

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
8 June 2006 (08.06.2006)

PCT

(10) International Publication Number
WO 2006/060527 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/US2005/043396

(22) International Filing Date:
29 November 2005 (29.11.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
102004057756.0
30 November 2004 (30.11.2004) DE
11/230,979 20 September 2005 (20.09.2005) US

(71) Applicant (for all designated States except US): **ADVANCED MICRO DEVICES, INC.** [US/US]; One AMD Place, Mail Stop 68, P.O. Box 3453, Sunnyvale, CA 94088-3453 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **HESSE, Kay** [DE/DE]; Kronenstr 45, 01129 Dresden (DE). **MUELLER, Sven** [DE/DE]; Platanenstrasse 10, 01129 Dresden (DE).

(74) Agent: **DRAKE, Paul, S.**; Advanced Micro Devices, Inc., 5204 East Ben White Boulevard, Mail Stop 562, Austin, TX 78741 (US).

(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, DK, DM, DZ, EC, EE, EG, ES, FI, 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.

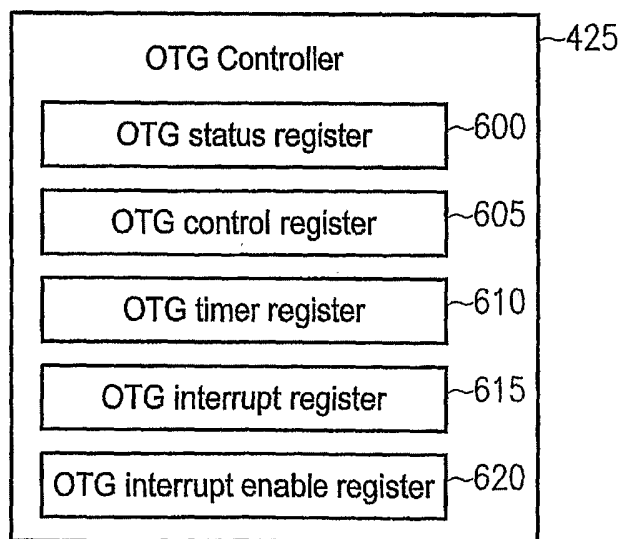
(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, 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:

— without international search report and to be republished upon receipt of that report

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.

(54) Title: USB ON-THE-GO CONTROLLER



(57) Abstract: A USB (Universal Serial Bus) controller technique for implementing OTG (On-The-Go) functionality is provided. The device may have an EHCI (Enhanced Host Controller Interface) compliant host control unit, and an OTG control unit to implement an OTG state machine partly in hardware and partly in software. The OTG control unit may have an OTG control register and an OTG status register which are accessible by software. Further, the USB controller device may have a device control unit to implement device functionality and a port multiplexer to assign a physical port to either the host or the device control unit. The OTG control unit may be comprised in the port multiplexer. Further, a software driver may read the OTG status register in response to receiving an interrupt from the USB controller device, and write to the OTG control register to force the USB controller device to change its OTG state.

TITLE: USB ON-THE-GO CONTROLLER**TECHNICAL FIELD**

The invention generally relates to USB (Universal Serial Bus) controller devices and methods, and in particular to implementing OTG (On-The-Go) functionality.

BACKGROUND ART

The Universal Serial Bus was originally developed in 1995 to define an external expansion bus which facilitates the connection of additional peripherals to a computer system. The USB technique is implemented by PC (Personal Computer) host controller hardware and software and by peripheral friendly master-slave protocols and achieves robust connections and cable assemblies. USB systems are extendable through multi-port hubs.

In USB systems, the role of the system software is to provide a uniformed view of the input/output architecture for all applications software by hiding hardware implementation details. In particular, it manages the dynamic attach and detach of peripherals and communicates with the peripheral to discover its identity. During run time, the host initiates transactions to specific peripherals, and each peripheral accepts its transactions and response accordingly.

Hubs are incorporated to the system to provide additional connectivity for USB peripherals, and to provide managed power to attached devices. The peripherals are slaves that must react to request transactions sent from the host. Such request transactions include requests for detailed information about the device and its configuration.

While these functions and protocols were already implemented in the USB 1.1 specification, this technique was still improved in order to provide a higher performance interface. FIG. 1 illustrates an example USB 2.0 system that comprises a host controller 100, a number of USB devices 115, 120, 125, 130, and two hubs 105, 110. In the system of FIG. 1, the hubs 105, 110 are introduced for increasing connectivity, but in other USB 2.0 systems, the USB devices can be connected directly to the host controller 100.

As mentioned above, USB 2.0 provides a higher performance interface, and the speed improvement may be up to a factor of 40. Moreover, as apparent from FIG. 1, USB 2.0 is backwards compatible with USB 1.1 because it allows for connecting USB 1.1 devices 120, 125, 130 to be driven by the same host controller 100. There may even be used USB 1.1 hubs 110.

As can be seen from FIG. 1, a USB 1.1 device 120 can be connected directly to a USB 2.0 hub 105. Moreover, it can also be connected directly to the host controller 100. This is made possible by the capability of USB 2.0 host controllers and hubs to negotiate higher as well as lower transmission speeds on a device-by-device basis.

Turning now to FIG. 2, the system software and hardware of a USB 2.0 system is illustrated. The system components can be organized hierarchically by defining several layers as shown in the figure.

In the uppermost layer, the client driver software 200 executes on the host PC and corresponds to a particular USB device 230. The client software is typically part of the operating system or provided with the device.

The USB driver 205 is a system software bus driver that abstracts the details of the particular host controller driver 210, 220 for a particular operating system. The host controller drivers 210, 220 provide a

software layer between a specific hardware 215, 225, 230 and the USB driver 205 for providing a driver-hardware interface.

While the layers discussed so far are software implemented, the uppermost hardware component layer includes the host controllers 215, 225. These controllers are connected to the USB device 230 that performs the end user function.

As apparent from the figure, there is one host controller 225 which is an enhanced host controller (EHC) for the high speed USB 2.0 functionality. This host controller operates in compliance with the EHCI (Enhanced Host Controller Interface) specification for USB 2.0. On the software side, host controller 225 has a specific host controller driver (EHCD) 220 associated.

Further, there are host controllers 215 for full and low speed operations. The UHCI (Universal Host Controller Interface) or OHCI (Open Host Controller Interface) are the two industry standards applied in the universal or open host controllers (UHC/OHC) 215 for providing USB 1.1 host controller interfaces. The host controllers 215 have assigned universal/open host controller drivers (UHCD/OHCD) 210 in the lowest software level.

Thus, the USB 2.0 compliant host controller system comprises driver software and host controller hardware which must be compliant to the EHCI specification. While this specification defines the register-level interface and associated memory-resident data structures, it does not define nor describe the hardware architecture required to build a compliant host controller.

Referring now to FIG. 3, the hardware components of a common motherboard layout are depicted. The basic elements found on a motherboard may include the CPU (Central Processing Unit) 300, a northbridge 305, a southbridge 310, and system memory 315. The northbridge 305 is usually a single chip in a core-logic chipset that connects the processor 300 to the system memory 315 and the AGP (Accelerated Graphic Port) and PCI (Peripheral Component Interface) buses. The PCI bus is commonly used in personal computers for providing a data path between the processor and peripheral devices like video cards, sound cards, network interface cards and modems. The AGP bus is a high-speed graphic expansion bus that directly connects the display adapter and system memory 315. AGP operates independently of the PCI bus. It is to be noted that other motherboard layouts exist that have no northbridge in it, or that have a northbridge without AGP or PCI options.

The southbridge 310 is usually the chip in a system core-logic chipset that controls the IDE (Integrated Drive Electronics) or EIDE (Enhanced IDE) bus, the USB bus, that provides plug-n-play support, controls a PCI-ISA (Industry Standard Architecture) bridge, manages the keyboard/mouse controller, provides power management features, and controls other peripherals.

USB functionality is often implemented in a southbridge device or in other companion chips that may be part of a personal computer's chipset. As described above, this functionality provides a popular interface for exchanging data between the computer that has built in the southbridge or companion chip, and peripherals.

In order to allow low-power devices to have limited host capabilities, the "On-The-Go Supplement to the USB 2.0 Specification" has been drafted. The supplement defines two protocols: SRP (Session Request Protocol), which lets a peripheral device ask the host to turn on the USB power, and HNP (Host Negotiation Protocol), which lets a peripheral device become the USB host. Thus, the supplement to the USB 2.0 specification allows a portable device to take on the role of a limited USB host without the burden of supporting all of the USB functions of a personal computer.

~~"Besides others, the OTG supplement"~~ specifies dual-role devices that may be operated in full speed (and optionally in high speed) as peripheral, and further in full speed (and optionally in low and high speed) as host. Further, an A-device is specified to be a host at the start of a session. Under certain conditions, the A-device will relinquish the role of host to a dual-role B-device. A B-device is a peripheral at the start of a session. If the device is dual-role, it may be granted the role of host from the A-device.

The two protocols mentioned above, SRP and HNP, are described in the OTG supplement by means of state diagrams. These state diagrams define a plurality of OTG states and the conditions under which transitions between these states occur. However, implementing state machine(s) to comply with the state diagrams specified in the OTG supplement is generally found to be quite complex. In particular, implementing the state machine(s) may require modifications to the circuits providing the conventional USB functionality. This may lead to increased circuit development and manufacturing costs.

DISCLOSURE OF INVENTION

An improved USB controller technique for implementing OTG functionality is provided that may be realized with reduced development and manufacturing costs.

In one embodiment, a USB controller device having OTG functionality is provided. The USB controller device comprises a host controller unit having an EHCI compliant host controller adapted to implement host functionality to perform data transfer to and/or from at least one USB peripheral device. The USB controller device further comprises an OTG control unit that is adapted to implement an OTG state machine to control operation of the host control unit. The OTG control unit is adapted to implement the OTG state machine partly in hardware and partly in software.

In another embodiment, a USB controller device is provided that has OTG functionality. The USB controller device comprises a host control unit that has an EHCI compliant host controller which is adapted to implement host functionality to perform data transfer and/or from at least one USB peripheral device. The USB controller device further comprises an OTG control unit that is adapted to control operation of the host control unit. The OTG control unit comprises an OTG control register that holds control information to control operation of the OTG control unit, and an OTG status register that holds status information of the OTG control unit. The OTG control and status registers are accessible by software.

According to a further embodiment, a USB controller device comprises a host control unit that has an EHCI compliant host controller which is adapted to implement host functionality to perform data transfer to and/or from at least one USB peripheral device. The USB controller device further comprises a device control unit which is adapted to implement device functionality to perform data transfer to and/or from at least one USB host device. Moreover, the USB controller device comprises a port multiplexer which is adapted to assign a physical port to either the host control unit or the device control unit, and an OTG control unit which is adapted to implement OTG functionality. The OTG control unit is comprised in the port multiplexer.

In still a further embodiment, a computer-readable storage medium stores instructions that, when executed on a processor, cause the processor to drive a USB controller device that has OTG functionality by implementing at least a part of an OTG state machine to control operation of an EHCI compliant host controller.

According to yet another embodiment, a computer-readable storage medium stores instructions that, when executed on a processor, cause the processor to drive a USB controller device that has OTG functionality

by accessing OTG control and status registers of the USB controller device. The OTG control and status registers hold OTG control and status information which is suitable for implementing the OTG functionality.

Furthermore, an embodiment provides a method of providing OTG functionality in a USB controller device that has an OTG control register and an OTG status register which are both accessible by a software driver. The method comprises reading, by the software driver, the OTG status register in response to receiving an interrupt from the USB controller device, and writing, by the software driver, to the OTG control register to force the USB controller device to change its OTG state.

BRIEF DESCRIPTION OF DRAWINGS

The accompanying drawings are incorporated into and form a part of the specification for the purpose of explaining the principles of the invention. The drawings are not to be construed as limiting the invention to only the illustrated and described examples of how the invention can be made and used. Further features and advantages will become apparent from the following and more particular description of the invention, as illustrated in the accompanying drawings, wherein:

- FIG. 1 illustrates an example USB 2.0 compliant system;
- FIG. 2 illustrates the hardware and software component layers in the system of FIG. 1;
- FIG. 3 illustrates a common motherboard layout;
- FIG. 4 is a block diagram illustrating the components of a USB controller device having OTG functionality, according to an embodiment;
- FIG. 5 is a block diagram illustrating the components of the host controller which is part of the arrangement of FIG. 4, according to an embodiment;
- FIG. 6 illustrates an OTG controller according to an embodiment; and
- FIG. 7 is a flow chart illustrating how a software controlled state transition can be achieved according to an embodiment.

MODE(S) FOR CARRYING OUT THE INVENTION

The illustrative embodiments of the present invention will be described with reference to the figure drawings wherein like elements and structures are indicated by like reference numbers.

Referring now to the drawings and in particular to FIG. 4, a USB controller device 400 according to an embodiment comprises a host control unit 415, a device control unit 410, a bridge 405, and a port multiplex controller 420. The port multiplex controller 420 of the present embodiment comprises an OTG controller 425 that implements OTG functionality to the USB controller device 400. As will be apparent from the description below, adding the OTG controller 425 to the port multiplex controller 420 allows for achieving OTG functionality with least efforts since standard or off-the-shelf circuits may be used to implement the host and device control units 415, 410 with only little extra hardware.

In the present embodiment, the host control unit 415 is an EHC/OHC combination, and the port multiplex controller 420 controls access of the host control unit 415 and the device control unit 410 to the shared physical port. As will be described in more detail below, the host control unit 415 comprises port route facilities for this purpose. The device control unit 410 is connected to the host control unit 415 by means of a device UTMI (USB 2.0 Transceiver Macrocell Interface) interface.

The bridge 405 may be applicable in a broad range of applications and may further be not restricted to the USB 2.0 subsystem of the present embodiment. The main task of bridge 405 is to convert bus protocols. Additionally, it may serve as an arbiter or decoder to map multiple ports of one protocol to one port of another protocol. In the present embodiment, the multiple ports may be non-packetized bus ports with burst and split operations, while the other port may be strictly packetized.

The device control unit 410 is used to implement device functionality to perform data transfer to and/or from a USB host device. The device control unit 410 may support multiple bi-directional logical endpoints.

The host control unit 415 is used to implement host functionality to perform data transfer to and/or from one or more USB peripheral devices. As will be discussed in the following, the host control unit 415 may support two (or more) USB ports, one EHC and one OHC companion controller.

Referring to FIG. 5, the main components of the host controller 415 according to an embodiment are shown. In general, the host controller 415, 500 comprises three main components: the enhanced host controller (EHC) 225, one or more companion host controllers 215, and the port router 515.

The enhanced host controller 225 handles the USB 2.0 high speed traffic.

In the companion host controller unit 215 of the present embodiment, there are two OHCI compliant host controllers, OHC0 505 and OHC1 510. These controllers handle all USB 1.1 compliant traffic and may contain the legacy keyboard emulation for non-USB aware environments.

The port router 515 assigns the physical port interfaces their respective owners. If a USB 2.0 aware driver is present in the system it will assign the ports to either a companion host controller 505, 510 for low and full speed devices and hubs (USB 1.1 traffic) or to the EHC 225 for high speed devices and hubs.

That is, the USB 2.0 host controller shown in FIG. 5 complies with the EHCI specification and allows for using existing OHCI USB 1.1 host controllers with the minimum alteration necessary to interface to the port router block 515, instead of USB 1.1 physical devices.

The USB 2.0 compliant host controller of FIG. 5 may be defined as hardware architecture to implement an EHCI-compliant host controller for integration into a southbridge 310. The host controller then resides between the USB-2 analog input/output pins and a link interface module for interfacing upstream towards system memory, e.g. interfacing to a northbridge if there is one present in the system.

As can be seen from FIG. 5, the port router 515 has additional interfaces to the device control unit 410 and the port multiplex controller 420 (which includes the OTG controller 425 in the present embodiment). The interface to the device controller is the above-mentioned UTMI interface. The interface to the port multiplex controller is a control interface for host/device switching.

Referring back to FIG. 4, it may be the port multiplex controller's basic function in the present embodiment to assign the shared physical port to either the host control unit 415 or the device control unit 410. In the present embodiment, this assignment is done in response to software instructions received at the port multiplex controller 420.

To achieve its functionality, the port multiplex controller 420 of the present embodiment comprises software accessible registers such as a capability register and a multiplexing control register. Both registers may be accessible by software to read data from given register fields, and write data to such fields. It is to be noted, however, that at least some of the fields may be read-only fields.

In the port multiplex controller capability register, there may be a register field to store automatic pull-up enable information which may be used to control how pull-up registers of the USB controller device are

activated when a port is assigned to the device control unit 410. The automatic pull-up enable information may indicate whether software needs to activate the pull-up, or whether the pull-up is activated as soon as the voltage on the USB port is in a valid voltage range.

There may be further register fields in the port multiplex controller capability register to store overcurrent polarities and information with respect to the overcurrent handling. For instance, information may be stored for controlling how the overcurrent reporting to certain statuses is handled. Moreover, the capability register may hold information on the port power polarity and handling for controlling how port power control information from the host controller is mapped.

A multiplexing control register of the port multiplex controller 420 may include register fields indicating whether the voltage on the USB port is in a valid range (i.e., above 4.4V), pull-up enable information indicating whether pull-up is disabled or activated, and port multiplexer control information indicating to which control unit a port is assigned, if any.

As described above, the OTG protocol support is implemented in the present embodiment as part of the port multiplex controller 420. This allows for using existing or off-the-shelf units to implement the controllers 410, 415. Further, the OTG control unit 425 of the present embodiment allows software to implement a part of the OTG state machines specified in the OTG supplement. To allow a USB software driver to implement a part of an OTG state machine, the OTG control unit 425 may provide status and control facilities which will now be described in more detail with reference to FIG. 6.

As apparent therefrom, the OTG control unit 425 has a number of registers 600-620 that are accessible by software in a read-only mode or in a write access mode. As shown in FIG. 6, the registers include an OTG status register 600, an OTG control register 605, an OTG timer register 610, an OTG interrupt register 615, and an OTG interrupt enable register 620.

Discussing first the OTG status register 600 which may be accessible by software in a read-only mode, the register holds status information of the OTG control unit 425 in a number of register fields.

In a first OTG status register field, timer expiration information may be stored. This information may be controlled by hardware to have a bit set when an OTG timer has expired. The bit may be deasserted when the timer is either currently counting or is in a stopped state.

In a further register field, the OTG status register 600 may hold port suspension information indicating whether a port assigned to the host control unit 415 or assigned to the device control unit 410 is suspended or is to be suspended by the respective control unit. In detail, when the port is assigned to host, the port suspension field bit may be set when the respective controller has suspended the port. When the port is assigned to device, the bit is set when the controller has detected a suspend condition on the bus.

Another register field of the OTG status register 600 may hold port connection information indicating an assignment of a port to the host control unit 415 or the device control unit 410. In detail, when the port is connected to host, the register field bit is set when the appropriate host controller has established a connection to a remote device. When the port is assigned to device, the bit is set when the device controller has been contacted by the remote host.

It is to be noted that the port connection register field bit may be overwritten at any time by software although this register field is accessible by software in a read-only mode only. To achieve this, the OTG control register 605 may include a (writeable) register field that may be used to overwrite the value of the OTG status register's port connection register field. This will be described in more detail below.

There may be further register fields in the OTG status register 600 to provide status information on the OTG control unit 425. For instance, there may be a register field storing FS (Full Speed) output enable information, port speed information, and information indicating whether certain line states or voltage values are reached.

Turning now to the OTG control register 605 which holds control information to control operation of the OTG control unit 425, there may again be a number of register fields. For instance, there may be a register field holding timer count condition information and control information with respect to whether a port is powered or not, is assigned to certain control units or not, or whether automatic or software controlled pull-up is enabled.

It is to be noted that some of the register fields of the OTG control register 605 may be aliased from the port multiplex controller registers discussed above, to facilitate both hardware implementation and software control of the OTG technology.

The OTG control register 605 may further comprise a write port connect register field holding write port connect information. As mentioned above, this register field may be used to override the value of the port connection register field of the OTG status register 600. That is, software may write to the write port connect register field of the OTG control register 605 to assert or deassert the respective register field bit of the OTG status register 600. This functionality allows for changing a port connection status by software even though the OTG status register 600 is a read-only register.

In detail, when the port is assigned to host, then the port connect status register field bit is asserted if the port is owned by the EHC and has been successfully brought into HS (High Speed) operational. Further, when the port is assigned to host, the status register bit may be asserted when the OHC has just detected a connect. When the port is assigned to device, the status register bit is asserted when downstream activity on the USB is detected. When the port is assigned to host, the status register bit is deasserted when the port is owned by the EHC and has just detected a disconnect. Similarly, the bit is deasserted when the OHC has just detected a disconnect. Moreover, the bit may be deasserted whenever the voltage of the USB port falls below 0.8V. All of this functionality may be achieved under hardware control.

The write port connect register field of the OTG control register 605 adds the possibility to change the status bit by software. This adds flexibility to the OTG control functionality, allowing software drivers to force the USB control device to enter specific port connection states.

Referring back to FIG. 6, the OTG control unit 425 may comprise further registers. For instance, the OTG timer register 610 may have register fields to store the current timer value (which may be read-only), and a preload value to preload for countdown. The OTG interrupt register 615 may store information in its fields indicating whether the timer has expired, port suspension or connections have changed, receiver activity is detected, the port speed has changed, an HS disconnect is detected, a line state has changed, or certain voltage ranges are fulfilled. The OTG interrupt enable register 620 may store interrupt enable information in its register fields indicating whether a global interrupt mechanism is enabled, or whether interrupts for which information is stored in the OTG interrupt register 615 are enabled.

Given the above description of the various embodiments, a USB controller device technique is provided to implement OTG functionality in a manner that allows for controlling the components by a state machine that can be implemented partly in hardware and partly in software. For this purpose, control and status information may be passed between the processor running the software driver, and the OTG control unit 425 through OTG

status and control registers 600, 605 (and optionally other OTG registers). The control register bits control internal and external switches, and the status register bits indicate the state of relevant signals from the OTG control unit 425.

In an embodiment, the OTG control register 605 has two addresses. Writing to one address sets selected bits, while writing to the other address clears the selected bits. This scheme provides for maximum compatibility with other components of the USB controller device.

Further, register aliasing may be used to achieve as much flexibility as possible with respect to system configurations that may be chosen. For instance, a configuration may be chosen having a static port assignment, e.g. to provide nothing more than device functionality. In this case, the BIOS (Basic Input Output System) sets the port multiplexer accordingly at the time the computer system boots. The register contents is then never changed as long as the system is up. If, however, another configuration is chosen where the OTG functionality is used the USB port acts as device port or as host port depending on the user's requests, and this may change at any time. The port multiplexer is then served by an OTG driver. Thus, register aliasing allows for having BIOS accesses and OTG driver accesses separated also at the register level. This makes it possible for the BIOS to act without any knowledge on the OTG driver.

In an embodiment, the USB controller device may support the following USB configurations: (i) at least one host port and one peripheral device port, (ii) two or more host ports, and (iii) one dual-role device port and at least one additional host port. The dual-role device port may also be used as an SRP capable peripheral device port.

When configuring the USB controller device of the present embodiment to have one dedicated USB host port in addition to one dual-role device port, the software driver may maintain an OTG state machine that is separate from the host controller state machine.

As mentioned above, embodiments are provided that implement the OTG state machine(s) partly in hardware and partly in software. This may be achieved by implementing some of the states in hardware and other states in software. For instance, taking the example of a dual-role device state machine, the software driver may implement all of the states except for the a_suspend, a_peripheral, and b_wait_acon states. These three states may be then implemented in hardware. Further, the OTG state machine(s) may be implemented partly in hardware and partly in software by implementing some but not all of the state transitions specified by the OTG state machine(s) in software.

Referring now to FIG. 7, a flow chart is provided illustrating how a software controlled state transition may take place. In step 700, the OTG control unit 425 detects an event that may require the software state machine to change its state. The OTG control unit 425 then saves the respective status information in the OTG status register 600 (step 705) and issues an interrupt to the processor (step 710). In response to the interrupt, the USB software driver reads the status register in step 715, makes the appropriate state change based on the current state and the status register contents (step 720), and writes to the OTG control register 605 to change the switch settings, i.e., to make the appropriate changes to the OTG hardware (step 725).

While the invention has been described with respect to the physical embodiments constructed in accordance therewith, it will be apparent to those skilled in the art that various modifications, variations and improvements of the present invention may be made in the light of the above teachings and within the purview of the appended claims without departing from the spirit and intended scope of the invention. In addition, those areas in which it is believed that those of ordinary skill in the art are familiar, have not been described herein in

order to not unnecessarily obscure the invention described herein. Accordingly, it is to be understood that the invention is not to be limited by the specific illustrative embodiments, but only by the scope of the appended claims.

5 INDUSTRIAL APPLICABILITY

This invention may generally be applicable to USB (Universal Serial Bus) controller devices and methods.

CLAIMS

1. A USB (Universal Serial Bus) controller device having OTG (On-The-Go) functionality, the USB controller device comprising:
 - 5 a host control unit having an EHCI (Enhanced Host Controller Interface) compliant host controller adapted to implement host functionality to perform data transfer to and/or from at least one USB peripheral device; and
 - an OTG control unit adapted to implement an OTG state machine to control operation of said host control unit,
 - 10 wherein said OTG control unit is adapted to implement said OTG state machine partly in hardware and partly in software.
2. The USB controller device of claim 1, wherein said OTG control unit comprises an OTG control register holding control information to control operation of said OTG control unit, said OTG control register being accessible by software.
- 15 3. The USB controller device of claim 2, wherein said OTG control register is writeable by software.
4. The USB controller device of claim 2, wherein said OTG control unit is adapted to change the state of hardware switches in response to changes of said control information.
- 20 5. The USB controller device of claim 2, wherein said OTG control register is addressable by writing to any one of two addresses, wherein writing to a first one of said two addresses setting control information while writing to a second one of said two addresses clearing the control information.
- 25 6. A USB (Universal Serial Bus) controller device having OTG (On-The-Go) functionality, the USB controller device comprising:
 - a host control unit having an EHCI (Enhanced Host Controller Interface) compliant host controller adapted to implement host functionality to perform data transfer to and/or from at least one
 - 30 USB peripheral device; and
 - an OTG control unit adapted to control operation of said host control unit,
 - wherein said OTG control unit comprises:
 - an OTG control register holding control information to control operation of said OTG control unit; and
 - an OTG status register holding status information of said OTG control unit,
- 35 7. The USB controller device of claim 6, wherein said OTG control register comprises a register field holding timer count condition information.
8. The USB controller device of claim 6, wherein said OTG control register comprises a register field holding write port connect information writeable by software to change the OTG port connect status.
- 40

9. ~~9. The USB controller device of claim 8, wherein said OTG port connect status is under hardware control as long as no software writes to said write port connect information register field.~~

10. A method of providing OTG (On-The-Go) functionality in a USB (Universal Serial Bus) controller device having an OTG control register and an OTG status register both being accessible by a software driver, the method comprising:

said software driver reading said OTG status register in response to receiving an interrupt from said USB controller device; and

said software driver writing to said OTG control register to force said USB controller device to change its OTG state.

1/4

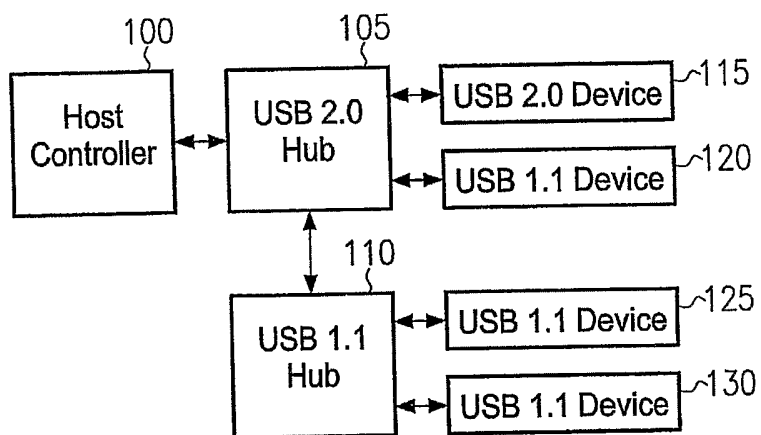


Fig. 1
(Prior Art)

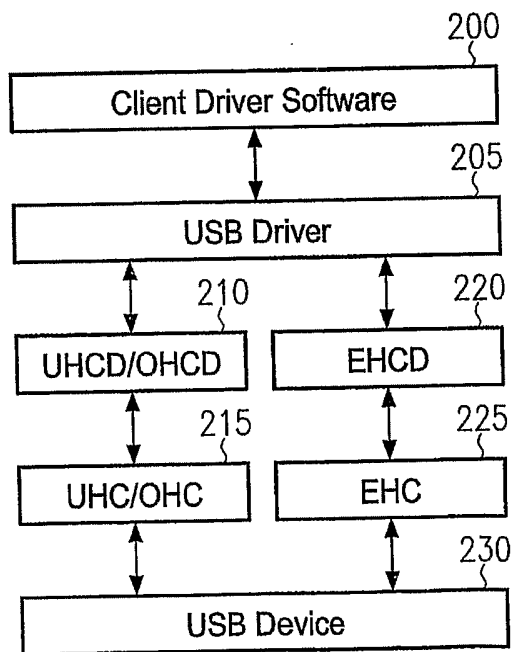


Fig. 2
(Prior Art)

2/4

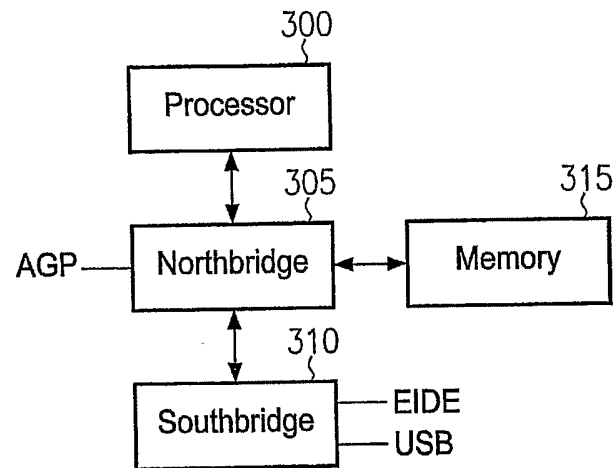


Fig. 3
(Prior Art)

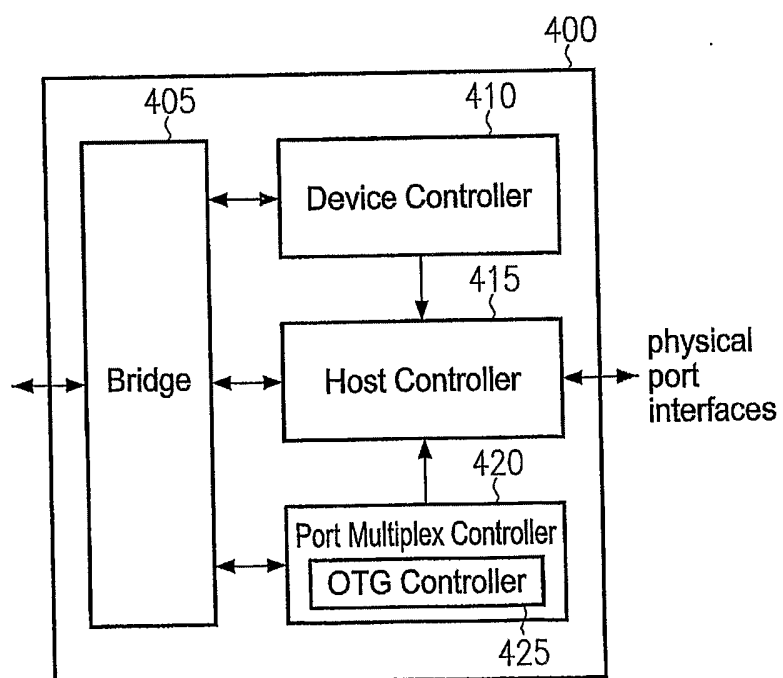


Fig. 4

3/4

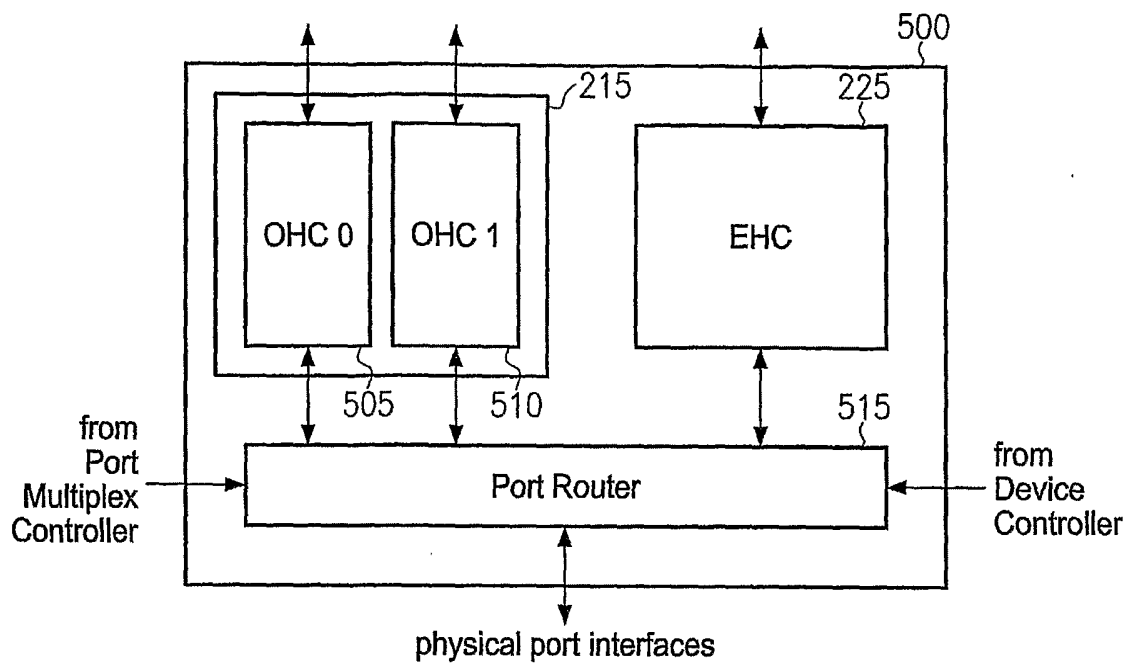


Fig. 5

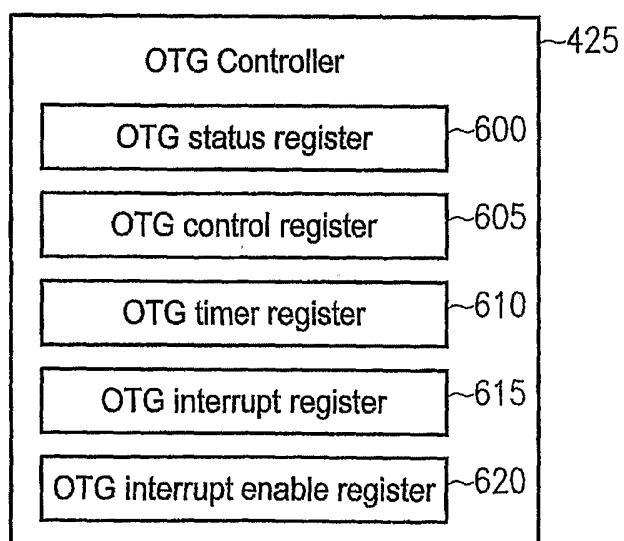


Fig. 6

4/4

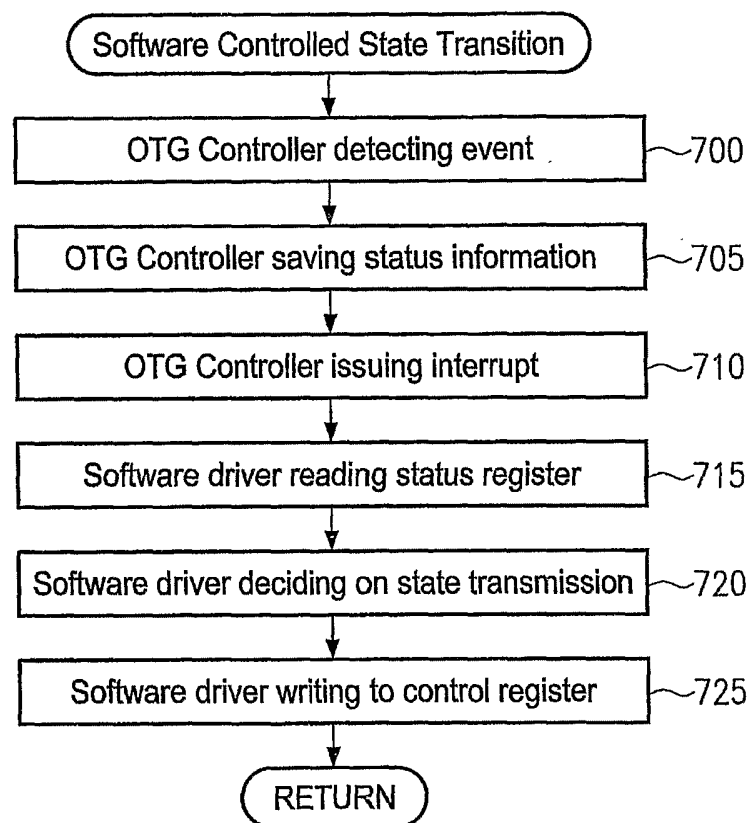


Fig. 7