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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 March 2014 (20.03.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/040194 A1

(51) International Patent Classification:
G06F 3/14 (2006.01) *G06F 13/20* (2006.01)

(21) International Application Number:
PCT/CA2013/050709

(22) International Filing Date:
17 September 2013 (17.09.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/613,218 13 September 2012 (13.09.2012) US

(71) Applicant: **ATI TECHNOLOGIES ULC** [CA/CA]; One Commerce Valley Drive East, Markham, Ontario L3T 7X6 (CA).

(72) Inventors: **HUNKINS, James D.**; 58 Craigmore Cres, Toronto, Ontario M2N 2Y7 (CA). **AU, Dennis K. W.**; 10 Burnt Bark Drive, Toronto, Ontario M1V 3J8 (CA).

(74) Agent: **SMART & BIGGAR**; Attn: ZISCHKA, Matthew, 438 University Avenue, Box 111, Suite 1500, Toronto, Ontario M5G 2K8 (CA).

(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, 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, SA,

[Continued on next page]

(54) Title: RENDERING SETTINGS IN A MULTI-GRAPHICS PROCESSING UNIT SYSTEM

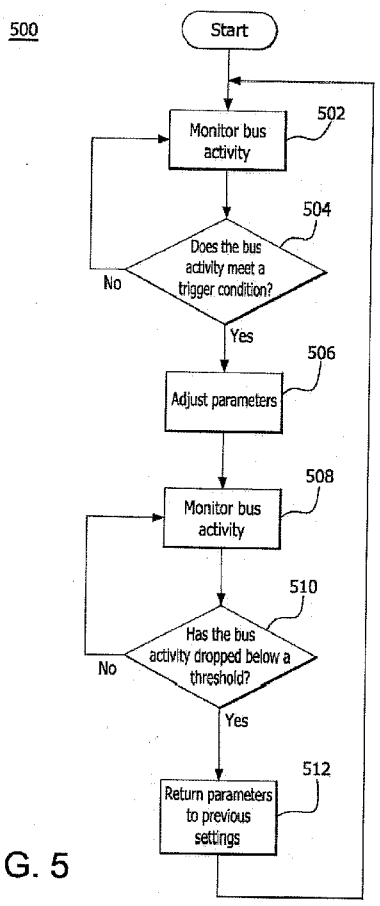

FIG. 5

(57) Abstract: Graphics rendering settings in a computer system are adjusted when an activity level on a bus meets a trigger condition. The graphics rendering settings of the system are returned to a previous level when the bus activity drops below a threshold. The trigger condition may be related to bandwidth usage on the bus or latency of data sent over the bus.

WO 2014/040194 A1

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, KM, ML, MR, NE, SN, TD, TG).

Published:

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

— *with information concerning request for restoration of the right of priority in respect of one or more priority claims; the decision of the receiving Office regarding the request for restoration is pending and will be published separately once available (Rules 26bis.3 and 48.2(j))*

RENDERING SETTINGS IN A MULTI-GRAPHICS PROCESSING UNIT SYSTEM

CROSS-REFERENCE TO RELATED APPLICATION

[0001] The present application claims priority to U.S. Patent Application No. 13/613,218, filed September 13, 2012, the disclosure of which is hereby expressly incorporated by reference.

FIELD OF THE INVENTION

[0002] The present invention is generally directed to graphics processing, and in particular, to adjusting graphics processing settings to allow for higher quality settings at higher frame rates, when supportable by the graphics system, while dynamically adjusting the rendering quality as needed to maintain the desired frame rates.

BACKGROUND

[0003] A user can adjust the initial display settings for an application that uses graphics processing (for example, rendered simulations or games). For example with games, this process currently involves trial and error, in which the user adjusts the settings and then has to play the game to determine whether those settings result in a desired display quality and display frame rate. If the display quality is not desirable (for example, the frame rate stalls), the user then needs to manually adjust the settings. Based on these adjustments, the user's selected settings generally do not stress the available hardware. Retaining smooth motion in the game (for example, so there is no frame rate stalling) means that a lower image quality sequence is displayed.

[0004] Multi-graphics processing unit (GPU) rendering systems require a certain amount of bus bandwidth to work properly, wherein a bus includes various types of communication fabrics including point-to-point protocols. Such systems work with an assumed maximum bus latency to help compensate for bus traffic issues related to available bandwidth and bus conflicts. If the bus usage exceeds the system's expected levels, different types of visible issues will become pronounced. It is noted that multiple GPUs can physically exist in multiple semiconductor packages, a single package with multiple semiconductor dies, or a single package with a single semiconductor die.

[0005] Because most graphics-intense software varies in how much rendering is required to display updated frames over time (i.e., variable frame rates and bus usage), the rendering settings normally attempt to minimize such bus-caused errors and visible issues. This results in lower frame rates and/or lower rendering quality.

[0006] Some multi-GPU rendering systems also use small buffer sizes to reduce system cost. But a small buffer leaves the system more vulnerable to potential latency issues. These cases are even more dependent on ensuring quality by maintaining a worst-case lower frame rate/rendering quality.

[0007] At times when the system is not heavily loaded, resources are left idle which could be used to produce faster (i.e., smoother) frame rates and/or improved graphics quality/reality. Because the system settings are normally preset, the worst-case scenario is normally assumed, to ensure a consistent viewing experience.

[0008] The existing solutions do not sufficiently address these problems. In one known solution, the frame rate is restricted to be low enough or some rendering settings are restricted to avoid frame failures. This results in less smooth display updates than other solutions or provides a lower quality display. Such restrictions result in less pleasing rendering results and could impact the immersion level in games or accuracy of simulations.

[0009] A second known solution is to use more memory for a larger frame buffer to avoid frame failures. This solution is expensive and power-hungry, and may also require more memory than is available in the system. For systems already installed with limited frame buffers, there is normally no option to increase the amount of memory.

[0010] A third known solution is to allow for some frame failures and supply a "recovery" mode, which will cause a visible disruption to the image during the frame error time period (of one or more frames). One method in this solution is to display the last "good" frame, resulting in tearing of part or all of the image. Another method in this solution is to allow the failure (visible in failed frames) and perform a recovery on each frame until the failure no longer happens. Either of these methods results in more disruptions to the image when the settings are not optimized to prevent frame errors.

SUMMARY OF EXEMPLARY EMBODIMENTS

[0011] Predictive error detection may be performed by monitoring different indicators that can help forecast potential "frame errors." As the potential error state is approached and errors start or become more frequent, the side effects or corrections will be more visible to the user. When the indicators reach a predetermined threshold, the system dynamically modifies different graphics settings to decrease the rendering and bus loading to prevent error cases with a small decrease of quality, which will be less noticeable than the error side effects. When the indicators indicate a decrease in the rendering and/or bus loading, the settings are adjusted to increase quality, which may include adjusting the settings so that they are restored to their earlier desired quality levels.

[0012] A method for adjusting graphics rendering settings in a computer system monitors an activity level on a bus. The graphics rendering settings of the system are adjusted when the bus activity meets a trigger condition. The graphics rendering settings of the system are returned to a previous level when the bus activity drops below a threshold.

[0013] A system for adjusting graphics rendering settings in a computer system includes a graphics processing unit (GPU), a frame buffer associated with the GPU, and a graphics driver in communication with the GPU and the frame buffer. The graphics driver is configured to monitor an activity level on a bus, adjust the graphics rendering settings of the system when the bus activity meets a trigger condition, and return the graphics rendering settings of the system to a previous level when the bus activity drops below a threshold.

[0014] A non-transitory computer-readable storage medium storing a set of instructions for execution by a general purpose computer to adjust graphics rendering settings in a computer system includes a monitoring code segment for monitoring an activity level on a bus, an adjusting code segment for adjusting the graphics rendering settings of the system when the bus activity meets a trigger condition, and a returning code segment for increasing the graphics rendering settings of the system when the bus activity drops below a threshold.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] A more detailed understanding may be had from the following description, given by way of example in conjunction with the accompanying drawings, wherein:

[0016] Figure 1 is a block diagram of an example device in which one or more disclosed embodiments may be implemented;

[0017] Figure 2 is a block diagram of an example system including one accelerated processing unit (APU) and one graphics processing unit (GPU);

[0018] Figure 3 is a block diagram of an example system including one central processing unit (CPU) and multiple GPUs;

[0019] Figure 4 is a block diagram of an example system including one CPU, multiple GPUs, and interim Peripheral Component Interconnect Express (PCIe) bridges;

[0020] Figure 5 is a flowchart of a method for adjusting graphics rendering settings based on a bus activity level;

[0021] Figure 6 is a diagram of a frame buffer showing fill levels;

[0022] Figure 7 is a flowchart of a bandwidth measurement method;

[0023] Figure 8 is a flowchart of a latency measurement method; and

[0024] Figure 9 is a flowchart of a method for returning the graphics rendering settings to their original values.

DETAILED DESCRIPTION

[0025] A computer system may be configured to run at a faster default frame rate, automatically detect when processing is beginning to slow down (back up), reduce some capabilities to maintain a smooth frame rate, and then return the capabilities to their original settings when processing has returned to normal levels. By automatically adjusting the settings as needed, the system may run at a faster frame rate and/or a higher quality than normal, with the results being transparent to the user. The application settings may be automatically checked and adjusted, without feedback to the user or intervention by the user.

[0026] Figure 1 is a block diagram of an example device 100 in which one or more disclosed embodiments may be implemented. The device 100 may include, for example, a computer, a gaming device, a handheld device, a set-top box, a television, a mobile phone, or a tablet computer. The device 100 includes a processor 102, a memory 104, a storage 106, one or more input devices 108, and one or more output devices 110. The device 100 may also optionally include an input driver 112 and an output driver 114. It is understood that the device 100 may include additional components not shown in Figure 1.

[0027] The processor 102 may include a central processing unit (CPU), a graphics processing unit (GPU), a CPU and GPU located on the same die, or one or more processor cores,

wherein each processor core may be a CPU or a GPU. The memory 104 may be located on the same die as the processor 102, or may be located separately from the processor 102. The memory 104 may include a volatile or non-volatile memory, for example, random access memory (RAM), dynamic RAM, or a cache.

[0028] The storage 106 may include a fixed or removable storage, for example, a hard disk drive, a solid state drive, an optical disk, or a flash drive. The input devices 108 may include a keyboard, a keypad, a touch screen, a touch pad, a detector, a microphone, an accelerometer, a gyroscope, a biometric scanner, or a network connection (e.g., a wireless local area network card for transmission and/or reception of wireless IEEE 802 signals). The output devices 110 may include a display, a speaker, a printer, a haptic feedback device, one or more lights, an antenna, or a network connection (e.g., a wireless local area network card for transmission and/or reception of wireless IEEE 802 signals).

[0029] The input driver 112 communicates with the processor 102 and the input devices 108, and permits the processor 102 to receive input from the input devices 108. The output driver 114 communicates with the processor 102 and the output devices 110, and permits the processor 102 to send output to the output devices 110. It is noted that the input driver 112 and the output driver 114 are optional components, and that the device 100 will operate in the same manner if the input driver 112 and the output driver 114 are not present.

[0030] Figure 2 is a block diagram of an example system 200 including one accelerated processing unit (APU) 202 and one graphics processing unit (GPU) 204 connected by a Peripheral Component Interconnect Express (PCIe) bus 206. The APU 202 communicates with a system memory 210 and is capable of sending information to one or more displays 212 for display to a user. The GPU 204 communicates with a frame buffer memory 220 and is capable of sending information to one or more displays 222 for display to the user.

[0031] Figure 3 is a block diagram of an example system 300 including one central processing unit (CPU) 302 and multiple GPUs 308. The CPU 302 communicates with a system application-specific integrated circuit (ASIC) 304. The system ASIC 304 communicates with a system memory 306 and each of the GPUs 308 via a separate PCIe bus 310. Each GPU 308 communicates with an associated frame buffer memory 312 and is capable of sending information to one or more displays 314 for display to the user. For discussion purposes, the below description of the operation of one embodiment focuses on the system 300 having two GPUs (308a, 308b). It is

noted that the operation of the embodiment is similar if the system 300 includes more than two GPUs.

[0032] Figure 4 is a block diagram of an example system 400 including one CPU 402, multiple GPUs 408, and interim PCIe bridges 410. The CPU 402 communicates with a system ASIC 404. The system ASIC 404 communicates with a system memory 406 and each of the GPUs 408 via the PCIe bridge 410 and PCIe bus segments 412, 414. Each GPU 408 communicates with an associated frame buffer memory 420 and is capable of sending information to one or more displays 422 for display to the user. For discussion purposes, the below description of the operation of one embodiment focuses on the system 400 having two GPUs (408a, 408b). It is noted that the operation of the embodiment is similar if the system 400 includes more than two GPUs.

[0033] With current systems (for example, systems 200, 300, 400), it is possible to determine when frame rate stalling or other types of system slow-down may occur. The system 200, 300, 400 may be configured to run at a faster default frame rate, automatically detect when processing is beginning to slow down (back up), reduce some capabilities to maintain a smooth frame rate, and then return the capabilities to their original settings when processing has returned to normal levels. By automatically adjusting the settings as needed, the system may run at a faster frame rate and/or a higher quality than normal, with the results being transparent to the user. The application settings may be automatically checked and adjusted, without feedback to the user or intervention by the user.

[0034] In one alternative implementation, the user may specify a minimum frame rate and adjust the rendering settings to match the specified minimum frame rate. In a second alternative implementation, a log of the changes to the rendering settings may be maintained, whereby the user could see how frequently the rendering settings were adjusted, to enable the user to select different initial rendering settings.

[0035] Figure 5 is a flowchart of a method 500 for adjusting graphics rendering settings based on a bus activity level. Bus activity (for example, activity on the PCIe bus) is monitored (step 502). A determination is made whether the bus activity meets a trigger condition (step 504). If there is not a high enough level of bus activity to meet the trigger condition, then the method continues to monitor the bus activity (step 502).

[0036] If the bus activity level meets the trigger condition (step 504), then the rendering parameters are adjusted (step 506). The bus activity is again monitored (step 508) and a

determination is made whether the bus activity has dropped below a predetermined threshold (step 510). If the bus activity has not dropped below the predetermined threshold, then the method continues to monitor the bus activity (step 508). If the bus activity has dropped below the predetermined threshold (step 510), then the rendering parameters are returned to their previous settings (step 512). The method 500 then loops back to step 502, to continuously monitor the bus activity.

[0037] The method 500 is a dynamic process, such that the rendering parameters are adjusted (step 506) when there is a probability of a frame failure or processing slowdown (as determined by the bus activity level). Adjustments may also be made based on the application that is currently running and the system's hardware, including the amount of available memory. Different adjustments may be made if the system has a low memory size, as opposed to a system with a larger memory size. For example, some adjustments would only need to be made because of a low memory size, and would not be necessary with a large memory size.

[0038] When predicting the potential frame failure rate, either the time to fill a buffer through the PCIe bus (bus bandwidth measurement) or the latency between the request and the arrival of the first "chunk" of pixel data may be monitored. The latency may be measured as the amount of time it takes from when the image data transfer is started to when the image data arrives at the destination.

[0039] By monitoring either of these indicators, an increase in the bus activity (for example, activity on the PCIe bus) can be observed as it happens. If the bus usage by the system, graphics devices, or multi-GPU rendering requirements rises too high, it may increase the latency of the rendering system, and therefore cause frame failures. As the bus activity increases, approaching the point where the buffer could underflow (not be filled in time to start the output to the display), the graphics driver may adjust the rendering parameters to decrease the bus load and therefore avoid a resulting visible frame error.

[0040] Two mechanisms may be used for error detection (step 502 of the method 500): bandwidth measurement and latency measurement. In some cases, the bandwidth measurement mechanism may be preferred when there is more variance in the performance. The latency measurement mechanism may be preferred when there is not much variance in the performance.

[0041] It is noted that other PCIe bus counters (other than bandwidth measurement and latency measurement) may be used to indicate retry rates and other factors which could indicate

bus issues. While such counters are usable, they may not be as accurate or as quick-responding as the bandwidth measurement and latency measurement mechanisms. Such other counters may also indicate other unrelated issues that may not be correctable via the methods described herein.

[0042] Lab measurements may be used to determine a starting point for different configurations and applications. The latency measurement may be used for an optimum setting of the Extended Dynamic Memory Access buffer size and underflow detection sensitivity. The bandwidth measurement may be used to check these results, confirming that the changes center around the "optimum" settings.

[0043] In gaming scenarios where scenes often change in complexity (for example, in shooter games), the latency method may be used when first turning on graphics multi-processing activity to check and set the initial settings, and then switch to the bandwidth measurement method to allow for faster measurements and response times.

[0044] In video streaming, there is a fairly consistent bandwidth usage. The latency method may be used both for the initial settings and for occasional adjustments over time. The bandwidth measurements will probably not make much of a difference in this situation, because of the more level bandwidth usage.

[0045] The bandwidth measurement method monitors how the video data flows from the rendering engine outwards. By using a bandwidth counter to time how many SCLKs (GPU clock cycles) happen from the time the first data enters the buffer until the buffer reaches a full level, the bandwidth used by the bus for all bus-related activities may be determined. A lower time indicates a lower bandwidth usage and less chance of a frame error. A higher time indicates increased bus usage and an increased chance of a frame error. If the time to fill the buffer increases too much, it will result in a buffer underflow, as the frame will need to start before the buffer is ready to send the data to the display.

[0046] If the frame buffer does not fill fast enough (meaning that the rendering engine and/or the bus are overloaded), then some capabilities will be reduced, so that the rendering engine can run faster and/or the bus has less traffic. Figure 6 shows an example of a frame buffer 600 with different fill levels. For example, the different fill levels include an empty level 602, a single item 604 in the buffer 600, and a desired fill level 606.

[0047] The desired buffer fill level 606 may be tuned (dynamically adjusted) based on the current system status, in terms of the available hardware and the application usage requirements. A

default buffer fill level 606 may be selected and then dynamically adjusted. Both the buffer fill level 606 and the application quality settings may be adjusted to achieve the desired frame rate output.

[0048] Figure 7 is a flowchart of a bandwidth measurement method 700 (which may be used in step 502 of the method 500). A determination is made whether there has been a frame buffer flip or a frame buffer reset (step 702). If either the frame buffer has been flipped or the frame buffer has been reset, then the buffer level is reset (step 704). Either of these conditions results in an empty buffer, so the buffer level should be reset. While the frame buffer has not flipped and has not been reset (step 702), the system continues to monitor for either of these events (repeating step 702).

[0049] Once the buffer level has been reset (step 704), a determination is made whether the buffer fill level is in a valid range (for example, greater than or equal to one and less than full) and whether a GPU clock cycle has occurred (step 706). If both conditions are met, then the bandwidth counter is increased (step 708). If both conditions are not met (step 706), then the system continues to monitor for both of these events (repeating step 706).

[0050] After the bandwidth counter is increased, a determination is made whether the buffer fill level indicates a full buffer (step 710). If the buffer is not full, then the system continues to monitor the buffer fill level (repeating step 710). If the buffer is full, then the maximum bandwidth value, the minimum bandwidth value, the average bandwidth value, and the trigger values are updated (step 712). The method 700 then repeats by performing the determinations over again, beginning with determining whether there has been a frame buffer flip or a frame buffer reset (step 702).

[0051] The trigger value for determining whether to adjust the rendering parameters (as used in step 504 of the method 500) may be based on the maximum bandwidth value, the minimum bandwidth value, and the average bandwidth value. In general, the maximum and minimum bandwidth measurement values provide a more instant and/or short-term change of the system behavior, while the average bandwidth measurement provides a long-term and more static bandwidth measurement. The adjustment should try to maintain the bandwidth utilization at or near the predefined average bandwidth most of the time. The adjustment should also respond to the maximum/minimum bandwidth change faster, to account for the short-term bandwidth fluctuation. If the maximum bandwidth measurement is detected for some amount of time, the adjustment to

the bandwidth should be conducted accordingly. If the maximum bandwidth reduces over time, the adjustment should bring the system to operate at or near the predefined average bandwidth. Hysteresis may be needed to smooth out the instantaneous adjustments from happening too frequently.

[0052] Figure 8 is a flowchart of a latency measurement method 800 (which may be used in step 502 of the method 500). A determination is made whether a request for a data chunk is the first request for the data chunk (step 802). It is noted that the term "data chunk" refers to an arbitrary size of a data request. If the first request is not made, the method waits until the request is made (repeating step 802). If the first request is made, a determination is made whether the first requested data chunk has been received and whether a GPU clock cycle has occurred (step 804). If the first requested data chunk has not been received and a GPU clock cycle has occurred, then the latency counter is increased (step 806) and the determination is repeated (step 804). Otherwise, the maximum latency value, the minimum latency value, the average latency value, and the trigger value are updated and the latency counter is reset (step 808). The method then repeats with determining whether a request for a data chunk is the first request for the data chunk (step 802).

[0053] In the latency measurement method, a latency counter may be used to measure the latency on the bus. This counter increments on every SCLK after the first request for a data "chunk" at a specific pixel location is made. Once the requested data chunk arrives, the counter stops incrementing. The lower time versus higher time has the same meaning as with the bandwidth measurement method. The latency method provides a faster first response than the bandwidth measurement method, but does not detect fluctuations during an individual buffer segment.

[0054] A frame buffer is an area of memory that holds the image data for one full screen (for example, a single monitor). The frame buffer is broken into smaller segments of memory to allow finer control. Therefore, the image can start drawing as soon as the first buffer segment is full, even though the system is still filling the remaining buffer segments for the specific frame buffer. If the buffer is not broken into segments, the system would have to wait for the entire frame buffer to fill completely before it could start displaying the image on the monitor. This buffer segment method reduces the latency from the start of the frame rendering until the image is displayed on the screen.

[0055] Another reason for segmenting the buffer is for the buffer fullness and emptiness monitoring. The data return to the buffer is out-of-order, so the traditional first-in, first-out read/write pointer base level monitoring does not work well. Segmenting the buffer simplifies monitoring the buffer fullness and emptiness, because the buffer level can be determined by the number of empty segments irrespective of the order of the segments in the buffer.

[0056] Lab testing using the bandwidth or latency counters along with maximum and minimum recordings of the counter and a frame error count may be used to determine thresholds for the trigger setting and the duration of parameter adjustment levels. An algorithm to smooth the results and avoid false triggers and too frequent adjustments may be used. There are different ways to implement the smoothing algorithm. In one example implementation, the algorithm does not respond to the first bandwidth increase, but responds to the average of several samples. The algorithm may respond on the high side or the low side, depending on prior patterns. The parameter settings would then return to the desired average. Assuming that the system indicated a return to "normal" usage, the settings would eventually be adjusted back to the starting level, possibly in steps.

[0057] Different parameter adjustment levels may be needed for different types of media. For example, high-resolution third person shooter-type games will likely have a larger and more frequent change in the bus activity level than, for example, a video playback. Therefore, different settings may be used for each situation, to better optimize the system.

[0058] Figure 9 is a flowchart of a method for returning the graphics rendering settings to their original values (step 512 of the method 500). How the graphics rendering settings are returned to their original values depends on the application running and the amount of change in the settings. The goal is to avoid presenting a visible difference to the user.

[0059] The relative change between the adjusted graphics rendering settings (step 506 of the method 500) and the original graphics rendering settings is determined (step 902). A determination is then made whether the relative change is greater than a predetermined threshold (step 904). If the relative change is less than the predetermined threshold, then the graphics rendering settings are returned to their original settings in one step (step 906) and the method terminates (step 908). If the relative change is greater than the predetermined threshold (step 904), then the graphics rendering settings are gradually returned to their original settings (step 910) and the method terminates (step 908).

[0060] Returning the graphics rendering settings from the adjusted values to the original values may be performed all at once (if the relative changes were minor) or gradually (if the relative changes were large). A gradual return to the original values is used so as to not visually disrupt the user's experience (making a large change all at once would result in a noticeable difference in the display). A smoothing algorithm may be used to determine the setting changes used to implement the gradual return. In one implementation of the smoothing algorithm, large changes may be made first, followed by successively smaller changes. The goal of the smoothing algorithm is to perform the changes in such a way that the user's visual experience is not noticeably disrupted.

[0061] Only items under control of the graphics drivers can be adjusted. Items not related to the graphics from the system side that may be using the PCIe bus, and therefore impacting the bus bandwidth/latency to the graphics system, are beyond the scope of this method. But the graphics adjustments can help compensate for non-graphics activity on the PCIe bus.

[0062] Several items may be used to decrease the PCIe bus activity and/or affect how long it takes to render new frames. The examples given here are a subset of what can be used. The type of application along with the hardware configurations of the graphics system and the main system will dictate which methods are the most effective.

[0063] Case 1: Reducing texture sizes and/or complexity may decrease the PCIe bus activity in some cases, especially for graphics systems with small frame buffers.

[0064] Case 2: Adjusting graphics rendering parameters that may cause more or less "data sharing" between GPUs (for example, GPU to GPU transfers). Some examples include super anti-aliasing, shadows, particle physics, etc.

[0065] Case 3: Limiting the frame rate will decrease the data transfers across the bus.

[0066] Case 4: Reducing the color depth or changing color formats may help in systems that support more than 24-bit ARGB (red green blue alpha) formats.

[0067] Case 5: Other physics or OpenCL™ activities that are heavy bus users may be limited or reduced in priority.

[0068] It should be understood that many variations are possible based on the disclosure herein. Although features and elements are described above in particular combinations, each feature or element may be used alone without the other features and elements or in various combinations with or without other features and elements. In one embodiment, the methods

described herein are implemented in the graphics driver. Other embodiments may be envisioned in which the methods are implemented elsewhere in the system 200, 300, 400. Such other embodiments would need to communicate with the graphics driver to obtain the necessary information to perform the methods.

[0069] The methods provided may be implemented in a general purpose computer, a processor, or a processor core. Suitable processors include, by way of example, a general purpose processor, a special purpose processor, a conventional processor, a digital signal processor (DSP), a plurality of microprocessors, one or more microprocessors in association with a DSP core, a controller, a microcontroller, Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs) circuits, any other type of integrated circuit (IC), and/or a state machine. Such processors may be manufactured by configuring a manufacturing process using the results of processed hardware description language (HDL) instructions and other intermediary data including netlists (such instructions capable of being stored on a computer readable media). The results of such processing may be maskworks that are then used in a semiconductor manufacturing process to manufacture a processor which implements aspects of the present invention.

[0070] The methods or flow charts provided herein may be implemented in a computer program, software, or firmware incorporated in a non-transitory computer-readable storage medium for execution by a general purpose computer or a processor. Examples of non-transitory computer-readable storage mediums include a read only memory (ROM), a random access memory (RAM), a register, cache memory, semiconductor memory devices, magnetic media such as internal hard disks and removable disks, magneto-optical media, and optical media such as CD-ROM disks, and digital versatile disks (DVDs).

* * *

CLAIMS

What is claimed is:

1. A method for adjusting graphics rendering settings in a computer system, comprising:

adjusting the graphics rendering settings of the system when an activity level on a bus meets a trigger condition; and

returning the graphics rendering settings of the system to a previous level when the bus activity level drops below a threshold.

2. The method according to claim 1, wherein the bus is a Peripheral Component Interconnect Express bus.

3. The method according to claim 1, further comprising:

monitoring the activity level on the bus.

4. The method according to claim 3, wherein the monitoring includes measuring bandwidth usage on the bus.

5. The method according to claim 4, wherein measuring the bandwidth usage on the bus includes:

increasing a bandwidth counter when a frame buffer level is in a predetermined range; and

updating a maximum bandwidth value, a minimum bandwidth value, an average bandwidth value, and a trigger value when the buffer level is full.

6. The method according to claim 5, wherein the predetermined range is greater than or equal to one and less than full.

7. The method according to claim 3, wherein the monitoring includes measuring latency on the bus.

8. The method according to claim 7, wherein measuring the latency includes:

increasing a latency counter when a first data chunk has not been received and a graphics processing unit clock cycle has occurred; and

updating a maximum latency value, a minimum latency value, an average latency value, and a trigger value when the first data chunk has been received.

9. The method according to claim 1, wherein the trigger condition is based on a maximum bandwidth value, a minimum bandwidth value, and an average bandwidth value.

10. The method according to claim 1, wherein the trigger condition is based on a maximum latency value, a minimum latency value, and an average latency value.

11. The method according to claim 1, wherein the returning includes:

returning the graphics rendering settings to the previous level in one step when a relative change between the previous graphics rendering settings and the adjusted graphics rendering settings is less than a threshold; and

gradually returning the graphics rendering settings to the previous level when the relative change between the previous graphics rendering settings and the adjusted graphics rendering settings is greater than the threshold.

12. The method according to claim 11, wherein gradually returning the graphics rendering settings to the previous level includes using a smoothing algorithm.

13. A system for adjusting graphics rendering settings in a computer system, comprising:

a graphics processing unit (GPU);

a frame buffer associated with the GPU; and

a graphics driver in communication with the GPU and the frame buffer, the graphics driver configured to:

adjust the graphics rendering settings of the system when an activity level on a bus meets a trigger condition; and

return the graphics rendering settings of the system to a previous level when the bus activity level drops below a threshold.

14. The system according to claim 13, wherein the system includes:

a plurality of GPUs;

a plurality of frame buffers, each frame buffer associated with one of the plurality of GPUs; and

the graphics driver is in communication with each of the plurality of GPUs and each of the plurality of frame buffers.

15. The system according to claim 13, wherein the bus is a Peripheral Component Interconnect Express bus.

16. The system according to claim 13, wherein the graphics driver is further configured to:

monitor the activity level on the bus.

17. The system according to claim 16, wherein the graphics driver is further configured to:

measure bandwidth usage on the bus;

increase a bandwidth counter when a frame buffer level is in a predetermined range; and

update a maximum bandwidth value, a minimum bandwidth value, an average bandwidth value, and a trigger value when the buffer level is full.

18. The system according to claim 17, wherein the predetermined range is greater than or equal to one and less than full.

19. The system according to claim 16, wherein the graphics driver is further configured to:

measure latency on the bus;

increase a latency counter when a first data chunk has not been received and a graphics processing unit clock cycle has occurred; and

update a maximum latency value, a minimum latency value, an average latency value, and a trigger value when the first data chunk has been received.

20. The system according to claim 13, wherein the trigger condition is based on a maximum bandwidth value, a minimum bandwidth value, and an average bandwidth value.

21. The system according to claim 13, wherein the trigger condition is based on a maximum latency value, a minimum latency value, and an average latency value.

22. The system according to claim 13, wherein the graphics driver is further configured to:

return the graphics rendering settings to the previous level in one step when a relative change between the previous graphics rendering settings and the adjusted graphics rendering settings is less than a threshold; and

gradually return the graphics rendering settings to the previous level when the relative change between the previous graphics rendering settings and the adjusted graphics rendering settings is greater than the threshold.

23. The system according to claim 22, wherein the graphics driver is further configured to gradually return the graphics rendering settings to the previous level using a smoothing algorithm.

24. A non-transitory computer-readable storage medium storing a set of instructions for execution by a general purpose computer to adjust graphics rendering settings in a computer system, the set of instructions comprising:

an adjusting code segment for adjusting the graphics rendering settings of the system when an activity level on a bus meets a trigger condition; and

a returning code segment for returning the graphics rendering settings of the system to a previous level when the bus activity level drops below a threshold.

25. The non-transitory computer-readable storage medium according to claim 24, wherein the instructions are hardware description language (HDL) instructions used for the manufacture of a device.

26. A method for adjusting graphics rendering settings in a computer system, comprising:

reducing a quality level of the graphics rendering settings of the system when a bus activity level exceeds a first threshold condition; and

increasing the quality level of the graphics rendering settings of the system when the bus activity level drops below a second threshold condition.

27. The method according to claim 26, wherein the first threshold condition and the second threshold condition differ.

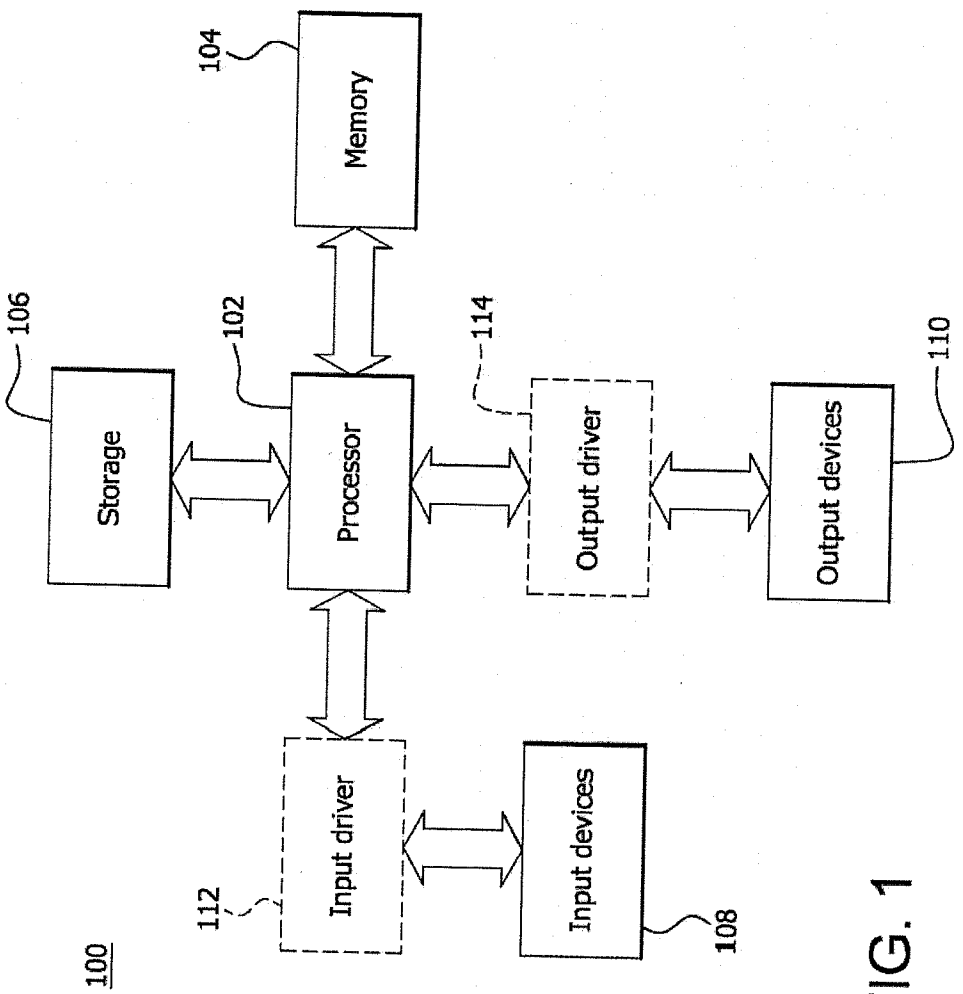
100
Memory
104
Storage
106
Processor
102
Output driver
114
Output devices
110
Input driver
112
Input devices
108

FIG. 1

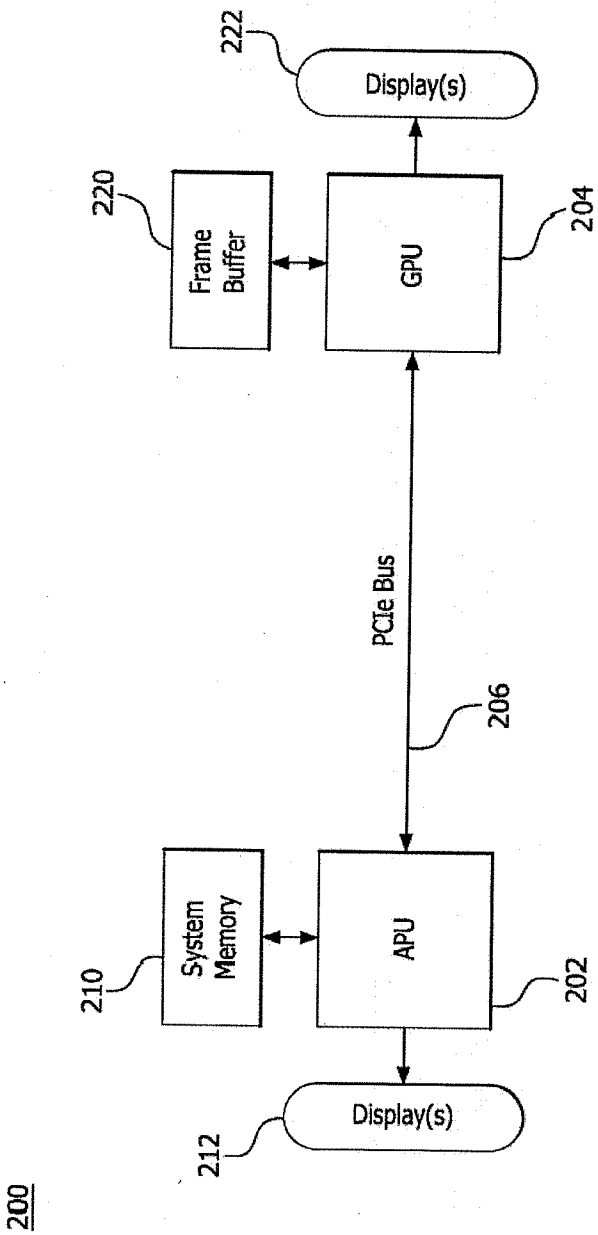
222
Display(s)
220
Frame Buffer
GPU
204
PCIe Bus
206
System Memory
210
APU
202
Display(s)
212
200

FIG. 2

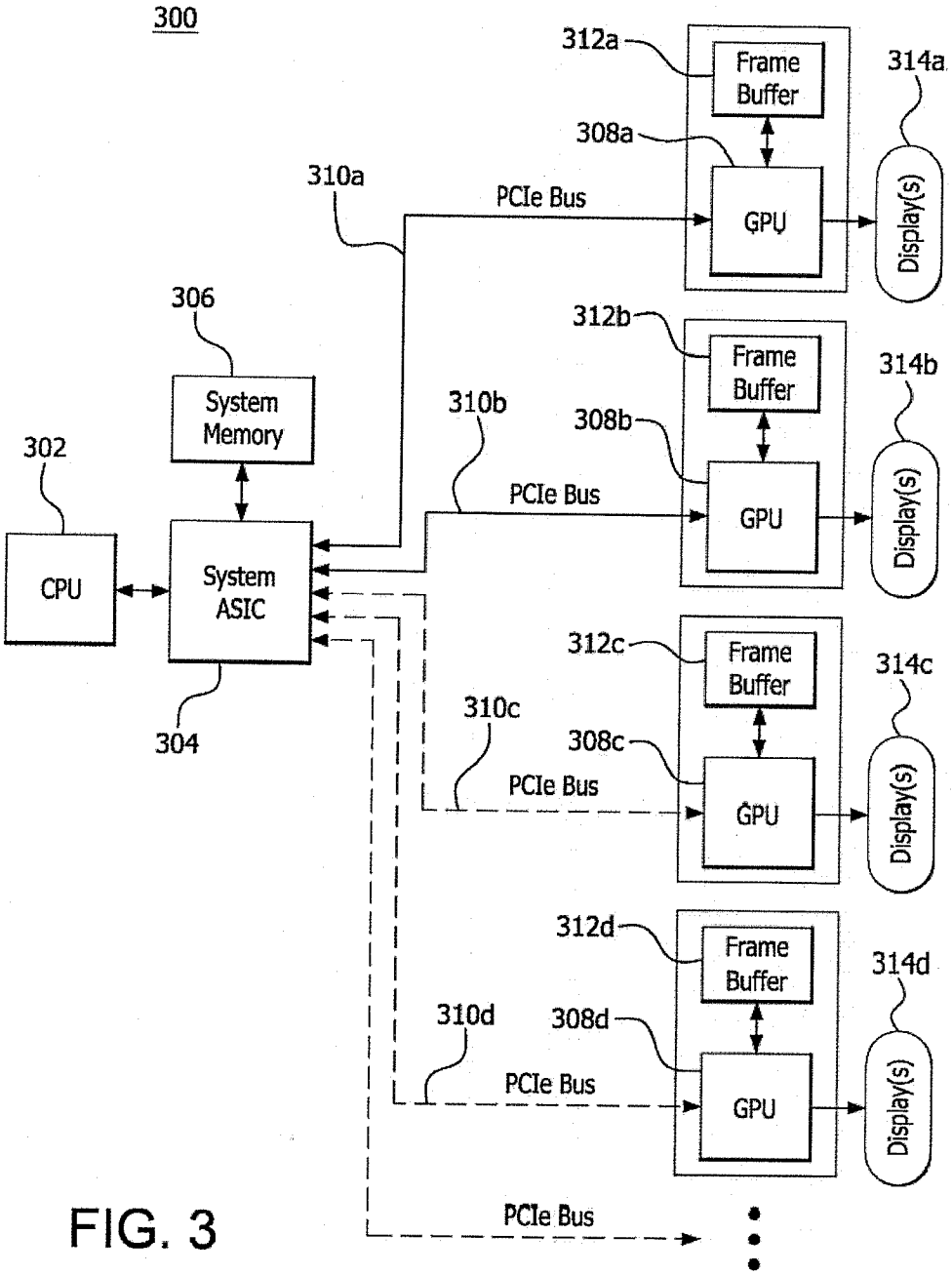
300
312a
Frame Buffer
314a
308a
310a
PCIe Bus
GPU
Display(s)
306
312b
Frame Buffer
314b
System Memory
310b
308b
302
PCIe Bus
GPU
Display(s)
CPU
System ASIC
312c
Frame Buffer
314c
310c
308c
304
PCIe Bus
GPU
Display(s)
312d
Frame Buffer
314d
310d
308d
PCIe Bus
GPU
Display(s)
PCIe Bus

FIG. 3

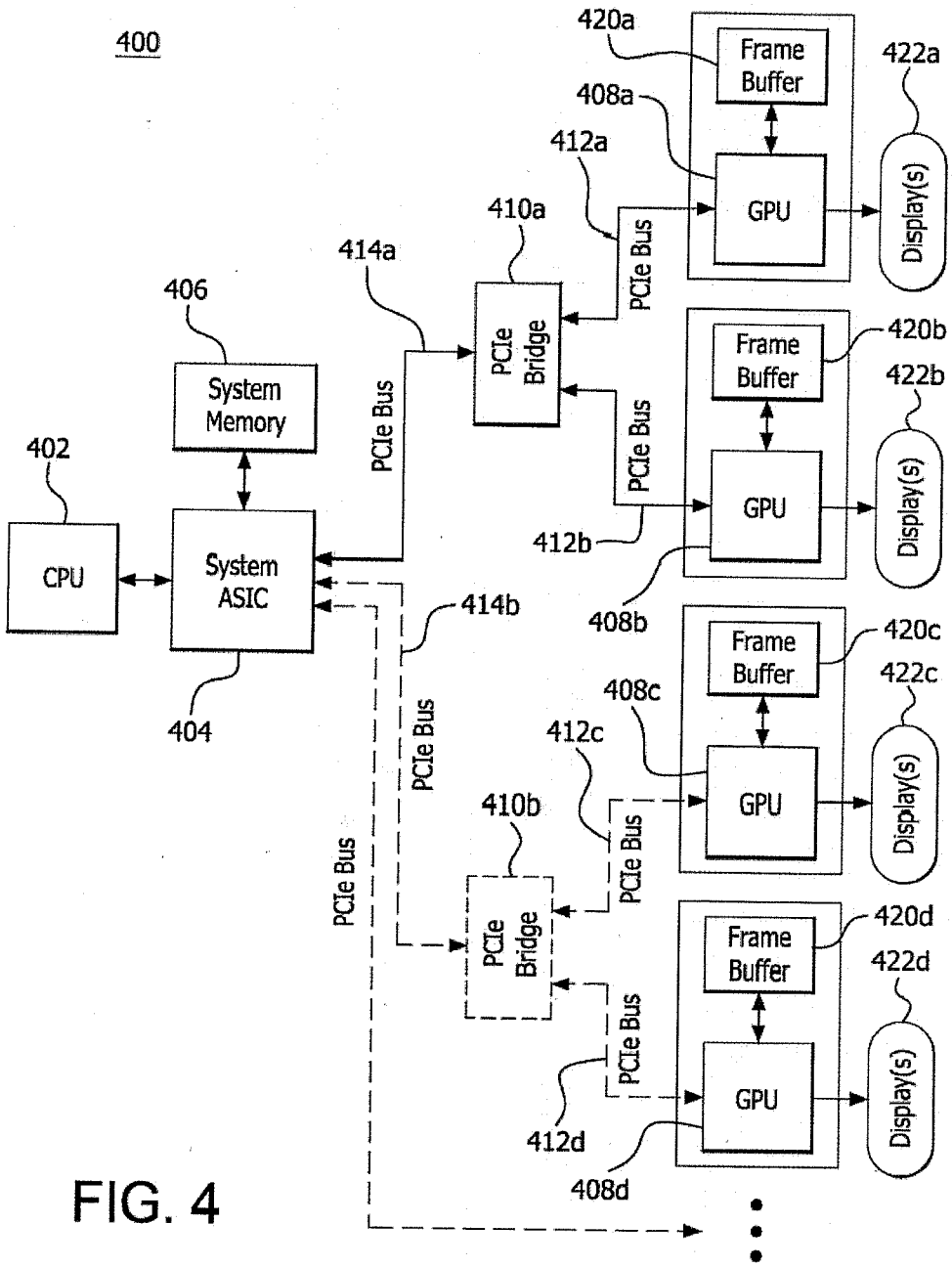
400
420a
Frame Buffer
422a
408a
412a
GPU
Display(s)
410a
414a
406
PCIe Bridge
PCIe Bus
System Memory
420b
Frame Buffer
422b
402
PCIe Bus
GPU
Display(s)
CPU
System ASIC
412b
408b
414b
420c
Frame Buffer
422c
408c
412c
GPU
Display(s)
PCIe Bus
410b
PCIe Bus
PCIe Bridge
420d
Frame Buffer
422d
PCIe Bus
GPU
Display(s)
412d
408d
404

FIG. 4

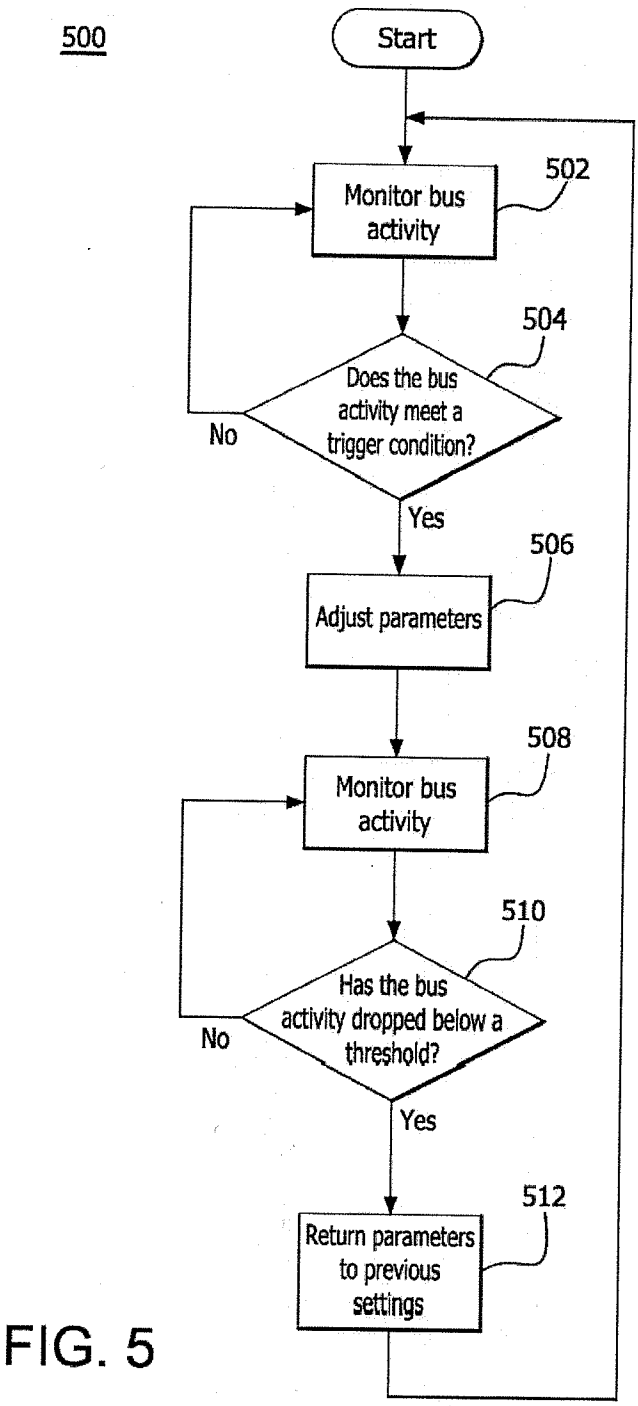
500
Start
Monitor bus activity
502
Does the bus activity meet a trigger condition?
504
No
Yes
Adjust parameters
506
Monitor bus activity
508
Has the bus activity dropped below a threshold?
510
No
Yes
Return parameters to previous settings
512

FIG. 5

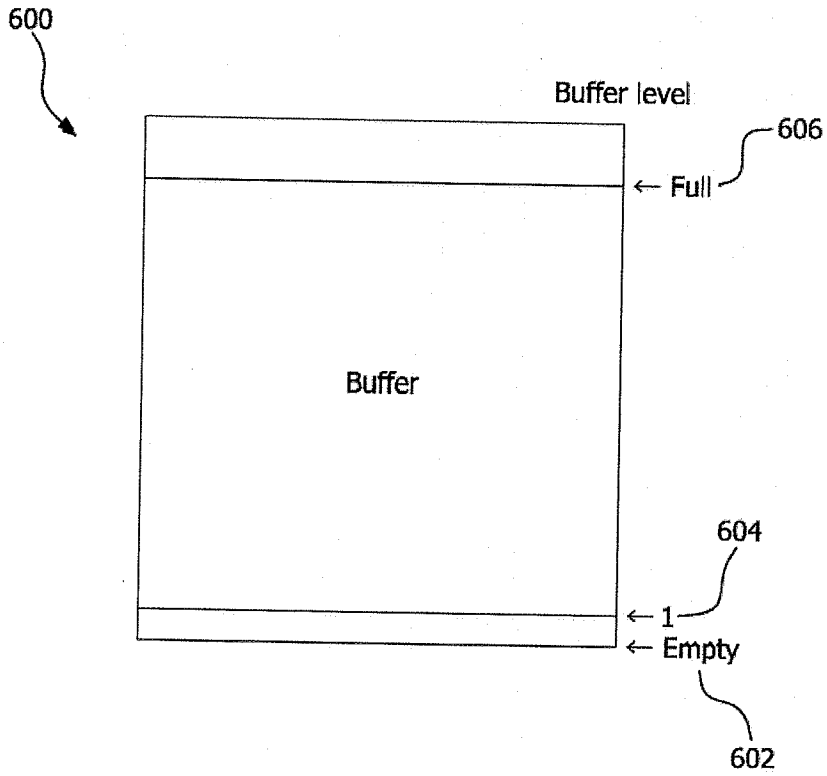

FIG. 6

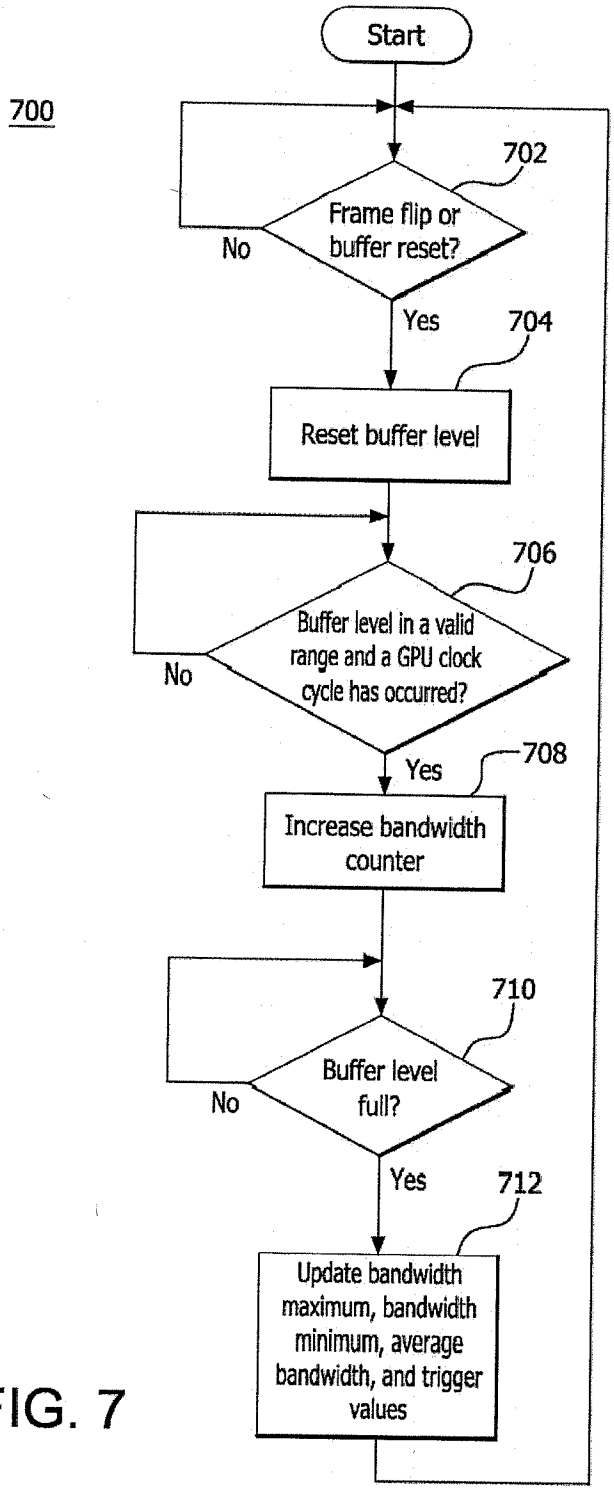
700
Start
702
Frame flip or buffer reset?
No
Yes
704
Reset buffer level
706
Buffer level in a valid range and a GPU clock cycle has occurred?
No
Yes
708
Increase bandwidth counter
710
Buffer level full?
No
Yes
712
Update bandwidth maximum, bandwidth minimum, average bandwidth, and trigger values

FIG. 7

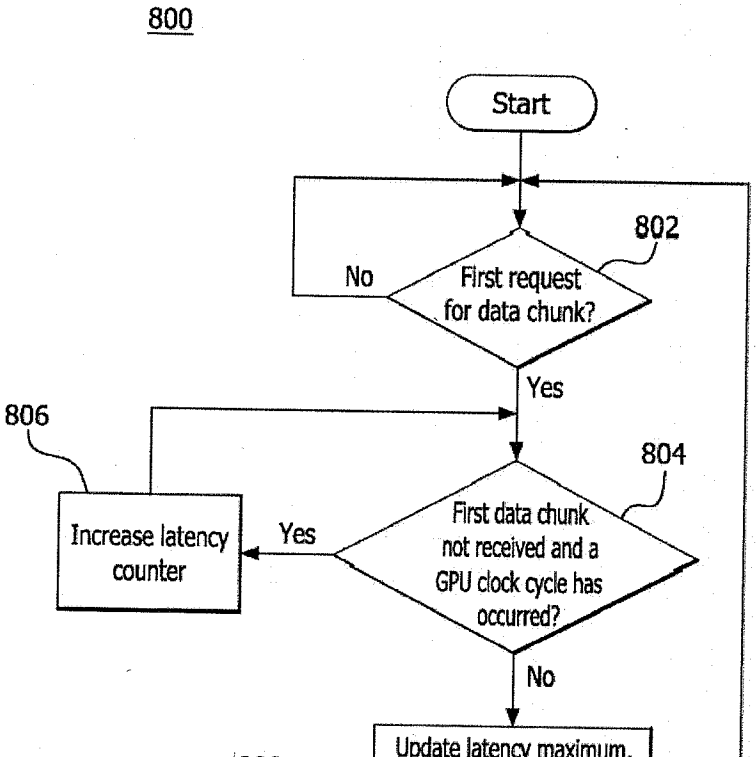
800
Start
802
First request for data chunk?
No
Yes
806
Increase latency counter
Yes
804
First data chunk not received and a GPU clock cycle has occurred?
No
808
Update latency maximum, latency minimum, average latency, and trigger values. Reset latency counter.

FIG. 8

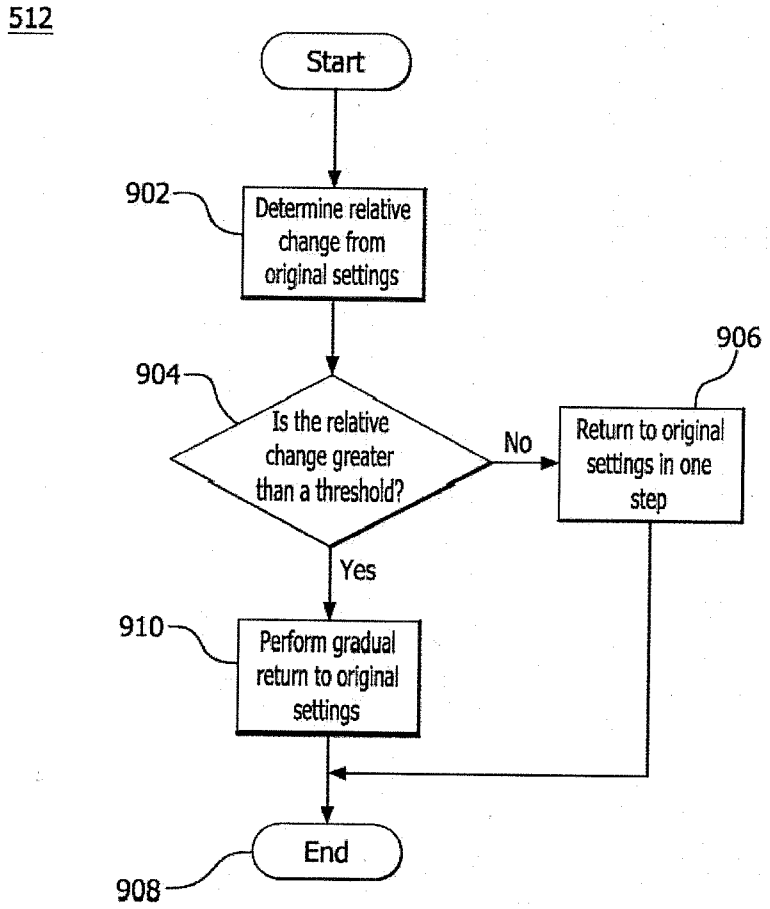
512
Start
902
Determine relative change from original settings
904
Is the relative change greater than a threshold?
No
906
Return to original settings in one step
Yes
910
Perform gradual return to original settings
End
908

FIG. 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2013/050709

A. CLASSIFICATION OF SUBJECT MATTER
IPC: ***G06F 3/14*** (2006.01) , *G06F 13/20* (2006.01)
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)

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

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Databases: Canadian Patents Database, TotalPatent, IEEE Xplore, Advanced GOOGLE Scholar, EPOQUE
Keywords: multi-graphics processing unit rendering systems, (adjusted/ modified/ altered/ changed) graphics rendering settings/ parameters/ conditions/ values, bus activity/ traffic level trigger/ threshold, bandwidth/ latency counter

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X, Y	US7539809B2, (*Juenger*) 26 May 2009 (26-05-2009) * abstract; col. 2, lines 32- 35; col. 2, lines 67- col. 3, line 2; col. 4, lines 4- 30; col. 4, lines 42- 46; col. 4, line 60- col. 5, line 10; Figures 1 and 2 *	1, 2, 3, 4, 9, 11, 15, 16, 20, 22, 24, 25 5, 6, 7, 8, 10, 12, 13, 14, 17, 18, 19, 21, 23, 26, 27
Y	US7526593B2, (*Mandal et al.*) 28 April 2009 (28-04-2009) * col. 8, lines 47- 38; col. 13, lines 14- 31, col. 22, lines 12- 22; Figures 1 and 4 *	5, 6, 7, 8, 10, 12, 13, 14, 17, 18, 19, 21, 23
Y	US7616206B1, (*Danilak*) 10 November 2009 (10-11-2009) * abstract; col. 3, lines 54- 65; col. 4, lines 20- 25, 51- 58; col. 5, lines 17- 41; col. 7, lines 11- 17; Figures 2, 4 and 6 *	6, 14, 18, 26, 27
A	US2008068389A1, (*Bakalash et al.*) 20 March 2008 (20-03-2008) * see whole document *	1- 27
A	US2008304738A1, (*Beilloin*) 11 December 2008 (11-12-2008) *see whole document*	1- 27
A	US2011063308A1, (*Hunkins et al.*) 17 March 2011 (17-03-2011) *see whole document*	1- 27

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents :

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

"E" earlier 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 another citation or other special reason (as specified)

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
01 December 2013 (01-12-2013)	27 December 2013 (27-12-2013)

Name and mailing address of the ISA/CA	Authorized officer
Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	Tanya Novo-Verde (819) 934-4891

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P, X	US2013179711A1, (*Aelion et al.*) 11 July 2013 (11-07-2013) *see whole document*	1- 27
A	"Graphics Processing Units for Handhelds " (*Akenine-Moller, T. ; Strom, J.*) Proceedings of the IEEE Volume: 96 , Issue: 5 Publication Year: 2008 , Page(s): 779 - 789 Retrieved 7 November 2013 (07-11-2013) from the internet at following location: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4483498 *see whole document*	1- 27

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
US7539809B2	26 May 2009 (26-05-2009)	US2007067548A1	22 March 2007 (22-03-2007)
US7526593B2	28 April 2009 (28-04-2009)	CN1904868A	31 January 2007 (31-01-2007)
		CN100524266C	05 August 2009 (05-08-2009)
		EP1750202A1	07 February 2007 (07-02-2007)
		US2007079044A1	05 April 2007 (05-04-2007)
US7616206B1	10 November 2009 (10-11-2009)	None	
US2008068389A1	20 March 2008 (20-03-2008)	CA2514296A1	19 August 2004 (19-08-2004)
		CA2546427A1	02 June 2005 (02-06-2005)
		CA2595085A1	09 November 2006 (09-11-2006)
		CA2637800A1	10 January 2008 (10-01-2008)
		CA2674351A1	10 July 2008 (10-07-2008)
		CA2676050A1	20 February 2010 (20-02-2010)
		CN1890660A	03 January 2007 (03-01-2007)
		CN1926579A	07 March 2007 (07-03-2007)
		CN101849227A	29 September 2010 (29-09-2010)
		EP1590769A2	02 November 2005 (02-11-2005)
		EP1590769A4	04 April 2007 (04-04-2007)
		EP1687732A2	09 August 2006 (09-08-2006)
		EP1687732A4	19 November 2008 (19-11-2008)
		EP1846834A2	24 October 2007 (24-10-2007)
		JP2007512613A	17 May 2007 (17-05-2007)
		JP2007528033A	04 October 2007 (04-10-2007)
		JP2008538620A	30 October 2008 (30-10-2008)
		US2006146072A1	06 July 2006 (06-07-2006)
		US7233964B2	19 June 2007 (19-06-2007)
		US2008088630A1	17 April 2008 (17-04-2008)
		US7777748B2	17 August 2010 (17-08-2010)
		US2008165197A1	10 July 2008 (10-07-2008)
		US7796129B2	14 September 2010 (14-09-2010)
		US2008165184A1	10 July 2008 (10-07-2008)
		US7796130B2	14 September 2010 (14-09-2010)
		US2008136825A1	12 June 2008 (12-06-2008)
		US7800610B2	21 September 2010 (21-09-2010)
		US2008238917A1	02 October 2008 (02-10-2008)
		US7800611B2	21 September 2010 (21-09-2010)
		US2008165198A1	10 July 2008 (10-07-2008)
		US7800619B2	21 September 2010 (21-09-2010)
		US2007279411A1	06 December 2007 (06-12-2007)
		US7808499B2	05 October 2010 (05-10-2010)
		US2008129745A1	05 June 2008 (05-06-2008)
		US7808504B2	05 October 2010 (05-10-2010)
		US2006232590A1	19 October 2006 (19-10-2006)
		US7812844B2	12 October 2010 (12-10-2010)
		US2008129744A1	05 June 2008 (05-06-2008)
		US7812845B2	12 October 2010 (12-10-2010)
		US2008117219A1	22 May 2008 (22-05-2008)
		US7812846B2	12 October 2010 (12-10-2010)
		US2006279577A1	14 December 2006 (14-12-2006)
		US7834880B2	16 November 2010 (16-11-2010)
		US2008122851A1	29 May 2008 (29-05-2008)
		US7843457B2	30 November 2010 (30-11-2010)
		US2008084420A1	10 April 2008 (10-04-2008)
		US7940274B2	10 May 2011 (10-05-2011)
		US2008100629A1	01 May 2008 (01-05-2008)
		US7944450B2	17 May 2011 (17-05-2011)
		US2009027402A1	29 January 2009 (29-01-2009)
		US7961194B2	14 June 2011 (14-06-2011)
		US2008117217A1	22 May 2008 (22-05-2008)
		US8085273B2	27 December 2011 (27-12-2011)
		US2008100630A1	01 May 2008 (01-05-2008)
		US8125487B2	28 February 2012 (28-02-2012)
		US2008129748A1	05 June 2008 (05-06-2008)
		US8134563B2	13 March 2012 (13-03-2012)
		US2009128551A1	21 May 2009 (21-05-2009)
		US8284207B2	09 October 2012 (09-10-2012)
		US2008158236A1	03 July 2008 (03-07-2008)
		US8497865B2	30 July 2013 (30-07-2013)
		US2007291040A1	20 December 2007 (20-12-2007)
		US2008074428A1	27 March 2008 (27-03-2008)
		US2008074429A1	27 March 2008 (27-03-2008)
		US2008074431A1	27 March 2008 (27-03-2008)
		US2008079737A1	03 April 2008 (03-04-2008)
		US2008084418A1	10 April 2008 (10-04-2008)
		US2008084419A1	10 April 2008 (10-04-2008)

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
		US2008084421A1	10 April 2008 (10-04-2008)
		US2008084422A1	10 April 2008 (10-04-2008)
		US2008084423A1	10 April 2008 (10-04-2008)
		US2008088631A1	17 April 2008 (17-04-2008)
		US2008088632A1	17 April 2008 (17-04-2008)
		US2008094402A1	24 April 2008 (24-04-2008)
		US2008094403A1	24 April 2008 (24-04-2008)
		US2008094404A1	24 April 2008 (24-04-2008)
		US2008117218A1	22 May 2008 (22-05-2008)
		US2008122850A1	29 May 2008 (29-05-2008)
		US2008129741A1	05 June 2008 (05-06-2008)
		US2008129742A1	05 June 2008 (05-06-2008)
		US2008129743A1	05 June 2008 (05-06-2008)
		US2008129747A1	05 June 2008 (05-06-2008)
		US2008136826A1	12 June 2008 (12-06-2008)
		US2008136827A1	12 June 2008 (12-06-2008)
		US2008158235A1	03 July 2008 (03-07-2008)
		US2008165196A1	10 July 2008 (10-07-2008)
		US2008198167A1	21 August 2008 (21-08-2008)
		US2008211817A1	04 September 2008 (04-09-2008)
		US2008246772A1	09 October 2008 (09-10-2008)
		US2008316216A1	25 December 2008 (25-12-2008)
		US2009027383A1	29 January 2009 (29-01-2009)
		US2009096798A1	16 April 2009 (16-04-2009)
		US2009128550A1	21 May 2009 (21-05-2009)
		US2009135190A1	28 May 2009 (28-05-2009)
		US2009179894A1	16 July 2009 (16-07-2009)
		US2011072056A1	24 March 2011 (24-03-2011)
		US2011169840A1	14 July 2011 (14-07-2011)
		US2011169841A1	14 July 2011 (14-07-2011)
		US2011279462A1	17 November 2011 (17-11-2011)
		US2012262463A1	18 October 2012 (18-10-2012)
		US2013120410A1	16 May 2013 (16-05-2013)
		US2013176322A1	11 July 2013 (11-07-2013)
		WO2004070652A2	19 August 2004 (19-08-2004)
		WO2004070652A3	14 September 2006 (14-09-2006)
		WO2005050557A2	02 June 2005 (02-06-2005)
		WO2005050557A3	01 September 2005 (01-09-2005)
		WO2006117683A2	09 November 2006 (09-11-2006)
		WO2006117683A3	22 May 2009 (22-05-2009)
		WO2008004135A2	10 January 2008 (10-01-2008)
		WO2008004135A9	20 March 2008 (20-03-2008)
		WO2008004135A3	23 April 2009 (23-04-2009)
		WO2008082641A2	10 July 2008 (10-07-2008)
		WO2008082641A3	09 October 2008 (09-10-2008)
US2008304738A1	11 December 2008 (11-12-2008)	US2008304738A1	11 December 2008 (11-12-2008)
		US8019151B2	13 September 2011 (13-09-2011)
		WO2008152513A2	18 December 2008 (18-12-2008)
		WO2008152513A3	07 July 2011 (07-07-2011)
US2011063308A1	17 March 2011 (17-03-2011)	CN101953033A	19 January 2011 (19-01-2011)
		JP2011507106A	03 March 2011 (03-03-2011)
		JP5341912B2	13 November 2013 (13-11-2013)
		KR20100093600A	25 August 2010 (25-08-2010)
		US2009157914A1	18 June 2009 (18-06-2009)
		US7861013B2	28 December 2010 (28-12-2010)
		WO2009076436A1	18 June 2009 (18-06-2009)
US2013179711A1	11 July 2013 (11-07-2013)	None	