РОССИЙСКАЯ ФЕДЕРАЦИЯ

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(19) **RU** (11) **2 520 411** (13) **C2**

(51) МПК
G06F 9/50 (2006.01)

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21)(22) Заявка: **2012141606/08, 17.02.2011**

(24) Дата начала отсчета срока действия патента:
17.02.2011

Приоритет(ы):
(30) Конвенционный приоритет:
01.03.2010 US 12/659,234

(43) Дата публикации заявки: **10.04.2014** Бюл. № 10

(45) Опубликовано: **27.06.2014** Бюл. № 18

(56) Список документов, цитированных в отчете о поиске: **US 2009/0213673 A1 (ARM LIMITED), 27.08.2009. US 2005/0232261 A1 (SAMSUNG ELECTRONICS CO.), 20.10.2005. US 6,501,999 B1 (INTEL CORPORATION), 31.12.2002. RU 2006 118 704 A (СОНИ КОРПОРЕЙШН), 20.12.2007. RU 2 182 353 C2 (АРМ ЛИМИТЕД), 10.05.2002**

(85) Дата начала рассмотрения заявки PCT на национальной фазе: **01.10.2012**

(86) Заявка PCT:
GB 2011/050317 (17.02.2011)

(87) Публикация заявки PCT:
WO 2011/107776 (09.09.2011)

Адрес для переписки:
129090, Москва, ул. Б. Спасская, 25, строение 3, ООО "Юридическая фирма Городисский и Партнеры"

(72) Автор(ы):
ГРИНХОЛ Питер Ричард (GB),
ГРАЙЗЕНТУЭЙТ Ричард Рой (GB)

(73) Патентообладатель(и):
АРМ ЛИМИТЕД (GB)

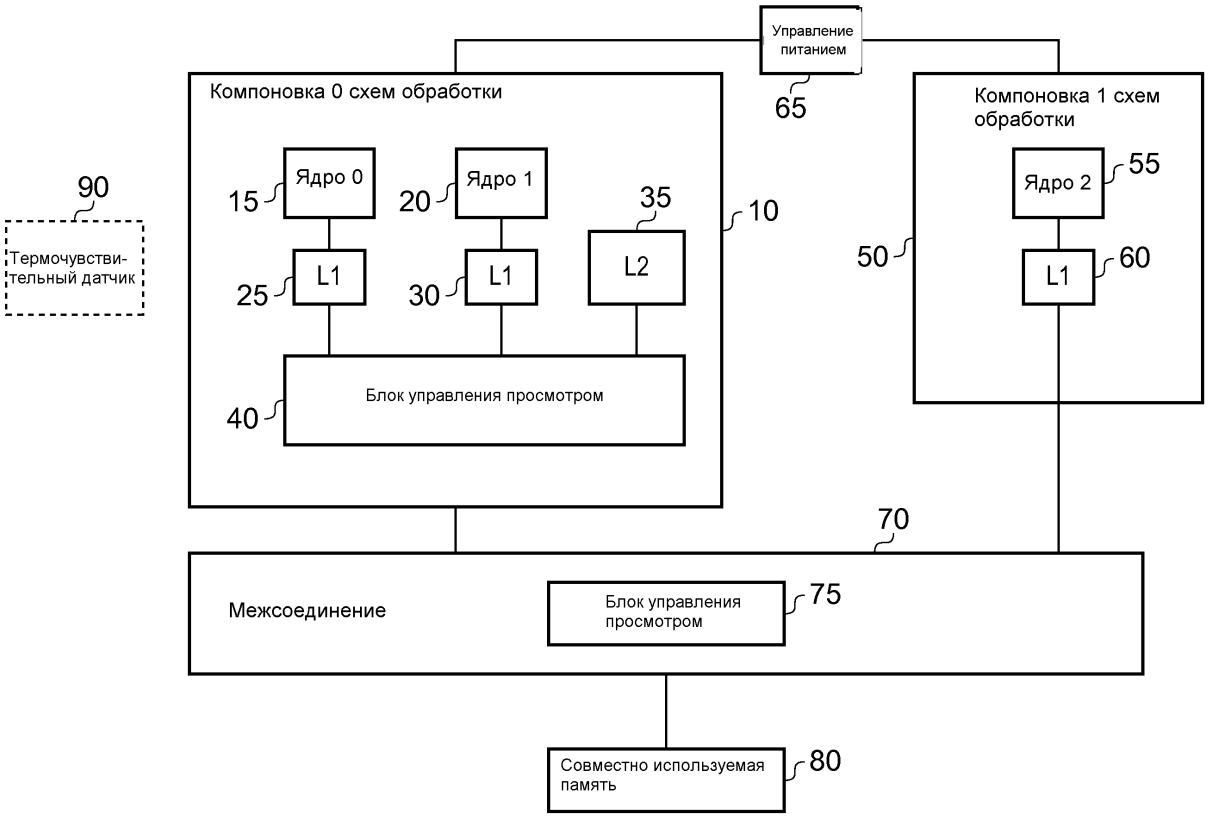
RU 2520411 C2

(54) УСТРОЙСТВО ОБРАБОТКИ ДАННЫХ И СПОСОБ ПЕРЕКЛЮЧЕНИЯ РАБОЧЕЙ НАГРУЗКИ МЕЖДУ ПЕРВОЙ И ВТОРОЙ КОМПОНОВКОЙ СХЕМ ОБРАБОТКИ

(57) Реферат:

Изобретение относится к устройству обработки данных и способу переключения рабочей нагрузки между первой и второй компоновкой схем обработки. Техническим результатом является повышение эффективности использования энергии устройством обработки данных. Устройство обработки данных содержит первую компоновку схем обработки, которая является архитектурно совместимой со второй компоновкой схем обработки, но при этом первая компоновка схем обработки отличается с точки зрения микроархитектуры от второй компоновки схем обработки. Во время операции передачи обслуживания, контроллер переключения обеспечивает предоставление исходной компоновкой схем обработки своего текущего состояния архитектуры целевой компоновке схем обработки, причем текущим состоянием

архитектуры является то состояние, которое не доступно из совместно используемой памяти в момент инициирования операции передачи обслуживания, и которое является необходимым целевой компоновке схем обработки, чтобы успешно принять на себя выполнение рабочей нагрузки из исходной компоновки схем обработки. 3 н. и 17 з.п. ф-лы, 19 ил.



Фиг. 1

RU 2520411 C2

RU 2520411 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 520 411** (13) **C2**

(51) Int. Cl.
G06F 9/50 (2006.01)

(12) ABSTRACT OF INVENTION

(21)(22) Application: **2012141606/08, 17.02.2011**

(24) Effective date for property rights:
17.02.2011

Priority:
(30) Convention priority:
01.03.2010 US 12/659,234

(43) Application published: **10.04.2014** Bull. № **10**

(45) Date of publication: **27.06.2014** Bull. № **18**

(85) Commencement of national phase: **01.10.2012**

(86) PCT application:
GB 2011/050317 (17.02.2011)

(87) PCT publication:
WO 2011/107776 (09.09.2011)

Mail address:
129090, Moskva, ul. B. Spasskaja, 25, stroenie 3, OOO "Juridicheskaja firma Gorodisskij i Partnery"

(72) Inventor(s):
GRINKhOL Piter Richard (GB),
GRAJZENTUEhJT Richard Roj (GB)

(73) Proprietor(s):
ARM LIMITED (GB)

RU 2520411 C2

(54) DATA PROCESSING APPARATUS AND METHOD OF SWITCHING WORKLOAD BETWEEN FIRST AND SECOND PROCESSING CIRCUITRY

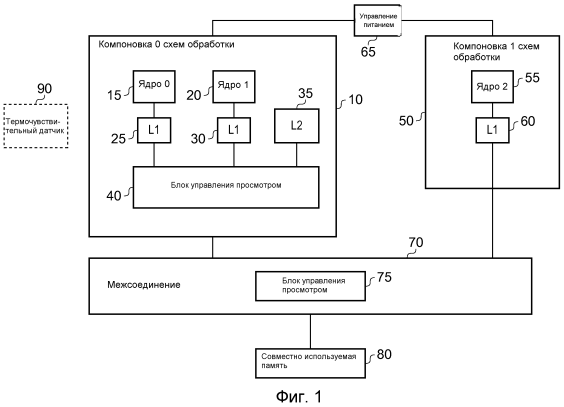
(57) Abstract:

FIELD: physics, computer engineering.

SUBSTANCE: invention relates to a data processing apparatus and a method of switching a workload between first and second processing circuitry. The data processing apparatus has first processing circuitry which is architecturally compatible with second processing circuitry, but with the first processing circuitry being micro-architecturally different from the second processing circuitry. During handover operation, the switch controller is enables to cause the source processing circuitry to make its current architectural state available to the destination processing circuitry, the current architectural state being that state which is not available from shared memory at a time the handover operation is initiated, and which is necessary for the destination processing circuitry to successfully take over performance of the workload from the source processing circuitry.

EFFECT: high efficiency of using power by a data processing apparatus.

20 cl, 19 dwg



Фиг. 1

RU 2520411 C2

Область техники, к которой относится изобретение

Настоящее изобретение относится к устройству обработки данных и способу переключения рабочей нагрузки между первой и второй компоновкой схем обработки, и в частности к способу выполнения указанного переключения для повышения эффективности использования энергии устройством обработки данных.

Предшествующий уровень техники

В современных системах обработки данных, различие в требовании к производительности между задачами, требующими большой производительности, например функционирование игр, и задачами, требующими малой производительности, например воспроизведение MP3-файлов, может превышать отношение 100:1. В случае использования одного процессора для всех задач, этот процессор должен иметь высокую производительность, но для микроархитектуры процессоров существует аксиома, что высокопроизводительные процессоры менее эффективно используют энергию, чем малопроизводительные процессоры. Известно, что для увеличения эффективности использования энергии на уровне процессора с использованием таких способов, как Динамическое масштабирование напряжения и частоты (Dynamic Voltage and Frequency Scaling, DVFS) или выборочная подача питания для обеспечения процессора диапазоном уровней производительности и соответствующими характеристиками потребления энергии. Однако обычно указанных способов становится недостаточно для обеспечения возможности одному процессору брать на себя задачи с указанным расхождением в требованиях к производительности.

Соответственно, предложено использование многоядерной архитектуры для обеспечения системы с эффективным использованием энергии для выполнения таких различных задач. В то время как, с обеспечением возможности разным ядрам параллельно выполнять разные задачи для увеличения пропускной способности, многоядерные системы уже в течение некоторого времени используются для увеличения производительности, исследование того, как такие системы могут использоваться для повышения эффективности использования энергии, является последним достижением техники.

В статье "Towards Better Performance Per Watt in Virtual Environments on Asymmetric Single-ISA Multi-Core Systems", V Kumar и др., ACM SIGOPS Operating Systems Review, Volume 43, Issue 3 (July 2009) обсуждаются многоядерные системы с Асимметричной единой архитектурой набора команд (Asymmetric Single Instruction Set Architecture, ASISA), состоящие из нескольких ядер, предоставляющих идентичную архитектуру набора команд (ISA), но отличающихся особенностями, сложностью, потреблением энергии и производительностью. В этой статье исследуются свойства виртуализированных рабочих нагрузок для понимания того, как эти рабочие нагрузки должны планироваться в системах ASISA, чтобы улучшить производительность и потребление энергии. В этой статье указывается, что определенные задачи больше подходят для микроархитектур с высокой частотой/производительностью (обычно задачи, требующие большой вычислительной мощности), в то время как другие больше подходят для микроархитектур с меньшей частотой/производительностью, и как побочный эффект потребляют меньше энергии (обычно задачи, требующие производительности ввода/вывода). Несмотря на то что эти исследования показывают то, как можно использовать системы ASISA для выполнения различных задач при эффективном использовании энергии, по-прежнему существует потребность в обеспечении механизма для планирования отдельных задач для более подходящих процессоров, и такое управление планированием обычно является существенной нагрузкой на операционную систему.

В статье "Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction", R Kumar и др., Proceedings of the 36th International Symposium of Microarchitecture (MICRO-36'03) обсуждается многоядерная архитектура, в которой все ядра исполняют идентичный набор команд, но имеют разные возможности и уровни производительности. Во время выполнения, системное программное обеспечение оценивает требования к ресурсам приложения и выбирает ядро, которое лучше других удовлетворяет этим требованиям наряду с тем, что минимизирует потребление энергии. Как обсуждается в разделе 2 этой статьи, во время исполнения приложения, программное обеспечение операционной системы сопоставляет это приложение с разными ядрами, чтобы подобрать ядро, удовлетворяющее определенному критерию выбора, например конкретному требованию к производительности. В разделе 2.3 отмечается, что существуют затраты на переключение ядер, которые неизбежно влекут за собой ограничение степени разбиения переключения. Далее обсуждается конкретный пример, в котором, если операционная система на основе логики принимает решение о переключении, то она включает питание нового ядра, запускает сброс кэша и сохраняет все измененные данные кэша в совместно используемой структуре памяти, и после этого подает сигнал в новое ядро для запуска в предопределенной точке входа в операционной системе. Питание старого ядра может после этого быть выключено, в то время как новое ядро осуществляет выборку данных из памяти. Такой подход описывается в разделе 2.3 как обеспечивающий возможность переключения приложения между ядрами операционной системой. В остальной части статьи обсуждается то, как динамически можно выполнять указанное переключение в условиях многоядерной окружающей среды с целью сокращения потребления энергии.

Несмотря на то что в вышеупомянутой статье обсуждается потенциальная возможность обеспечения сокращения потребления энергии посредством неоднородных многоядерных архитектур с единой ISA, по-прежнему требуется обеспечить операционную систему достаточной функциональностью для обеспечения возможности принятия решений по планированию отдельных приложений. При переключении между экземплярами процессоров с разными архитектурными особенностями, Функция операционной системы в этом отношении становится более сложной. В связи с этим следует отметить, что ядра Alpha EV4-EV8 рассматриваются в этой статье как не полностью совместимые с ISA, как обсуждается, например, в пятом параграфе раздела 2.2.

Кроме того, в этой статье не решается проблема существования значительных затрат, связанных с переключением приложений между ядрами, которые могут значительно уменьшить преимущества, которые получают от указанного переключения.

Сущность изобретения

С точки зрения первого аспекта, настоящее изобретение обеспечивает устройство обработки данных, содержащее: первую компоновку схем обработки для выполнения операций обработки данных, вторую компоновку схем обработки для выполнения операций обработки данных, причем первая компоновка схем обработки является архитектурно совместимой со второй компоновкой схем обработки, так что рабочая нагрузка, выполняемая устройством обработки данных, может выполняться или на первой компоновке схем обработки, или на второй компоновке схем обработки, причем упомянутая рабочая нагрузка содержит, по меньшей мере, одно приложение и, по меньшей мере, одну операционную систему для выполнения упомянутого, по меньшей мере, одного приложения, причем первая компоновка схем обработки отличается с точки зрения микроархитектуры от второй компоновки схем обработки, так что

производительность первой компоновки схем обработки отличается от производительности второй компоновки схем обработки, причем первая и вторая компоновка схем обработки сконфигурированы так, что рабочая нагрузка выполняется одной из первой компоновки схем обработки и второй компоновки схем обработки в любой момент времени, контроллер переключения, реагирующий на управляющее воздействие для переноса, для выполнения операции передачи обслуживания для переноса выполнения рабочей нагрузки из исходной компоновки схем обработки на целевую компоновку схем обработки, причем исходной компоновкой схем обработки является одна из первой компоновки схем обработки и второй компоновки схем обработки, а целевой компоновкой схем обработки является другая из первой компоновки схем обработки и второй компоновки схем обработки, причем контроллер переключения выполнен с возможностью, во время операции передачи обслуживания: (i) вызова предоставления исходной компоновкой схем обработки своего текущего состояния архитектуры целевой компоновке схем обработки, причем текущим состоянием архитектуры является то состояние, которое не доступно из совместно используемой памяти, разделяемой между первой и второй компоновкой схем обработки, в момент инициирования операции передачи обслуживания, и которое является необходимым целевой компоновке схем обработки, чтобы успешно принять на себя выполнение рабочей нагрузки из исходной компоновки схем обработки, и (ii) маскирования предопределенной конкретной для процессора информации о конфигурации от упомянутой, по меньшей мере, одной операционной системы, так что перенос рабочей нагрузки является прозрачным для упомянутой, по меньшей мере, одной операционной системы.

Согласно настоящему изобретению, устройство обработки данных обеспечено с первой и второй компоновкой схем обработки, которые являются архитектурно совместимыми друг с другом, но разными с точки зрения микроархитектуры. Вследствие архитектурной совместимости первой и второй компоновки схем обработки, рабочая нагрузка, состоящая не только из одного или нескольких приложений, но также включающая в себя, по меньшей мере, одну операционную систему для выполнения этих одного или нескольких приложений, может перемещаться между первой и второй компоновкой схем обработки. Кроме того, так как первая и вторая компоновки схем обработки являются разными с точки зрения микроархитектуры, то характеристики производительности (а следовательно, характеристики потребления энергии) первой и второй компоновки схем обработки отличаются.

Согласно настоящему изобретению, в любой момент времени рабочая нагрузка выполняется одной из первой или второй схем обработки, и контроллер переключения реагирует на управляющее воздействие для переноса для выполнения операции передачи обслуживания для переноса выполнения рабочей нагрузки между упомянутыми схемами обработки. При приеме управляющего воздействия для переноса, та из двух схем обработки, которая в настоящее время выполняет рабочую нагрузку, рассматривается как исходная компоновка схем обработки, а другая рассматривается как целевая компоновка схем обработки. Контроллер переключения, отвечающий за выполнение операции передачи обслуживания, вызывает предоставление текущего состояния архитектуры исходной компоновки схем обработки целевой компоновке схем обработки, а также маскирует предопределенную конкретную для процессора информацию о конфигурации от, по меньшей мере, одной операционной системы, формирующей часть рабочей нагрузки, так что перенос рабочей нагрузки является прозрачным для этой операционной системы.

С использованием настоящего изобретения можно переносить всю рабочую нагрузку с одной компоновки схем обработки на другую при маскировании этого переноса от операционной системы и при обеспечении того, что необходимое состояние архитектуры, которое не доступно в совместно используемой памяти в момент инициирования операции передачи обслуживания, предоставляется целевой компоновке схем обработки, так что она успешно может принять на себя выполнение рабочей нагрузки.

При рассмотрении всей рабочей нагрузки как макроскопического объекта, который выполняется только на одной из первой и второй схем обработки в любой конкретный момент времени, способ настоящего изобретения обеспечивает возможность быстрого переключения рабочей нагрузки между первой и второй схемами обработки прозрачно для операционной системы, и в то же время обеспечивает то, что целевая схема обработки имеет всю информацию, необходимую для обеспечения ей возможности принимать на себя выполнение рабочей нагрузки. Такой подход решает ранее упомянутые проблемы, которые возникают в результате использования операционной системы для управления планированием приложений для конкретных схем обработки, и, как обнаружено, обеспечивает возможность получения значительной экономии потребления энергии.

В одном варианте осуществления, устройство обработки данных также содержит: компоновку схем управления питанием для независимого управления питанием, обеспечиваемого первой компоновке схем обработки и второй компоновке схем обработки, причем до осуществления управляющего воздействия для переноса целевая компоновка схем обработки находится в режиме экономии энергии, а во время операции передачи обслуживания компоновка схем управления питанием вызывает выход целевой компоновки схем обработки из режима экономии энергии до того, как целевая компоновка для обработки примет на себя выполнение рабочей нагрузки. С использованием указанной компоновки схем управления питанием можно сократить энергию, потребляемую компоновкой схем обработки, которая в настоящее время не выполняет рабочую нагрузку.

В одном варианте осуществления, после операции передачи обслуживания, компоновка схем управления питанием вызывает переход компоновки схем обработки в режим экономии энергии. Это может происходить немедленно после операции передачи обслуживания, или в альтернативных вариантах осуществления исходная компоновка схем обработки может быть выполнена с возможностью перехода в режим экономии энергии только после истечения некоторого предопределенного периода времени, что может обеспечивать возможность предоставления данных, все еще хранящихся в памяти исходной компоновки схем обработки, целевой компоновке схем обработки с большей эффективностью использования энергии и более высокой производительностью.

Следующей проблемой, которая существует в известном уровне техники, независимо от способа, которым выполняется переключение между разными схемами обработки, является то, как быстро и при эффективном использовании энергии перенести информацию, требуемую для того, чтобы этот перенос был успешным. В частности, необходимо предоставлять вышеупомянутое текущее состояние архитектуры целевой компоновке схем обработки. Одним способом для достижения этого является переписывание всего этого текущего состояния архитектуры в совместно используемую память как часть операции передачи обслуживания, так что целевая компоновка схем обработки может его впоследствии считать из совместно используемой памяти. Как используется в этом описании, термин "совместно используемая память" относится к памяти, к которой может осуществить прямой доступ как первая компоновка схем

обработки, так и вторая компоновка схем обработки, например основной памяти, соединенной как с первой, так и со второй компоновкой схем обработки через межсоединение.

Однако проблема, которая возникает при переписывании всего текущего состояния архитектуры в совместно используемую память, заключается в том, что указанный процесс не только занимает значительное количество времени, но также потребляет значительную энергию, что может существенно нейтрализовать потенциально возможные преимущества, которые могут быть получены с выполнением переключения.

Согласно одному варианту осуществления, во время операции переноса, контроллер переключения вызывает применение исходной компоновкой схем обработки ускоренного механизма для предоставления своего текущего состояния архитектуры целевой компоновке схем обработки без обращения целевой компоновки схем обработки к совместно используемой памяти для получения текущего состояния архитектуры. Следовательно, согласно указанным вариантам осуществления, обеспечен механизм, посредством которого избегают потребности в передаче состояния архитектуры через совместно используемую память для того, чтобы предоставить его целевой компоновке схем обработки. В результате не только повышается производительность во время операции переноса, но также сокращается потребление энергии, связанное с операцией переноса.

В одном варианте осуществления, по меньшей мере, упомянутая исходная компоновка схем имеет ассоциированный кэш, устройство обработки данных также содержит компоновку схем управления просмотром, и ускоренный механизм содержит перенос текущего состояния архитектуры в целевую компоновку схем обработки с использованием упомянутого ассоциированного кэша и упомянутой компоновки схем управления просмотром.

Согласно этому способу, локальный кэш исходной компоновки схем обработки используется для сохранения текущего состояния архитектуры, которое должно предоставляться целевому процессору. Это состояние далее помечается как совместно используемое, что обеспечивает возможность просмотра этого состояния целевой компоновкой схем обработки с использованием компоновки схем управления просмотром. Следовательно, в указанном варианте осуществления, первая и вторая компоновка схем обработки выполняются с когерентными друг другу аппаратными кэшами, это сокращает количество времени, энергию и сложность аппаратного обеспечения, связанные с переключением с исходной компоновки схем обработки на целевую компоновку схем обработки.

В одном конкретном варианте осуществления, ускоренный механизм является механизмом сохранения и восстановления, который вызывает сохранение исходной компоновкой схем обработки своего текущего состояния архитектуры в ассоциированном с ней кэше, и вызывает выполнение целевой компоновкой схем обработки операции восстановления, посредством которой компоновка схем управления просмотром извлекает текущее состояние архитектуры из ассоциированного кэша исходной компоновки схем обработки и обеспечивает это извлеченное текущее состояние архитектуры в целевую компоновку схем обработки. Механизм сохранения/восстановления обеспечивает особенно эффективный способ сохранения состояния архитектуры в локальном кэше исходной компоновки схем обработки и впоследствии извлечения этого состояния для целевой компоновки схем обработки.

Такой подход может быть использован вне зависимости от того, имеет ли целевая компоновка схем обработки свой собственный ассоциированный с ней локальный кэш

или нет. Каждый раз, когда компоновка схем управления просмотром принимает запрос на элемент состояния архитектуры, либо непосредственно из целевой компоновки схем обработки или из ассоциированного локального кэша целевой компоновки схем обработки в случае промаха кэша, она устанавливает то, что этот требуемый элемент состояния архитектуры хранится в локальном кэше, ассоциированном с исходной компоновкой схем, и извлекает эти данные из локального кэша исходной компоновки схем для возвращения в целевую компоновку схем обработки (либо непосредственно или через ассоциированный кэш целевой компоновки схем обработки, если он существует).

В одном конкретном варианте осуществления, целевая компоновка схем обработки имеет ассоциированный кэш, в котором переносимое состояние архитектуры, полученное компоновкой схем управления просмотром, сохраняется для обращения (к нему) целевой компоновки схем обработки.

Однако подход когерентности аппаратного кэша, описанный выше, не является единственным способом, который можно использовать для обеспечения ранее упомянутого ускоренного механизма. Например, в альтернативном варианте осуществления, ускоренный механизм содержит выделенную шину между исходной компоновкой схем обработки и целевой компоновкой схем обработки, по которой исходная компоновка схем обработки обеспечивает свое текущее состояние архитектуры в целевую компоновку схем обработки. Несмотря на то что при таком подходе обычно существуют большие затраты на аппаратное обеспечение, чем при подходе когерентности кэша, он обеспечивает еще более быстрый способ выполнения переноса, от которого можно получить преимущества в определенных реализациях.

Контроллер переключения может принимать множество форм. Однако в одном варианте осуществления контроллер переключения содержит, по меньшей мере, программное обеспечение виртуализации, логически отделяющее, по меньшей мере, одну операционную систему от первой компоновки схем обработки и второй компоновки схем обработки. Известно использование виртуальных машин для обеспечения возможности исполнения приложений, написанных с использованием конкретного набора собственных команд, на аппаратном обеспечении, имеющем отличный набор собственных команд. Приложения исполняются в среде виртуальной машины, причем команды приложений являются собственными для виртуальной машины, но виртуальная машина реализуется программным обеспечением, исполняющимся на аппаратном обеспечении, имеющем отличный набор собственных команд. Программное обеспечение виртуализации, обеспечиваемое контроллером переключения вышеупомянутого варианта осуществления, может рассматриваться как функционирующее аналогично гипервизору в среде виртуальной машины, так как оно обеспечивает разделение между рабочей нагрузкой и лежащей в основе аппаратной платформой. В контексте настоящего изобретения, программное обеспечение виртуализации обеспечивает эффективный механизм для переноса рабочей нагрузки с одной компоновки схем обработки на другую компоновку схем обработки при маскировании конкретной для процессора информации о конфигурации от операционных (ой) систем(ы), формирующих(ей) эту рабочую нагрузку.

Управляющее воздействие для переноса может генерироваться по множеству причин. Однако, в одном варианте осуществления, тайминг управляющего воздействия для переноса выбирается так, чтобы улучшить эффективность использования энергии устройством обработки данных. Это может достигаться множеством способов. Например, могут быть установлены счетчики производительности для подсчета событий,

чувствительных к производительности (например, количество исполнимых команд или количество операций загрузки/сохранения). По сравнению со счетчиком циклов или системным таймером, это обеспечивает возможность идентификации того, что исполняется приложение, требующее очень большой вычислительной мощности, которое может лучше обслуживаться при переключении на компоновку схем обработки с большей производительностью, идентификации большого количества операций загрузки/сохранения, указывающие на приложение, требующее производительности IO (ввода/вывода), которые могут лучше обслуживаться на компоновке схем обработки с эффективным использованием энергии, и т.д. Альтернативным подходом является задание профиля приложений и пометка их как "большой", "маленький" или "большой/маленький", в результате чего операционная система может служить интерфейсом с контроллером переключения для перемещения рабочей нагрузки соответственно (здесь термин "большой" относится к компоновке схем обработки с большей производительностью, а термин "маленький" относится к компоновке схем обработки с эффективным использованием энергии).

Состояние архитектуры, которое требуется для целевой компоновки схем обработки, чтобы успешно принять на себя выполнение рабочей нагрузки от исходной компоновки схем обработки, может принимать множество форм. Однако, в одном варианте осуществления, состояние архитектуры содержит, по меньшей мере, текущее значение одного или нескольких специальных регистров исходной компоновки схем обработки, в том числе значение счетчика команд. Наряду со значением счетчика команд, различную другую информацию, которая может храниться в специальных регистрах. Например, другие специальные регистры включают в себя регистры состояния процессора (например, CPSR и SPSR в архитектуре ARM), которые содержат в себе управляющие биты для режима процессора, маскирования прерываний, рабочего состояния и флагов. Другие специальные регистры включают в себя архитектурный надзор (регистр управления системой CP15 в архитектуре ARM), который содержит биты для изменения порядка следования байтов данных, включения или выключения MMU, включения или выключения кэшей данных/команд и т.д. В других специальных регистрах в CP15 хранится информация о состоянии и адресе исключения.

В одном варианте осуществления, состояние архитектуры также содержит текущие значения, хранящиеся в архитектурном регистровом файле исходной компоновки схем обработки. Специалистам в данной области техники очевидно, что архитектурный регистровый файл содержит регистры, к которым обращаются команды, исполняемые во время выполнения приложения, причем эти регистры содержат исходные операнды для вычислений и обеспечивают адреса ячеек памяти, в которых сохраняются результаты этих вычислений.

В одном варианте осуществления, по меньшей мере, одна из первой компоновки схем обработки и второй компоновки схем обработки содержит один блок обработки. Кроме того, в одном варианте осуществления, по меньшей мере, одна из первой компоновки схем обработки и второй компоновки схем обработки содержит группу блоков обработки с идентичной микроархитектурой. В одном конкретном варианте осуществления, первая компоновка схем обработки может содержать группу блоков обработки с идентичной микроархитектурой, в то время как вторая компоновка схем обработки содержит один блок обработки (с отличной микроархитектурой по сравнению с микроархитектурой блоков обработки в пределах группы, формирующей первую компоновку схем обработки).

Режим экономии энергии, в который компоновка схем управления питанием может

выборочно переводить первую и вторую схемы обработки, может принимать множество форм. В одном варианте осуществления, режим экономии энергии является одним из: режима с отключенным питанием, режима частичного/полного сохранения данных или режима ожидания. Для специалиста в данной области техники такие режимы являются очевидными, и, соответственно, в данном описании более подробно не обсуждаются.

Существует несколько способов выполнения первой и второй схем обработки с разной микроархитектурой. В одном варианте осуществления, первая компоновка схем обработки и вторая компоновка схем обработки отличаются с точки зрения микроархитектуры наличием, по меньшей мере, одного из: разных длин исполнительного конвейера и разных исполнительных ресурсов. Различия в длине конвейера обычно в результате приводят к различиям в рабочей частоте, которые, в свою очередь, оказывают влияние на производительность. Аналогично, различия в исполнительных ресурсах оказывают влияние на пропускную способность и, следовательно, на производительность. Например, схема обработки, имеющая более значительные исполнительные ресурсы, обеспечивает возможность обработки большей информации в любой конкретный момент времени, при этом повышает пропускную способность. Дополнительно, или в качестве альтернативы, одна схема обработки может иметь больше исполнительных ресурсов, чем другая, например, большее количество арифметико-логических устройств (АЛУ, ALU), которые также повышают пропускную способность. В качестве другого примера разных исполнительных ресурсов, схема обработки с эффективным использованием энергии может быть обеспечена простым конвейером с последовательностью исполнения команд по порядку, в то время как схема обработки с более высокой производительностью может быть обеспечена суперскалярным конвейером с переупорядочением последовательности команд.

Следующей проблемой, которая может возникнуть при использовании высокопроизводительных схем обработки, например, функционирующих с частотами, измеряемыми в гигагерцах (ГГц), является то, что такие процессоры приближаются к пределам по нагреву, для функционирования в которых они предназначены, а иногда превышают их. Известные способы поиска решения этих проблем могут включать в себя перевод схемы обработки в режим с малым энергопотреблением для сокращения тепловыделения, что может включать в себя пропуск отдельных тактовых импульсов и/или снижение напряжения, а возможно даже полное отключение схемы обработки на некоторый период времени. Однако при внедрении способа вариантов осуществления настоящего изобретения возможна реализация альтернативного подхода для устранения превышения пределов по нагреву. В частности, в одном варианте осуществления, исходная компоновка схем обработки имеет более высокую производительность, чем целевая компоновка схем обработки, и устройство обработки данных также содержит компоновку схем текущего контроля нагрева для текущего контроля теплоотдачи исходной компоновки схем обработки и для запуска упомянутого управляющего воздействия для переноса, когда упомянутая теплоотдача достигнет предопределенного уровня. В соответствии с указанными способами, вся рабочая нагрузка может быть перемещена с компоновки схем обработки с более высокой производительностью на компоновку схем обработки с меньшей производительностью, после чего будет выделяться меньше тепла, что обеспечит возможность охлаждения исходной компоновки схем обработки. Следовательно, комплект, содержащий две схемы обработки, может охлаждаться наряду с продолжением исполнения программ, хотя и с меньшей пропускной способностью.

Устройство обработки данных может быть выполнено множеством способов. Однако в одном варианте осуществления первая компоновка схем обработки и вторая компоновка схем обработки находятся на одной интегральной схеме.

С точки зрения второго аспекта, настоящее изобретение обеспечивает устройство обработки данных, содержащее: первое средство обработки для выполнения операций обработки данных, второе средство обработки для выполнения операций обработки данных, причем первое средство обработки является архитектурно совместимым со вторым средством обработки, так что рабочая нагрузка, выполняемая устройством обработки данных, может выполняться на любом из первого средства обработки и второго средства обработки, причем упомянутая рабочая нагрузка содержит, по меньшей мере, одно приложение и, по меньшей мере, одну операционную систему для выполнения упомянутого одного приложения, причем первое средство обработки отличается с точки зрения микроархитектуры от второго средства обработки, так что производительность первого средства обработки отличается от производительности второго средства обработки, причем первое и второе средство обработки сконфигурированы так, что рабочая нагрузка выполняется одним из первого средства обработки и второго средства обработки в любой момент времени, средство управления переносом, реагирующее на управляющее воздействие для переноса, для выполнения операции передачи обслуживания для переноса выполнения рабочей нагрузки с исходного средства обработки на целевое средство обработки, причем исходным средством обработки является одно из первого средства обработки и второго средства обработки, а целевым средством обработки является другое из первого средства обработки и второго средства обработки, причем это средство управления переносом (выполнено с возможностью), во время операции передачи обслуживания: (i) вызова предоставления исходным средством обработки своего текущего состояния архитектуры целевому средству обработки, причем текущим состоянием архитектуры является то состояние, которое не доступно из совместно используемого средства памяти, разделяемого между первым и вторым средством обработки, в момент инициирования операции передачи обслуживания, и которое является необходимым целевому средству обработки, чтобы успешно принять на себя выполнение рабочей нагрузки из исходного средства обработки, и (ii) маскирования предопределенной конкретной для процессора информации о конфигурации от упомянутой, по меньшей мере, одной операционной системы, так что передача рабочей нагрузки является прозрачной для упомянутой, по меньшей мере, одной операционной системы.

С точки зрения третьего аспекта, настоящее изобретение обеспечивает способ функционирования устройства обработки данных, содержащего первую компоновку схем обработки для выполнения операций обработки данных и вторую компоновку схем обработки для выполнения операций обработки данных, причем первая компоновка схем обработки является архитектурно совместимой со второй компоновкой схем обработки, так что рабочая нагрузка, выполняемая устройством обработки данных, может выполняться или на первой компоновке схем обработки, или на второй компоновке схем обработки, причем упомянутая рабочая нагрузка содержит, по меньшей мере, одно приложение и, по меньшей мере, одну операционную систему для выполнения упомянутого, по меньшей мере, одного приложения, и причем первая компоновка схем обработки отличается с точки зрения микроархитектуры от второй компоновки схем обработки, так что производительность первой компоновки схем обработки отличается от производительности второй компоновки схем обработки, причем этот способ содержит шаги: выполнения, в любой момент времени, рабочей

нагрузки на одной из первой компоновки схем обработки и второй компоновки схем обработки, выполнения, в ответ на управляющее воздействие для переноса, операции передачи обслуживания для переноса выполнения рабочей нагрузки с исходной компоновки схем обработки на целевую компоновку схем обработки, причем исходной компоновкой схем обработки является одна из первой компоновки схем обработки и второй компоновки схем обработки, а целевой компоновкой схем обработки является другая из первой компоновки схем обработки и второй компоновки схем обработки, во время операции передачи обслуживания: (i) вызова предоставления исходной компоновкой схем обработки своего текущего состояния архитектуры целевой компоновке схем обработки, причем текущим состоянием архитектуры является то состояние, которое не доступно из совместно используемой памяти, разделяемой между первой и второй компоновкой схем обработки, в момент инициирования операции передачи обслуживания, и которое является необходимым целевой компоновке схем обработки, чтобы успешно принять на себя выполнение рабочей нагрузки из исходной компоновки схем обработки, и (ii) маскирования предопределенной конкретной для процессора информации о конфигурации от упомянутой, по меньшей мере, одной операционной системы, так что передача рабочей нагрузки является прозрачной для упомянутой, по меньшей мере, одной операционной системы.

Краткое описание чертежей

Далее настоящее изобретение описывается, только для примера, со ссылкой на его варианты осуществления, иллюстрированные на прилагаемых чертежах, в которых:

Фиг.1 - блок-схема системы обработки данных в соответствии с одним вариантом осуществления.

На фиг.2 схематически изображено обеспечение контроллера переключения (в этом описании также называемого контроллером переноса рабочей нагрузки) в соответствии с одним вариантом осуществления для логического отделения рабочей нагрузки, выполняемой устройством обработки данных, от конкретной аппаратной платформы внутри устройства обработки данных, используемого для выполнения этой рабочей нагрузки.

Фиг.3 - схема, на которой схематически изображены шаги, выполняемые как исходным процессором, так и целевым процессором в ответ на управляющее воздействие для переключения для переноса рабочей нагрузки с исходного процессора на целевой процессор, в соответствии с одним вариантом осуществления.

На фиг.4А схематически изображено сохранение текущего состояния архитектуры исходной компоновки схем обработки в ассоциированном с ней кэше в течение операции сохранения по фиг.3.

На фиг.4В схематически изображено использование блока управления просмотром для управления переносом текущего состояния архитектуры исходной схемы обработки в целевую схему обработки во время операции восстановления по фиг.3.

На фиг.5 изображена альтернативная структура для обеспечения ускоренного механизма для переноса текущего состояния архитектуры исходной компоновки схем обработки на целевую компоновку схем обработки во время операции переноса, в соответствии с одним вариантом осуществления.

На фиг.6А-6I схематически изображены шаги выполнения переноса рабочей нагрузки с исходной схемы обработки на целевую схему обработки в соответствии с одним вариантом осуществления.

Фиг.7 - график, на котором представлено изменение эффективности использования энергии в зависимости от производительности, и на котором иллюстрируется то, как

различные ядра процессора, изображенные на фиг.1, используются в различных точках вдоль этой кривой, в соответствии с одним вариантом осуществления.

На фиг.8A-8B схематически изображены малопроизводительный процессорный конвейер и высокопроизводительный процессорный конвейер, соответственно, используемые в одном варианте осуществления.

Фиг.9 - график, на котором представлено изменение в энергии, потребляемой системой обработки данных, когда выполнение рабочей нагрузки по обработке переключается между малопроизводительной схемой обработки с высокой эффективностью использования энергии и высокопроизводительной схемой обработки с малой эффективностью использования энергии.

Описание вариантов осуществления

Фиг.1 является блок-схемой, на которой схематически иллюстрируется система обработки данных в соответствии с одним вариантом осуществления. Как представлено на фиг.1, система содержит два архитектурно совместимых экземпляра схем обработки (компоновка 0 10 схем обработки и компоновка 1 50 схем обработки), но при этом эти разные экземпляры схем обработки имеют разную микроархитектуру. В частности, компоновка 10 схем обработки выполнена с возможностью функционирования с более высокой производительностью, чем компоновка 50 схем обработки, но с компромиссным решением в отношении того, что компоновка 10 схем обработки менее эффективно использует энергию, чем компоновка 50 схем обработки. Примеры микроархитектурных различий более подробно описаны ниже со ссылкой на фиг.8A-8B.

Каждая схема обработки может включать в себя один блок обработки (в этом описании также называемый ядром процессора), или, в качестве альтернативы, по меньшей мере, один из экземпляров схем обработки может сам содержать группу блоков обработки с идентичной микроархитектурой.

В примере, изображенном на фиг.1, схема 10 обработки включает в себя два ядра 15, 20 процессора, которые являются идентичными как по архитектуре, так и по микроархитектуре. Напротив, схема 50 обработки содержит только одно ядро 55 процессора. В нижеследующем описании, ядра 15, 20 процессора называются "большими" ядрами, в то время как ядро 55 процессора называется "маленьким" ядром, так как ядра 15, 20 процессора обычно являются более сложными, чем ядро 55 процессора, вследствие того, что эти ядра разрабатываются с учетом (высокой) производительности, тогда как ядро 55 процессора, напротив, обычно является менее сложным вследствие разработки с учетом эффективности использования энергии.

На фиг.1, предполагается, что каждое из ядер 15, 20, 55 имеет свой собственный ассоциированный с ним кэш 25, 30, 60 уровня 1, соответственно, который может быть выполнен как объединенный кэш для хранения как команд, так и данных, для обращения (к нему) ассоциированного ядра или может быть выполнен с гарвардской архитектурой, обеспечивающей отдельные кэши уровня 1 команд и данных. Несмотря на то что представлено, что каждое из ядер имеет свой собственный ассоциированный с ним кэш уровня 1, это не является необходимым условием, и, в альтернативных вариантах осуществления, одно или несколько ядер могут не иметь локального кэша.

В варианте осуществления, представленном на фиг.1, компоновка 10 схем обработки также включает в себя кэш 35 уровня 2, совместно используемый между ядром 15 и ядром 20, причем используется блок 40 управления просмотром для обеспечения когерентности кэша между двумя кэшами 25, 30 уровня 1 и кэшем 35 уровня 2. В одном варианте осуществления, кэш уровня 2 выполнен как инклюзивный кэш, и,

следовательно, любые данные, хранящиеся в любом из кэшей 25, 30 уровня 1, также находятся в кэше 35 уровня 2. Как очевидно специалистам в данной области техники, целью блока 40 управления просмотром является обеспечение когерентности кэша между различными кэшами, так что любому ядру 15, 20 всегда может быть обеспечен доступ к самой последней версии любых данных, когда оно выдает запрос на доступ. Следовательно, только в качестве примера, если ядро 15 выдает запрос на доступ к данным, которых нет в ассоциированном кэше 25 уровня 1, то блок 40 управления просмотром перехватывает этот запрос как транслируемый далее из кэша 25 уровня 1 и устанавливает с обращением к кэшу 30 уровня 1 и/или кэшу 35 уровня 2, может ли этот запрос на доступ быть обслужен исходя из частей содержимого одного из этих других кэшей. Только если данные отсутствуют во всех упомянутых кэшах, запрос на доступ транслируется далее через межсоединение 70 в основную память 80, причем основная память 80 является памятью, которая совместно используется компоновкой 10 схем обработки и компоновкой 50 схем обработки.

Блок 75 управления просмотром, обеспеченный в межсоединении 70, функционирует аналогично блоку 40 управления просмотром, но в этом случае целью является поддержка когерентности между структурой кэша, обеспеченной в компоновке 10 схем обработки, и структурой кэша, обеспеченной в компоновке 50 схем обработки. В примерах, в которых кэш 35 уровня 2 является инклюзивным кэшем, блок управления просмотром поддерживает когерентность аппаратного кэша между кэшем 35 уровня 2 компоновки 10 схем обработки и кэшем 60 уровня 1 компоновки 50 схем обработки. Однако, если кэш 35 уровня 2 выполнен как эксклюзивный кэш уровня 2, то блок 75 управления просмотром также просматривает данные, содержащиеся в кэшах 25, 30 уровня 1, для обеспечения когерентности кэша между кэшами компоновки 10 схем обработки и кэшем 60 компоновки 50 схем обработки.

Согласно одному варианту осуществления, только одна из компоновки 10 схем обработки и компоновки 50 схем обработки активно обрабатывает рабочую нагрузку в любой момент времени. Для целей настоящей заявки можно предполагать, что рабочая нагрузка содержит, по меньшей мере, одно приложение и, по меньшей мере, одну операционную систему для выполнения этого, по меньшей мере, одного приложения, как схематически изображено по ссылочной позиции 100 на фиг.2. В этом примере приложения 105, 110 выполняются под управлением операционной системы 115, и в совокупности приложения 105, 110 и операционная система 115 формируют рабочую нагрузку 100. Можно предполагать, что приложения расположены на уровне пользователя, в то время как операционная система расположена на привилегированном уровне, и в совокупности рабочая нагрузка, формируемая приложениями и операционной системой, выполняется на аппаратной платформе 125 (представляющей вид аппаратного уровня). В любой момент времени эта аппаратная платформа обеспечивается либо компоновкой 10 схем обработки или компоновкой 50 схем обработки.

Как изображено на фиг.1, компоновка 65 схем управления питанием обеспечивается для выборочного и независимого обеспечения питанием компоновки 10 схем обработки и компоновки 50 схем обработки. До переноса рабочей нагрузки с одной схемы обработки на другую, обычно только одна из упомянутых схем обработки в полной мере снабжается энергией, т.е. схема обработки, которая в настоящее время выполняет рабочую нагрузку (исходная компоновка схем обработки), а другая схема обработки (целевая компоновка схем обработки) обычно находится в режиме экономии энергии. Когда установлено, что рабочая нагрузка должна быть перенесена с одной схемы обработки на другую, во время операции переноса существует период времени, когда

обе схемы обработки находятся в состоянии с включенным питанием "включено", но в некоторый момент времени после операции переноса, исходная схема обработки, с которой перенесена рабочая нагрузка, переводится в режим экономии энергии.

Режим экономии энергии может принимать различные формы, в зависимости от реализации, и, следовательно, например, может являться одним из режима с отключенным питанием, режима частичного/полного сохранения данных, спящего режима или режима ожидания. Для специалиста в данной области техники такие режимы являются очевидными и, соответственно, в данном описании более подробно не обсуждаются.

Целью описанных вариантов осуществления является выполнение переключения рабочей нагрузки между схемами обработки в зависимости от требуемого уровня энергии/производительности рабочей нагрузки. Соответственно, когда рабочая нагрузка включает в себя исполнение одной или нескольких задач, требующих большой производительности, например, исполнение приложений-игр, то эта рабочая нагрузка может исполняться на высокопроизводительной схеме 10 обработки с использованием одного или обоих больших ядер 15, 20. Однако, напротив, когда рабочая нагрузка является выполнением только задач, требующих малой производительности, например, воспроизведение MP3-файлов, то вся рабочая нагрузка может быть перенесена на схему 50 обработки для получения преимущества от эффективности использования энергии, которое может быть реализовано посредством использования схемы 50 обработки.

Для лучшего использования указанных возможностей переключения, необходимо обеспечить механизм, который обеспечивает возможность осуществления переключения простым и эффективным способом, так что при выполнении переноса рабочей нагрузки не потребляется такое количество энергии, которое сводит на нет преимущества, полученные от переключения, а также обеспечить то, что процесс переключения является достаточно быстрым, чтобы сам он сколько-нибудь существенно не ухудшал производительность.

В одном варианте осуществления, указанные преимущества получены, по меньшей мере, частично посредством выполнения компоновки 10 схем обработки так, что она является архитектурно совместимой с компоновкой 50 схем обработки. Это обеспечивает возможность перемещения рабочей нагрузки с одной компоновки схем обработки на другую при обеспечении правильного функционирования. В качестве абсолютного минимума указанная архитектурная совместимость требует совместного использования обеими схемами 10 и 50 обработки идентичной архитектуры набора команд. Однако в одном варианте осуществления указанная архитектурная совместимость также влечет за собой более высокие требования к совместимости для обеспечения того, что эти два экземпляра схем обработки воспринимаются программистом как идентичные. В одном варианте осуществления, это подразумевает использование идентичных регистров архитектуры и одного или нескольких специальных регистров, в которых хранятся данные, используемые операционной системой во время исполнения приложений. С указанным уровнем совместимости архитектуры возможно маскирование от операционной системы 115 переноса рабочей нагрузки между схемами обработки, так что операционной системе абсолютно не известно то, исполняется ли рабочая нагрузка на компоновке 10 схем обработки или на компоновке 50 схем обработки.

В одном варианте осуществления, обслуживанием переноса с одной схемы обработки на другую управляет контроллер переключения 120, изображенный на фиг.2 (также называемый там виртуализатором, а в других местах этого описания - контроллером переноса рабочей нагрузки). Контроллер переключения может быть осуществлен

комбинацией аппаратных, программно-аппаратных и/или программных компонентов, но в одном варианте осуществления включает в себя программное обеспечение, аналогичное по своему характеру программному обеспечению гипервизора, которое обеспечивается в виртуальных машинах, для обеспечения возможности исполнения приложений, написанных в пределах одного набора собственных команд, на аппаратной платформе, на которой внедрен другой набор собственных команд. Вследствие архитектурной совместимости между двумя схемами 10, 50 обработки контроллер переключения 120 может маскировать перенос от операционной системы 115 только посредством маскирования одного или нескольких элементов предопределенной конкретной для процессора информации о конфигурации от операционной системы. Например, конкретная для процессора информация о конфигурации может включать в себя части содержимого регистра ID процессора CP15 и регистра типа кэша CP15.

В указанном варианте осуществления, контроллеру переключения тогда только необходимо обеспечить возможность того, что текущее состояние архитектуры, фиксируемое исходной схемой обработки в момент переноса, и которое не является (состоянием) в момент, когда перенос инициируется уже как доступный из совместно используемой памяти 80, является доступным для целевой схемы обработки, чтобы обеспечить возможность этой целевой схеме успешно принять на себя выполнение рабочей нагрузки. С использованием ранее описанного примера указанное состояние архитектуры обычно содержит текущие значения, сохраненные в архитектурном регистровом файле исходной компоновки схем обработки, вместе с текущими значениями одного или нескольких специальных регистров исходной компоновки схем обработки. Вследствие архитектурной совместимости схем 10, 50 обработки, если это текущее состояние архитектуры может быть перенесено из исходной схемы обработки в целевую схему обработки, то целевая схема обработки может успешно принять на себя выполнение рабочей нагрузки из исходной схемы обработки.

В то время как архитектурная совместимость схем 10, 50 обработки способствует переносу всей рабочей нагрузки между этими двумя схемами обработки, в одном варианте осуществления схемы 10, 50 обработки отличаются друг от друга с точки зрения микроархитектуры, так что существуют различные характеристики производительности и, следовательно, характеристики потребления энергии, ассоциированные с этими двумя схемами обработки. Как обсуждалось ранее, в одном варианте осуществления, схема 10 обработки является высокопроизводительной схемой обработки с высоким потреблением энергии, в то время как схема 50 обработки является схемой обработки с меньшей производительностью и с меньшим потреблением энергии. Эти две схемы обработки могут отличаться друг от друга с точки зрения микроархитектуры во многих отношениях, но обычно имеют, по меньшей мере, одно из разных длин исполнительного конвейера и/или разных исполнительных ресурсов. Различия в длине конвейера обычно в результате приводят к различиям в рабочей частоте, которые, в свою очередь, оказывают влияние на производительность. Аналогично различия в исполнительных ресурсах оказывают влияние на пропускную способность и, следовательно, на производительность. Следовательно, например, компоновка 10 схем обработки может иметь более значительные исполнительные ресурсы и/или больше исполнительных ресурсов, чтобы повысить пропускную способность. Кроме того, конвейеры в ядрах 15, 20 процессора могут быть выполнены с возможностью суперскалярной обработки с переупорядочением последовательности команд, в то время как более простое ядро 55 в схеме 50 обработки с эффективным использованием энергии может быть выполнено в виде конвейера с

последовательностью исполнения команд по порядку. Дальнейшее обсуждение микроархитектурных различий обеспечено ниже со ссылкой на фиг.8A и фиг.8B.

Генерация управляющего воздействия для переноса, чтобы вызвать инициирование контроллером переключения 120 операции передачи обслуживания для переноса рабочей нагрузки с одной схемы обработки на другую, может быть запущена по множеству причин. Например, в одном варианте осуществления, может быть задан профиль приложений, и они могут быть помечены как "большой", "маленький" или "большой/маленький", в результате чего операционная система может служить интерфейсом с контроллером переключения для перемещения рабочей нагрузки соответственно. Следовательно, при таком подходе генерация управляющего воздействия для переноса может отображаться в конкретные комбинации исполняемых приложений для обеспечения того, что когда требуется высокая производительность, рабочая нагрузка исполняется на высокопроизводительной схеме 10 обработки, тогда как, когда такая производительность не требуется, вместо нее используется схема 50 обработки с эффективным использованием энергии. В одном варианте осуществления, могут исполняться алгоритмы для динамического определения того, когда запускать перенос рабочей нагрузки с одной схемы обработки на другую, на основе одного или нескольких входных значений. Например, могут быть установлены счетчики производительности для подсчета событий, чувствительных к производительности (например, количество исполнимых команд или количество операций загрузки/сохранения). По сравнению со счетчиком циклов или системным таймером, это обеспечивает возможность идентификации того, что исполняется приложение, требующее очень большой вычислительной мощности, которое может лучше обслуживаться при переключении на компоновку схем обработки с большей производительностью, или идентификации большого количества операций загрузки/сохранения, указывающие на приложение, требующее производительности IO (ввода/вывода), которые могут лучше обслуживаться на компоновке схем обработки с эффективным использованием энергии, и т.д.

В качестве еще одного примера того, когда может генерироваться управляющее воздействие для переноса, система обработки данных может включать в себя один или несколько термочувствительных датчиков 90 для текущего контроля температуры этой системы обработки данных во время функционирования. Это может быть в случае, когда современные высокопроизводительные схемы обработки, например, функционирующие с частотами, измеряемыми в гигагерцах (ГГц), иногда приближаются к пределам по нагреву, для функционирования в которых они предназначены, или превышают их. С использованием указанных термочувствительных датчиков 90 можно обнаруживать приближение к указанным пределам по нагреву, и в этих условиях может генерироваться управляющее воздействие для переноса для запуска переноса рабочей нагрузки на схему обработки с более эффективным использованием энергии, чтобы вызвать полное охлаждение системы обработки данных. Следовательно, при рассмотрении примера по фиг.1, в котором схема 10 обработки является высокопроизводительной схемой обработки, а схема 50 обработки является схемой обработки с меньшей производительностью, потребляющей меньше энергии, перемещение рабочей нагрузки из схема 10 обработки на схему 50 обработки при приближении к пределам по нагреву устройства вызывает последующее охлаждение этого устройства, наряду с обеспечением возможности продолжения исполнения программ, хотя и с меньшей пропускной способностью.

Несмотря на то что на фиг.1 представлены две схемы 10, 50 обработки, очевидно, что способы вышеописанных вариантов осуществления также могут быть применены

к системам, содержащим более двух разных схем обработки, с обеспечением возможности охвата системой обработки данных большего диапазона уровней энергии/производительности. В указанных вариантах осуществления, все упомянутые разные схемы обработки выполнены архитектурно совместимыми друг с другом для обеспечения возможности быстрого перемещения всей рабочей нагрузки между этими схемами обработки, но они отличается друг от друга с точки зрения микроархитектуры для обеспечения возможности выбора использования этих схем обработки в зависимости от требуемых уровней энергии/производительности.

Фиг.3 является схемой последовательности операций, иллюстрирующей последовательность шагов, выполняемых как на исходном процессоре, так и на целевом процессоре при переносе рабочей нагрузки из исходного процессора на целевой процессор после приема управляющего воздействия. Такое управляющее воздействие для переноса может генерироваться операционной системой 115 или виртуализатором 120 через системный программно-аппаратный интерфейс, что в результате приводит к обнаружению переключательного управляющего воздействия на шаге 200 исходным процессором (который выполняет не только рабочую нагрузку, но также и программное обеспечение виртуализатора, формирующее, по меньшей мере, часть контроллера переключения 120). Прием управляющего воздействия для переноса (в этом описании также называемого переключательным управляющим воздействием) на шаге 200 вызывает инициирование контроллером 65 управления питанием операции 205 включения питания и сброса на целевом процессоре. После указанного включения питания и сброса, на шаге 210 целевой процессор объявляет недействительным свой локальный кэш и далее на шаге 215 обеспечивает возможность просмотра. В этот момент, целевой процессор подает сигнал в исходный процессор о том, что он готов для переноса рабочей нагрузки, при этом этот сигнал вызывает исполнение исходным процессором операции сохранения состояния на шаге 225. Эта операция сохранения состояния более подробно обсуждается ниже со ссылкой на фиг.4А, но в одном варианте осуществления подразумевает, что исходная компоновка схем обработки сохраняет в своем локальном кэше свое любое текущее состояние, которое не является доступным из совместно используемой памяти во время инициирования операции передачи обслуживания, и которое является необходимым для того, чтобы целевой процессор успешно принял на себя выполнение рабочей нагрузки.

После операции 225 сохранения состояния в целевой процессор выдается сигнал переключения состояния 230, указывающий целевому процессору на то, что он должен в данный момент начать просмотр исходного процессора для извлечения требуемого состояния архитектуры. Этот процесс происходит посредством операции 230 восстановления состояния, которая более подробно обсуждается ниже со ссылкой на фиг.4В, но которая в одном варианте осуществления подразумевает, что целевая компоновка схем обработки инициирует последовательность доступов, которые перехватываются блоком 75 управления просмотром в пределах межсоединения 70, и которая вызывает извлечение кэшированной копии состояния архитектуры в локальном кэше исходного процессора и возвращения ее в целевой процессор.

После шага 230, целевой процессор может принять на себя обработку рабочей нагрузки, и, соответственно, на шаге 235 начинается обычное функционирование.

В одном варианте осуществления, после начала обычного функционирования в целевом процессоре, кэш исходного процессора может быть очищен, как указано на шаге 250, со сбросом всех измененных данных в совместно используемую память 80, и после этого питание исходного процессора может быть выключено на шаге 255.

Однако, в одном варианте осуществления, для дополнительного повышения эффективности целевого процессора, исходный процессор выполнен с возможностью оставаться включенным в течение некоторого периода времени, на фиг.3 обозначенного как период просмотра. В течение этого времени, по меньшей мере, один из кэшей исходной схемы остается включенным, так что его части содержимого могут просматриваться схемой 75 управления просмотром в ответ на запрос на доступ, выдаваемый целевым процессором. После переноса всей рабочей нагрузки с использованием процесса, описанного на фиг.3, предполагается, что, по меньшей мере, в начальный период времени, после которого целевой процессор начнет оперативное управление рабочей нагрузкой, некоторые данные, требуемые во время выполнения рабочей нагрузки, остаются в кэше исходного процессора. Если исходный процессор сбросил части своего содержимого в память, и его питание выключено, то целевой процессор на этих ранних этапах будет функционировать относительно неэффективно, так как будет много промахов кэша в его локальном кэше и много выборок данных из совместно используемой памяти, что в результате существенно повлияет на производительность во время "разогрева" кэша целевого процессора, т.е. заполнения значениями данных, требуемых схеме целевого процессора для выполнения операций, задаваемых рабочей нагрузкой. Однако в результате того, что кэш исходного процессора остается включенным в течение периода просмотра, схема 75 управления просмотром может обслуживать множество этих запросов промаха кэша со ссылкой на кэш исходной схемы, что дает существенные преимущества по производительности по сравнению с извлечением этих данных из совместно используемой памяти 80.

Однако, как предполагается, что это преимущество по производительности длится только в течение определенного периода времени после переключения, после которого содержимое кэша исходного процессора становится устаревшим. Соответственно, в некоторый момент времени, на шаге 245, генерируется событие прекращения просмотра для блокировки просмотра, после чего, на шаге 250, очищается кэш исходного процессора, и после этого, на шаге 255, питание исходного процессора выключается. Различные сценарии, по которым может генерироваться событие прекращения просмотра, более подробно обсуждаются ниже со ссылкой на фиг.6G.

На фиг.4A схематически изображена операция сохранения, выполняемая на шаге 225 по фиг.3, в соответствии с одним вариантом осуществления. В частности, в одном варианте осуществления, состояние архитектуры, которое должно быть сохранено из исходной компоновки 300 схем обработки в локальном кэше 330, состоит из частей содержимого регистрового файла 310, к которому обращается арифметико-логическое устройство 305 (ALU, АЛУ) во время выполнения операций обработки данных, а также частей содержимого различных специальных регистров 320, идентифицирующих различные порции информации, требуемые рабочей нагрузкой для обеспечения возможности целевой схеме обработки успешно принять на себя эту рабочую нагрузку. Части содержимого специальных регистров 320 включают в себя, например, значение счетчика команд, идентифицирующего текущую исполняемую команду, а также различную другую информацию. Например, другие специальные регистры включают в себя регистры состояния процессора (например, CPSR и SPSR в архитектуре ARM), которые содержат в себе управляющие биты для режима процессора, маскирования прерываний, рабочего состояния и флагов. Другие специальные регистры включают в себя архитектурный надзор (регистр управления системой CP15 в архитектуре ARM), который содержит биты для изменения порядка следования байтов данных, включения или выключения MMU, включения или выключения кэшей данных/команд и т.д. В

других специальных регистрах в CP15 хранится информация о состоянии и адресе исключения.

Как схематически изображено на фиг.4A, исходная схема 300 обработки также обычно содержит некоторую конкретную для процессора информацию 315 о конфигурации, но эту информацию не требуется сохранять в кэше 330, так как она не относится к целевой схеме обработки. Конкретная для процессора информация 315 о конфигурации обычно является жестко закодированной в исходной схеме 300 обработки с использованием логических констант и может включать в себя, например, регистр ID процессора CP15 (который отличается для каждой схемы обработки) или части содержимого регистра типа кэша CP15 (который зависит от конфигурации кэшей 25, 30, 60, например, с указанием того, что эти кэши имеют разную длину строки). Когда операционная система 115 требует порцию конкретной для процессора информации 315 о конфигурации, то, если только процессор уже не находится в режиме гипервизора, происходит системное прерывание исполнения (и переход) в режим гипервизора. В ответ виртуализатор 120 в одном варианте осуществления может включать в себя значение требуемой информации, а в другом варианте осуществления возвращает "виртуальное" значение. В случае значения ID процессора, это виртуальное значение может выбираться идентичным как для "большого", так и для "маленького" процессоров, тем самым вызывая сокрытие фактической конфигурации аппаратного обеспечения от операционной системы 115 посредством виртуализатора 120.

Как схематически изображено на фиг.4A, во время операции сохранения, части содержимого регистрового файла 310 специальных регистров 320 сохраняются исходной компоновкой схем обработки в кэше 330 для формирования кэшированной копии. Эта кэшированная копия после этого помечается как совместно используемая, что обеспечивает возможность просмотра этого состояния целевым процессором посредством блока 75 управления просмотром.

Операция восстановления, впоследствии выполняемая на целевом процессоре, схематически изображена на фиг.4B. В частности, целевая компоновка 350 схем обработки, в которой может существовать или отсутствовать свой собственный локальный кэш, выдает запрос на конкретный элемент состояния архитектуры, причем этот запрос перехватывается блоком 75 управления просмотром. Блок управления просмотром после этого выдает запрос на просмотр в локальный кэш 330 исходной схемы обработки для определения того, существует ли этот элемент состояния архитектуры в кэше этого источника. Вследствие шагов, выполняемых во время операции сохранения, изображенной на фиг.4A, будет обнаружено "попадание" в кэше 330 источника, что в результате приводит к возвращению этого кэшированного состояния архитектуры посредством блока 75 управления просмотром в целевую схему 350 обработки. Этот процесс может повторяться итеративно до тех пор, пока все элементы состояния архитектуры не будут извлечены посредством просмотра кэша исходной схемы обработки. Как ранее обсуждалось, любая конкретная для процессора информация о конфигурации, относящаяся к целевой схеме 350 обработки, обычно является жестко закодированной в этой целевой схеме 350 обработки. Соответственно, после завершения операции восстановления, в целевой компоновке схем обработки существует вся информация, требуемая для обеспечения ей возможности успешно принять на себя обслуживание рабочей нагрузки.

Кроме того, в одном варианте осуществления, вне зависимости от того, выполняется ли рабочая нагрузка 100 "большой" схемой 10 обработки, "маленькой" схемой 50 обработки, виртуализатор 120 предоставляет операционной системе 115 информацию

о виртуальной конфигурации, содержащую идентичные значения, и поэтому различия в аппаратном обеспечении между "большой" и "маленькой" схемами 10, 50 обработки маскируются от операционной системы 115 виртуализатором 120. Это означает, что операционной системе 115 не известно о переносе выполнения рабочей нагрузки 100 на другую аппаратную платформу.

Согласно операциям сохранения и восстановления, описанным со ссылкой на фиг.4A и фиг.4B, различные экземпляры 10, 50 процессоров выполнены с когерентностью аппаратных кэшей друг другу для сокращения количества времени, энергии и сложности аппаратного обеспечения, связанных с переносом состояния архитектуры с исходного процессора на целевой процессор. В упомянутом способе используется локальный кэш исходного процессора для сохранения всего состояния, которое должно быть перенесено с исходного процессора на целевой процессор, и которое не является доступным из совместно используемой памяти во время выполнения операции переноса. Так как упомянутое состояние помечается как совместно используемое в пределах кэша исходного процессора, это обеспечивает возможность целевому процессору с когерентным аппаратным кэшем просматривать это состояние в течение операции переноса. С использованием такого способа можно переносить упомянутое состояние между экземплярами процессоров без необходимости сохранения этого состояния в основной памяти или в запоминающем элементе, отображенном в локальную память. Это, следовательно, дает существенные преимущества по производительности и потреблению энергии с увеличением множества ситуаций, в которых целесообразно переключать рабочую нагрузку с целью реализации преимуществ по потреблению энергии.

Однако, несмотря на то что вышеописанный способ использования когерентности кэша обеспечивает один ускоренный механизм предоставления текущего состояния архитектуры целевому процессору без передачи текущего состояния архитектуры через совместно используемую память, он не является единственным способом, которым такой ускоренный механизм может быть реализован. Например, на фиг.5 изображен альтернативный механизм, в котором обеспечивается выделенная шина 380 между исходной компоновкой 300 схем обработки и целевой компоновкой 350 схем обработки для обеспечения возможности переноса состояния архитектуры во время операции передачи обслуживания. Следовательно, в таких вариантах осуществления, операции 225, 230 сохранения и восстановления по фиг.3 заменяются альтернативным механизмом переноса с использованием выделенной шины 380. Несмотря на то что при таком подходе обычно существуют большие затраты на аппаратное обеспечение, чем при подходе когерентности кэша (причем подход когерентности кэша обычно использует аппаратное обеспечение, уже установленное в системе обработки данных), он обеспечивает еще более быстрый способ выполнения переноса, от которого можно получить преимущества в определенных реализациях.

На фиг.6A-6I изображена последовательность шагов, которые выполняются для выполнения переноса рабочей нагрузки из исходной компоновки 300 схем обработки в целевую компоновку 350 схем обработки. Исходная компоновка 300 схем обработки является той из схем 10, 50 обработки, которая выполняет рабочую нагрузку до переноса, при этом целевая компоновка схем обработки является другой из схем 10, 50 обработки.

На фиг.6A представлена система в начальном состоянии, в котором исходная компоновка 300 схем обработки снабжается энергией контроллером 65 управления питанием и выполняет рабочую нагрузку 100 по обработке, в то время как целевая

компоновка 350 схем обработки находится в режиме экономии энергии. В этом варианте осуществления, режим экономии энергии является режимом с отключением питания, но, как упомянуто выше, также могут быть использованы другие типы режима экономии энергии. Рабочая нагрузка 100, включающая в себя приложения 105, 110 и операционную систему 115 для выполнения приложений 105, 110, абстрагируется от аппаратной платформы исходной компоновки 300 схем обработки посредством виртуализатора 120. Во время выполнения рабочей нагрузки 100, исходная компоновка 300 схем обработки поддерживает состояние 400 архитектуры, которое может содержать, например, части содержимого регистрового файла 310 и специальных регистров 320, как изображено на фиг.4A.

На фиг.6B виртуализатор 120 обнаруживает управляющее воздействие 430 для переноса. Несмотря на то что на фиг.6B управляющее воздействие 430 для переноса представлено как внешнее событие (например, обнаружение ухода нагрева термочувствительным датчиком 90), управляющее воздействие 430 также может являться событием, запускаемым самим виртуализатором 120 или операционной системой 115 (например, операционная система 115 может быть сконфигурирована для сообщения виртуализатору 120 о том, когда должен обрабатываться конкретный тип приложения). В ответ на управляющее воздействие 430 виртуализатор 120 осуществляет управление контроллером 65 управления питанием для подачи питания в целевую компоновку 350 схем обработки для перевода ее в состояние с включенным питанием.

На фиг.6C целевая компоновка 350 схем обработки начинает исполнение виртуализатора 120. Виртуализатор 120 осуществляет управление целевой компоновкой 350 схем обработки для объявления своего кэша 420 недействительным для предотвращения ошибок при обработке, вызываемых ошибочными значениями данных, которые могут существовать в кэше 420 при включении целевой компоновки 350 схем обработки, в то время как целевой кэш 420 объявляется недействительным, исходная компоновка 350 схем обработки продолжает выполнять рабочую нагрузку 100. Когда объявление целевого кэша 420 недействительным завершается, виртуализатор 120 осуществляет управление целевой компоновкой 350 схем обработки для подачи сигнала в исходную компоновку 300 схем обработки о том, что оно готово для передачи обслуживания рабочей нагрузки 100. С продолжением обработки рабочей нагрузки 100 на исходной компоновке 300 схем обработки до тех пор, пока целевая компоновка 350 схем обработки не будет готова к операции передачи обслуживания, влияние на производительность передачи обслуживания может быть сокращено.

На следующем этапе, представленном на фиг.6D, исходная компоновка 300 схем обработки прекращает выполнение рабочей нагрузки 100. В течение этого этапа, ни исходная компоновка 300 схем обработки, ни целевая компоновка 350 схем обработки не выполняет рабочую нагрузку 100. Из исходной компоновки 300 схем обработки в целевую компоновку 350 схем обработки переносится копия состояния 400 архитектуры. Например, состояние 400 архитектуры может быть сохранено в исходном кэше 410 и восстановлено в целевой компоновке 350 схем обработки, как представлено на фиг.4A и фиг.4B, или может быть перенесено по выделенной шине, как представлено на фиг.5. Состояние 400 архитектуры содержит всю информацию, требуемую для выполнения рабочей нагрузки 100 целевой компоновкой 350 схем обработки, за исключением информации, которая уже существует в совместно используемой памяти 80.

После переноса состояния 400 архитектуры в целевую компоновку 350 схем обработки, исходная компоновка 300 схем обработки переводится в состояние экономии энергии компоновкой 65 схем управления питанием (см. Фиг.6E) за исключением того,

что исходный кэш 410 продолжает снабжаться энергией. При этом целевая компоновка 350 схем обработки начинает выполнение рабочей нагрузки 100 с использованием перенесенного состояния 400 архитектуры.

Когда целевая компоновка 350 схем обработки начинает обработку рабочей нагрузки 100, начинается период просмотра (см. фиг.6F). В течение периода просмотра, блок 75 управления просмотром может просматривать данные, сохраняемые в исходном кэше 410, и извлекать эти данные для целевой компоновки 350 схем обработки. Когда целевая компоновка 350 схем обработки запрашивает данные, которые отсутствуют в целевом кэше 420, эта целевая компоновка 350 схем обработки запрашивает данные у блока 75 управления просмотром. После этого блок 75 управления просмотром просматривает исходный кэш 410, и если результатом этого просмотра является кэш-попадание, то блок 75 управления просмотром извлекает эти данные, полученные в результате просмотра, из исходного кэша 410 и возвращает их в целевую компоновку 350 схем обработки, в которой эти данные, полученные в результате просмотра, могут быть сохранены в целевом кэше 420. Напротив, если результатом этого просмотра является промах кэша в исходном кэше 410, то запрашиваемые данные выбираются из совместно используемой памяти 80 и возвращаются в целевую компоновку 350 схем обработки. Так как доступы к данным в исходном кэше 410 осуществляются быстрее и требуют меньше энергии, чем доступы к совместно используемой памяти 80, то просмотр исходного кэша 410 за некоторый период повышает производительность обработки и сокращает потребление энергии в течение начального периода после передачи обслуживания рабочей нагрузки 100 в целевую компоновку 350 схем обработки.

На шаге, представленном на фиг.6G, блок 75 управления просмотром обнаруживает событие прекращения просмотра, которое указывает на то, что поддержание исходного кэша 410 в состоянии с включенным питанием больше не является целесообразным. Событие прекращения просмотра вызывает окончание периода просмотра. Событие прекращения просмотра может быть любым из набора событий прекращения просмотра, контролируемых компоновкой 75 схем управления просмотром. Например, набор событий прекращения просмотра может включать в себя любое одно или большее количество из следующих событий:

a) когда доля в процентах или доля попаданий при просмотре, которые в результате приводят к кэш-попаданиям в исходном кэше 410 (т.е. количество, пропорциональное "количество попаданий при просмотре"/"общее количество просмотров") падает ниже предопределенного порогового уровня после начала выполнения рабочей нагрузки 100 целевой компоновкой 350 схем обработки,

b) когда количество транзакций или количество транзакций предопределенного типа (например, кэшируемых транзакций), выполненных с момента, когда целевая компоновка 350 схем обработки начала выполнение рабочей нагрузки 100, превысит предопределенный порог,

c) когда количество циклов обработки, прошедших с момента, когда целевая компоновка 350 схем обработки начала выполнение рабочей нагрузки 100, превысит предопределенный порог,

d) когда к конкретной области совместно используемой памяти 80 осуществлен доступ в первый раз с момента, когда целевая компоновка 350 схем обработки начала выполнение рабочей нагрузки 100,

e) когда к конкретной области совместно используемой памяти 80, к которой осуществлялся доступ в течение начального периода после того, когда целевая компоновка 350 схем обработки начала выполнение рабочей нагрузки 100, не

осуществляется доступ в течение предопределенного количества циклов или предопределенного периода времени,

f) когда целевая компоновка 350 схем обработки записывает в предопределенную ячейку памяти в первый раз с момента начала выполнения перенесенной рабочей нагрузки 100.

Эти события прекращения просмотра могут быть определены с использованием программируемых счетчиков в когерентном межсоединении 70, которое включает в себя блок 75 управления просмотром. В набор событий прекращения просмотра также могут быть включены другие типы события прекращения просмотра.

При обнаружении события прекращения просмотра, блок 75 управления просмотром отправляет сигнал 440 прекращения просмотра в исходный процессор 300. Блок 75 управления просмотром прекращает просмотр исходного кэша 410 и с этого момента и далее в ответ на запросы на доступ к данным из целевой компоновки 350 схем обработки осуществляет выборку запрашиваемых данных из совместно используемой памяти 80 и возвращает выбранные данные в целевую компоновку 350 схем обработки, в которой выбранные данные могут быть помещены в кэш.

На фиг.6H схема управления исходного кэша в ответ на сигнал 440 прекращения просмотра очищает кэш 410 для сохранения в совместно используемой памяти 80 всех действительных и измененных значений данных (то есть кэшируемое значение которых является более новым, чем соответствующее значение в совместно используемой памяти 80).

На фиг.6I питание исходного кэша 410 после этого выключается контроллером 65 управления питанием для полного перевода исходной компоновки 300 схем обработки в состояние экономии энергии. Целевая компоновка 350 схем обработки продолжает выполнять рабочую нагрузку 100. С точки зрения операционной системы 115, текущая ситуация является идентичной ситуации по фиг.6A. Операционной системе 115 не известно, что исполнение рабочей нагрузки перенесено из одной схемы обработки на другую схему обработки. Когда осуществляется еще одно управляющее воздействие для переноса, можно использовать идентичные шаги по фиг.6A-6I для переключения выполнения рабочей нагрузки обратно в первый процессор (в этом случае та из схем 10, 50 обработки, которая была "исходной компоновкой схем обработки", станет "целевой компоновкой схем обработки" и наоборот).

В варианте осуществления по фиг.6A-6I, независимое управление питанием для кэша 410 и исходной компоновки 300 схем обработки является доступным, чтобы питание исходной компоновки 300 схем обработки, за исключением исходного кэша 410, могло быть выключено после того, как целевая компоновки 350 схем обработки начнет выполнение рабочей нагрузки (см. фиг.6E), в то время как только кэш 410 исходной компоновки 350 схем обработки остается в состоянии с включенным питанием (см. фиг.6F-6H). Далее на фиг.6I питание исходного кэша 410 выключается. Этот подход может быть полезным для экономии энергии особенно тогда, когда исходная компоновка 300 схем обработки является "большой" схемой 10 обработки.

Однако также можно продолжать снабжать энергией всю компоновку 300 схем обработки в течение периода просмотра, и далее на фиг.6I переводить исходную компоновку 300 схем обработки в целом в состояние экономии энергии, после окончания периода просмотра и очистки исходного кэша 410. Это может быть полезным в случае, когда исходный кэш 410 является глубоко встроенным в ядро исходного процессора и не может снабжаться независимо от ядра исходного процессора. Этот подход также может являться более целесообразным, когда исходный процессор является "маленькой"

схемой 50 обработки, потребление энергии которой является незначительным по сравнению с "большой" схемой 10 обработки, так как после того, как "большая" схема 10 обработки начнет обработку перенесенной рабочей нагрузки 100, то переключение "маленькой" схемы 50 обработки, отличной от кэша 60, в состояние экономии энергии в течение периода просмотра может незначительно влиять на общее потребление энергии системой. Это может означать, что дополнительная сложность аппаратного обеспечения при обеспечении отдельного управления питанием для "маленькой" схемы 50 обработки и кэша 60 "маленького" ядра может быть неоправданным.

В некоторых случаях, о том, что данные, сохраненные в исходном кэше 410, не потребуются целевой компоновке 350 схем обработки, когда она начнет выполнение рабочей нагрузки 100, может быть известно до переноса этой рабочей нагрузки. Например, исходная компоновка 300 схем обработки может только что завершить приложение, когда произойдет перенос, и, следовательно, данные, находящиеся в исходном кэше 410 во время переноса, относятся к завершенному приложению, а не к приложению, которое должно выполняться целевой компоновкой 350 схем обработки после переноса. В этом случае, контроллер отмены просмотра может запустить виртуализатор 120 и компоновку 75 схем управления просмотром для отмены просмотра исходного кэша 410 и для управления исходной схемой 300 обработки для очистки и выключения питания исходного кэша 410 без ожидания события прекращения просмотра для подачи сигнала окончания периода просмотра. В этом случае, способ по фиг.6A-6I выполняет переход от шага по фиг.6E сразу к шагу по фиг.6G без шага по фиг.6F, на котором данные просматриваются (и выбираются) из исходного кэша 410. Соответственно, если заранее известно, что данные в исходном кэше 410 не будут использоваться целевой компоновкой 350 схем обработки, то можно сэкономить энергию посредством перевода исходного кэша 410 и исходной компоновки 300 схем обработки в режим экономии энергии без ожидания события прекращения просмотра. Контроллер отмены просмотра может быть частью виртуализатора 120 или может быть реализован как программно-аппаратное обеспечение, исполняемое в компоновке 300 схем обработки. Контроллер отмены просмотра также может быть реализован как комбинация элементов, например, операционная система 115 может сообщать в виртуализатор 120 о том, когда заканчивается приложение, и виртуализатор 120 тогда может отменять просмотр исходного кэша 410, если перенос происходит тогда, когда приложение закончено.

Фиг.7 является графиком, на котором линия 600 иллюстрирует то, как изменяется потребление энергии в зависимости от производительности. Для различных участков этого графика, система обработки данных может быть выполнена с возможностью использования разных комбинаций ядер 15, 20, 55 процессора, изображенных на фиг.1, с целью получения подходящего оптимального соотношения между производительностью и потреблением энергии. Следовательно, например, когда должно исполняться множество очень высокопроизводительных задач, можно задействовать оба больших ядра 15, 20 схемы 10 обработки для достижения требуемой производительности. Дополнительно можно использовать способы изменения напряжения источника питания для обеспечения возможности некоторого изменения в производительности и потреблении энергии при использовании этих двух ядер.

Когда требования к производительности падают до уровня, на котором требуемая производительность может быть достигнута с использованием только одного из больших ядер, то задачи могут быть перемещены только на одно из больших ядер 15, 20, при этом питание другого ядра выключается, или оно переводится в некоторый

другой режим экономии энергии. И опять же, можно использовать изменение напряжения источника питания для обеспечения возможности некоторого изменения между производительностью и потреблением энергии при использовании этого одного большого ядра. Следует отметить, что переход с двух больших ядер на одно большое ядро не требует генерации управляющего воздействия для переноса или использования вышеописанных способов переноса рабочей нагрузки, так как во всех случаях используется схема 10 обработки, а схема 50 обработки будет находиться в режиме экономии энергии. Однако, как указано пунктирной линией на фиг.7, когда производительность падает до уровня, на котором посредством небольшого ядра может быть достигнута требуемая производительность, то может генерироваться управляющее воздействие для переноса, чтобы запустить ранее описанный механизм для переноса всей рабочей нагрузки из схемы 10 обработки на схему 50 обработки, так что вся рабочая нагрузка после этого выполняется на небольшом ядре 55, при этом схема 10 обработки переводится в режим экономии энергии. И опять же, можно использовать изменение напряжения источника питания для обеспечения возможности некоторого изменения в производительности и потреблении энергии небольшого ядра 55.

На фиг.8А-8В соответственно изображены микроархитектурные различия между малопроизводительным процессорным конвейером 800 и высокопроизводительным процессорным конвейером 850 согласно одному варианту осуществления. Малопроизводительный процессорный конвейер 800 по фиг.8А подходит для маленького ядра 55 обработки данных по фиг.1, тогда как высокопроизводительный процессорный конвейер 850 по фиг.8В подходит для больших ядер 15, 20.

Малопроизводительный процессорный конвейер 800 по фиг.8А содержит этап 810 выборки для выборки команд из памяти 80, этап 820 декодирования для декодирования выбранных команд, этап 830 выдачи для выдачи команд на исполнение и множество конвейеров исполнения, включающих в себя целочисленный конвейер 840 для выполнения целочисленных операций, конвейер 842 MAC для выполнения операций умножения с накоплением и конвейер 844 SIMD/FPU для выполнения операций SIMD (один поток команд-много потоков данных) или операций с плавающей запятой. В малопроизводительном процессорном конвейере 800, на этапе 830 выдается одна команда за раз, и команды выдаются в порядке, в котором они выбираются.

Высокопроизводительный процессорный конвейер 850 по фиг.8В содержит этап 860 выборки для выборки команд из памяти 80, этап 870 декодирования для декодирования выбранных команд, этап 875 переименования для переименования регистров, заданных в декодированных командах, этап диспетчеризации для диспетчеризации команд для исполнения и множество конвейеров исполнения, включающих в себя два целочисленных конвейера 890, 892, конвейер 894 MAC и два конвейера 896, 898 SIMD/FPU. В высокопроизводительном процессорном конвейере 850, этап 880 диспетчеризации является параллельным этапу выдачи, на котором может выдаваться множество команд в различные конвейеры 890, 892, 894, 896, 898 одновременно. На этапе 880 диспетчеризации команды также могут выдаваться с изменением порядка следования. В отличие от малопроизводительного процессорного конвейера 800, длина конвейеров 896, 898 SIMD/FPU может изменяться, это означает то, что можно управлять обработкой операций, выполняемых посредством конвейеров 896, 898 SIMD/FPU, для пропуска определенных этапов. Преимущество такого подхода состоит в том, что, если все из множества конвейеров исполнения имеют разные ресурсы, то нет необходимости искусственно удлинять самый короткий конвейер для того, чтобы его длина стала

идентичной длине самого длинного конвейера, а вместо этого требуется логика для того, чтобы справиться с беспорядочной сущностью результатов, выводимых разными конвейерами (например, заново все упорядочить, если произойдет исключительная ситуация при обработке).

Этап 875 переименования обеспечен для отображения спецификаторов регистра, которые включены в команды программы, и идентификации конкретных регистров архитектуры, если смотреть с точки зрения модели устройства программного управления, в физические регистры, которые являются фактическими регистрами аппаратной платформы. Этап 875 переименования обеспечивает возможность обеспечения микропроцессором большего пула физических регистров, чем существует с точки зрения модели устройства программного управления на микропроцессор. Этот больший пул физических регистров является полезным в течение исполнения с изменением порядка следования команд, так как он обеспечивает возможность избегать конвейерных конфликтов, например конвейерных конфликтов запись-после-записи (WAW), посредством отображения одного регистра архитектуры, заданного в двух или нескольких разных командах, в два или несколько физических регистров, чтобы можно было параллельно исполнять разные команды. За более подробной информацией о способах переименования регистров читателю следует обратиться к заявке на патент США того же заявителя US 2008/114966 и патенту США 7590826.

Малопроизводительный конвейер 800 и высокопроизводительный конвейер 850 отличаются с точки зрения микроархитектуры по нескольким направлениям. Отличия с точки зрения микроархитектуры могут включать в себя:

a) то, что конвейеры содержат разные этапы. Например, высокопроизводительный конвейер содержит этап 875 переименования, который отсутствует в малопроизводительном конвейере 800.

b) то, что этапы конвейеров имеют разные возможности. Например, на этапе 830 выдачи малопроизводительного конвейера 800 команды могут выдаваться только по одной, тогда как на этапе 880 диспетчеризации высокопроизводительного конвейера 850 команды могут выдаваться параллельно. Параллельная выдача команд повышает пропускную способность обработки конвейера и, следовательно, повышает производительность.

c) то, что этапы конвейеров имеют разные длины. Например, этап 870 декодирования высокопроизводительного конвейера 850 может включать в себя три подэтапа, тогда как этап 820 декодирования малопроизводительного конвейера 800 может включать в себя один подэтап. Чем длиннее этап конвейера (большее количество подэтапов), тем больше количество команд, которые одновременно могут находиться "в полете" (в процессе обработки), и, соответственно, больше рабочая частота, с которой конвейер может функционировать, что в результате приводит к более высокому уровню производительности.

d) разное количество конвейеров исполнения (например, высокопроизводительный конвейер 850 содержит больше конвейеров исполнения, чем малопроизводительный конвейер 800). С обеспечением большего количества конвейеров исполнения, может быть обработано параллельно большее количество команд, и, соответственно, производительность увеличивается.

e) обеспечение исполнения команд по порядку (как в конвейере 800) или исполнения с изменением порядка следования команд (как в конвейере 850). Когда команды могут исполняться с изменением порядка следования, тогда производительность повышается, так как исполнение команд может планироваться динамически для оптимизации

производительности. Например, в малопроизводительном конвейере 800 с последовательностью исполнения команд по порядку последовательность команд MAC должны быть исполнены конвейером 842 MAC поочередно до того, как одним из целочисленного конвейера 840 и конвейера 844 SIMD/с плавающей запятой может быть исполнена последующая команда. Напротив, в высокопроизводительном конвейере 850 команды MAC могут исполняться конвейером 894 MAC, в то время как (за исключением любых конвейерных конфликтов данных, которые не могут быть разрешены посредством переименования) последующая команда, использующая другой конвейер 890, 892, 896, 898 исполнения, может исполняться параллельно с командами MAC. Это означает, что исполнение с изменением порядка следования команд может повысить производительность обработки.

Эти и другие примеры отличия микроархитектур в результате приводят к тому, что конвейер 850 обеспечивает более высокую производительность обработки, чем конвейер 800. С другой стороны, упомянутые отличия микроархитектур также вызывают большее потребление энергии конвейером 850, чем конвейером 800. Соответственно, обеспечение конвейеров 800, 850 с разными микроархитектурами дает возможность оптимизировать обработку рабочей нагрузки либо для высокой производительности (посредством использования "большой" схемы 10 обработки, содержащей высокопроизводительный конвейер 850) или для эффективности использования энергии (посредством использования "маленькой" схемы 50 обработки, содержащей малопроизводительный конвейер 800).

На фиг.9 изображен график, иллюстрирующий изменение потребления энергии системой обработки данных, когда выполнение рабочей нагрузки 100 переключается между большой схемой 10 обработки и маленькой схемой 50 обработки.

В точке A фиг.9 рабочая нагрузка 100 выполняется на маленькой компоновке 50 схем обработки, и поэтому потребление энергии является низким. В точке B осуществляется управляющее воздействие для переноса, указывающее на то, что должна выполняться обработка, требующая большой производительности, и поэтому выполнение рабочей нагрузки передается в большую компоновку 10 схем обработки. Тогда потребление энергии возрастает и остается высоким в точке C, пока большая компоновка 10 схем обработки выполняет упомянутую рабочую нагрузку. В точке D предполагается, что функционируют оба больших ядра в комбинации для обработки упомянутой рабочей нагрузки. Если, однако, требования к производительности падают до уровня, когда рабочая нагрузка может обслуживаться только одним из больших ядер, то рабочая нагрузка переносится только на одно из больших ядер, а питание другого выключается, как указано, падением энергии до уровня, близкого к точке E. Однако в точке E осуществляется еще одно управляющее воздействие для переноса (указывающее на то, что требуется возврат к обработке, требующей малой производительности) для запуска переноса выполнения рабочей нагрузки обратно на маленькую компоновку 50 схем обработки.

Когда маленькая компоновка 50 схем обработки начинает обработку рабочей нагрузки по обработке, большая часть большой компоновки схем обработки находится в состоянии экономии энергии, но кэш большой компоновки 10 схем обработки продолжает снабжаться энергией в течение периода просмотра (точка F на фиг.9) для обеспечения возможности извлечения данных, находящихся в кэше, для маленькой компоновки 50 схем обработки. Следовательно, кэш большой компоновки 10 схем обработки вызывает большее потребление энергии в точке F, чем в точке A, в которой снабжается энергией только маленькая компоновка 50 схем обработки. В конце периода

просмотра питание кэша большой компоновки 10 схем обработки выключается, и в точке G потребление энергии возвращается на низкий уровень, когда только маленькая компоновка 50 схем обработки является активной.

Как упоминалось выше, на фиг.9 в течение периода просмотра в точке F энергии потребляется больше, чем в точке G, вследствие того, что в течение периода просмотра кэш большой компоновки схем обработки снабжается энергией. Несмотря на то что увеличение потребления энергии указывается только после перехода большой-маленький, период просмотра также может существовать после перехода маленький-большой, в течение которого данные, находящиеся в кэше маленькой компоновки 50 схем обработки, могут просматриваться для большой компоновки 10 схем обработки блоком 75 управления просмотром. Период просмотра для перехода маленький-большой не указан на фиг.9, так как энергия, потребляемая в результате того, что кэш маленькой компоновки 50 схем обработки находится в состоянии с включенным питанием в течение периода просмотра, является незначительной по сравнению с энергией, потребляемой большой схемой 10 обработки при выполнении рабочей нагрузки по обработке, и поэтому очень маленькое увеличение потребления энергии вследствие снабжения энергией кэша маленькой компоновки 50 схем обработки не заметно на графике по фиг.9.

В вышеописанных вариантах осуществления описывается система, содержащая два или несколько архитектурно совместимых экземпляра процессоров с микроархитектурами, оптимизированными по эффективности использования энергии или производительности. Состояние архитектуры, требуемое операционной системой и приложениями, может переключаться между экземплярами процессоров, в зависимости от требуемого уровня энергии/производительности, для обеспечения возможности переключения всей рабочей нагрузки между экземплярами процессоров. В одном варианте осуществления, только один из экземпляров процессоров выполняет рабочую нагрузку в любой данный момент времени, при этом другой экземпляр (процессора) находится в режиме экономии энергии или в процессе перехода в режим экономии энергии/выхода из него.

В одном варианте осуществления, экземпляры процессоров могут быть выполнены в виде аппаратных кэшей, когерентных друг с другом, для сокращения количества времени, энергии и сложности аппаратного обеспечения, связанных с переключением состояния архитектуры с исходного процессора на целевой процессор. Это сокращает время выполнения операции переключения, что увеличивает возможности использования способов вариантов осуществления.

Такие системы могут быть использованы во многих ситуациях, в которых эффективность использования энергии является важной для времени работы батареи и/или управления нагревом, и диапазон производительности является таким, что процессор с более эффективным использованием энергии может использоваться для меньших рабочих нагрузок по обработке, в то время как более высокопроизводительный процессор может использоваться для больших рабочих нагрузок по обработке.

Так как упомянутые два или несколько экземпляров (процессоров) являются архитектурно совместимыми, то с точки зрения приложения единственным различием между двумя процессорами является имеющаяся в распоряжении производительность. Посредством способов одного варианта осуществления все требуемое состояние архитектуры может перемещаться между процессорами без необходимости обращения к операционной системе, так что тогда то, на каком процессоре операционная система и приложения выполняются, является прозрачным для этой операционной системы и

приложений, выполняющихся под ее управлением.

При использовании архитектурно совместимых экземпляров процессора, как описано в вышеупомянутых вариантах осуществления, общий объем состояние архитектуры, который должен быть перенесен, легко может поместиться в пределах кэша данных, и так как в современных системах обработки данных часто реализуется когерентность кэша, то при сохранении состояния архитектуры, которое должно быть переключено, внутри кэша данных, целевой процессор может быстро просмотреть это состояние при эффективном использовании энергии с использованием структур существующих схем.

В одном варианте осуществления, используется механизм переключения для обеспечения пределов по нагреву, чтобы система обработки данных не была нарушена. В частности, при приближении к пределам по нагреву, вся рабочая нагрузка может быть переключена на процессор с более эффективным использованием энергии с обеспечением возможности охлаждения всей системы, в то время как продолжается исполнение программы, хотя и с более низкой пропускной способностью.

Несмотря на то что в данном документе описывается конкретный вариант осуществления, очевидно, что изобретение не ограничивается им, и что может быть сделано множество его модификаций и дополнений к нему в рамках объема изобретения. Например, могут быть осуществлены различные комбинации признаков нижеследующих зависимых пунктов многозвенной формулы изобретения с признаками независимых пунктов формулы изобретения, не выходя за пределы объема настоящего изобретения.

Формула изобретения

1. Устройство обработки данных, содержащее:

первую компоновку схем обработки для выполнения операций обработки данных,

вторую компоновку схем обработки для выполнения операций обработки данных,

причем первая компоновка схем обработки является архитектурно совместимой со второй компоновкой схем обработки, так что рабочая нагрузка, выполняемая устройством обработки данных, может выполняться или на первой компоновке схем обработки, или на второй компоновке схем обработки, причем упомянутая рабочая нагрузка содержит по меньшей мере одно приложение и по меньшей мере одну операционную систему для выполнения упомянутого по меньшей мере одного приложения,

причем первая компоновка схем обработки отличается от второй компоновки схем обработки с точки зрения микроархитектуры, так что производительность первой компоновки схем обработки отличается от производительности второй компоновки схем обработки,

причем первая и вторая компоновки схем обработки сконфигурированы так, что рабочая нагрузка выполняется одной из первой компоновки схем обработки и второй компоновки схем обработки в любой момент времени,

контроллер переключения, реагирующий на управляющее воздействие для переноса, для выполнения операции передачи обслуживания для переноса выполнения рабочей нагрузки из исходной компоновки схем обработки на целевую компоновку схем обработки, причем исходной компоновкой схем обработки является одна из первой компоновки схем обработки и второй компоновки схем обработки, а целевой компоновкой схем обработки является другая из первой компоновки схем обработки и второй компоновки схем обработки,

причем контроллер переключения выполнен с возможностью, во время операции передачи обслуживания:

(i) обеспечивать предоставление исходной компоновкой схем обработки своего текущего состояния архитектуры целевой компоновке схем обработки, причем текущим состоянием архитектуры является то состояние, которое не доступно из совместно используемой памяти, разделяемой между первой и второй компоновкой схем обработки, в момент инициирования операции передачи обслуживания, и которое является необходимым целевой компоновке схем обработки, чтобы успешно принять на себя выполнение рабочей нагрузки от исходной компоновки схем обработки, и

(ii) маскировать заранее определенную конкретную для процессора информацию о конфигурации от упомянутой по меньшей мере одной операционной системы, так что перенос рабочей нагрузки является прозрачным для упомянутой по меньшей мере одной операционной системы.

2. Устройство обработки данных по п.1, дополнительно содержащее:

компоновку схем управления питанием для независимого управления питанием, подаваемым на первую компоновку схем обработки и вторую компоновку схем обработки,

причем до осуществления управляющего воздействия для переноса целевая компоновка схем обработки находится в режиме экономии энергии, а во время операции передачи обслуживания компоновка схем управления питанием вызывает выход целевой компоновки схем обработки из режима экономии энергии до того, как целевая компоновка для обработки примет на себя выполнение рабочей нагрузки.

3. Устройство обработки данных по п.2, в котором после операции передачи обслуживания компоновка схем управления питанием вызывает переход исходной компоновки схем обработки в режим экономии энергии.

4. Устройство обработки данных по любому предыдущему пункту, в котором во время операции переноса контроллер переключения вызывает применение исходной компоновкой схем обработки ускоренного механизма для предоставления своего текущего состояния архитектуры целевой компоновке схем обработки без обращения целевой компоновки схем обработки к совместно используемой памяти для получения текущего состояния архитектуры.

5. Устройство обработки данных по п.4, в котором:

по меньшей мере, упомянутая исходная компоновка схем имеет ассоциированный кэш,

устройство обработки данных также содержит компоновку схем управления просмотром, и

ускоренный механизм содержит перенос текущего состояния архитектуры в целевую компоновку схем обработки с использованием ассоциированного кэша упомянутой исходной компоновки схем и упомянутой компоновки схем управления просмотром.

6. Устройство обработки данных по п.5, в котором ускоренный механизм является механизмом сохранения и восстановления, который вызывает сохранение исходной компоновкой схем обработки своего текущего состояния архитектуры в ассоциированном с ней кэше, и вызывает выполнение целевой компоновкой схем обработки операции восстановления, посредством которой компоновка схем управления просмотром извлекает текущее состояние архитектуры из ассоциированного кэша исходной компоновки схем обработки и обеспечивает это извлеченное текущее состояние архитектуры в целевую компоновку схем обработки.

7. Устройство обработки данных по п.5, в котором целевая компоновка схем обработки имеет ассоциированный кэш, в котором переносимое состояние архитектуры, полученное компоновкой схем управления просмотром, сохраняется для обращения

к нему со стороны целевой компоновки схем обработки.

8. Устройство обработки данных по п.4, в котором ускоренный механизм содержит выделенную шину между исходной компоновкой схем обработки и целевой компоновкой схем обработки, по которой исходная компоновка схем обработки обеспечивает свое текущее состояние архитектуры в целевую компоновку схем обработки.

9. Устройство обработки данных по одному из пп.1, 2, 3, в котором контроллер переключения содержит, по меньшей мере, программное обеспечение виртуализации, логически отделяющее упомянутую по меньшей мере одну операционную систему от первой компоновки схем обработки и второй компоновки схем обработки.

10. Устройство обработки данных по одному из пп.1, 2, 3, в котором тайминг управляющего воздействия для переноса выбирается так, чтобы повысить эффективность использования энергии устройства обработки данных.

11. Устройство обработки данных по одному из пп.1, 2, 3, в котором упомянутое состояние архитектуры содержит, по меньшей мере, текущее значение одного или нескольких специальных регистров исходной компоновки схем обработки, в том числе значение счетчика команд.

12. Устройство обработки данных по п.11, в котором упомянутое состояние архитектуры дополнительно содержит текущие значения, хранящиеся в архитектурном регистровом файле исходной компоновки схем обработки.

13. Устройство обработки данных по одному из пп.1, 2, 3, в котором по меньшей мере одна из первой компоновки схем обработки и второй компоновки схем обработки содержит один блок обработки.

14. Устройство обработки данных по одному из пп. 1, 2, 3, в котором по меньшей мере одна из первой компоновки схем обработки и второй компоновки схем обработки содержит группу блоков обработки с идентичной микроархитектурой.

15. Устройство обработки данных по п.2, в котором упомянутый режим экономии энергии является одним из:

режима с отключенным питанием,

режима частичного/полного сохранения данных,

спящего режима, или

режима ожидания.

16. Устройство обработки данных по одному из пп.1, 2, 3, 15, в котором первая компоновка схем обработки и вторая компоновка схем обработки отличается с точки зрения микроархитектуры наличием по меньшей мере одного из:

разных длин исполнительного конвейера, или

разных исполнительных ресурсов.

17. Устройство обработки данных по одному из пп.1, 2, 3, 15, в котором исходная компоновка схем обработки имеет более высокую производительность, чем целевая компоновка схем обработки, и устройство обработки данных дополнительно содержит:

компоновку схем текущего контроля нагрева для текущего контроля теплоотдачи исходной компоновки схем обработки и для запуска упомянутого управляющего воздействия для переноса, когда упомянутая теплоотдача достигнет заранее определенного уровня.

18. Устройство обработки данных по одному из пп.1, 2, 3, 15, в котором первая компоновка схем обработки и вторая компоновки схем обработки находятся на одной интегральной схеме.

19. Устройство обработки данных, содержащее:

первое средство обработки для выполнения операций обработки данных,

второе средство обработки для выполнения операций обработки данных,

причем первое средство обработки является архитектурно совместимым со вторым средством обработки, так что рабочая нагрузка, выполняемая устройством обработки данных, может выполняться или на первом средстве обработки, или на втором средстве обработки, причем упомянутая рабочая нагрузка содержит по меньшей мере одно приложение и по меньшей мере одну операционную систему для выполнения упомянутого по меньшей мере одного приложения,

причем первое средство обработки отличается от второго средства обработки с точки зрения микроархитектуры, так что производительность первого средства обработки отличается от производительности второго средства обработки,

причем первое и второе средство обработки сконфигурированы так, что рабочая нагрузка выполняется одним из первого средства обработки и второго средства обработки в любой момент времени,

средство управления переносом, реагирующее на управляющее воздействие для переноса, для выполнения операции передачи обслуживания для переноса выполнения рабочей нагрузки из исходного средства обработки на целевое средство обработки, причем исходным средством обработки является одно из первого средства обработки и второго средства обработки, а целевым средством обработки является другое из первого средства обработки и второго средства обработки,

причем средство управления переносом выполнено с возможностью, во время операции передачи обслуживания:

(i) обеспечивать предоставление исходным средством обработки своего текущего состояния архитектуры целевому средству обработки, причем текущим состоянием архитектуры является то состояние, которое не доступно из совместно используемого средства памяти, разделяемого между первым и вторым средством обработки, в момент инициирования операции передачи обслуживания, и которое является необходимым целевому средству обработки, чтобы успешно принять на себя выполнение рабочей нагрузки от исходного средства обработки, и

(ii) маскировать заранее определенную конкретную для процессора информацию о конфигурации от упомянутой по меньшей мере одной операционной системы, так что перенос рабочей нагрузки является прозрачным для упомянутой по меньшей мере одной операционной системы.

20. Способ функционирования устройства обработки данных, содержащего первую компоновку схем обработки для выполнения операций обработки данных и вторую компоновку схем обработки для выполнения операций обработки данных, причем первая компоновка схем обработки является архитектурно совместимой со второй компоновкой схем обработки, так что рабочая нагрузка, выполняемая устройством обработки данных, может выполняться или на первой компоновке схем обработки, или на второй компоновке схем обработки, причем упомянутая рабочая нагрузка содержит по меньшей мере одно приложение и по меньшей мере одну операционную систему для выполнения упомянутого по меньшей мере одного приложения, и причем первая компоновка схем обработки отличается от второй компоновки схем обработки с точки зрения микроархитектуры, так что производительность первой компоновки схем обработки отличается от производительности второй компоновки схем обработки, причем способ содержит этапы, на которых:

выполняют, в любой момент времени, рабочую нагрузку на одной из первой компоновки схем обработки и второй компоновки схем обработки,

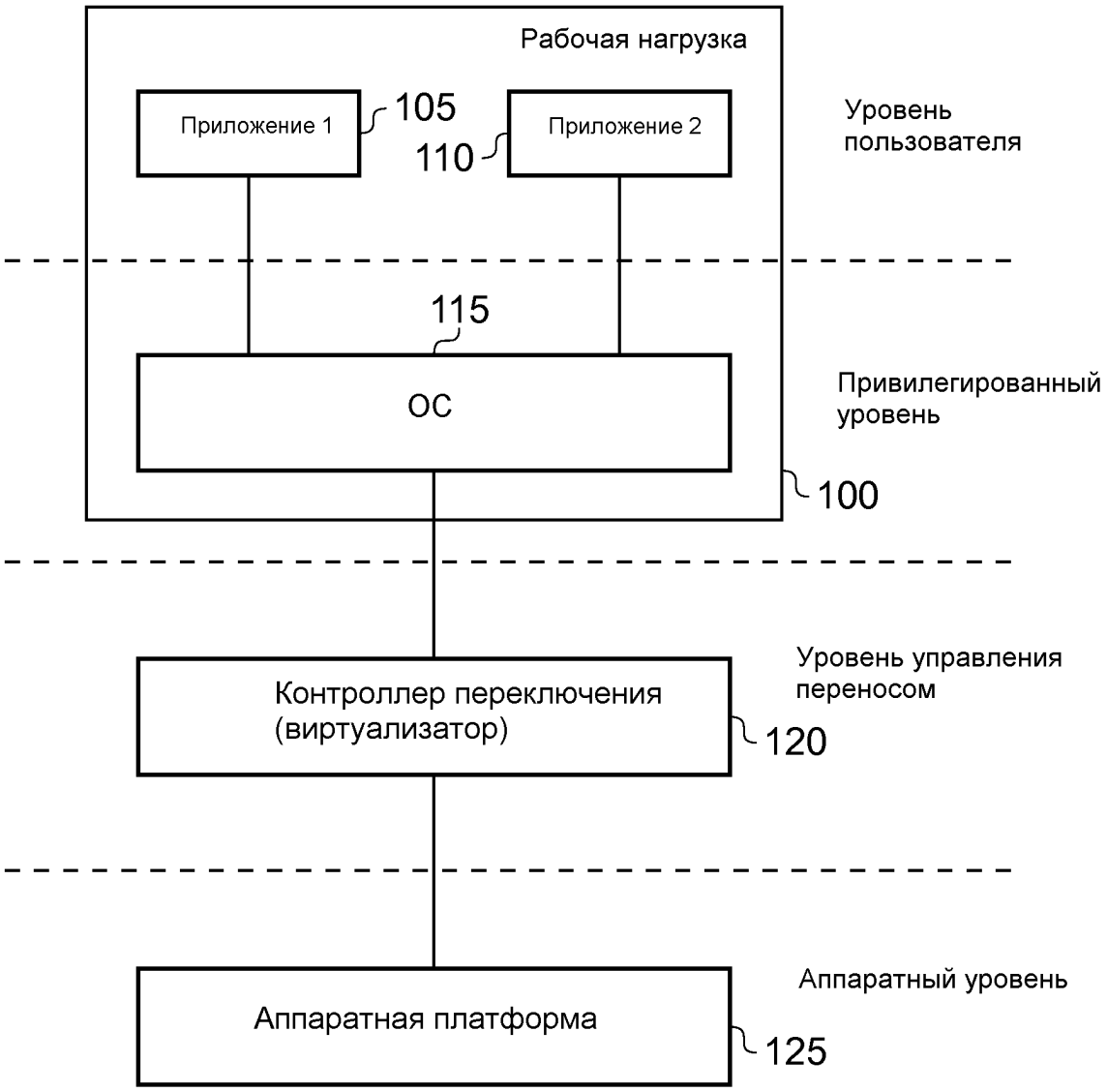
выполняют, в ответ на управляющее воздействие для переноса, операцию передачи

обслуживания для переноса выполнения рабочей нагрузки из исходной компоновки схем обработки на целевую компоновку схем обработки, причем исходной компоновкой схем обработки является одна из первой компоновки схем обработки и второй компоновки схем обработки, а целевой компоновкой схем обработки является другая из первой компоновки схем обработки и второй компоновки схем обработки,

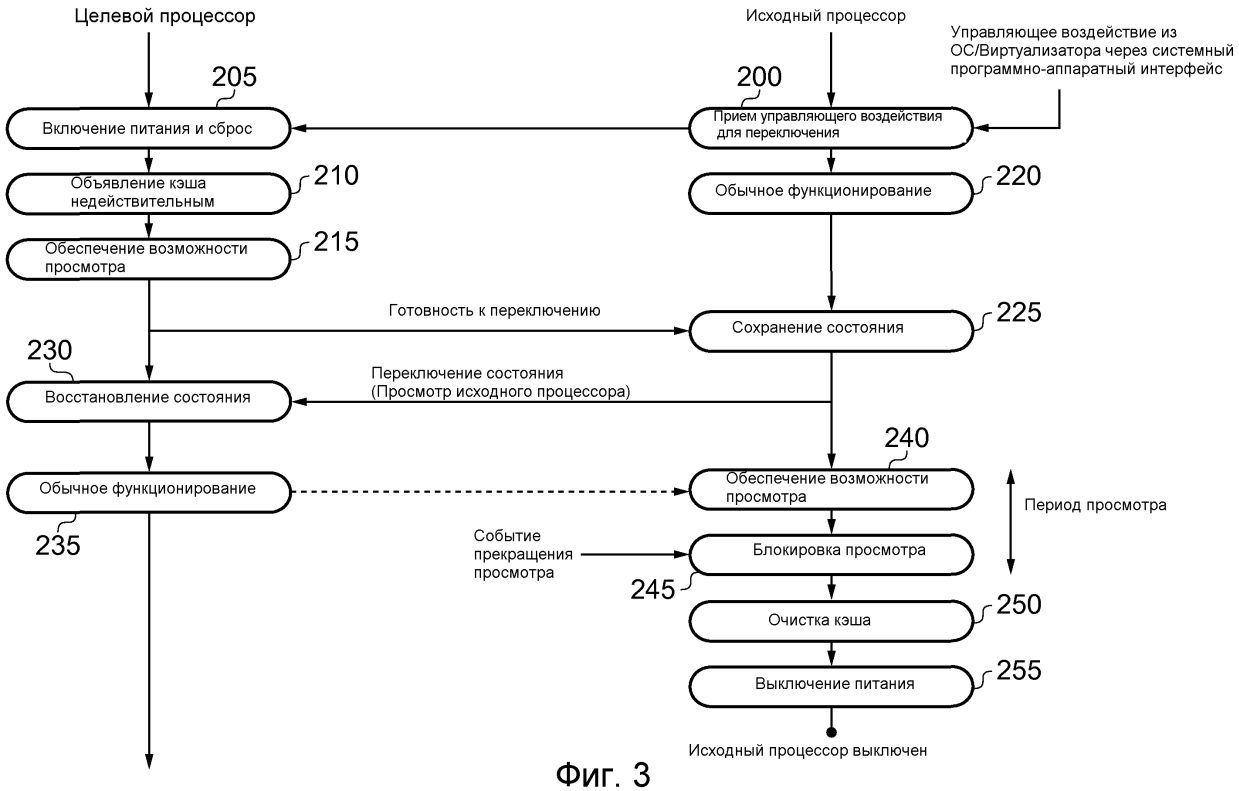
во время операции передачи обслуживания:

(i) обеспечивают предоставление исходной компоновкой схем обработки своего текущего состояния архитектуры целевой компоновке схем обработки, причем текущим состоянием архитектуры является то состояние, которое не доступно из совместно используемой памяти, разделяемой между первой и второй компоновкой схем обработки, в момент инициирования операции передачи обслуживания, и которое является необходимым целевой компоновке схем обработки, чтобы успешно принять на себя выполнение рабочей нагрузки от исходной компоновки схем обработки, и

(ii) маскируют заранее определенную конкретную для процессора информацию о конфигурации процессора от упомянутой по меньшей мере одной операционной системы, так что передача рабочей нагрузки является прозрачной для упомянутой по меньшей мере одной операционной системы.

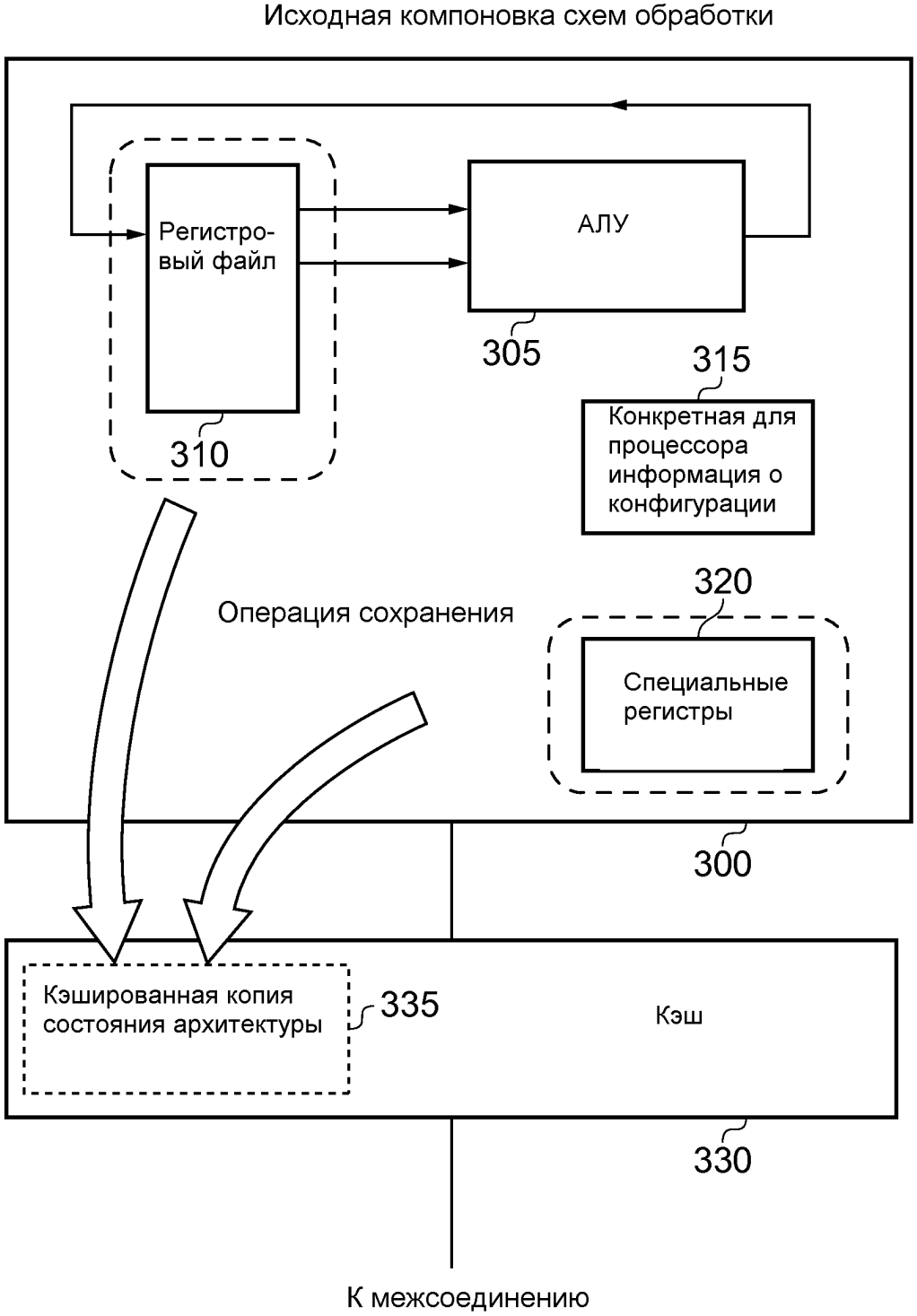

Рабочая нагрузка
Приложение 1
105
110
Приложение 2
Уровень пользователя
115
ОС
Привилегированный уровень
100
Контроллер переключения (виртуализатор)
Уровень управления переносом
120
Аппаратная платформа
Аппаратный уровень
125

Фиг. 2

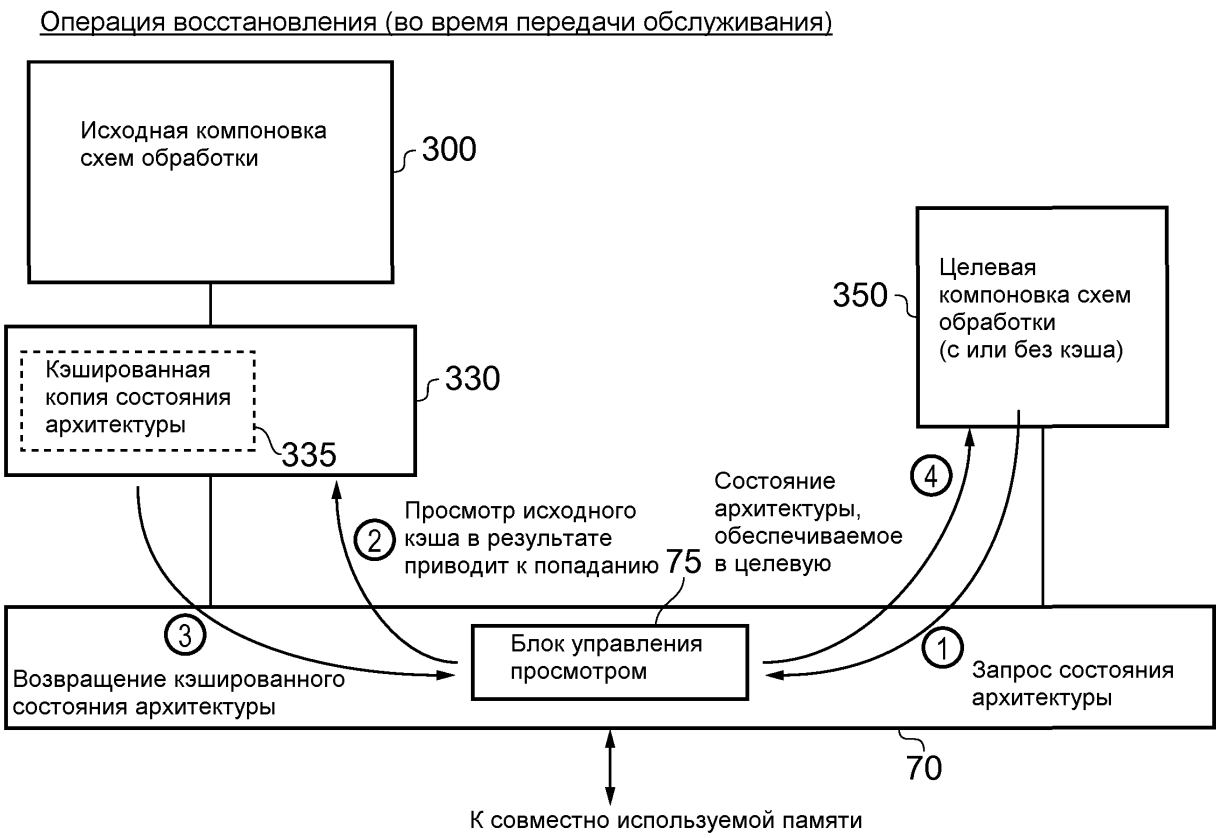

Целевой процессор
Исходный процессор
Управляющее воздействие из ОС/Виртуализатора через системный программно-аппаратный интерфейс
205
200
Включение питания и сброс
Прием управляющего воздействия для переключения
Объявление кэша недействительным
210
Обычное функционирование
220
Обеспечение возможности просмотра
215
Готовность к переключению
Сохранение состояния
225
230
Переключение состояния (Просмотр исходного процессора)
Восстановление состояния
240
Обычное функционирование
Обеспечение возможности просмотра
Период просмотра
235
Событие прекращения просмотра
Блокировка просмотра
245
Очистка кэша
250
Выключение питания
255
Исходный процессор выключен

Фиг. 3

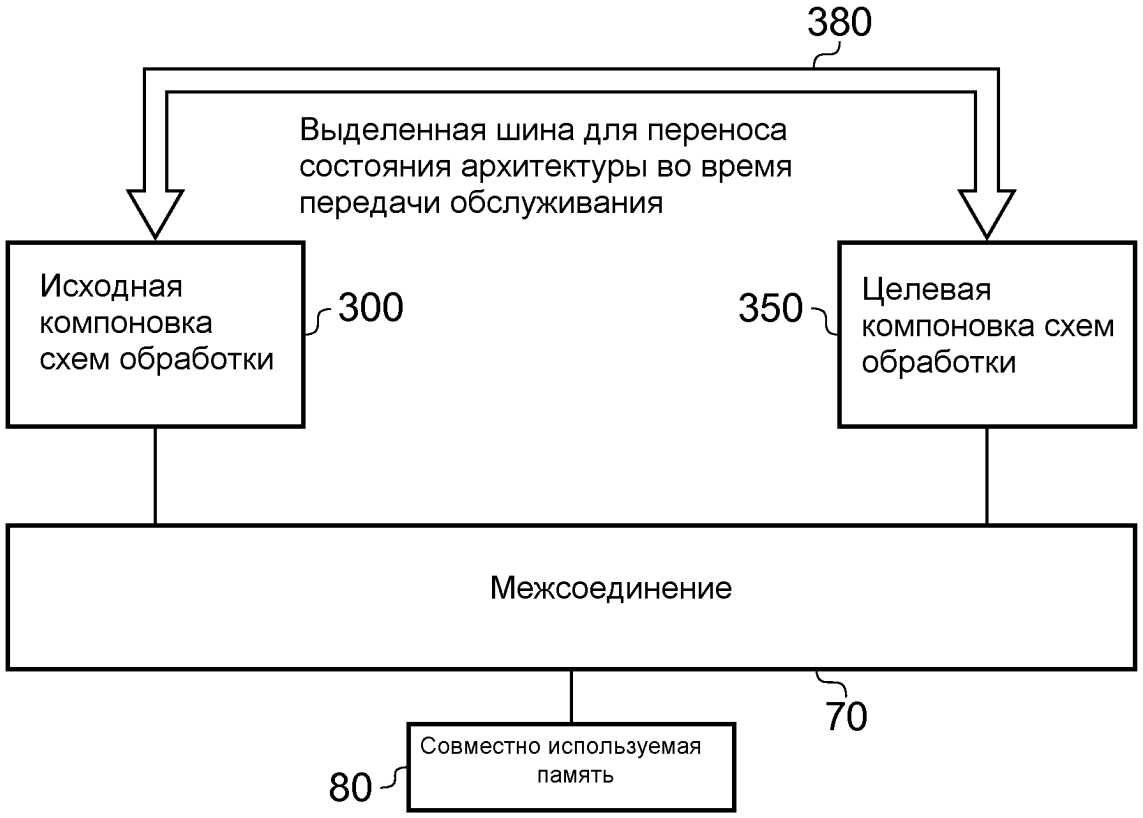
Операция сохранения (во время передачи обслуживания)



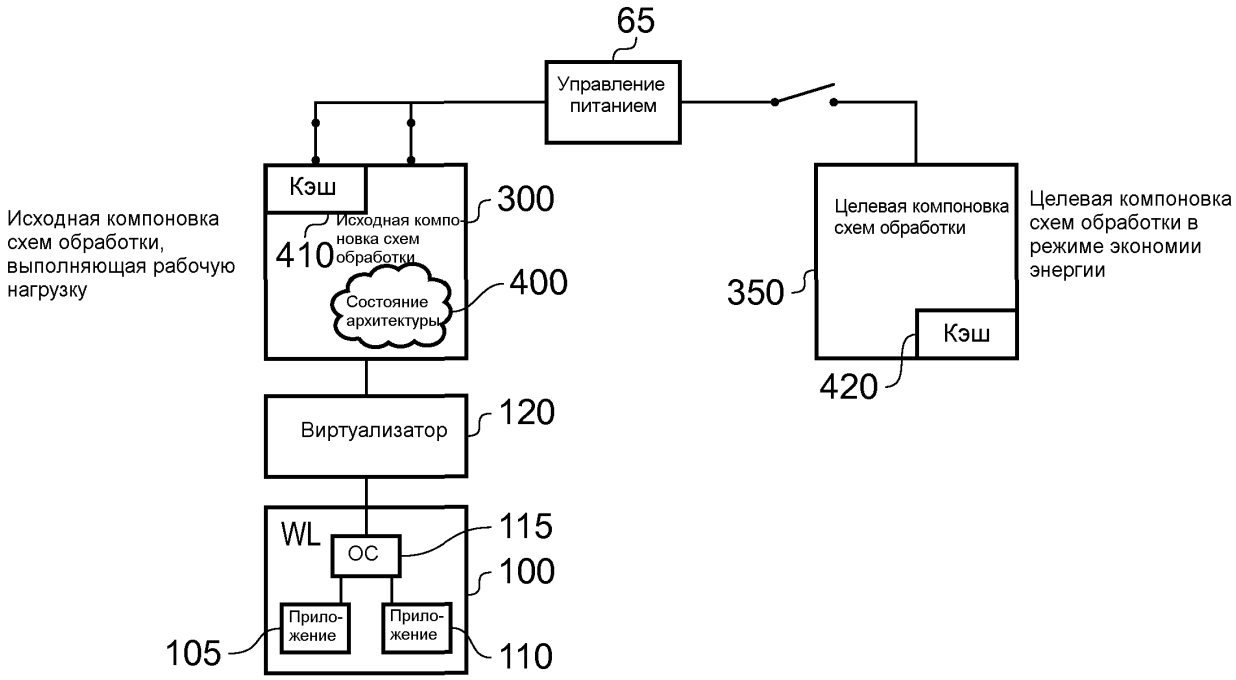
Фиг. 4A


Операция восстановления (во время передачи обслуживания)
Исходная компоновка схем обработки
300
Кэшированная копия состояния архитектуры
335
330
Целевая компоновка схем обработки (с или без кэша)
350
Просмотр исходного кэша в результате приводит к попаданию
2
75
Состояние архитектуры, обеспечиваемое в целевую
4
3
Возвращение кэшированного состояния архитектуры
Блок управления просмотром
1
Запрос состояния архитектуры
70
К совместно используемой памяти

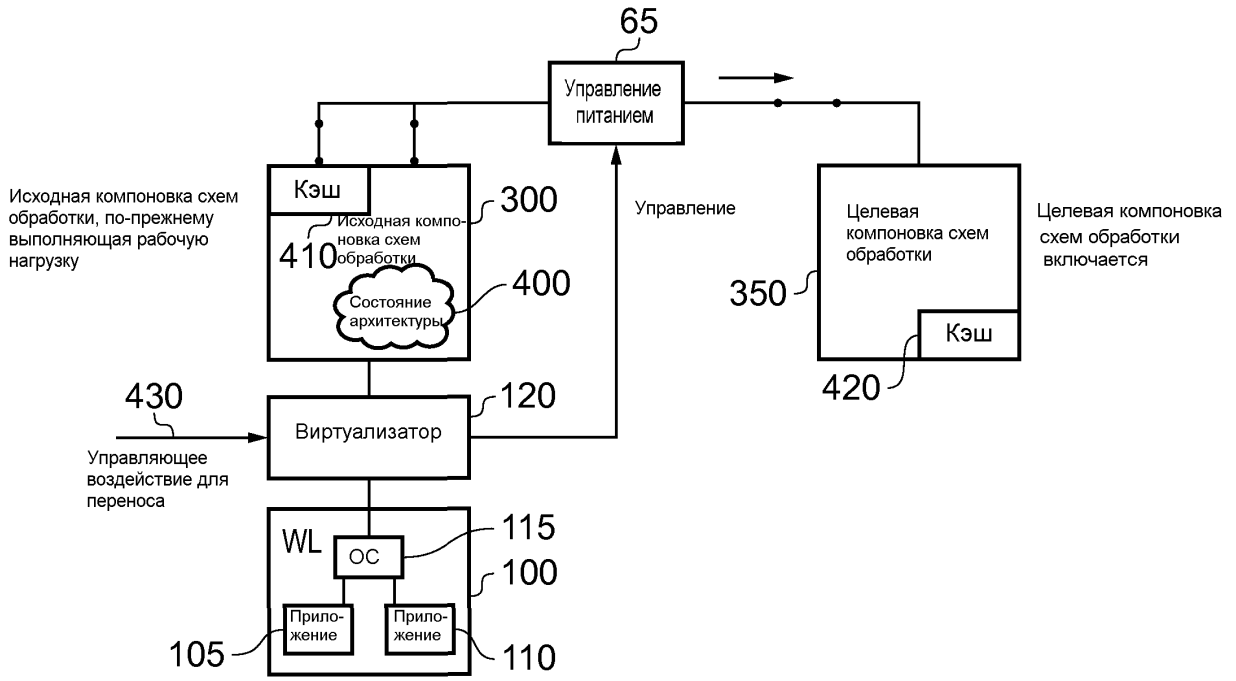
Фиг. 4B


380
Выделенная шина для переноса состояния архитектуры во время передачи обслуживания
Исходная компоновка схем обработки
300
350
Целевая компоновка схем обработки
Межсоединение
70
Совместно используемая память
80

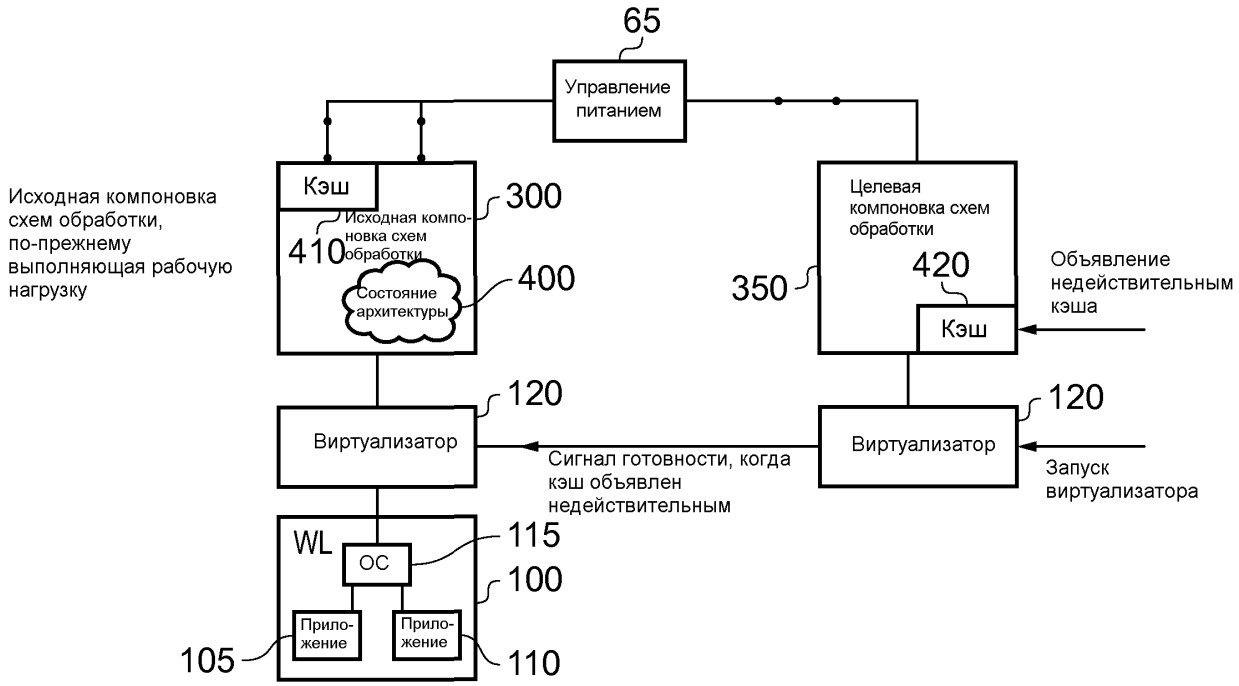
Фиг. 5


65
Управление питанием
Кэш
300
Исходная компоновка схем обработки
410
400
Состояние архитектуры
Исходная компоновка схем обработки, выполняющая рабочую нагрузку
Виртуализатор
120
WL
ОС
115
100
Прило-жение
Прило-жение
105
110
Целевая компоновка схем обработки
350
Кэш
420
Целевая компоновка схем обработки в режиме экономии энергии

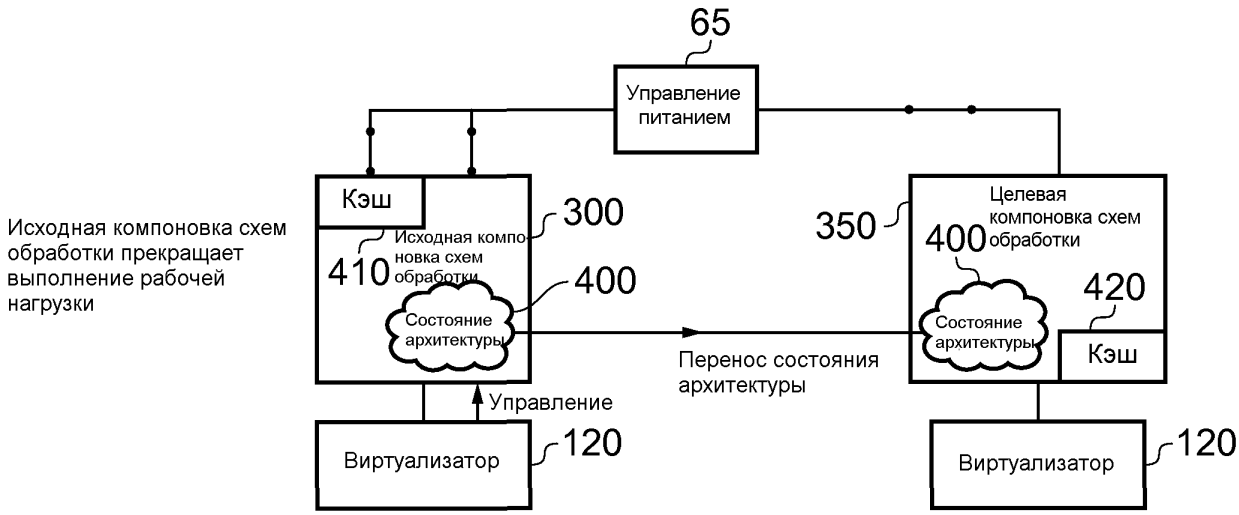
Фиг. 6A


65
Управление питанием
Кэш
300
Исходная компоновка схем обработки
410
400
Состояние архитектуры
Исходная компоновка схем обработки, по-прежнему выполняющая рабочую нагрузку
Управление
430
Виртуализатор
120
Управляющее воздействие для переноса
WL
ОС
115
100
Прило-жение
Прило-жение
105
110
Целевая компоновка схем обработки
350
Кэш
420
Целевая компоновка схем обработки включается

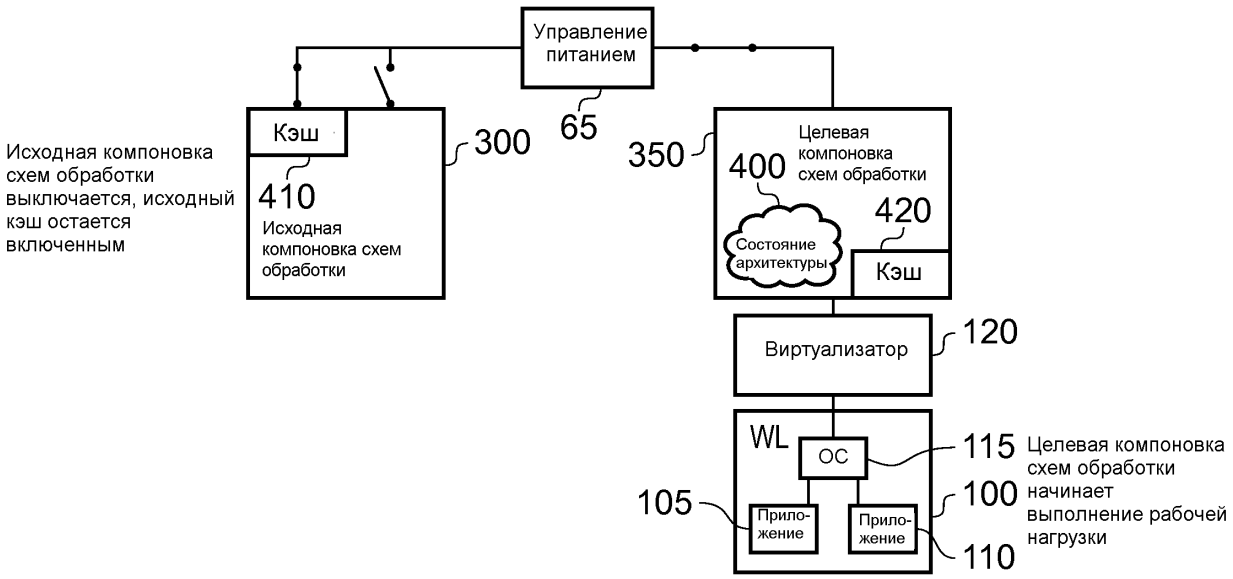
Фиг. 6B


65
Управление питанием
Исходная компоновка схем обработки, по-прежнему выполняющая рабочую нагрузку
Кэш
300
Исходная компоновка схем обработки
410
Состояние архитектуры
400
Целевая компоновка схем обработки
420
350
Кэш
Объявление недействительным кэша
120
Виртуализатор
Сигнал готовности, когда кэш объявлен недействительным
Виртуализатор
120
Запуск виртуализатора
WL
ОС
115
100
Прило-жение
Прило-жение
105
110

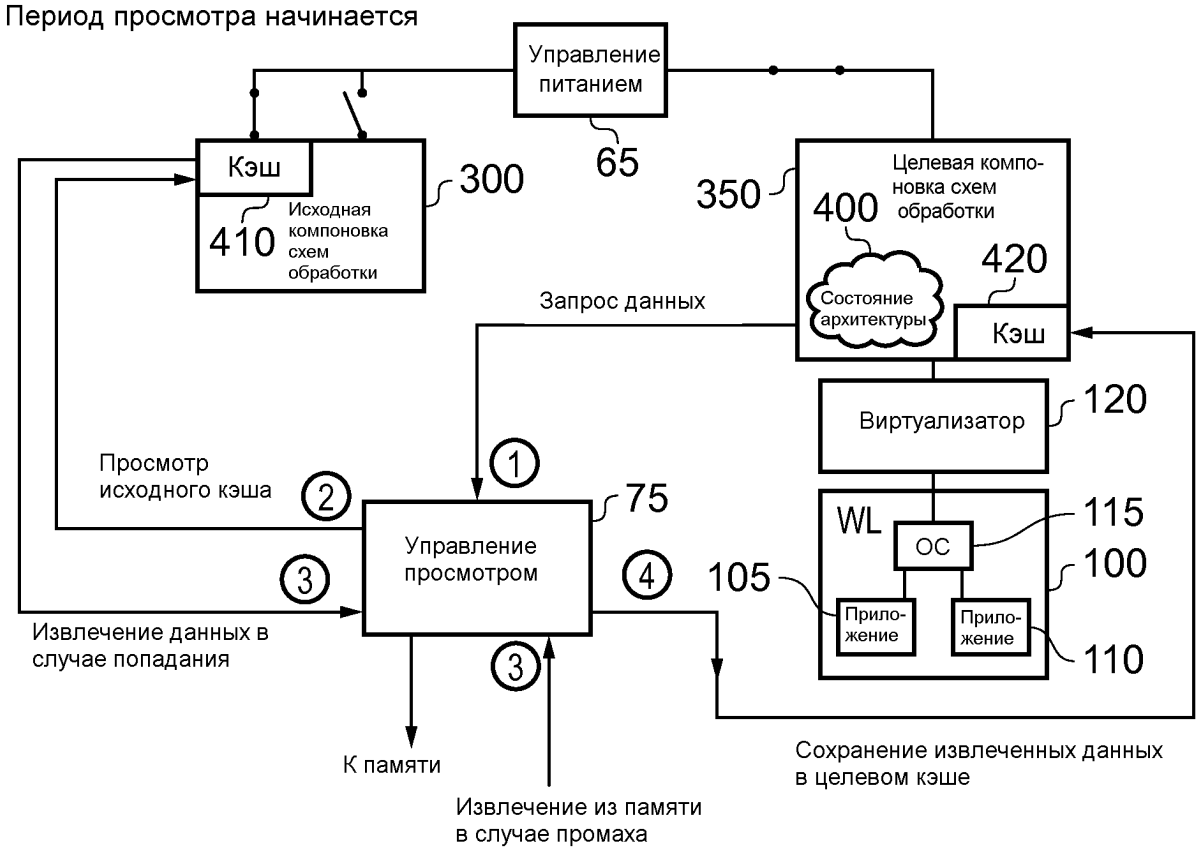
Фиг. 6C


65
Управление питанием
Исходная компоновка схем обработки прекращает выполнение рабочей нагрузки
Кэш
300
Исходная компоновка схем обработки
410
Состояние архитектуры
400
Целевая компоновка схем обработки
350
400
420
Состояние архитектуры
Кэш
Перенос состояния архитектуры
Управление
Виртуализатор
120
Виртуализатор
120

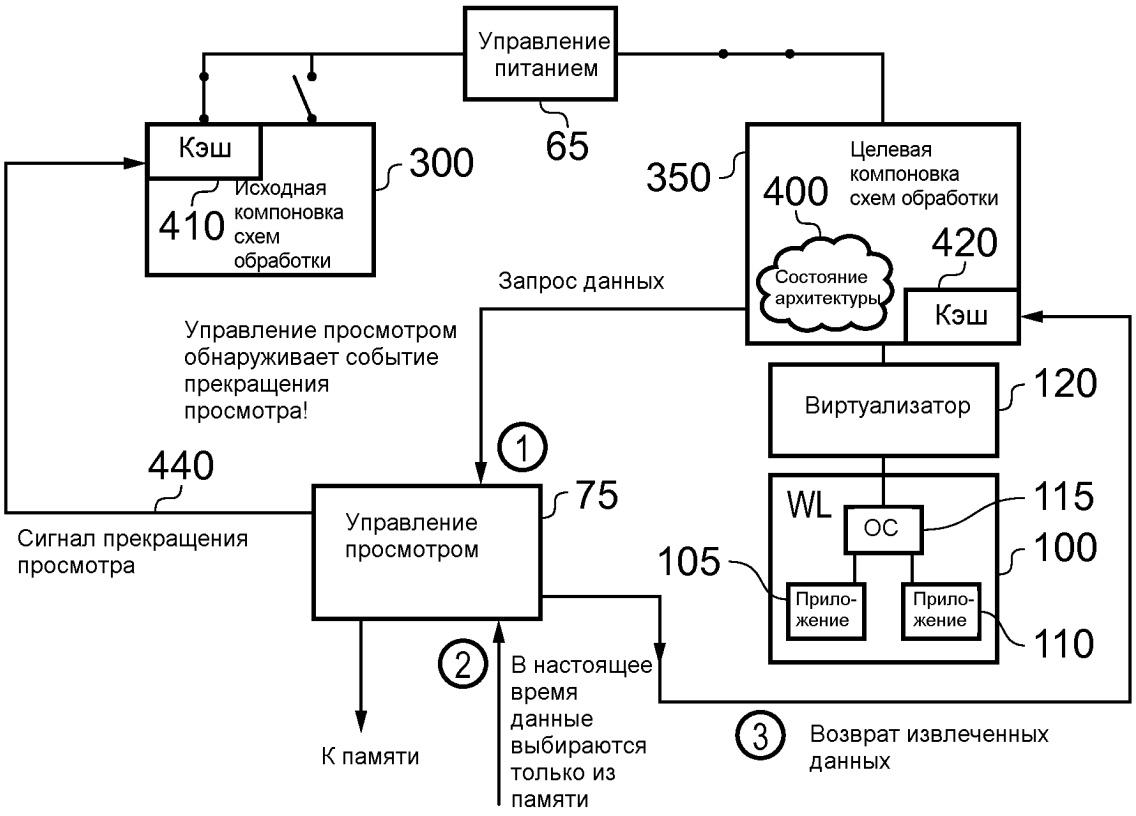
Фиг. 6D


Управление питанием
65
Кэш
410
Исходная компоновка схем обработки
300
Исходная компоновка схем обработки выключается, исходный кэш остается включенным
350
Целевая компоновка схем обработки
400
Состояние архитектуры
420
Кэш
Виртуализатор
120
WL
ОС
115
105
Прило-жение
Прило-жение
100
110
Целевая компоновка схем обработки начинает выполнение рабочей нагрузки

Фиг. 6E

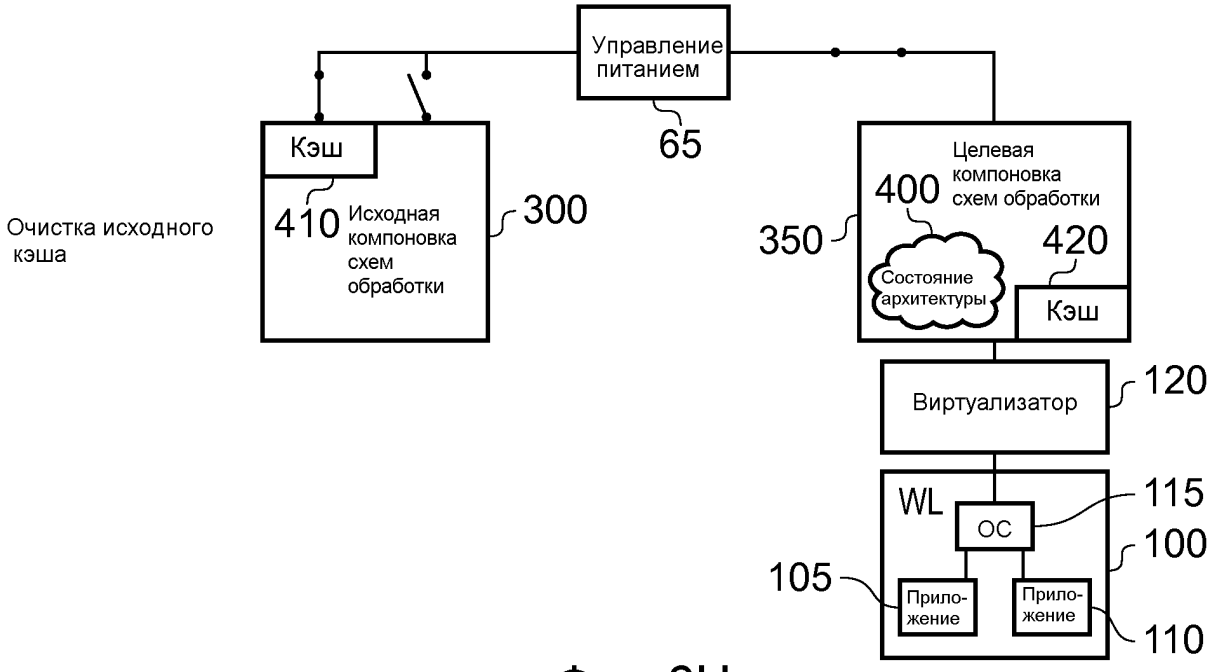

Период просмотра начинается
Управление питанием
65
Кэш
410
Исходная компоновка схем обработки
300
350
Целевая компоновка схем обработки
400
Состояние архитектуры
420
Кэш
Запрос данных
Виртуализатор
120
Просмотр исходного кэша
1
2
Управление просмотром
75
3
4
105
WL
ОС
115
100
Прило-жение
Прило-жение
110
Извлечение данных в случае попадания
3
К памяти
Извлечение из памяти в случае промаха
Сохранение извлеченных данных в целевом кэше

Фиг. 6F

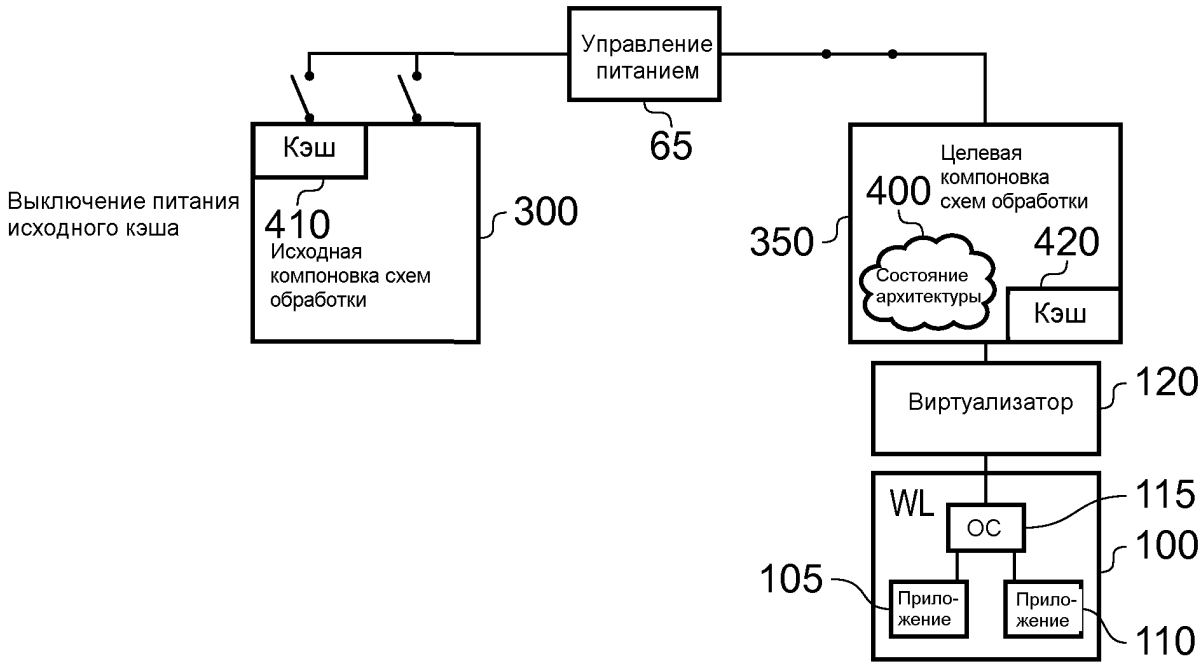


Фиг. 6G

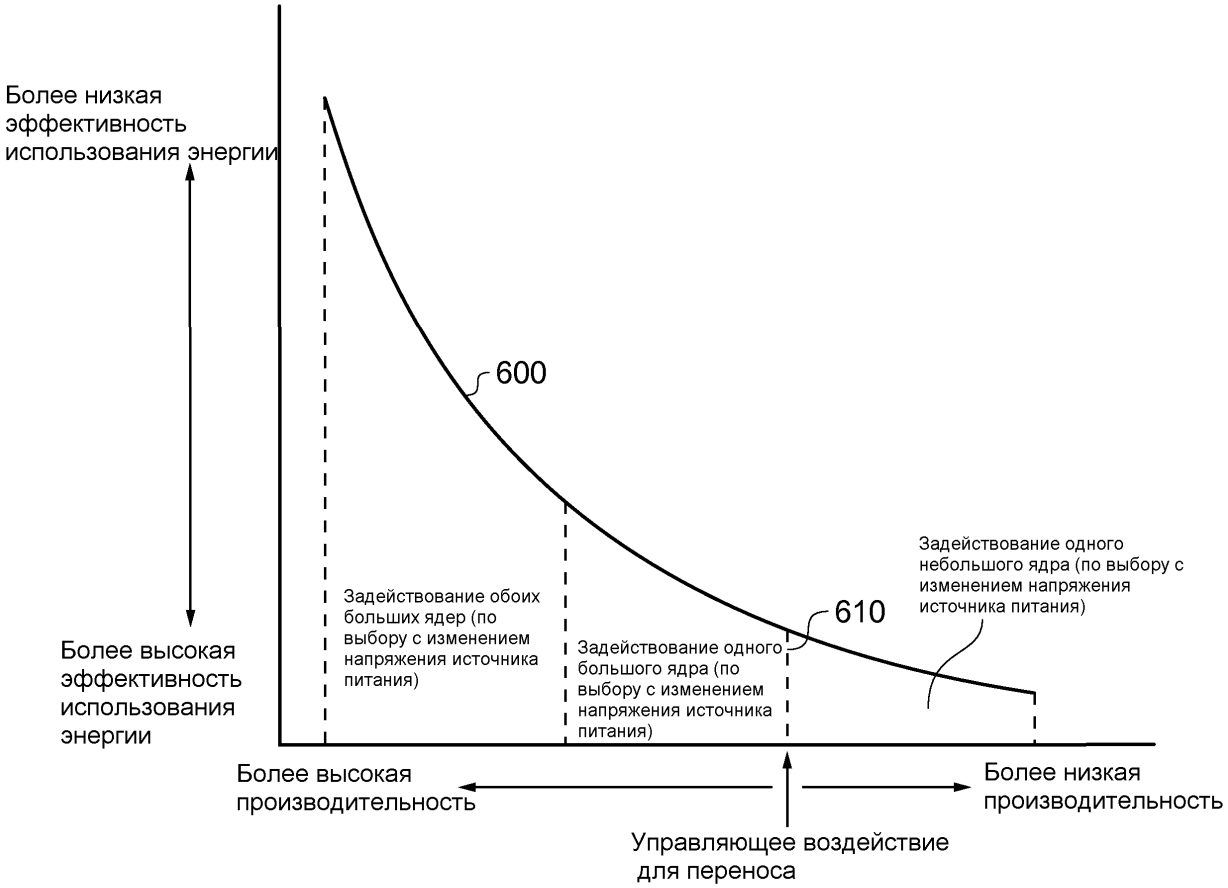
окончание периода просмотра



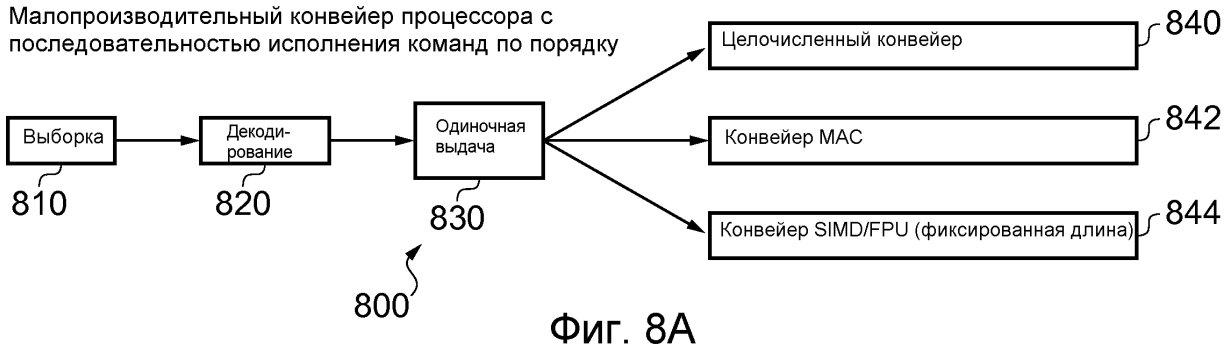
Фиг. 6H


Управление питанием
65
Кэш
410
Исходная компоновка схем обработки
300
Выключение питания исходного кэша
Целевая компоновка схем обработки
400
350
420
Состояние архитектуры
Кэш
Виртуализатор
120
WL
ОС
115
100
105
Прило-жение
Прило-жение
110

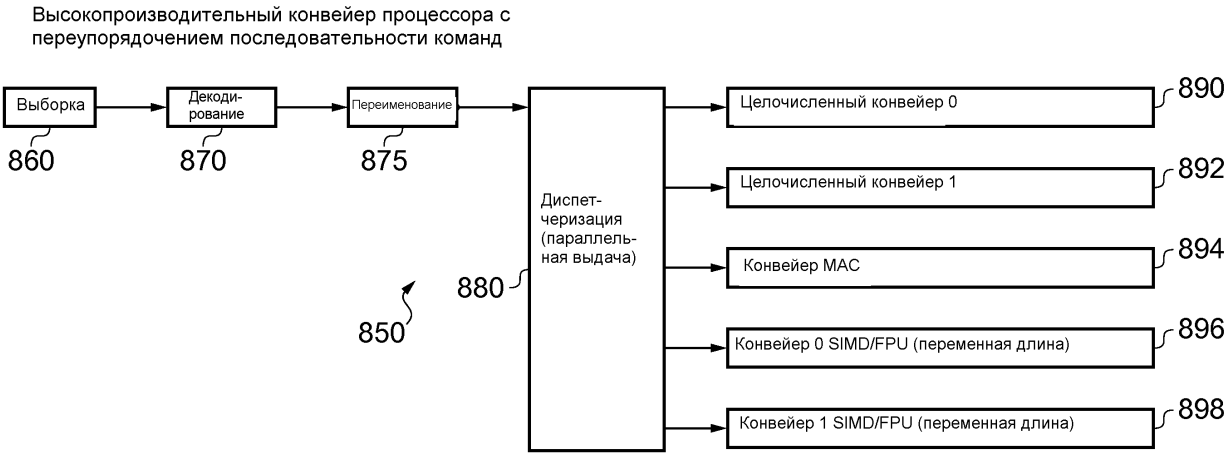
Фиг. 6I


Более низкая эффективность использования энергии
600
Задействование обоих больших ядер (по выбору с изменением напряжения источника питания)
Задействование одного большого ядра (по выбору с изменением напряжения источника питания)
610
Задействование одного небольшого ядра (по выбору с изменением напряжения источника питания)
Более высокая эффективность использования энергии
Более высокая производительность
Более низкая производительность
Управляющее воздействие для переноса

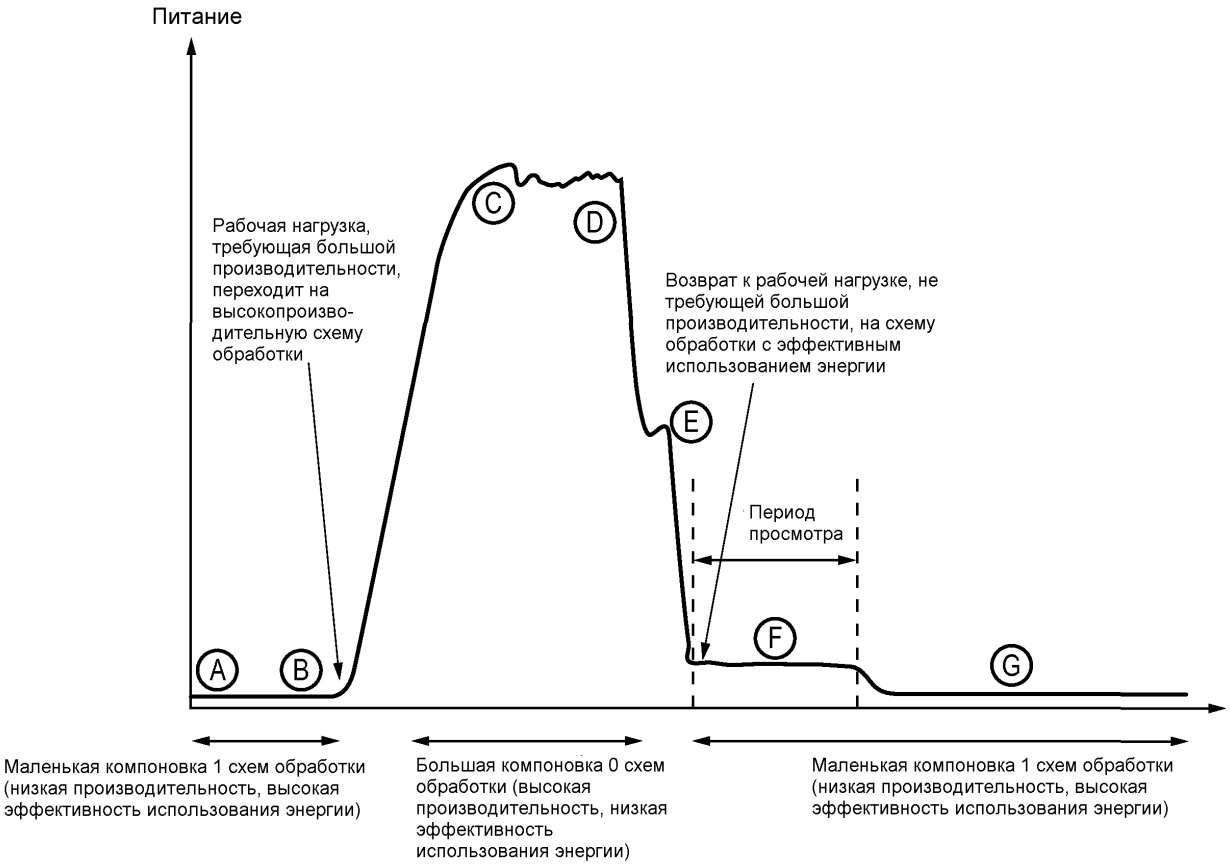
Фиг. 7



Фиг. 8A



Фиг. 8B



Фиг. 9