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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 October 2012 (26.10.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/143947 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/IN2012/000280

(22) International Filing Date:
18 April 2012 (18.04.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
1334/CHE/2011 18 April 2011 (18.04.2011) IN

(71) Applicant *(for all designated States except US)*: **INEDA SYSTEMS PVT. LTD** [IN/IN]; 8-2-120/115/C, Sudha Enclave, Road No. 2, Banjara Hills, Hyderabad 500034 (IN).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **KANIGICHERLA, Balaji** [US/IN]; 8-2-120/115/C, Sudha Enclave, Road No. 2, Banjara Hills, Hyderabad 500034 (IN). **TANDABOINA, Krishna, Mohan** [IN/IN]; 8-2-120/115/C, Sudha Enclave, Road No. 2, Banjara Hills, Hyderabad 500034 (IN). **VOLETI, Siva Raghuram** [IN/IN]; 8-2-120/115/C, Sudha Enclave, Road No. 2, Banjara Hills, Hyderabad 500034 (IN). **CHETTIAR, Chandra, Kumar** [IN/IN]; 8-2-120/115/C, Sudha Enclave, Road No. 2, Banjara Hills, Hyderabad 500034 (IN). **DOMMETI, Surya, Narayana** [IN/IN]; 8-2-120/115/C, Sudha Enclave, Road No. 2, Banjara Hills, Hyderabad 500034 (IN). **ARUMILLI, Kishor** [IN/IN]; 8-2-120/115/C, Sudha Enclave, Road No. 2, Banjara Hills, Hyderabad 500034 (IN).

(74) Agents: **LAKSHMIKUMARAN, Varadachari** et al.; B-6/10, Safdarjung Enclave, New Delhi 110029 (IN).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: MULTI-HOST PERIPHERAL CONTROLLER

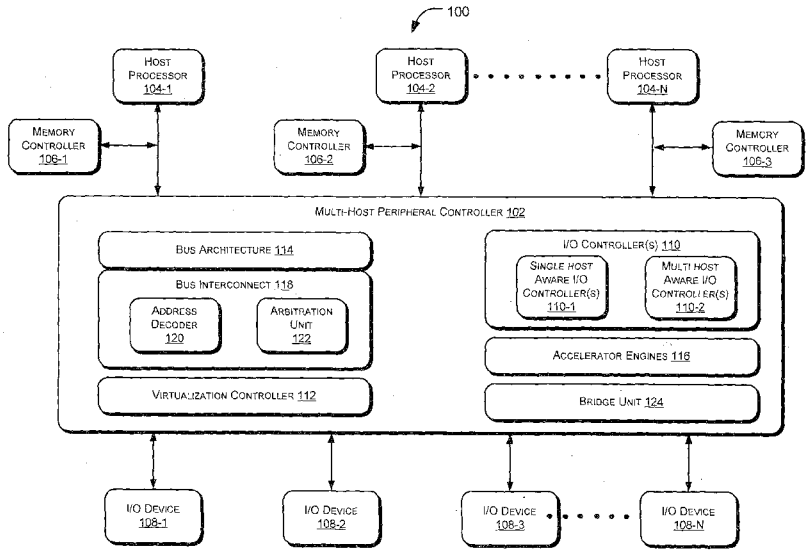

Fig. 1

(57) Abstract: Described herein is a multi-host peripheral controller configured to allow a plurality of host processors to simultaneously access the I/O devices through one of a per-host dedicated system bus architecture and common system bus architecture. Such a bus architecture is one of AXI, AHB, PCI, PCIe, OCP, etc. The multi-host peripheral controller allows operating systems, running on the plurality of host processors, to operate asynchronously with minimal or no inter operating system dependency. For addressability, either address based or sideband signal based demarcation may be used.

WO 2012/143947 A2

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

TECHNICAL FIELD

MULTI-HOST PERIPHERAL CONTROLLER

[0001] The present subject matter relates, in general, to multi-host processors and, in particular, to a multi-host peripheral controller for the multi-host processors.

BACKGROUND

[0002] A typical computing system includes, among other things, host processors, memory controllers (or a north bridge), media hardware accelerators, and a peripheral controller or a south bridge). While the memory controller helps the host processor to interface with a local memory, the peripheral controller provides the accesses to all the input and output (I/O) devices, viz., serial ATA (hard disk drive or BD/DVD/CD ROM drives), Universal Serial Bus (Key board/Mouse, external ports), Peripheral Connect Interface-Express (PCIe) expansion slots, Ethernet (network connectivity), etc.

[0003] Generally, the I/O devices are virtualized by software, such as a hypervisor, to enable multiple hosts to share the same system resources. The hypervisor or a virtual machine monitor (VMM) provides a platform for isolated execution of system images, and manages access between the system images and the attached I/O devices. Standards for PCIe based I/O virtualization, where multiple system images are implemented on a single host processor, are specified by Peripheral Component Interconnect Special Interest Group (PCI-SIG) in the single root input-output virtualization (SR-IOV) standard. However, the overhead of such virtualization techniques results in lesser performance and consume higher compute resources.

[0004] The capabilities of the SR-IOV standard have been extended by a multi root input-output virtualization (MR-IOV) standard to allow virtualization of the I/O devices between multiple host processors based on the standards of MR-IOV provided by the PCI-SIG. However, in order to virtualize the complete system in hardware following MR-IOV standards, a lot of components need to be made MR-IOV compliant. In addition, a MR-IOV switch needs to be implemented which adds additional component to the system.

SUMMARY

[0005] This summary is provided to introduce concepts related to a multi-host peripheral controller, which are further described in the detailed description. This summary is not intended to identify essential features of the present subject matter nor is it intended for use in determining or limiting the scope of the present subject matter.

[0006] In one implementation, the multi-host peripheral controller is configured to allow a plurality of host processors to simultaneously access the I/O devices through one of per-host dedicated system bus architecture and a common system bus architecture. Such bus architecture is one of AXI, AHB, PCI, PCIe, OCP, etc. The multi-host peripheral controller allows operating systems, running on the plurality of host processors, to operate asynchronously with minimal or no inter operating system dependency. For addressability, either address based or sideband signal based demarcation may be used.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] The detailed description is described with reference to the accompanying figures. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The same numbers are used throughout the drawings to reference like features and components.

[0008] Fig. 1 illustrates a multi-host computing system implementing a multi-host peripheral controller, according to an embodiment of the present subject matter.

[0009] Fig. 2 illustrates a multi-host computing system implementing a multi-host peripheral controller based on per-host dedicated system bus architecture, according to an embodiment of the present subject matter.

[0010] Fig. 3(a) illustrates a multi-host computing system implementing a multi-host peripheral controller based on common system bus architecture, according to another embodiment of the present subject matter.

[0011] Fig. 3(b) illustrates address based demarcation, according to another embodiment of the present subject matter.

[0012] Fig. 3(c) illustrates sideband signal based demarcation, according to another embodiment of the present subject matter.

DETAILED DESCRIPTION

[0013] The present subject matter is based on the platform system implementation, and, in particular, to integrated circuit chip implementation, which enables multiple hosts to concurrently access the I/O devices. In general, I/O

virtualization relates to a capability of the I/O devices to be used by more than one system image, in other words, by more than one operating system executing on a single host processor. Conventionally, a virtualization intermediary (VI) or a hypervisor, such as a virtual machine monitor (VMM), is used to enable sharing of the I/O devices connected to the single host processor. For a single root I/O virtualization, where multiple system images work on a single host processor and share virtualized I/O devices, a single root input-output virtualization (SR-IOV) has been developed by the Peripheral Component Interconnect Special Interest Group (PCI-SIG).

[0014] Additionally, the standard for virtualizing I/O devices and further, routing information between the virtualized I/O devices and the multiple host processors based on PCIe protocol has been defined by the PCI-SIG in the multi-root I/O virtualization (MR-IOV) standard. However, for virtualizing the entire system using MR-IOV, a lot of MR-IOV compliant components are used. To this end, method and systems to enable a plurality of hosts running parallel operating systems to share a given set of I/O devices with minimum or no communication between each other are described herein.

[0015] In one embodiment, a multi-host peripheral controller includes I/O controllers, which may either be single host aware or multi-host aware. The single host aware I/O controllers may be virtualized through a virtualization controller. Depending on the nature of the interface protocol supported by the I/O controller, or processing protocol supported by one or more acceleration engines, the virtualization method changes. Accordingly, the I/O controllers and the accelerator engines can arbitrate between the host processors at a determined boundary. Further, for communication between the host processors and I/O controllers/accelerator engines, a bus architecture is provided. The implementation of the multi-host peripheral controller may either be based on per-host dedicated system bus architecture or common system bus architecture.

[0016] Devices that can implement the disclosed system(s) and method(s) include, but are not limited to, desktop computers, hand-held devices, multiprocessor systems, microprocessor based programmable consumer electronics, laptops, network computers, minicomputers, mainframe computers, and the like which utilize multiple processors on the same hardware platform. In one implementation, the method can also be implemented for systems running any operating system such as Linux, Unix, Microsoft® Windows®, Mac OS X®, Android, and the like. Although the description herein is with reference to certain multi-host computing systems running particular

operating systems, the systems and methods may be implemented in other operating systems and computing systems, albeit with a few variations, as will be understood by a person skilled in the art.

[0017] Multiple operating systems are typically used to perform different functions on the same hardware platform. Each operating system may provide a particular advantage over different operating system (OS). For example, in a multi-host computing system which may run two different operating systems, OS 1 and OS 2, the OS 1 may provide better performance or support more applications than OS 2 however, the OS 2 may consume less resources such as memory, processing power, battery power when compared to OS 1. In such a scenario, the computing system may implement OS 1 for application processing and computational purposes whereas may implement OS 2 during idle state.

[0018] Fig. 1 illustrates a multi-host computing system 100 implementing a multi-host peripheral controller 102, according to an embodiment of the present subject matter. In said embodiment, the multi-host computing system 100, hereinafter referred to as system 100, includes host processors 104-1, 104-2,...,104-N, collectively referred to as host processors 104. The host processors 104 may include microprocessors, microcomputers, microcontrollers, digital signal processors, central processing units, state machines, logic circuitries, and/or any devices that manipulate signals and data based on operational instructions. Further, the system 100 includes memory controllers 106-1, 106-2,...,106-N. Optionally, each of the host processors 104 may have a dedicated graphics controller (not shown in the figure).

[0019] The system 100 further includes the intelligent peripheral controller 102 to allow the host processors 104 to simultaneously access one or more I/O devices 108-1, 108-2, ...,108-N, collectively referred to as I/O device(s) 108. Examples of the I/O devices 108 include USB devices, storage devices, communication devices, human interface devices, audio devices, etc. The I/O devices 108 may or may not be multi-host aware. Accordingly, in one implementation, the single host aware I/O controller(s) 110-1 corresponding to the I/O device 108 may be virtualized by building a virtualization wrapper around the I/O controller 110. Alternatively, the virtualization may be supported through a virtualization controller 112 and virtualization related components like inter host communication channels (not shown in the figure) included within the multi-host peripheral controller 102. The multi-host aware I/O controllers 110-2, on the other hand, are pre-configured to arbitrate between multiple host processors 104.

[0020] The I/O controllers 110 can be integrated with any standard bus architecture, such as the bus architecture 114. The bus architecture 114 includes one or more buses each capable of supporting different interface standard for different type of standard. Further, the bus architecture 114 is used to send commands and data from each of the I/O controllers 110 or accelerator engines 116, such as graphics engines, media accelerators, and security engines via buses. For example there may be an Advanced High-performance Bus (AHB) bus for commands and AXI bus for data transfers. Further, the I/O controllers 110 may read commands from the host processor's system memory connected to the memory controller 106 through inbuilt DMAs. The I/O controllers 119 may be further configured to implement data transfers through DMA read-writes directly from the system memory connected to the host's memory controller 106. Further, the bus architecture 114 may either be per-host dedicated system bus architecture or common system bus architecture. Based on the type of bus architecture 114, the implementation of the multi-host controller 102 may vary. This is explained in detail with reference to subsequent figures.

[0021] The bus architecture 114 is also associated with a bus interconnect 118, which includes an address decoder 120 and an arbitration unit 122. In one implementation, the bus interconnect 118 helps to interconnect master units and slave units with each other. A master unit, for example, the host processor 104-1, can request ownership of the bus, while a slave unit, such as the I/O device 108, is passive and only respond to requests. The arbitration unit(s) 122 is configured to determine bus ownership based on inputs received from each of requesting unit, for example the host processors 104 and I/O devices 108. The arbitration unit(s) 122 may then grant bus ownership to the requesting unit, as determined by the bus protocol. Further, the bus interconnect 118 may either be single layered or multiple layered. In case of single layered bus interconnect 118; one transfer is supported at a time. Thus, at any point of time, one master unit is allowed to interact with one slave unit. In this case, the address decoding and inter host arbitration may be performed by a single address decoder 120 and the arbitration unit 122 included within a central block.

[0022] However, if the bus interconnect 118 is multiple layered, multiple transfers can be supported, which means that while one master unit is interacting with one slave unit, another master unit may interact with another slave unit. Such parallel paths enable higher bandwidth. As each of the slave unit can have an active transfer in parallel, an individual arbitration unit 122 is associated with each slave unit. Further, in

order to determine which slave unit is being targeted by the master unit, address decoders 120 may be placed physically closer to the master unit.

[0023] Further, the multi-host peripheral controller 102 may also include a bridge unit 124 for handling protocol conversion between two buses operating on different protocols, traffic segmentation, frequency translation, and/or bus-width modification.

[0024] In operation, the multi-host peripheral controller 102 is configured to enable a plurality of host processors 104 running on different operating systems to share the I/O devices 108 with minimal or no inter-communication. The multi-host peripheral controller 102 does not require virtualization of the bus architecture 114 as the I/O controllers 110 are configured to cater to multiple host processors 104 simultaneously. Depending on the nature of the interface protocol supported by the I/O controller 110 or processing protocol supported by the acceleration engines 116, the virtualization method changes. Accordingly, the I/O controllers 110 and the accelerator engines 116 can arbitrate between the host processors 104 at a determined boundary. It will be noted that a part of the I/O controllers 110 and the accelerator engines 116 support only stateless transfers (i.e., context can be changed from transaction to transaction), whereas a part of the I/O controllers 110 and the accelerator engines 116 complete a given flow of transactions before the context can be changed. For example, for a USB flash drive, a set of transactions go through the USB host controller before the transaction context can change. So a read transfer going to a sector needs to be completely finished before a read to another sector is initiated.

[0025] In one implementation, for all the I/O controllers 110 and the accelerator engines 116, which can have a context change per transfer, the host processors 104 may be arbitrated per transfer. For the I/O controllers 110 and the accelerator engines 116, which require maintaining context for a few transfers, the arbitration can be done at a context switch point. The multi-host aware I/O controllers and/or accelerator engines 116 take care of this arbitration while serving the host processors 104, simultaneously.

[0026] Fig. 2 illustrates a multi-host computing system 100 implementing a multi-host peripheral controller 102 based on per-host dedicated system bus architecture, according to an embodiment of the present subject matter. As mentioned before, the system 100 includes host processors 104, memory controllers 106, and a multi-host peripheral controller 102 to interact with I/O devices 108 (not shown in this figure). As an illustration, only two host processors 104 are shown to interact with the

I/O devices 108; however multiple host processors 104 may be present as will be understood by a person skilled in the art.

[0027] In said embodiment, the multi-host peripheral controller 102 includes one or more bridge units 124 for protocol conversion between buses operating on different standards, etc., the accelerator engines 116, and the virtualization controller 112. In addition, as mentioned in Fig. 1, the multi-host peripheral controller 102 includes I/O controllers 110, which may either be single host aware or multi-host aware. Further, such I/O controllers 110 may either be dedicated to the host processors 104 or shared between the plurality of host processors 104. For example, the dedicated I/O controllers 202-1 are I/O controllers for I/O devices 108 dedicated to host processor 104-1, whereas the dedicated I/O controllers 202-2 are peripheral controllers for I/O devices 108 dedicated to host processor 104-2. Shared I/O controllers 204 are peripheral controllers for peripherals that are shared between the host processor 104-1 and 104-2. Further, each of the host processors 104 has a dedicated bus. Thus, a bus 206-1 sends commands and data to and from the host processor 104-1 and another bus 206-2 is meant for same purpose for host processor 104-2. It will be appreciated that even though the bus 206-1 and 206-2 is shown as single interfaces, both may support multiple interface standards for different types of transfers.

[0028] Further, as shown in Fig. 2 with the help of bi-directional arrows, the shared I/O controllers 204 can interface with multiple host-specific bus architecture. Thus, the shared I/O controllers 204 are configured to receive commands/requests from the plurality of host processors 104 through the corresponding host-specific bus interface. In said implementation, the shared I/O controllers 204 keep a register copy of each of the host processors 104, which may command the shared I/O controllers 204 for I/O device access. The host-specific interface also supports the copy of host registers.

[0029] The shared I/O controllers 204 based on the commands received from each of the host processors 104, on the host specific bus interface, schedule the received requests from each of the host processor 104 and route the data, responses and interrupts back to the appropriate host processor 104. Accordingly, host-specific commands are executed.

[0030] Fig. 3(a) illustrates a multi-host computing system 100 implementing a multi-host peripheral controller 102 based on common system bus architecture, according to another embodiment of the present subject matter. As mentioned before, the system 100 includes host processors 104, memory controllers 106, and a multi-host

peripheral controller 102 to interact with I/O devices 108 (not shown in this figure). As an illustration, only two host processors are shown to interact with the I/O devices 108; however multiple host processors may be present as will be understood by a person skilled in the art.

[0031] In said embodiment, the multi-host peripheral controller 102 includes one or more bridge units 124 for protocol conversion between buses operating on different standards, etc., the accelerator engines 116, and the virtualization controller 112. In addition, as mentioned in Fig. 1, the multi-host peripheral controller 102 includes I/O controllers 110, which may either be single host aware or multi-host aware. Further, each of the host processors 104 has a dedicated bus. Thus, the bus 206-1 sends commands and data to and from the host processor 104-1 and the bus 206-2 commands and data to and from the host processor 104-2. However, the bus 206-1 and bus 206-2 merge into a bus 302. The bus 302 is connected to all the I/O controllers 110, accelerators engines 116, and the virtualization controller 112. It will be appreciated that even though the bus 206-1 and 206-2 is shown as single interfaces, both may support multiple interface standards for different types of transfers.

[0032] As shown in Fig. 3(a) with the help of bi-directional arrows, the DMAs (not shown in this figure) within the I/O controllers 110 transfer data to the desired host processor's 104 system memory connected to the Memory controller 106 through a common bus interface. As a result, address/protocol specific signal based demarcation of the I/O controllers 110 and/or the accelerator engines 116 configuration space from the host's 104 perspective and address based demarcation of the host's 104 memory space from the I/O controllers 110 and/or accelerator engines 116 perspective is employed to ensure that the transfers happen from a source to an intended destination, i.e., between a particular host processor 104 and a particular I/O controller 110 space. To address addressability on the common system bus, either address based demarcation or sideband signal based demarcation may be performed.

[0033] According to an embodiment of the present subject matter, the address based demarcation includes allocating separate address spaces to each host processor 104, I/O controllers 110, and host memory, etc. Referring to Fig. 3(b), such address demarcation may be enabled through a bus interconnect 118 (also shown in Fig. 3(a)). The bus interconnect 118 extends addresses say from "N" bits on the host processor 104 side to "N+M" bits on the I/O controller 110 side through a control path arbiter 304, which also arbitrates between the host processor 104-1 and the host processor 104-2. As

mentioned before, the bus interconnect 118 interfaces the host processors 104, for example a first host processor 104-1 and a second host processor 104-2, with the I/O devices 108. The addressing on the host side is N-bit, while the addressing on the peripheral side is N+M bits wide, where M is a function of the number of host processors 104. The bus interconnect 118 maps the host's system views to contiguous spaces. While each host can see only 2^N space from its side, the I/O devices 108 in the system can see up to 2^{N+M} address space. From each of the I/O controller 110 and/or accelerator engine's 116 perspective, each host processor 104 can address its corresponding register set through a unique address space. While initiating a DMA transfer from the I/O controller, each host's 104 local memory can be addressed at a unique location. Though the above example states the spaces to be contiguous, different implementations are possible where host processors 104 can be placed in non-contiguous space as well.

[0034] According to another embodiment of the present subject matter shown in Fig. 3(c), the sideband signal based demarcation includes assigning a separate ID to each host processor 104, which can then be routed through sideband signals. The following example describes a system where all host's 104 interface with the I/O controllers 110 through a single on chip bus interconnect 118 and the DMA 306 of the I/O controllers 110 initiate transfers on to the hosts 104 through the same bus interconnect 118. For the sake of simplicity in explanation, the implementation described herein shows only one standard being followed in both the directions, however there can be different implementations using different bus standards, such as AHB, PCI, PCIe, OCP, etc., for the host processor 104 to the I/O controller 110 and the IO controller 110 to the host processor 104.

[0035] Further, in said implementation, the bus interconnect 118 adds a host ID sideband signal for each of the host interface. After arbitration by the control path arbiter 304, the interface is routed to an I/O controller 110, such as the multi-host aware I/O controller 110-2. Based on the host ID generated by the bus interconnect 118, the I/O controller 110-2 can direct the accesses on the command bus IF to the host specific register space. On the Data Bus IF from a data path arbiter 308, the DMA 306 generates the host ID sideband signal based on the target host processor 104. The bus interconnect 118 uses the host ID to route the transfers from the I/O controller 110-2 to the appropriate host processor 104.

[0036] Even though the example shows one implementation of the sideband signals; however, it will be understood by a person skilled in the art that different variations in the implementation may be possible where the hosts 104 can generate a host ID along with the bus interface signals (instead of bus interconnect 118), I/O controllers 110 can use the host ID as an extension to the address. In short the value on the host ID sideband signal can be used in different ways to enable transfers accesses between multiple hosts and slaves.

[0037] Although implementations of the multi-host peripheral controller 102 have been described in language specific to structural features and/or methods, it is to be understood that the invention is not necessarily limited to the specific features or methods described. Rather, the specific features and methods are disclosed as exemplary implementations for the multi-host peripheral controller 102.

We claim:

1. A multi-host computing system (100) comprising:

a multi-host peripheral controller (102) configured to share a input-output (I/O) device (116) amongst a plurality of host processors (104), wherein each of the plurality of host processors (104) is configured to run operating systems asynchronously, in a multi-host computing system (100), the multi-host peripheral controller (102) comprising:

at least one I/O controller (110), communicatively coupled to at least one of the plurality of host processors (104), wherein the at least one I/O controller (110) is directly and simultaneously addressable by the host processors (104), and configured to:

control the I/O device (116); and

access a local memory associated with each of the plurality of host processors (104) for fetching commands and transferring data;

a bus architecture (114) configured to transmit instructions generated by of the plurality of host processors (104) to the I/O device (116);

a virtualization controller (112) configured to act as virtualization wrapper for the I/O controller (110), wherein the virtualization wrappers enable virtualization of the I/O device (116) with respect to the operating system running on each of the plurality of host processors (104).

2. The multi-host computing system (100) as claimed in claim 1, wherein the bus architecture (114) is further configured to support bus architecture conforming to at least one of Advanced eXtensible Interface (AXI) standards, Advanced High-performance Bus (AHB) standards, Peripheral Controller Interface (PCI) standards, Peripheral Controller Interface express (PCIe) standards, and an Open Core Protocol (OCP) standards.

3. The multi-host computing system (100) as claimed in claim 1, wherein the bus architecture (114) is dedicatedly controlled by each of the plurality of host processors (104).

4. The multi-host computing system (100) as claimed in claim 1, wherein the bus architecture (114) is dedicatedly controlled by the plurality of host processors (104) simultaneously.

5. The multi-host computing system (100) as claimed in claim 1, wherein the multi-host peripheral controller (102) further comprises a bus interconnect (118) configured to interconnect at least one of the plurality of host processors (104) with the I/O device (116).

6. The multi-host computing system (100) as claimed in claim 5, wherein the bus interconnect (118) further comprises:

an arbitration unit (122) configured to,

determine ownership of the bus architecture (114) based on signals received from at least one of the plurality of host processors (104) and the I/O device (116);

grant access of the bus architecture (114) to at least one of the plurality of host processors (104) based on the determined ownership; and

an address decoder (120), communicatively coupled to the arbitration unit (122) and configured to arbitrate requests to access the bus architecture (114) from a plurality of host processors (114).

7. The multi-host computing system (100) as claimed in claim 1, wherein the multi-host peripheral controller (102) is further configured to demarcate each of the plurality of host processors (114) based on access of addresses on the bus architecture (114)

8. The multi-host computing system (100) as claimed in claim 1, wherein the multi-host peripheral controller (102) further comprises a bridge unit (124) configured to translate commands received from the plurality of host processors (104) from a first protocol to a second protocol, wherein the first protocol and the second protocol conform to at least one of Advanced eXtensible Interface (AXI) standards, Advanced High-performance Bus (AHB) standards, Peripheral Controller Interface (PCI) standards, Peripheral Controller Interface express (PCIe) standards, and an Open Core Protocol (OCP) standards.

9. The multi-host computing system (100) as claimed in claim 1, wherein the at least one I/O controller (110) further comprises at least one of a Single Root I/O Virtualization (SRIOV) controller (110-1) configured to interface with I/O device (116) not supporting virtualization and a Multi-Root I/O Virtualization (MRIOV) controller (110-2) configured to interface with the I/O device (116) supporting virtualization.

10. The multi-host computing system (100) as claimed in claim 1, wherein the multi-host peripheral controller (102) is further configured to assign each of the plurality host processors (104) an unique identity (ID) so as to demarcate between each of the plurality of host processors (104) using a sideband signal, wherein the sideband signal comprises the ID for each of the plurality of host processors (104).

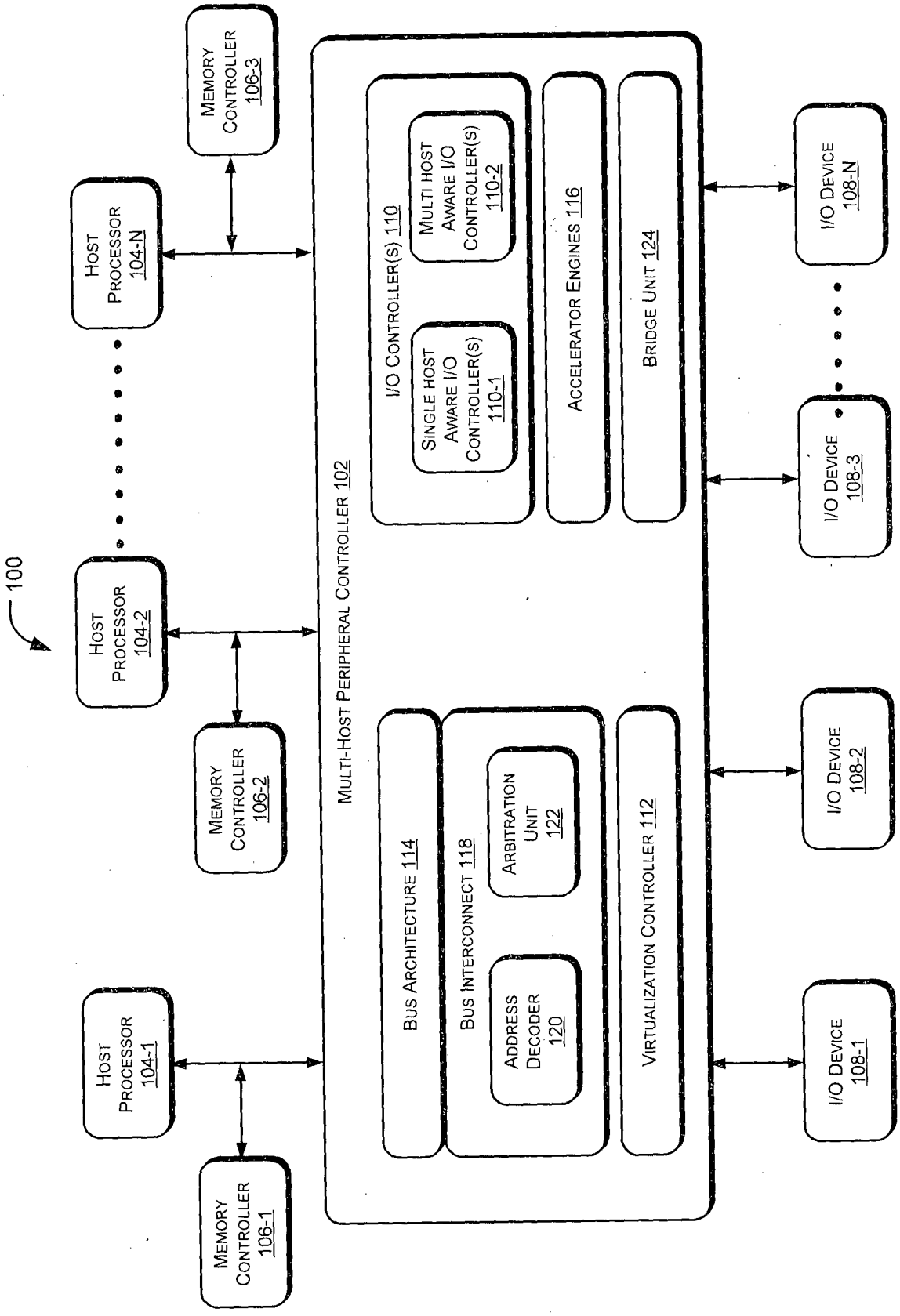
100
HOST PROCESSOR 104-1
HOST PROCESSOR 104-2
HOST PROCESSOR 104-N
MEMORY CONTROLLER 106-1
MEMORY CONTROLLER 106-2
MEMORY CONTROLLER 106-3
MULTI-HOST PERIPHERAL CONTROLLER 102
BUS ARCHITECTURE 114
BUS INTERCONNECT 118
ARBITRATION UNIT 122
ADDRESS DECODER 120
VIRTUALIZATION CONTROLLER 112
I/O CONTROLLER(S) 110
SINGLE HOST AWARE I/O CONTROLLER(S) 110-1
MULTI HOST AWARE I/O CONTROLLER(S) 110-2
ACCELERATOR ENGINES 116
BRIDGE UNIT 124
I/O DEVICE 108-1
I/O DEVICE 108-2
I/O DEVICE 108-3
I/O DEVICE 108-N

Fig. 1

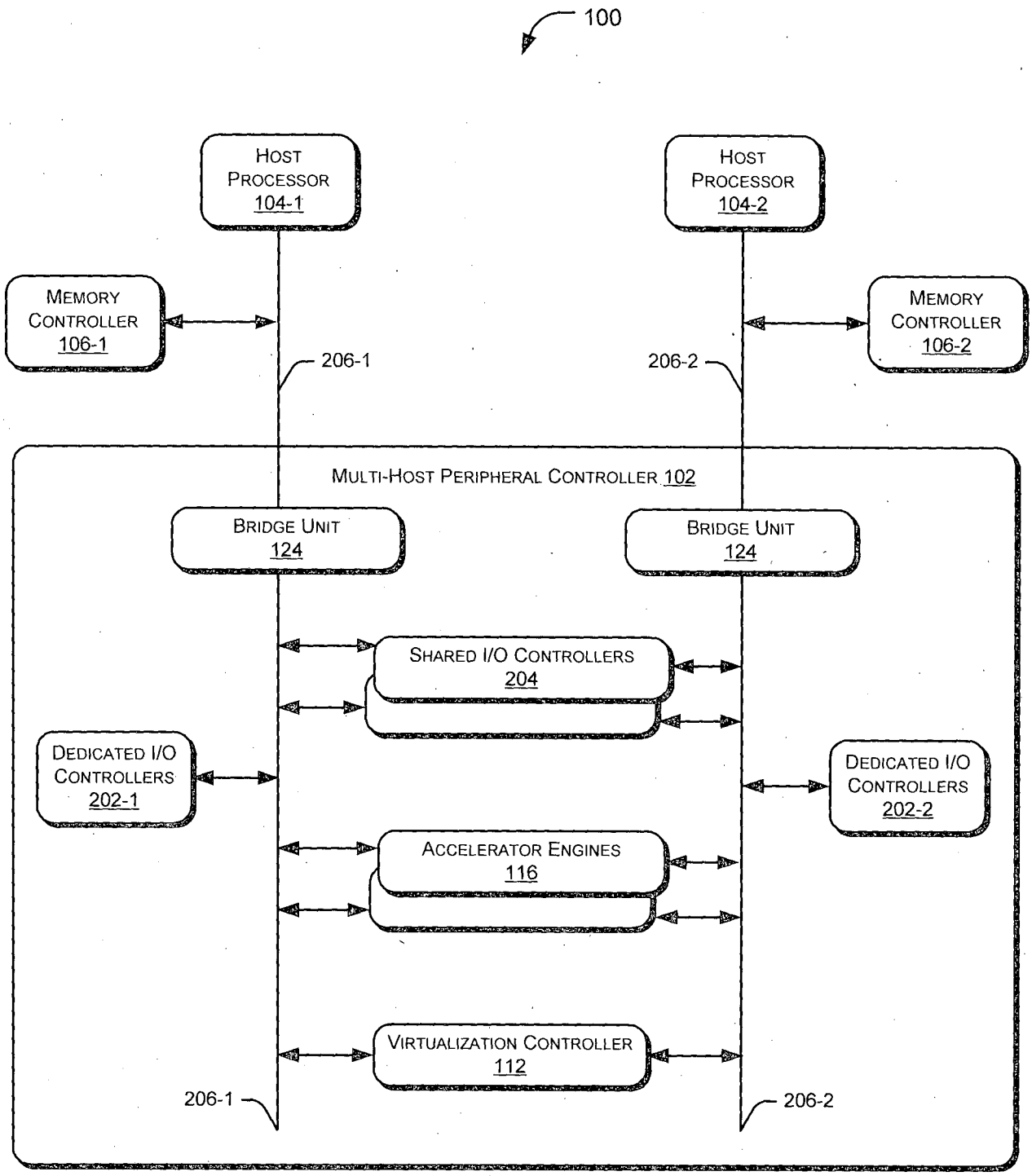
100
HOST PROCESSOR 104-1
HOST PROCESSOR 104-2
MEMORY CONTROLLER 106-1
MEMORY CONTROLLER 106-2
206-1
206-2
MULTI-HOST PERIPHERAL CONTROLLER 102
BRIDGE UNIT 124
BRIDGE UNIT 124
SHARED I/O CONTROLLERS 204
DEDICATED I/O CONTROLLERS 202-1
DEDICATED I/O CONTROLLERS 202-2
ACCELERATOR ENGINES 116
VIRTUALIZATION CONTROLLER 112
206-1
206-2

Fig. 2

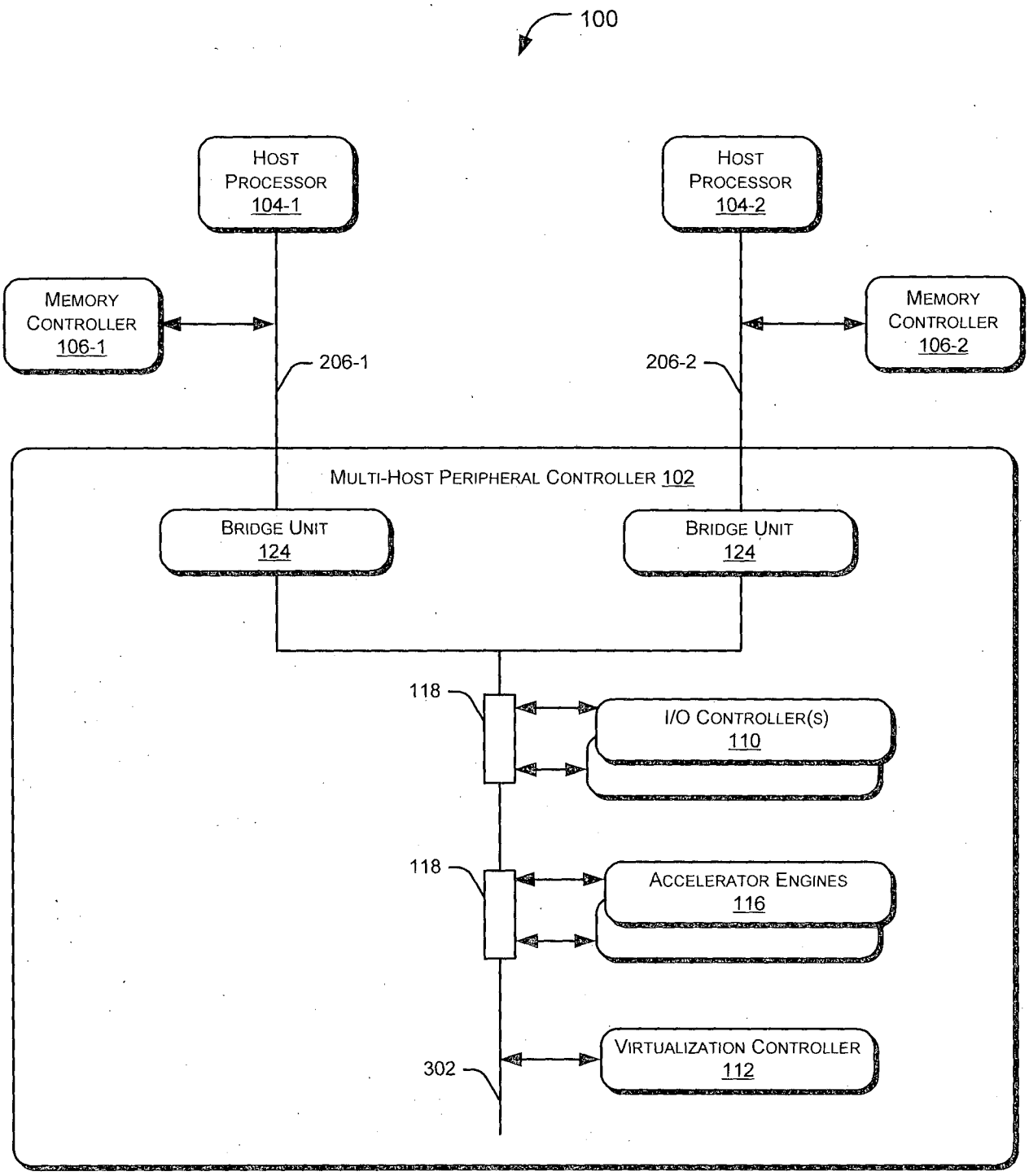
100
HOST PROCESSOR 104-1
HOST PROCESSOR 104-2
MEMORY CONTROLLER 106-1
MEMORY CONTROLLER 106-2
206-1
206-2
MULTI-HOST PERIPHERAL CONTROLLER 102
BRIDGE UNIT 124
BRIDGE UNIT 124
118
I/O CONTROLLER(S) 110
118
ACCELERATOR ENGINES 116
302
VIRTUALIZATION CONTROLLER 112

Fig. 3(a)

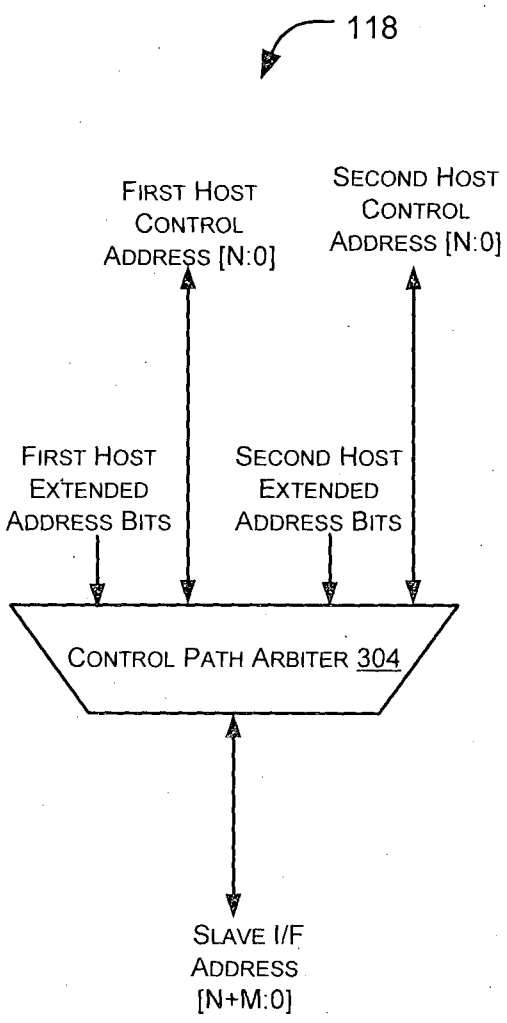
118
FIRST HOST CONTROL ADDRESS [N:0]
SECOND HOST CONTROL ADDRESS [N:0]
FIRST HOST EXTENDED ADDRESS BITS
SECOND HOST EXTENDED ADDRESS BITS
CONTROL PATH ARBITER 304
SLAVE I/F ADDRESS [N+M:0]

Fig. 3(b)

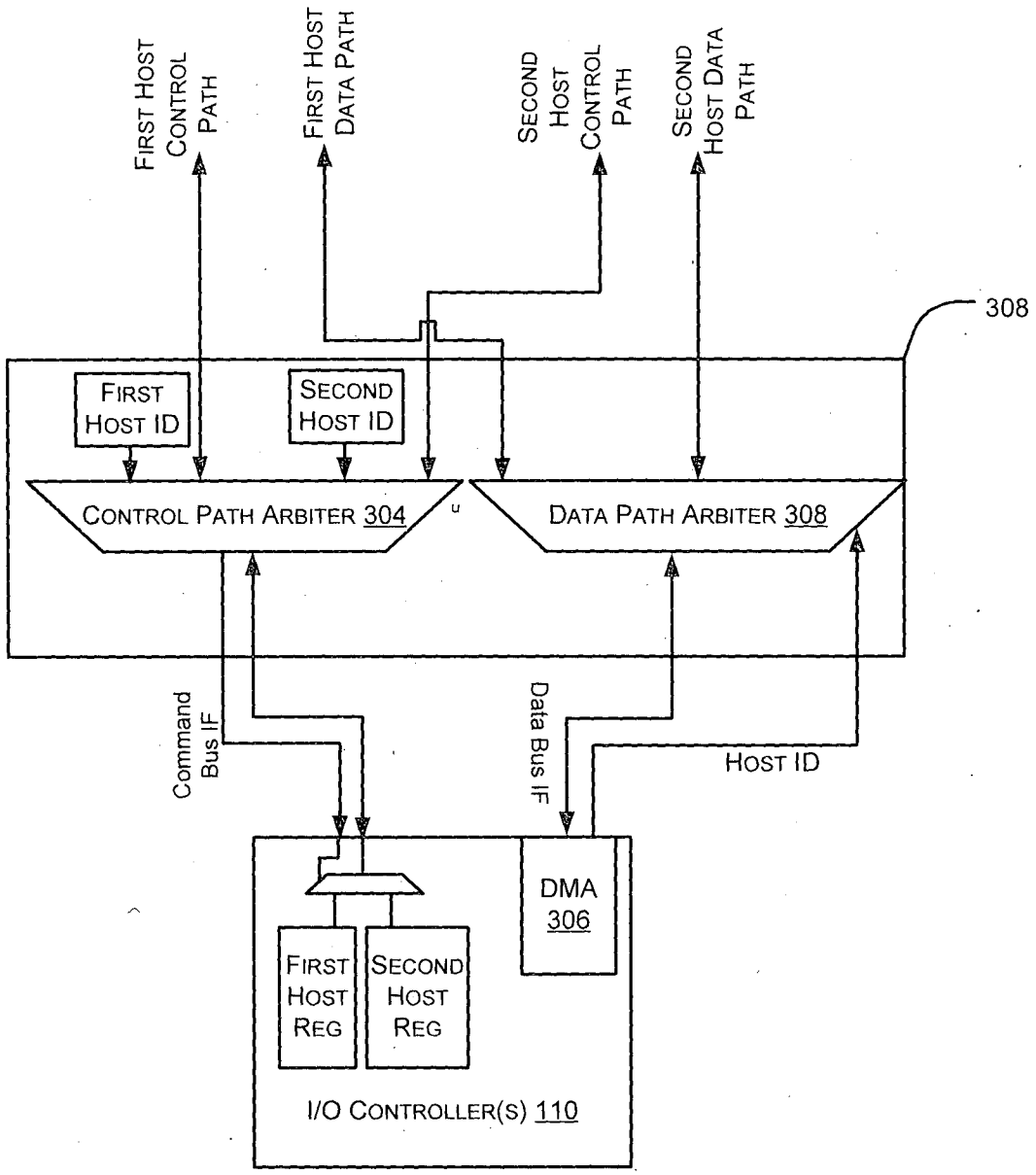
First Host Control Path
First Host Data Path
Second Host Control Path
Second Host Data Path
308
First Host ID
Second Host ID
Control Path Arbiter 304
Data Path Arbiter 308
Command Bus IF
Data Bus IF
Host ID
DMA 306
First Host Reg
Second Host Reg
I/O Controller(s) 110

Fig. 3(c)