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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 May 2006 (11.05.2006)

PCT

(10) International Publication Number
WO 2006/050534 A1

(51) International Patent Classification:
G06F 9/455 (2006.01)

(21) International Application Number:
PCT/US2005/040450

(22) International Filing Date: 31 October 2005 (31.10.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/976,970 29 October 2004 (29.10.2004) US

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

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **BENNETT, Steven** [US/US]; 6469 Se Sigrid Street, Hillsboro, OR 97123 (US). **NEIGER, Gilbert** [US/US]; 2424 NE 11th Avenue, Portland, OR 97212 (US). **ANDERSON, Andrew** [US/US]; 677 SE 68th Ave, Hillsboro, OR 97123 (US).

(74) Agents: **VINCENT, Lester, J.** et al.; Blakely Sokoloff Taylor & Zafman, 12400 Wilshire Boulevard, 7th Floor, Los Angeles, CA 950025 (US).

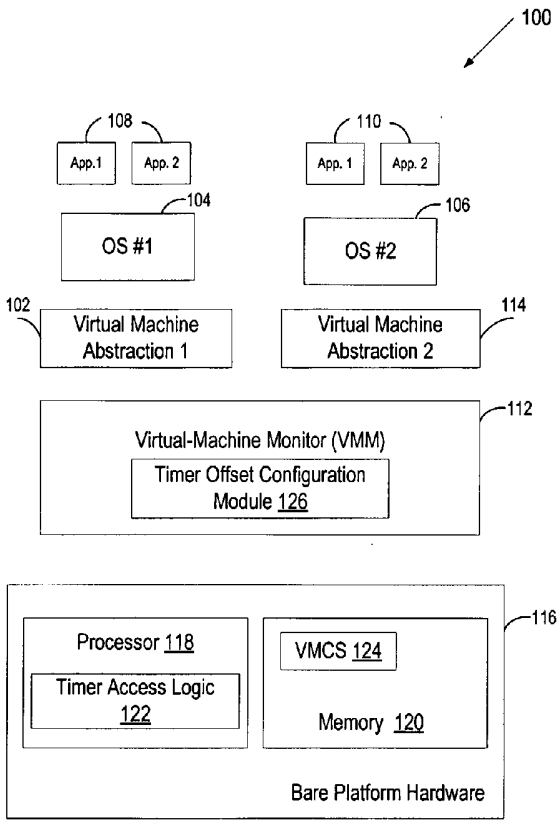
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC,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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,

[Continued on next page]

(54) Title: TIMER OFFSETTING MECHANISM IN A VIRTUAL MACHINE ENVIRONMENT

(57) Abstract: In one embodiment, a method includes receiving a request to transition control to a virtual machine (vm) from virtual machine monitor (vmm), calculating an offset value, receiving, during operation of the vm, a request for a current value of the timer, adjusting the current value of the time based on the offset value, and providing the adjusted timer value to the vm.



WO 2006/050534 A1

ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *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*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

TIMER OFFSETTING MECHANISM IN A VIRTUAL MACHINE ENVIRONMENT

Field

[0001] Embodiments of the invention relate generally to virtual machines, and more specifically to a timer offsetting mechanism in a virtual machine environment.

Background

[0002] Timers are typically used by operating systems and application software to schedule activities. For example, an operating system kernel may use a timer to allow a plurality of user-level applications to time-share the resources of the system (e.g., the central processing unit (CPU)). An example of a timer used on a personal computer (PC) platform is the 8254 Programmable Interval Timer. This timer may be configured to issue interrupts after a specified interval or periodically.

[0003] Another example of a timer is the timestamp counter (TSC) used in the instruction set architecture (ISA) of the Intel® Pentium® 4 (referred to herein as the IA-32 ISA). The TSC is a 64-bit counter that is set to 0 following the hardware reset of the processor, and then incremented every processor clock cycle, even when the processor is halted by the HLT instruction. The TSC cannot be used to generate interrupts. It is a time reference only, useful to measure time intervals. The IA-32 ISA provides an instruction (RDTSC) to read the value of the TSC and an instruction (WRMSR) to write the TSC. When WRMSR is used to write the time-stamp counter, only the 32 low-order bits may be written; the high-order 32 bits are cleared to 0. Because of these writing restrictions, software generally cannot set the TSC forward nor set it backwards to an arbitrary value.

[0004] In multi-processor (MP) systems, the TSC is used to properly synchronize processors and schedule processes appropriately. At boot time, the values of the TSC on

all processors are synchronized and most operating systems assume that the TSC then count at the same rate. If the TSC values drift between processors (e.g., if one processor counts at a different rate that the others), scheduling of processes by the operating system may be confused.

Brief Description of the Drawings

[0005] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0006] **Figure 1** illustrates one embodiment of a virtual-machine environment, in which the present invention may operate;

[0007] **Figure 2** is a flow diagram of one embodiment of a process for controlling access of VMs to a timer;

[0008] **Figure 3** is a flow diagram of one embodiment of a process for configuring fields associated timer offsetting; and

[0009] **Figures 4 and 5** are flow diagrams of two alternative embodiments of a process for calculating a timer offset value for a VM.

Description of Embodiments

[0010] A method and apparatus for controlling access of virtual machines to a timer is described. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention can be practiced without these specific details.

[0011] Some portions of the detailed descriptions that follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer system's registers or memory. These algorithmic descriptions and

representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0012] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or the like, may refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer-system memories or registers or other such information storage, transmission or display devices.

[0013] In the following detailed description of the embodiments, reference is made to the accompanying drawings that show, by way of illustration, specific embodiments in which the invention may be practiced. In the drawings, like numerals describe substantially similar components throughout the several views. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention. Other embodiments may be utilized and structural, logical, and electrical changes may be

made without departing from the scope of the present invention. Moreover, it is to be understood that the various embodiments of the invention, although different, are not necessarily mutually exclusive. For example, a particular feature, structure, or characteristic described in one embodiment may be included within other embodiments. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims, along with the full scope of equivalents to which such claims are entitled.

[0014] Although the below examples may describe controlling access of virtual machine to a timer in the context of execution units and logic circuits, other embodiments of the present invention can be accomplished by way of software. For example, in some embodiments, the present invention may be provided as a computer program product or software which may include a machine or computer-readable medium having stored thereon instructions which may be used to program a computer (or other electronic devices) to perform a process according to the present invention. In other embodiments, processes of the present invention might be performed by specific hardware components that contain hardwired logic for performing the processes, or by any combination of programmed computer components and custom hardware components.

[0015] Thus, a machine-readable medium may include any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer), but is not limited to, floppy diskettes, optical disks, Compact Disc, Read-Only Memory (CD-ROMs), and magneto-optical disks, Read-Only Memory (ROMs), Random Access Memory (RAM), Erasable Programmable Read-Only Memory (EPROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), magnetic or optical cards, flash memory, a transmission over the Internet, electrical, optical, acoustical or other forms of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.) or the like.

[0016] Further, a design may go through 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, 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 of data representing the physical placement of various devices in the hardware model. In the case where conventional semiconductor fabrication techniques are used, data representing a hardware model may be the data specifying the presence or absence of various features on different mask layers for masks used to produce the integrated circuit. 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 such as a disc may be the machine readable medium. Any of these mediums may "carry" or "indicate" the design or software information. When an electrical carrier wave indicating or carrying the code or design is transmitted, to the extent that copying, buffering, or re-transmission of the electrical signal is performed, a new copy is made. Thus, a communication provider or a network provider may make copies of an article (a carrier wave) embodying techniques of the present invention.

[0017] **Figure 1** illustrates one embodiment of a virtual-machine environment 100, in which the present invention may operate. In this embodiment, bare platform hardware 116 comprises a computing platform, which may be capable, for example, of executing a standard operating system (OS) or a virtual-machine monitor (VMM), such as a VMM 112.

[0018] The VMM 112, though typically implemented in software, may emulate and export a bare machine interface to higher level software. Such higher level software may comprise a standard or real-time OS, may be a highly stripped down operating environment with limited operating system functionality, may not include traditional OS facilities, etc. Alternatively, for example, the VMM 112 may be run within, or on top of, another VMM. VMMs may be implemented, for example, in hardware, software, firmware or by a combination of various techniques.

[0019] The platform hardware 116 can be of a personal computer (PC), mainframe, handheld device, portable computer, set-top box, or any other computing system. The platform hardware 116 includes a processor 118 and memory 120.

[0020] Processor 118 can be any type of processor capable of executing software, such as a microprocessor, digital signal processor, microcontroller, or the like. The processor 118 may include microcode, programmable logic or hardcoded logic for performing the execution of method embodiments of the present invention. Although **Figure 1** shows only one such processor 118, there may be one or more processors in the system.

[0021] Memory 120 can be a hard disk, a floppy disk, random access memory (RAM), read only memory (ROM), flash memory, any combination of the above devices, or any other type of machine medium readable by processor 118. Memory 120 may store instructions and/or data for performing the execution of method embodiments of the present invention.

[0022] The VMM 112 presents to other software (i.e., "guest" software) the abstraction of one or more virtual machines (VMs), which may provide the same or different abstractions to the various guests. **Figure 1** shows two VMs, 102 and 114. The guest software running on each VM may include a guest OS such as a guest OS 104 or

106 and various guest software applications 108 and 110. Each of the guest OSs 104 and 106 expects to access physical resources (e.g., processor registers, memory and I/O devices) within the VMs 102 and 114 on which the guest OS 104 or 106 is running and to perform other functions. For example, the guest OS 104 and 106 expects to have access to all registers, caches, structures, I/O devices, memory and the like, according to the architecture of the processor and platform presented in the VMs 102 and 114. The resources that can be accessed by the guest software may either be classified as "privileged" or "non-privileged." For privileged resources, the VMM 112 facilitates functionality desired by guest software while retaining ultimate control over these privileged resources. Non-privileged resources do not need to be controlled by the VMM 112 and can be accessed by guest software.

[0023] Further, each guest OS expects to handle various fault events such as exceptions (e.g., page faults, general protection faults, etc.), interrupts (e.g., hardware interrupts, software interrupts), and platform events (e.g., initialization (INIT) and system management interrupts (SMIs)). Some of these fault events are "privileged" because they must be handled by the VMM 112 to ensure proper operation of VMs 102 and 114 and for protection from and among guest software.

[0024] When a privileged fault event occurs or guest software attempts to access a privileged resource, control may be transferred to the VMM 112. The transfer of control from guest software to the VMM 112 is referred to herein as a VM exit. After facilitating the resource access or handling the event appropriately, the VMM 112 may return control to guest software. The transfer of control from the VMM 112 to guest software is referred to as a VM entry.

[0025] In one embodiment, the processor 118 controls the operation of the VMs 102 and 114 in accordance with data stored in a virtual machine control structure (VMCS)

124. The VMCS 124 is a structure that may contain state of guest software, state of the VMM 112, execution control information indicating how the VMM 112 wishes to control operation of guest software, information controlling transitions between the VMM 112 and a VM, etc. The processor 118 reads information from the VMCS 124 to determine the execution environment of the VM and to constrain its behavior. In one embodiment, the VMCS is stored in memory 120. In some embodiments, multiple VMCS structures are used to support multiple VMs.

[0026] The VMM 112 may need to use a timer to schedule resources, provide quality of service, assure security, and perform other functions. For example, in the instruction set architecture (ISA) of the Intel® Pentium® 4 (referred to herein as the IA-32 ISA), the VMM 112 may use the timestamp counter (TSC) to perform these functions. Each of the VMs 102 and 114 may also need to use the timer to calibrate timing loops and do performance optimization. Because the VMs 102 and 114 have no knowledge of each other or the VMM 112, the timer's values provided to the VM 102 or 114 may need to be adjusted to present the illusion that the guest OS 104 or 106 is running on a dedicated hardware platform, not a virtual platform. A timer offsetting mechanism is provided to properly virtualize the timer and thus preserve the illusion for the guest OSes 104 and 106. In one embodiment, the timer offsetting mechanism includes a timer offset configuration module 126 and timer access logic 122.

[0027] The timer offset configuration module 126 is responsible for providing values for fields associated with timer offsetting prior to requesting a transition of control to the VM 102 or 114. In one embodiment, these values may include an offset value specifying an offset to be used by the processor 118 when providing timer values to the VM 102 or 114 and a timer offsetting indicator specifying whether timer offsetting is enabled for the VM 102 or 114. In an embodiment, the timer offset value is a signed

value, allowing the VMM 112 to present a timer value to guest software that is less than or greater than the actual hardware timer value. In one embodiment, the timer offset value is added to the timer value before returning the value to the VM 102 or 114. In one embodiment, the fields associated with timer offsetting also include a timer access control indicator specifying whether VM requests to access the timer are associated with a transition of control to the VMM (e.g., whether VM requests to access the timer should cause VM exits).

[0028] In one embodiment, the offset value for the VM 102 or 114 accounts for the aggregation of time intervals during which this VM was not running due to the execution of the VMM 112 and the other VM. For example, suppose that, when the timer's value is 1000 ticks, the VM 102 expects the value of the timer to be 1000. Then, at 1500 ticks, the VM 102 may be interrupted by a VM exit, following by the execution of the VMM 112 for 100 ticks (timer value of 1600 ticks), after which the VMM 112 may request to enter the VM 114. The VM 114 may then execute for 600 ticks (timer value of 2200 ticks) until a VM exit, resulting in the execution of the VMM 112 for 200 ticks (timer value 2400), which may then request to re-enter the VM 102. At the time of re-entry, the VM 102 expects the timer to have the value of 1500 ticks. Instead, the actual timer value at this time is 2400 ticks. The offset value provided by the VMM 112 for the VM 102 will be 900 ticks, which is the aggregation of time intervals during which the VM 112 was not running due to the execution of the VM 114 and the VMM 112. Hence, the VMM 112 would store the offset value of 900 to the timer offset field so that when the VM 102 attempts to read the timer, the value provided to the VM 102 will be computed by reducing the current timer value by 900. The offset value may also be calculated by computing the difference between the actual timer value and the timer value expected by the VM 102. In an alternative embodiment, the value provided to the VM 102 when it

attempts to read the timer is computed by adding the current timer value to the offset value configured by the VMM 112. In this embodiment, the offset value stored by the VMM is a negative number. In the example described above, the value stored is -900.

[0029] As discussed above, in one embodiment, the offset value is determined by the VMM 112. In an alternative embodiment, the offset value is determined by the processor 118, and the fields and controls associated with timer offsetting may include a timer offsetting indicator, an adjust offset indicator, a guest timer field, a save timer indicator, and a timer offset field. In one embodiment, the three indicator values may be combined in a variety of ways. For example, the adjust offset indicator and the save timer indicator may be the same control (i.e., enabling the adjustment of the timer offset implicitly enables the saving of the timer), as will be described in more detail below. In some embodiments, some of the above indicators may not be present. For example, the timer offsetting indicator may not be present, and timer offsetting is assumed to be always enabled, as will be discussed in greater detail below.

[0030] In one embodiment, the fields and controls associated with timer offsetting are stored in the VMCS 124. Alternatively, the fields and controls associated with timer offsetting may reside in the processor 118, a combination of the memory 120 and the processor 118, or in any other storage location or locations. In one embodiment, separate fields and controls associated with timer offsetting are maintained for each of the VMs 102 and 114. Alternatively, the same fields and controls associated with timer offsetting are maintained for both VMs 102 and 144 and are updated by the VMM 112 before each VM entry.

[0031] In one embodiment, in which the system 100 includes multiple processors, multiple cores or multi thread processors, each of the multiple logical processors is associated with separate fields and controls associated with timer offsetting, and the VMM

112 configures the fields and controls associated with timer offsetting for each of the multiple logical processors.

[0032] In one embodiment, the processor 118 includes timer access logic 122 that is responsible for virtualizing accesses of the VM 102 and 114 to the timer based on the timer offsetting values. In particular, if the timer access logic 122 determines that timer offsetting is enabled, it provides an adjusted timer value to the VM 102 or 114. In one embodiment, the timer access logic 122 determines if timer offsetting is enabled by examining a timer offsetting indicator value. In one embodiment, when the timer access logic 122 receives a request for a current value of the timer from the VM 102 or 114, it reads the current value of the timer, adds the offset value to the current value of the timer, and returns the resulting value to the VM 102 or 114, thus presenting the VM 102 or 114 with the illusion that it is running on a dedicated hardware platform. In an embodiment, the offset value is a signed value.

[0033] The offset values may be determined for the VM 102 and 114 by the VMM 112 (e.g., by the timer offset configuration module 126). One embodiment of a process for determining an offset value by the VMM 112 will be discussed in greater detail below in conjunction with **Figures 3 and 4**. Alternatively, the offset values are determined by the processor 118 (e.g., by the timer access logic 122). One embodiment of a process for determining an offset value by the processor 118 will be discussed in more detail below in conjunction with **Figure 5**.

[0034] **Figure 2** is a flow diagram of one embodiment of a process 200 for controlling access of VMs to a timer. The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (such as that run on a general purpose computer system or a

dedicated machine), or a combination of both. In one embodiment, process 200 is performed by timer access logic 122 of **Figure 1**.

[0035] Referring to **Figure 2**, process 200 begins with processing logic receiving a request to transition control to a VM from a VMM (i.e., the request for VM entry) (processing block 202). In one embodiment, the VM entry request is received via a VM entry instruction executed by the VMM.

[0036] Next, processing logic determines whether timer offsetting is enabled (processing block 204). In one embodiment, processing logic makes this determination as part of the transition to the VM (e.g., when checking and loading VM state and execution control information stored in the VMCS). In one embodiment, the determination is based on the current value of a timer offsetting indicator stored in the VMCS for the VM being entered.

[0037] If timer offsetting is enabled, processing logic uses a timer offset value when responding to the requests of the VM for a current timer value. In one embodiment, the timer offset value is determined by the VMM prior to issuing the request to transition control to this VM. One embodiment of a process for determining the offset value by the VMM will be discussed in greater detail below in conjunction with **Figures 3 and 4**. Alternatively, the offset value is automatically determined by the processor when transitioning control to this VM. One embodiment of a process for determining the offset value by the processor will be discussed in more detail below in conjunction with **Figure 5**.

[0038] In one embodiment, if timer offsetting is enabled, processing logic loads an offset register with the timer offset value stored in the VMCS (processing block 206). Alternatively, if timer offsetting is disabled, processing logic loads the offset register with

0 (processing block 206). Next, processing logic begins the execution in the VM (processing block 210).

[0039] During the VM's execution, processing logic may receive the VM's requests for a current value of the timer. In the IA-32 ISA, for example, the VM may issue a request for a current value of the TSC by executing the RDMSR instruction or a RDTSC instruction to read the TSC.

[0040] At processing box 212, processing logic determines whether the VM requests a current value of the timer. If so, then in one embodiment, processing logic determines whether this request is associated with a transition of control to the VMM (processing box 214). In one embodiment, a timer access control indicator may be set to an "exit" value to cause a VM exit on each request of the VM to access the timer. For example, in one embodiment, the timer access control indicator is a single bit, which if set to 1 indicates that requests of a VM to access the timer cause VM exits. In one embodiment, this indicator may be stored in the VMCS. If the request is not associated with a transition of control to the VMM, processing logic proceeds to processing block 216. If the request is associated with a transition of control to the VMM, processing logic transitions control to the VMM, indicating to the VMM that the VM exit was caused by an attempt to access the timer (processing block 220). In one embodiment, prior to transitioning control to the VMM, processing logic loads the offset register with 0 (processing block 218) to allow the VMM to obtain the actual value of the timer.

[0041] In another embodiment, a timer access control indicator is not used, and processing logic does not check for a transition of control to the VMM in response to VM requests for a current value of the timer. Instead, processing logic skips processing block 214 and proceeds directly to processing block 216.

[0042] At processing block 216, processing logic reads the current value of the timer, adds the offset value to the current value of the timer, and returns the result to the VM. In one embodiment, the timer offset value is a signed value that is combined with the content of the timer using signed addition.

[0043] During the execution of the VM, various other events associated with a VM exit (e.g., page faults, interrupts, etc.) may occur. In one embodiment, if processing logic detects an event associated with a VM exit (processing box 222), processing logic loads the offset register with 0 (processing block 218) and transitions control to the VMM, indicating the cause of the VM exit detected in processing block 222 (processing block 220). If processing logic does not detect any events associated with a VM exit, processing logic returns to processing box 212.

[0044] **Figure 3** is a flow diagram of one embodiment of a process 300 for configuring fields associated with timer offsetting. The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (such as that run on a general purpose computer system or a dedicated machine), or a combination of both. In one embodiment, process 300 is performed by a timer offset configuration module 126 of **Figure 1**.

[0045] Referring to **Figure 3**, process 300 begins with processing logic deciding that a transition of control to a VM is needed. Prior to issuing a request to transition control to the VM, processing logic determines a timer offset value for the VM (processing block 302) and stores the timer offset value in the VMCS (processing block 304). In one embodiment, the timer offset value is the aggregation of time intervals during which the VM being entered was not running due to the execution of the VMM and the other VMs. One embodiment of a process for calculating the timer offset value will be discussed in more detail below in conjunction with **Figure 4**.

[0046] Next, processing logic sets a timer offset indicator to an enabling value (processing block 306) and issues a request to transition control to the VM (e.g., a VM entry request) (processing logic 308).

[0047] Subsequently, when a VM exit from the VM is generated, processing logic receives control back (processing block 310), determines the current value of the timer (processing block 312), and handles the VM exit as needed (e.g., perform operations addressing the cause of the VM exit) (processing block 314). As described below, the value of the timer at VM exit may be used to calculate the timer offset before returning control to the VM.

[0048] **Figure 4** is a flow diagram of one embodiment of a process 400 for calculating a timer offset value for a VM (as referenced in processing block 302 of **Figure 3** for example). The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (such as that run on a general purpose computer system or a dedicated machine), or a combination of both. In one embodiment, process 400 is performed by a timer offset configuration module 126 of **Figure 1**.

[0049] Referring to **Figure 4**, process 400 begins with processing logic calculating the time spent in this VM since the last VM entry (processing block 402). In one embodiment, this time is calculated by determining a VM-entry time (i.e., the timer value just prior to issuing a request to enter the VM) and a VM-exit time (e.g., the timer value at the time of receiving control returned from the VM), and subtracting the VM-entry time from the VM-exit time.

[0050] At processing block 404, processing logic calculates the cumulative time spent in the VM by adding the time spent in the VM during the last entry to the previously calculated cumulative time. In one embodiment, processing logic calculates the

cumulative time spent in the VM when receiving control returned from the VM. Alternatively, processing logic calculates the cumulative time spent in the VM when issuing a request to return control to this VM.

[0051] When processing logic decides to return control to the VM, it reads the current value of the timer (processing block 406), and calculates the timer offset value as a difference between the cumulative time spent in the VM and the current value of the timer (processing block 408). In one embodiment, the timer offset value is a signed value.

[0052] **Figure 5** is a flow diagram of an alternative embodiment of a process for calculating a timer offset value for a VM. The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (such as that run on a general purpose computer system or a dedicated machine), or a combination of both. In one embodiment, process 500 is performed by timer offset logic 122 of **Figure 1**.

[0053] Referring to **Figure 5**, process 500 begins with processing logic detecting an event associated with a transition of control from VM1 to the VMM (processing block 502).

[0054] At processing block 504, processing logic determines whether a save timer indicator is enabled. In one embodiment, the save timer indicator is configured by the VMM and stored in the VMCS.

[0055] If the save timer indicator is enabled, processing logic saves a current timer value as a VM1 timer value to a guest timer field (processing block 506) and proceeds to processing block 508. In one embodiment, the guest timer field is stored in the VMCS.

[0056] If the save timer indicator is disabled, processing logic skips processing block 506 and proceeds directly to processing block 508.

[0057] At processing block 508, processing logic transitions control to the VMM.

[0058] Subsequently, at processing block 510, processing logic receives a request to return control to VM1. In response, processing logic determines whether an adjust offset indicator is enabled (processing block 512). In one embodiment, the adjust offset indicator is configured by the VMM and stored in the VMCS. In another embodiment, the save timer indicator and the adjust offset indicator are represented by the same indicator which is checked both at processing blocks 504 and 512. In another embodiment, a timer offsetting indicator may be evaluated to determine if timer offsetting should be used. In one embodiment, the timer offsetting indicator is configured by the VMM and stored in the VMCS.

[0059] If the adjust offset indicator is enabled, processing logic reads the current timer value (processing block 514), and determines the difference between the current timer value and the VM1 timer value saved at processing block 506 by subtracting the saved VM1 timer value from the current timer value (processing block 516). Further, processing logic calculates the adjusted timer offset value by subtracting this difference from the timer offset value in the timer offset field (processing block 518). In one embodiment, this adjusted timer offset value is stored to the timer offset field. Next, processing logic transitions control to VM1 (processing block 520).

[0060] If the adjust offset indicator is disabled, in one embodiment, processing logic skips processing blocks 514 through 518 and proceeds directly to processing block 520. In another embodiment, if the adjust offset indicator is disabled, processing logic retrieves the value of timer offset field for use as the adjusted timer offset value and then proceeds to processing block 520.

[0061] In an alternative embodiment, at VM exit, a virtual guest timer value is calculated (by computing the sum of the current timer offset value and the current value of the timer) and stored to a virtual guest timer field. In one embodiment, the virtual guest

timer field is stored in the VMCS. Subsequently, at VM entry, the offset value is computed by subtracting the timer value at the time of the VM entry from the virtual guest timer value. While the VM executes, attempts to read the timer will return the current value of the timer adjusted by the offset value.

[0062] Thus, a method and apparatus for controlling access of VMs to a timer have been described. It is to be understood that the above description is intended to be illustrative, and not restrictive. Many other embodiments will be apparent to those of skill in the art upon reading and understanding the above description. The scope of the invention should, therefore, be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

CLAIMS

What is claimed is:

1. A method comprising:

receiving a request to transition control to a virtual machine (VM) from a virtual machine monitor (VMM);

calculating an offset value;

receiving, during operation of the VM, a request for a current value of a timer;

adjusting the current value of the timer based on the offset value; and

providing the adjusted timer value to the VM.

2. The method of claim 1 further comprising:

transitioning control to the VM upon receiving the request from the VMM, the transitioning comprising calculating the offset value.

3. The method of claim 1 wherein calculating the offset value comprises:

determining a difference between a timer value read upon receiving the request to transition control to the VM and a timer value saved upon detecting a prior event associated with a transition of control from the VM to the VMM; and

subtracting the difference from a value of a timer offset field.

4. The method of claim 3 further comprising storing the calculated timer offset value to the timer offset field.

5. The method of claim 3 further comprising:

determining that an adjust timer offset indicator is set to an enabled value prior to determining the difference.

6. The method of claim 1 further comprising:

determining that an adjust timer offset indicator is set to a disabled value; and

reading a timer offset field, a value of the timer offset field being subsequently used as the offset value.

7. The method of claim 3 wherein the timer value is saved upon detecting the prior event associated with the transition of control to the VMM if a save timer indicator is enabled.

8. The method of claim 1 further comprising:

determining that a timer access control indicator is set to an exit value; and

transitioning control to the VMM in response to the request.

9. The method of claim 1 further comprising:

determining that offsetting of the timer is disabled; and

loading an offset register with zero.

10. An apparatus comprising:

a virtual machine monitor (VMM);

a data structure controlled by the VMM, the data structure storing an offset value for a virtual machine (VM); and

timer access logic to calculate the offset value upon receiving a request to transition control to the VM from the VMM, and to provide a value of the timer to the VM during an operation of the VM, the value of the timer being adjusted based on the offset value.

11. The apparatus of claim 10 wherein the timer access logic is further to transition control to the VM upon receiving the request from the VMM, the transitioning comprising calculating the offset value.

12. The apparatus of claim 10 wherein the timer access logic is to calculate the offset value by determining a difference between a timer value read upon receiving the request to transition control to the VM and a timer value saved upon detecting a prior event associated with a transition of control from the VM to the VMM, and subtracting the difference from a value of a timer offset field.

13. The apparatus of claim 12 wherein the data structure is further to include a timer offset field storing the calculated timer offset value.

14. The apparatus of claim 12 wherein the data structure is further to include an adjust timer offset indicator.

15. The apparatus of claim 14 wherein the timer access logic is further to determine that the adjust timer offset indicator is set to an enabled value prior to determining the difference.

16. The apparatus of claim 12 wherein the timer access logic is further to determine that an adjust timer offset indicator is set to a disabled value, and to read a timer offset field, a value of the timer offset field being subsequently used as the offset value.

17. The apparatus of claim 12 wherein the data structure is to store a save timer indicator used by the timer access logic to determine whether to save the timer value upon detecting the prior event associated with the transition of control to the VMM.

18. The apparatus of claim 10 further comprising:

an offset register to store the offset value.

19. A system comprising:

a memory to store a set of fields associated with timer offsetting for a virtual machine (VM); and

a processor, coupled to the memory, to use the set of fields associated with timer offsetting to calculate an offset value, and to provide, during operation of the VM, a timer value, adjusted based on the offset value, to the VM in response to a request of the VM for a current value of a timer.

20. The system of claim 19 wherein the processor is to calculate the offset value when transitioning control to the VM.

21. The system of claim 19 wherein the processor is further to load an offset register with the offset value.

22. The system of claim 19 wherein the processor is to calculate the offset value by determining a difference between a timer value read upon receiving the request to transition control to the VM and a timer value saved upon detecting a prior event associated with a transition of control from the VM to the VMM, and subtracting the difference from a value of a timer offset field.

23. The system of claim 22 wherein the processor is to read the timer value upon receiving the request to transition control to the VM if an adjust timer offset indicator is enabled.

24. The system of claim 22 wherein the processor is to save the timer value upon detecting the prior event associated with the transition of control to the VMM if a save timer indicator is enabled.

25. A machine-readable medium containing instructions which, when executed by a processing system, cause the processing system to perform a method, the method comprising:

receiving a request to transition control to a virtual machine (VM) from a virtual machine monitor (VMM);

calculating an offset value;

receiving, during operation of the VM, a request for a current value of a timer;

adjusting the current value of the timer based on the offset value; and

providing the adjusted timer value to the VM.

26. The machine-readable medium of claim 25 wherein the method further comprises:

transitioning control to the VM upon receiving the request from the VMM, the transitioning comprising calculating the offset value.

27. The machine-readable medium of claim 25 wherein calculating the offset value comprises:

determining a difference between a timer value read upon receiving the request to transition control to the VM and a timer value saved upon detecting a prior event associated with a transition of control from the VM to the VMM; and

subtracting the difference from a value of a timer offset field.

28. The machine-readable medium of claim 27 wherein the timer value is read upon receiving the request to transition control to the VM if an adjust timer offset indicator is enabled.

29. The machine-readable medium of claim 27 wherein the timer value is saved upon detecting the prior event associated with the transition of control to the VMM if a save timer indicator is enabled.

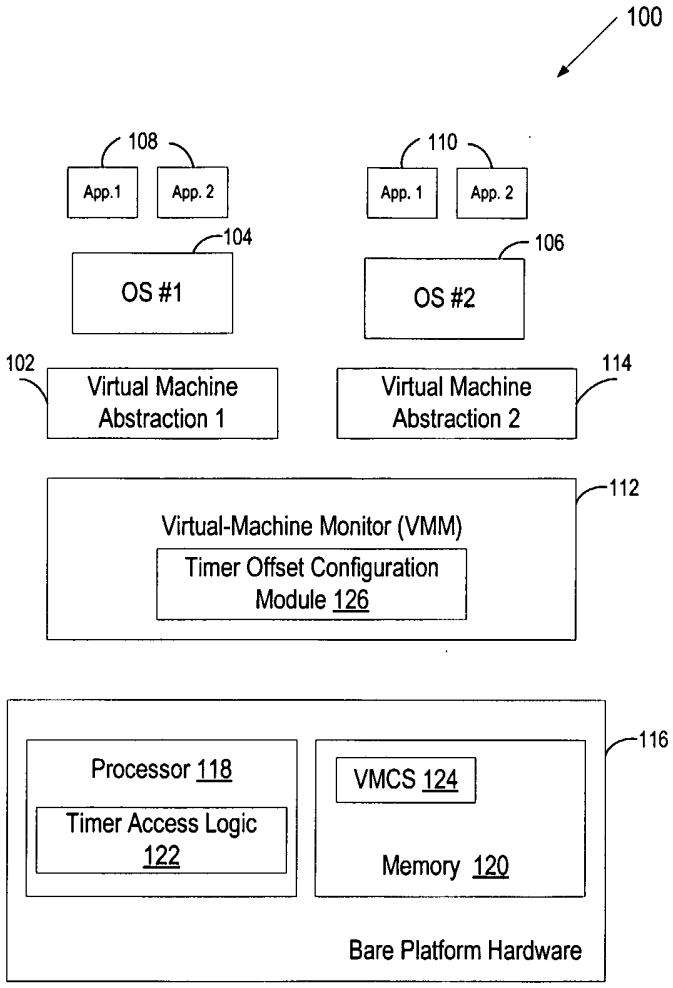

100
108
App.1
App. 2
110
App. 1
App. 2
104
OS #1
106
OS #2
102
Virtual Machine Abstraction 1
114
Virtual Machine Abstraction 2
112
Virtual-Machine Monitor (VMM)
Timer Offset Configuration Module 126
116
Processor 118
Timer Access Logic 122
VMCS 124
Memory 120
Bare Platform Hardware

FIG. 1

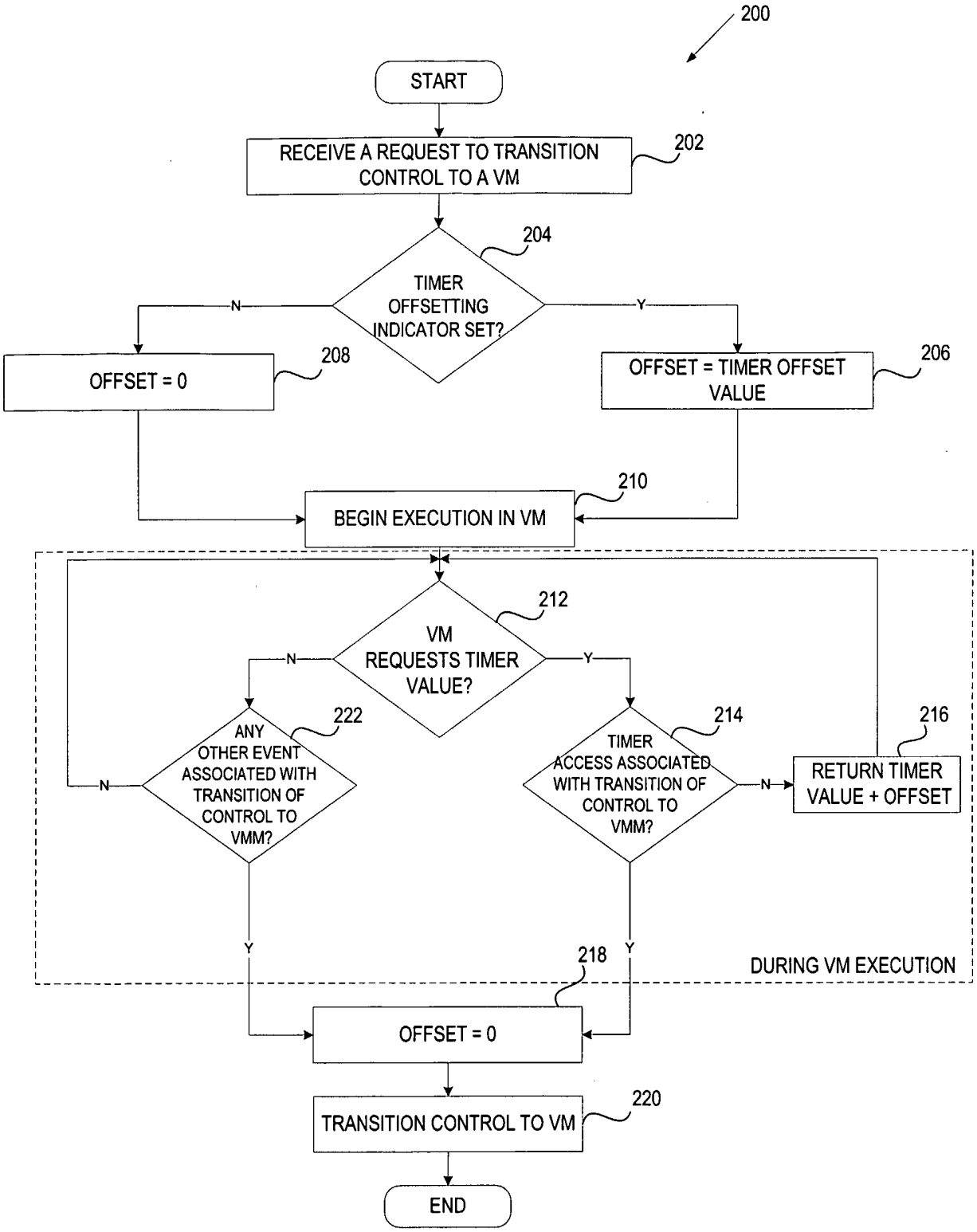

200
START
RECEIVE A REQUEST TO TRANSITION CONTROL TO A VM
202
204
TIMER OFFSETTING INDICATOR SET?
N
Y
OFFSET = 0
208
OFFSET = TIMER OFFSET VALUE
206
210
BEGIN EXECUTION IN VM
212
VM REQUESTS TIMER VALUE?
N
Y
222
ANY OTHER EVENT ASSOCIATED WITH TRANSITION OF CONTROL TO VMM?
214
TIMER ACCESS ASSOCIATED WITH TRANSITION OF CONTROL TO VMM?
216
RETURN TIMER VALUE + OFFSET
DURING VM EXECUTION
218
OFFSET = 0
220
TRANSITION CONTROL TO VM
END

FIG. 2

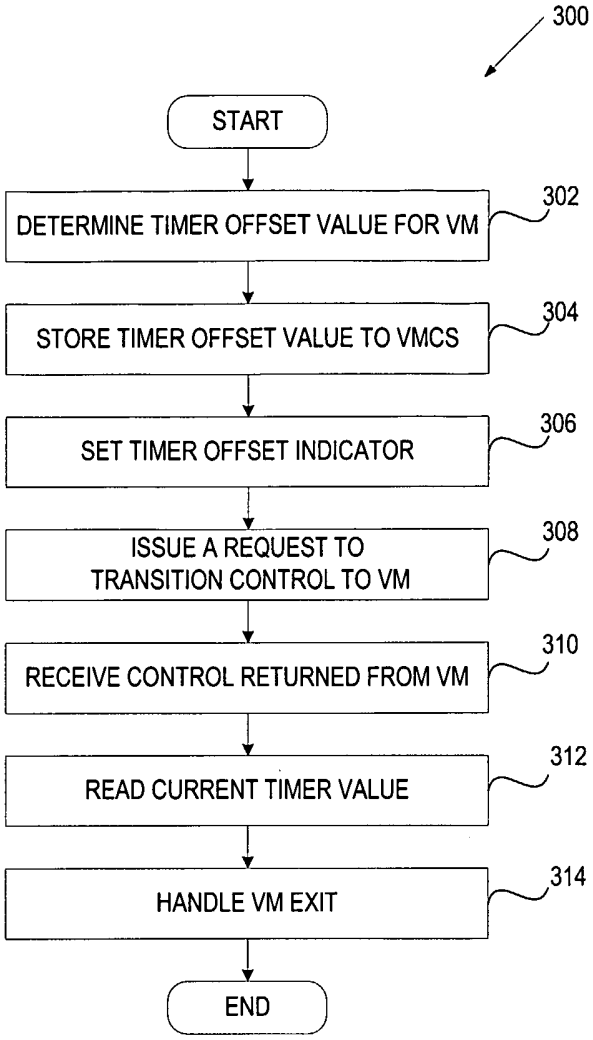

300
START
DETERMINE TIMER OFFSET VALUE FOR VM
302
STORE TIMER OFFSET VALUE TO VMCS
304
SET TIMER OFFSET INDICATOR
306
ISSUE A REQUEST TO TRANSITION CONTROL TO VM
308
RECEIVE CONTROL RETURNED FROM VM
310
READ CURRENT TIMER VALUE
312
HANDLE VM EXIT
314
END

FIG. 3

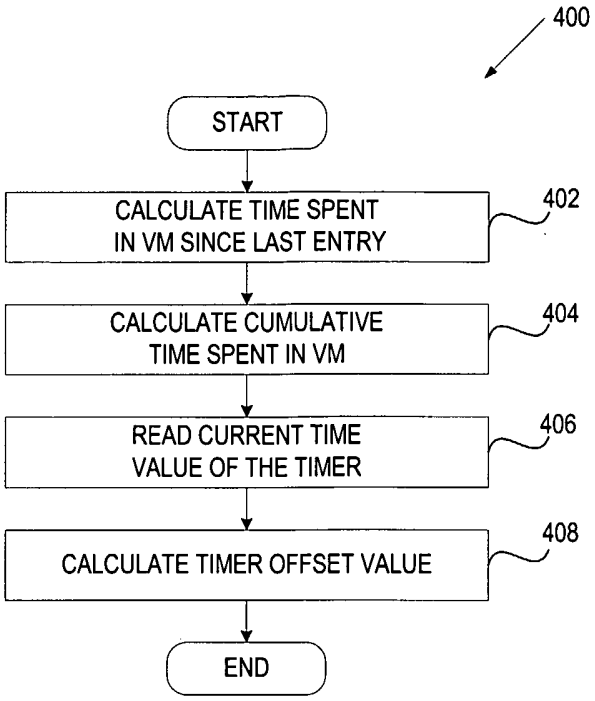

400
START
CALCULATE TIME SPENT IN VM SINCE LAST ENTRY
402
CALCULATE CUMULATIVE TIME SPENT IN VM
404
READ CURRENT TIME VALUE OF THE TIMER
406
CALCULATE TIMER OFFSET VALUE
408
END

FIG. 4

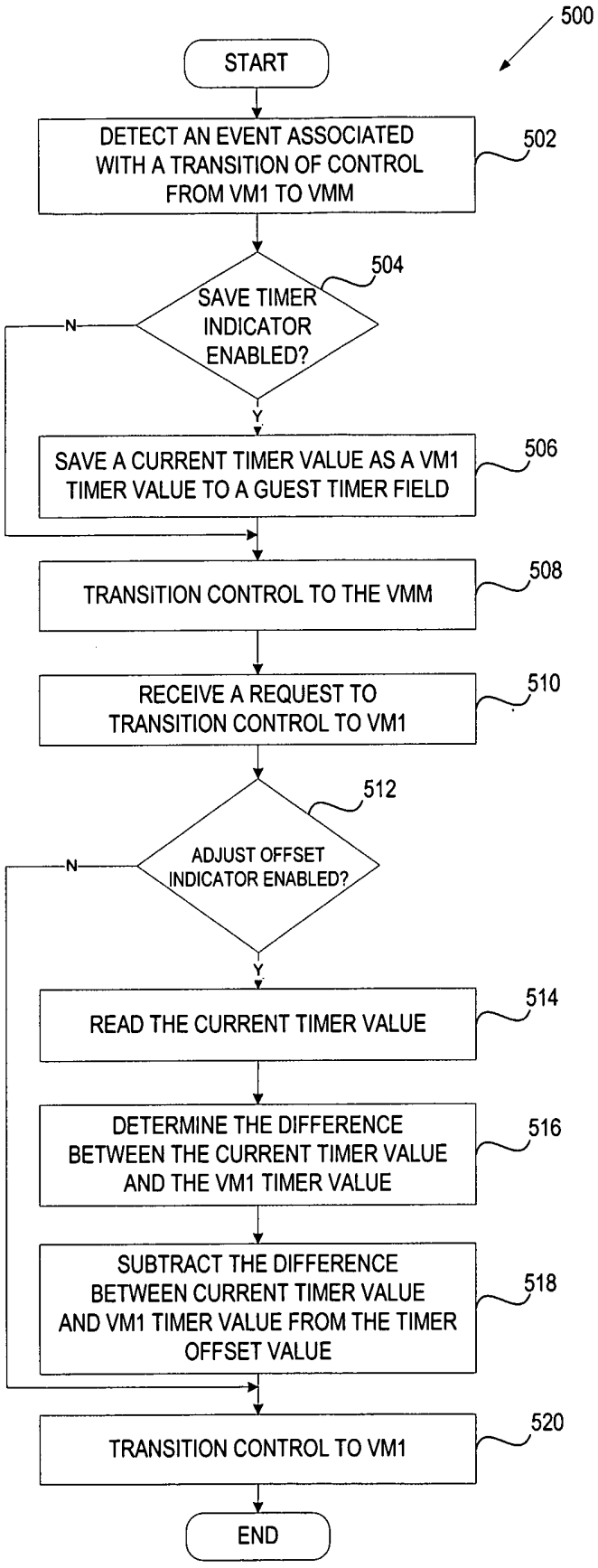

500
START
DETECT AN EVENT ASSOCIATED WITH A TRANSITION OF CONTROL FROM VM1 TO VMM
502
SAVE TIMER INDICATOR ENABLED?
504
N
Y
SAVE A CURRENT TIMER VALUE AS A VM1 TIMER VALUE TO A GUEST TIMER FIELD
506
TRANSITION CONTROL TO THE VMM
508
RECEIVE A REQUEST TO TRANSITION CONTROL TO VM1
510
ADJUST OFFSET INDICATOR ENABLED?
512
N
Y
READ THE CURRENT TIMER VALUE
514
DETERMINE THE DIFFERENCE BETWEEN THE CURRENT TIMER VALUE AND THE VM1 TIMER VALUE
516
SUBTRACT THE DIFFERENCE BETWEEN CURRENT TIMER VALUE AND VM1 TIMER VALUE FROM THE TIMER OFFSET VALUE
518
TRANSITION CONTROL TO VM1
520
END

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No
F.., JS2005/040450

A. CLASSIFICATION OF SUBJECT MATTER
G06F9/455

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

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

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5 317 754 A (BLANDY ET AL) 31 May 1994 (1994-05-31) column 2, line 45 - line 58	1-29
Y	JOHN FRANCISCOVICH: "NATURAL/ADABAS in VM/VSE"'Online! 9 July 1997 (1997-07-09), XP002367201 Retrieved from the Internet: URL:http://listserv.uark.edu/scripts/wa.exe?A2=ind9707&L=vmesa-1&P=7530> 'retrieved on 2006-02-09! the whole document	1-29
A	US 5 381 535 A (GUM ET AL) 10 January 1995 (1995-01-10) the whole document	1-29

☐ 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 document 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
10 February 2006	28/02/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Dewyn, T

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2005/040450

Patent document cited in search report		Publication date	Patent family member(s)	Publication date
US 5317754	A	31-05-1994	NONE	
US 5381535	A	10-01-1995	NONE	