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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 February 2008 (21.02.2008)

PCT

(10) International Publication Number
WO 2008/021794 A1

(51) International Patent Classification:
G06F 9/06 (2006.01) *G06F 9/54* (2006.01)

(21) International Application Number:
PCT/US2007/075286

(22) International Filing Date: 6 August 2007 (06.08.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/501,632 8 August 2006 (08.08.2006) US

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95052 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **COTA-ROBLES, Erik** [US/US]; 251B Chiquita Avenue, Mountain View, California 94041 (US). **NEIGER, Gilbert** [US/US]; 2424 NE 11th Avenue, Portland, Oregon 97212 (US). **BENNETT, Steven** [US/US]; 6469 SE Sigrid Street, Hillsboro, Oregon 97123 (US). **ANDERSON, Andrew** [US/US]; 677 SE 68th Ave, Hillsoboro, Oregon 97123 (US).

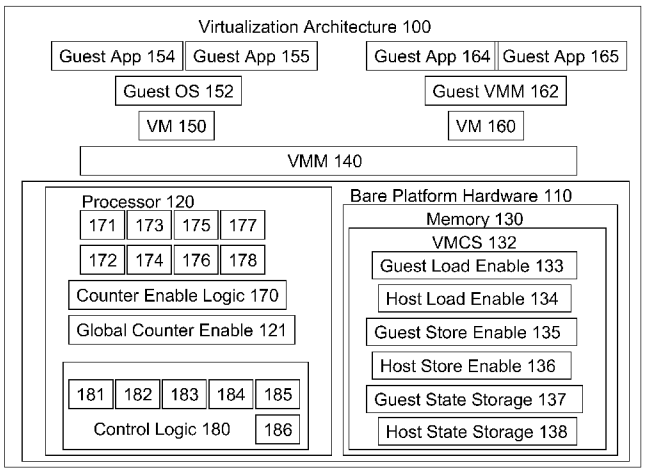
(74) Agents: **VINCENT, Lester, J.** et al.; Blakely Sokoloff Taylor & Zafman, 12400 Wilshire Boulevard, 7th Floor, Los Angeles, California 90025 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, 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, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

(54) **Title:** VIRTUALIZING PERFORMANCE COUNTERS



(57) **Abstract:** Embodiments of apparatuses, methods, and systems for virtualizing performance counters are disclosed. In one embodiment, an apparatus includes a counter, a counter enable storage location, counter enable logic, and virtual machine control logic. The counter enable storage location is store a counter enable indicator. The counter enable logic is to enable the counter based on the counter enable indicator. The virtual machine control logic is to transfer control of the apparatus to a guest. The virtual machine control logic includes guest state load logic to cause a guest value from a virtual machine control structure to be loaded into the counter enable storage location in connection with a transfer of control of the apparatus to a guest.

WO 2008/021794 A1

VIRTUALIZING PERFORMANCE COUNTERS

BACKGROUND

1. Field

[0001] The present disclosure pertains to the field of information processing, and more particularly, to the field of using performance counters in a virtualization environment.

2. Description of Related Art

[0002] Generally, the concept of virtualization in information processing systems allows multiple instances of one or more operating systems (each, an "OS") to run on a single information processing system, even though each OS is designed to have complete, direct control over the system and its resources. Virtualization is typically implemented by using software (e.g., a virtual machine monitor, or a "VMM") to present to each OS a "virtual machine" ("VM") having virtual resources, including one or more virtual processors, that the OS may completely and directly control, while the VMM maintains a system environment for implementing virtualization policies such as sharing and/or allocating the physical resources among the VMs (the "virtualization environment"). Each OS, and any other software, that runs on a VM is referred to as a "guest" or as "guest software," while a "host" or "host software" is software, such as a VMM, that runs outside of the virtualization environment.

[0003] A physical processor in an information processing system may support virtualization, for example, by supporting an instruction to enter a virtualization environment to run a guest on a virtual processor (i.e., a physical processor under constraints imposed by a VMM) in a VM. In the virtualization environment, certain events, operations, and situations, such as external interrupts or attempts to access privileged registers or resources, may be intercepted, i.e., cause the processor to exit the virtualization environment so that a VMM may operate, for example, to implement virtualization policies. A physical processor may also support other instructions for maintaining a virtualization environment, and may include memory or register bits that indicate or control virtualization capabilities of the physical processor.

[0004] A physical processor supporting a virtualization environment may include performance counters for logging performance monitoring information. Typically, each

performance counter would be controlled by one or more control or configuration registers, or portions of a one or more control or configuration registers, associated with the counter. To virtualize these performance counters, each transition between a guest and a host or between two guests would typically require saving the state of all of the counters and their associated control registers and loading a new state for all of the counters and their associated control registers.

Brief Description of the Figures

[0005] The present invention is illustrated by way of example and not limitation in the accompanying figures.

[0006] Figure 1 illustrates an embodiment of the present invention in a virtualization architecture.

[0007] Figure 2 illustrates an embodiment of the present invention in a method for a virtualizing performance counters.

[0008] Figure 3 illustrates another embodiment of the present inventions in a virtualization architecture.

[0009] Figure 4 illustrates another embodiment of the present invention in a method for a virtualizing performance counters.

Detailed Description

[0010] Embodiments of apparatuses, methods, and systems for virtualizing performance counters are described below. In this description, numerous specific details, such as component and system configurations, may be set forth in order to provide a more thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art, that the invention may be practiced without such specific details. Additionally, some well known structures, circuits, and the like have not been shown in detail, to avoid unnecessarily obscuring the present invention.

[0011] The performance of a virtualization environment may be improved by reducing the amount of state information that must be saved and loaded on transitions between a host and a guest and between guests. Embodiments of the present invention may be used

to virtualize performance counters or other counters without requiring the saving and the loading of the contents of the counters and their associated control registers. Therefore, performance may be improved over a virtualization environment in which the contents of the counters and their associated control registers are saved on transitions. Embodiments of the present invention provide for performance counters to be efficiently enabled or disabled for any number of hosts and guests, for performance counters to be assigned for the exclusive use of a host or a guest, and for performance counters to be shared between any number of hosts and/or guests.

[0012] Figure 1 illustrates an embodiment of the present invention in virtualization architecture 100. In Figure 1, bare platform hardware 110 may be any data processing apparatus capable of executing any OS or VMM software. For example, bare platform hardware may be that of a personal computer, mainframe computer, portable computer, handheld device, set-top box, server, or any other computing system. Bare platform hardware 110 includes processor 120 and memory 130.

[0013] Processor 120 may be any type of processor, including a general purpose microprocessor, such as a processor in the Intel® Pentium® Processor Family, Itanium® Processor Family, or other processor family from Intel® Corporation, or another processor from another company, or a digital signal processor or microcontroller. Although Figure 1 shows only one such processor 120, bare platform hardware 110 may include any number of processors, including any number of multicore processors, each with any number of execution cores, and any number of multithreaded processors, each with any number of threads.

[0014] Memory 130 may be static or dynamic random access memory, semiconductor-based read-only or flash memory, magnetic or optical disk memory, any other type of medium readable by processor 120, or any combination of such mediums. Processor 120 and memory 130 may be coupled to or communicate with each other according to any known approach, such as directly or indirectly through one or more buses, point-to-point, or other wired or wireless connections. Bare platform hardware 110 may also include any number of additional devices or connections.

[0015] In addition to bare platform hardware 100, Figure 1 illustrates VMM 140, VMs 150 and 160, guest operating systems 152 and 162, and applications 154, 155, 164, and 165.

[0016] VMM 140 may be any software, firmware, or hardware host installed on or accessible to bare platform hardware 110, to present VMs, i.e., abstractions of bare platform hardware 110, to guests, or to otherwise create VMs, manage VMs, and implement virtualization policies. In other embodiments, a host may be any VMM, hypervisor, OS, or other software, firmware, or hardware capable of controlling bare platform hardware 110. A guest may be any OS, any VMM, including another instance of VMM 140, any hypervisor, or any application or other software.

[0017] Each guest expects to access physical resources, such as processor and platform registers, memory, and input/output devices, of bare platform hardware 110, according to the architecture of the processor and the platform presented in the VM. Figure 1 shows two VMs, 150 and 160, with guest OS 152 and guest applications 154 and 155 installed on VM 150 and guest OS 162 and guest applications 164 and 165 installed on VM 160. Although Figure 1 shows only two VMs and two applications per VM, any number of VMs may be created, and any number of applications may run on each VM within the scope of the present invention.

[0018] A resource that can be accessed by a guest may either be classified as a "privileged" or a "non-privileged" resource. For a privileged resource, VMM 140 facilitates the functionality desired by the guest while retaining ultimate control over the resource. Non-privileged resources do not need to be controlled by VMM 140 and may be accessed directly by a guest.

[0019] Furthermore, each guest OS expects to handle various events such as exceptions (e.g., page faults, and general protection faults), interrupts (e.g., hardware interrupts and software interrupts), and platform events (e.g., initialization and system management interrupts). These exceptions, interrupts, and platform events are referred to collectively and individually as "virtualization events" herein. Some of these virtualization events are referred to as "privileged events" because they must be handled by VMM 140 to ensure proper operation of VMs 150 and 160, protection of VMM 140 from guests, and protection of guests from each other.

[0020] At any given time, processor 120 may be executing instructions from VMM 140 or any guest, thus VMM 140 or the guest may be running on, or in control of, processor 120. When a privileged event occurs or a guest attempts to access a privileged resource, control may be transferred from the guest to VMM 140. The transfer of control

from a guest to VMM 140 is referred to as a "VM exit" herein. After handling the event or facilitating the access to the resource appropriately, VMM 140 may return control to a guest. The transfer of control from VMM 140 to a guest is referred to as a "VM entry" herein.

[0021] In the embodiment of Figure 1, processor 120 controls the operation of VMs 150 and 160 according to data stored in virtual machine control structure ("VMCS") 132. VMCS 132 is a structure that may contain state of a guest or guests, state of VMM 140, execution control information indicating how VMM 140 is to control operation of a guest or guests, information regarding VM exits and VM entries, and any other such information. Processor 120 reads information from VMCS 132 to determine the execution environment of a VM and constrain its behavior. In this embodiment, VMCS 132 is stored in memory 130. In some embodiments, multiple VMCSs are used to support multiple VMs. Although Figure 1 shows VMCS 132 stored in memory 130, storing a VMCS in a memory is not required by the present invention.

[0022] Processor 120 includes counters 171, 173, 175, and 177. In this embodiment, counters 171, 173, 175, and 177 are performance counters that may be programmed by software running on processor 120 to log performance monitoring information; however, other embodiments may include any number of counters and/or any type or size of counter. For example, any of performance counters 171, 173, 175, or 177 may be programmed to increment for each occurrence of a selected event, or to increment for each clock cycle during a selected event. The events may include any of a variety of events related to execution of program code on processor 120, such as branch mispredictions, cache hits, cache misses, translation lookaside buffer hits, translation lookaside buffer misses, etc. Therefore, performance counters 171, 173, 175, and 177 may be used for tuning or profiling program code to yield the best possible performance on processor 120.

[0023] Processor 120 also includes counter control storage locations 172, 174, 176, and 178, corresponding to counters 171, 173, 175, and 177, respectively. Counter control storage locations 172, 174, 176, and 178 may be registers or any other structures of any size, or portions of one or more registers of other structures, for storing information to control or configure counters 171, 173, 175, and 177, respectively. Counter control storage locations 172, 174, 176, and 178 may be programmed to store information to control or configure counters 171, 173, 175, and 177, respectively, such as information to

enable the counter, to select the event to be counted, to select the method of counting (e.g., number of occurrences or duration of event), to select conditions for counting (e.g., based on privilege level of software executing when event is detected) and to set any other control, configuration, or other variables.

[0024] Processor 120 also includes global counter enable storage location 121, which may be one or more bits in a control register, configuration register, model specific register, or any other storage location to store a global counter enable indicator. In one embodiment, global counter enable storage location 121 may be a single bit of a programmable register, where the bit may be set to a logical one to set the global counter enable indicator to an "enable" value or a logical zero to set the global counter enable indicator to a "disable" value.

[0025] In another embodiment, a global counter enable storage location may include one bit per performance counter, where each bit may be set to a logical one to set an individual counter enable indicator for a corresponding performance counter to an "enable" value. In other embodiments, a single bit or field of a global counter enable storage location may correspond to a group of any number of performance counters. In any of these embodiments, a global enable storage location may include any number of bits or fields that correspond to any number of performance counters, and may further include or be associated with one or more additional enable indicators that may be used to control any number of bits or fields that more directly control performance counters. For example, a storage location may include a field to store a "counter enable vector," which includes one bit per counter, and an additional bit within this storage location or in a storage location or data structure elsewhere in processor 120 or virtualization architecture 100 may be used to enable or disable the counter enable vector. In this way, individual counters may be individually controlled by guests according to a counter enable vector stored in a VMCS, but the default value of the additional bit may be set such that the individual counter control is automatically disabled when executing a VMM or other host software designed for a processor that does not support a counter enable vector model.

[0026] The value of the global counter enable indicator is used, along with any other relevant information stored in counter control storage locations 172, 174, 176, and 178, by counter enable logic 170 to control the operation of counters 171, 173, 175, and 177, respectively. If the global counter enable indicator is set to an "enable" value, then each

counter operates according to its individual control and configuration information. For example, counter 171 may increment on a cache hit because the contents of counter control storage location 172 include an individual counter enable bit set to an "enable" value and an event select field set to count cache hits, while at the same time counter 173 may not increment because the contents of counter control storage location 174 include an individual counter enable bit set to a "disable" value. However, if the global counter enable indicator is set to a "disable" value, all counters 171, 173, 175, and 177 are disabled and do not operate according to their individual control and configuration information. For example, counter 171 would not increment on a cache hit even if the contents of counter control storage location 172 include an individual counter enable bit set to an "enable" value and an event select field set to count cache hits.

[0027] Additionally, processor 120 includes control logic 180 to support virtualization, including the virtualization of counters 171, 173, 175, and 177. Control logic 180 may be microcode, programmable logic, hard-coded logic, or any other form of control logic within processor 120. In other embodiments, control logic 180 may be implemented in any form of hardware, software, or firmware, such as a processor abstraction layer, within a processor or within any component accessible or medium readable by a processor, such as memory 130.

[0028] Control logic 180 causes processor 120 to execute method embodiments of the present invention, such as the method embodiments described below with reference to Figure 2, for example, by causing processor 120 to include the execution of one or more micro-operations, e.g., to support virtualization, in its response to virtualization instructions or other instructions from a host or guest.

[0029] Control logic 180 includes VM entry logic 181 to transfer control of processor 120 from a host to a guest (i.e., a VM entry) and VM exit logic 182 to transfer control of processor 120 from a guest to a host (i.e., a VM exit). In some embodiments, control may also be transferred from a guest to a guest or from a host to a host. For example, in an embodiment supporting layered virtualization, software running on a VM on processor 120 may be both a guest and a host (e.g., a VMM running on a VM is a guest to the VMM that controls that VM and a host to a guest running on a VM that it controls).

[0030] Control logic 180 also includes guest state load logic 183, host state load logic 184, and guest state store logic 185. Guest state load logic 183 is to load guest state from

VMCS 132 to processor 120 on a VM entry. Host state load logic 184 is to load host state from VMCS 132 to processor 120 on a VM exit. Guest state store logic 185 is to store guest state from processor 120 to VMCS 132 on a VM exit. In some embodiments control logic 180 also includes host state store logic 186 to store host state from processor 120 to VMCS 132 on a VM entry prior to guest state load logic 183 loading guest state into processor 120. In some embodiments, control logic 180 may load and store state residing in other system components, e.g., input-output devices, memory controllers.

[0031] VMCS 132 may include fields, control bits, or other data structures to support virtualization. These data structures may be checked or otherwise referred to by control logic 180 to determine how to manage a VM environment. For example, guest state load enable indicator 133 may be set to cause guest state load logic 183 to load a guest value from guest state storage location 137 into global counter enable storage location 121 in connection with a VM entry, host state load enable indicator 134 may be set to cause host state load logic 184 to load a host value from host state storage location 138 into global counter enable storage location 121 in connection with a VM exit, guest state store enable indicator 135 may be set to cause guest state store logic 185 to store the contents of global enable storage location 121 in guest state storage location 137, and host state store enable indicator 136 may be set to cause host state store logic 186 to store the contents of global enable storage location 121 host state storage location 138, all as further described below. In this description of this embodiment, these indicators are control bits that are set to enable or cause a desired effect, where set means writing a logical one to the bit, but any logic convention or nomenclature may be used within the scope of the present invention.

[0032] Using the mechanism described herein, a VMM may enable counting of events that occur only while a guest executes, only while the VMM executes or while both the VMM and a guest execute. Additionally, a VMM may, on a guest by guest basis, enable or disable performance counters as part of the transition (i.e., the VM entry or VM exit) between the guest and the VMM. This mechanism allows the VMM to "hide" its effect on the performance counters from a guest. Alternatively, this mechanism allows counting of events in the VMM for use by a guest, or in a guest for use by the VMM.

[0033] Figure 2 illustrates an embodiment of the present invention in method 200, a method for virtualizing performance counters. Although method embodiments are not limited in this respect, reference is made to virtualization architecture 100 of Figure 1 to

describe the method embodiment of Figure 2.

[0034] In box 210 of Figure 2, a first performance counter is configured to count occurrences of a first event. For example, counter 171 may be configured, by programming counter control storage location 172, to count cache misses. In box 212, a second performance counter is configured to count occurrences of a second event. For example, counter 173 may be configured, by programming counter control storage location 174, to count cycles not halted. In other embodiments, the performance counters may be configured to count occurrences of any other events, e.g., translation lookaside buffer misses, branch mispredictions, etc.

[0035] In box 220 of Figure 2, VMM 140 of Figure 1 creates a VMCS (e.g., VMCS 132) for a VM. In boxes 222 to 226, VMM 140 configures VMCS 132 to implement support for virtualizing counters 171, 173, 175, and 177. In box 222, guest state load enable indicator 133 is set to cause guest state load logic 183 to load a guest value from guest state storage location 137 into global counter enable storage location 121 in connection with a VM entry. In box 224, host state load enable indicator 134 is set to cause host state load logic 184 to load a host value from host state storage location 138 into global counter enable storage location 121 in connection with a VM exit. In box 226, guest state store enable indicator 135 is set to cause guest state store logic 185 to store the contents of global enable storage location 121 in guest state storage location 136.

[0036] In box 230, a transfer of control of processor 120 from the host (i.e., VMM 140) to a guest is initiated. For example, the VMM may initiate a VM entry. The VM entry may include VM entry logic 181 causing processor 120 to execute operations or micro-operations to save the host state and load the guest state. In box 232, VM entry logic determines whether to load a guest value from guest state storage location 137 into global counter enable storage location 121 based on guest state load enable indicator 133. If guest state load enable indicator 133 is set, then, in box 234, guest state load logic 183 causes the guest value from guest state storage location 137 to be loaded into global counter enable storage location 121; otherwise, box 234 is not performed. In box 236, the VM entry is completed and control is transferred to the guest. In box 238, the guest begins or continues to execute.

[0037] In box 239, counter enable logic 170 determines whether the global counter enable indicator is set. If not, method 200 continues at box 258. If so, method 200

continues at box 240. In box 240, it is determined whether the first event has occurred. If so, then, in box 244, the first performance counter operates as configured, i.e., in this embodiment, where it has been configured in box 210 to count cache misses, the first performance counter increments. In box 250, it is determined whether the second event has occurred. If so, then, in box 254, the second performance counter operates as configured, i.e., in this embodiment, where it has been configured in box 212 to count occurrences of the second event, the second performance counter increments.

[0038] In box 258, it is determined whether a virtualization event has occurred. If not, method 200 continues at box 238. If so, then, in box 260, a transfer of control of processor 120 from the guest to the host is initiated. The VM exit may include VM exit logic 182 causing processor 120 to execute operations or micro-operations to save the guest state and load the host state.

[0039] In box 262, VM exit logic determines whether to store the contents of global counter enable storage location 121 in guest state storage location 137 based on guest state store enable indicator 135. If guest state store enable indicator 135 is set, then, in box 264, guest state store logic 185 stores the contents of global counter enable storage location 121 in guest state storage location 137; otherwise, box 264 is not performed. In some embodiments, there is no guest state store enable indicator 135. In some embodiments, guest state store logic 185 always stores the contents of global counter enable storage location 121 in guest state storage location 137. In other embodiments, guest state store logic 185 never stores the contents of global counter enable storage location 121 in guest state storage location 137 as part of VM exit processing. In some embodiments, attempts by guest software to access the global counter enable storage location 121 may cause VM exits, either by software convention (e.g., the VMM is required to appropriately set controls in the VMCS to cause VM exits on access to global counter enable storage location 121) or by forcing such a VM exit by the processor.

[0040] In box 266, VM exit logic determines whether to load a host value from host state storage location 138 into global counter enable storage location 121 based on host state load enable indicator 134. If host state store enable indicator 134 is set, then, in box 268, host state load logic 184 causes the host value from host state storage location 138 to be loaded into global counter enable storage location 121; otherwise, box 268 is not performed.

[0041] In box 270, the VM exit is completed and control is transferred to the host. In box 272, the host begins or continues to execute.

[0042] In box 275, counter enable logic 170 determines whether the global counter enable indicator is set. If not, method 200 continues at box 298. If so, method 200 continues at box 280. In box 280, it is determined whether the first event has occurred. If so, then, in box 284, the first performance counter operates as configured, i.e., in this embodiment, where it has been configured in box 210 to count cache misses, the first performance counter increments. In box 290, it is determined whether the second event has occurred. If so, then, in box 294, the second performance counter operates as configured, i.e., in this embodiment, where it has been configured in box 212 to count occurrences of the second event, the second performance counter increments.

[0043] In box 298, it is determined whether a VM entry is to occur. If so, method 200 continues at box 230. If not, method 200 continues at box 299. In box 299, it is determined whether a halt or other such instruction is to be executed. If not, method 200 continues at box 272. If so, method 200 ends.

[0044] Figure 3 illustrates another embodiment of the invention in bare platform hardware 310, which may be used in virtualization architecture 100 or another virtualization architecture. Except as otherwise described, the description of bare platform hardware and its elements also applies to bare platform hardware 310, and its corresponding elements. Bare platform hardware 310 includes processor 320 and memory 330.

[0045] Processor 320 includes counters 371, 373, 375, and 377. Processor 320 also includes counter control storage locations 372, 374, 376, and 378, corresponding to counters 371, 373, 375, and 377, respectively, which each contain a counter valid storage location to store a counter valid indicator to enable or disable the corresponding counter, along with other control or configuration information. Additionally, processor 320 includes counter identifier storage locations 391, 393, 395, and 397, corresponding to counters 371, 373, 375, and 377, respectively. Counter identifier storage locations 391, 393, 395, and 397 may be registers or any other structures of any size, or portions of one or more registers of other structures, for storing information to identify or label counters 371, 373, 375, and 377, respectively. Counter identifier storage locations 391, 393, 395, and 397 may be programmed to store an identifier to uniquely or redundantly (i.e., two

counter identifier storage locations could be programmed with the same value) identify counters 371, 373, 375, and 377, respectively.

[0046] VMCS 332 includes counter identifier field 333, which in this embodiment is the same number of bits as each of counter identifier storage location 391, 393, 395, and 397, but in other embodiments may be any number of bits. Other embodiments may include additional counter identifier fields. Counter identifier field 333 may be programmed, by the VMM that creates VMCS 332 or by any other software, to store a value that may or may not match one or more of the values stored in counter identifier storage locations 391, 393, 395, and 397.

[0047] Processor 320 also includes comparison logic 322, which may be implemented according to any known approach, to generate a counter match signal for each counter based on a comparison of the contents of a corresponding counter identifier storage location and the contents of counter identifier field 333. In this embodiment, the counter match signal for a counter is asserted if the contents of the corresponding counter identifier storage location match the contents of counter identifier field 333. In this embodiment, the counter match signal for a counter is also asserted if the contents of the corresponding counter identifier storage location equals a value of zero, regardless of the contents of counter identifier field 333, to provide for software that does not use the counter identifier feature of the present invention to operate as expected on hardware that does include the counter identifier feature. In another embodiment, the counter match logic may determine a "match" based at least in part on other factors such as the contents of other fields within the VMCS. For example, the value of various control registers such as the CR0, CR3 or CR4 registers in the Intel Architecture may be consulted. Alternatively, the counter match logic may include a determination of whether the software currently executing is the VMM or a guest, and enable the counters only if guest software is currently executing, based on one or more control bits in the VMCS.

[0048] Embodiments may include one or more "counter match enable" control bits or fields to enable or disable the counter match feature, by disabling the comparison logic or by any other desired approach. For example, if a counter match enable bit is set, then the comparison happens as described above. However, if the counter match enable bit is cleared, the counters are always enabled and no comparison is made, so that a "zero" value in a counter identifier storage location does not require the special treatment

described above.

[0049] The counter match signal for each counter is used, along with any other relevant information stored in counter control storage locations 372, 374, 376, and 378, by counter enable logic 370 to control the operation of counters 371, 373, 375, and 377, respectively. If the counter match signal for a counter is asserted, then the counter operates according to its individual control and configuration information. For example, counter 371 may increment on a cache hit if the contents of counter control storage location 372 include an individual counter valid bit set to an "enable" value and an event select field set to count cache hits. However, if the counter match signal for a counter is not asserted, the counter is disabled and does not operate according to its individual control and configuration information. For example, counter 371 would not increment on a cache hit even if the contents of counter control storage location 372 include an individual counter valid bit set to an "enable" value and an event select field set to count cache hits.

[0050] Additionally, processor 320 includes control logic 380 to support virtualization, including the virtualization of counters 371, 373, 375, and 377. Control logic 380 causes processor 320 to execute method embodiments of the present invention, such as the method embodiments described below with reference to Figure 4, for example, by causing processor 320 to include the execution of one or more micro-operations, e.g., to support virtualization, in its response to virtualization instructions or other instructions from a host or guest.

[0051] Figure 4 illustrates an embodiment of the present invention in method 400, another method for virtualizing performance counters. Although method embodiments are not limited in this respect, reference is made to Figures 1 and 3 to describe the method embodiment of Figure 4.

[0052] In box 410 of Figure 4, a performance counter is configured to count occurrences of an event. For example, counter 371 may be configured, by programming counter control storage location 372, to count cache misses. In box 412, counter identifier storage location 391 is programmed with a desired counter identification value.

[0053] In box 420, a host creates a VMCS (e.g., VMCS 332) for a VM. In box 422, the host configures VMCS 332 to implement support for virtualizing counters 371, 373, 375, and 377, including programming counter identifier field 333 with a value equal to the

counter identification value used on box 412. In this embodiment, any number of other counter identifier fields in VMCS 332 or any other VMCSes, corresponding to any number of other guests or hosts, may also be programmed with the same value to allow any number of guests or hosts to share one or more counters.

[0054] In box 430, control of processor 320 is transferred from the host to a guest. In box 432, the guest begins or continues to execute. In box 434, a counter match signal is generated based on a comparison of counter identifier storage location 391 and counter identifier field 333. In this embodiment, the counter match signal is asserted if the contents of the counter identifier storage location match the contents of the counter identifier field.

[0055] In box 440, an occurrence of the event, i.e., in this embodiment, a cache miss, is recognized. In box 442, counter enable logic 370 determines whether to increment the performance counter based on the counter match signal and the contents of the counter valid storage location in the corresponding counter control storage locations (e.g., counter control storage location 372).

[0056] If the counter match signal is asserted and the counter's counter valid indicator is not set, then, in box 444, the performance counter operates as configured, i.e., in this embodiment, where it has been configured in box 410 to count cache misses, the performance counter increments. If the counter match signal is not asserted or the counter's valid indicator is not set, box 444 is not performed.

[0057] Within the scope of the present invention, the methods illustrated in Figures 2 and 4 may be performed in a different order, with illustrated boxes omitted, with additional boxes added, or with a combination of reordered, omitted, or additional boxes. For example, a VMCS may configure a VMCS in any order, e.g., boxes 222 to 226 may be rearranged in any order.

[0058] Processor 120, or any other component or portion of a component designed according to an embodiment of the present invention, may be designed in various stages, from creation to simulation to fabrication. Data representing a design may represent the design in a number of manners. First, as is useful in simulations, the hardware may be represented using a hardware description language or another functional description language. Additionally or alternatively, a circuit level model with logic and/or transistor gates may be produced at some stages of the design process. Furthermore, most designs,

at some stage, reach a level where they may be modeled with data representing the physical placement of various devices. In the case where conventional semiconductor fabrication techniques are used, the data representing the device placement model may be the data specifying the presence or absence of various features on different mask layers for masks used to produce an integrated circuit.

[0059] In any representation of the design, the data may be stored in any form of a machine-readable medium. An optical or electrical wave modulated or otherwise generated to transmit such information, a memory, or a magnetic or optical storage medium, such as a disc, may be the machine-readable medium. Any of these media may "carry" or "indicate" the design, or other information used in an embodiment of the present invention. When an electrical carrier wave indicating or carrying the information is transmitted, to the extent that copying, buffering, or re-transmission of the electrical signal is performed, a new copy is made. Thus, the actions of a communication provider or a network provider may constitute the making of copies of an article, e.g., a carrier wave, embodying techniques of the present invention.

[0060] Thus, apparatuses, methods, and systems for virtualizing performance counters have been disclosed. While certain embodiments have been described, and shown in the accompanying drawings, it is to be understood that such embodiments are merely illustrative and not restrictive of the broad invention, and that this invention not be limited to the specific constructions and arrangements shown and described, since various other modifications may occur to those ordinarily skilled in the art upon studying this disclosure. In an area of technology such as this, where growth is fast and further advancements are not easily foreseen, the disclosed embodiments may be readily modifiable in arrangement and detail as facilitated by enabling technological advancements without departing from the principles of the present disclosure or the scope of the accompanying claims.

What is claimed is:

1. An apparatus comprising:
   a counter;
   a counter enable storage location to store a counter enable indicator;
   counter enable logic to enable the counter based on the counter enable indicator; and
   virtual machine control logic to transfer control of the apparatus to a guest, including guest state load logic to cause a guest value from a virtual machine control structure to be loaded into the counter enable storage location in connection with a transfer of control to the guest.
2. The apparatus of claim 1, wherein the counter is to log performance monitoring information.
3. The apparatus of claim 1, wherein the virtual machine control logic is also to transfer control of the apparatus from the guest to a host, and also includes host state load logic to cause a host value from the virtual machine control structure to be loaded into the counter enable storage location in connection with a transfer of control from the guest to the host.
4. The apparatus of claim 3, wherein the virtual machine control logic also includes guest state store logic to cause the contents of the counter enable storage location to be stored in the virtual machine control structure in connection with the transfer of control from the guest to the host.
5. The apparatus of claim 1, wherein the virtual machine control logic also includes virtual machine entry logic to enable the guest state load logic based on a guest state load enable indicator in the virtual machine control structure.
6. The apparatus of claim 3, wherein the virtual machine control logic also includes virtual machine exit logic to enable the host state load logic based on a host state load enable indicator in the virtual machine control structure.
7. The apparatus of claim 4, wherein the virtual machine control logic also includes virtual machine exit logic to enable the guest state store logic based on a guest state

store enable indicator in the virtual machine control structure.

8. An apparatus comprising:

a counter;

a counter identifier storage location to store a counter identifier;

comparison logic to generate a counter match signal based on a comparison of the counter identifier to the contents of a counter identifier field from a virtual machine control structure; and

enable logic to generate a counter enable signal to enable the counter, wherein the counter enable signal is based on the counter match signal.

9. The apparatus of claim 8, further comprising a counter valid storage location to store a counter valid indicator, wherein the counter enable signal is also based on the counter valid indicator.

10. A method comprising:

configuring a first counter in a processor to count occurrences of a first event;

configuring a second counter in the processor to count occurrences of a second event;

initiating a transfer of control of the processor to a guest;

loading a guest value from a virtual machine control structure into a counter enable storage location;

completing the transfer of control of the processor to the guest;

determining whether to change the count of the first counter based on the contents of the counter enable storage location and a first occurrence of the first event; and

determining whether to change the count of the second counter based on the contents of the counter enable storage location and a first occurrence of the second event.

11. The method of claim 10, further comprising:

initiating a transfer of control of the processor from the guest to a host;

loading a host value from the virtual machine control structure into the counter enable storage location;

completing the transfer of control of the processor from the guest to the host;

determining whether to change the count of the first counter based on the contents

of the counter enable storage location and a second occurrence of the first event; and

determining whether to change the count of the second counter based on the contents of the counter enable storage location and a second occurrence of the second event.

12. The method of claim 11, further comprising storing the contents of the counter enable storage location in the virtual machine control structure after initiating the transfer of control of the processor from the guest to the host.

13. The method of claim 10, further comprising enabling the processor to load the guest value based on a guest state load enable indicator in the virtual machine control structure.

14. The method of claim 11, further comprising enabling the processor to load the host value based on a host state load enable indicator in the virtual machine control structure.

15. The method of claim 11, further comprising enabling the processor to store the contents of the counter enable storage location based on a guest state store enable indicator in the virtual machine control structure.

16. A method comprising:

configuring a counter in a processor to count occurrences of an event;

transferring control of the processor to a guest;

recognizing an occurrence of the event;

generating a match signal based on a comparison of the contents of a counter identifier storage location in the processor to the contents of a counter identifier field in a virtual machine control structure; and

determining whether to change the count of the counter based on the match signal.

17. The method of claim 16, wherein determining whether to change the count of the counter is also based on a counter valid indicator.

18. A system comprising:

a memory to store a data structure to control a virtual machine; and

a processor including:

a plurality of counters;

a counter enable storage location to store a counter enable indicator;

counter enable logic to enable the plurality of counters based on the counter enable indicator; and

virtual machine control logic to transfer control of the processor to a guest, including guest state load logic to cause a guest value from the data structure to be loaded into the counter enable storage location in connection with a transfer of control to the guest.

19. The system of claim 18, wherein the plurality of counters is to log performance monitoring information.

20. The system of claim 18, wherein the memory is dynamic random access memory.

FIGURE 1

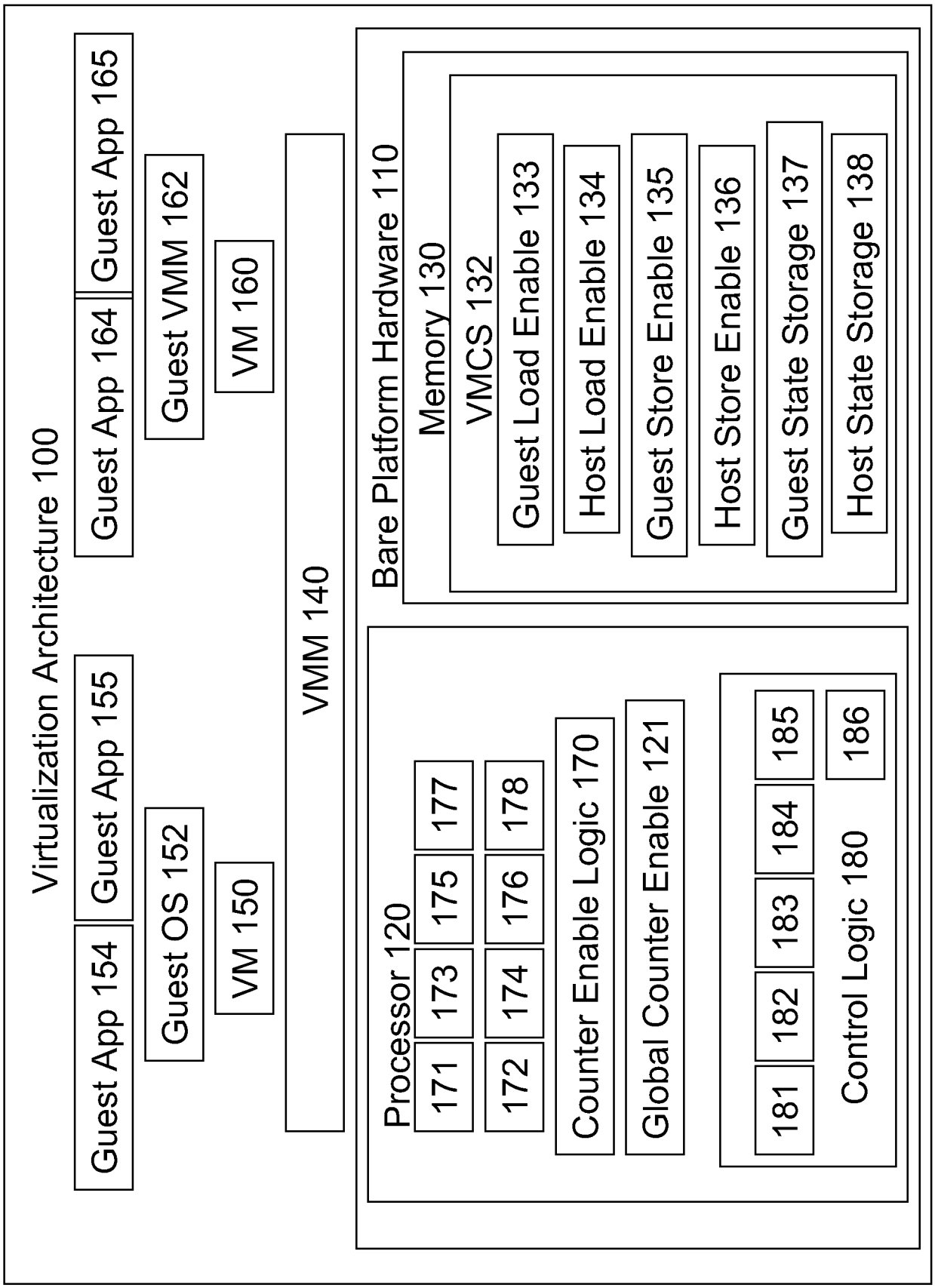

Virtualization Architecture 100
Guest App 154
Guest App 155
Guest OS 152
VM 150
Guest App 164
Guest App 165
Guest VMM 162
VM 160
VMM 140
Bare Platform Hardware 110
Memory 130
VMCS 132
Guest Load Enable 133
Host Load Enable 134
Guest Store Enable 135
Host Store Enable 136
Guest State Storage 137
Host State Storage 138
Processor 120
171
173
175
177
172
174
176
178
Counter Enable Logic 170
Global Counter Enable 121
181
182
183
184
185
Control Logic 180
186

FIGURE 2
METHOD 200

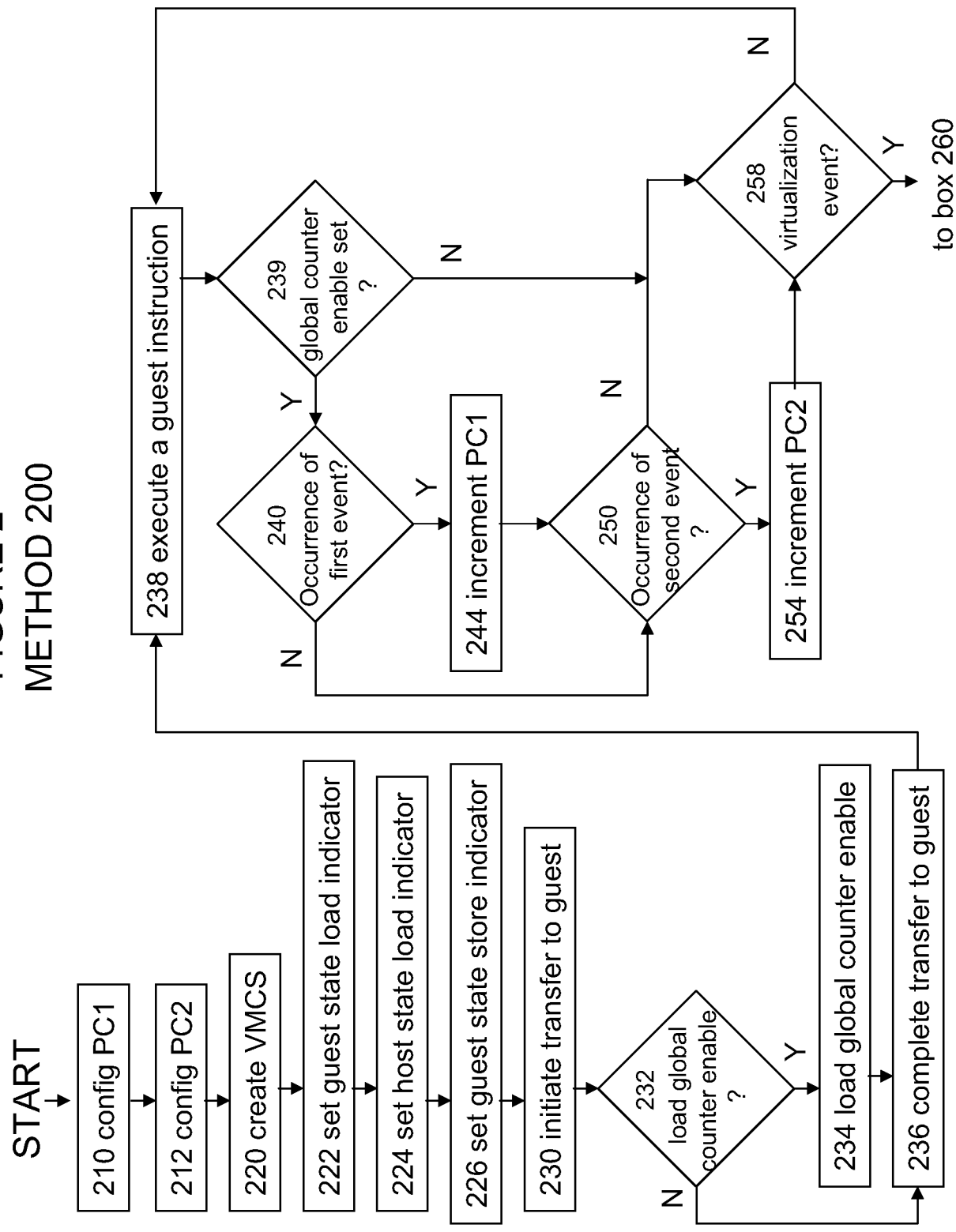


FIGURE 2 (continued)
METHOD 200 (continued)

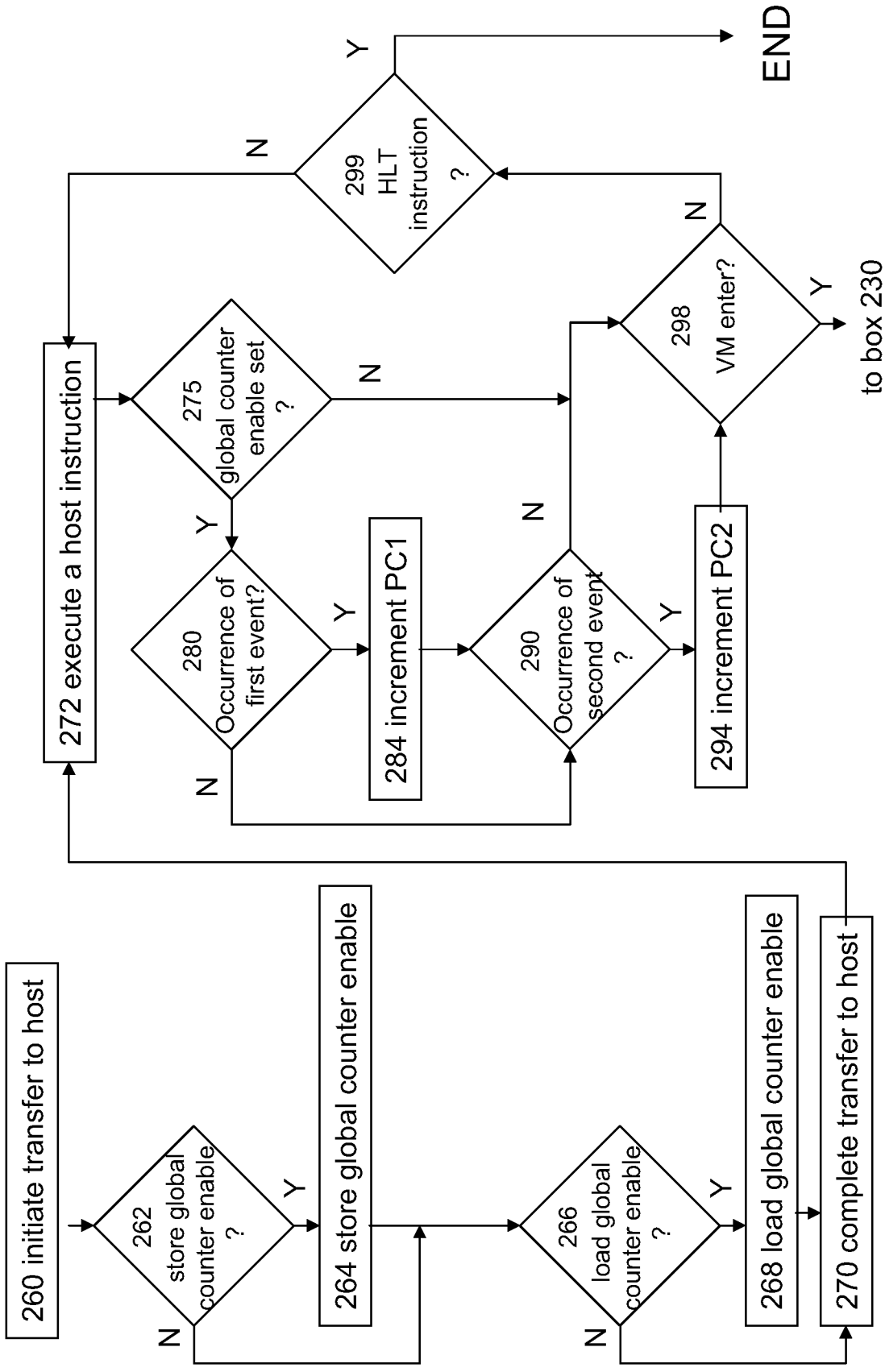


FIGURE 3

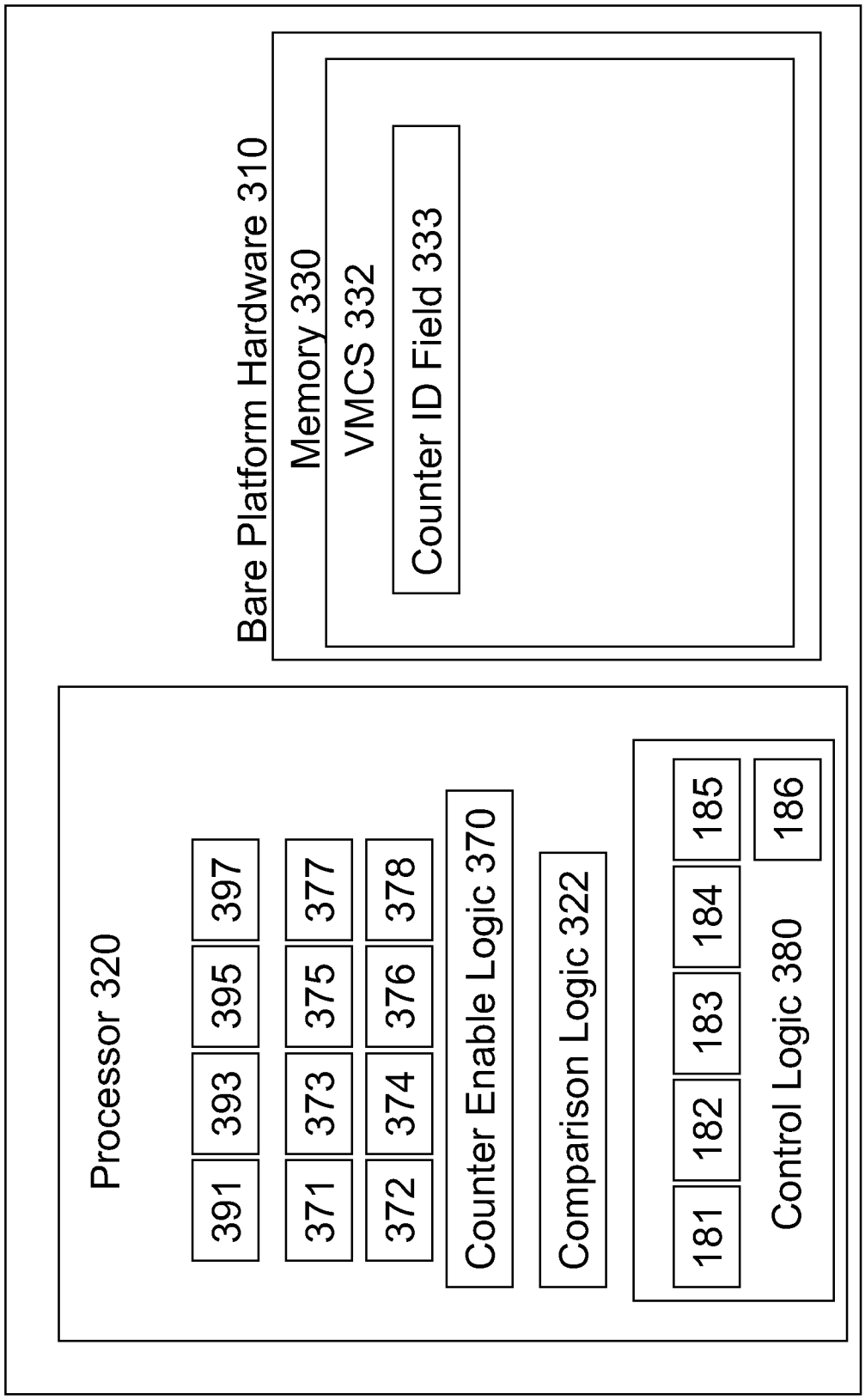
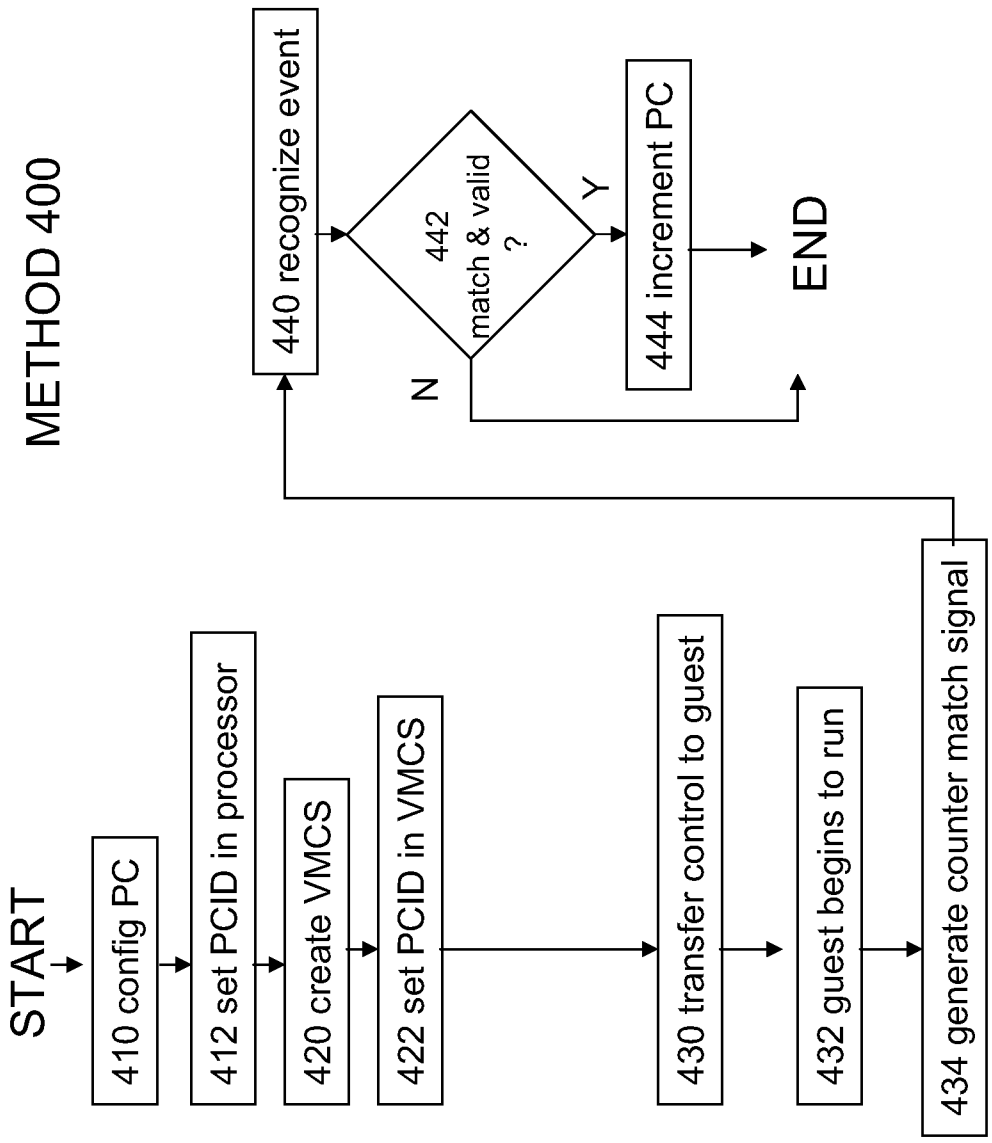

Bare Platform Hardware 310
Memory 330
VMCS 332
Counter ID Field 333
Processor 320
391
393
395
397
371
373
375
377
372
374
376
378
Counter Enable Logic 370
Comparison Logic 322
181
182
183
184
185
Control Logic 380
186

FIGURE 4
METHOD 400



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2007/075286

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/06(2006.01)i, G06F 9/54(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)

IPC 8 G06F, H04Q

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 since 1975

Japanese Utility models and applications for Utility models since 1975

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

eKIPASS(KIPO internal) "Keywords: VM, VMM, performance counter, switching and similar terms"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y A	US 2003/0037089 A1 (COTA-ROBLES, E., et al.) 20 February 2003 See the abstract, figures 1A-3 and 7-8B, and paragraphs [0033]-[0042].	1, 3-4, 10-12, 18, 20 2, 5-9, 13-17, 19
Y A	US 2003/0217250 A1 (BENNETT, S., et al.) 20 November 2003 See the abstract, figures 1-3, and paragraph [0033]-[0039].	1, 3-4, 10-12, 18, 20 2, 5-9, 13-17, 19
A	KR 2006071307 A (MICROSOFT CORP.) 26 June 2006 See the abstract, figures 3a-6, and pages 2-3.	1-20
A	US 7,069,413 B1 (AGESEN, O., et al.) 27 June 2006 See the abstract, figures 5-6 and 9, and col.11 line 21 - col.14 line 14.	1-20
A	US 4,975,836 A (HIROSAWA, T., et al.) 04 December 1990 See the abstract, figures 1 and 3, and col.3 line 41 - col.4 line 51.	1-20

☐ 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	Date of mailing of the international search report
28 DECEMBER 2007 (28.12.2007)	**28 DECEMBER 2007 (28.12.2007)**

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
920 Dunsan-dong, Seo-gu, Daejeon 302-701,
Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

NHO, Ji Myong

Telephone No. 82-42-481-8528

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2007/075286

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 20030037089 A1	20.02.2003	US 07191440 B2	13.03.2007
US 2003217250 A1	20.11.2003	AU 2003224791 A1	03.11.2003
		CN 1659518 A	24.08.2005
		EP 1520227 A2	06.04.2005
		KR 1020040101516 A	02.12.2004
		TW 252427 B	01.04.2006
		US 7127548 B2	24.10.2006
		WO 2003090070 A2	30.10.2003
		WO 2003090070 A3	27.01.2005
KR 2006071307 A	26.06.2006	CN 1794177 A	28.06.2006
		EP 01674987 A2	28.06.2006
		EP 01674987 A3	24.10.2007
		JP 2006178933 A2	06.07.2006
		US 2006136653 A1	22.06.2006
US 7069413 B1	27.06.2006	None	
US 4975836 A	04.12.1990	DE 3582662 C0	29.05.1991
		EP 0185378 B1	24.04.1991
		EP 0185378 A2	25.06.1986
		EP 0185378 A3	23.12.1987
		JP 61145646 A2	03.07.1986
		JP 61204743 A2	10.09.1986
		JP 61208133 A2	16.09.1986
		JP 7054468 B4	07.06.1995