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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 July 2013 (04.07.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/101840 A1

(51) **International Patent Classification:**
G06F 9/06 (2006.01) *G06F 9/30* (2006.01)

(21) **International Application Number:**
PCT/US20 12/07 1668

(22) **International Filing Date:**
26 December 2012 (26. 12.2012)

(25) **Filing Language:** English

(26) **Publication Language:** English

(30) **Priority Data:**
13/341,812 30 December 201 1 (30. 12.201 1) US

(71) **Applicant** *(for all designated States except US):* **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95054 (US).

(72) **Inventors; and**

(71) **Applicants** *(for US only):* **CHEE, Nicholas Cheng Hwa** [MY/MY]; 98-17-12A Sinar Bukit Dumbar, Man Faraday, Gelugor, 11700 (MY). **FOSSUM, Tryggve** [NO/US]; 19 Increase Ward Drive, Northborough, Massachusetts 01532 (US). **HASENPLAUGH, William C.** [US/US]; 44 St. Botolph St., Apt. ID, Boston, Massachusetts 021 16 (US).

(74) **Agents: MENNEMEIER, Lawrence** et al; Mnemoglyphics, LLC, c/o CPA GLOBAL, P.O. Box 52050, Minneapolis, Minnesota 55402 (US).

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

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available):* ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ,

[Continued on next page]

(54) **Title:** HARDWARE COMPILATION AND/OR TRANSLATION WITH FAULT DETECTION AND ROLL BACK FUNCTIONALITY

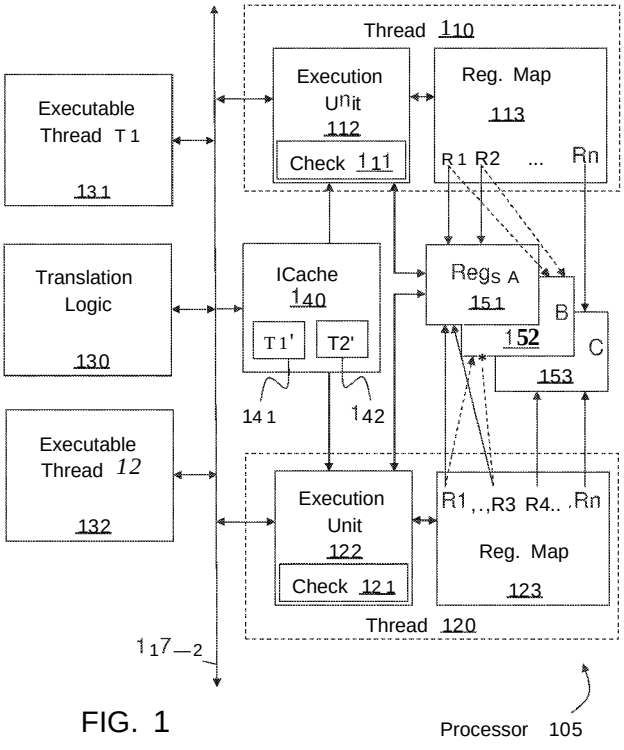


FIG. 1

(57) **Abstract:** Hardware compilation and/or translation with fault detection and roll back functionality are disclosed. Compilation and/or translation logic receives programs encoded in one language, and encodes the programs into a second language including instructions to support processor features not encoded into the original language encoding of the programs. In one embodiment, an execution unit executes instructions of the second language including an operation-check instruction to perform a first operation and record the first operation result for a comparison, and an operation-test instruction to perform a second operation and a fault detection operation by comparing the second operation result to the recorded first operation result. In some embodiments, an execution unit executes instructions of the second language including commit instructions to record execution checkpoint states of registers mapped to architectural registers, and roll-back instructions to restore the registers mapped to architectural registers to previously recorded execution checkpoint states.

WO 2013/101840 A1

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))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

HARDWARE COMPILATION AND/OR TRANSLATION WITH FAULT DETECTION AND ROLL BACK FUNCTIONALITY

FIELD OF THE DISCLOSURE

This disclosure relates generally to the field of microprocessors. In particular, the disclosure relates to hardware compilation and/or translation with fault detection and roll back functionality.

BACKGROUND OF THE DISCLOSURE

Modern processors often include new features, for example, to improve security or support virtualization or multithreading, and/or new instructions to provide operations that are computationally intensive, but offer a high level of data parallelism that can be exploited through an efficient implementation using various data storage devices, such as for example, single instruction multiple data (SIMD) vector registers.

For many of these processors compiler adaptation lags far behind the hardware release, so the new features, instructions and/or resources are under utilized.

On the other hand, fault tolerance and fault detection features are not typically applied to these new features, instructions and/or resources in high-production processors because the implementation costs exceed the benefit. However, reducing the dimensions and increasing the number of transistors in a package and/or processing elements in a multiprocessor, while making devices faster and more efficient, increases the probability of faults due to alpha particles and other causal factors.

Additionally, there are some extreme environments in which fault tolerance is a highly desirable feature of computer systems. For example, a configurable fault tolerant processor (CFTP) was developed by the Space Systems Academic Group at the Naval Postgraduate School using field programmable gate arrays (FPGAs). It was then deployed as an experimental payload on board the United States Naval Academy's (USNA) MidSTAR-1 satellite. A second CFTP system, CFTP-2, was deployed as an entirely ground-based system and was tested in a proton beam using the University of California at Davis' cyclotron.

Binary translation is the emulation of one instruction set by another through translation of code. Sequences of instructions are translated from the source to the target instruction set. Static binary translation aims to convert all of the code of an executable file into code that runs on the target architecture without having to run the code first, as is done in dynamic binary translation. This is very difficult to do correctly, since not all the

code can be discovered by the translator. For example, some parts of the executable may be reachable only through indirect branches, whose value is known only at run-time.

Dynamic binary translation looks at a short sequence of code—typically on the order of a single basic block—then translates it and caches the resulting sequence. Code is only translated as it is discovered and when possible branch instructions are made to point to already translated and saved code. In some cases such as instruction set simulation, the target instruction set may be the same as the source instruction set, providing testing and debugging features such as instruction trace, conditional breakpoints and hot spot detection. While such implementations may provide debugging and software/hardware development support, they are not typically directed to fault tolerance or fault detection. Additionally, binary translation techniques typically do little to improve utilization of new features, instructions and/or resources.

Some compiler techniques have sought to directly target features of hardware. These techniques have been applied, for example, to FPGA technology. A compiler may take high level source code as an input and compile the source code to produce FPGA logic as an output. In general, such FPGA implementations may limit performance advantages otherwise available for example, from very-large-scale integration (VLSI) and their implementations may additionally be larger and/or heavier, and may require higher supply voltages.

To date, potential solutions to such under utilization, fault tolerance, performance and efficiency limiting issues have not been adequately explored.

BRIEF DESCRIPTION OF THE DRAWINGS

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

Figure 1 illustrates one embodiment of a processor system using hardware compilation and/or translation with fault detection and roll back functionality.

Figure 2 illustrates one embodiment of a multiprocessing system using hardware compilation and/or translation with fault detection and roll back functionality.

Figure 3 illustrates a flow diagram for one embodiment of a process to use hardware compilation and/or translation with fault detection and roll back functionality.

Figure 4 illustrates a flow diagram for an alternative embodiment of a process to use hardware compilation and/or translation with fault detection and roll back functionality.

Figure 5 illustrates a flow diagram for another alternative embodiment of a process to use hardware compilation and/or translation with fault detection and roll back functionality.

DETAILED DESCRIPTION

Hardware compilation and/or translation with fault detection and roll back functionality are disclosed. Compilation and/or translation logic receives programs encoded in one language, and encodes the programs into a second language including instructions to support processor features not encoded into the original language encoding of the programs. In one embodiment, an execution unit executes instructions of the second language including an operation-check instruction to perform a first operation and record the first operation result for a comparison, and an operation-test instruction to perform a second operation and a fault detection operation by comparing the second operation result to the recorded first operation result. In some embodiments, an execution unit executes instructions of the second language including commit instructions to record execution checkpoint states of registers mapped to architectural registers, and roll-back instructions to restore the registers mapped to architectural registers to previously recorded execution checkpoint states.

In the disclosure that follows, terms such as threads, programs, and processes may be used interchangeably to describe embodiments of instruction sequences or command sequences of programming languages or machine languages or both, or alternatively as the executing sequences themselves or as the processing devices executing the sequences. Through hardware compilation and/or translation, instructions to support new processor features may be used to improve performance, fault tolerance, security, virus detection, etc., by recompilation, translation or instrumentation of programs previously encoded in languages, which did not have support for these new processor features. It will be appreciated that some embodiments supporting hardware compilation and/or translation with roll back functionality may, without significantly increasing implementation costs, provide fault tolerance and/or fault detection features that are atypical in high-production processors. In extreme environments where fault tolerance is a highly desirable feature of computer systems, commercial processors with hardware compilation and/or translation with roll back functionality may provide fault tolerance and/or fault detection along with advantages of high level integration, while being smaller and lighter, and having better power efficiency than other alternatives such as FPGA technology. Such features are increasingly desirable since reducing the dimensions and increasing the number of

transistors in a package increases the probability of faults due to alpha particles and other causal factors.

These and other embodiments of the present invention may be realized in accordance with the following teachings and it should be evident that various modifications and changes may be made in the following teachings without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense and the invention measured only in terms of the claims and their equivalents.

Figure 1 illustrates one embodiment of a processor 105 using hardware compilation and/or translation with fault detection and roll back functionality. Processor 105 includes multiprocessing or multithreading hardware, thread 110 and thread 120 comprising respectively execution unit 112 and register map 113, and execution unit 122 and register map 123.

A compilation and/or translation logic 130 receives (e.g. via a bus 117) executable program portions, thread T1 131 and thread T2 132 encoded in one language, and encodes them into a second language to include instructions to support one or more processor 105 features not encoded into the original language encoding of executable program portions, thread T1 131 and thread T2 132. In some alternative embodiments, the original language may be, for example, a high level programming language, or an interpreted language, or a pseudo-language, or a machine language of a different processor architecture, or even the same or a similar machine language, for example of a legacy processor. In some embodiments, the original language may be generated by software or middleware from a high level language into the pseudo-language, or the machine language of a different processor architecture, or into the same or similar machine language for a legacy processor. The executable program portions, T1' 141 and T2' 142, encoded in the second language are stored, for example in Icache 140. In some embodiments, the second language may include new instructions to support new processor 105 features but may also have a base of instructions executable by processors with different micro-architectures which share at least a portion of a common instruction set. For example, Intel® Pentium 4 processors, Intel® Core™ processors, and processors from Advanced Micro Devices, Inc. of Sunnyvale CA implement nearly identical versions of the x86 instruction set (with some extensions that have been added with newer versions), but have different internal designs. Similarly, processors designed by other processor development

companies, such as ARM Holdings, Ltd., MIPS, or their licensees or adopters, may share at least a portion a common instruction set, but may include different processor designs.

Responsive to one embodiment of instructions in said second language to support new processor 105 features, execution unit 112 can execute a commit instruction to record a first execution checkpoint state of registers mapped to architectural registers (for example registers in registers A 151 mapped to R1 and R2 by register map 113 may be recorded in registers B 152 as an execution checkpoint state, while temporary storage in registers C 153 mapped to Rn by register map 113 is not recorded in the execution checkpoint state) of the executable program portion, for example T1' 141. Execution unit 122 can execute another commit instruction to record an execution checkpoint state of registers mapped to architectural registers (for example registers in registers A 151 mapped to R1 and R3 by register map 123 may be recorded in registers B 152 as an execution checkpoint state, while temporary storage in registers C 153 mapped to R4 and Rn by register map 123 are not recorded in the execution checkpoint state) of the executable program portion, for example T2' 142. Also responsive to instructions in said second language to support new processor 105 features, execution unit 112 and/or execution unit 122 can execute a roll-back instruction if necessary to restore, respectively, the registers mapped to architectural registers to the most recent execution checkpoint states respectively recorded.

As one example of new processor 105 features, embodiments of thread 110 and thread 120 of processor 105 are capable of executing a series of instructions and making the net effect of the entire series visible to another process at a specific time. Thus from another thread's point of view, the series of instructions appears to have been executed atomically. In order to create this illusion, processor 105 needs to be able to roll back threads to a previous checkpoint state before the atomic execution began. There are any number of reasons why thread 110 or thread 120 may need to roll back an atomic region, e.g. to avoid deadlock due to entangled memory references, or to execute a piece of code that has been unsafely optimized with assert checks, or to correct for a fault due to alpha particles and other causal factors, or to resolve precise exceptions, or to handle interrupts etc. In some embodiments of thread 110 and thread 120 of processor 105, it may not be necessary to read or to write registers from different threads in the same cycle. It will, therefore, be appreciated that a register design may comprise a unified storage cell, containing storage cells from two different threads, but restricting access to one or the other thread and thereby simplifying the storage cell design. According to such

embodiments, a storage cell design may contain three actual storage cells, e.g. from A 151, B 152 and C 153. A thread may read from A 151 or B 152, write to A 151 or B 152, move from A 151 to B 152 or move from B 152 to A 151. The lower portion of a thread's register set may be shadowed, such that it is always possible for that subset of registers (the canonical set) to roll back to a previous checkpoint state. Accordingly for some embodiments register indices may be mapped into the storage structure such that canonical registers are mapped, for example, to A 151 and to B 152 for some storage cells and the non-canonical (or temporary) registers are mapped to C 153 for some other storage cells.

Responsive to another embodiment of instructions in the second language to support new processor 105 features, execution unit 112 (or execution unit 122) can execute an operation-check instruction (such as an add-check, subtract-check, multiply-check, compute-effective-address-check, etc.) to perform an operation (such as an add, subtract, multiply, compute-effective-address, etc.) and record the operation result for a comparison using check logic 111 (or check logic 121). Execution unit 112 or execution unit 122 can also execute an operation-test instruction to perform a second operation (such as an add, subtract, multiply, compute-effective-address, etc.) and to perform fault detection by comparing the second operation result to the previously recorded operation result. For some embodiments a trap or an interrupt may be taken, for example upon detecting an inequality or upon exceeding a maximum value or upon some other condition. Alternatively a flag may be set to conditionally execute a roll-back instruction if necessary to restore the registers mapped to architectural registers to the most recent execution checkpoint state recorded prior to detection of the fault. In some embodiments operation-check instructions and operation-test instructions in the second language may be implemented by providing a check indicator and a test indicator to any operation or to a subset of operations performed by instructions of the second language. In such embodiments, a check latch in check logic 111 or in check logic 121 may be loaded with the output of the operation (along with loading of the normal output latch) whenever the check indicator is set for an operation. Similarly whenever the test indicator is set for an operation, the contents of the normal output latch may be compared to the contents of the check latch in check logic 111 or in check logic 121 to detect a fault or some other condition. A check latch may be added to an execution unit of virtually any type to enable fault detection, security checks, virus detection, debugging, etc.

Figure 2 illustrates one embodiment of a multiprocessing system 201 using hardware compilation and/or translation with fault detection and roll back functionality.

System 201 includes memory 260 and processor 205. Processor 205 includes multiprocessing or multithreading hardware, thread 210 and thread 220 comprising respectively execution unit 212 and register map 213, and execution unit 222 and register map 223. A compilation and/or translation logic 230 receives (e.g. via buses 217 and 267 from memory 260) executable program portions, thread T1 261 and thread T2 262 encoded in one language, and encodes them into a second language to include instructions to support one or more processor 205 features not encoded into the original language encoding of executable program portions, thread T1 261 and thread T2 262. In some alternative embodiments, the original language may be, for example, a high level programming language, or an interpreted language, or a pseudo-language, or a machine language of a different processor architecture, or even the same or a similar machine language, for example of a legacy processor. The executable program portions, thread T1' 241 and thread T2' 242, encoded in the second language are stored, for example in memory 260 and in Icache 240. In some embodiments, the second language may include new instructions to support new processor 205 features but may also have a base of instructions executable by processors with different micro-architectures which share at least a portion of a common instruction set. Responsive to one embodiment of instructions in said second language to support new processor 205 features, execution unit 212 can execute a commit instruction to record a first execution checkpoint state of registers mapped to architectural registers (for example registers in registers A 251 mapped to R1 and R2 by register map 213 may be recorded in registers B 252 as an execution checkpoint state, while temporary storage in registers C 253 mapped to Rn by register map 213 is not recorded in the execution checkpoint state) of the executable program portion, for example T1' 241. Execution unit 222 can execute another commit instruction to record an execution checkpoint state of registers mapped to architectural registers (for example registers in registers A 251 mapped to R1 and R3 by register map 223 may be recorded in registers B 252 as an execution checkpoint state, while temporary storage in registers C 253 mapped to R4 and Rn by register map 223 are not recorded in the execution checkpoint state) of the executable program portion, for example T2' 242. Also responsive to instructions in said second language to support new processor 205 features, execution unit 212 and/or execution unit 222 can execute a roll-back instruction if necessary to restore, respectively, the registers mapped to architectural registers to the most recent execution checkpoint states respectively recorded.

Responsive to another embodiment of instructions in the second language to support new processor 205 features, execution unit 212 (or execution unit 222) can execute an operation-check instruction (such as an add-check, subtract-check, multiply-check, compute-effective-address-check, etc.) to perform an operation (such as an add, subtract, multiply, compute-effective-address, etc.) and record the operation result for a comparison using check logic 211 (or check logic 221). Execution unit 212 or execution unit 222 can also execute an operation-test instruction to perform a second operation (such as an add, subtract, multiply, compute-effective-address, etc.) and to perform fault detection by comparing the second operation result to the previously recorded operation result. For some embodiments a trap or an interrupt may be taken, for example upon detecting an inequality or upon exceeding a maximum value or upon some other condition. Alternatively a flag may be set to conditionally execute a roll-back instruction if necessary to restore the registers mapped to architectural registers to the last execution checkpoint state recorded prior to detection of the fault.

Figure 3 illustrates a flow diagram for one embodiment of a process to use hardware compilation and/or translation with fault detection and roll back functionality. Process 301 and other processes herein disclosed are performed by processing blocks that may comprise dedicated hardware or software or firmware operation codes executable by general purpose machines or by special purpose machines or by a combination of both.

In processing block 310 an executable thread portion is received in a processor encoded in a first language. In processing block 320 the executable thread portion is encoded into a second language to include one or more instruction to support new processor features not supported by the first language encoding. In processing block 330 the executable thread portion is stored in the second language, for example in system memory 260 and/or in a processor cache 240. In processing block 340 an operation of the executable thread portion in the second language is executed to utilize the new processor features.

It will be appreciated that process 301 may be used to improve performance, fault tolerance, security, virus detection, etc., by recompilation, translation or instrumentation of programs previously encoded in languages, which did not have support for these new processor features. It will be appreciated that such programs may have been previously encoded in languages, including but not limited to a high level programming language, or an interpreted language, or a different machine language, or even the same or a similar machine language, for example for a legacy processor.

Figure 4 illustrates a flow diagram for an alternative embodiment of a process to use hardware compilation and/or translation with fault detection and roll back functionality. In processing block 410 an executable thread portion is received in a processor encoded in a first language, for example, a high level programming language, or an interpreted language, or a machine language of a different processor architecture, or even the same or a similar machine language, i.e. for a legacy processor. In processing block 420 the executable thread portion is encoded into a second language to include one or more instruction to support new processor features not supported by the first language encoding. In processing block 430 the executable thread portion is stored in the second language, for example in system memory 260 and/or in a processor cache 240.

In processing block 440 a commit operation of the second language is executed to utilize the new processor features including recording an execution checkpoint state of registers, which are mapped to architectural registers. In processing block 550 a determination is made whether or not the registers mapped to architectural registers should be restored to a previous execution checkpoint state. If so, in processing block 560 a roll-back operation of the second language is executed to utilize the new processor features including restoring the previous execution checkpoint state, and processing proceeds until another commit operation is executed in processing block 440.

Otherwise in processing block 470 execution of the thread portion in the second language continues until it finishes in processing block 480. Then in processing block 490 another commit operation is executed to record another execution checkpoint state of registers, which are mapped to architectural register.

It will be appreciated that process 401 may, without significantly increasing implementation costs, provide fault tolerance and/or fault detection features that are atypical in high-production processors. In extreme environments where fault tolerance is a highly desirable feature of computer systems, commercial processors with hardware compilation and/or translation with roll back functionality may provide fault tolerance and/or fault detection along with advantages of high level integration, while being smaller and lighter, and having better power efficiency than other alternatives such as FPGA technology. Such features are increasingly desirable since reducing the dimensions and increasing the number of transistors in a package increases the probability of faults due to alpha particles and other causal factors.

Figure 5 illustrates a flow diagram for another alternative embodiment of a process 501 to use hardware compilation and/or translation with fault detection and roll

back functionality. In processing block 510 an executable thread portion is received in a processor encoded in a first language, for example, a high level programming language, or an interpreted language, or a machine language of a different processor architecture, or even the same or a similar machine language, i.e. for a legacy processor. In processing block 520 the executable thread portion is encoded into a second language to include one or more instruction to support new processor features not supported by the first language encoding. In processing block 530 the executable thread portion is stored in the second language, for example in system memory and/or in a processor cache.

In processing block 540 a commit operation is executed recording an execution checkpoint state of registers, which are mapped to architectural registers. In processing block 544 an operation-check instruction of the second language is executed to utilize the new processor features including recording an operation result for a comparison by a future operation, for example to detect a fault or virus. In processing block 548 an operation-test instruction of the second language is executed to utilize the new processor features including a fault detection operation by comparing an operation result to a previous operation result. In processing block 550 a determination is made whether or not the registers mapped to architectural registers should be restored to a previous execution checkpoint state. If so, in processing block 560 a roll-back operation is executed to restore the previous execution checkpoint state, and processing proceeds until another commit operation is executed in processing block 540.

Otherwise in processing block 570 execution of the thread portion in the second language continues until it finishes in processing block 580. Then in processing block 590 another commit operation is executed to record another execution checkpoint state of registers, which are mapped to the architectural registers.

Through hardware compilation and/or translation, instructions to support new processor features may be used to improve performance, or fault tolerance, security, or virus detection, etc., by recompilation, translation or instrumentation of programs previously encoded in languages, which did not have support for these new processor features. It will be appreciated that such programs may have been previously encoded in languages, including but not limited to a high level programming language, or an interpreted language, or a different machine language, or even the same or a similar machine language, for example for a legacy processor.

It will also be appreciated that some embodiments supporting hardware compilation and/or translation with roll back functionality may, without significantly

increasing implementation costs, provide fault tolerance and/or fault detection features that are atypical in high-production processors. In extreme environments where fault tolerance is a highly desirable feature of computer systems, commercial processors with hardware compilation and/or translation with roll back functionality may provide fault tolerance and/or fault detection along with advantages of high level integration, while being smaller and lighter, and having better power efficiency than other alternatives such as FPGA technology. Such features are increasingly desirable since reducing the dimensions and increasing the number of transistors in a package increases the probability of faults due to alpha particles and other causal factors.

The above description is intended to illustrate preferred embodiments of the present invention. From the discussion above it should also be apparent that especially in such an area of technology, where growth is fast and further advancements are not easily foreseen, the invention may be modified in arrangement and detail by those skilled in the art without departing from the principles of the present invention within the scope of the accompanying claims and their equivalents.

CLAIMS

What is claimed is:

1. A method for deploying processor features, the method comprising:

receiving an executable thread portion in a processor, the executable thread encoded in a first language;

encoding the executable thread portion into a second language to include one or more instruction to support one or more processor features not encoded into the first language encoding of the executable thread portion;

storing the executable thread portion in the second language; and

executing the executable thread portion in the second language to utilize said one or more processor features.

2. The method of Claim 1 wherein said one or more instruction to support one or more processor features comprises a roll-back instruction.

3. The method of Claim 2 wherein said one or more processor features supported by the roll-back instruction comprises restoring registers mapped to architectural registers to a previous execution checkpoint state.

4. The method of Claim 2 wherein said one or more instruction to support one or more processor features comprises a commit instruction.

5. The method of Claim 4 wherein said one or more processor features supported by the commit instruction comprises recording an execution checkpoint state of registers mapped to architectural registers.

6. The method of Claim 4 wherein said one or more processor features supported by the roll-back instruction and the commit instruction comprise speculative modification of state in registers mapped to architectural registers and restoration of those registers to a previous execution checkpoint state.

7. The method of Claim 6 wherein said one or more processor features supported by the roll-back instruction and the commit instruction comprise fault tolerance and fault detection.

8. The method of Claim 6 wherein said one or more processor features supported by the roll-back instruction and the commit instruction comprise virus detection.

9. The method of Claim 1 wherein said one or more instruction to support one or more processor features comprises an operation-test instruction.

10. The method of Claim 9 wherein said one or more processor features supported by the operation-test instruction comprises a fault detection operation comparing an operation result to a previous operation result.

11. The method of Claim 9 wherein said one or more instruction to support one or more processor features comprises an operation-check instruction.

12. The method of Claim 11 wherein said one or more processor features supported by the operation-check instruction comprises recording an operation result for a comparison by a future operation-test instruction.

13. An article of manufacture comprising:

a machine-accessible medium including data and instructions for hardware translation such that, when accessed by a machine, cause the machine to:

receive an executable thread portion in a processor, the executable thread encoded in a first language;

encode the executable thread portion into a second language to include one or more instruction to support one or more processor features not encoded into the first language encoding of the executable thread portion;

store the executable thread portion in the second language; and

execute the executable thread portion in the second language to utilize said one or more processor features.

14. The article of manufacture of Claim 13 wherein said one or more instruction to support one or more processor features comprises a roll-back instruction.

15. The article of manufacture of Claim 14 wherein said one or more processor features supported by the roll-back instruction comprises restoring registers mapped to architectural registers to a previous execution checkpoint state.

16. The article of manufacture of Claim 14 wherein said one or more instruction to support one or more processor features comprises a commit instruction.

17. The article of manufacture of Claim 16 wherein said one or more processor features supported by the commit instruction comprises recording an execution checkpoint state of registers mapped to architectural registers.

18. The article of manufacture of Claim 16 wherein said one or more processor features supported by the roll-back instruction and the commit instruction comprise speculative modification of state in registers mapped to architectural registers and restoration of those registers to a previous execution checkpoint state.

19. The article of manufacture of Claim 18 wherein said one or more processor features supported by the roll-back instruction and the commit instruction comprise fault tolerance and fault detection.

20. The article of manufacture of Claim 18 wherein said one or more processor features supported by the roll-back instruction and the commit instruction comprise virus detection.

21. A processor comprising:

a compilation/translation logic to receive an executable program portion encoded in a first language, and encode the executable program portion into a second language to include instructions to support one or more processor features not encoded into the first language encoding of the executable program portion;

an execution unit, responsive to said instructions of the second language to support the one or more processor features, to execute:

an operation-check instruction to perform a first operation and record the first operation result for a comparison, and

an operation-test instruction to perform a second operation and a fault detection by comparing the second operation result to the recorded first operation result.

22. The processor of Claim 21 wherein said execution unit, responsive to said instructions of the second language to support the one or more processor features, is also to execute:

a commit instruction to record an execution checkpoint state of registers mapped to architectural registers of the processor, and

a roll-back instruction to restore the registers mapped to architectural registers of the processor to the execution checkpoint state.

23. A multiprocessor comprising:

a compilation/translation logic to receive a first executable program portion encoded in a first language, and encode the first executable program portion into a second language to include instructions to support one or more processor features not encoded into the first language encoding of the first executable program portion;

a first execution unit, responsive to said instructions of the second language to support the one or more processor features, to execute:

a first commit instruction to record a first execution checkpoint state of registers mapped to architectural registers of the first executable program portion, and

a roll-back instruction to restore the registers mapped to architectural registers of the first executable program portion to the first execution checkpoint state.

24. The multiprocessor of Claim 23 wherein said compilation/translation logic is to receive a second executable program portion encoded in the first language, and encode the second executable program portion into the second language to include the instructions to support the one or more processor features not encoded into the first language encoding of the second executable program portion, the multiprocessor comprising:

a second execution unit, responsive to said instructions of the second language to support the one or more processor features, to execute:

a second commit instruction to record a second execution checkpoint state of registers mapped to architectural registers of the second executable program portion, and

a second roll-back instruction to restore the registers mapped to architectural registers of the second executable program portion to the second execution checkpoint state.

25. The multiprocessor of Claim 24 wherein the registers mapped to architectural registers of the first executable program portion, and the registers mapped to architectural registers of the second executable program portion are allocated from the same physical register file in the multiprocessor.

26. The multiprocessor of Claim 25 wherein said first and second execution units, responsive to said instructions of the second language to support the one or more processor features, are also to execute:

an operation-check instruction to perform a first operation and record the first operation result for a comparison, and

an operation-test instruction to perform a second operation and a fault detection by comparing the second operation result to the recorded first operation result.

27. A computing system comprising:

an addressable memory to store a first executable program portion encoded in a first language; and

a multiprocessor including:

a compilation/translation logic to receive the first executable program portion and encode the first executable program portion into a second language to include instructions to support one or more processor features not encoded into the first

language encoding of the first executable program portion; and

a first execution unit, responsive to said instructions of the second language to support the one or more processor features, to execute:

a first commit instruction to record a first execution checkpoint state of registers mapped to architectural registers of the first executable program portion, and

a roll-back instruction to restore the registers mapped to architectural registers of the first executable program portion to the first execution checkpoint state.

28. The system of Claim 27, said addressable memory to store a second executable program portion encoded in the first language, wherein said compilation/translation logic is to receive the second executable program portion and encode the second executable program portion into the second language to include the instructions to support the one or more processor features not encoded into the first language encoding of the second executable program portion, the multiprocessor comprising:

a second execution unit, responsive to said instructions of the second language to support the one or more processor features, to execute:

a second commit instruction to record a second execution checkpoint state of registers mapped to architectural registers of the second executable program portion, and

a second roll-back instruction to restore the registers mapped to architectural registers of the second executable program portion to the second execution checkpoint state.

29. The system of Claim 28 wherein said first and second execution units, responsive to said instructions of the second language to support the one or more processor features, are also to execute:

an operation-check instruction to perform a first operation and record the first operation result for a comparison, and

an operation-test instruction to perform a second operation and a fault detection by comparing the second operation result to the recorded first operation result.

30. The system of Claim 29 wherein the registers mapped to architectural registers of the first executable program portion, and the registers mapped to architectural registers of the second executable program portion are allocated from the same physical register file in the multiprocessor.

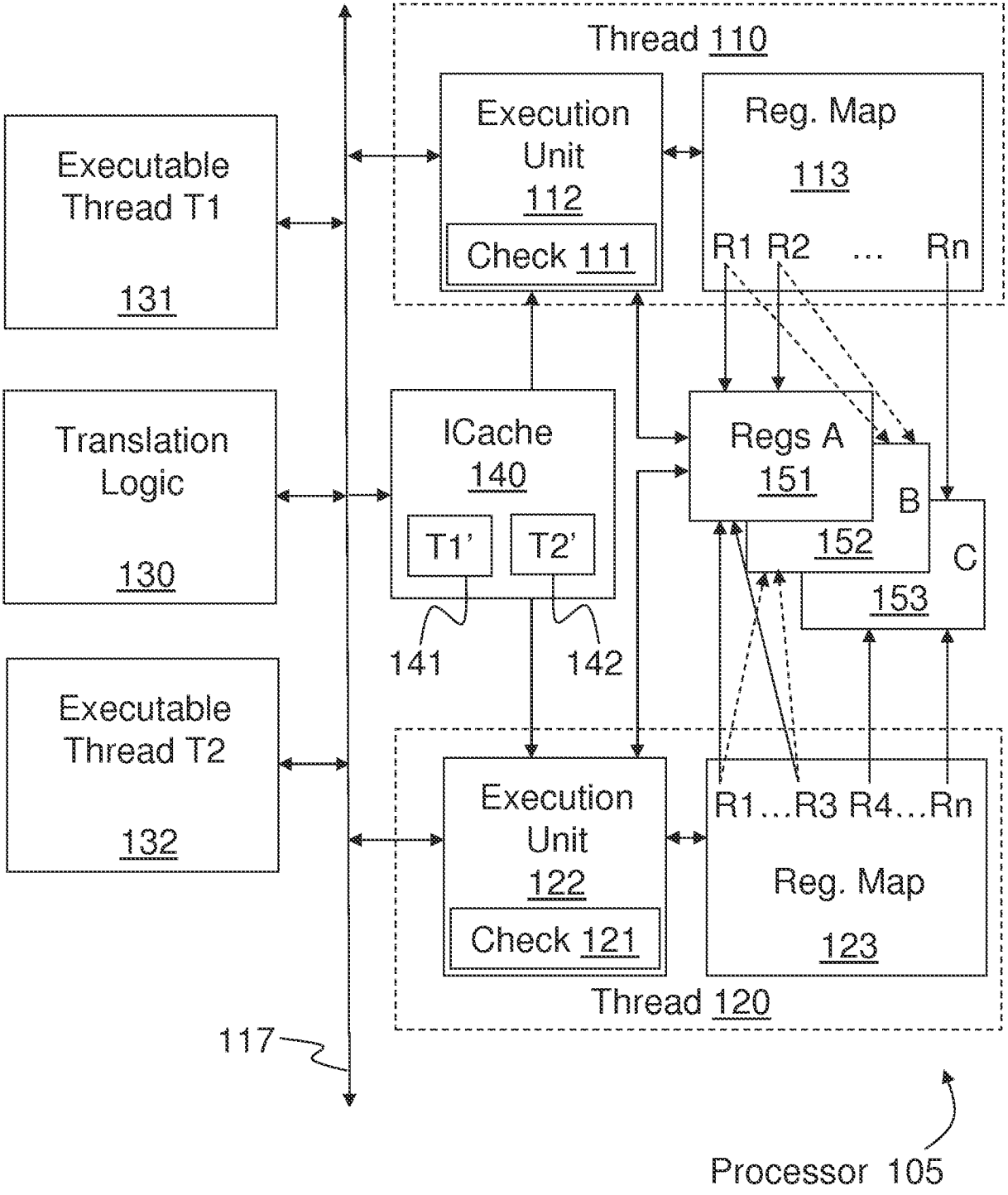

Thread 110
Execution Unit 112
Check 111
Reg. Map 113
R1 R2 ... Rn
Executable Thread T1 131
Translation Logic 130
Executable Thread T2 132
ICache 140
T1'
T2'
141
142
Regs A 151
B
152
C
153
Execution Unit 122
Check 121
R1...R3 R4...Rn
Reg. Map 123
Thread 120
117
Processor 105

FIG. 1

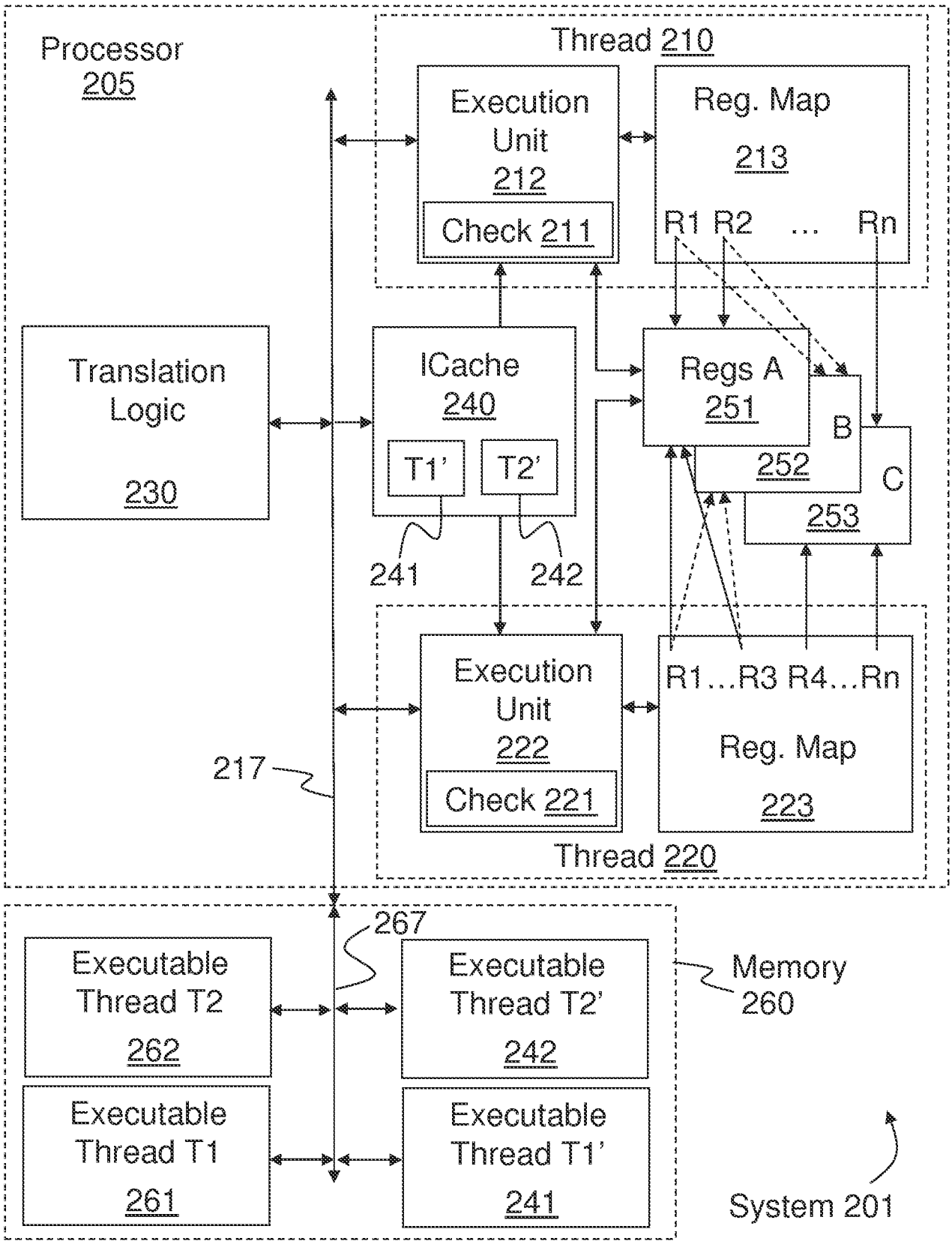

Processor
205
Thread 210
Execution
Unit
212
Check 211
Reg. Map
213
R1 R2 ... Rn
Translation
Logic
230
ICache
240
T1'
T2'
241
242
Regs A
251
B
252
C
253
Execution
Unit
222
Check 221
R1…R3 R4…Rn
Reg. Map
223
Thread 220
217
267
Executable
Thread T2
262
Executable
Thread T2'
242
Memory
260
Executable
Thread T1
261
Executable
Thread T1'
241
System 201

FIG. 2

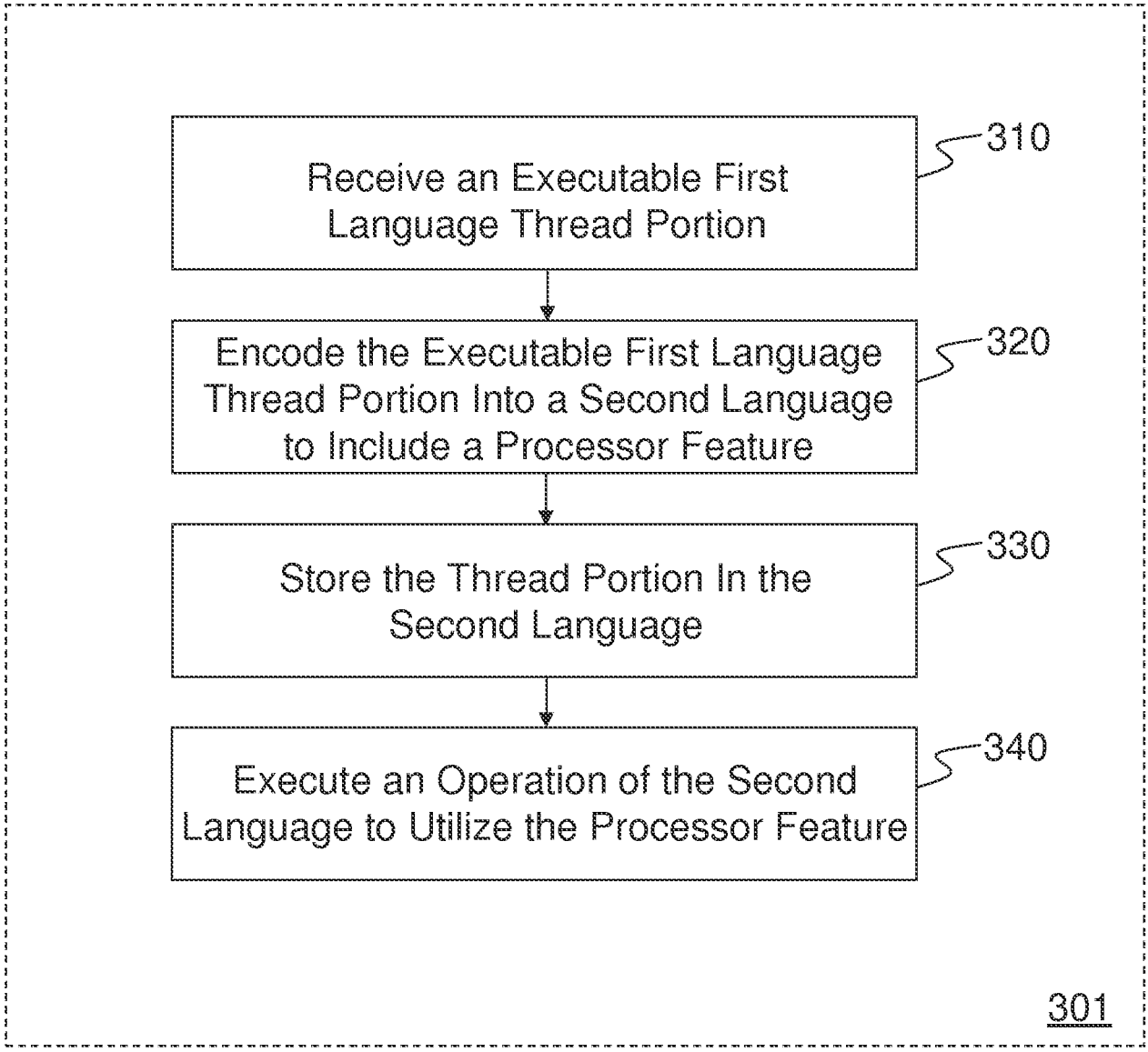

Receive an Executable First Language Thread Portion
310
Encode the Executable First Language Thread Portion Into a Second Language to Include a Processor Feature
320
Store the Thread Portion In the Second Language
330
Execute an Operation of the Second Language to Utilize the Processor Feature
340
301

FIG. 3

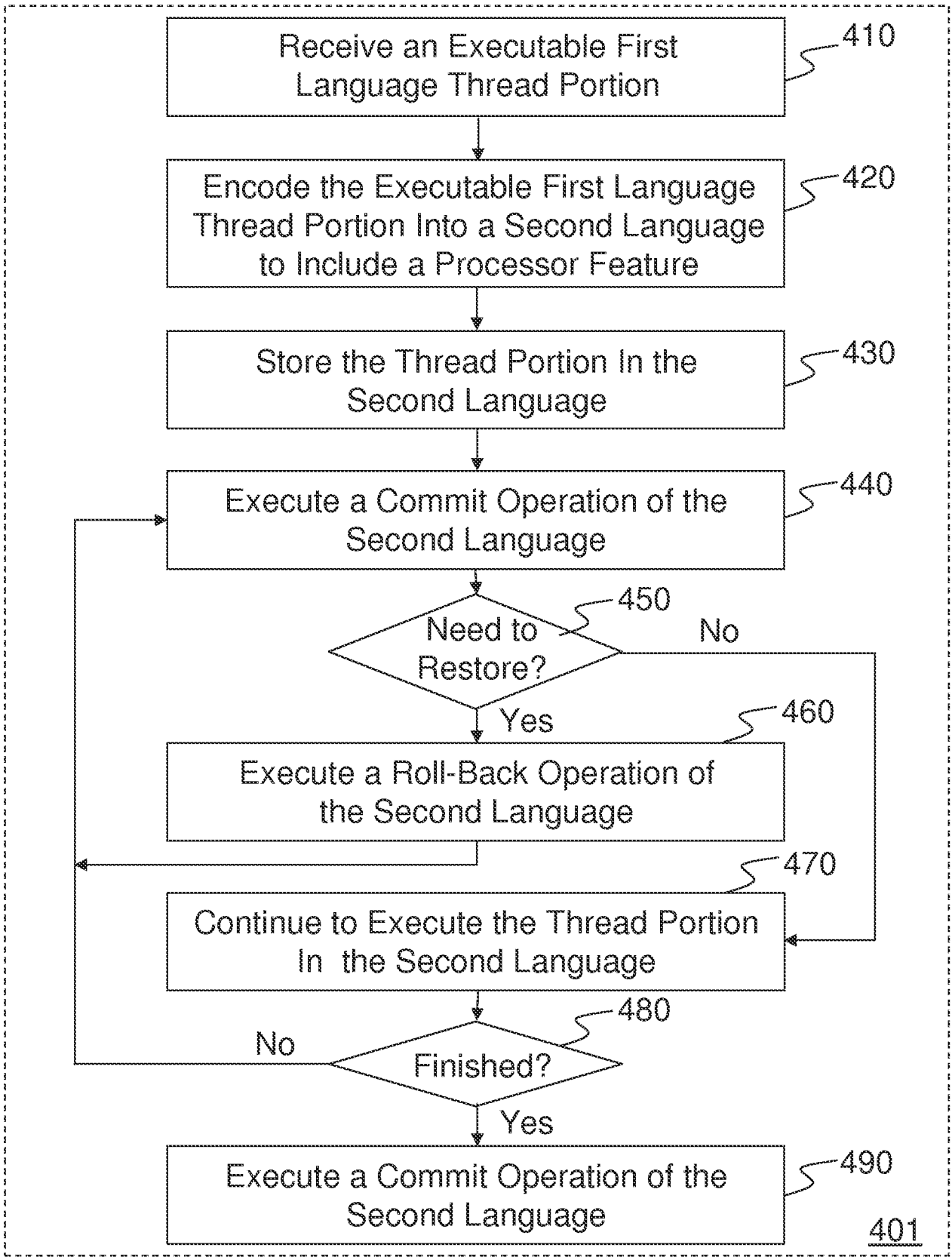

Receive an Executable First Language Thread Portion
410
Encode the Executable First Language Thread Portion Into a Second Language to Include a Processor Feature
420
Store the Thread Portion In the Second Language
430
Execute a Commit Operation of the Second Language
440
Need to Restore?
450
No
Yes
Execute a Roll-Back Operation of the Second Language
460
Continue to Execute the Thread Portion In the Second Language
470
Finished?
480
No
Yes
Execute a Commit Operation of the Second Language
490
401

FIG. 4

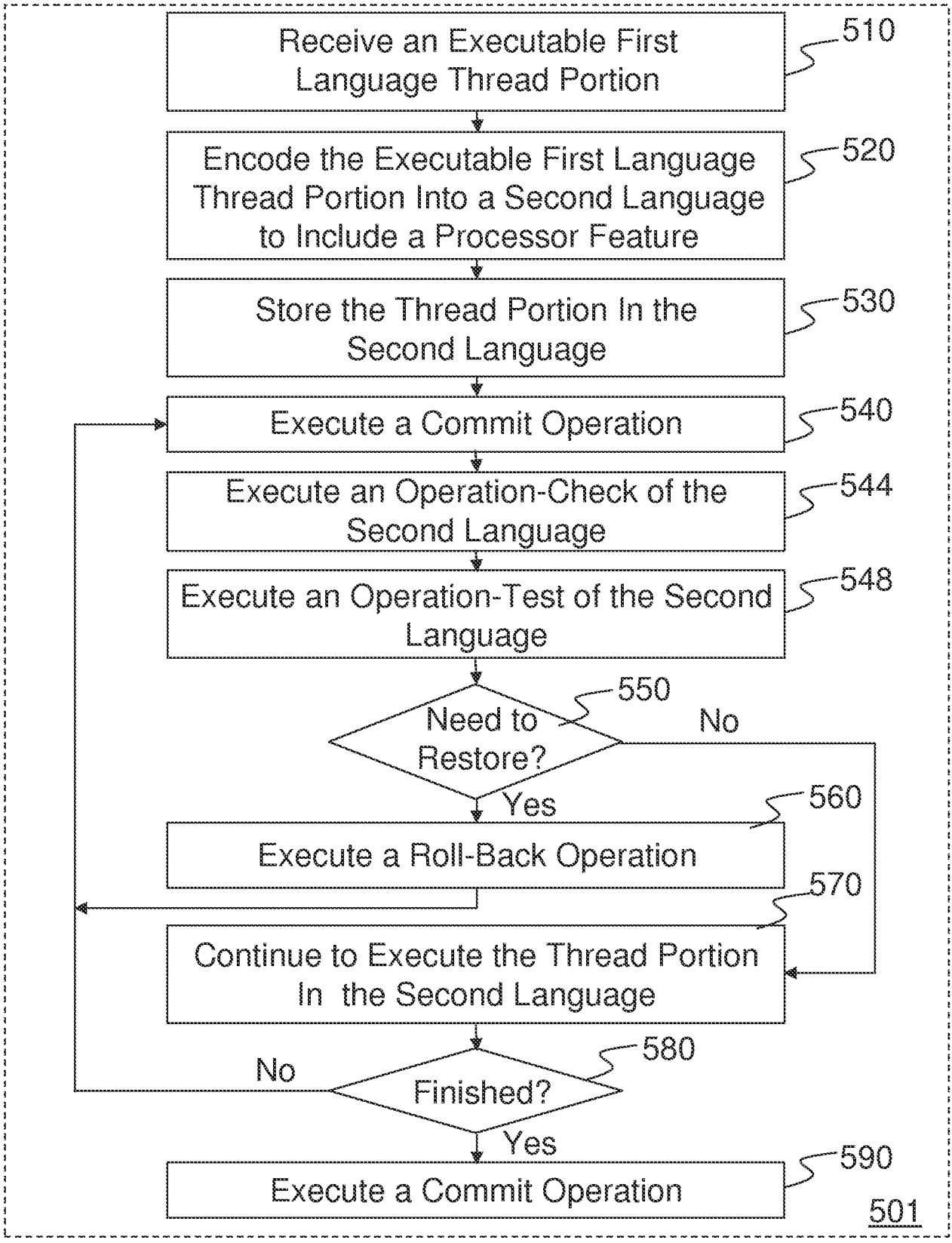

Receive an Executable First Language Thread Portion
510
Encode the Executable First Language Thread Portion Into a Second Language to Include a Processor Feature
520
Store the Thread Portion In the Second Language
530
Execute a Commit Operation
540
Execute an Operation-Check of the Second Language
544
Execute an Operation-Test of the Second Language
548
Need to Restore?
550
No
Yes
Execute a Roll-Back Operation
560
Continue to Execute the Thread Portion In the Second Language
570
Finished?
580
No
Yes
Execute a Commit Operation
590
501

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2012/071668

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/06(2006.01)i, G06F 9/30(2006.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 9/06; G06F 9/30; G06F 15/80; G06F 9/38

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: compilation, translation, roll back, fault detection, encoding

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2010-0146241 A1 (RAKIB SHL0M0 SELIM et al.) 10 June 2010 See abstract, figures 1-5, paragraph 0028-0060.	1-30
A	US 2011-0283089 A1 (INGALE HARSHAL) 17 November 2011 See abstract, figures 3, 7, paragraph 0007-0013, paragraph 0051-0052, paragraph 0066-0069.	1-30
A	US 6988183 B1 (DEREK CHI-LAN WONG) 17 January 2006 See abatract, figures 1-3, claims 1-5,12,15-17,22.	1-30
A	US 2006-0242385 A1 (KAZUAKI MURAKAMI et al.) 26 October 2006 See abstract, figures 1-2, claims 18-28, 38-42.	1-30

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:

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

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 April 2013 (25.04.2013)	**26 April 2013 (26.04.2013)**

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

HEO, Young Han

Telephone No. 82-42-481-8456

Form PCT/ISA/210 (second sheet) **(July** 2009)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2012/071668

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2010-0146241 A1	10.06.2010	None	
US 2011-0283089 A1	17.11.2011	WO 2010-086871 A1	05.08.2010
US 6988183 B1	17.01.2006	None	
US 2006-0242385 A1	26.10.2006	CN 101111834 AO	23.01.2008
		EP 1836601 A2	26.09.2007
		JP 04-390211 B2	16.10.2009
		JP 2006-185423 A	13.07.2006
		KR 10-2007-0097051 A	02.10.2007
		WO 2006-059775 A2	08.06.2006
		WO 2006-059775 A3	08.06.2006