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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 July 2013 (25.07.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/109451 A1

(51) International Patent Classification:
G06F 9/06 (2006.01) *G06F 9/30* (2006.01)
G06F 9/44 (2006.01) *G06T 15/00* (2006.01)

(21) International Application Number:
PCT/US2013/020916

(22) International Filing Date:
10 January 2013 (10.01.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/352,121 17 January 2012 (17.01.2012) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **MALAKAPALLI, Meher Prasad**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **ZHANG, Hao**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **TAN, Lin**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US).

(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, BN, 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, PA, 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, 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).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

(54) Title: PARA-VIRTUALIZED HIGH-PERFORMANCE COMPUTING AND GDI ACCELERATION

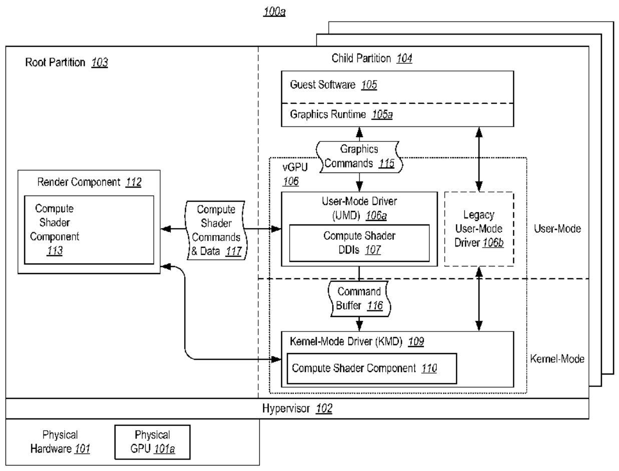


Figure 1A

(57) **Abstract**: The present invention extends to methods, systems, and computer program products for para-virtualized GPGPU computation and GDI acceleration. Some embodiments provide a compute shader to a guest application within a para-virtualized environment. A vGPU in a child partition presents compute shader DDIs for performing GPGPU computations to a guest application. A render component in a root partition receives compute shader commands from the vGPU and schedules the commands for execution at the physical GPU. Other embodiments provide GPU-accelerated GDI rendering capabilities to a guest application within a para-virtualized environment. A vGPU in a child partition provides an API for receiving GDI commands, and sends GDI commands and data to a render component in a root partition. The render component schedules the GDI commands on a 3D rendering device. The 3D rendering device executes the GDI commands at the physical GPU using a sharable GDI surface.

PARA-VIRTUALIZED HIGH-PERFORMANCE COMPUTING AND GDI ACCELERATION

BACKGROUND

1. Background and Relevant Art

[0001] Computer systems and related technology affect many aspects of society. Indeed, the computer system's ability to process information has transformed the way we live and work. Computer systems now commonly perform a host of tasks (e.g., word processing, scheduling, accounting, etc.) that prior to the advent of the computer system were performed manually. More recently, computer systems have been coupled to one another and to other electronic devices to form both wired and wireless computer networks over which the computer systems and other electronic devices can transfer electronic data. Accordingly, the performance of many computing tasks is distributed across a number of different computer systems and/or a number of different computing environments.

[0002] Some computer systems are configured to provide para-virtualized execution environments, which allow guest software to share hardware devices of a single computer system in an isolated manner. Generally, para-virtualized execution environments provide a plurality of partitions, supported by a hypervisor. The partitions provide isolation between different guest software. The partitions generally include a root partition and one or more child partitions. The root partition runs a host operating system and manages a virtualization stack. The root partition may gain access to physical devices. Each child partition hosts guest software (e.g., guest operating systems and guest applications). Child partitions are provided access to physical devices through virtual devices and software interfaces of the hypervisor.

[0003] Some para-virtualized execution environments provide child partitions (and guest software executing therein) with para-virtualized access to one or more physical graphics processing units ("GPUs"). Generally, each implementation of para-virtualized access to physical GPUs supports particular three-dimensional rendering framework(s). As such, guest software may be unable to access capabilities of a physical GPU if that guest software is executing within a para-virtualized execution environment that does not support those capabilities. In some cases, the guest software may rely on using a virtualized CPU to perform tasks not supported para-virtualized access to a physical GPU, incurring a potentially significant performance penalty.

BRIEF SUMMARY

[0004] The present invention extends to methods, systems, and computer program products for providing high-performance computing and graphics device interface ("GDI") acceleration in a para-virtualized environment.

[0005] Some embodiments include a method for providing graphics processing unit ("GPU") accelerated computing functionality to a guest application executing in a child partition of a para-virtualized execution environment. A virtual machine session is instantiated. A hypervisor in the virtual machine session provides (i) a root partition (which has access to a physical GPU), and (ii) the child partition (which executes the guest application).

[0006] A virtualized graphics processing unit ("vGPU") executing within the child partition is presented to the guest application. A user-mode driver ("UMD") of the vGPU presents compute shader device driver interfaces ("DDIs") to the guest application. The compute shader DDIs provide an application programming interface ("API") that enables the guest application to send compute shader commands to the vGPU. The compute shader commands are used to perform general-purpose graphics processing unit ("GPGPU") computations at the physical GPU using a compute shader. A render component executing within the root partition receives a physical GPU-specific compute shader command from the vGPU, and schedules the command for execution at the physical GPU.

[0007] Additional embodiments include a method for providing GPU-accelerated GDI functionality to a guest application executing in a child partition of a para-virtualized execution environment. A virtual machine session is instantiated. A hypervisor in the virtual machine session provides (i) a root partition having access to a physical GPU, and (ii) the child partition which executes the guest application.

[0008] A vGPU, which executes within the child partition, is presented to the guest application. An API of a kernel-mode driver ("KMD") of the vGPU enables the guest operating system to accelerate GDI rendering commands submitted by a guest application. These commands are then processed by the KMD of the vGPU.

[0009] A render component executing within the root partition receives a GDI acceleration rendering command from the vGPU. In response, the render component schedules the GDI acceleration rendering command on a GDI composition device within the root partition. The GDI composition device is configured to execute the GDI acceleration rendering command at the physical GPU. The GDI composition device

marks a GDI surface corresponding for the GDI command as sharable for composition by the desktop.

[0010] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

[0011] Additional features and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by the practice of the invention. The features and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims. These and other features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

[0013] Figure 1A illustrates an example computer architecture that enables para-virtualized access to compute shader functionality of physical graphics processing unit ("GPU") hardware.

[0014] Figure 1B illustrates an example computer architecture that enables para-virtualized graphics device interface ("GDI") acceleration by physical GPU hardware.

[0015] Figure 1C illustrates an example computer architecture that enables para-virtualized access to compute shader functionality and para-virtualized GDI acceleration by physical GPU hardware.

[0016] Figure 2A illustrates a flow chart of an example method for providing GPU-accelerated computing functionality to a guest application executing in a child partition of a para-virtualized execution environment.

[0017] Figure 2B illustrates a flow chart of an example method for providing GPU-accelerated GDI functionality to a guest application executing in a child partition of a para-virtualized execution environment.

DETAILED DESCRIPTION

[0018] The present invention extends to methods, systems, and computer program products for providing high-performance computing and graphics device interface ("GDI") acceleration in a para-virtualized environment.

[0019] Some embodiments include a method for providing graphics processing unit ("GPU") accelerated computing functionality to a guest application executing in a child partition of a para-virtualized execution environment. A virtual machine session is instantiated. A hypervisor in the virtual machine session provides (i) a root partition (which has access to a physical GPU), and (ii) the child partition (which executes the guest application).

[0020] A virtualized graphics processing unit ("vGPU") executing within the child partition is presented to the guest application. A user-mode driver ("UMD") of the vGPU presents compute shader device driver interfaces ("DDIs") to the guest application. The compute shader DDIs provide an application programming interface ("API") that enables the guest application to send compute shader commands to the vGPU. The compute shader commands are used to perform general-purpose graphics processing unit ("GPGPU") computations at the physical GPU using a compute shader. A render component executing within the root partition receives a physical GPU-specific compute shader command from the vGPU, and schedules the command for execution at the physical GPU.

[0021] Additional embodiments include a method for providing GPU-accelerated GDI functionality to a guest application executing in a child partition of a para-virtualized execution environment. A virtual machine session is instantiated. A hypervisor in the virtual machine session provides (i) a root partition having access to a physical GPU, and (ii) the child partition which executes the guest application.

[0022] A vGPU, which executes within the child partition, is presented to the guest application. An API of a kernel-mode driver ("KMD") of the vGPU enables the guest operating system to accelerate GDI rendering commands submitted by a guest application. These commands are then processed by the KMD of the vGPU.

[0023] A render component executing within the root partition receives a GDI acceleration rendering command from the vGPU. In response, the render component schedules the GDI acceleration rendering command on a GDI composition device within the root partition. The GDI composition device is configured to execute the GDI acceleration rendering command at the physical GPU. The GDI composition device marks a GDI surface corresponding for the GDI command as sharable for composition by the desktop.

[0024] Embodiments of the present invention may comprise or utilize a special purpose or general-purpose computer including computer hardware, such as, for example, one or more processors and system memory, as discussed in greater detail below. Embodiments within the scope of the present invention also include physical and other computer-readable media for carrying or storing computer-executable instructions and/or data structures. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer system. Computer-readable media that store computer-executable instructions are computer storage media (devices). Computer-readable media that carry computer-executable instructions are transmission media. Thus, by way of example, and not limitation, embodiments of the invention can comprise at least two distinctly different kinds of computer-readable media: computer storage media (devices) and transmission media.

[0025] Computer storage media (devices) includes RAM, ROM, EEPROM, CD-ROM, solid state drives ("SSDs") (e.g., based on RAM), Flash memory, phase-change memory ("PCM"), other types of memory, other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer.

[0026] A "network" is defined as one or more data links that enable the transport of electronic data between computer systems and/or modules and/or other electronic devices. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a transmission medium. Transmissions media can include a network and/or data links which can be used to carry desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer. Combinations of the above should also be included within the scope of computer-readable media.

[0027] Further, upon reaching various computer system components, program code means in the form of computer-executable instructions or data structures can be transferred automatically from transmission media to computer storage media (devices) (or vice versa). For example, computer-executable instructions or data structures received over a network or data link can be buffered in RAM within a network interface module (e.g., a "NIC"), and then eventually transferred to computer system RAM and/or to less volatile computer storage media (devices) at a computer system. Thus, it should be understood that computer storage media (devices) can be included in computer system components that also (or even primarily) utilize transmission media.

[0028] Computer-executable instructions comprise, for example, instructions and data which, when executed at a processor, cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, or even source code. Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the described features or acts described above. Rather, the described features and acts are disclosed as example forms of implementing the claims.

[0029] Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including, personal computers, desktop computers, laptop computers, message processors, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, mobile telephones, PDAs, tablets, pagers, routers, switches, and the like. The invention may also be practiced in distributed system environments where local and remote computer systems, which are linked (either by hardwired data links, wireless data links, or by a combination of hardwired and wireless data links) through a network, both perform tasks. In a distributed system environment, program modules may be located in both local and remote memory storage devices.

Para-Virtualized Compute Shader (GPGPU) Functionality

[0030] Figure 1A illustrates an example computer architecture 100a that enables para-virtualized access to compute shader functionality of physical GPU hardware. Referring to Figure 1A, computer architecture 100a includes physical hardware 101. Physical hardware 101 can include any appropriate hardware devices, such as one or more general purpose processors, system memory, and the like. As depicted, physical hardware includes at least one physical GPU 101a.

[0031] Physical GPU 101a is a processing device configured to perform parallel processing tasks, such as graphics rendering. Physical GPU 101a includes support for executing a compute shader, which enables physical GPU 101a to perform general-purpose (i.e., non-graphics rendering) calculations. In other words, physical GPU 101a supports GPGPU computation on a compute shader device.

[0032] Computer architecture 100a also includes hypervisor 102. Hypervisor 102 executes on top of physical hardware 101 and supports a virtualization platform. The virtualization platform provides a plurality of partitions. Each partition provides a logical unit of isolation, in which guest software can be executed. For example, computer architecture 100a includes root partition 103 and child partition 104.

[0033] Root partition 103 executes a host operating system, and has direct access to physical hardware 101 (as depicted by root partition 103 appearing over physical hardware 101). Each child partition provides an execution environment for executing guest software (e.g., operating systems and/or applications) and may access physical hardware 101 indirectly in a para-virtualized manner. That is, through hypervisor 102, each child partition provides one or more software interfaces (e.g., virtualized hardware) to guest software. The guest software, in turn, uses the software interface(s) to access physical hardware 101. Hypervisor 102 can provide support for a plurality of child partitions.

[0034] As depicted, guest software 105 executes within child partition 104. Guest software 105 comprises any appropriate guest software, such as an operating system and/or an application program executing within an operating system. Guest software 105 includes or uses graphics runtime 105a. Graphics runtime 105a provides a framework (e.g., APIs) for rendering graphics and/or performing GPGPU computation.

[0035] Child partition 104 provides guest software 105 access to vGPU 106. vGPU 105 virtualizes physical GPU 101a, enabling guest software 105 to indirectly access physical GPU 101a. As such, vGPU 106 is configured to expose all, or a subset, of the functionality of at least one rendering framework (corresponding to graphics runtime 105a) to guest software 105, along with any corresponding functionality of physical GPU 101a.

[0036] In particular, vGPU 106 is configured to expose one or more software interfaces that enable guest software 105 to call vGPU 106 to access compute shader functionality of physical GPU 101a for performing GPGPU computation at physical GPU 101a. vGPU 106, in turn, works in conjunction with a render component in root partition 103 to perform compute shader functionality on physical GPU 101a. As depicted, for example, root partition 103 includes render component 112. Render component 112, in turn, includes compute shader component 113 for handing compute shader commands and data. vGPU 106 remotes compute shader commands and data 117 to render component 112 to perform the rendering on physical GPU 101a.

[0037] Render component 112 schedules any graphics commands received from vGPU 106 for execution on physical GPU 101a. Render component 112 also creates proper context for executing those commands. As such, render component 112 is configured to use compute shader component 113 to schedule execution of compute shader-related commands that are received from vGPU 106 in child partition 104 on physical GPU 101a.

[0038] As depicted, vGPU 106 includes user-mode driver 106a executing in a user-mode of child partition 104 and kernel-mode driver 109 executing in a kernel-mode of child partition 104. User-mode driver 106a exposes device driver interfaces ("DDIs") of at least one rendering framework, including DDIs related to compute shader (GPGPU) functionality, depicted as compute shader DDIs 107. Compute shader DDIs 107 enable guest software 105 to make calls to vGPU 106 for performance of GPGPU computations.

[0039] In some embodiments, user-mode driver 106b exposes DDIs of a rendering framework that supports compute shader functionality (e.g., DirectX® versions 10 and/or 11 from Microsoft® Corporation). For example, in embodiments when user-mode driver 106a exposes compute shader functionality of DirectX® versions 10 and 11, user-mode driver 106a may expose the one or more of the following DDIs as part of compute shader DDIs 107:

```
PFND3D11DDI_SETSHADER_WITH_IFACES         pfnCsSetShaderWithIfaces
PFND3D10DDI_SETSHADER                     pfnCsSetShader
PFND3D10DDI_SETSHADERRESOURCES            pfnCsSetShaderResources
PFND3D10DDI_SETSAMPLERS                   pfnCsSetSamplers
PFND3D10DDI_SETCONSTANTBUFFERS            pfnCsSetConstantBuffers
PFND3D11DDI_SETUNORDEREDACCESSVIEWS       pfnCsSetUnorderedAccessViews
```

In some embodiments, user-mode driver 106b presents all DDIs of a rendering framework that supports compute shader functionality (e.g., DirectX® version 11), such as DDIs related to one or more of a domain shader, a hull shader, or a geometry shader. However, user-mode driver 106b may expose any set of DDIs from any rendering framework supporting compute shader functionality.

[0040] In some embodiments, vGPU 106 may also include legacy user-mode driver 106b executing in user-mode of child partition 104. Legacy user-mode driver 106b may expose DDIs of a legacy version of one or more rendering frameworks. For example, legacy user-mode driver 106b may support a legacy version of DirectX® (e.g., DirectX® version 9), or a legacy version any other rendering framework (e.g., OpenGL® from Silicon Graphics, Inc.).

[0041] Generally, user-mode driver 106a is configured to construct hardware contexts and command buffers. In particular, user-mode driver 106a converts graphic commands issued by guest software 105 (or graphics runtime 105a of guest software 105) into hardware-specific commands. For example, user-mode driver 106a may receive graphics commands 115 relating to GPGPU computations using a compute shader from guest software 105. User-mode driver 106a is configured to convert graphics commands 115 into hardware-specific commands (i.e., commands that are specific to physical GPU 101a). As part of the conversion, user-mode driver 106a maintains proper hardware context for physical GPU 101a. For example, user-mode driver 106a translates logical values for settings affecting a graphics pipeline into values and corresponding physical settings. User-mode driver 106a is also configured to store converted hardware-specific commands in command buffer 116 and send command buffer 115 to kernel-mode driver 109.

[0042] In addition, vGPU 106 includes kernel-mode driver 109 executing in kernel-mode of child partition 104, which includes compute shader component 110. Kernel-mode driver 109 is configured to receive command buffers (e.g., command buffer 116) and to construct corresponding direct memory access ("DMA") buffers. When it is time for a DMA buffer to be processed, a GPU scheduler calls kernel-mode driver 109. Kernel-mode driver 109 then handles the specifics of actually submitting the DMA buffer to physical GPU 101a.

[0043] Figure 2A illustrates a flow chart of an example method 200a for providing GPU-accelerated computing functionality to a guest application executing in a child partition of a para-virtualized execution environment. Method 200a will be described with respect to the components and data of computer architecture 100a.

[0044] Method 200a includes an act of instantiating a virtual machine session, including instantiating a hypervisor that provides (i) a root partition having access to the physical GPU, and (ii) the child partition which executes the guest application (act 201). For example, computing environment 100a can instantiate hypervisor 102 as part of instantiating a virtual machine session. Hypervisor 102 provides root partition 103 and child partition 104. Root partition 103 has access to physical GPU 101a. Child partition 104 executes one or more guest applications, including guest application 105, and has indirect access to physical GPU 101a.

[0045] Method 200a also includes an act of presenting a vGPU to the guest application, the vGPU executing within the child partition, including presenting a plurality of compute shader DDIs to the guest application as part of a UMD of the vGPU, the plurality of compute shader DDIs providing an API that enables the guest application to send compute shader commands to the vGPU for performing GPGPU computations at the physical GPU using a compute shader (act 202). For example, child partition 104 can present vGPU 106 to guest software 105. vGPU 106 includes user-mode driver 106a. User-mode driver 106a presents compute shader DDIs 107, which enable guest software 105 to call vGPU 106 to perform compute shader GPGPU calculations at physical GPU 101a. For example, guest software 105 (or graphics runtime 105a) can send graphics commands 115 to vGPU 106 for performance of GPGPU calculations at physical GPU 101a.

[0046] User-mode driver 106a converts received graphics commands 115 to physical-hardware specific commands (e.g., as part of command buffer 116). vGPU 106 then remotes compute shader commands and data 117 (including physical-hardware specific commands) to render component 112.

[0047] Method 200a also includes an act of a render component executing within the root partition receiving a physical GPU-specific compute shader command from the vGPU (act 203). For example, render component 112, which executes in root partition 103, can receive compute shader commands and data 117, including a physical GPU-specific compute shader command, from vGPU 106.

[0048] Method 200a also includes an act of the render component scheduling the physical GPU-specific compute shader command for execution at the physical GPU (act 204). For example, compute shader component 113 can schedule the received physical GPU-specific compute shader command for execution on a compute shader device at physical GPU 101a. In doing so, compute shader component 113 can configure and maintain appropriate context for execution of the physical GPU-specific compute shader command.

Para-Virtualized GDI Acceleration

[0049] In addition to providing guest software para-virtualized compute shader access to physical GPU 101a (i.e., GPGPU computational ability), embodiments extend to providing guest software para-virtualized GDI acceleration at a physical GPU. Figure 1B illustrates an alternate computer architecture 100b that enables para-virtualized GDI acceleration by physical GPU hardware. Thus, computer architecture 100b enables guest software to request GDI command acceleration using a physical GPU, as opposed to handling GDI commands with a physical or virtual central processing unit.

[0050] Within child partition 104', computer architecture 100b provides components that enable accelerated rendering of GDI commands on physical GPU 101a' when those commands are issued by guest software 105'. As depicted, for example, child partition 104' includes GDI interface 108, which is configured to communicate with graphics runtime 105a' (e.g., a graphics runtime that is part of a guest operating system). Furthermore, vGPU 106' is configured to process GDI acceleration commands 120 received from GDI interface 108 at GDI component 111 of kernel-mode driver 109'.

[0051] GDI interface 108 is configured to expose one or more GDI command interfaces to graphics runtime 105a', and to forward GDI accelerated commands received from graphics runtime 105a' to kernel-mode driver 109' of vGPU'. Thus, at the request of guest software 105', graphics runtime 105a' is enabled to send GDI accelerated graphics commands 120 to kernel-mode driver 109' of vGPU 106' via GDI interface 108 to accelerate GDI command execution at physical GPU 101a'.

[0052] GDI component 111 of kernel-mode driver 109' is configured to implement one or more GDI command interfaces, along with corresponding hardware rendering operation(s). In some embodiments, vGPU 106' implements and exposes GDI commands corresponding to a particular version of DirectX® (e.g., DirectX® versions 10 and/or 11). For example, GDI component 111 may expose and implement a 'DxgkDdiRenderKm' interface, along with corresponding hardware rendering operations. While GDI component 111 may implement any appropriate GDI operations, in some embodiments GDI component 111 implements the following operations:

```
typdef struct _DXGK_RENDERKM_COMMAND
{
    DXGK_RENDERKM_OPERATION   Opcode;
    UINT   CommandSize;
    union
```

```
        {
            DXGK_GDIARG_BITBLT             BitBlt;
            DXGK_GDIARG_COLORFILL          ColorFill;
            DXGK_GDIARG_ALPHABLEND         Alpha Bend;
            DXGK_GDIARG_STRETCHBLT         StretchBlt;
            DXGK_GDIARG_TRANSPARENTBLT     TransparentBlt;
            DXGK_GDIARG_CLEARTYPEBLEND     ClearTypeBlend;
        } Command;
    } DXGK_RENDERKM_COMMAND;
```

[0053] Kernel-mode driver 109' is configured to send GDI commands and data 118 to render component 112' within root partition 103'. GDI commands and data 118 include information relating to 3D rendering devices and contexts for executing GDI acceleration commands. Render component 112' includes GDI component 114, which is configured to receive GDI commands and data 118 and to execute received GDI acceleration commands at physical GPU 101a'.

[0054] In particular, GDI component 114 is configured to create a GDI surface and a corresponding 3D rendering (or composition) device (e.g., a D3D device). The 3D rendering device is configured to provide appropriate context for executing GDI accelerated commands. The GDI surface can comprise any appropriate GDI surface, such as any of the following GDI surface types:

D3DKMDT_GDISURFACE_TEXTURE
D3DKMDT_GDISURFACE_STAGING_CPUVISIBLE
D3DKMDT_GDISURFACE_STAGING
D3DKMDT_GDISURFACE_LOOKUPTABLE
D3DKMDT_GDISURFACE_EXISTINGSYSMEM
3DKMDT_GDISURFACE_TEXTURE_CPUVISIBLE

GDI component 114 marks any created GDI surface as a sharable surface. This enables composition to a desktop to succeed.

[0055] Figure 2B illustrates a flow chart of an example method 200b for providing GPU-accelerated GDI functionality to a guest application executing in a child partition of a para-virtualized execution environment. Method 200b will be described with respect to the components and data of computer architecture 100b.

[0056] Method 200b includes an act of instantiating a virtual machine session, including instantiating a hypervisor that provides (i) a root partition having access to the physical GPU, and (ii) the child partition which executes the guest application (act 205). For example, computing environment 100b can instantiate hypervisor 102’ as part of instantiating a virtual machine session. Hypervisor 102’ provides root partition 103’ and child partition 104’. Root partition 103’ has access to physical GPU 101a’. Child partition 104’ executes one or more guest applications, including guest application 105’, and has indirect access to physical GPU 101a’.

[0057] Method 200b also includes an act of presenting a vGPU to the guest application, the vGPU executing within the child partition, including presenting an API of a KMD of the vGPU that enables a guest operating system to accelerate GDI rendering commands used by the guest application to the vGPU for processing by the KMD of the vGPU (act 206). For example, child partition 104’ presents vGPU 106’ and GDI interface 108 to guest software 105’. GDI interface 108 enables graphics runtime 105a’ (e.g., a graphics runtime of an operating system) to accelerate GDI rendering commands used by guest software 105’ to vGPU 106’ for processing by kernel-mode driver 109’ and for acceleration by physical GPU 101a’. For example, graphics runtime 105a’ can send graphics commands 120 to kernel-mode driver 109’ through GDI interface 108 for performance of GDI acceleration commands. vGPU 106’ uses GDI component 111 of kernel-mode driver 109’ to process received GDI commands and to send GDI commands and data 118 to render component 112’ in root partition 103’.

[0058] Method 200b also includes an act of a render component executing within the root partition receiving a GDI acceleration rendering command from the vGPU (act 207). For example, render component 112’, which executes in root partition 103’, can receive GDI commands and data 118 from vGPU 106’.

[0059] Method 200b also includes an act of the render component scheduling the GDI acceleration rendering command on a GDI composition device within the root partition, the GDI composition device being configured to execute the at least one GDI acceleration rendering command at the physical GPU, the GDI composition device also being configured to mark a GDI surface corresponding to the at least one GDI acceleration rendering command as sharable for composition by a desktop (act 208). For example, GDI component 114 can create a 3D rendering (composition) device and a GDI surface for execution of GDI acceleration commands and can schedule the GDI acceleration commands for execution on the 3D rendering (composition) device. The GDI surface can be marked as sharable for composition at the desktop.

Para-Virtualized Compute Shader (GPGPU) Functionality and Para-Virtualized GDI Acceleration

[0060] In some embodiments, a single computer architecture can provide both compute shader (GPGPU) functionality and para-virtualized GDI acceleration. Figure 1C illustrates an example computer architecture 100c that enables para-virtualized access to compute shader functionality and para-virtualized GDI acceleration by physical GPU hardware. For example, computer architecture 100c includes GDI interface 108 and vGPU 106''. vGPU 106''contains user-mode driver 106a'', which includes compute shader DDIs 107 . vGPU 106'' also contains kernel-mode driver 109'' that includes both compute shader component 110a and GDI component 111.

[0061] vGPU 106'' (executing in child partition 104''), can communicate both compute shader commands and data and GDI commands and data to render component 112'' executing in root partition 103''. Render component 112'' includes both compute shader component 113 and GDI component 114 for executing compute shader and GDI commands at GPU 101a''. It will be appreciated that computer architecture 100c, by including both compute shader and GDI functionality, can provide increased functionality and greater compatibility with particular rendering frameworks.

[0062] The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

CLAIMS

1. At a computer system including one or more processors and system memory, the computer system also including a physical graphics processing unit ("GPU"), a method for providing GPU-accelerated computing functionality to a guest application executing in a child partition of a para-virtualized execution environment, the method comprising:

an act of instantiating a virtual machine session, including instantiating a hypervisor that provides (i) a root partition having access to the physical GPU, and (ii) the child partition which executes the guest application;

an act of presenting a virtualized graphics processing unit ("vGPU") to the guest application, the vGPU executing within the child partition, including presenting a plurality of compute shader device driver interfaces ("DDIs") to the guest application as part of a user-mode driver ("UMD") of the vGPU, the plurality of compute shader DDIs providing an application programming interface that enables the guest application to send compute shader commands to the vGPU for performing general-purpose graphics processing unit ("GPGPU") computations at the physical GPU using a compute shader; and

an act of a render component executing within the root partition receiving a physical GPU-specific compute shader command from the vGPU; and

an act of the render component scheduling the physical GPU-specific compute shader command for execution at the physical GPU.

2. The method as recited in claim 1, wherein the UMD converts any compute shader commands received from the guest application into corresponding physical GPU-specific compute shader commands and stores the corresponding physical GPU-specific compute shader commands in a command buffer.

3. The method as recited in claim 1, wherein the act of presenting a vGPU to the guest application comprises an act of presenting a vGPU that includes a kernel-mode driver ("KMD"), the KMD being configured to construct a direct memory access buffer from the command buffer.

4. The method as recited in claim 3, wherein the KMD is configured to implement support for graphics device interface ("GDI") acceleration rendering commands.

5. The method as recited in claim 4, wherein support for GDI acceleration rendering commands by the KMD enables the creation of a compute shader composition device.

6. The method as recited in claim 1, wherein the render component uses a compute shader composition device within the root partition to execute the physical GPU-specific compute shader command.

7. At a computer system including one or more processors and system memory, the computer system also including a physical graphics processing unit ("GPU"), a method for providing GPU-accelerated graphics device interface ("GDI") functionality to a guest application executing in a child partition of a para-virtualized execution environment, the method comprising:

an act of instantiating a virtual machine session, including instantiating a hypervisor that provides (i) a root partition having access to the physical GPU, and (ii) the child partition which executes the guest application;

an act of presenting a virtualized graphics processing unit ("vGPU") to the guest application, the vGPU executing within the child partition, including presenting an application programming interface of a kernel-mode driver ("KMD") of the vGPU that enables a guest operating system to accelerate graphics device interface ("GDI") rendering commands used by the guest application to the vGPU for processing by the KMD of the vGPU;

an act of a render component executing within the root partition receiving a GDI acceleration rendering command from the vGPU; and

an act of the render component scheduling the GDI acceleration rendering command on a GDI composition device within the root partition, the GDI composition device being configured to execute the GDI acceleration rendering command at the physical GPU, the GDI composition device also being configured to mark a GDI surface corresponding to the GDI acceleration rendering command as sharable for composition by a desktop.

8. The method as recited in claim 7, wherein a user-mode driver ("UMD") of the vGPU is configured to provide the guest application access to a plurality of device driver interfaces ("DDIs"), at least some of the plurality of DDIs being usable to provide the at least one application access to functionality of a compute shader at the physical GPU.

9. The method as recited in claim 8, wherein the GDI acceleration rendering commands are usable as part of use of the compute shader at the physical GPU.

10. The method as recited in claim 8, wherein the UMD executes in a user-mode of child partition.

11. The method as recited in claim 8, wherein the KMD executes in a kernel-mode of child partition.

12. The method as recited in claim 8, wherein the UMD converts graphics commands received from the guest application into corresponding hardware-specific compute shader

commands and stores the corresponding hardware-specific compute shader commands in a command buffer.

13. The method as recited in claim 8, wherein the KMD constructs a direct memory access buffer from the command buffer.

14. The method as recited in claim 8, wherein the UMD comprises a new UMD that is enabled to present all DDIs corresponding to a rendering framework that includes a compute shader, including a rendering framework that also includes one or more of a domain shader, a hull shader, or a geometry shader.

15. A computer program product for use at a computer system, the computer program product for implementing a method for providing GPU-accelerated computing functionality to a guest application executing in a child partition of a para-virtualized execution environment, the computer program product comprising one or more computer storage media having stored thereon computer-executable instructions that, when executed at a processor, cause the computer system to perform the method, including the following:

instantiate a virtual machine session, including instantiating a hypervisor that provides (i) a root partition having access to a physical GPU, and (ii) the child partition which executes the guest application;

present a virtualized graphics processing unit ("vGPU") to the guest application, the vGPU executing within the child partition, including:

presenting a plurality of compute shader device driver interfaces ("DDIs") to the guest application as part of a user-mode driver ("UMD") of the vGPU, the plurality of compute shader DDIs providing an application programming interface that enables the guest application to send compute shader commands to the vGPU for performing general-purpose graphics processing unit ("GPGPU") computations at the physical GPU using a compute shader; and

presenting an application programming interface of a kernel-mode driver ("KMD") of the vGPU that enables a guest operating system to accelerate graphics device interface ("GDI") rendering commands used by the guest application to the vGPU for processing by at least the KMD of the vGPU;

receive one or both of at least one physical GPU-specific compute shader command and/or at least one GDI acceleration rendering command from the vGPU at a render component executing within the root partition;

schedule a physical GPU-specific compute shader command for execution at the physical GPU; and schedule a GDI acceleration rendering command on a GDI composition device within the root partition, the GDI composition device being configured to execute the GDI acceleration rendering command at the physical GPU, the GDI composition device also being configured to mark a GDI surface corresponding to the GDI acceleration rendering command as sharable for composition by a desktop.

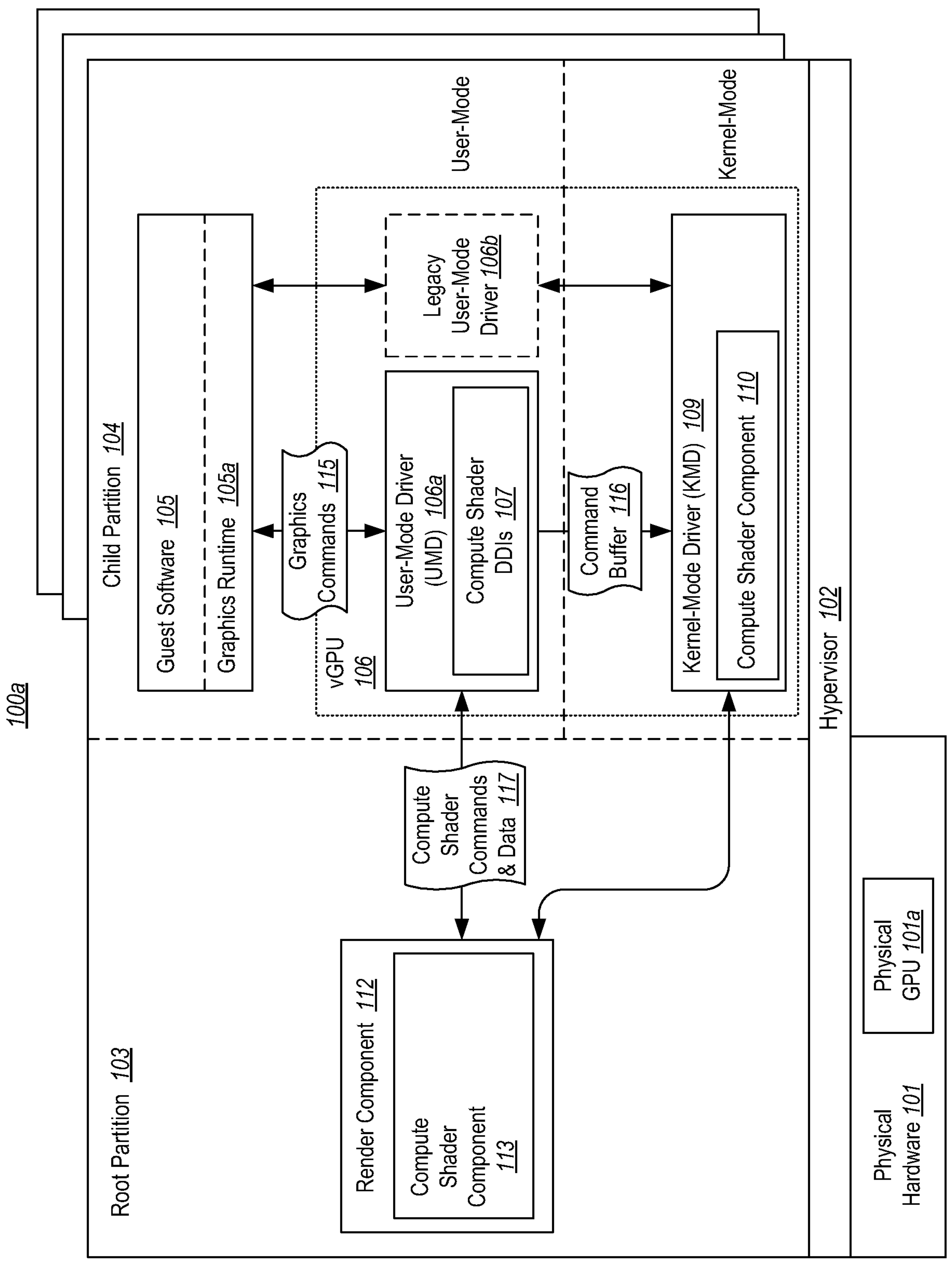

100a
Child Partition 104
Guest Software 105
Graphics Runtime 105a
Graphics Commands 115
vGPU 106
User-Mode Driver (UMD) 106a
Compute Shader DDIs 107
Legacy User-Mode Driver 106b
User-Mode
Kernel-Mode
Command Buffer 116
Kernel-Mode Driver (KMD) 109
Compute Shader Component 110
Hypervisor 102
Compute Shader Commands & Data 117
Root Partition 103
Render Component 112
Compute Shader Component 113
Physical Hardware 101
Physical GPU 101a

Figure 1A

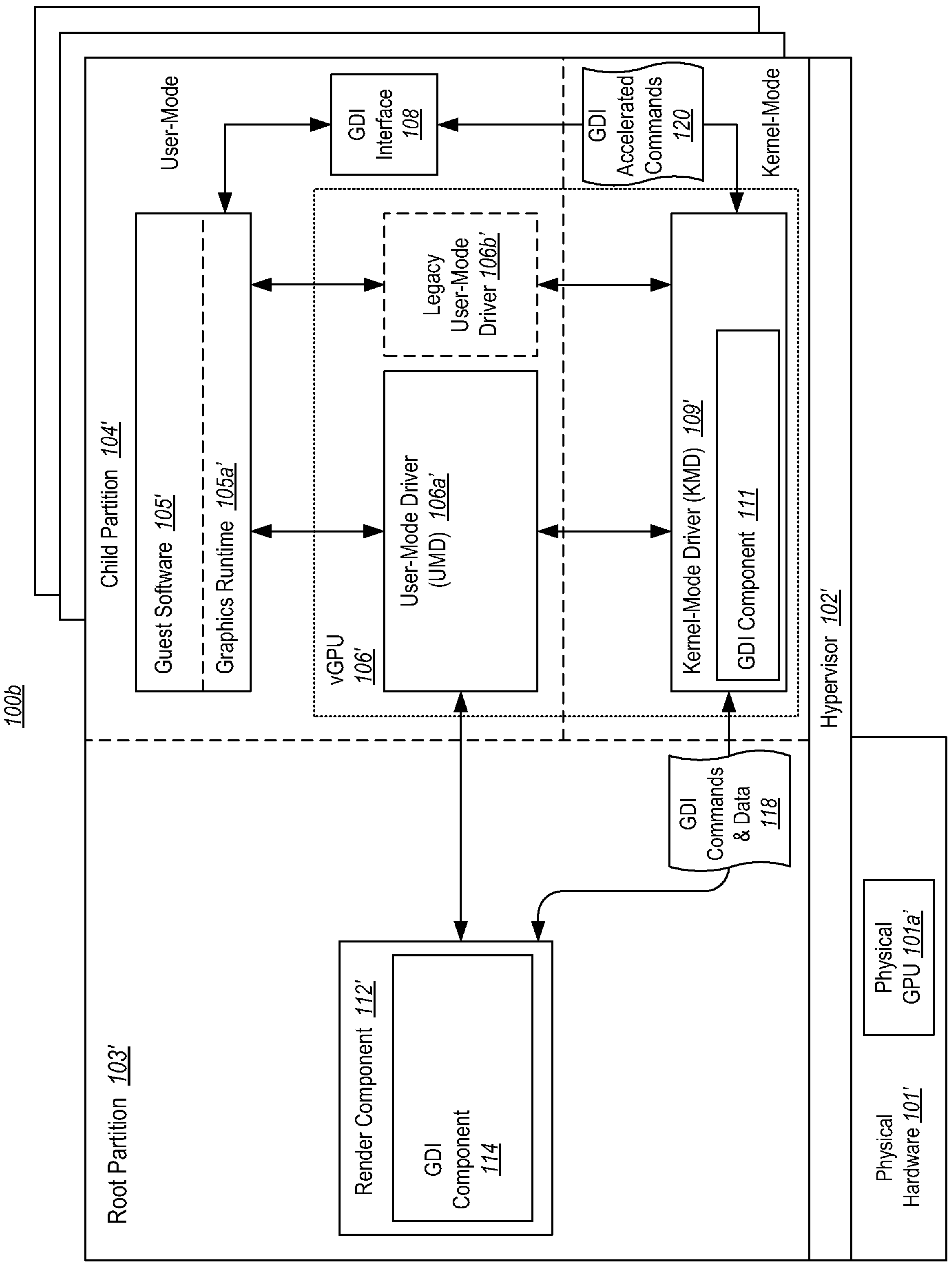

100b
Root Partition 103'
Render Component 112'
GDI Component 114
Child Partition 104'
User-Mode
Kernel-Mode
Guest Software 105'
Graphics Runtime 105a'
GDI Interface 108
GDI Accelerated Commands 120
vGPU 106'
User-Mode Driver (UMD) 106a'
Legacy User-Mode Driver 106b'
Kernel-Mode Driver (KMD) 109'
GDI Component 111
GDI Commands & Data 118
Hypervisor 102'
Physical Hardware 101'
Physical GPU 101a'

Figure 1B

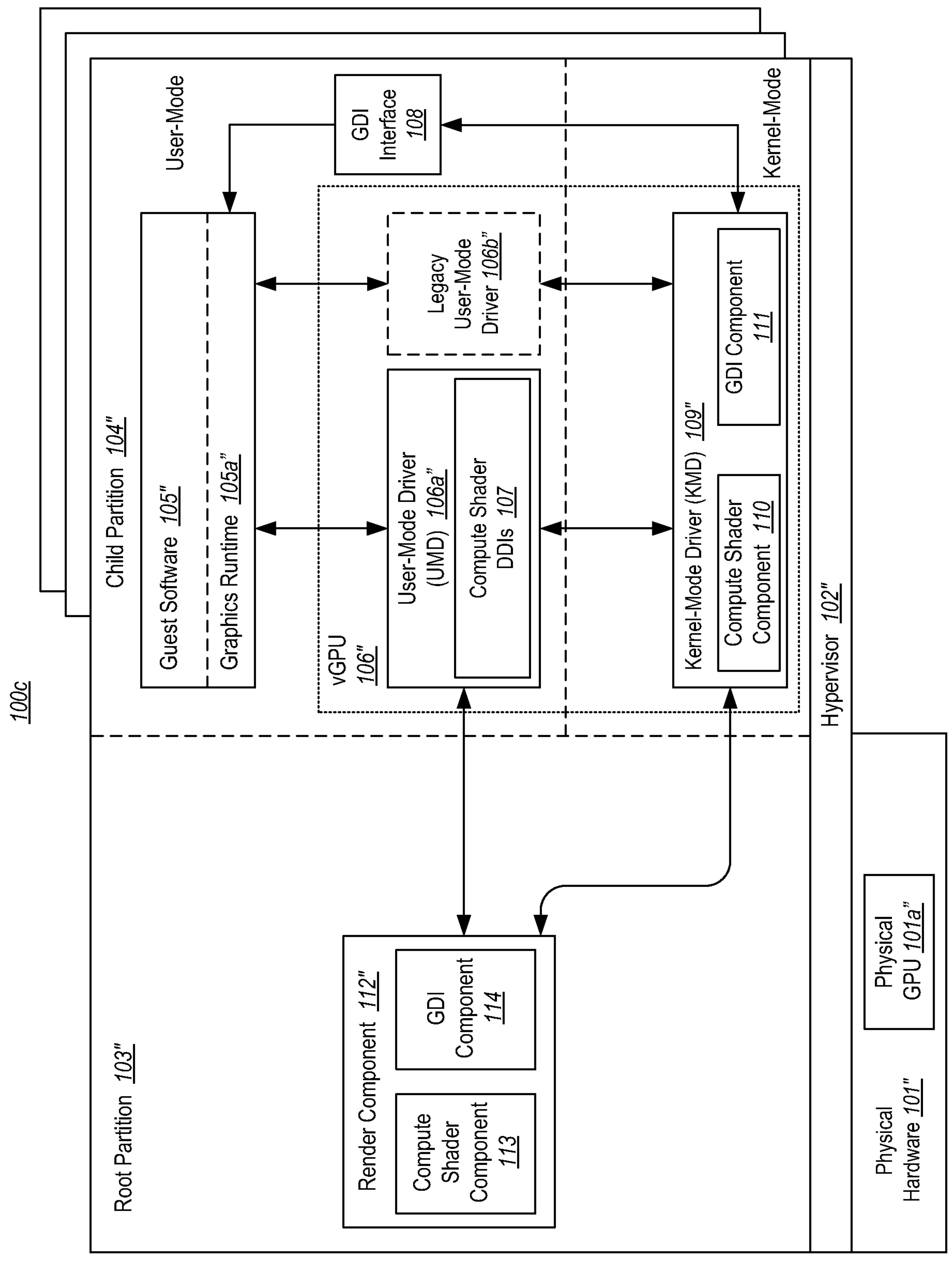
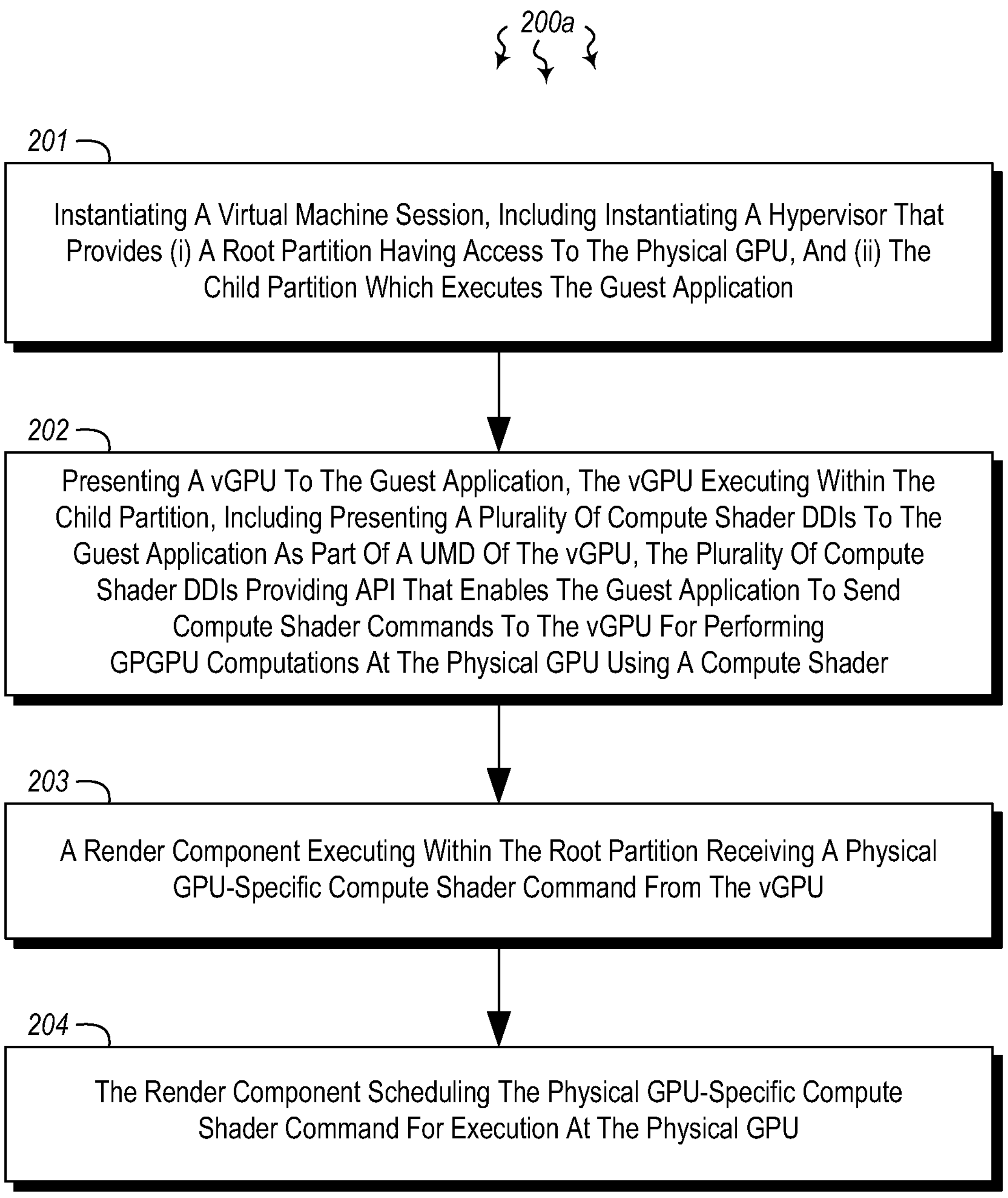

100c
Child Partition 104"
User-Mode
Kernel-Mode
GDI Interface 108
Guest Software 105"
Graphics Runtime 105a"
vGPU 106"
Legacy User-Mode Driver 106b"
User-Mode Driver (UMD) 106a"
Compute Shader DDIs 107
Kernel-Mode Driver (KMD) 109"
Compute Shader Component 110
GDI Component 111
Hypervisor 102"
Root Partition 103"
Render Component 112"
Compute Shader Component 113
GDI Component 114
Physical Hardware 101"
Physical GPU 101a"

Figure 1C



Instantiating A Virtual Machine Session, Including Instantiating A Hypervisor That Provides (i) A Root Partition Having Access To The Physical GPU, And (ii) The Child Partition Which Executes The Guest Application

Presenting A vGPU To The Guest Application, The vGPU Executing Within The Child Partition, Including Presenting A Plurality Of Compute Shader DDIs To The Guest Application As Part Of A UMD Of The vGPU, The Plurality Of Compute Shader DDIs Providing API That Enables The Guest Application To Send Compute Shader Commands To The vGPU For Performing GPGPU Computations At The Physical GPU Using A Compute Shader

A Render Component Executing Within The Root Partition Receiving A Physical GPU-Specific Compute Shader Command From The vGPU

The Render Component Scheduling The Physical GPU-Specific Compute Shader Command For Execution At The Physical GPU

Figure 2A

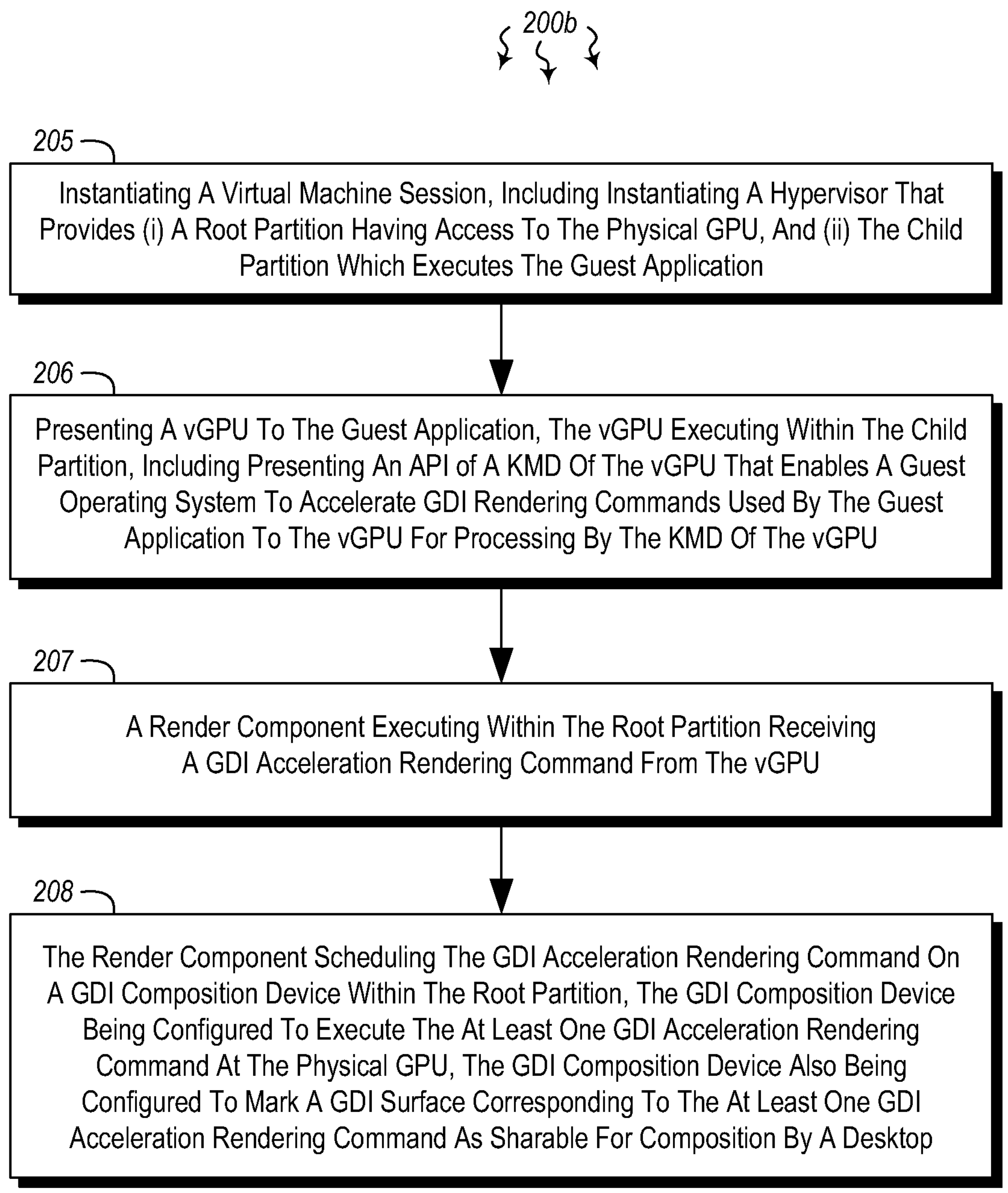

200b
205
Instantiating A Virtual Machine Session, Including Instantiating A Hypervisor That Provides (i) A Root Partition Having Access To The Physical GPU, And (ii) The Child Partition Which Executes The Guest Application
206
Presenting A vGPU To The Guest Application, The vGPU Executing Within The Child Partition, Including Presenting An API of A KMD Of The vGPU That Enables A Guest Operating System To Accelerate GDI Rendering Commands Used By The Guest Application To The vGPU For Processing By The KMD Of The vGPU
207
A Render Component Executing Within The Root Partition Receiving A GDI Acceleration Rendering Command From The vGPU
208
The Render Component Scheduling The GDI Acceleration Rendering Command On A GDI Composition Device Within The Root Partition, The GDI Composition Device Being Configured To Execute The At Least One GDI Acceleration Rendering Command At The Physical GPU, The GDI Composition Device Also Being Configured To Mark A GDI Surface Corresponding To The At Least One GDI Acceleration Rendering Command As Sharable For Composition By A Desktop

Figure 2B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/020916

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/06(2006.01)i, G06F 9/44(2006.01)i, G06F 9/30(2006.01)i, G06T 15/00(2006.01)i

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 9/06; H04N 7/173; G06T 1/20; G09G 5/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: "GPU", "VIRTUAL MACHINE", "SHADER", "PARTITION"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2007-0091102 A1 (JOHN BROTHERS et al.) 26 April 2007 See abstract, paragraphs [0044] - [0067] and figure 3.	1-15
A	US 2009-0322784 A1 (SARTORI GABRIELE) 31 December 2009 See abstract, paragraphs [0018] - [0064] and figure 1.	1-15
A	US 2011-0084973 A1 (MASOOD TARIQ) 14 April 2011 See abstract, paragraphs [0016] - [0036] and figure 4.	1-15
A	US 2006-0146057 A1 (DAVID BLYTHE) 06 July 2006 See abstract, paragraphs [0030] - [0126] and figure 2C.	1-15

☐ Further documents are listed in the continuation of Box C. ☒ 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 application or patent 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 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 26 April 2013 (26.04.2013)	Date of mailing of the international search report **29 April 2013 (29.04.2013)**
Name and mailing address of the ISA/KR Korean Intellectual Property Office 189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea Facsimile No. 82-42-472-7140	Authorized officer YUN, Byeong Soo Telephone No. 82-42-481-8530

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2013/020916

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2007-0091102 A1	26.04.2007	CN 100538737 C	09.09.2009
		CN 100590655 C	17.02.2010
		CN 101034469 A0	12.09.2007
		CN 101034469 B	12.05.2010
		CN 101034470 A0	12.09.2007
		CN 1983326 A	20.06.2007
		CN 1983326 C0	20.06.2007
		CN 1991905 A	04.07.2007
		CN 1991905 B	12.05.2010
		CN 1991905 C0	04.07.2007
		TW I326852A	01.07.2010
		TW I326852B	01.07.2010
		TW I331299A	01.10.2010
		TW I331299B	01.10.2010
		US 2007-0091101 A1	26.04.2007
		US 2007-0115292 A1	24.05.2007
		US 7737983 B2	15.06.2010
		US 7755632 B2	13.07.2010
		US 8004533 B2	23.08.2011
US 2009-0322784 A1	31.12.2009	TW 200948088 A	16.11.2009
		WO 2009-108354 A1	03.09.2009
US 2011-0084973 A1	14.04.2011	None	
US 2006-0146057 A1	06.07.2006	AU 2005-232324 A1	20.07.2006
		AU 2005-232324 B2	09.12.2010
		BR PI0505081A	12.09.2006
		CA 2528116 A1	30.06.2006
		CN 1797345 A	05.07.2006
		CN 1797345 B	13.07.2011
		CN 1797345 C0	05.07.2006
		EP 1677190 A2	05.07.2006
		JP 2006-190281 A	20.07.2006
		KR 10-1220072 B1	08.01.2013
		KR20060079088A	05.07.2006
		MX PA05012972A	29.06.2006
		RU 2005136419 A	27.05.2007
		US 8274518 B2	25.09.2012