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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 November 2013 (14.11.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/167394 A1

(51) International Patent Classification:
G06F 11/36 (2006.01)

(21) International Application Number:
PCT/EP2013/058688

(22) International Filing Date:
26 April 2013 (26.04.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
1254162 7 May 2012 (07.05.2012) FR

(71) Applicant: **STMICROELECTRONICS (GRENOBLE 2) SAS** [FR/FR]; 12, Rue Jules Horowitz, F-38000 Grenoble (FR).

(72) Inventors: **PAIRE, Eric**; 15, Rue Augereau, F-38000 Grenoble (FR). **SAUZEDE, Nicolas**; 4, allée du Morvan, F-38130 Echirolles (FR).

(74) Agent: **CABINET BEAUMONT**; 1, Rue Champollion, F-38000 Grenoble (FR).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

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

(54) Title: INSTRUCTION EXECUTION SIMULATION METHOD USING TRAPPED ADDRESSES AND ASSOCIATED HARDWARE

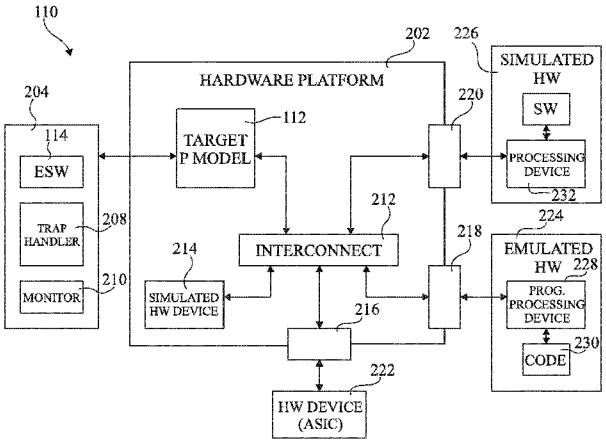

Fig 2

(57) Abstract: The invention concerns a method of simulating, using a target processor model (112) comprising at least one host processing device, the execution of program instructions on a target processing device (102), the program instructions including at least one transaction instruction associated with a hardware address, each hardware address of said at least one transaction instruction corresponding to one or more trapped memory addresses, said instructions being in the instruction set of said at least one host processing device of said target processor model, the method comprising: executing the instructions of said computer program by said one or more host processing devices of said target processor model, wherein a software trap is generated upon each execution of said at least one transaction instruction; and in response to each software trap, decoding and implementing the transaction instruction causing the software trap.

WO 2013/167394 A1

INSTRUCTION EXECUTION SIMULATION METHOD USING TRAPPED ADDRESSES AND ASSOCIATED HARDWARE

FIELD

The present invention relates to a method of simulating the execution of program instructions by a target processing device, and to a hardware platform for performing the same.

BACKGROUND

A major part of the design cycle of an electronics device, such as a mobile telephone, tablet computer or the like, is the development of embedded software. In particular, it is necessary to generate and then rigorously test and verify the embedded software that will be executed by the one or more processors of the electronics device, and which will also drive the various peripheral hardware devices incorporated in the device.

In order to reduce the time to market, it is often desirable that embedded software can be tested and verified before the hardware chip is available. In order to perform such tests, models of the one or more target processors and peripheral devices may be implemented as host processor programs executed on a host system, and are used to simulate the operation of the target processors and peripheral devices.

In many cases, the programming languages, in other words the instruction sets, used by the target processor and by the host system are different. It has been proposed to execute the embedded software on the host system using one of two solutions.

A first solution is to provide the embedded software in the instruction set of its target processor. To execute this embedded software using one or more processors of the host system, each instruction is decoded, so that it can then be simulated by instructions executable by the host system. Such processing is performed by what is generally known as an "Instruction set simulator". However, a problem with such a solution is that it is inherently slow, due to the processing that is needed for each instruction.

A second solution is to provide the embedded software in the instruction set of the host system. The embedded software should be modified such that for each instruction that accesses a hardware component, a custom function is called that simulates the execution of the instruction. For example, a method in which software is modified to use explicit call functions to perform hardware transactions in a native simulation environment is described in more detail in US patent application published as US2005/0149897. However, a problem with this solution is that it is error-prone and costly to modify the source code of the embedded software in this way, and furthermore, such a modification is not always possible.

Thus there is a technical problem in providing a fast, cheap and reliable solution for simulating the operation of a target processor interacting with any target hardware platform component.

SUMMARY

It is an aim of the embodiments described herein to at least partially address one or more needs in the prior art.

According to one aspect, there is provided a method of simulating, using a target processor model comprising at least one host processing device, the execution of program

instructions on a target processing device, the program instructions including at least one transaction instruction involving a transaction between said target processing device and at least one hardware component associated with a hardware address, each hardware address of said at least one transaction instruction corresponding to one or more trapped memory addresses, said instructions being in the instruction set of said at least one host processing device of said target processor model, the method comprising: executing the instructions of said computer program by said one or more host processing devices of said target processor model, wherein a software trap is generated upon each execution of said at least one transaction instruction; and in response to each software trap, decoding and implementing, for example by a monitor rather than by the target processor model, the transaction instruction causing the software trap. For example, the method further comprises, after implementing the transaction instruction by the monitor, executing by the target processor model, a subsequent instruction of said computer program.

According to one embodiment, the method further comprises, prior to executing the instructions of the computer program, assigning each hardware address of the at least one transaction instruction to one or more trapped memory addresses.

According to one embodiment, the method further comprises, after implementing the instruction, resuming by the at least one host processing device of said target processor model the execution of said program instructions.

According to another embodiment, the step of implementing the transaction instruction comprises calling a monitor modelling the behaviour of said target processing device in relation to a transaction instruction, wherein upon execution, said monitor implements the transaction instruction.

According to another embodiment, the execution of said transaction instruction by said monitor involves accessing said hardware component or a hardware device implementing said hardware component.

According to another embodiment, in response to each software trap, a trap handler is configured to call said monitor.

According to a further aspect, there is provided a computer readable medium storing a computer program that when executed by at least one processing device causes the above method to be implemented.

According to a further aspect, there is provided a hardware platform for simulating, using a target processor model comprising at least one host processing device, the execution of program instructions on a target processing device, the program instructions including at least one transaction instruction involving a transaction between said target processing device and at least one hardware component associated with a hardware address, each hardware address of said at least one transaction instruction corresponding to one or more trapped memory addresses, said instructions being in the instruction set of said at least one host processing device of said target processor model, the target processor model being configured to: execute the program instructions by said at least one host processing device of said target processor model and generate a software trap upon each execution of said at least one transaction instruction; and decode and implement, in response to each software trap, the transaction instruction causing the software trap.

According to one embodiment, the target processor model is further configured to assign a trapped memory address to each hardware address of said at least one transaction instruction.

According to one embodiment, the hardware platform further comprises an interconnection coupling said target processor model to at least one hardware device representing said at least one hardware component.

According to another embodiment, said at least one hardware device comprises one or more of the following: a hardware device implementing said hardware component; an

interface to an external device implementing said hardware component; an interface to an external device emulating said hardware component; and an interface to an external device simulating said hardware component.

According to a further aspect, there is provided a simulation system comprising: the above hardware platform; a memory device storing said program instructions; and at least one device representing said hardware component.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other purposes, features, aspects and advantages will become apparent from the following detailed description of embodiments, given by way of illustration and not limitation with reference to the accompanying drawings, in which:

Figure 1A schematically illustrates an example of an electronic device;

Figure 1B schematically illustrates a simulation system according to an example embodiment;

Figure 2 schematically illustrates the simulation system of Figure 1B in more detail according to an example embodiment;

Figure 3 represents a portion of an execution space of a target processor model according to an example embodiment;

Figure 4A is a flow diagram illustrating operations in a method according to an example embodiment;

Figure 4B is a flow diagram illustrating in more detail the steps performed during a step of Figure 4A of executing embedded software according to an example embodiment;

Figure 5 is a time line representing the execution of a transaction instruction according to an example embodiment; and

Figure 6 is a flow diagram illustrating in more detail the execution of a transaction instruction according to an example embodiment.

DETAILED DESCRIPTION

Throughout the following description, only those aspects useful for an understanding of the present disclosure will be described in detail, and have been represented in the drawings. Other features, such as the particular features implemented by the embedded software, have not been described in detail, it being clear to those skilled in the art that the embodiments described herein could be applied to embedded software implementing a broad range of features.

In the following text, the following terms will be given the following definitions:

"target processing device" - circuitry comprising one or more processors forming at least part of the hardware of an end product for which software is to be developed;

"host processing device" - circuitry comprising one or more processors forming at least part of a hardware platform for simulating the behaviour of the target processing device upon execution of the software under development;

"target processor model" - a module implemented by a combination of at least one host processing device and for example one or more programs, and configured to model the execution of program instructions on the target processing device;

"embedded software" - any software that for example forms an integral part of an electronics device, such as an end product, and has instructions that when executed cause an interaction with target hardware components, for example including drivers for one or more further target hardware devices, as well as one or more features;

"software trap" - an event resulting from a host instruction execution error, such as a trapped address, that causes a trap that diverts the instruction execution flow of the embedded software, permitting a trap handler to take control;

"hardware component" - any hardware device, such as a peripheral device, memory, register, input/output port, video

decoder, etc., which may be implemented by a specific hardware device, or the operation of which may be simulated;

"transaction instruction" - an instruction, such as a load or store instruction, that involves a transaction, such as a read or write operation, between a target processor and a hardware component; and

"monitor" is software that forms part of the target processor model and which models the behaviour of the target processing device with respect to transaction instructions.

Figure 1A schematically illustrates an example of an electronic device 100. The device 100 comprises a processing device 102, which will be referred to herein as a target processing device. Furthermore, the device 100 comprises a memory device storing embedded software 104 in the form of program instructions for execution by the target processing device 102, and a hardware component 106 coupled to the target processing device 102. The device 100 is for example any electronics device such as, but not limited to, a laptop, tablet or personal computer, a mobile telephone, a digital camera, a portable games console, a personal digital assistant, an electronic book reader, a set-top-box or a digital media player.

The target processing device 102 for example comprises one or more processors, DMA (direct memory access) units, and/or other hardware under the control of the program instructions of the embedded software.

The hardware component 106 is for example a memory such as a RAM (random access memory) or flash memory, a register, an input/output port, a video decoder, a graphics card, a peripheral device such as a printer, scanner, microphone, speaker, etc. or the controller of such a peripheral device. While in the example of Figure 1A the device 100 comprises only one hardware component, in practise it may comprise several such components, or even tens of hardware components. While not illustrated in Figure 1A, the device 100 may further comprise input/output devices such as a keyboard and/or mouse, a display and/or a touch screen, etc.

As explained in the background section above, during the development phase of the embedded software to be executed by the target processing device 102, one or more components can be used to simulate the operation of the target processing device 102 and the hardware component 106. This permits software development to start before the hardware of the device 100 is available.

Figure 1B schematically illustrates an example of a simulation system 110 comprising a host subsystem 111 that models the operation of the target processing device 102 and the embedded software 104. The subsystem 111 comprises a target processor model 112 coupled to a memory device storing embedded software (ESW) 114, which interacts with the target processor model 112. The target processor model 112 is implemented by one or more host processing devices (not illustrated in Figure 1B), and for example models the behaviour of the target processing device 102 at a certain level of accuracy in terms of its functionality, its timing, its bit and cycle behaviour, or its gate level operation.

The subsystem 111 for example communicates with a hardware device 116, which represents the hardware component 106. In particular, as will be described in more detail below, the device 116 may correspond to the hardware component 106 itself, or to an implementation of the hardware component 106, for example using simulation/emulation tools.

While the target processing device 102 uses a first instruction set for executing program instructions of the embedded software 104, the one or more host processing devices of the target processor model 112 use a second instruction set, which is for example different to the first, for executing the program instructions of the embedded software 114. However, even if the instruction sets of the software 104 and 114 are different, each software 104, 114 is for example derived from the same source code, or at least from source code that is substantially identical. As will be described in more detail below, the embedded software 114 corresponds to a computer

program comprising instructions in the programming language of the one or more host processors of the target processor model 112. These instructions for example comprise code for not only controlling the target processor model 112, but also for driving the hardware component 116. The computer program for example implements one or more tests, verification sequences, OS (operating system) drivers, or more complex applications.

Figure 2 illustrates the simulation system 110 of Figure 1B in more detail according to an example embodiment.

As illustrated, the target processor model 112 is for example implemented on a hardware platform 202. The hardware platform 202 is for example implemented by any technology such as software models, FPGA, hardware emulation, hardware devices, or any combination thereof. Furthermore, a memory device 204 is coupled to the target processor model 112. The memory device 204 may be implemented by one or more physical memories, and stores the embedded software 114. The memory device 204 is managed by a trap handler 208, the purpose of which will be described in more detail below. Memory device 204 for example further comprises software 210 corresponding to a monitor, which takes control in the event of a trap, as will be described in more detail below.

The hardware platform 202 for example comprises an interconnect 212 that couples the target processor model 112 to each of a number of hardware devices. In particular, in the example of Figure 2, the interconnect 212 is coupled to a simulated hardware device 214, and to input/output ports 216, 218 and 220 respectively providing access to external hardware devices 222, 224 and 226. Device 222 is for example an ASIC (application specific integrated circuit), and corresponds to a hardware component, such as the component 106 of Figure 1A. Device 224 is for example a device that emulates the operation of a hardware component, such as the component 106 of Figure 1A. In particular, it for example comprises a programmable processing device 228, such as an FPGA (field programmable gate array), which is configured by RTL (register transfer level) code stored in a memory 230. The device 226 is for example a

software model simulating the operation of a hardware component, such as the component 106 of Figure 1A. In particular, it for example comprises a processing device 232 controlled by modelling code, that describes the hardware function to be implemented.

Of course, while the example of Figure 2 illustrates four different types of devices 214, 222, 224 and 226 implementing hardware components, it will be apparent to those skilled in the art that only one or some of these devices could be provided, and that there may be more than one of each type of device. Furthermore, one or more of the hardware component implemented by each of the devices 214, 222, 224 and 226 may be the same, or they could all be different.

In operation, during a simulation using the hardware platform 202, the program instructions 114 are executed by the one or more host processing devices of the target processor model 112. The instructions being in the instruction set of the one or more host processors, such an execution is relatively rapid. However, as will be described in more detail below, each instruction that corresponds to a transaction instruction accessing a hardware component, such as the hardware components 214, 222, 224, 226 of Figure 2, causes a software trap to occur in the memory device 204. Control is then passed to the trap handler 208 controlling the hardware platform 202, and the trap handler 208 in turn passes control to the monitor 210, which permits the transaction instruction to be processed.

The operation of the simulation system 110 of Figure 2 will now be described in more detail with reference to Figures 3, 4A and 4B.

Figure 3 represents a portion of a virtual execution space of the target processor model 112. For example, this execution space corresponds to a physical memory space in the device 204, and/or in an on-chip memory of the hardware platform 202 (not illustrated in Figure 2). In the example of Figure 3, the execution space spans addresses "0000" to "FFFF", although of course the execution space could be of a different size and

position. The instruction code is for example stored in addresses "A" to "B" of the execution space, while addresses "C" to "D" are assigned for use in storing data during execution of the embedded software. The other zones, which are shaded with diagonal stripes in Figure 3, correspond to addresses not corresponding to the embedded software. In the example of Figure 3, these zones correspond to the addresses from "0000" to "A", from "B" to "C" and from "D" to "FFFF". In the example of Figure 3, a portion of those addresses from address "0000" to an address "E" are actual hardware addresses of the hardware components used during the transactions instructions. Thus, when any of these addresses is used during instruction execution, a software trap is automatically generated by the one or more host processing devices of the target processor model 112, due to the presence of a trapped address.

Those skilled in the art will appreciate that in some cases, trapped addresses may also correspond to memory addresses of one or more memory devices that store the embedded software instructions and/or the data.

Figure 4A is a flow diagram illustrating an example of steps performed by the hardware platform 202 for simulating the operation of the target processing device 102.

In a step S1, the hardware address ranges are marked as trapped areas by the hardware platform, and internally designated as corresponding to transaction instructions. During this procedure, each time an address corresponds to the hardware address of a hardware component, the assigned virtual address is a trapped address, for example an address in the virtual space "0000" to "E" of Figure 3. In at least some cases, the hardware address of the hardware component may already correspond to an invalid, and thus trapped, memory address. In such cases, it is for example not necessary to assign a new trapped address.

In a subsequent step S2, the instructions of the embedded software are executed, as will now be described in more detail with reference to Figure 4B.

Figure 4B is a flow diagram illustrating an example of the steps performed during the instruction execution step S2 of Figure 4A.

In a step S2A, an instruction counter N is initialized, for example to A.

In a subsequent step S2B, instruction N of the program instructions is executed, this initial instruction for example corresponding to a first instruction A.

In a subsequent step S2C, the occurrence of a trap is detected. In particular, if the instruction N is a transaction instruction involving the hardware address of a hardware component, a trap will be generated by the one or more host processing devices of the target processor model 112. In this case, the next step is S2D, in which the transaction instruction N is decoded and implemented by monitor 210.

After step S2D, or after step S2C in the case that no trap occurs on execution of instruction N, the next step is S2E, in which the program counter N is incremented by the monitor 210 or by the one or more host processing devices of the target processor model 112, and then the method returns to repeat step S2B based on the new instruction counter value N.

The operation of the hardware platform during the execution of a transaction instruction will now be described in more detail with reference to Figures 5 and 6.

Figure 5 is a time line illustrating the execution sequences during the execution of a transaction instruction.

Figure 6 is a flow diagram illustrating steps in a method of processing a transaction instruction.

Initially, as represented by a line in Figure 5 representing the embedded software, the program instructions of the embedded software are executed by the one or more host processors of the target processor model 112. Then, at a point 502 shown in Figure 5, corresponding to step 602 of Figure 6, a trapped address of an instruction n triggers a trap.

As illustrated by a line 504 in Figure 5 and a corresponding step 604 of Figure 6, the control is passed from the embedded software to the trap handler 208 of Figure 2.

Then, as illustrated by an arrow 506 in Figure 5 and by a corresponding step 606 in Figure 6, the trap handler 208 calls the monitor 210 of Figure 2.

Then, as illustrated by a line 508 in Figure 5 and by a corresponding step 608A in Figure 6, the monitor 210 decodes the transaction instruction n that caused the trap. For example, decoding the instruction involves retrieving, based on the instruction counter at the trap time, the corresponding instruction from the embedded software. Decoding this instruction for example provides at least an instruction type, such as "load" or "store", the hardware address, a size corresponding to the number of bytes of data to be read or written, and/or the data to be written in the case of a store instruction. Then, if the instruction is a load instruction, a read transaction is generated by the monitor 210 in order to implement the read operation based on the decoded instruction parameters. Alternatively, if the instruction is a store instruction, a write transaction is generated by the monitor 210 in order to implement the write operation based on the decoded instruction parameters. The read and write transactions are for example executed by the target processor model 112, resulting in communication with one of the hardware devices 214, 222, 224 or 226 of Figure 2, via the interconnect 212 and/or the input/output ports 216, 218, 220.

As illustrated by a subsequent step 608B in Figure 6, the monitor 210 implements the operation of the transaction instruction once the transaction, for example a read or write, has been completed. For example, in the case of a read instruction, the destination register of the read instruction is updated with an appropriate value provided by the read transaction and appropriate host processor flags are set. In the case of a write instruction, appropriate host processor flags are set.

The monitor 210 increments the embedded software instruction counter then returns control briefly to the trap handler, which in turn returns control to the embedded software, as represented by the point 510 of Figure 5, and instruction execution proceeds from instruction n+1.

The process then for example continues until a subsequent trap is encountered, at which point the method starts again from step 602.

An advantage of the embodiments described herein is that, by assigning trapped addresses to the hardware addresses of one or more hardware components corresponding to transaction instructions, traps will be generated by the one or more host processing devices of the target processor model when these instructions are executed, providing a simple and efficient manner in which to allow another control system to take control, and process the transaction instructions. This permits the embedded software to be programmed in the instruction set of the host system, and thus the execution of the embedded software is particularly fast.

Having thus described at least one illustrative embodiment, various alterations, modifications and improvements will readily occur to those skilled in the art.

For example, while certain examples have been provided for how a transaction instruction is implemented following a software trap, it will be apparent to those skilled in the art that there are alternative ways that this could be achieved, for example by using a look-up table to directly simulate the behaviour of the hardware component.

Furthermore, while embodiments have been described in which the software 104 of the electronics device 102 and the software 114 of the simulation system is embedded software, it will be apparent to those skilled in the art that in some cases the software 104 and 114 may not be embedded.

Furthermore, while examples of transaction instructions corresponding to store and load instructions have been described, it will be apparent to those skilled in the art that embodiments described herein could be equally applied to other types of instructions, such as a complex “compare-and-swap” or cache management instruction.

Furthermore, as it will be apparent to those skilled in the art, the various features described in relation with the various embodiments could be combined in any combination.

CLAIMS

1. A method of simulating, using a target processor model (112) comprising at least one host processing device, the execution of program instructions on a target processing device (102), the program instructions including at least one transaction instruction involving a transaction between said target processing device (102) and at least one hardware component (116) associated with a hardware address, each hardware address of said at least one transaction instruction corresponding to one or more trapped memory addresses, said instructions being in the instruction set of said at least one host processing device of said target processor model, the method comprising:

executing the instructions of said computer program by said one or more host processing devices of said target processor model, wherein a software trap is generated upon each execution of said at least one transaction instruction; and

in response to each software trap, decoding and implementing the transaction instruction causing the software trap.

2. The method of claim 1, further comprising, prior to executing the instructions of said computer program, assigning each hardware address of said at least one transaction instruction to one or more trapped memory addresses.

3. The method of claim 1 or 2, further comprising, after implementing the instruction, resuming by the at least one host processing device of said target processor model the execution of said program instructions.

4. The method of any of claims 1 to 3, wherein the step of implementing the transaction instruction comprises calling a monitor modelling the behaviour of said target processing device in relation to a transaction instruction, wherein upon execution, said monitor implements the transaction instruction.

5. The method of claim 4, wherein the execution of said transaction instruction by said monitor involves accessing said hardware component (106, 116, 222) or a hardware device (214, 224, 226) implementing said hardware component.

6. The method of claim 4 or 5, wherein in response to each software trap, a trap handler (208) is configured to call said monitor (210).

7. A computer readable medium storing a computer program that when executed by at least one processing device causes the method of any of claims 1 to 6 to be implemented.

8. A hardware platform for simulating, using a target processor model comprising at least one host processing device (112), the execution of program instructions on a target processing device (102), the program instructions including at least one transaction instruction involving a transaction between said target processing device and at least one hardware component (106, 116, 214, 222, 224, 226) associated with a hardware address, each hardware address of said at least one transaction instruction corresponding to one or more trapped memory addresses, said instructions being in the instruction set of said at least one host processing device of said target processor model, the target processor model being configured to:

execute the program instructions by said at least one host processing device of said target processor model and generate a software trap upon each execution of said at least one transaction instruction; and

decode and implement, in response to each software trap, the transaction instruction causing the software trap.

9. The hardware platform of claim 8, wherein the target processor model is further configured to assign a trapped memory address to each hardware address of said at least one transaction instruction.

10. The hardware platform of claim 8 or 9, further comprising an interconnection (212) coupling said target processor model to at least one hardware device (214, 222, 224, 226) representing said at least one hardware component.

11. The hardware platform of any of claims 8 to 10, wherein said at least one hardware device comprises one or more of the following:

a hardware device (214) implementing said hardware component;

an interface (216) to an external device (222) implementing said hardware component;

an interface (218) to an external device (224) emulating said hardware component; and

an interface (220) to an external device (226) simulating said hardware component.

12. A simulation system comprising:

the hardware platform (202) of any of claims 8 to 11;

a memory device (204) storing said program instructions; and

at least one device (214, 222, 224, 226) representing said hardware component.

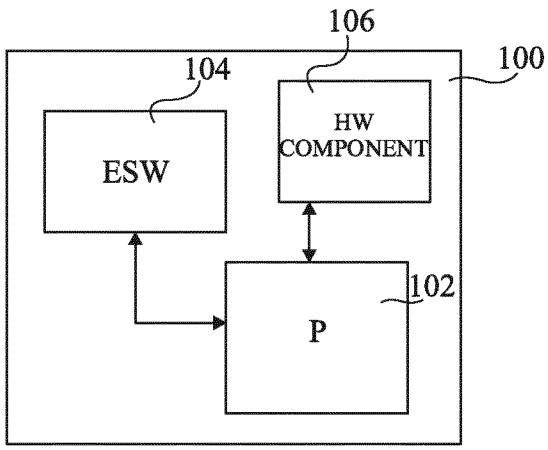

Fig 1A

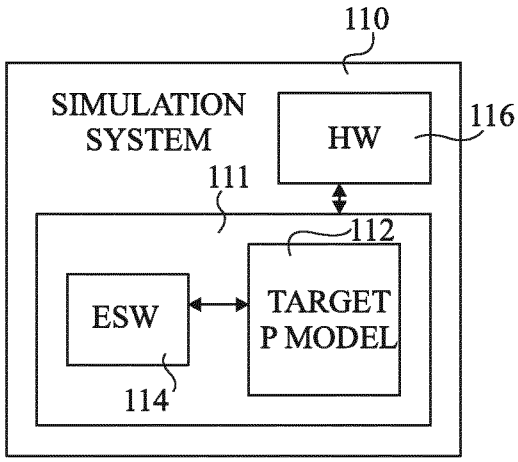

Fig 1B

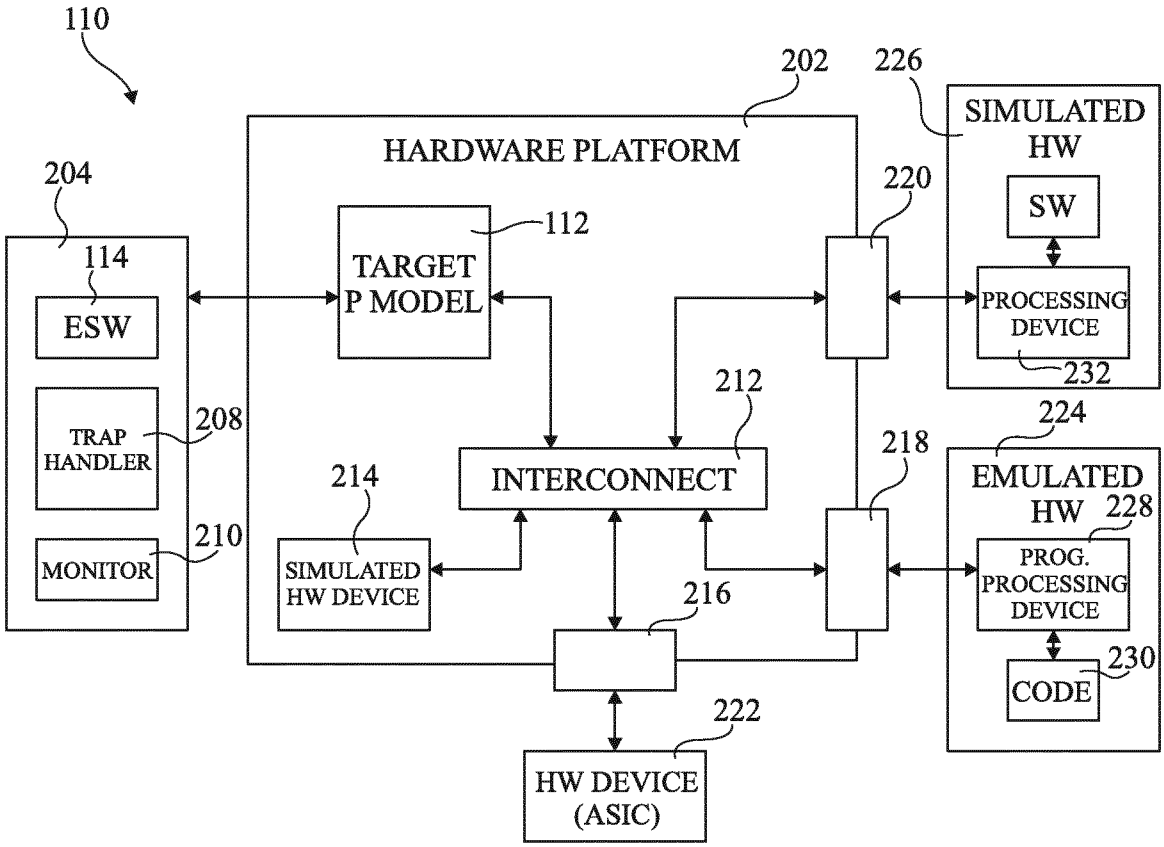

Fig 2

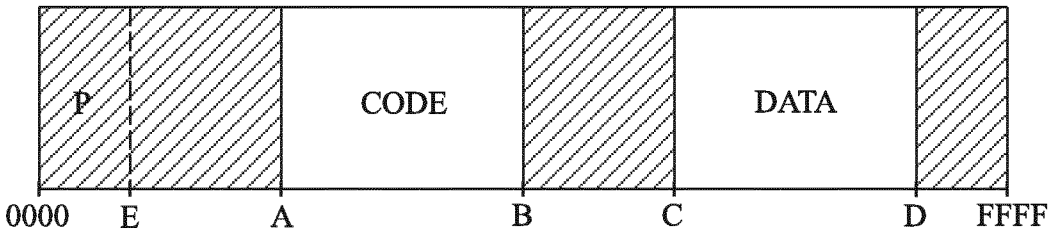

Fig 3

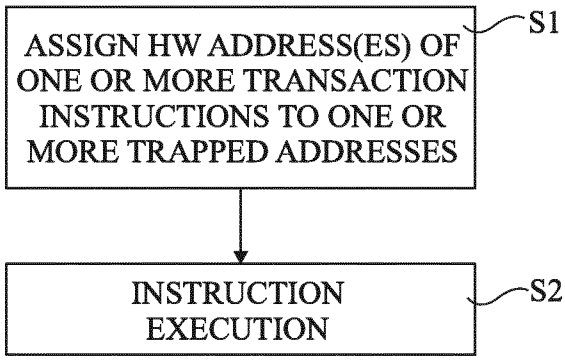

Fig 4A

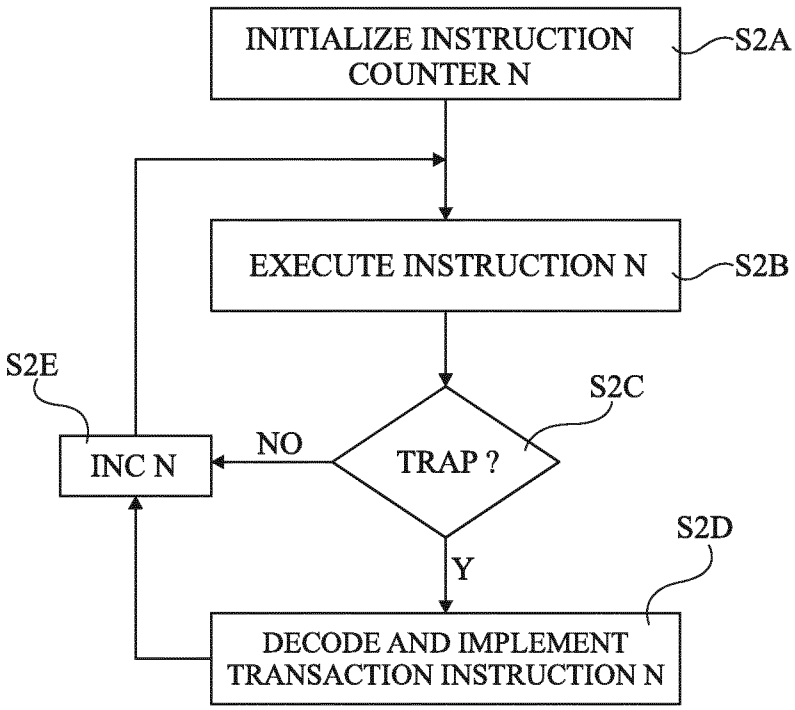

Fig 4B

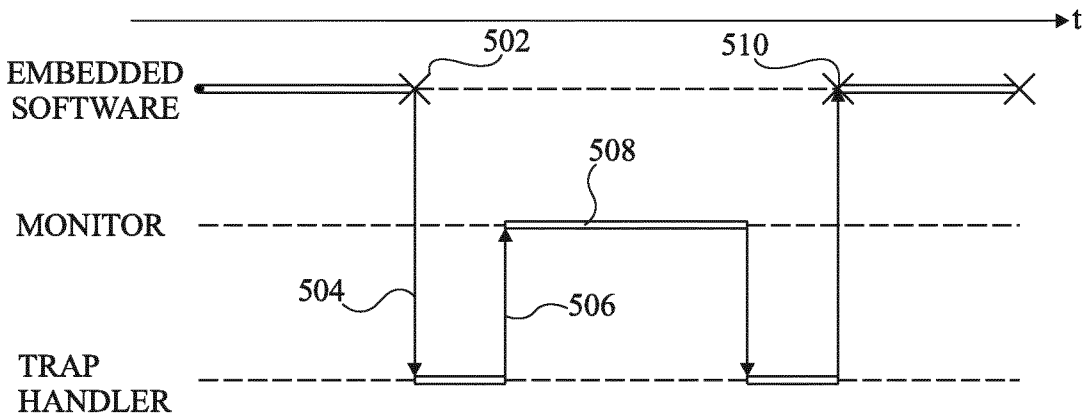
t
502
510
EMBEDDED SOFTWARE
508
MONITOR
504
506
TRAP HANDLER

Fig 5

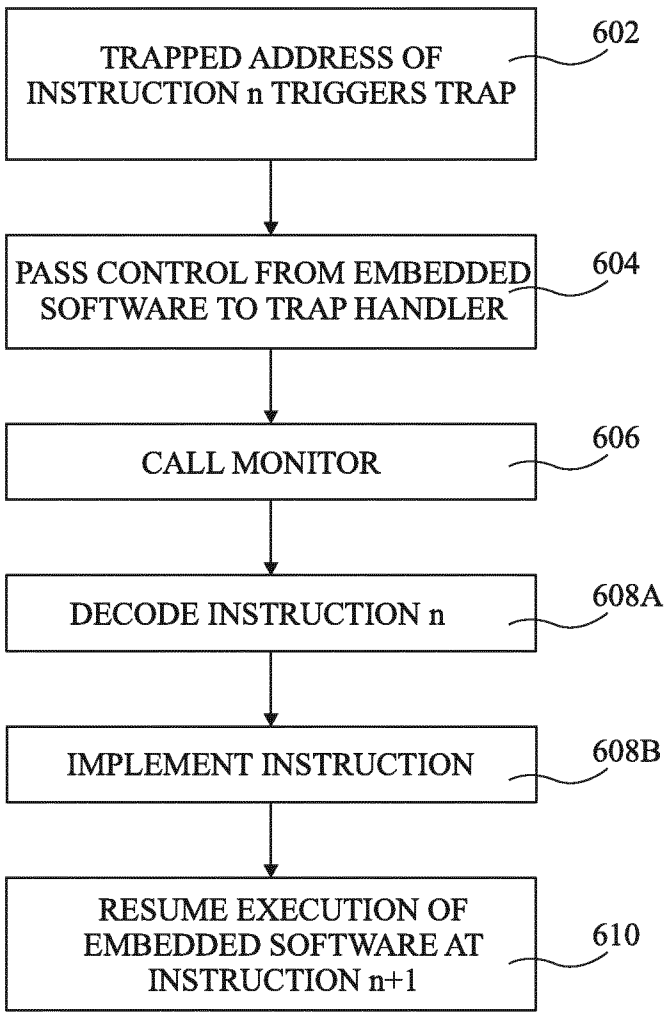
TRAPPED ADDRESS OF INSTRUCTION n TRIGGERS TRAP
602
PASS CONTROL FROM EMBEDDED SOFTWARE TO TRAP HANDLER
604
CALL MONITOR
606
DECODE INSTRUCTION n
608A
IMPLEMENT INSTRUCTION
608B
RESUME EXECUTION OF EMBEDDED SOFTWARE AT INSTRUCTION n+1
610

Fig 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2013/058688

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F11/36
ADD.

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 practicable, search terms used)
EPO-Internal, WPI Data, IBM-TDB, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JASON E MILLER ET AL: "Graphite: A distributed parallel simulator for multicores", HIGH PERFORMANCE COMPUTER ARCHITECTURE (HPCA), 2010 IEEE 16TH INTERNATIONAL SYMPOSIUM ON, IEEE, PISCATAWAY, NJ, USA, 9 January 2010 (2010-01-09), pages 1-12, XP031670702, ISBN: 978-1-4244-5658-1 abstract page 1, left-hand column, paragraph 1 page 1, column d, paragraph 2 - page 1, column d, paragraph 4 page 2, left-hand column, paragraph 5 - page 4, right-hand column, paragraph 2; figure 2 ----- -/--	1-12

[X] 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 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
13 June 2013	24/06/2013

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, Fax: (+31-70) 340-3016	Lelait, Sylvain

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2013/058688

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	HÉCTOR POSADAS ET AL: "Automatic HW/SW Interface Modeling for Scratch-Pad and Memory Mapped HW Components in Native Source-Code Co-simulation", THIRD IFIP TC 10 INTERNATIONAL EMBEDDED SYSTEMS SYMPOSIUM, IESS 2009,, September 2009 (2009-09), pages 12-23, XP009167095, abstract page 12, paragraph 2 - page 13, paragraph 2 page 13, last paragraph - page 14, paragraph 4; figure 1 page 15, paragraph 5 - page 22, paragraph 4; figures 3,4,7	1-12

Y	RICHARD UHLIG ET AL: "Trap-driven memory simulation with Tapeworm II", ACM TRANSACTIONS ON MODELING AND COMPUTER SIMULATION, vol. 7, no. 1, January 1997 (1997-01), pages 7-41, XP055053456, ISSN: 1049-3301, DOI: 10.1145/244804.244805 abstract page 9, paragraph 1 - page 9, paragraph 3; figure 1; table I page 12, paragraph 1 - page 20, paragraph 1	1-12

A	S. SHUBHENDU MUKHERJEE ET AL: "Wisconsin Wind Tunnel II: A fast, portable parallel architecture simulator", IEEE CONCURRENCY, vol. 8, no. 4, October 2000 (2000-10), pages 12-20, XP055053454, ISSN: 1092-3063, DOI: 10.1109/4434.895100 page 12, left-hand column, paragraph 1 page 13, left-hand column, paragraph 3 page 13, middle column, paragraph 4 - page 13, right-hand column, paragraph 1	1-12

A	FREDERIC PETROT ET AL: "On MPSoC Software Execution at the Transaction Level", IEEE DESIGN & TEST OF COMPUTERS, IEEE SERVICE CENTER, NEW YORK, NY, US, vol. 28, no. 3, May 2011 (2011-05), pages 32-43, XP011322673, ISSN: 0740-7475, DOI: 10.1109/MDT.2010.118 page 32, left-hand column, paragraph 1 - page 32, right-hand column, paragraph 2 page 35, left-hand column, paragraph 4 - page 35, right-hand column, paragraph 1 page 38, left-hand column, paragraph 3 - page 39, left-hand column, paragraph 1	1-12
