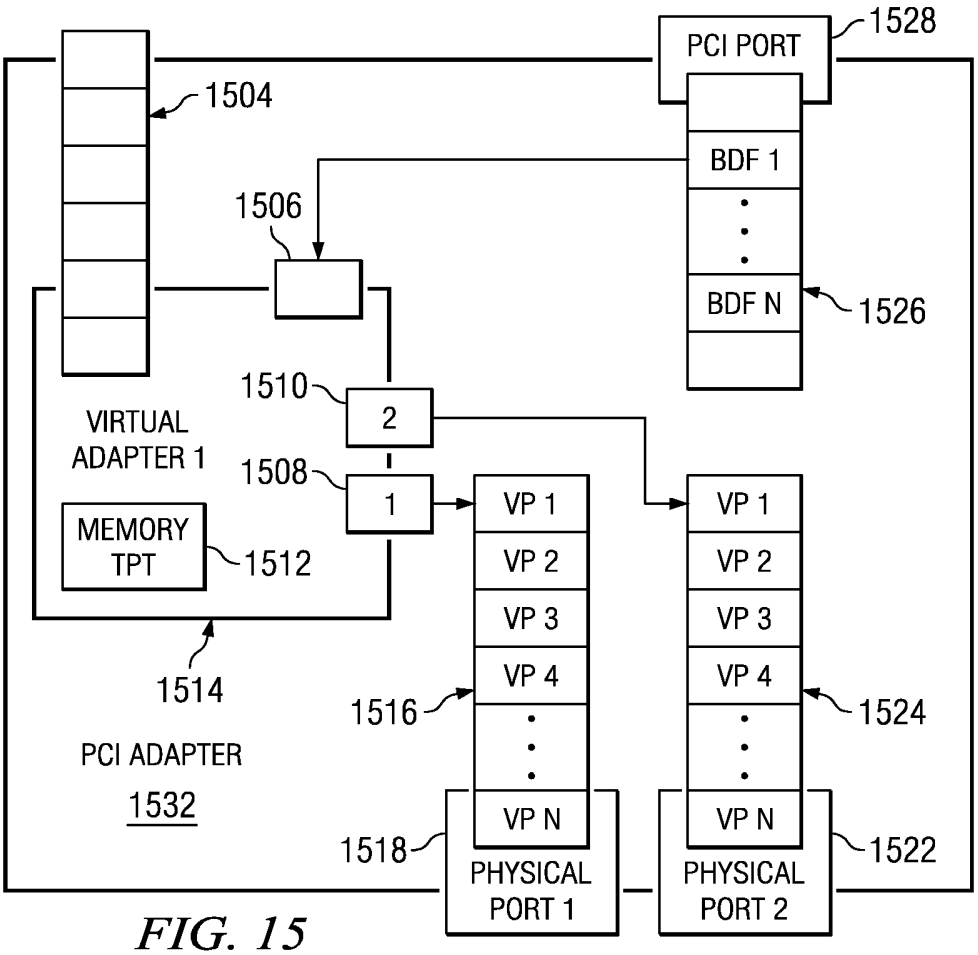

PCI PORT
1528
1504
BDF 1
1506
BDF N
1526
1510
2
VIRTUAL
ADAPTER 1
1508
1
VP 1
VP 2
VP 3
VP 4
VP N
MEMORY
TPT
1512
1514
1516
1524
PCI ADAPTER
1532
1518
PHYSICAL
PORT 1
PHYSICAL
PORT 2
1522

FIG. 15

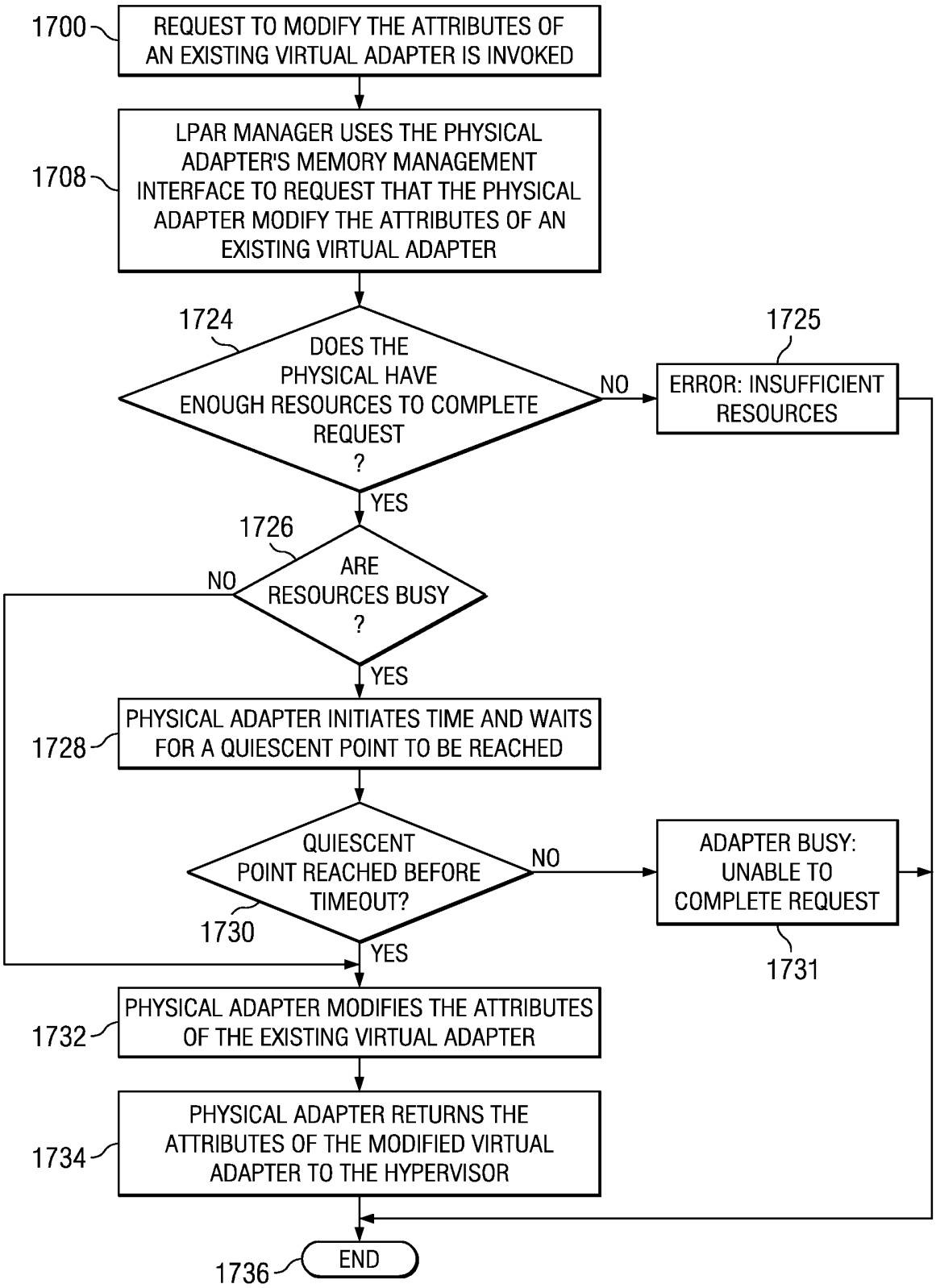

1700
REQUEST TO MODIFY THE ATTRIBUTES OF AN EXISTING VIRTUAL ADAPTER IS INVOKED
1708
LPAR MANAGER USES THE PHYSICAL ADAPTER'S MEMORY MANAGEMENT INTERFACE TO REQUEST THAT THE PHYSICAL ADAPTER MODIFY THE ATTRIBUTES OF AN EXISTING VIRTUAL ADAPTER
1724
DOES THE PHYSICAL HAVE ENOUGH RESOURCES TO COMPLETE REQUEST ?
NO
1725
ERROR: INSUFFICIENT RESOURCES
YES
1726
ARE RESOURCES BUSY ?
NO
YES
1728
PHYSICAL ADAPTER INITIATES TIME AND WAITS FOR A QUIESCENT POINT TO BE REACHED
QUIESCENT POINT REACHED BEFORE TIMEOUT?
1730
NO
ADAPTER BUSY: UNABLE TO COMPLETE REQUEST
1731
YES
1732
PHYSICAL ADAPTER MODIFIES THE ATTRIBUTES OF THE EXISTING VIRTUAL ADAPTER
1734
PHYSICAL ADAPTER RETURNS THE ATTRIBUTES OF THE MODIFIED VIRTUAL ADAPTER TO THE HYPERVISOR
1736
END

FIG. 16

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2006/060187

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/455

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

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

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

EPO-Internal, WPI Data, PAJ, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	EP 1 508 855 A (VIRTUAL IRON SOFTWARE, INC) 23 February 2005 (2005-02-23) paragraphs [0004] - [0045] paragraphs [0052] - [0058] paragraphs [0062], [0063] paragraphs [0069] - [0076] paragraphs [0086] - [0095] paragraphs [0120] - [0126] figures 1-9	1-20

-/--

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
23 May 2006	19/06/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Noll, J

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2006/060187

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	ANONYMOUS: "Virtual Interface Architecture Specification" MICROSOFT (SPECIFICATION), [Online] 16 December 1997 (1997-12-16), XP002377442 Retrieved from the Internet: URL:http://rimonbarr.com/repository/cs614/san_10.pdf> [retrieved on 2006-04-19] cited in the application page 11 - page 12 page 20 - page 22 pages 55-57 pages 64-66	1-20
A	US 6 111 894 A (BENDER ET AL) 29 August 2000 (2000-08-29) column 1, line 49 - column 3, line 37 column 3, line 60 - column 8, line 67 figures 1-5	1-20
P,X	US 2005/102682 A1 (SHAH RAJESH ET AL) 12 May 2005 (2005-05-12) paragraphs [0011] - [0038] paragraphs [0001] - [0007]	1-20

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2006/060187

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1508855 A	23-02-2005	US 2005044301 A1	24-02-2005
		US 2005080982 A1	14-04-2005
		WO 2005020073 A2	03-03-2005
US 6111894 A	29-08-2000	NONE	
US 2005102682 A1	12-05-2005	WO 2005050443 A2	02-06-2005