



(19) **RU** ⁽¹¹⁾ **2 233 470** ⁽¹³⁾ **C2**
(51) МПК⁷ **G 06 F 15/00, 9/38**

РОССИЙСКОЕ АГЕНТСТВО
ПО ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) **ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ РОССИЙСКОЙ ФЕДЕРАЦИИ**

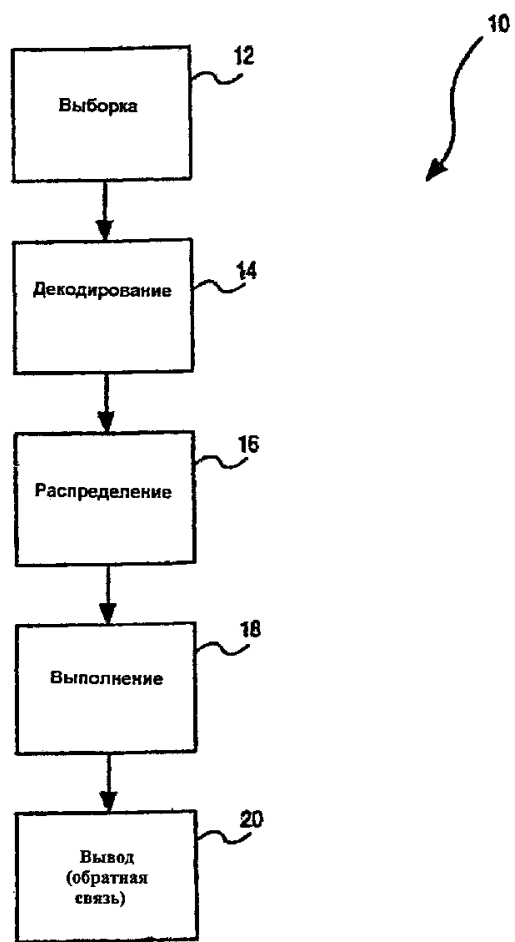
(21), (22) Заявка: 2002118211/09, 11.10.2000
(24) Дата начала действия патента:
11.10.2000ii.1-24
(30) Приоритет: 09.12.1999 US 09/458,589
(43) Дата публикации заявки: 10.02.2004
(46) Дата публикации: 27.07.2004
(56) Ссылки: RU 2029359 C1, 20.02.1995. RU
2110089 C1, 27.04.1998. US 5392437 A,
21.02.1995. US 5787297 A, 28.07.1998.
(85) Дата перевода заявки РСТ на национальную
фазу: 09.07.2002
(86) Заявка РСТ:
US 00/28213 (11.10.2000)
(87) Публикация РСТ:
WO 01/41529 (14.06.2001)
(98) Адрес для переписки:
129010, Москва, ул. Б. Спасская, 25, стр.3,
ООО "Юридическая фирма Городисский и
Партнеры", пат.пов. Ю.Д.Кузнецову, рег.№ 595

(72) Изобретатель: РОДЖЕРС Дион (US),
ТОЛЛ Брет (US), ВУД Эми (US)
(73) Патентообладатель:
ИНТЕЛ КОРПОРЕЙШН (US)
(74) Патентный поверенный:
Кузнецов Юрий Дмитриевич

(54) СПОСОБ И УСТРОЙСТВО ДЛЯ БЛОКИРОВКИ СИГНАЛА СИНХРОНИЗАЦИИ В МНОГОПОТОЧНОМ ПРОЦЕССОРЕ

(57)
Изобретение относится к многопоточным процессорам, а именно к способу и устройству блокировки сигнала синхронизации в многопоточном процессоре. Технический результат заключается в увеличении использования совместно используемых ресурсов в зависимости от изменения в состоянии потоков, обслуживаемых в многопоточном процессоре. Способ включает в себя осуществление индикации ожидания обработки события для каждого из множества потоков, поддерживаемых в многопоточном

процессоре, установление индикации активного или неактивного состояний для каждого из множества потоков, обнаружение условия блокировки синхронизации, которое может быть индицировано отсутствием ожидающих обработки событий для каждого из множества потоков и неактивного состояния для каждого из множества потоков. Сигнал синхронизации, если он разрешен, затем блокируется для по меньшей мере одного функционального блока в многопоточном процессоре в зависимости от обнаружения условия блокировки синхронизации. 3 с. и 21 з.п. ф-лы, 19 ил.



Фиг. 1

RU 2233470 C2

RU 2233470 C2



(19) **RU** ⁽¹¹⁾ **2 233 470** ⁽¹³⁾ **C2**
(51) Int. Cl.⁷ **G 06 F 15/00, 9/38**

RUSSIAN AGENCY
FOR PATENTS AND TRADEMARKS

(12) **ABSTRACT OF INVENTION**

(21), (22) Application: 2002118211/09, 11.10.2000
(24) Effective date for property rights:
11.10.2000 c1. 1-24
(30) Priority: 09.12.1999 US 09/458,589
(43) Application published: 10.02.2004
(46) Date of publication: 27.07.2004
(85) Commencement of national phase: 09.07.2002
(86) PCT application:
US 00/28213 (11.10.2000)
(87) PCT publication:
WO 01/41529 (14.06.2001)
(98) Mail address:
129010, Moskva, ul. B. Spasskaja, 25, str.3,
OOO "Juridicheskaja firma Gorodisskij i
Partnery", pat.pov. Ju.D.Kuznetsovu, reg.No 595

(72) Inventor: RODZhERS Dion (US),
TOLL Bret (US), VUD Ehmi (US)
(73) Proprietor:
INTEL KORPOREJShN (US)
(74) Representative:
Kuznetsov Jurij Dmitrievich

(54) **METHOD AND DEVICE FOR BLOCKING SYNCHRONIZATION SIGNAL IN MULTITHREADED PROCESSOR**

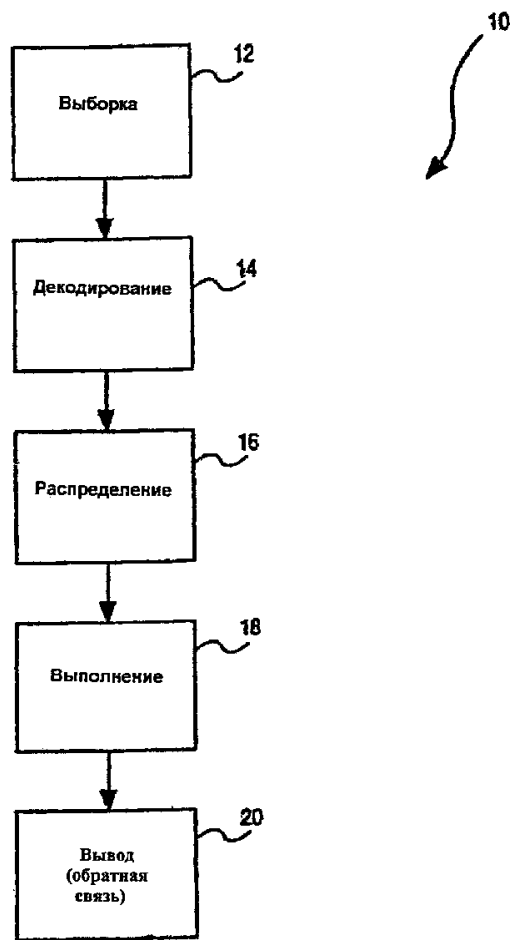
(57) Abstract:
FIELD: multithreaded processors
engineering.

SUBSTANCE: method includes performing of
indication of delay for handling an event
for each of several threads supported in
multithreaded processor, setting of
indication of active or non-active
conditions for each of several threads,
detection of synchronization blocking
condition, which may be indicated by absence

of events awaiting handling and inactive
state for each of several threads.
Synchronization signal, if it is allowed, is
then blocked for at least one functional
block in multithreaded processor dependent
on detection of synchronization blocking
condition.

EFFECT: better use of shared resources in
dependence of changes in state of threads
being maintained in multithreaded processor.

3 cl, 24 dwg



Область техники

Настоящее изобретение в целом относится к области многопоточных процессоров, а более конкретно к способу и устройству блокировки сигнала синхронизации в многопоточном (МП) процессоре.

Предшествующий уровень техники

Многопоточный (МП) проект процессора недавно рассматривался как все более и более привлекательная возможность увеличения производительности процессоров. Многопоточность в процессоре, помимо прочего, обеспечивает возможность более эффективного использования различных ресурсов процессора и особенно более эффективного использования исполнительных схем в процессоре. А именно, подавая множественные потоки к схемам исполнения процессора, тактовые циклы, которые бы иначе не использовались из-за останова или другой задержки обработки конкретного потока, могут использоваться, чтобы обслужить дополнительный поток. Останов в обработке конкретного потока может быть результатом множества событий в конвейере процессора. Например, неудачное обращение в кэш или ошибка предсказания перехода (т.е. операции с длительной задержкой) для команды, включенной в поток, обычно приводят к остановке обработки соответствующего потока. Отрицательное влияние операций с длительной задержкой на эффективность исполнительных схем усилено недавним увеличением производительности исполнительных схем, которые имеют опережающие достижения в уменьшении времени доступа к памяти и среднего времени выборки.

Многопоточные компьютерные приложения также становятся все более и более обычными ввиду поддержки, обеспеченной таким многопоточным приложением множеством популярных операционных систем, таких как операционные системы Windows NT и Unix. Многопоточные компьютерные приложения особенно эффективны в области мультимедиа.

Многопоточные процессоры могут быть в общих чертах классифицированы на две категории (т.е. точные или грубые модели проектов) в зависимости от схемы чередования или переключения потоков, используемой в соответствующем процессоре. Точные многопоточные проекты поддерживают множество активных потоков в процессоре и обычно чередуют два различных потока на основе принципа цикл за циклом. Грубые модели многопоточных проектов обычно чередуют команды различных потоков при возникновении некоторого события с длительной задержкой, такого как неудачное обращение в кэш. Грубые модели многопоточных проектов обсуждаются в книге Eickemayer, R.; Johnson, R.; et al, "Evaluation of Multithreaded Uniprocessors for Commercial Application Environments", The 23rd Annual International Symposium on Computer Architecture, pp.203-212, May 1996. Различия между точными и грубыми моделями проектов дополнительно обсуждаются в книге Laudon, J.; Gupta, A, "Architectural and

Implementation Tradeoffs in the Design of Multiple-Context Processors", Multithreaded Computer Architectures: A Summary of the State of the Art, edited by R.A. Iannuci et al, pp.167-200, Kluwer Academic Publishers, Norwell, Massachusetts, 1994. В работе Laudon дополнительно предлагает схему чередования, которая объединяет переключение по принципу цикл за циклом точной модели проекта с полной взаимной блокировкой конвейеров грубой модели проекта (или блокированную схему). Для этой цели в работе Laudon предлагается команда "возврат", которая делает определенный поток (или контекст) недоступным в течение определенного числа циклов. Такая команда "возврат" может быть выдана после возникновения предопределенных событий, таких как неудачное обращение в кэш. Таким образом Laudon избегает необходимости фактически переключать поток, просто делая один из потоков недоступным.

Многопоточная архитектура для процессора представляет множество дополнительных сложных проблем в контексте архитектуры процессора с исполнением команд с изменением последовательности и спекулятивного исполнения (исполнения по предположению). Более конкретно обработка событий (например, команд перехода, исключений или прерываний), которые могут приводить к неожиданному изменению в потоке команд, усложняется при рассмотрении множества потоков. В процессоре, где осуществляется разделение ресурсов между множеством потоков (т.е. существует ограничение или отсутствие дублирования функциональных блоков для каждого потока, поддерживаемого процессором), обработка возникновения событий, имеющих отношение к определенному потоку, усложняется тем, что при обработке таких событий должны рассматриваться дополнительные потоки.

Когда осуществляется совместное использование ресурсов в многопоточном процессоре, дополнительно желательно попытаться увеличить использование совместно используемых ресурсов в зависимости от изменений в состоянии потоков, обслуживаемых в многопоточном процессоре.

Сущность изобретения

Согласно одному аспекту настоящего изобретения обеспечивается способ, который включает в себя осуществление индикации ожидающего события для каждого из множества потоков, поддерживаемых в многопоточном процессоре. Индикация активного или неактивного состояния выполняется для каждого из множества потоков, поддерживаемых в многопоточном процессоре. Обнаруживается условие блокировки синхронизации, указанное индикацией отсутствия ожидающего события обработки событий, относящихся к каждому из множества потоков, и неактивным состоянием для каждого из потоков. Сигнал синхронизации, если он разрешен, блокируется для по меньшей мере одного функционального блока в многопоточном процессоре в зависимости от обнаружения условия блокировки синхронизации.

Способ может включать в себя обнаружение условия разрешения

синхронизации, на которое указывает индикация ожидающего обработки события, для, по меньшей мере, одного потока из множества потоков, поддерживаемых в многопоточном процессоре, или индикация активного состояния для, по меньшей мере, одного потока из множества потоков, поддерживаемых в многопоточном процессоре, и разрешение сигнала синхронизации, если он заблокирован, для, по меньшей мере, одного функционального блока в многопоточном процессоре в ответ на обнаружение условия разрешения синхронизации.

Способ может дополнительно включать в себя обеспечение индикации операции доступа к шине и обнаружение условия блокировки синхронизации только в отсутствие индикации операции доступа к шине. В некоторых вариантах осуществления операция доступа к шине содержит операцию отслеживания, и индикация операции доступа к шине обеспечивается, когда операция доступа к шине активна в течение predetermined продолжительности отслеживания.

Способ может дополнительно включать в себя поддержание индикации запрещения события для, по меньшей мере, первого потока из множества потоков, поддерживаемых многопоточным процессором, причем индикация запрещения события идентифицирует, по меньшей мере, одно событие для первого потока, которое не является событием, которое вызывает переключение первого потока между неактивным и активным состояниями. В варианте осуществления настоящего изобретения обнаружение условия блокировки синхронизации и условия разрешения синхронизации зависит от индикации запрещения события для первого потока, и условие блокировки синхронизации является обнаруженным или условие разрешения синхронизации является необнаруженным, если ожидающее обработки событие обозначено индикацией запрещения события для первого потока как запрещенное. В некоторых вариантах осуществления поддержание индикации активности потока для каждого из множества потоков содержит поддержание работы конечного автомата активного потока, который выдает сигнал для каждого из множества поддерживаемых потоков для того, чтобы указать, находится ли соответствующий поток в активном или неактивном состоянии.

В примерном варианте осуществления изобретения блокировка сигнала синхронизации для, по меньшей мере, одного функционального блока может включать в себя операции логической обработки сигнала синхронизации, блокировки распространения сигнала синхронизации процессора по магистрали синхронизации процессора, блокировки контура фазовой автоподстройки частоты, который генерирует сигнал синхронизации процессора, или поддержание подачи сигнала синхронизации к входам шины многопоточного процессора для того, чтобы разрешить обнаружение операции или события доступа к шине, когда сигнал синхронизации заблокирован.

В соответствии с другим аспектом настоящего изобретения обеспечивается

устройство, содержащее индикатор ожидающего обработки события, который поддерживает индикацию ожидающего обработки события для каждого из множества потоков, поддерживаемых в многопоточном процессоре, индикатор активного потока, который осуществляет индикацию активного или неактивного состояния для каждого из множества потоков, поддерживаемых в многопоточном процессоре, и логическую схему управления синхронизацией для обнаружения условия блокировки синхронизации, обозначенного индикацией отсутствия ожидающих обработки событий для каждого из множества потоков, и неактивного состояния для каждого из множества потоков, и отключения сигнала синхронизации для, по меньшей мере, одного функционального блока в многопоточном процессоре в ответ на обнаружение условия блокировки синхронизации.

В примерном варианте осуществления изобретения логическая схема управления синхронизацией обнаруживает условие разрешения синхронизации, обозначенное индикацией ожидающего обработки события или активного состояния для, по меньшей мере, одного потока из множества потоков, поддерживаемых в многопоточном процессоре, и разрешает сигнал синхронизации, если он заблокирован, для, по меньшей мере, одного функционального блока в многопоточном процессоре в ответ на обнаружение условия разрешения синхронизации.

В некоторых вариантах осуществления устройство может включать в себя логическую схему слежения за шиной, которая обеспечивает индикацию операции доступа к шине. В этом варианте осуществления логическая схема управления синхронизацией обнаруживает условие блокировки синхронизации только в отсутствие индикации операции доступа к шине от логической схемы слежения за шиной. Операция доступа к шине может включать в себя операцию слежения, причем логическая схема слежения за шиной может обеспечивать индикацию операции доступа к шине, когда операция доступа к шине активна в течение predetermined продолжительности слежения.

В одном из вариантов осуществления настоящего изобретения устройство может включать в себя индикатор запрещения события, поддерживающий индикацию запрещения события для, по меньшей мере, первого потока из множества потоков, поддерживаемых многопоточным процессором. Такая индикация запрещения события идентифицирует, по меньшей мере, одно событие для первого потока, которое не является событием, которое вызывает переход первого потока между неактивным и активным состоянием. Логическая схема управления синхронизацией обнаруживает условие блокировки синхронизации и условие разрешения синхронизации в зависимости от индикации запрещения события для первого потока. Условие блокировки синхронизации обнаруживается или условие разрешения синхронизации не обнаруживается логической схемой управления синхронизацией, если обозначенное ожидающее обработки событие указано как запрещенное индикацией

запрещения события для первого потока.

В варианте осуществления информации устройство обнаруживает условие разрешения синхронизации, обозначенное индикацией активного состояния для, по меньшей мере, одного потока из множества потоков, поддерживаемых в многопоточном процессоре, и разрешает сигнал синхронизации, если он заблокирован, для, по меньшей мере, одного функционального блока в многопоточном процессоре в ответ на обнаружение условия разрешения синхронизации.

В некоторых вариантах осуществления устройство может включать в себя конечный автомат активного потока, который выводит сигнал для каждого из множества поддерживаемых потоков для того, чтобы указать, находится ли соответствующий поток в активном состоянии или неактивном состоянии.

Логическая схема управления синхронизацией может блокировать сигнал синхронизации для, по меньшей мере, одного функционального блока с помощью логической обработки сигнала синхронизации или блокируя распространение сигнала синхронизации процессора по магистрали синхронизации процессора. Логическая схема управления синхронизацией может блокировать сигнал синхронизации для, по меньшей мере, одного функционального блока, блокируя контур фазовой автоподстройки частоты, который генерирует сигнал синхронизации процессора. Она может, в одном из вариантов осуществления, поддерживать подачу сигнала синхронизации к входам шины многопоточного процессора для того, чтобы разрешить обнаружение операции или события доступа к шине, когда сигнал синхронизации заблокирован.

В соответствии с еще одним аспектом настоящего изобретения обеспечивается устройство, содержащее средство для поддержания индикации ожидающего обработки события для каждого из множества потоков, поддерживаемых в многопоточном процессоре, средство для поддержания индикации активного или неактивного состояния для каждого из множества потоков, поддерживаемых в многопоточном процессоре, и средство для обнаружения условия блокировки синхронизации, обозначенного индикацией отсутствия ожидающих обработки событий, для каждого из множества потоков и неактивного состояния для каждого из множества потоков, и для отключения сигнала синхронизации для, по меньшей мере, одного функционального блока в многопоточном процессоре в ответ на обнаружение условия блокировки синхронизации.

Другие признаки настоящего изобретения будут очевидны из сопроводительных чертежей и из подробного описания, которое следует далее.

Краткое описание чертежей

Настоящее изобретение иллюстрировано с помощью примеров и не ограничено фигурами сопроводительных чертежей, в которых соответствующие ссылки указывают на подобные элементы и в которых:

Фиг.1 - структурная схема, показывающая один из вариантов осуществления конвейера процессора с многопоточной поддержкой.

Фиг.2 - структурная схема, показывающая примерный вариант осуществления процессора в форме универсального многопоточного микропроцессора.

Фиг.3 - структурная схема, показывающая выбранные компоненты типичного многопоточного микропроцессора, и конкретно изображает различные функциональные блоки, которые обеспечивают возможность буферизации (или хранения), будучи логически разделенными для того, чтобы адаптировать множество потоков.

Фиг.4 - структурная схема, показывающая кластер обработки с изменением последовательности согласно одному из вариантов осуществления.

Фиг.5 - схематическое представление таблицы псевдонимов регистров и файла регистров, используемых в одном из вариантов осуществления.

Фиг.6A - схема, показывающая подробности, относящиеся к буферу переупорядочивания, который логически разделен для того, чтобы обслуживать множество потоков в многопоточном процессоре согласно одному из вариантов осуществления.

Фиг.6B - схематическое представление регистра ожидающего обработки события и регистра запрещения события согласно одному из вариантов осуществления.

Фиг.7A - последовательность операций, показывающая способ обработки события в многопоточном процессоре согласно одному из вариантов осуществления.

Фиг.7B - последовательность операций, показывающая способ обработки события "виртуальное удаление" в многопоточном процессоре согласно одному из вариантов осуществления.

Фиг.8 - схематическое представление множества примеров событий, которые могут быть обнаружены блоком обнаружения событий, осуществленным в многопоточном процессоре согласно одному из вариантов осуществления.

Фиг.9 и 10 - соответствующие схемы, показывающие примерное содержимое таблицы переупорядочивания в типичном буфере переупорядочивания, таком как показанный на фиг.6A.

Фиг.11A - последовательность операций, показывающая способ выполнения операции очистки (или удаления) в многопоточном процессоре, поддерживающем по меньшей мере первый и второй потоки согласно примеру варианта осуществления.

Фиг.11B - структурная схема, показывающая схему конфигурирования, которая работает таким образом, чтобы сконфигурировать функциональный блок в соответствии с выходным сигналом конечного автомата активного потока согласно одному варианту осуществления.

Фиг.12 - временная диаграмма, показывающая формирование сигнала удаления согласно одному из вариантов осуществления.

Фиг.13 - последовательность операций, показывающая способ обеспечения монопольного доступа к блоку обработки событий в многопоточном процессоре согласно одному из вариантов осуществления.

Фиг.14 - диаграмма состояний, показывающая работу конечного автомата монопольного доступа, осуществленного в многопоточном процессоре согласно одному из вариантов осуществления.

Фиг.15 - диаграмма состояний, показывающая состояния, которые могут быть приняты конечным автоматом активного потока, осуществленным в многопоточном процессоре согласно одному из вариантов осуществления.

Фиг.16A - последовательность операций, показывающая способ выхода активного потока при обнаружении события неактивного состояния (состояния "она") для активного потока в многопоточном процессоре согласно одному из вариантов осуществления.

Фиг.16B - схематическое представление сохранения состояния и обратного распределения регистров после выхода из потока согласно одному из вариантов осуществления.

Фиг.17 - последовательность операций, показывающая способ перехода потока из неактивного в активное состояние после обнаружения события перезагрузки конвейера для неактивного потока согласно одному из вариантов осуществления.

Фиг.18 - последовательность операций, показывающая способ управления разрешением и блокировкой сигнала синхронизации для, по меньшей мере, одного функционального блока в многопоточном процессоре согласно одному из вариантов осуществления.

Фиг.19A - структурная схема, показывающая схему управления синхронизацией для разрешения и блокировки сигнала синхронизации в многопоточном процессоре согласно одному из вариантов осуществления.

Фиг.19B - схематическая диаграмма, показывающая один из вариантов осуществления схемы управления синхронизацией, показанной на фиг.19A.

Подробное описание

Описываются способ и устройство для управления сигналом синхронизации в многопоточном процессоре. В нижеследующем описании сформулированы многочисленные конкретные подробности с целью объяснения и для того, чтобы обеспечить полное понимание настоящего изобретения. Однако специалисту очевидно, что настоящее изобретение может быть осуществлено без этих конкретных подробностей.

Для целей настоящего описания термин "событие" следует понимать таким образом, чтобы он включал в себя любое событие, внутреннее или внешнее для процессора, которое вызывает изменение или прерывание в обслуживании потока команд (макро- или микрокоманд) в процессоре. Соответственно, термин "событие" должен включать в себя обработку команд перехода, исключения и прерывания, которые могут быть сгенерированы в процессоре или вне процессора, но не ограничивался ими.

Для целей настоящего описания термин "процессор" следует понимать таким образом, чтобы он относился к любому устройству, которое способно выполнять последовательности команд (например, макро- или микрокоманд), и чтобы он включал

в себя универсальные микропроцессоры, специальные микропроцессоры, графические контроллеры, звуковые контроллеры, контроллеры мультимедиа, микроконтроллеры или сетевые контроллеры, но не ограничивался ими. Дополнительно термин "процессор" следует использовать по отношению, помимо прочего, к процессорам с полной системой команд (CISC), с сокращенным набором команд (RISC) или с системой команд сверхбольшой разрядности (VLIW).

Дополнительно термин "точка очистки" следует понимать таким образом, чтобы он включал в себя любую команду в потоке команд (который может включать в себя микрокоманду или поток макрокоманд), определение местоположения которой в потоке команд обеспечивается посредством маркера потока или другой команды, в которой событие может быть обработано или выполнено.

Термин "команда" следует понимать таким образом, чтобы он включал в себя макрокоманду или микрокоманду, но не ограничивался ими.

Описываются некоторые примерные варианты осуществления настоящего изобретения, которые осуществлены прежде всего или в аппаратных средствах, или в программном обеспечении. Тем не менее специалисты должны признать, что многие признаки могут легко быть осуществлены в аппаратных средствах, программном обеспечении или комбинации аппаратных средств и программного обеспечения. Программное обеспечение (например, микрокоманды или макрокоманды) для реализации вариантов осуществления изобретения может находиться, полностью или по меньшей мере частично, в оперативной памяти, доступной процессору и/или в процессоре непосредственно (например, в кэше или блоке упорядочивания микропрограммы). Например, блок обработки событий и конечные автоматы могут быть осуществлены в микропрограмме, отправленной блоком упорядочивания микропрограммы.

Программное обеспечение может дополнительно передаваться или приниматься через сетевое интерфейсное устройство.

Для целей настоящего описания термин "машинно-считываемый носитель" следует понимать таким образом, чтобы он включал в себя любой носитель, на котором возможно сохранение или кодирование последовательности команд для исполнения устройством, что приводило бы к выполнению устройством любой команды согласно способам настоящего изобретения. Термин "машинно-считываемый носитель" следует соответственно понимать так, чтобы он включал в себя полупроводниковую память, оптические и магнитные диски и волновой канал передачи, но не ограничивался ими.

Конвейер процессора

Фиг.1 - укрупненная структурная схема, показывающая один из вариантов осуществления конвейера 10 процессора. Конвейер 10 включает в себя множество ступеней конвейера, начинающихся со ступени 12 выборки, на которой команды (например, макрокоманды) выбираются и

подаются на конвейер 10. Например, макрокоманда может быть выбрана из кэш-памяти, которая интегрирована с процессором, или непосредственно соединена с ним, или может быть выбрана из внешней оперативной памяти через шину процессора. От ступени 12 выборки макрокоманды передаются к ступени 14 декодирования, где макрокоманды транслируются в микрокоманды (также называют "микропрограммой"), пригодные для выполнения в процессоре. Затем микрокоманды передаются по "течению" потока к ступени 16 распределения, где ресурсы процессора распределяются различным микрокомандам согласно доступности и потребности. Затем микрокоманды выполняются на ступени 18 выполнения до выполнения завершения или "обратной записи" (например, перехода в архитектурное состояние) на ступени 20 сброса.

Архитектура микропроцессора

Фиг.2 - структурная схема, показывающая примерный вариант осуществления процессора 30 в форме универсального микропроцессора. Описанный ниже процессор 30 является многопоточным (МП) процессором и соответственно способен обрабатывать множественные потоки команд (или контексты). Однако множество описаний, обеспеченных ниже в описании, не являются специфическими для многопоточного процессора и могут найти применение в однопоточном процессоре. В примере варианта осуществления процессор 30 может содержать микропроцессор архитектуры фирмы Intel (IA), который способен выполнять систему команд архитектуры фирмы Intel. Примером такого микропроцессора архитектуры фирмы Intel является микропроцессор Пентиум Про или микропроцессор Пентиум III, изготовленные корпорацией Intel, расположенной в г. Санта-Клара, Калифорния.

В одном из вариантов осуществления процессор 30 содержит входной блок упорядоченной обработки и внутренний блок обработки с изменением последовательности. Входной блок упорядоченной обработки включает в себя блок 32 интерфейсной шины, который функционирует как средство передачи между процессором 30 и другими компонентами (например, оперативной памятью) компьютерной системы, в которой процессор 30 может использоваться. Для этой цели блок 32 интерфейсной шины соединяет процессор 30 с шиной процессора (не показана), через которую данные и информация управления могут приниматься и передаваться в/из процессора 30. Блок 32 интерфейсной шины включает в себя схему 34 входной (системной) шины (ССШ), которая управляет передачей по шине процессора. Блок 32 интерфейсной шины дополнительно включает в себя блок 36 шинной очереди, который обеспечивает функцию буферизации при передаче по шине процессора. Показанный блок 32 интерфейсной шины принимает запросы 38 к шине и посылает результаты слежения или ответы шины к исполнительному блоку 42 памяти, который обеспечивает функцию местной памяти в процессоре 30. Исполнительный блок 42 памяти включает в себя объединенный кэш 44

данных и команд, буфер 46 быстрого преобразования адреса (ББПА) данных и буфер 48 упорядочивания памяти. Исполнительный блок 42 памяти принимает запросы 50 на выборку команды из механизма 54 трансляции и доставляет необработанные команды 52 (т.е. кодированные макрокоманды) к механизму 54 трансляции микрокоманд, который транслирует принятые макрокоманды в соответствующий набор микрокоманд.

Механизм 54 трансляции микрокоманд эффективно работает как "блок обработки промахов" кэша трасс, доставляя микрокоманды кэшу 62 трасс в случае неудачного обращения в кэш трасс. Для этой цели механизм 54 трансляции микрокоманды функционирует таким образом, чтобы обеспечить выполнение ступеней 12 и 14 выборки и декодирования конвейера в случае неудачного обращения в кэш трасс. Показанный механизм 54 трансляции микрокоманды включает в себя блок 100 указания следующей команды (УСК), буфер 102 быстрого преобразования адреса (ББПА) команды, блок 104 прогнозирования перехода, буфер 106 потока команд, предварительный декодер 108 команды, схему 110 направления команды, декодер 112 команды и блок 114 вычисления адреса перехода. Блок 100 указания следующей команды, ББПА 102, блок прогнозирования 104 перехода и буфер 106 потока команды вместе составляют блок 99 предсказания ветвлений (БПВ). Декодер 112 команды и блок 114 вычисления адреса перехода вместе составляют блок 113 трансляции команды (IX).

Блок 100 указания следующей команды выдает запрос на следующую команду к объединенному кэшу 44. В примере варианта осуществления, когда процессор 30 содержит многопоточный микропроцессор, способный к обработке двух потоков, блок 100 указания следующей команды может включать в себя мультиплексор (МПЛ) (не показан), который выбирает между указателями команды, связанными или с первым, или со вторым потоком, для включения в следующий запрос на команду, выданный оттуда. В одном из вариантов осуществления блок 100 указания следующей команды чередует запросы на следующую команду для первого и второго потоков способом цикл-за-циклом ("пинг-понг", попеременно), принимая команды, которые были запрошены, для обоих потоков до тех пор, пока ресурсы буфера 106 потока команд для обоих потоков не будут исчерпаны. Запросы блока указания следующей команды могут быть по 16, 32 или 64 байтов в зависимости от того, находится ли начальный адрес запроса в верхней половине строки, выровненной по границе 32-байт или 64-байт. Блок 100 указания следующей команды может быть переадресован блоком 104 прогнозирования перехода, блоком 114 вычисления адреса перехода или кэшем 62 трасс, причем запрос неудачного обращения в кэш трасс является запросом на переадресацию с самым высоким приоритетом.

Когда блок 100 указания следующей команды осуществляет запрос за командой к объединенному кэшу 44, он генерирует два бита "идентификатора запроса", который

связан с запросом команды и функционирует как “тэг” для соответствующего запроса команды. При возвращении данных, соответствующих запросу команды, объединенный кэш 44 возвращает следующие тэги или идентификаторы вместе с данными:

1. “Идентификатор запроса”, переданный блоком 100 указания следующей команды;

2. Трехбитный “идентификатор порции”, который идентифицирует возвращенную порцию; и

3. “Идентификатор потока”, который идентифицирует поток, которому принадлежат возвращенные данные.

Запросы следующей команды передаются от блока 100 указания следующей команды к ББПА 102 команды, который выполняет операцию поиска адреса и доставляет физический адрес объединенному кэшу 44. Объединенный кэш 44 доставляет соответствующую макрокоманду к буферу 106 потока команд. Каждый следующий запрос команды также передается непосредственно от блока 100 указания следующей команды к буферу 106 потока команд, чтобы позволить буферу 106 потока команд идентифицировать поток, которому принадлежит макрокоманда, принятая из объединенного кэша 44. Затем макрокоманды первого и второго потоков выдаются буфером потока 106 потока команд на предварительный декодер 108 команды, который выполняет множество вычислений длины и операций маркировки байта в соответствии с принятым потоком команд (макрокомандами). А именно, предварительный декодер 108 команды генерирует ряд векторов маркировки байта, которые служат, помимо прочего, для разграничения макрокоманд в потоке команд, переданных к схеме 110 направления команд.

Затем схема 110 направления команд использует векторы маркировки байта, чтобы направлять дискретные макрокоманды на декодер 112 команды с целью декодирования. Макрокоманды также передаются из схемы 110 направления команд на блок 114 вычисления адреса перехода с целью вычисления адреса перехода. Затем микрокоманды доставляются из декодера 112 команды на механизм 60 трассировки доставки.

Во время декодирования маркеры потока связываются с каждой микрокомандой, в которую транслируется макрокоманда. Маркер потока указывает на характеристику связанной микрокоманды и может, например, обозначать соответствующую микрокоманду как первую или последнюю микрокоманду в последовательности микропрограммы, представляющей макрокоманду. Маркеры потока включают в себя маркеры потока “начало макрокоманды” (НМК) и “конец макрокоманды” (КМК). Согласно настоящему изобретению, декодер 112 может дополнительно декодировать микрокоманды, которые имеют маркеры потока совместного использования ресурса (мультипроцессорной системы) (СИРМП) и маркеры потока синхронизации (СИНХ), связанные с ними. А именно, маркер потока совместного использования ресурса идентифицирует микрокоманду как место в конкретном потоке, в котором поток может быть прерван (например, перезапущен или приостановлен)

с меньшим количеством отрицательных последствий, чем в другом месте в потоке. Декодер 112 в примере варианта осуществления настоящего изобретения сконструирован для того, чтобы отметить микрокоманды, которые содержат конец или начало родительской макрокоманды, маркером потока совместного использования ресурса, а также неустойчивые точки в более длинных последовательностях

5 микропрограммы. Маркер потока синхронизации идентифицирует микрокоманду как место в конкретном потоке, в котором поток может быть синхронизирован с другим потоком, в зависимости от, например, команды синхронизации в другом

10 потоке. Для целей настоящего описания термин “синхронизирует” следует понимать таким образом, чтобы он относился к идентификации по меньшей мере первой точки в, по меньшей мере, одном потоке, в которой состояние процессора может изменяться относительно того потока и/или по меньшей мере одного дополнительного потока, с уменьшенным или более низким нарушением работы процессора, по сравнению со второй точкой в том же потоке или в другом потоке.

25 Декодер 112 в примерном варианте осуществления настоящего изобретения сконструирован для того, чтобы отметить микрокоманды, которые расположены в выбранных границах макрокоманды, где состояние, разделенное потоками, сосуществующими в одном и том же процессоре, может быть изменено одним потоком без неблагоприятного влияния на выполнение других потоков.

30 Из механизма 54 трансляции микрокоманды декодированные команды (т.е. микрокоманды) посылаются механизму 60 трассировки доставки. Механизм 60 трассировки доставки включает в себя кэш 62 трасс, блок 64 прогнозирования перехода (БПП) трасс, блок 66 упорядочения микропрограмм и блок 68 очереди микропрограмм (микроопераций). Механизм 60 трассировки доставки функционирует как кэш микрокоманд и является первичным источником микрокоманд для находящегося “ниже по потоку” блока 70 исполнения.

45 Обеспечивая функцию кэширования микрокоманды в конвейере процессора, механизм 60 трассировки доставки, и особенно кэш 62 трасс, позволяет улучшить работу по трансляции, сделанную механизмом 54 трансляции микрокоманды, чтобы обеспечить увеличенную полосу пропускания микрокоманды. В одном типичном варианте осуществления кэш 62 трасс может содержать 256 наборов ассоциированной памяти с 8 путями. Термин “трасса” в настоящем примерном варианте осуществления может относиться к последовательности микрокоманд, сохраненных в записях кэша 62 трасс, причем каждая запись включает в себя указатели на предшествующие и действующие

60 микрокоманды, составляющие трассу. Таким образом, кэш 62 трасс облегчает высокоэффективное упорядочивание тем, что адрес следующей записи, к которой нужно обратиться с целью получения последующей микрокоманды, известен прежде, чем закончится текущий доступ. Трассы, в одном

из вариантов осуществления, могут рассматриваться как "блоки" команд, которые различаются друг от друга заголовками трассы, и завершаются после достижения косвенного перехода или при достижении одного из многих существующих пороговых состояний, таких как число условных переходов, которые могут быть размещены в одной трассе, или максимальное число микрокоманд, которые может содержать трасса.

Блок 64 прогнозирования перехода кэш трассы обеспечивает локальные предсказания ветвлений, имеющие отношение к трассам в кэше 62 трасс. Кэш 62 трасс и блок 66 упорядочения микропрограммы обеспечивают передачу микрокоманд к блоку 68 очереди микропрограммы, откуда микрокоманды затем подаются к кластеру (группе) выполнения с изменением последовательности.

Показанный блок 66 упорядочения микропрограммы, кроме того, включает в себя множество блоков 67 обработки событий, воплощенных в микропрограмме, которые выполняют множество операций в процессоре 30 в ответ на возникновение события, такого как исключение или прерывание. Блоки 67 обработки событий, как будет описано ниже более подробно, вызываются блоком 188 обнаружения событий, который включен в блок 74 переименования регистров во внутренней части процессора 30.

Рассматриваемый процессор 30 имеет входной блок упорядоченной обработки, содержащий блок 32 интерфейса шины, исполнительный блок 42 памяти, механизм 54 трансляции микрокоманды и механизм 60 трассировки доставки, и внутренний блок обработки с изменением последовательности, который будет описан подробно ниже.

Микрокоманды, отправленные из блока 68 очереди микропрограммы, принимаются в кластер 71 обработки с изменением последовательности, содержащий блок 72 планирования, блок 74 переименования регистров, блок 76 распределения, буфер 78 переупорядочивания и блок 80 очереди повторного выполнения. Блок 72 планирования включает в себя набор станций резервирования и работает таким образом, чтобы спланировать и послать микрокоманды для выполнения блоком 70 исполнения. Блок 74 переименования регистров выполняет функцию переименования скрытых регистров целого числа и с плавающей точкой (которые могут использоваться вместо любого из восьми универсальных регистров или любого из восьми регистров с плавающей точкой, когда процессор 30 выполняет систему команд архитектуры фирмы Intel). Блок 76 распределения работает таким образом, чтобы распределить ресурсы блока 70 исполнения и кластера 71 микрокомандам согласно доступности и потребности. Когда недостаточно ресурсов, требуемых для обработки микрокоманды, блок 76 распределения отвечает за установление сигнала 82 останова, который передается через механизм 60 трассировки доставки механизму 54 трансляции микрокоманды, как показано позицией 58. Микрокоманды, которые ранее скорректировали свои поля источника с помощью блока 74 переименования регистров, помещаются в

буфер 78 переупорядочивания в строгом порядке программы. Когда микрокоманды в буфере 78 переупорядочивания закончили выполнение и готовы к выводу, они удаляются из буфера переупорядочивания и извлекаются упорядоченным способом (т.е. согласно первоначальному порядку программы). Блок 80 очереди повторного выполнения передает микрокоманды, которые должны быть заново выполнены, блоку 70 исполнения.

Показанный блок 70 исполнения включает в себя механизм 84 выполнения операций с числами с плавающей точкой, механизм 86 выполнения целочисленных операций и кэш 88 данных 0-го уровня. В одном типовом варианте осуществления, в котором процессор 30 выполняет систему команд архитектуры Intel, механизм 84 выполнения операций с числами с плавающей точкой может дополнительно выполнять команды MMX (мультимедийного расширения) и команды потокового SIMD (один поток команд, множеств потоков данных) расширения (SSE).

Многопоточное выполнение

В примере варианта осуществления процессора 30, показанном на фиг.2, может существовать ограниченное дублирование ресурсов, предназначенных для поддержания многопоточности, и соответственно необходимо в некоторой степени осуществить разделение (совместное использование) ресурсов между потоками. Используемая схема совместного использования ресурсов, следует признать, зависит от числа потоков, которые процессор может одновременно обрабатывать. Поскольку функциональные блоки в процессоре обычно обеспечивают некоторые функциональные возможности буферизации (или хранения) и функциональные возможности распространения, проблема совместного использования ресурсов может рассматриваться как проблема (1) хранения и (2) полосы частот обработки/распространения разделенных компонентов. Например, в процессоре, который поддерживает одновременную обработку двух потоков, буферные ресурсы в различных функциональных блоках могут постоянно или логически разделяться между двумя потоками. Точно так же полоса частот, которая обеспечивается магистралью распространения информации между двумя функциональными блоками, должна быть разделена и распределена между этими двумя потоками. Поскольку эти проблемы разделения ресурсов могут возникать во множестве местоположений в конвейере процессора, различные схемы разделения ресурсов могут использоваться в данных различных местоположениях в соответствии с условиями и характеристикой определенного местоположения. Следует признать, что различные схемы разделения ресурсов могут подходить для различных местоположений ввиду изменяющихся функциональных возможностей и характеристик работы.

Фиг.3 - структурная схема, показывающая выбранные компоненты для одного из вариантов осуществления процессора 30, показанного на фиг.2, и изображающая различные функциональные блоки, которые обеспечивают возможность буферизации, будучи логически разделенными, чтобы

разместить два потока (т.е. поток 0 и поток 1). Логическое разделение средств буферизации (или хранения) функционального блока на два потока может быть достигнуто, распределяя первый предопределенный набор элементов в буферизированном ресурсе для первого потока и распределяя второй предопределенный набор элементов в буферизированном ресурсе для второго потока. Однако в альтернативных вариантах осуществления буферизация также может быть динамически разделенной. А именно, это может быть достигнуто, обеспечивая две пары указателей чтения и записи, причем первая пара указателей чтения и записи связана с первым потоком, а вторая пара указателей чтения и записи связана со вторым потоком. Первый набор указателей чтения и записи может быть ограничен первым предопределенным числом элементов в ресурсе буферизации, в то время как второй набор указателей чтения и записи может быть ограничен вторым предопределенным числом элементов в том же самом ресурсе буферизации. В иллюстрируемом варианте осуществления показаны потоковый буфер 106 команд, кэш 62 трасс и блок 103 очереди команд, причем для каждого обеспечивается емкость памяти, которая логически разделена между первым и вторым потоками.

Кластер (71) обработки с изменением последовательности

Фиг.4 - структурная схема, показывающая дополнительные подробности одного из вариантов осуществления кластера 71 обработки с изменением последовательности. Кластер 71 обеспечивает выполнение функциональных возможностей станции резервирования, переименования регистров, повторного выполнения и завершения в процессоре 30. Кластер 71 принимает микрокоманды от механизма 60 трассировки доставки, распределяет ресурсы этим микрокомандам, переименовывает регистры источника и адресата для каждой микрокоманды, планирует микрокоманды для отправки соответствующим блокам 70 исполнения, обрабатывает микрокоманды, которые повторно выполняются вследствие получения данных по предположению, и затем, наконец, завершает микрокоманды (т.е. передает микрокоманды в постоянное архитектурное состояние).

Микрокоманды, полученные в кластере 71, одновременно доставляются в таблицу 120 псевдонимов регистров и схему 122 управления распределением и списком свободных ресурсов. Таблица 120 псевдонимов регистров отвечает за трансляцию имен логических регистров в адреса физических регистров, используемым блоком 72 планирования и блоками 70 исполнения. Более конкретно, обращаясь к фиг.5, таблица 120 псевдонимов регистров переименовывает регистры целых чисел, с плавающей точкой и сегментов, содержащиеся в файле 124 физических регистров. Показанный файл 124 физических регистров включает в себя 126 физических регистров, которые являются псевдонимами восьми (8) архитектурных регистров. В иллюстрированном варианте осуществления показанная таблица 120 псевдонимов

регистров включает в себя таблицу 126 входной части и таблицу 128 внутренней части для использования соответственно входной и внутренней частями процессора 30. Каждая запись в таблице 120 псевдонимов регистров связана с архитектурным регистром или рассматривается как архитектурный регистр и включает в себя указатель 130, который указывает на местоположение в файле 124 регистров, в котором хранятся данные, принадлежащие соответствующему архитектурному регистру. Таким образом решаются сложные проблемы, обеспеченные унаследованной архитектурой микропроцессора, которая определяет относительно небольшое количество архитектурных регистров, которые могут быть адресованы.

Схема 122 управления распределением и списком свободных ресурсов отвечает за распределение ресурсов и восстановление состояния в кластере 71. Схема 122 распределяет следующие ресурсы для каждой микрокоманды:

1. Порядковый номер, который дается каждой микрокоманде, чтобы отследить ее логический порядок в потоке, когда микрокоманда обрабатывается в кластере 71. Порядковый номер, приписанный каждой микрокоманде, сохраняется вместе с информацией состояния для микрокоманды в таблице 180 (показана далее на фиг.10) в буфере 162 переупорядочивания.

2. Запись в блоке управления списком свободных ресурсов, которая дается каждой микрокоманде, чтобы позволить проследить и восстановить предысторию микрокоманды в случае операции восстановления состояния.

3. Запись в буфере переупорядочивания (БП), которая индексируется порядковым номером.

4. Запись в файле 124 физических регистров (известном как "набор"), в которой микрокоманда может сохранять полезные результаты.

5. Запись в буфере загрузки (не показан).

6. Запись в буфере останова (не показан).

7. Запись в блоке очереди команд (например, в блоке очереди команд обращения к памяти или в блоке очереди адреса общих команд, как будет описано ниже).

В случае, если схема 122 не способна получить необходимые ресурсы для принятой последовательности микрокоманд, схема 122 запросит, чтобы механизм 60 трассировки доставки остановил доставку микрокоманд, пока достаточное количество ресурсов не будут доступно. Этот запрос передается установлением сигнала 82 останова, показанного на фиг.2.

Что касается назначения микрокоманде записи в файле 124 регистров, фиг.5 показывает массив 132 неиспользуемых регистров, который поддерживает список записей в файле 124 регистров, которые не были распределены архитектурным регистром (т.е. для которых они не являются указателями в таблице 120 псевдонимов регистров). Схема 122 обращается к массиву 132 неиспользуемых регистров для того, чтобы идентифицировать записи в файле 124 регистров, которые доступны для распределения принятой микрокоманде. Схема 122 также отвечает за повторную

выдачу запросов на записи в файле 124 регистров, которые стали доступными.

Схема 122 дополнительно поддерживает блок 134 управления списком свободных ресурсов (БУССР), который позволяет отслеживать архитектурные регистры. А именно, блок 134 управления списком свободных ресурсов поддерживает хронологию изменений таблицы 120 псевдонимов регистров, когда в нее распределяются микрокоманды. Блок 134 управления списком свободных ресурсов обеспечивает возможность "вернуть в исходное состояние" таблицу 120 псевдонимов регистров для того, чтобы указать на неспекулятивное (выполненное не по предположению) состояние для данной ошибки предсказания или данного события. Блок 134 управления списком свободных ресурсов также "определяет возраст" хранения данных в записях файла 124 регистров для гарантии того, что вся информация о состоянии является текущей. Наконец, при завершении идентификаторы физических регистров передаются от блока 134 управления списком свободных ресурсов массиву 132 неиспользуемых регистров для распределения дальнейшей микрокоманды.

Блок 136 очереди команд доставляет микрокоманды блоку 138 планирования и "табло" (БПТ) в последовательном порядке программы, содержит и посылает информацию микрокоманды, необходимую блокам 70 исполнения. Блок 136 очереди команд может включать в себя две разные структуры, а именно схему 140 очереди команд (СОК) и схему очереди адреса команд (СОАК) 142. Схемы 142 очереди адреса команд являются маленькими структурами, предназначенными для передачи критической информации (например, об источниках микрокоманды, адресатах и продолжительности) на блок 138, когда необходимо. Схема 142 очереди адреса команд может, кроме того, содержать схему очереди адреса команд обращения к памяти (СОАКП), которая устанавливает очередность информации для операций с памятью, и схему очереди адреса общих команд (СОАОК), которая устанавливает очередность информации для операций без обращения к памяти. Схема 140 очереди команд хранит менее критичную информацию, такую как код операции и непосредственные данные для микрокоманд. Микрокоманды перемещаются из блока 136 очереди команд, когда соответствующие микрокоманды считываются и записываются в блок 138 планирования и "табло".

Блок 138 планирования и "табло" отвечает за планирование (диспетчеризацию) микрокоманд для исполнения с помощью определения времени, за которое каждый из источников микрокоманд может быть готов, и когда соответствующий блок исполнения доступен для выполнения работы. Показанный на фиг.4 блок 138 содержит блок 144 "табло" файла регистров, блок 146 планирования операций обращения к памяти, блок 148 планирования матричных операций, блок 150 планирования медленных микрокоманд и блок 152 планирования операций с плавающей точкой.

Блок 138 определяет, когда исходный регистр готов, исследуя информацию,

имеющуюся в блоке 144 "табло" файла регистров. Для этой цели в одном из вариантов осуществления блок 144 "табло" файла регистров имеет 256 битов, соответствующих каждому регистру в файле 124 регистров, которые отслеживают данные о готовности ресурсов. Например, биты "табло" для конкретной записи файла 124 регистров могут быть очищены при распределении данных соответствующей записи или при операции записи в блоке 138.

Блок 146 планирования операций обращения к памяти буферизирует микрокоманды класса обращения к памяти, проверяет готовность ресурса и затем планирует микрокоманды класса обращения к памяти. Блок 148 планирования матричных операций содержит два тесно связанных блока планирования арифметико-логических устройств (АЛУ), которые позволяют планировать взаимозависимые микрокоманды. Блок 152 планирования операций с плавающей точкой буферизирует и планирует микрокоманды с плавающей точкой, в то время как блок 150 планирования медленных микрокоманд планирует микрокоманды, не обработанные вышеупомянутыми блоками планирования.

Показанный блок 160 проверки, повторного выполнения и завершения (БППВЗ) включает в себя буфер 162 переупорядочивания, блок 164 проверки, блок 166 организации очереди и схему 168 управления выводом. Блок 160 имеет три основные функции, а именно функцию проверки, функцию повторного выполнения и функцию завершения. А именно, функции блока проверки и повторного выполнения заключаются в повторном выполнении микрокоманд, которые неправильно выполнились. Функция вывода заключается в передаче архитектурного последовательного состояния процессору 30. Более конкретно, блок 164 проверки работает таким образом, чтобы гарантировать, что каждая микрокоманда должным образом выполнила правильные данные. В случае, когда микрокоманда не выполнялась с правильными данными (например, из-за ошибки предсказания перехода), тогда соответствующая микрокоманда повторно выполняется с правильными данными.

Буфер 162 переупорядочивания отвечает за передачу архитектурного состояния процессору 30 с помощью вывода микрокоманд по порядку программы. Указатель 182 вывода, сгенерированный схемой 168 управления выводом, указывает на запись в буфере 162 переупорядочивания, которая выводится. Когда указатель 182 вывода "проходит" микрокоманду записи, соответствующая запись блока 134 управления списком свободных ресурсов освобождается, и соответствующая запись файла регистров может теперь быть повторно востребована и передается массиву 132 неиспользуемых регистров. Показанная схема 168 управления выводом также воплощает конечный автомат 171 активного потока, задача и функционирование которого будет объясняться ниже. Схема 168 управления выводом управляет передачей спекулятивных (выполненных по предположению) результатов, содержащихся в буфере 162 переупорядочивания, соответствующему

архитектурному состоянию в файле 124 регистров.

Буфер 162 переупорядочивания также отвечает за обработку внутренних и внешних событий, как будет описано более подробно ниже. После обнаружения буфером 162 переупорядочивания возникновения события выдается сигнал 170 "удаление". Сигнал 170 удаления вызывает подавление всех микрокоманд на конвейере процессора, которые в настоящее время передаются. Буфер 162 переупорядочивания также обеспечивает механизм 60 трассировки доставки адреса, с которого необходимо начать упорядочивать микрокоманды, чтобы обслужить событие (т.е. с которого необходимо вызвать блок 67 обработки событий, воплощенный в микропрограмме).

Буфер (162) переупорядочивания
Фиг.6А - структурная схема, показывающая дополнительные подробности примера варианта осуществления буфера 162 переупорядочивания, который логически разделен, чтобы обслуживать множественные потоки в многопоточном процессоре 30. А именно, показанный буфер 162 переупорядочивания включает в себя таблицу 180 переупорядочивания, которая может быть логически разделена, чтобы разместить записи для первого и второго потоков, когда процессор 30 работает в многопоточном режиме. При работе в режиме одного потока вся таблица 180 может использоваться для обслуживания одного потока. В одном из вариантов осуществления таблица 180 содержит объединенную структуру памяти, куда при работе в многопоточном режиме обращаются два (2) указателя 182 и 183 вывода, которые ограничены предопределенными и отличающимися наборами записей в таблице 180. Точно так же при работе в режиме одного потока в таблицу 180 обращается один указатель 182 вывода.

Таблица 180 содержит запись, соответствующую каждой записи файла 124 регистров, и сохраняет порядковый номер и информацию состояния в форме информации отказа (сбоя), логического адреса назначения и бита разрешения для каждой записи в данных микрокоманды файла 124 регистров. Каждая запись в таблице 180 индексирована порядковым номером, который является уникальным идентификатором для каждой микрокоманды. Записи в таблице 180 в соответствии с порядковыми номерами распределяются и освобождаются последовательным и упорядоченным способом. В дополнение к другим маркерам потока показанная таблица 180, кроме того, содержит маркер 184 потока разделенных ресурсов и маркер 186 потока синхронизации для каждой микрокоманды.

Буфер 162 переупорядочивания включает в себя блок 188 обнаружения событий, который присоединен таким образом, чтобы принимать запросы на прерывания в форме векторов прерывания, а также обращаться к записям таблицы 180, на которые указывают указатели 182 и 183 вывода. Кроме того, показанный блок 188 обнаружения событий выводит сигнал 170 удаления и сигнал 172 очистки.

Принимая, что для конкретной микрокоманды конкретного потока (например,

потока 0) не происходит ошибки предсказания перехода, исключения или прерывания, информация, сохраненная в записи таблицы 180 для конкретной команды, будет выведена в архитектурное состояние, когда указатель 182 или 183 вывода увеличивается, чтобы адресовать соответствующую запись. В этом случае блок 190 вычисления указателя команды, который формирует часть схемы 168 управления выводом, увеличивает указатель макро- или микрокоманды, чтобы он указывал на (1) адрес перехода, указанный в соответствующей записи файла 124 регистров, или на (2) следующую макро- или микрокоманду, если переход не произошел.

Если произошла ошибка предсказания перехода, то информация передается через область информации отказа на схему 168 управления выводом и блок 188 обнаружения событий. Ввиду того, что ошибка предсказания перехода обозначена с помощью информации отказа, процессор 30, возможно, выбрал по меньшей мере некоторые неправильные команды, которые прошли сквозь конвейер процессора. Поскольку записи таблицы 180 расположены в последовательном порядке, то все записи, расположенные после микрокоманды с ошибкой предсказания перехода, являющиеся микрокомандами, на которые повлияла ошибка предсказания перехода потока команд. В ответ на предпринятый вывод микрокоманды, для которой зарегистрирована ошибка предсказания перехода в информации отказа, блок 188 обнаружения событий устанавливает сигнал 172 очистки, который очищает все состояния всего внутреннего блока обработки с изменением последовательности процессора, и соответственно сбрасывает все состояния всего внутреннего блока обработки с изменением последовательности процессора, являющиеся результатом команд, следующих после микрокоманды с ошибкой предсказания. Выдача сигнала 172 очистки также блокирует выдачу последовательно выбранных микрокоманд, которые могут быть расположены во входном блоке упорядоченной обработки процессора 30.

В схеме 168 управления завершением после уведомления об ошибке предсказания перехода через информацию отказа для завершаемой микрокоманды блок 190 вычисления УК (указателя команды) проверяет, чтобы указатели 179 и/или 181 команды были модифицированы, чтобы представить правильное значение указателя команды. Основываясь на том, произошел переход или нет, блок 190 вычисления УК обновляет указатели 179 и/или 181 команды результирующими данными из записи файла регистров, которая соответствует соответствующей записи таблицы 180, или увеличивает указатели 179 и 181 команды, когда переход не произошел.

Блок 188 обнаружения событий также включает в себя множество регистров 200 для поддержания информации о событиях, обнаруженных для каждого из множества потоков. Регистры 200 включают в себя регистр 202 информации о событии, регистр 204 ожидания обработки события, регистр 206 запрещения события и регистр 208 возврата в исходное состояние и регистр 210 состояния входных линий. Каждый из регистров 202-210

способен хранить информацию, относящуюся к событию, сгенерированному для определенного потока. Соответственно, информация о событии для множества потоков может сохраняться с помощью регистров 200.

Фиг.6В - схематическая иллюстрация примера регистра 204 ожидания обработки события и примера регистра 206 запрещения события для первого потока (например, Т0).

Регистры 204 и 206 ожидания обработки события и запрещения события предусмотрены для каждого потока, поддерживаемого в многопоточном процессоре 30. Отдельные регистры 204 и 206 могут быть предусмотрены для каждого потока, или, альтернативно, один физический регистр может быть логически разделен, чтобы поддержать множество потоков.

Примерный регистр 204 ожидания обработки события содержит бит или другой элемент данных для каждого типа событий, которые регистрируются блоком обнаружения событий (например, событий, описанных ниже относительно фиг.8). Эти события могут быть внутренними событиями, которые генерируются внутри процессора 30, или внешними событиями, сгенерированными вне процессора 30 (например, события на входных линиях, которые принимаются от шины процессора). Регистр 204 ожидания обработки события для каждого потока в иллюстрированном варианте осуществления не включает в себя бит для события обратной записи, поскольку такие события не соответствуют определенному потоку, и соответственно не "ставятся в очередь" в регистре ожидания обработки события. Для этой цели блок 188 обнаружения событий может включать в себя схему 205 обнаружения обратной записи, которая выдает сигнал обратной записи при обнаружении события обратной записи. Биты в регистре 204 ожидания обработки события для каждого потока устанавливаются блоком 188 обнаружения событий, который запускает триггер, который устанавливает соответствующий бит в регистре 204 ожидания обработки события. В примере варианта осуществления установленный бит в регистре 204 ожидания обработки события, связанный с предопределенным событием, обеспечивает индикацию того, что событие соответствующего типа находится в ожидании обработки, как будет описано ниже.

Регистр 206 запрещения события для каждого потока точно также содержит бит или другую структуру данных для каждого типа событий, которые распознаются блоком 188 обнаружения событий, причем этот бит устанавливается или сбрасывается (т.е. очищается) для того, чтобы регистрировать события, являющиеся событиями перезагрузки конвейера для определенного потока. Соответствующие биты в регистре 206 запрещения события устанавливаются операцией записи регистра управления, которая использует специальную микрокоманду, которая изменяет непереименованное состояние в процессоре 30. Бит в регистре 206 запрещения события может также быть сброшен (или очищен), используя операцию записи регистра управления.

Примерный процессор может также иметь

некоторые режимы, в которых биты в регистре 206 запрещения события могут быть установлены для того, чтобы запретить выбранные события в соответствующих режимах.

Биты для определенного типа событий, установленные в каждом из регистров 204 и 206 ожидания обработки события и запрещения события для определенного потока, выводятся на логическую схему 209 И, которая, в свою очередь, выдает сигнал 211 обнаружения события для каждого типа событий, когда содержимое регистров 204 и 206 указывает, что соответствующий тип событий находится в ожидании обработки и не запрещен. Например, когда событие определенного типа не запрещено, при регистрации события в регистре 204 ожидания обработки события, о событии будет немедленно сообщено обнаружением выдачи сигнала 211 обнаружения события для соответствующего типа событий. С другой стороны, если данный тип событий будет запрещен содержимым регистра 206 запрещения события, то возникновение события будет зарегистрировано в регистре 204 ожидания обработки события, но сигнал 211 обнаружения события будет выдан только в том случае, если соответствующий бит в регистре 206 запрещения события будет очищен в то время, когда событие все еще зарегистрировано как ожидающее обработки в регистре 204. Таким образом, событие может быть зарегистрировано в регистре 204 ожидания обработки события, но сигнал 211 обнаружения события для соответствующего возникновения события может сообщаться только в более позднее время, когда запрещение события для определенного потока будет снято.

Сигналы 211 обнаружения события для каждого типа событий для каждого потока подаются к схемам обработки события (схемам выбора и установки приоритета события) и схемам управления синхронизацией, как будет дополнительно описано ниже.

Блок обработки событий для определенного события отвечает за очистку соответствующего бита в регистре 204 ожидания обработки события для определенного потока, как только обработка события была закончена. В альтернативном варианте осуществления регистр ожидания обработки события может быть очищен аппаратными средствами.

Возникновение событий и обработка событий в среде многопоточного процессора

События в многопоточном процессоре 30 могут быть обнаружены и сообщены разнообразными источниками. Например, входной блок упорядоченной обработки процессора 30 может сообщить о событии, и блоки 70 исполнения могут аналогично сообщить о событии. События могут содержать прерывания и исключения. Прерывания-события, которые сгенерированы вне процессора 30 и могут быть инициализированы из устройства на процессор 30 через общую шину (не показана). Прерывания могут вызвать поток управления, направленный на блок 67 обработки событий микропрограммы. Исключения могут быть свободно классифицированы как отказы, ловушки и

помощь, наряду с другими. Исключения-события, которые типично генерируются в процессоре 30.

События передаются непосредственно на блок 188 обнаружения событий в буфере 162 переупорядочивания, в соответствии с которым блок 188 обнаружения событий выполняет множество операций, имеющих отношение к потоку, для которого или в отношении которого событие было сгенерировано. Укрупнено, блок 188 обнаружения событий в ответ на обнаружение события приостанавливает вывод микрокоманд потока, записывает соответствующую информацию отказа в таблицу 180, выдает сигнал 170 удаления, вызывает блок 67 обработки событий, чтобы обработать событие, определяет адрес рестарта и затем повторно выполняет выборку микрокоманд. События могут быть переданы непосредственно на блок 188 обнаружения событий в форме запроса на прерывание (или программы прерывания) или через информацию отказа, зарегистрированную в таблице 180 переупорядочивания для команды первого или второго потока, которая завершается.

Установление сигнала 170 удаления имеет эффект очистки состояний входного блока упорядоченной обработки и внутреннего блока обработки с изменением последовательности многопоточного процессора 30. А именно, в многочисленных функциональных блоках, но не обязательно во всех, очищаются состояния и микрокоманды в ответ на установление сигнала 170 удаления. Некоторые части буфера 48 упорядочивания памяти и блока 32 интерфейса шины не очищаются (например, вывезенные, но не переданные в память результаты слежения за шиной и т.д.). Установление сигнала 170 удаления дополнительно останавливает выборку команды входным блоком и также останавливает упорядочивание микрокоманд в блоке 68 очереди микропрограммы. В то время как эта операция может быть выполнена безнаказанно в однопоточной мультипроцессорной системе или мультипроцессорной системе, обрабатывающей единственный поток, там, где множество потоков существуют и обрабатываются в многопоточном процессоре 30, присутствие других потоков не может игнорироваться при адресации возникновения события, относящегося к одному потоку. Соответственно, настоящее изобретение предлагает способ и устройство для обработки события в многопоточном процессоре, который осведомлен об обработке и присутствии множества потоков в многопоточном процессоре 30, когда происходит событие для одного потока.

Фиг.7А - последовательность операций, иллюстрирующая способ 220 обработки возникновения события в многопоточном процессоре 30 согласно примерному варианту осуществления настоящего изобретения. Способ 220 начинается на этапе 222 с обнаружения блоком 188 обнаружения событий первого события для первого потока. Фиг.8 - схематическое представление множества примеров событий 224, которые могут быть обнаружены блоком 188 обнаружения событий на этапе 222. События,

представленные на фиг.8, были свободно сгруппированы согласно характеристикам ответов на события 224. Первая группа событий включает в себя событие 226 СБРОС и событие 228 МАШИННАЯ ОШИБКА, о которых блок 188 обнаружения событий сообщает множеству потоков в многопоточном процессоре 30 способом, описанным ниже, немедленно после обнаружения, и вынуждает направлять все потоки в один и тот же блок 67 обработки событий в одно и то же время. Вторая группа событий включает в себя событие 230 ОТКАЗ, событие 232 ПОМОЩЬ, событие 234 ДВОЙНОЙ ОТКАЗ, событие 236 ОТКЛЮЧЕНИЕ и событие 238 СМК (самомодифицирующийся код), каждое из которых сообщает о выводе микрокоманды из определенного потока, который передал сигнал об этом событии. А именно, блок 188 обнаружения событий обнаружит событие второй группы при выводе микрокоманды, для которой информация отказа указывает на условие отказа. Об обнаружении события второй группы блок 188 обнаружения событий сообщает только потоку, для которого соответствующее событие было сгенерировано.

Третья группа событий включает в себя событие 240 INIT (короткая инициализация (КИ)), событие 242 INTR (локальное прерывание (ЛП)), событие 244 NMI (немаскируемое прерывание (НМП)), событие 246 КОНТРОЛЬНАЯ ТОЧКА ДАННЫХ (КТД), событие 248 СООБЩЕНИЕ ТРАССИРОВКИ (СТ) и событие 250 A20M (переполнение адреса). События третьей группы сообщают о выводе микрокоманды, имеющей маркер принятия прерывания или принятия потока ловушки. Об обнаружении события третьей группы блок 188 обнаружения событий сообщает только потоку, для которого соответствующее событие было сгенерировано.

Четвертая группа событий включает в себя событие 250 SMI (прерывание управления системой (ПУС)), событие 252 ОСТАНОВ СИНХРОНИЗАЦИИ и событие 254 PREQ (запрос образца (30)). События четвертой группы сообщаются всем потокам, существующим в многопоточном процессоре 30, и они сообщаются, когда любой из множества потоков выдает микрокоманду, имеющую соответствующий маркер потока прерывания. Никакой синхронизации не осуществляется между множественными потоками в ответ на любое событие четвертой группы.

Пятая группа событий, согласно примеру варианта осуществления, является специфической для многопоточной архитектуры процессора и осуществляется в описанном варианте осуществления для того, чтобы обратиться к ряду соображений, которые являются специфическими для многопоточной среды процессора. Пятая группа событий включает в себя событие 260 ВИРТУАЛЬНОЕ УДАЛЕНИЕ, событие 262 СИНХРОНИЗАЦИЯ и событие 264 СОН.

Событие 260 ВИРТУАЛЬНОЕ УДАЛЕНИЕ - событие, которое регистрируется для второго потока, когда (1) первый поток в многопоточном процессоре 30 имеет ожидающее обработки событие (например, любое из описанных выше событий является

ожидающим), (2) второй поток не имеет никаких ожидающих обработки событий (других, чем событие 260), и (3) микрокоманда, имеющая или маркер 184 потока совместно используемого ресурса, или маркер 186 потока синхронизации, выдается буфером 162 переупорядочивания. Событие 260 ВИРТУАЛЬНОЕ УДАЛЕНИЕ имеет эффект вызова блока обработки события виртуального удаления, который повторно запускает выполнение второго потока на микрокоманде, последующей за выданной микрокомандой, имеющей маркеры 184 или 186 потока.

Событие 262 СИНХРОНИЗАЦИИ сообщается микропрограммой, когда конкретный поток (например, первый поток) требуется, чтобы изменить совместно используемое состояние или ресурс в многопоточном процессоре 30. Для этой цели блок 66 упорядочения микропрограммы вставляет микрокоманду синхронизации в поток для первого потока и, с целью избежать ситуации взаимоблокировки, отмечает "микрокоманду синхронизации" маркером 184 потока совместно используемого ресурса и маркером 186 потока синхронизации. Событие 262 СИНХРОНИЗАЦИИ обнаруживается (или регистрируется) только при выдаче микрокоманды синхронизации для первого потока и при выдаче микрокоманды для второго потока, которые имеют маркер 186 потока синхронизации, связанный с ними. Событие 262 СИНХРОНИЗАЦИИ имеет эффект вызова блока обработки события синхронизации, который повторно запускает выполнение первого потока по указателю команды, сохраненному во временном регистре микропрограммы. Дополнительные подробности относительно обработки события 262 СИНХРОНИЗАЦИИ представлены ниже. Второй поток исполняет виртуальное УДАЛЕНИЕ 260.

Событие 264 СОН - событие, которое вызывает переход соответствующего потока из активного состояния в неактивное состояние (или состояние "сна"). Неактивный поток может затем снова быть переведен из неактивного в активное состояние соответствующим событием ПЕРЕЗАГРУЗКА КОНВЕЙЕРА. Характер события ПЕРЕЗАГРУЗКА КОНВЕЙЕРА, которое переводит поток обратно в активное состояние, зависит от события 264 СОН, которое переводит поток в неактивное состояние. Вход потоков в активное состояние и выход из него подробно описан ниже.

Фиг.9 - схема, показывающая пример содержимого таблицы 180 переупорядочивания в буфере 162 переупорядочивания, который будет описан ниже с целью объяснения обнаружения событий и точки очистки (также названа "точкой удаления") в примере варианта осуществления настоящего изобретения. Обнаружение любого из вышеупомянутых событий блоком 188 обнаружения событий на этапе 222 может происходить в ответ на событие 266, переданное на блок 188 обнаружения событий из внутреннего источника в многопоточном процессоре 30 или из внешнего по отношению к процессору 30 источника. Примером такой передачи события 266 может быть вектор прерывания.

Альтернативно, о возникновении события может быть сообщено на блок 188 обнаружения событий с помощью информации 268 отказа для микрокоманды конкретного потока (например, потока 1), который выдается и который соответственно идентифицирован указателем 182 выдачи. Следует отметить, что для внешних событий существует один (1) сигнал в потоке (например, сигналы 266 и 267 соответственно). Для внутренних событий запись в буфере 162 переупорядочивания, содержащем поток, обуславливает поток, которому принадлежит отказ, с помощью его позиции (например, T0 против T1). После обнаружения события блок 188 обнаружения событий сохраняет информацию о событии (например, тип события, источник события и т.д.) для конкретного события в регистре 202 информации о событии и, кроме того, регистрирует ожидающее обработки событие для соответствующего потока в регистре 204 ожидающего обработки события. Как описано выше, регистрация ожидающего обработки события в регистре 204 ожидающего обработки события для соответствующего потока содержит установку бита, связанного с определенным событием, в регистре 204. Следует дополнительно отметить, что событие может быть эффективно обнаружено с помощью установления соответствующего сигнала 211 обнаружения события, если событие не запрещено с помощью установления бита в регистре 206 запрещения события для соответствующего потока, и в некоторых случаях микрокоманда включает в себя соответствующий маркер потока.

Возвращаясь теперь к последовательности операций, показанной на фиг.7А, после обнаружения первого события для первого потока на этапе 222 блок 188 обнаружения событий останавливает вывод первого потока на этапе 270 и устанавливает сигнал 169 "предварительное удаление". Сигнал 169 предварительного удаления устанавливается для того, чтобы избежать ситуации взаимной блокировки, в которой первый поток доминирует над конвейером обработки команды при исключении второго потока. А именно, если второй поток будет исключаться от доступа к конвейеру команды, условия для второго потока, которые требуются, чтобы начать многопоточную операцию удаления, не могут произойти. Сигнал 169 предварительного удаления соответственно передается к входному блоку процессора, а конкретно к исполнительному блоку 42 памяти, для того, чтобы подавить работу конвейера процессора для микрокоманд, составляющих первый поток, для которого событие было обнаружено. Подавление работы конвейера процессора может, просто для примера, быть выполнено, отключая предварительную выборку команды и операцию самомодифицирующегося кода (СМК), которые выполняются исполнительным блоком 42 памяти или другими компонентами входного блока процессора. Подводя итог, останавливая вывод микрокоманд первого потока и/или останавливая или существенно сокращая подачу микрокоманд первого потока на конвейер процессора, второму потоку отдают предпочтение в процессоре и вероятность

возникновения ситуации взаимной блокировки уменьшается.

На этапе 272 принятия решения выполняется определение того, является ли второй поток активным в многопоточном процессоре 30, и соответственно выводится с помощью буфера 162 переупорядочивания. Если нет активного второго потока, способ 220 переходит непосредственно на этап 274, где выполняется первый вид операции очистки, названный "операция удаления". Определение того, является ли конкретный поток активным или неактивным, может быть выполнено с помощью обращения к конечному автомату 171 активного потока, который поддерживается схемой 168 управления выводом. Операция удаления начинается с установления сигнала 170 удаления, который имеет эффект очистки состояний и входного блока упорядоченной обработки, и внутреннего блока обработки с изменением последовательности многопоточного процессора 30, как описано выше. Поскольку только первый поток активен, нет необходимости рассматривать влияние операции удаления на любые другие потоки, которые могут присутствовать и существовать в многопоточном процессоре 30.

С другой стороны, если определено, что второй поток является активным в многопоточном процессоре 30 на этапе 272 принятия решения, способ 220 переходит на выполнение ряда операций, в которых заключается обнаружение точки очистки (или точки удаления) для второго потока, в которой операция удаления может быть выполнена с меньшими отрицательными последствиями для второго потока. Операция удаления, которая выполняется после обнаружения точки очистки, является той же самой операцией, которая выполняется на этапе 274, и соответственно очищает состояния многопоточного процессора 30 (т.е. состояния и для первого, и для второго потоков). Очистка состояния включает в себя микрокоманду операции "дренаж", описанную в другом месте описания. В примере варианта осуществления, раскрытом в настоящей заявке, операция удаления, выполняющаяся после обнаружения точки очистки, не делает различий между состоянием, установленным для первого потока, и состоянием, установленным для второго потока в многопоточном процессоре 30. В альтернативном варианте осуществления операция удаления, выполняющаяся при обнаружении точки очистки, может очищать состояние только для одного потока (т.е. потока, для которого событие было обнаружено), однако, когда существует значительное совместное использование ресурсов в многопоточном процессоре 30 и когда такие совместно используемые ресурсы динамически распределяются и перераспределяются, чтобы обслуживать множественные потоки, очистка состояния для одного потока особенно сложна. Однако этот альтернативный вариант осуществления может потребовать очень сложных аппаратных средств.

После положительного определения на этапе 272 принятия решения дальнейшее определение происходит на этапе 278 принятия решения относительно того,

встретилось ли во втором потоке событие. Такое событие может содержать любое из событий, обсужденных выше, кроме события 260 ВИРТУАЛЬНОЕ УДАЛЕНИЕ. Это определение снова происходит на этапе 188 обнаружения события в зависимости от сигнала 266 события или сигнала 269 информации отказа для второго потока. Информация относительно любого события, с которым сталкивается второй поток, сохраняется в части регистра 202 информации о событии, выделенной второму потоку, и возникновение события регистрируется в регистре 204 ожидающего обработки события.

Если второй поток независимо столкнулся с событием, то способ переходит непосредственно на этап 280, где выполняется многопоточная операция удаления для того, чтобы очистить состояния многопоточного процессора 30. Альтернативно, если второй поток не столкнулся с событием, то выполняется определение на этапе 282 принятия решения того, требует ли первое событие, с которым столкнулся первый поток, чтобы совместно используемое состояние или совместно используемые ресурсы были изменены для обработки первого события. Например, когда первое событие содержит событие 262 СИНХРОНИЗАЦИЯ, как описано выше, это указывает на то, что первый поток требует доступа к ресурсу совместно используемого состояния. Событие 262 СИНХРОНИЗАЦИЯ может быть идентифицировано с помощью вывода микрокоманды синхронизации для первого потока, который имел и совместно используемые ресурсы, и маркеры 184 и 186 потока синхронизации, связанные с ними. Фиг.10 - схема, подобная показанной на фиг.9, которая показывает типичное содержимое таблицы 180 переупорядочивания. Показанная часть таблицы 180, распределенная первому потоку (т.е. потоку 0), включает в себя микрокоманду синхронизации, на которую указывает указатель 182 вывода. Кроме того, показанная микрокоманда синхронизации имеет маркер потока 184 совместно используемых ресурсов и маркер 186 потока синхронизации, связанные с ней. Вывод указанной микрокоманды синхронизации будет зарегистрирован блоком 188 обнаружения событий как возникновение события 262 СИНХРОНИЗАЦИЯ.

Если определено, что первое событие для первого потока (например, потока 0) не изменяет совместно используемое состояние или ресурс, то способ 220 переходит на этап 284 принятия решения, где выполняется определение того, выводит ли второй поток (например, поток 1) микрокоманду, которая имеет маркер 184 потока совместно используемых ресурсов, связанный с ней. Переходя к фиг.9, показанный указатель 182 вывода для потока 1 указывает на микрокоманду, имеющую и маркер 184 потока совместно используемых ресурсов, и маркер 186 потока синхронизации. В этой ситуации условие, представленное на этапе 284 принятия решения, будет выполнено, и способ 220 соответственно переходит на этап 280, где выполняется многопоточная операция удаления. Альтернативно, если указатель 182 вывода для второго потока

(например, потока 1) не указывает на микрокоманду, имеющую или маркер 184 потока совместно используемых ресурсов, или маркер 186 потока синхронизации, то способ продолжается на этапе 286, где вывод второго потока продолжается с помощью продвижения указателя 182 вывода. С этапа 286 способ 220 переходит обратно на этап 278 принятия решения, где снова происходит определение, столкнулся ли второй поток с событием.

Если на этапе 282 принятия решения определено, что обработка первого события для первого потока (например, потока 0) требует модификации ресурса совместно используемого состояния, то способ 220 переходит на этап 288 принятия решения, где происходит определение того, выводит ли второй поток (например, поток 1) микрокоманду, которая имеет маркер 186 потока синхронизации, связанный с ней. Если да, то многопоточная операция удаления выполняется на этапе 280. Если нет, то вывод микрокоманды для второго потока продолжается на этапе 286, пока или второй поток не столкнется с событием, или указатель 182 вывода для второго потока не индексирует микрокоманду, имеющую маркер 186 потока синхронизации, связанный с ней.

После начала операции удаления на этапе 280 на этапе 290 соответствующий блок 67 обработки событий, осуществленный в микропрограмме и упорядоченный блоком 66 упорядочения микропрограммы, продолжает обрабатывать соответствующее событие.

Событие виртуального удаления

Как описано выше, событие 260 ВИРТУАЛЬНОЕ УДАЛЕНИЕ обрабатывается немного другим способом, чем другие события. Для этой цели фиг.7В - последовательность операций, показывающая способ 291 обнаружения и обработки события 260 ВИРТУАЛЬНОЕ УДАЛЕНИЕ согласно типичному варианту осуществления. Способ 291 предполагает, что никакие события для второго потока в настоящее время не ожидают обработки (т.е. не записаны в регистре ожидающего обработки для второго потока).

Способ 291 начинается на этапе 292 с обнаружения блоком 188 обнаружения событий первого события для первого потока. Такое событие может быть любым из событий, которые описаны выше относительно фиг.8.

На этапе 293 блок 188 обнаружения событий останавливает вывод первого потока. На этапе 294 блок 188 обнаружения событий обнаруживает вывод микрокоманды с помощью или маркера 184 потока совместно используемого ресурса, или маркера потока синхронизации. На этапе 295 блок обработки "виртуального удаления" вызывается блоком 66 упорядочения микропрограммы. На этапе 296 блок обработки события "виртуального удаления" повторно запускает выполнение второго потока на микрокоманде, последующей за микрокомандой, выведенной на этапе 294. Затем способ 291 заканчивается на этапе 297.

Операция удаления

Фиг.11А - последовательность операций, показывающая способ 300 выполнения операции очистки (или удаления) в многопоточном процессоре,

поддерживающем по меньшей мере первый и второй потоки согласно примеру варианта осуществления. Способ 300 начинается на этапе 302 с установлением сигнала 170 удаления блоком 188 обнаружения событий в зависимости от возникновения и обнаружения события. Сигнал 170 удаления передается многочисленным функциональным блокам в многопоточном процессоре 30, и его установка или сброс определяется "окном", в котором выполняются действия по подготовке к очистке состояния и конфигурированию функциональных модулей. Фиг.12 - временная диаграмма, показывающая формирование сигнала 170 удаления, появляющегося синхронно с передним фронтом сигнала 304 синхронизации.

На этапе 303 оценивается конечный автомат активного потока.

На этапе 306 порядковый номер и сигнал последней микрокоманды, которые указывают, происходит или нет вывод микрокоманды, на которой происходит событие, для первого и второго потоков, передаются к схеме 122 управления распределением и списком свободных ресурсов и TBIT, который является структурой в блоке трассы предсказания ветвлений (БТПВ) (который является, в свою очередь, частью МТД 60) для отслеживания макрокоманд и информации указателя микрокоманды во входном блоке упорядоченной обработки процессора 30. TBIT использует эту информацию, чтобы зафиксировать информацию относительно события (например, микрокоманду и указатель команды для макрокоманды).

На этапе 308 блок 188 обнаружения событий создает и передает вектор события для каждого из первого и второго потоков на блок 66 упорядочения микропрограммы. Каждый вектор события включает в себя, между прочим, информацию, которая идентифицирует (1) физическое местоположение буфера переупорядочивания, который был выведен, когда была обнаружена точка удаления (или точка очистки) (т.е. значение каждого указателя 182 вывода, когда точка удаления была идентифицирована), (2) идентификатор блока обработки событий, который идентифицирует местоположение в блоке 66 упорядочения микропрограммы, где расположена микропрограмма, составляющая блок 67 обработки событий, для обработки обнаруженного события, и (3) идентификатор потока для идентификации или первого, или второго потока, и (4) бит приоритета потока, который определяет приоритет данного блока 67 обработки событий относительно блока обработки событий, вызванного для других потоков.

На этапе 310 схема 122 управления распределением и списком свободных ресурсов использует порядковые номера, переданные на этапе 306, для того, чтобы продвинуться к точке в теневой таблице псевдонимов регистров (теневая ТПР), в которой была обнаружена точка удаления, и на этапе 312 состояние первичной таблицы 120 псевдонимов регистров восстанавливается из теневой таблицы псевдонимов регистров.

На этапе 314 схема 122 управления распределением и списком свободных

ресурсов восстанавливает номера регистров (или "набор"), которые поступают из блока 134 управления списком свободных ресурсов, и назначает восстановленные номера регистров массиву 132 неиспользуемых регистров, из которого номера регистров могут снова быть распределены. Схема 122 управления распределением и списком свободных ресурсов, кроме того, устанавливает сигнал "восстановлено" (не показан), когда все соответствующие номера регистра восстановлены из блока 134 управления списком свободных ресурсов. Сигнал 170 удаления находится в установленном состоянии до того, как данный сигнал "восстановлено" будет принят от схемы 122 управления распределением и списком свободных ресурсов.

На этапе 316 вся "последняя" память (т.е. память, которая была очищена, но еще не модифицирована) и для первого, и для второго потоков, выполняет "дренаж" буфера упорядочивания памяти, используя схему передачи в память (не показана).

Затем на этапе 320 блок 188 обнаружения событий сбрасывает сигнал 170 удаления по переднему фронту сигнала синхронизации 304, как показано на фиг.12. Следует отметить, что сигнал 170 удаления удерживался в установленном состоянии в течение как минимум трех тактов сигнала 304 синхронизации. Однако в случае, когда сигнал "восстановлено" от схемы 122 управления распределением и списком свободных ресурсов не установлен в течение первых двух тактов сигнала 304 синхронизации после установления сигнала 170 удаления, блок 188 обнаружения событий будет продолжать установленное состояние сигнала 170 удаления дольше показанных трех тактов. В одном из вариантов осуществления сигнал 170 удаления может удерживаться достаточно долго (например, три тактовых цикла), чтобы позволить завершить этапы 303, 306 и 308, которые были обсуждены выше. Может потребоваться, чтобы сигнал 170 удаления удерживался в течение дополнительных циклов, чтобы позволить завершить этапы 310, 312, 314 и 316. Для этой цели буфер упорядочивания памяти устанавливает сигнал "дренаж буфера памяти выполнен", чтобы продлить выдачу сигнала удаления.

На этапе 322 блок 66 упорядочения микропрограммы и другие функциональные блоки в многопоточном процессоре 30 исследуют "активные биты", установленные конечным автоматом 171 активного потока для того, чтобы определить, находится ли каждый из первого и второго потоков в активном или неактивном состоянии после возникновения события. Более конкретно, конечный автомат 171 активного потока устанавливает соответствующий бит индикации для каждого потока, находящегося в многопоточном процессоре 30, который указывает, находится ли соответствующий поток в активном или неактивном ("спящем") состоянии. Событие, обнаруженное блоком 188 обнаружения событий и в соответствии с которым блок 188 обнаружения событий установил сигнал 170 удаления, может содержать событие 264 СОН или событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА, которые переключают первый или второй поток между

активным и неактивным состояниями. Как обозначено 324 на фиг.12, конечный автомат 171 активного потока оценивается в течение установленного сигнала 170 удаления, и состояние "активных битов" соответственно расценивается как разрешенное при сбросе сигнала 170 удаления.

На этапе 326 принятия решения каждый из функциональных блоков, которые исследовали активные биты конечного автомата 171 активного потока, определяет, являются ли первый или второй потоки активными. Если определено, что оба потока активные, основываясь на состоянии активных битов, то способ 300 переходит на этап 328, где каждый из функциональных блоков конфигурируется таким образом, чтобы поддерживать и обслуживать и первый, и второй активные потоки. Например, возможности хранения и буферизации, обеспеченные в различных функциональных блоках, могут быть логически распределены, активизируя второй указатель, или второй набор указателей, которые ограничены определенным набором (или диапазоном) записей массива памяти. Дополнительно, специфическая МП поддержка может быть активизирована, если два потока активны. Например, схема выбора потока, связанная с блоком упорядочения микропрограммы, может устанавливать последовательность потоков первого потока (например, T0), второго потока (например, T1) или и первого, и второго потоков (например, T0 и T1) способом "пинг-понг" (поочередно), основываясь на выходе конечного автомата 171 активного потока. Дополнительно ограничение пропускания синхронизации может быть выполнено, основываясь на выходных битах конечного автомата активного потока. В дополнительном варианте осуществления любое число конечных автоматов в процессоре может изменять свое поведение или изменять состояние, основываясь на выходе конечного автомата активного потока. На этапе 330 блок 66 упорядочения микропрограммы затем переходит к последовательности микрокоманд и для первого, и для второго потоков.

Альтернативно, если на этапе 326 принятия решения определено, что только один из первого и второго потоков является активным или что оба потока являются неактивными, то каждый из функциональных блоков конфигурируется таким образом, чтобы поддерживать и обслуживать только один активный поток на этапе 332, и специфическая МП поддержка может быть дезактивирована. Когда никакие потоки не активны, функциональные блоки конфигурируются по умолчанию для поддержания одного активного потока. В случае, когда функциональный блок был предварительно сконфигурирован (например, логически разделен), чтобы поддерживать множество потоков, указатели, используемые, чтобы поддерживать дополнительные потоки, могут быть заблокированы, и набор записей массива данных, на которые указывает оставшийся указатель, может быть расширен, чтобы они включали в себя записи, на которые ранее указывали заблокированные указатели. Таким образом, следует признать, что записи данных, которые предварительно

были распределены другим потокам, могут затем стать доступными для использования одним активным потоком. При наличии больших ресурсов, доступных одному активному потоку, когда дополнительные потоки неактивны, производительность одного оставшегося потока может быть увеличена относительно его производительности в случае, когда другие потоки также поддерживаются в многопоточном процессоре 30.

На этапе 334 блок 66 упорядочения микропрограммы игнорирует векторы события для неактивного потока или неактивных потоков и упорядочивает микрокоманды только для возможного активного потока. Когда никакие потоки не активны, блок 66 упорядочения микропрограммы игнорирует векторы события для всех потоков.

Обеспечивая активные биты, установленные конечным автоматом 171 активного потока, которые могут быть проверены различными функциональными блоками на предмет сброса сигнала 170 удаления (передающего сигнал конца операции удаления), обеспечивается удобная и централизованная индикация, согласно которой различные функциональные блоки могут быть сконфигурированы, чтобы поддержать правильное число активных потоков в многопоточном процессоре 30 после завершения операции удаления.

Фиг.11В - структурная схема, показывающая типичную схему 329 конфигурирования, которая связана с функциональным блоком 331 и которая работает, чтобы сконфигурировать функциональный блок 331 для поддержки одного или большего количества активных потоков в многопоточном процессоре. Функциональный блок 331 может быть любым из функциональных блоков, описанных выше, или любым функциональным блоком, который будет включать в себя процессор, как будет понятно специалисту. Показанный функциональный блок 331 имеет и логические компоненты, и компоненты памяти, которые конфигурируются схемой 329 конфигурирования. Например, компонент памяти может содержать совокупность регистров. Каждый из этих регистров может быть распределен для хранения микрокоманды или данных для одного определенного потока, когда активны множество потоков (т.е. когда процессор работает в режиме МП). Соответственно, показанный на фиг.11В компонент памяти логически разделен для того, чтобы поддержать первый и второй потоки (например, T0 и T1). Конечно, компонент памяти может быть разделен так, чтобы поддержать любое число активных потоков.

Показанный логический компонент включает в себя МП схему, которая должна определенно поддерживать многопоточную работу в процессоре (т.е. режим МП).

Показанная схема 329 конфигурирования устанавливает значения 333 указателя, которые выводятся к компоненту памяти функционального блока 331. В одном типичном варианте осуществления эти значения 333 указателя используются для того, чтобы логически разделять компонент памяти. Например, отдельная пара значений указателя чтения и записи могла бы быть

сгенерирована для каждого активного потока. Верхние и нижние границы значений указателя для каждого потока определяются схемой 329 конфигурирования, в зависимости от количества активных потоков. Например, диапазон регистров, которые могут обозначаться набором значений указателя для конкретного потока, может быть увеличен для того, чтобы охватить регистры, предварительно распределенные другому потоку, если другой поток стал неактивным.

Схема 329 конфигурирования также включает в себя индикацию 335 разрешения поддержки МП, которая выводится к логическому компоненту функционального блока для того, чтобы разрешать или отключать схему поддержки МП функциональной схемы 331.

Активные биты 327, выводимые конечным автоматом 174 активного потока, обеспечивают передачу на вход схемы конфигурирования и используются схемой 329 конфигурирования для того, чтобы генерировать соответствующую точку значений 333 и обеспечивать передачу соответствующего выходного сигнала разрешения поддержки МП.

Монопольный доступ посредством блока обработки событий

Некоторые блоки обработки событий (например, те, которые обрабатывают события страничной переадресации и синхронизации) требуют монопольного доступа к многопоточному процессору 30, чтобы использовать совместно используемые ресурсы и изменять совместно используемое состояние.

Соответственно, блок 66 упорядочения микропрограммы, воплощенный в конечном автомате 69 монопольного доступа, который дает монопольный доступ, в свою очередь, на блоки обработки событий для первого и второго потоков, где каждый из этих блоков обработки событий требует такого монопольного доступа. К конечному автомату 69 монопольного доступа можно обращаться только тогда, когда больше чем один поток активен в многопоточном процессоре 30. Маркер потока, связанный с блоком обработки события, который обеспечивается монопольным доступом, вставляется в данный поток, чтобы отметить конец исключительного кода, содержащегося в блоке обработки события. Как только монопольный доступ закончен для всех потоков, блок 66 упорядочения микропрограммы восстанавливает нормальную выдачу микрокоманд.

Фиг.13 - последовательность операций, показывающая способ 400 обеспечения монопольного доступа к блоку 67 обработки событий в многопоточном процессоре 30 согласно типичному варианту осуществления. Способ 400 начинается на этапе 402 с принятия блоком 66 упорядочения микропрограммы первого и второго векторов события для соответственно первого и второго потоков от блока 188 обнаружения событий. Как описано выше, каждый из первого и второго векторов события идентифицирует соответствующий блок 67 обработки событий.

На этапе принятия решения 403 выполняется определение того, являются ли больше чем один (1) поток активным. Это определение выполняется блоком

упорядочения микропрограммы на основе конечного автомата 171 активного потока. Если нет, то способ 400 продолжается на этапе 434. Если да, то способ 400 продолжается на этапе 404 принятия решения.

На этапе 404 принятия решения блок 66 упорядочения микропрограммы определяет, требует ли первый или второй блок 67 обработки событий монопольного доступа к совместно используемому ресурсу или изменяет совместно используемое состояние. Если да, то на этапе 406 блок 66 упорядочения микропрограммы реализует конечный автомат 69 монопольного доступа для обеспечения монопольного доступа по очереди каждому из первого и второго блоков 67 обработки событий. Фиг.14 - диаграмма состояний, изображающая работу конечного автомата 69 монопольного доступа согласно примерному варианту осуществления. Показанный конечный автомат 69 включает в себя пять состояний. В первом состоянии 408 обе микропрограммы для первого и второго потоков выдаются блоком 66 упорядочения микропрограммы. При возникновении операции 410 удаления в ответ на событие, которое требует монопольного доступа блока обработки событий, конечный автомат 69 переходит ко второму состоянию 412, в котором вызывается блок 67 обработки первого события (т.е. микрокоманды), связанного с событием для первого потока. После упорядочивания всех микрокоманд, которые составляют блок 67 обработки первого события, и также после завершения всех операций, заданных такими микрокомандами, блок 66 упорядочения микропрограммы затем выдает микрокоманду 414 останова (например, микрокоманду, имеющую связанный маркер потока "останов") для перехода конечного автомата 69 от второго состояния 412 к третьему состоянию 416, в котором выдача микрокоманд первого потока остановлена. На 418, микрокоманда останова, выданная на 414, возвращается от буфера 162 переупорядочивания, таким образом переводя конечный автомат 69 из третьего состояния 416 в четвертое состояние 420, в котором блок 66 упорядочения микропрограммы вызывает блок 67 обработки второго события, связанного с событием второго потока. После упорядочивания всех микрокоманд, которые составляют блок 67 обработки второго события, и также после завершения всех операций, заданных такими микрокомандами, блок 66 упорядочения микропрограммы затем выдает дополнительную микрокоманду останова на 422, чтобы перевести конечный автомат 69 из четвертого состояния в пятое состояние 424, в котором остановлен блок 67 обработки второго события. На 426, микрокоманда останова, выданная на 422, возвращается от буфера 162 переупорядочивания, таким образом переводя конечный автомат 69 из пятого состояния 424 назад к первому состоянию 408.

На этапе 432 возобновляется нормальное упорядочивание и выдача микрокоманд для первого и второго потоков, предполагая, что оба потока являются активными.

Альтернативно, если на этапе принятия решения 404 определено, что ни один из

первого или второго блоков обработки событий не требует монопольного доступа к совместно используемым ресурсам или состоянию процессора 30, то способ продолжается на этапе 434, на котором блок 66 упорядочения микропрограммы упорядочивает микропрограммы, составляющие блоки 67 обработки первого и второго событий, не монопольным, переключающимся способом.

10 Конечный автомат активного потока (171) Фиг.15 - диаграмма 500 состояний, показывающая состояния, которые может принимать конечный автомат 171 активного потока согласно примеру варианта осуществления, а также показывающая события перехода, которые могут вызывать переход конечного автомата 171 активного потока между различными состояниями согласно типичному варианту осуществления.

15 Показанный конечный автомат 171 активного потока постоянно находится в одном из четырех состояний, а именно состояние 502 единственного потока 0 (ЕПО), состояние 504 единственного потока 1 (ЕП1), многопоточное (МП) состояние 506 и состояние 508 нулевого потока (НП). 20 Конечный автомат 171 активного потока устанавливает один активный бит для каждого потока, который, когда установлен в 1, идентифицирует связанный с ним поток как активный, а когда сброшен, указывает, что связанный с ним поток неактивен или "спит".

30 Переходы между четырьмя состояниями 502-508 вызываются парами событий, каждое событие из пары событий относится к первому или второму потоку. На диаграмме 500 состояний множество типов события обозначено как содействующие переходу между состояниями. А именно, событие СОН - событие, который заставляет поток стать неактивным. Событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА - событие, появление которого для определенного потока вызывает переход потока из неактивного состояния в активное состояние. То, будет ли квалифицировано конкретное событие как событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА, может зависеть от события СОН, которое заставляет поток стать неактивным. А именно, только некоторые события заставляют поток стать активным из неактивного состояния, в который он перешел в результате определенного события СОН. Событие УДАЛЕНИЕ любое событие, появление которого для определенного потока приводит к выполнению операции удаления, как описано выше. Все события, которые 35 обсуждались выше относительно фиг.8, потенциально содержат события удаления. Наконец, возникновение "отсутствия события" для определенного потока также показано в диаграмме 500 состояний как состояние, которое может присутствовать в комбинации с возникновением события для дополнительного потока, чтобы вызвать переход в другое состояние.

40 В одном из вариантов осуществления, если событие СОН сообщается для конкретного потока, а событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА для того же потока ожидает обработки, то событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА обслуживается немедленно (например, поток не "засыпает" и потом "пробуждается", чтобы обслужить

событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА). Обратное также может быть истинным, в этом случае событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА может быть сообщено для конкретного потока, а событие СОН ожидает обработки, после чего обслуживается событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА.

При установлении сигнала 170 удаления блоком 188 обнаружения событий оценивается конечный автомат 171 активного потока, что обозначено как 324 на фиг.12. При сбросе сигнала 170 удаления все функциональные блоки в многопоточном процессоре 30 конфигурируются, основываясь на активных битах, установленных конечным автоматом 171 активного потока. А именно, блок 160 проверки повторного выполнения и вывода (ППВВ) передает сгенерированный сигнал, основанный на активных битах, ко всем задействованным функциональным блокам, чтобы указать функциональным блокам, сколько потоков существует в многопоточном процессоре и который из этих потоков является активным. После установления сигнала 170 удаления конфигурирование функциональных блоков (например, разделен или не разделен) типично завершается в течение одного такта сигнала 304 синхронизации.

Выход и вход потока

Настоящее изобретение предлагает примерный механизм, посредством которого потоки могут входить и выходить в многопоточном процессоре 30 (например, стать активными или неактивными), где такие вход и выход происходят в постоянной последовательности независимо от номеров выполняемых потоков и где сигналы синхронизации к различным функциональным блокам могут быть постепенно остановлены, когда никакие дополнительные потоки в многопоточном процессоре 30 не являются активными или выполняемыми.

Как описано выше в отношении диаграммы 500 состояний, вход потока (или активация) происходит в зависимости от обнаружения события ПЕРЕЗАГРУЗКА КОНВЕЙЕРА для неактивного в настоящее время потока. Определение события ПЕРЕЗАГРУЗКА КОНВЕЙЕРА для определенного неактивного потока зависит от причины того, почему соответствующий поток неактивен. Выход из потока происходит в зависимости от события СОН для активного в настоящее время потока. Примеры событий типа СОН включают в себя выполнение команды останова (HLT, OCT), содержащейся в активном потоке, обнаружение условия ОТКЛЮЧЕНИЕ или ОШИБКИ ОТКЛЮЧЕНИЯ или условия "ожидания SIPI" (запуска межпроцессорного прерывания (ЗМПП)) для активного потока.

Фиг.16А - последовательность операций, показывающая способ 600 выхода из активного потока при обнаружении события СОН для активного потока согласно типичному варианту осуществления настоящего изобретения. Способ 600 начинается на этапе 602, на котором все требуемое состояние для активного потока сохраняется и освобождаются все записи в регистрах файла 124 регистров, которые были предварительно распределены микрокомандам активного потока. Например,

освобождаются 28 записей, которые были предварительно распределены микрокомандам активного потока, из 128 записей регистров в файле 124 регистров. Содержимое освобожденных регистров для активного потока сохраняется в "рабочей области", которая может содержать массив регистров или оперативную память (ОП), присоединенную к шине управляющих регистров в многопоточном процессоре 30.

Освобождение записей регистров в файле 124 регистров может быть выполнено освобождающей последовательностью микропрограммы, которая запускается блоком 66 упорядочения микропрограммы в ответ на обнаружение событий STOPCLK (ОСТАНОВ СИНХРОНИЗАЦИИ), ОСТАНОВ (HLT) или ОТКЛЮЧЕНИЕ для активного потока.

Освобождающая последовательность микропрограммы выполняется для того, чтобы удалить (или сделать недействительными) записи в элементах файла регистров в блоке 134 управления списком свободных ресурсов и создать (или сделать действительными) записи для элементов файла регистров в массиве 132 неиспользуемых регистров. Другими словами, записи для освобожденных элементов файла регистров перемещаются от блока 134 управления списком свободных ресурсов к массиву 132 неиспользуемых регистров с помощью освобождающей последовательности микропрограммы.

Фиг.16В - схематическое представление типичного варианта осуществления операций, которые могут быть выполнены на этапе 602. Например, показывается перемещение в рабочую область содержимого первого набора регистров файла 124 регистров, которые были предварительно распределены первому потоку (например, T0). Дополнительные операции, которые могут быть выполнены при сохранении состояния, включают в себя сохранение содержимого архитектурных регистров для выходящего потока в рабочей области, а также сохранение содержимого временных регистров микропрограммы, распределенных первому потоку, в рабочей области при выходе данного первого потока. Регистры, освобожденные при выходе потока, затем доступны для перераспределения другому потоку (например, T1).

Следует признать, что при повторном вхождении конкретного потока (например, T0), содержимое регистров, распределенных этому потоку, может быть восстановлено из рабочей области, как обозначено прерывистой линией на фиг.16В.

На этапе 604 "микрокоманда заграждения" (которая ориентирована на конкретный поток) для выходящего потока вставляется в поток микрокоманд выходящего потока, чтобы выполнить "дренаж" (селекцию) любых оставшихся ожидающих обработки обращений к памяти, связанных с потоком из буфера 48 упорядочивания памяти, различных кэш и шин процессора. Эта операция не выводится, пока все эти этапы не закончены.

Поскольку блоки 20 исполнения выполняют микрокоманды относительно быстро, все новые микрокоманды, добавленные на вход блока исполнения, очищаются при установлении сигнала

удаления в зависимости от обнаружения события СОН. Как описано выше, сигнал 170 удаления сохраняется установленным в течение достаточного периода времени (например, три тактовых цикла), чтобы позволить микрокомандам, которые вошли в блок 70 исполнения до выдачи сигнала 170 удаления, появиться оттуда. Когда эти микрокоманды появляются из блока 70 исполнения, то они очищаются и обратная запись отменяется.

На этапе 606 регистр 208 обратного восстановления, находящийся в блоке 188 обнаружения событий, устанавливается для того, чтобы указать, что выходящий поток находится в неактивном состоянии (или состоянии "сна"), с помощью сгенерированной блоком 66 упорядочения микропрограммы микрокоманды, которая выполняет обратную запись значения, которое устанавливает состояние регистра обратного восстановления.

На этапе 608 регистры 206 запрещения события для выходящего потока устанавливаются для того, чтобы запретить события, не являющиеся событиями перезагрузки конвейера для выходящего потока, с помощью микрокоманд записи регистра управления, выданных блоком 66 упорядочения микропрограммы. Команда на установку регистра запрещения события для выходящего потока, выданная как микрокоманда управления регистром, зависит от типа обслуживаемого события СОН. Как было обсуждено выше, в зависимости от события СОН, которое вызвало переход к неактивному состоянию, только некоторые события квалифицируются как события перезагрузки конвейера для неактивного потока. Определение того, квалифицировать ли событие как событие перезагрузки конвейера для конкретного неактивного потока, выполняется с конкретной ссылкой на состояние регистра 206 запрещения события для неактивного потока.

На этапе 612 о событии СОН для выходящего потока сообщается с помощью использования специальной микрокоманды, которая помещает код событие СОН в поле информации отказа обратной записи специальной микрокоманды.

Фиг.17 - последовательность операций, показывающая способ 700 ввода неактивного потока в активное состояние после обнаружения события ПЕРЕЗАГРУЗКА КОНВЕЙЕРА для неактивного потока согласно типичному варианту осуществления. Способ 700 начинается на этапе 702 с обнаружением возникновения события для события, которое может или не может квалифицироваться как событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА для неактивного потока. На этапе 703 принятие решения происходит определение схемой 185 обнаружения событий для соответствующего события, квалифицируется ли событие как событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА для неактивного потока. Для этой цели схема 185 обнаружения событий исследует регистр 206 запрещения события в регистрах 200 из блока 188 обнаружения событий. Если соответствующий тип событий не обозначен как запрещенное событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА для неактивного потока, то способ 700 переходит на этап 704, где при

необходимости включается синхронизация, о событии сообщается обычным образом (ожидание точки удаления в другом потоке), и блок обработки вызывается как для любого события. Блок обработки событий проверяет состояние СОН для потока и, если оно установлено, продолжает восстанавливать состояние микропрограммы на этапе 706. Блок 67 обработки событий подтверждает неактивное состояние потока, обращаясь к регистру 208 обратного восстановления.

Более конкретно, блок 67 обработки событий переходит к восстановлению состояния микропрограммы для вводимого потока, восстанавливая все сохраненные регистры: регистр состояния, регистр запрещения состояния и информацию указателя команды.

При восстановлении состояния микропрограммы на этапе 706 способ 700 переходит на этап 708, где архитектурное состояние восстанавливается для вводимого потока. На этапе 710 регистр 206 запрещения события для вводимого потока сбрасывается или очищается соответствующей микрокомандой, выданной блоком 66 упорядочения микропрограммы. На этапе 712 блок 67 обработки событий переходит к обслуживанию событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА. В этой точке микропрограмма, составляющая блок 67 обработки событий, выполняется в многопоточном процессоре 30 для того, чтобы исполнить ряд операций в зависимости от возникновения события. На этапе 716 операции выборки команды затем снова продолжаются в процессоре 30 для вводимого потока. Способ 700 затем заканчивается на этапе 718.

Схема управления синхронизацией. Чтобы уменьшить потребляемую мощность и рассеиваемое тепло в многопоточном процессоре 30, желательно останавливать или приостанавливать по меньшей мере некоторые сигналы синхронизации в процессоре 30 при некоторых условиях. Фиг.18 - последовательность операций, показывающая согласно примерному варианту осуществления способ 800 остановки или приостановки выбранных сигналов синхронизации в многопоточном процессоре, таком как описанный выше типовой процессор 30. Для целей настоящего описания ссылка на приостановку или остановку сигналов синхронизации в процессоре должна использоваться для того, чтобы охватить множество способов приостановки или остановки сигнала или сигналов синхронизации в процессоре 30. Например, контур фазовой автоподстройки частоты (КФАП) в процессоре 30 может быть приостановлен, распространение сигнала синхронизации ядра по магистрали синхронизации может быть запрещено, или распространение сигнала синхронизации через магистраль синхронизации к отдельным функциональным блокам в процессоре может быть запрещено с помощью логических схем или предотвращено иным образом. Один из вариантов осуществления предусматривает дальнейшую ситуацию, в которой доставка внутреннего сигнала синхронизации к функциональным блокам в процессоре 30 приостановлена или остановлена в функциональном блоке с помощью базового

компонента функционального блока. Соответственно, внутренний сигнал синхронизации может передаваться некоторым функциональным блокам, в то время как запрещаться для других функциональных блоков. Такое средство описано в контексте однопоточного микропроцессора в патенте США №5655127.

Способ 800, показанный на фиг.18, в одном из вариантов осуществления может быть выполнен схемой 35 управления синхронизацией, которая содержится в блоке 32 интерфейса шины процессора 30. В альтернативных вариантах осуществления схема 35 управления синхронизацией, конечно, может располагаться где-нибудь вне процессора 30. Фиг.19А и 19В - структурная схема и схематическая диаграмма, соответственно иллюстрирующие более подробно пример схемы 35 управления синхронизацией.

Возвращаясь к фиг.19А, показанная схема 35 управления синхронизацией принимает три первичных входных сигнала, а именно (1) активные биты 820 (например, TO_ACTIVE и T1_ACTIVE), выводимые через конечный автомат 174 активного потока; (2) сигналы 211 обнаружения события, выводимые блоком 188 обнаружения событий, и (3) сигнал 822 управления слежением, выводимый блоком 32 интерфейса шины, который обнаруживает отслеживаемый доступ к шине и устанавливает сигнал 882. Схема 35 управления синхронизацией использует эти входные сигналы для того, чтобы сгенерировать сигнал 826 остановки синхронизации, который, в свою очередь, подавляет или запрещает синхронизацию некоторых функциональных модулей в процессоре 30.

Фиг.19В - схематическая диаграмма, показывающая типичную комбинационную схему, которая использует входные сигналы 211, 820 и 822 для того, чтобы вывести сигнал 826 остановки синхронизации. А именно, сигнал 211 блока обнаружения событий передается на вход логической схемы 822 ИЛИ, откуда, в свою очередь, передается на вход следующей логической схемы 824 ИЛИ. Активные биты 820 и сигнал 822 управление слежением также передаются на вход этой логической схемы 824 ИЛИ-НЕ, которая логически объединяет эти входные сигналы и выводит сигнал 826 остановки синхронизации.

Обращаясь к фиг.18, способ 800 начинается на этапе 802 принятия решения с определения того, является ли какой-нибудь поток (например, первый поток и второй поток) активным в многопоточном процессоре 30. Это определение отражается выводом активных битов 820 на логическую схему ИЛИ 824 на фиг.19В. В то время как примерный вариант осуществления показывает определение, которое может быть выполнено для двух потоков, следует признать, что данное определение может быть сделано для любого числа потоков, поддерживаемых в многопоточном процессоре.

При отрицательном определении на этапе 802 принятия решения способ 800 переходит на этап 804 принятия решения, где происходит определение того, существуют ли какие-нибудь не запрещенные ожидающие обработки события для любого из потоков,

поддерживаемых в многопоточном процессоре. Снова в данном типичном варианте осуществления данный этап содержит определение, ожидают ли обработки какие-нибудь события для первого или второго потока. Данное определение представлено вводом сигнала 211 обнаружения события на логическую схему 822 ИЛИ, показанную на фиг.19В.

При отрицательном определении на этапе 804 принятия решения выполняется дополнительное определение на этапе 806 принятия решения, обрабатывается ли какое-нибудь слежение (например, слежение за шиной, SNC слежение или другие виды слежения) шиной процессора. В данном примере варианта осуществления настоящего изобретения данное определение осуществляется вводом сигнала 822 управления слежением на логическую схему 824 ИЛИ.

При отрицательном определении на этапе 806 принятия решения способ 800 переходит на этап 808, где внутренние сигналы синхронизации к выбранным функциональным блокам останавливаются или подавляются. А именно, сигналы синхронизации к схеме обработки ожидания обращения к шине и схеме обращения к шине не приостанавливаются или не останавливаются, поскольку это позволяет блоку 32 интерфейса шины обнаруживать события ПЕРЕЗАГРУЗКА КОНВЕЙЕРА или слежения, происходящие на системной шине (например, событие на входных контактах), и восстанавливать синхронизацию к функциональным блокам в зависимости от таких событий ПЕРЕЗАГРУЗКА КОНВЕЙЕРА. Подавление внутренних сигналов синхронизации к функциональным блокам осуществляется установлением сигнала 826 остановки синхронизации, наличие которого не пропускает сигнал синхронизации к предопределенным функциональным блокам.

При завершении этапа 808 способ 800 возвращается назад на этап 802 принятия решения. После этого определения на этапах 802, 804 и 806 принятия решения могут выполняться в цикле обычным образом.

При положительном определении на любом из этапов принятия решения 802, 804 и 806 способ 800 переходит на этап 810, где, если сигналы синхронизации к некоторым функциональным блокам были запрещены, эти внутренние сигналы синхронизации затем снова активизируются. Альтернативно, если сигналы синхронизации уже активны, эти сигналы синхронизации поддерживаются в активном состоянии.

Когда выполняется этап 810 в ответ на событие ПЕРЕЗАГРУЗКА КОНВЕЙЕРА (например, после положительного определения на этапе 804 принятия решения), функциональные блоки в микропроцессоре могут быть активно разделены способом, описанным выше, основываясь на количестве активных потоков, при установленном сигнале удаления. Например, в многопоточном процессоре 30, имеющем два или более потоков, некоторые из этих потоков могут быть неактивны, и в таком случае функциональные блоки не будут разделены для того, чтобы разместить неактивные потоки.

После завершения этапа 810 способ 800

снова переходит назад на этап 802 принятия решения, и начинают другую итерацию принятия решений, представленную этапами 802, 804 и 806 принятия решения.

Таким образом, были описаны способ и устройство для управления сигналом синхронизации в многопоточном процессоре. Хотя настоящее описание было выполнено в отношении определенных типичных вариантов осуществления, будет очевидно, что различные модификации и изменения данных вариантов осуществления могут быть сделаны без отрыва от более широкой формы и объема изобретения. Соответственно, описание и чертежи должны быть расценены в иллюстративном, а не в ограничительном смысле.

Формула изобретения:

1. Способ управления сигналом синхронизации в многопоточном процессоре, включающий в себя следующие этапы: осуществляют индикацию ожидающего обработки события для каждого из множества потоков, поддерживаемых в многопоточном процессоре, осуществляют индикацию активного или неактивного состояния для каждого из множества потоков, поддерживаемых в многопоточном процессоре, обнаруживают условие блокировки синхронизации, указываемое индикацией отсутствия ожидающих обработки событий для каждого из множества потоков и неактивное состояние для каждого из множества потоков, и блокируют сигнал синхронизации, если он разрешен, для, по меньшей мере, одного функционального блока в многопоточном процессоре в ответ на обнаружение условия блокировки синхронизации.

2. Способ по п.1, включающий в себя обнаружение условия разрешения синхронизации, на которое указывает индикация ожидающего обработки события, для, по меньшей мере, одного потока из множества потоков, поддерживаемых в многопоточном процессоре, или индикация активного состояния для, по меньшей мере, одного потока из множества потоков, поддерживаемых в многопоточном процессоре, и разрешение сигнала синхронизации, если он заблокирован, для, по меньшей мере, одного функционального блока в многопоточном процессоре в ответ на обнаружение условия разрешения синхронизации.

3. Способ по п.2, включающий в себя осуществление индикации запрещения события для, по меньшей мере, первого потока из множества потоков, поддерживаемых многопоточным процессором, причем индикация запрещения события идентифицирует, по меньшей мере, одно событие для первого потока, которое не является событием, которое вызывает переключение первого потока между неактивным и активным состояниями.

4. Способ по п.3, в котором обнаружение условия блокировки синхронизации и условия разрешения синхронизации зависит от индикации запрещения события для первого потока и условия блокировки синхронизации обнаруживают или условие разрешения синхронизации не обнаруживают, если ожидающее обработки событие обозначено индикацией запрещения события для первого

потока как запрещенное.

5. Способ по п.1, включающий в себя осуществление индикации операции доступа к шине и обнаружение условия блокировки синхронизации только в отсутствие индикации операции доступа к шине.

6. Способ по п.3, в котором операция доступа к шине содержит операцию отслеживания и индикация операции доступа к шине обеспечивается, когда операция доступа к шине активна в течение предопределенной продолжительности отслеживания.

7. Способ по п.1, в котором осуществление индикации активности потока для каждого из множества потоков содержит поддержание работы конечного автомата активного потока, который выдает сигнал для каждого из множества поддерживаемых потоков для того, чтобы указать, находится ли соответствующий поток в активном или неактивном состоянии.

8. Способ по п.1, в котором блокировка сигнала синхронизации для, по меньшей мере, одного функционального блока содержит этап логической обработки сигнала синхронизации.

9. Способ по п.1, в котором блокировка сигнала синхронизации для, по меньшей мере, одного функционального блока содержит этап блокировки распространения сигнала синхронизации процессора по магистрали синхронизации процессора.

10. Способ по п.1, в котором блокировка сигнала синхронизации для, по меньшей мере, одного функционального блока содержит блокировку контура фазовой автоподстройки частоты, который генерирует сигнал синхронизации процессора.

11. Способ по п.1, в котором блокировка сигнала синхронизации содержит поддержание подачи сигнала синхронизации к входам шины многопоточного процессора для того, чтобы разрешить обнаружение операции или события доступа к шине, когда сигнал синхронизации заблокирован.

12. Устройство управления сигналом синхронизации в многопоточном процессоре, содержащее индикатор ожидающего обработки события, который осуществляет индикацию ожидающего обработки события для каждого из множества потоков, поддерживаемых в многопоточном процессоре, индикатор активного потока, который осуществляет индикацию активного или неактивного состояния для каждого из множества потоков, поддерживаемых в многопоточном процессоре, и логическую схему управления синхронизацией для обнаружения условия блокировки синхронизации, обозначенного индикацией отсутствия ожидающих обработки событий для каждого из множества потоков, и неактивного состояния для каждого из множества потоков, и отключения сигнала синхронизации для, по меньшей мере, одного функционального блока в многопоточном процессоре в ответ на обнаружение условия блокировки синхронизации.

13. Устройство по п.12, в котором логическая схема управления синхронизацией обнаруживает условие разрешения синхронизации, обозначенное индикацией ожидающего обработки события для, по меньшей мере, одного потока из

множества потоков, поддерживаемых в многопоточном процессоре, и разрешает сигнал синхронизации, если он заблокирован, для, по меньшей мере, одного функционального блока в многопоточном процессоре в ответ на обнаружение условия разрешения синхронизации.

14. Устройство по п.13, включающее в себя индикатор запрещения события, поддерживающий индикацию запрещения события для, по меньшей мере, первого потока из множества потоков, поддерживаемых многопоточным процессором, причем индикация запрещения события идентифицирует, по меньшей мере, одно событие для первого потока, которое не является событием, которое вызывает переход первого потока между неактивным и активным состоянием.

15. Устройство по п.14, в котором логическая схема управления синхронизацией обнаруживает условие блокировки синхронизации и условие разрешения синхронизации в зависимости от индикации запрещения события для первого потока, причем условие блокировки синхронизации обнаруживается или условие разрешения синхронизации не обнаруживается логической схемой управления синхронизацией, если обозначенное ожидающее обработки событие указано как запрещенное индикацией запрещения события для первого потока.

16. Устройство по п.12, в котором логическая схема управления синхронизацией обнаруживает условие разрешения синхронизации, обозначенное индикацией активного состояния для, по меньшей мере, одного потока из множества потоков, поддерживаемых в многопоточном процессоре, и разрешает сигнал синхронизации, если он заблокирован, для, по меньшей мере, одного функционального блока в многопоточном процессоре в ответ на обнаружение условия разрешения синхронизации.

17. Устройство по п.12, включающее в себя схему слежения за шиной, которая обеспечивает индикацию операции доступа к шине и в котором логическая схема управления синхронизацией обнаруживает условие блокировки синхронизации только в отсутствие индикации операции доступа к шине от схемы слежения за шиной.

18. Устройство по п.15, в котором операция доступа к шине содержит операцию слежения и в котором схема слежения за шиной обеспечивает индикацию операции доступа к шине, когда операция доступа к шине активна в течение predetermined продолжительности слежения.

19. Устройство по п.12, включающее в себя конечный автомат активного потока, который выдает сигнал для каждого из множества поддерживаемых потоков для того, чтобы указать, находится ли соответствующий поток в активном состоянии или неактивном состоянии.

20. Устройство по п.12, в котором логическая схема управления синхронизацией блокирует сигнал синхронизации для, по меньшей мере, одного функционального блока с помощью логической обработки сигнала синхронизации.

21. Устройство по п.12, в котором логическая схема управления синхронизацией блокирует сигнал синхронизации для, по меньшей мере, одного функционального блока, блокируя распространение сигнала синхронизации процессора по магистрали синхронизации процессора.

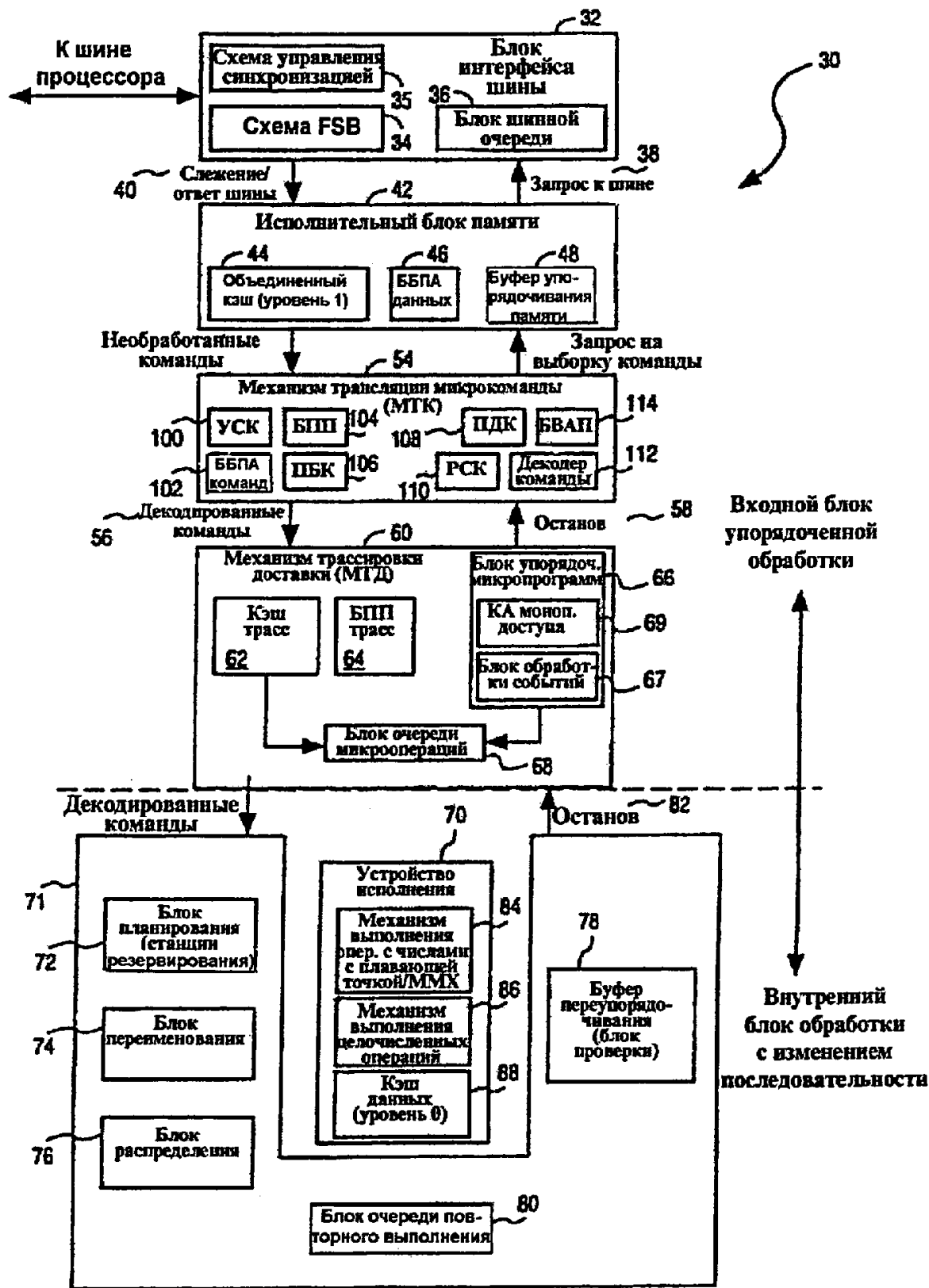
22. Устройство по п.12, в котором логическая схема управления синхронизацией блокирует сигнал синхронизации для, по меньшей мере, одного функционального блока, блокируя контур фазовой автоподстройки частоты, который генерирует сигнал синхронизации процессора.

23. Устройство по п.12, в котором логическая схема управления синхронизацией блокирует сигнал синхронизации для, по меньшей мере, одного функционального блока, но поддерживает подачу сигнала синхронизации к входам шины многопоточного процессора для того, чтобы разрешить обнаружение операции или события доступа к шине, когда сигнал синхронизации заблокирован.

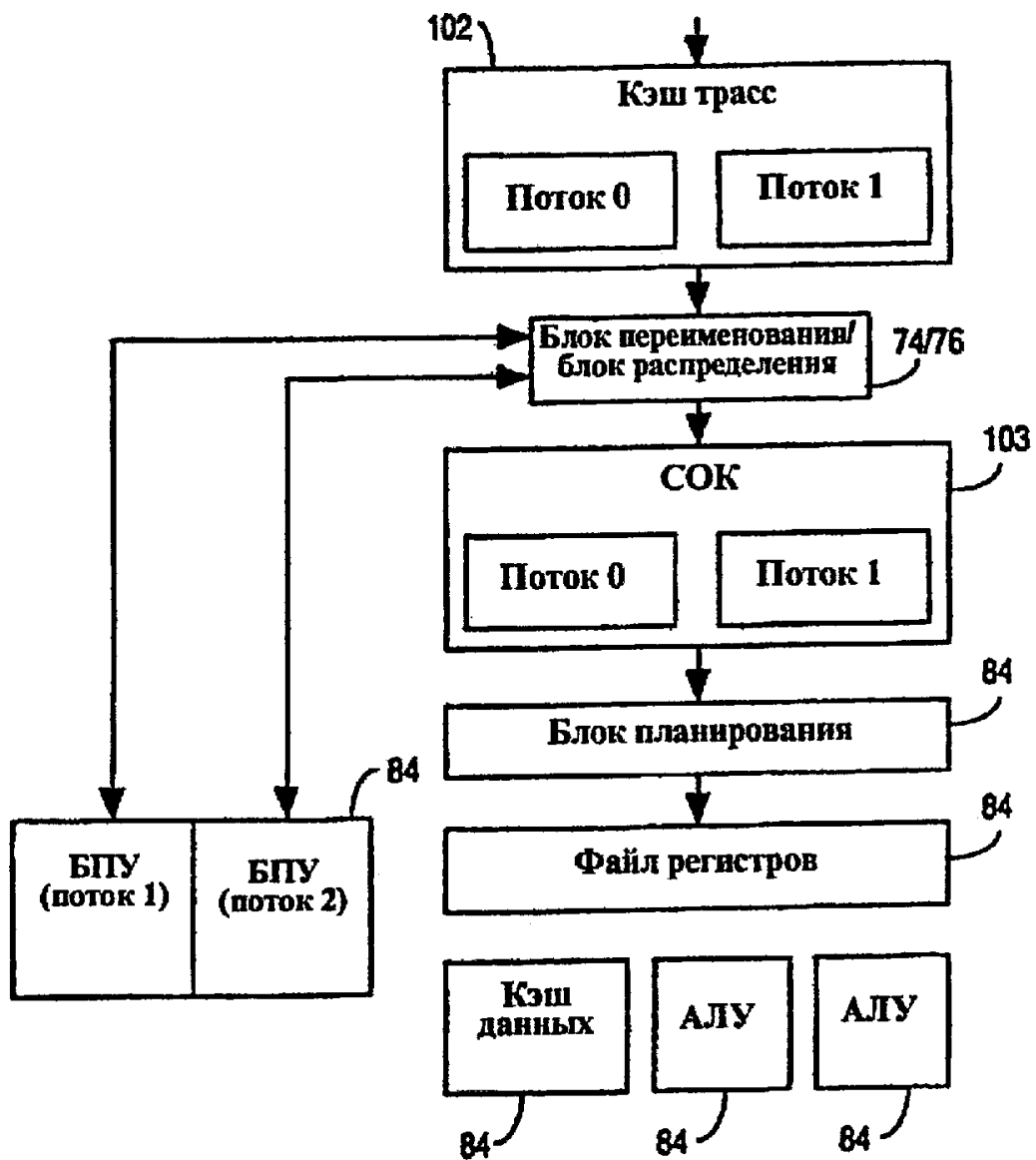
24. Устройство управления сигналом синхронизации в многопоточном процессоре, содержащее первое средство для поддержания индикации ожидающего обработки события для каждого из множества потоков, поддерживаемых в многопоточном процессоре, второе средство для поддержания индикации активного или неактивного состояния для каждого из множества потоков, поддерживаемых в многопоточном процессоре, и третье средство для обнаружения условия блокировки синхронизации, обозначенного индикацией отсутствия ожидающих обработки событий, для каждого из множества потоков и неактивного состояния для каждого из множества потоков и для отключения сигнала синхронизации для, по меньшей мере, одного функционального блока в многопоточном процессоре в ответ на обнаружение условия блокировки синхронизации.

55

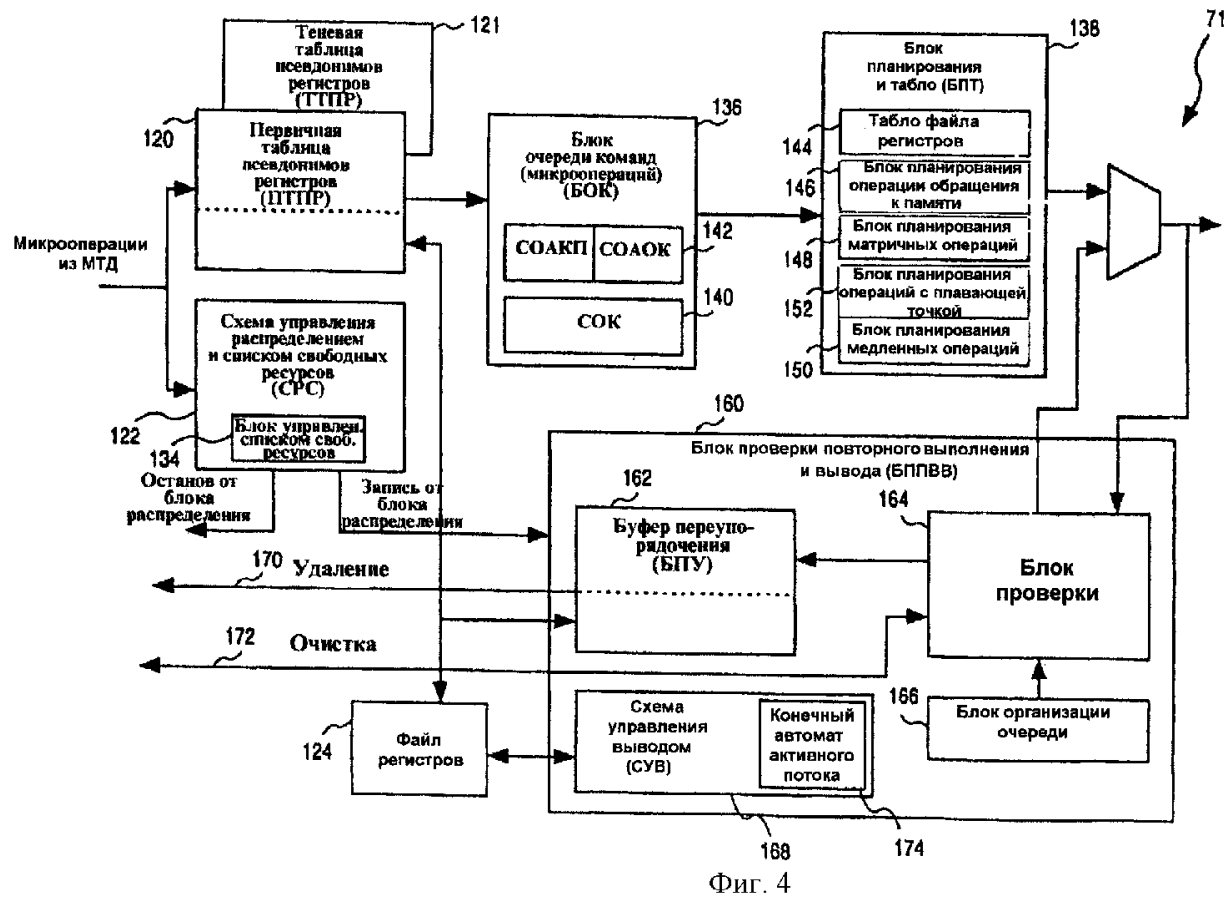
60



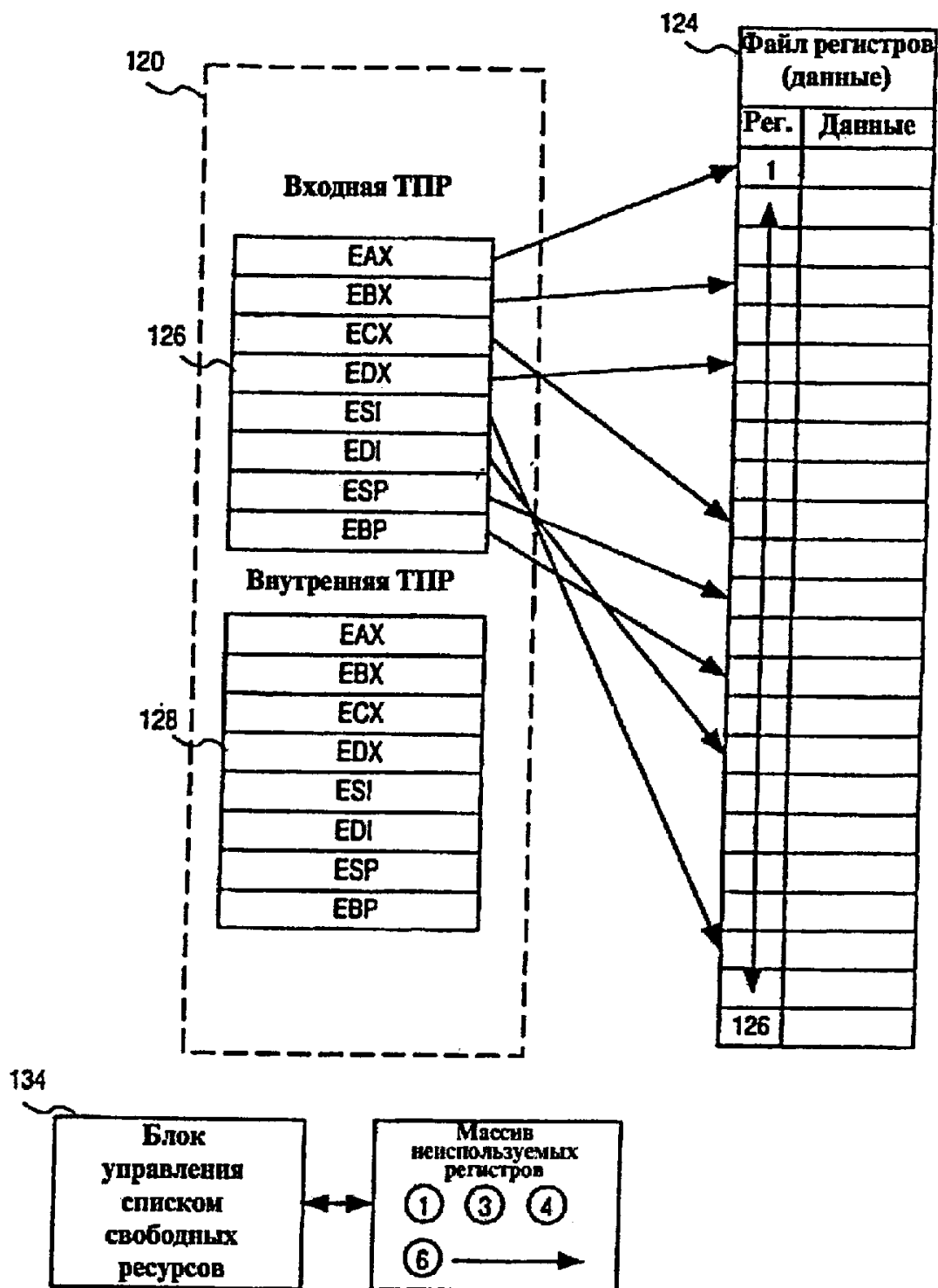
Фиг. 2



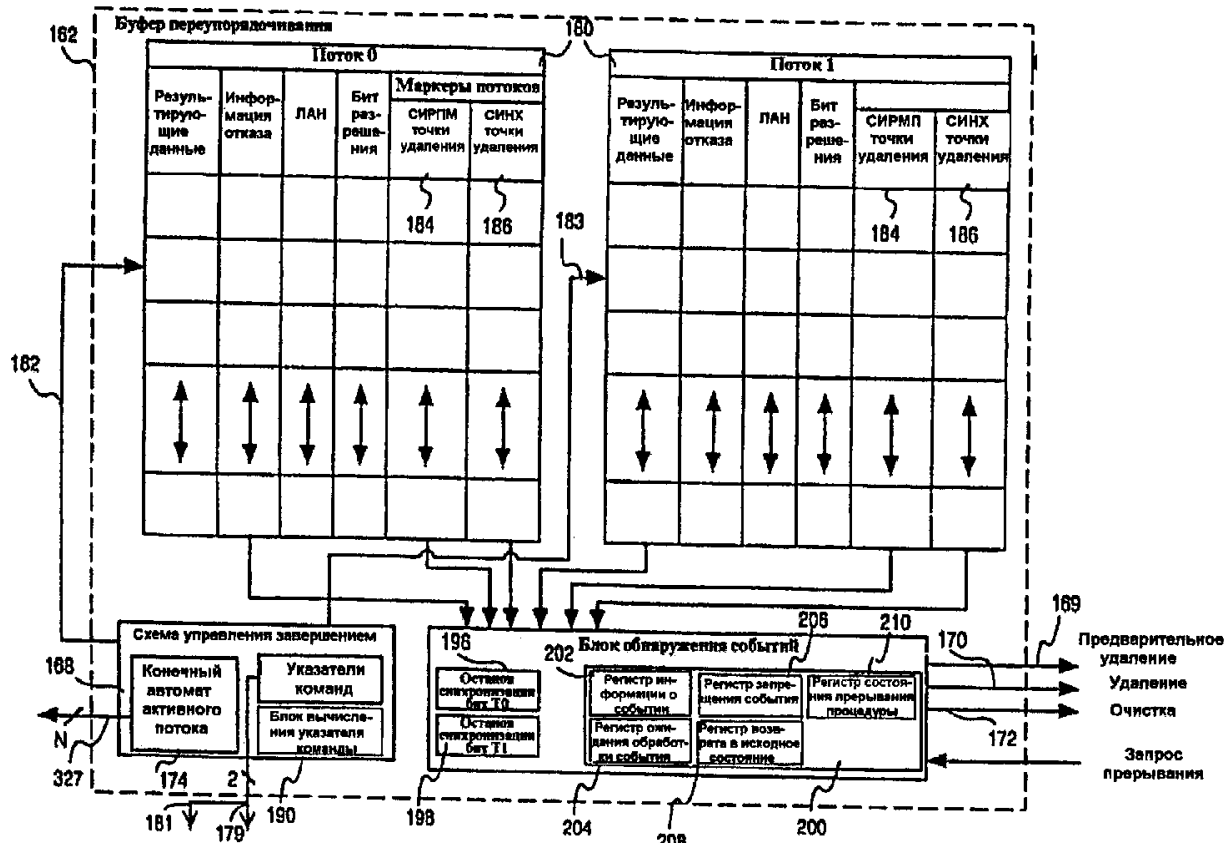
Фиг. 3



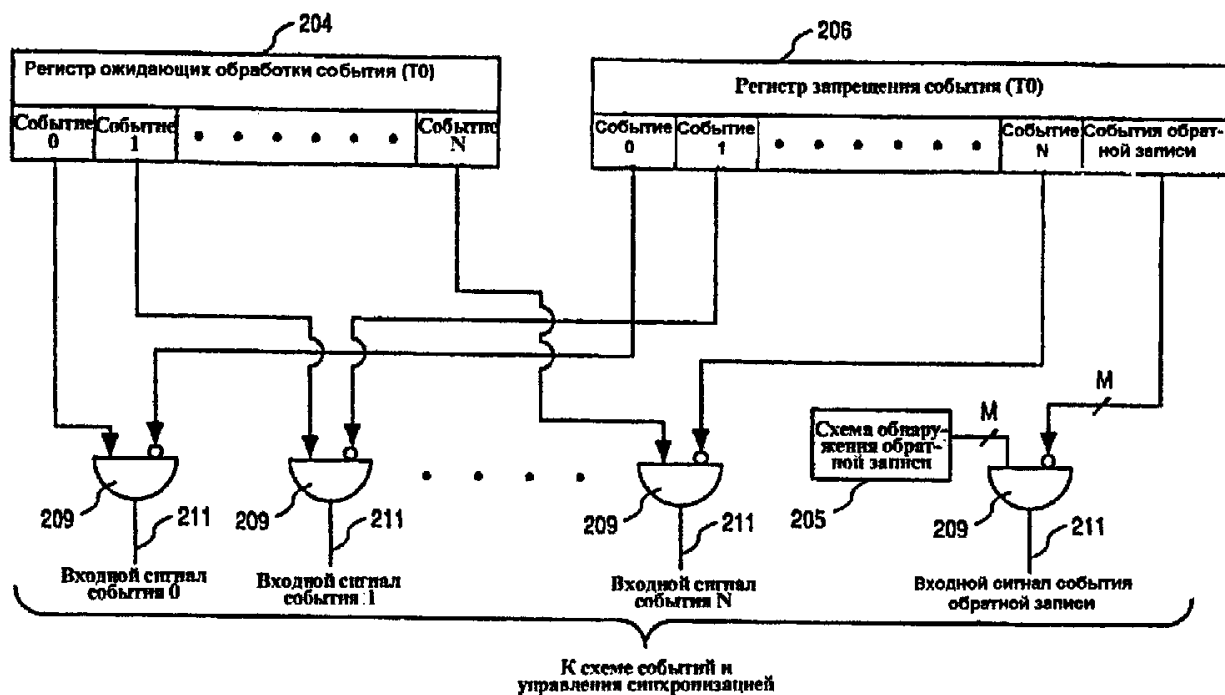
Фиг. 4



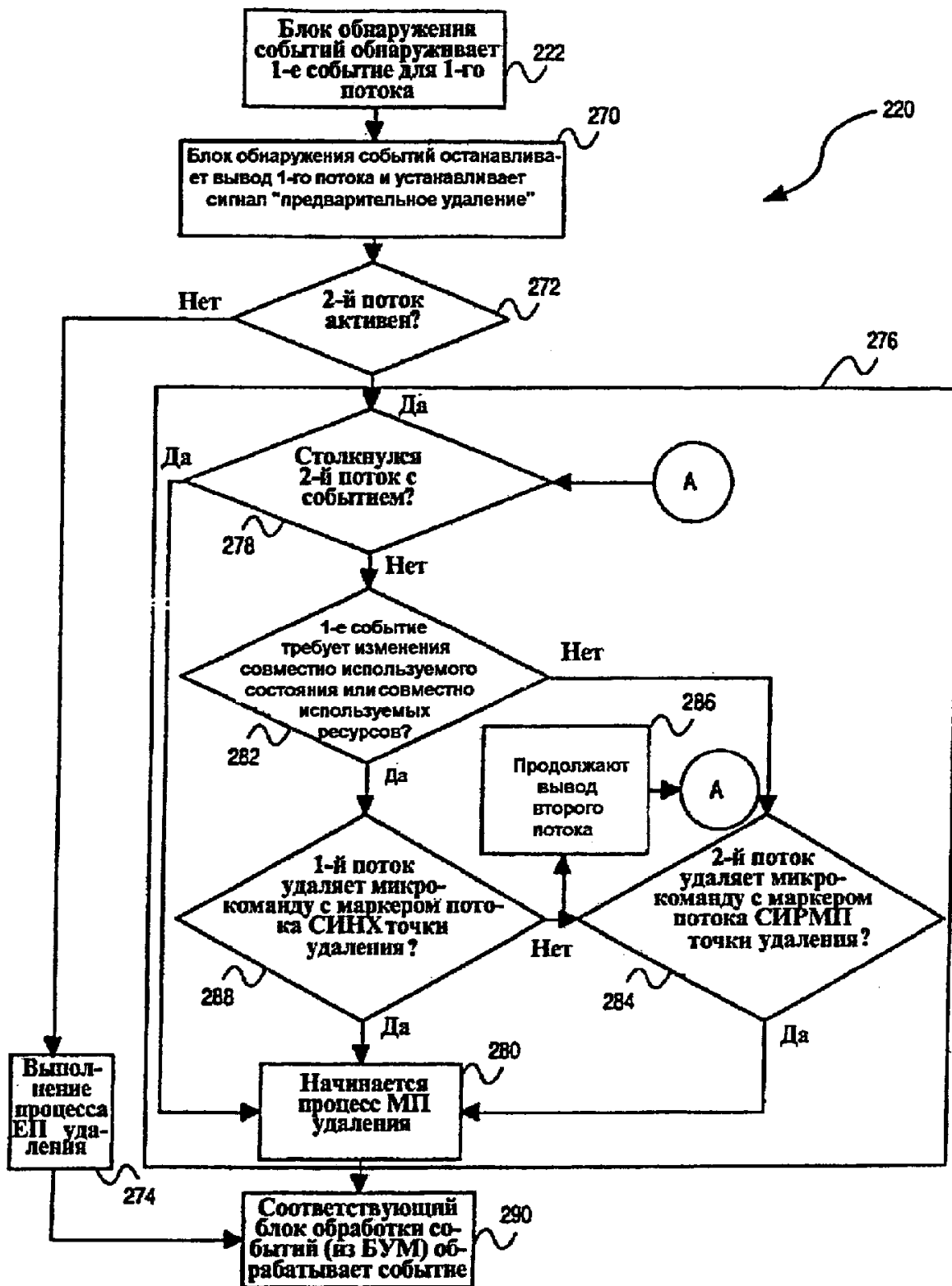
Фиг. 5



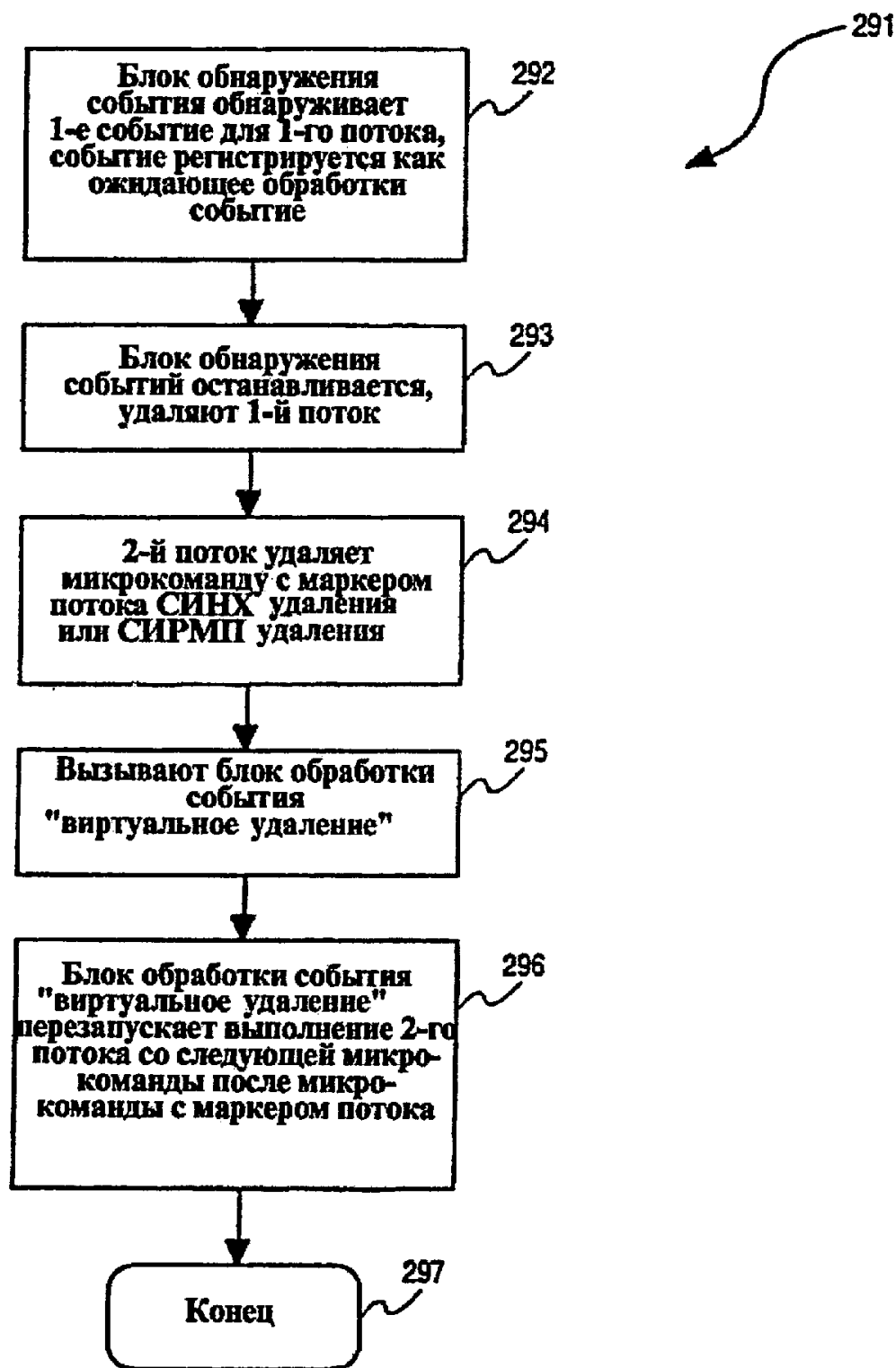
Фиг. 6А



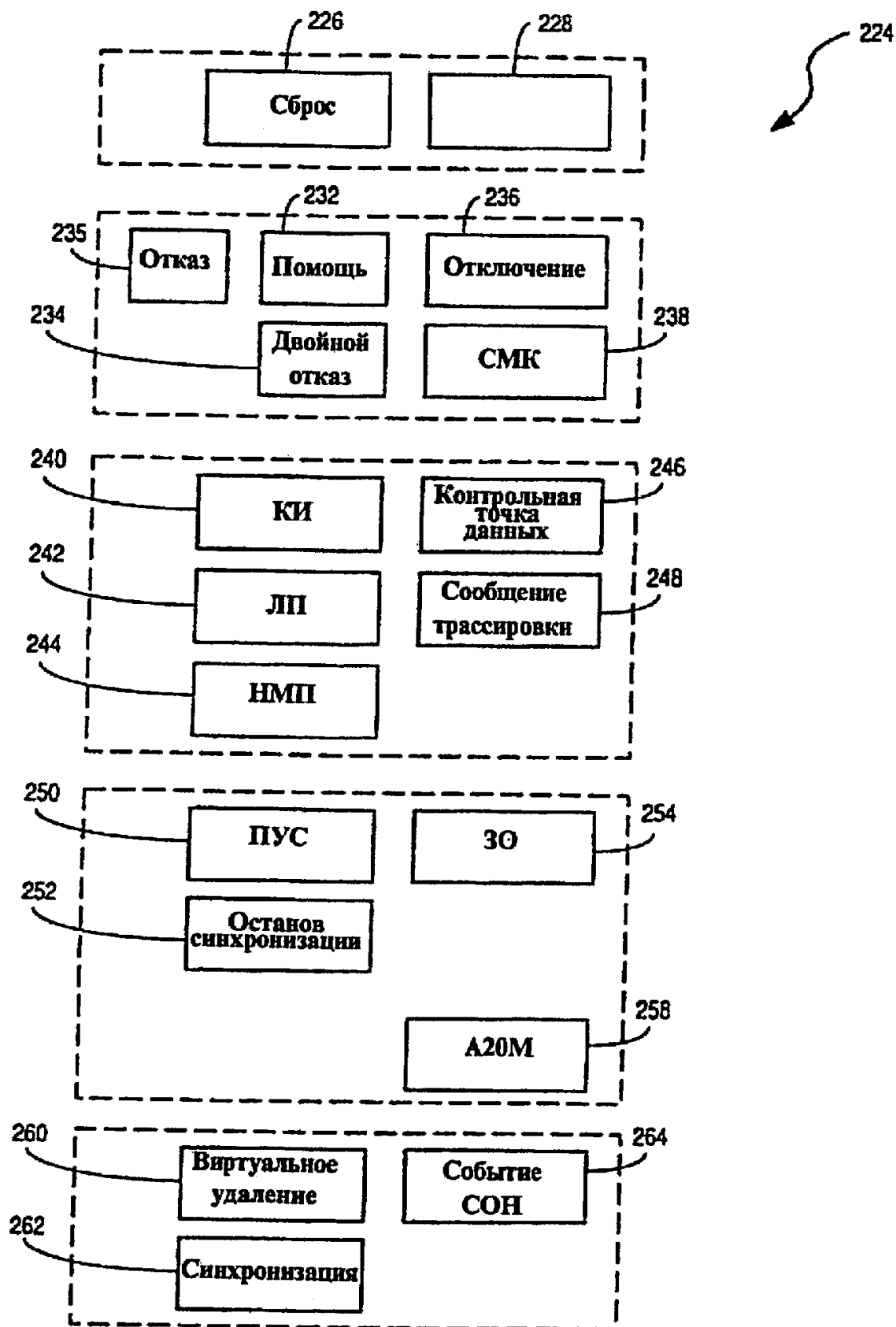
Фиг. 6В



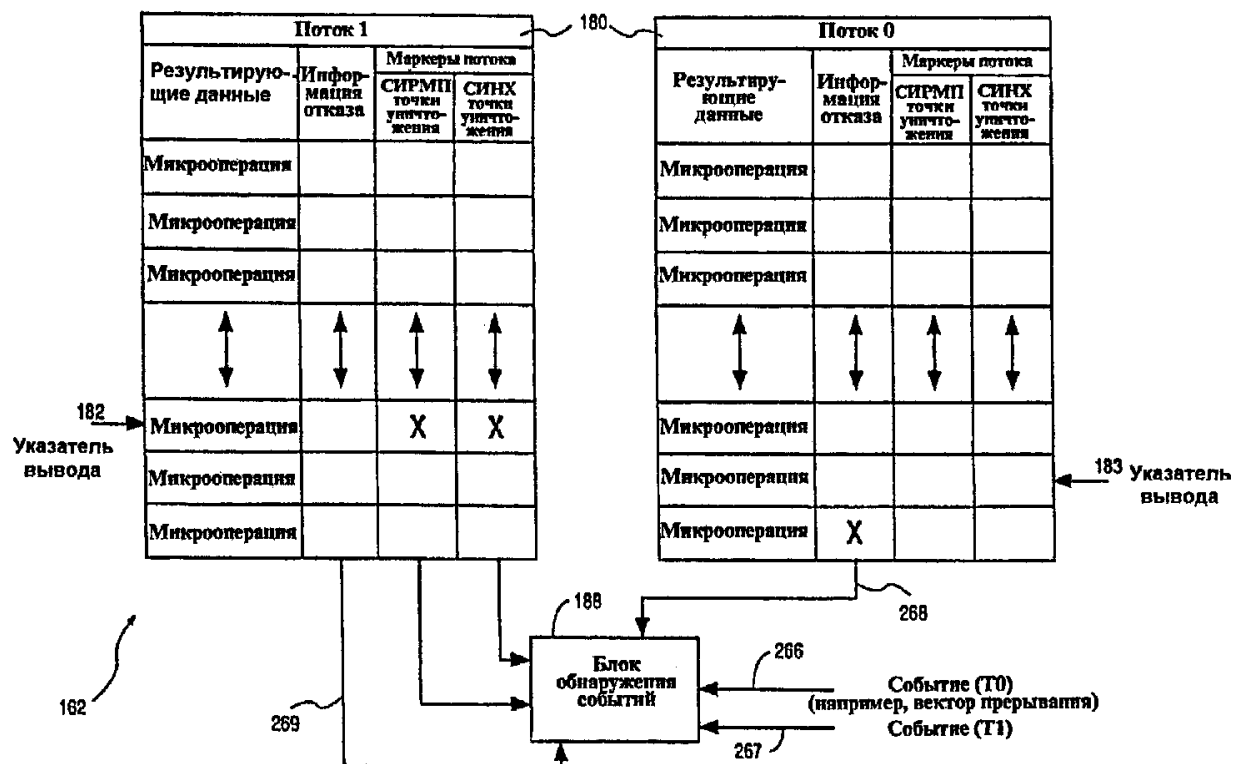
Фиг. 7А



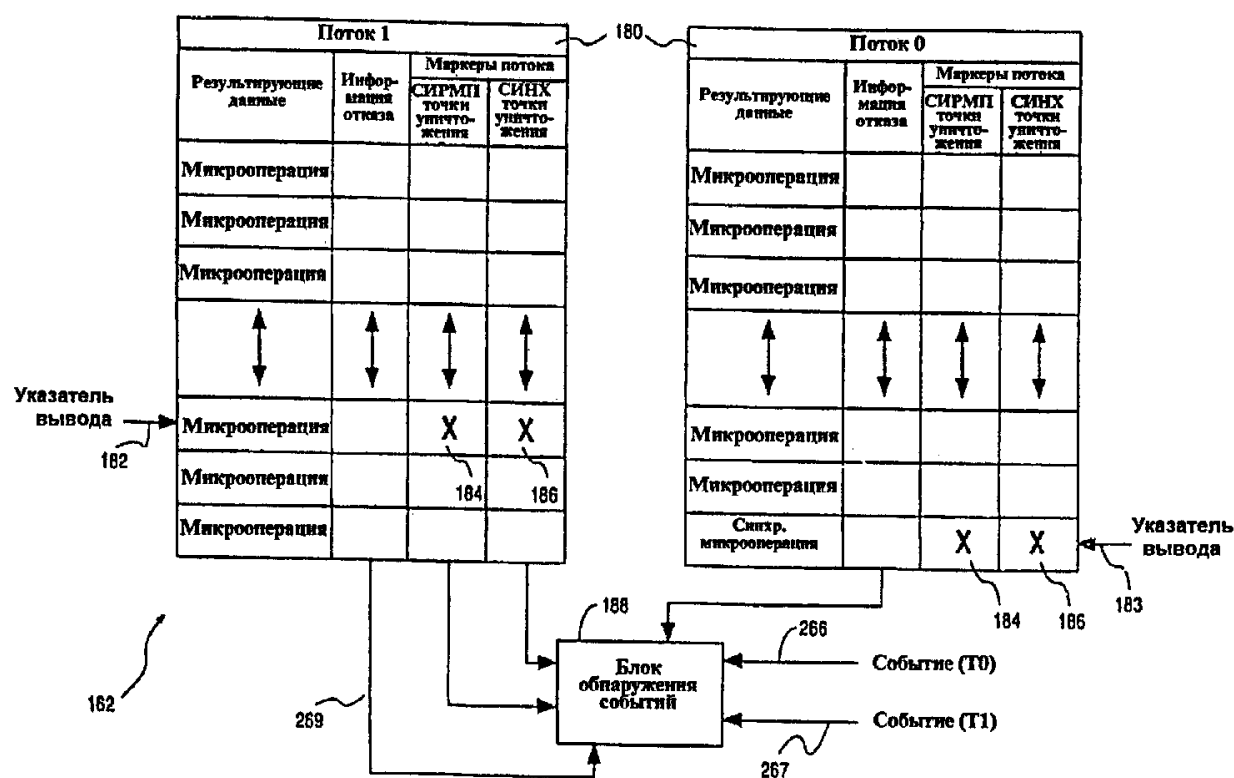
Фиг. 7В



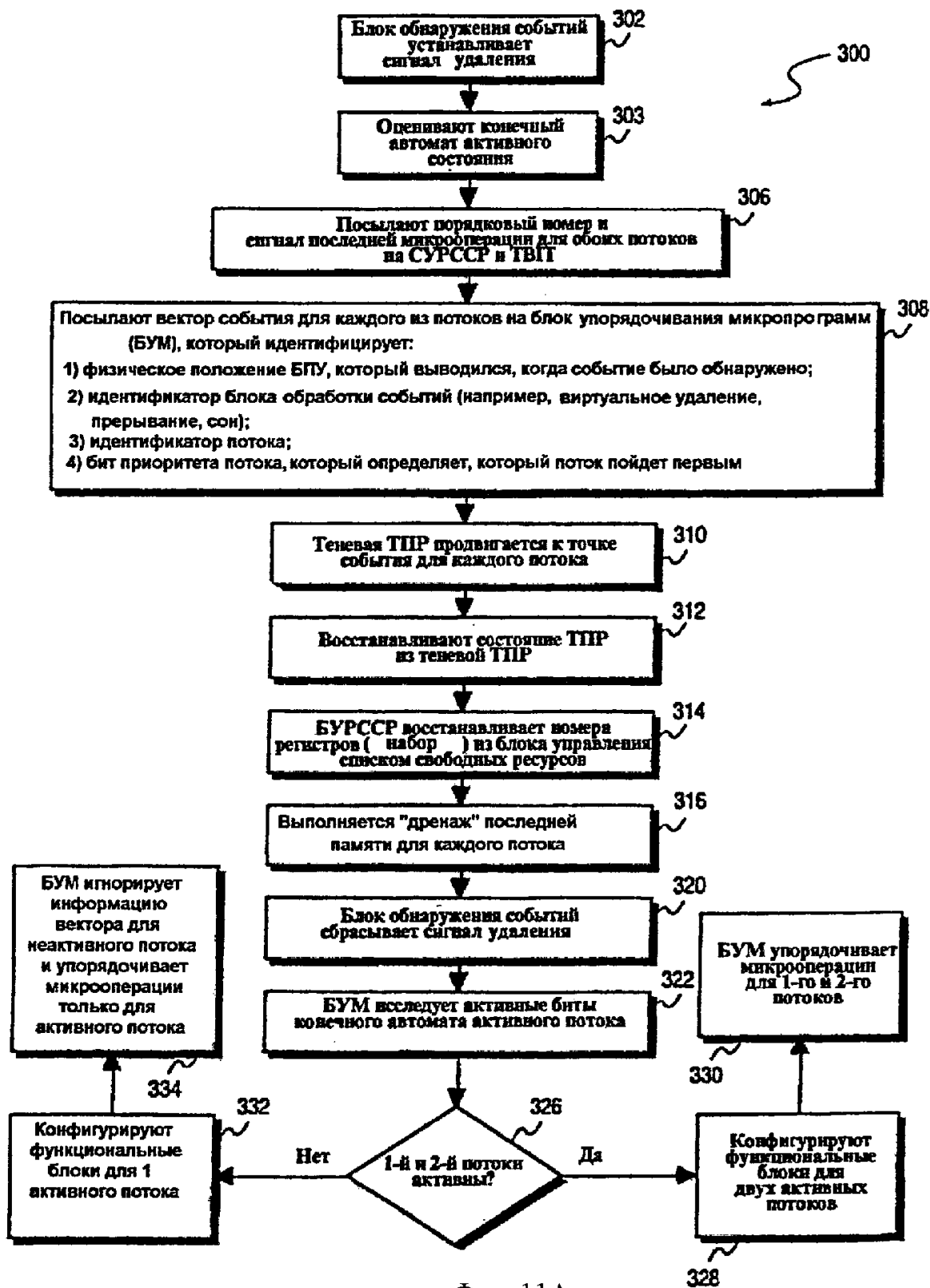
Фиг. 8



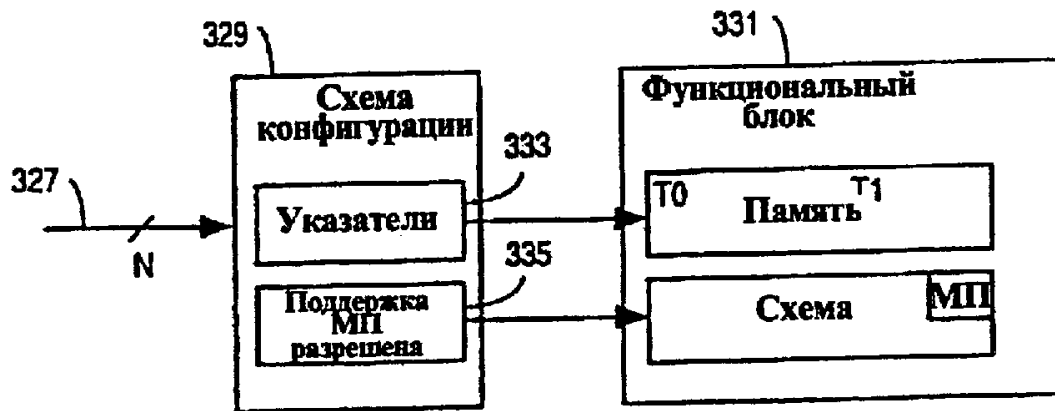
Фиг. 9



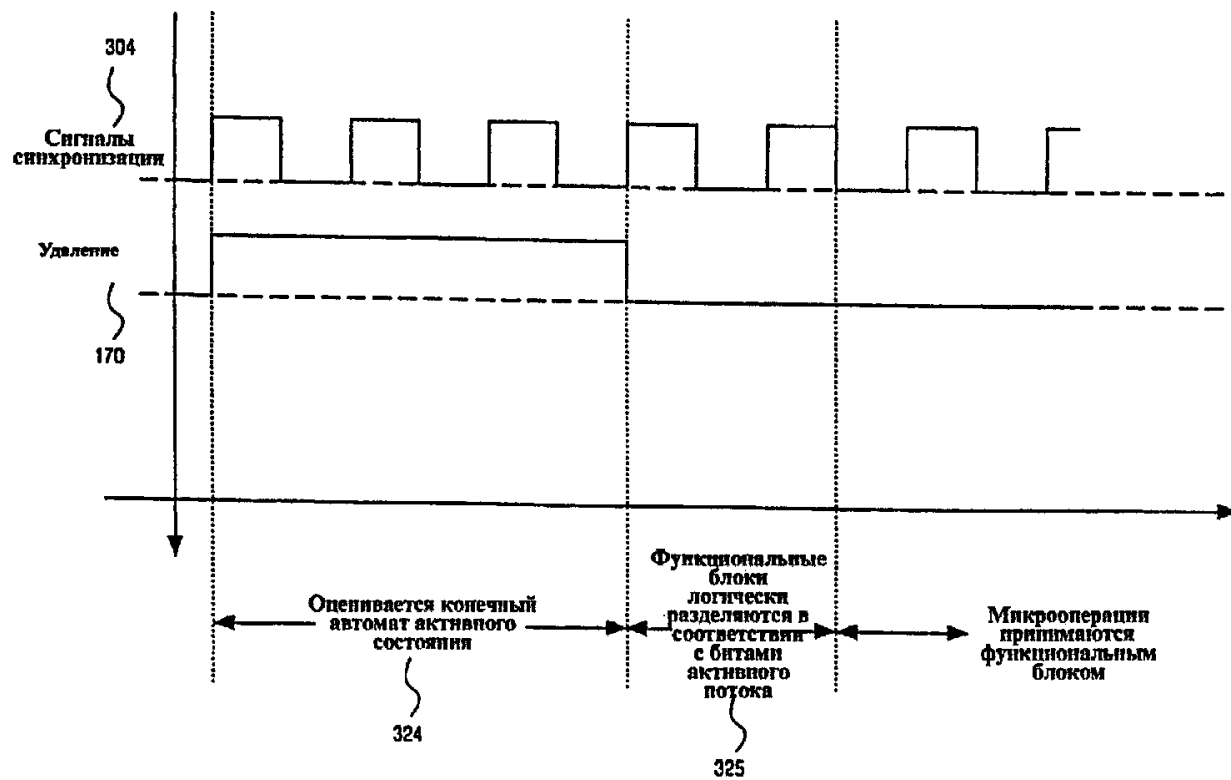
Фиг. 10



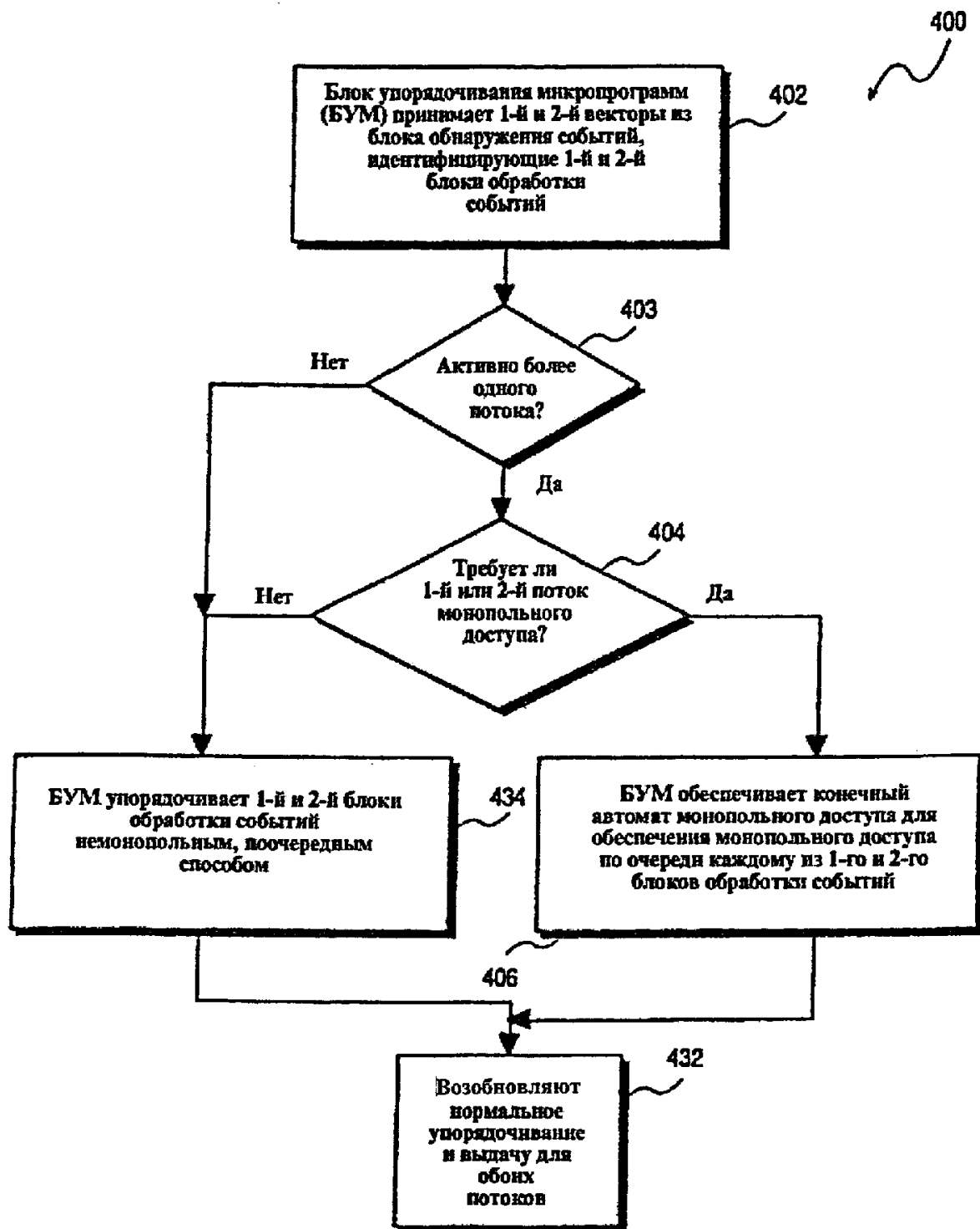
Фиг. 11А



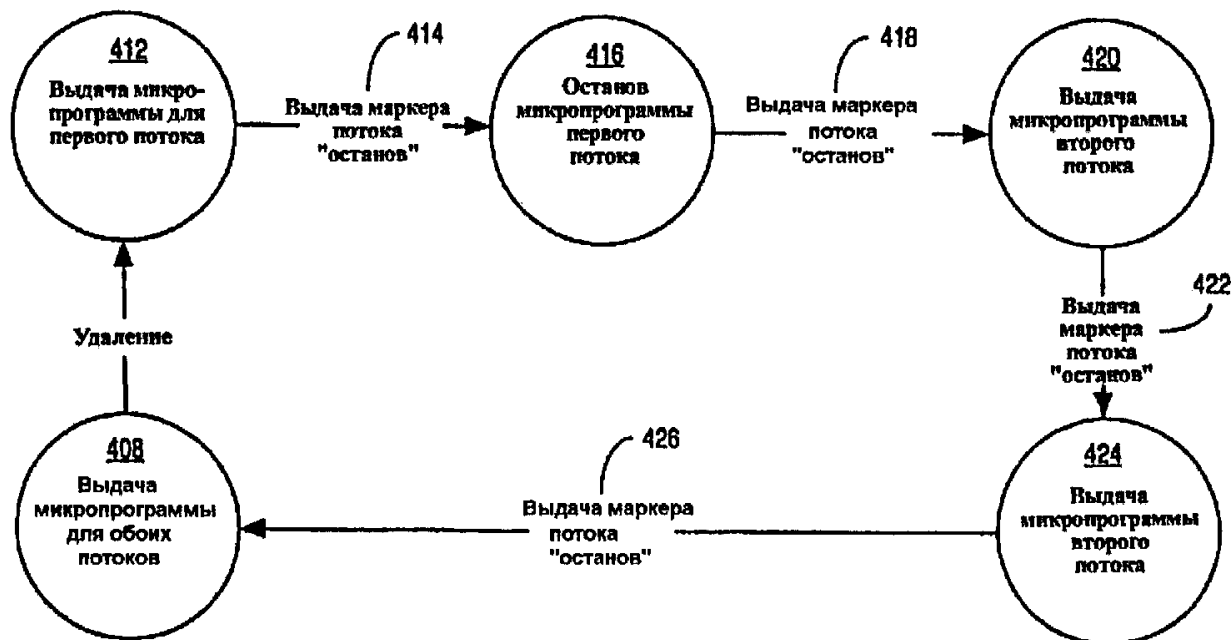
Фиг. 11В



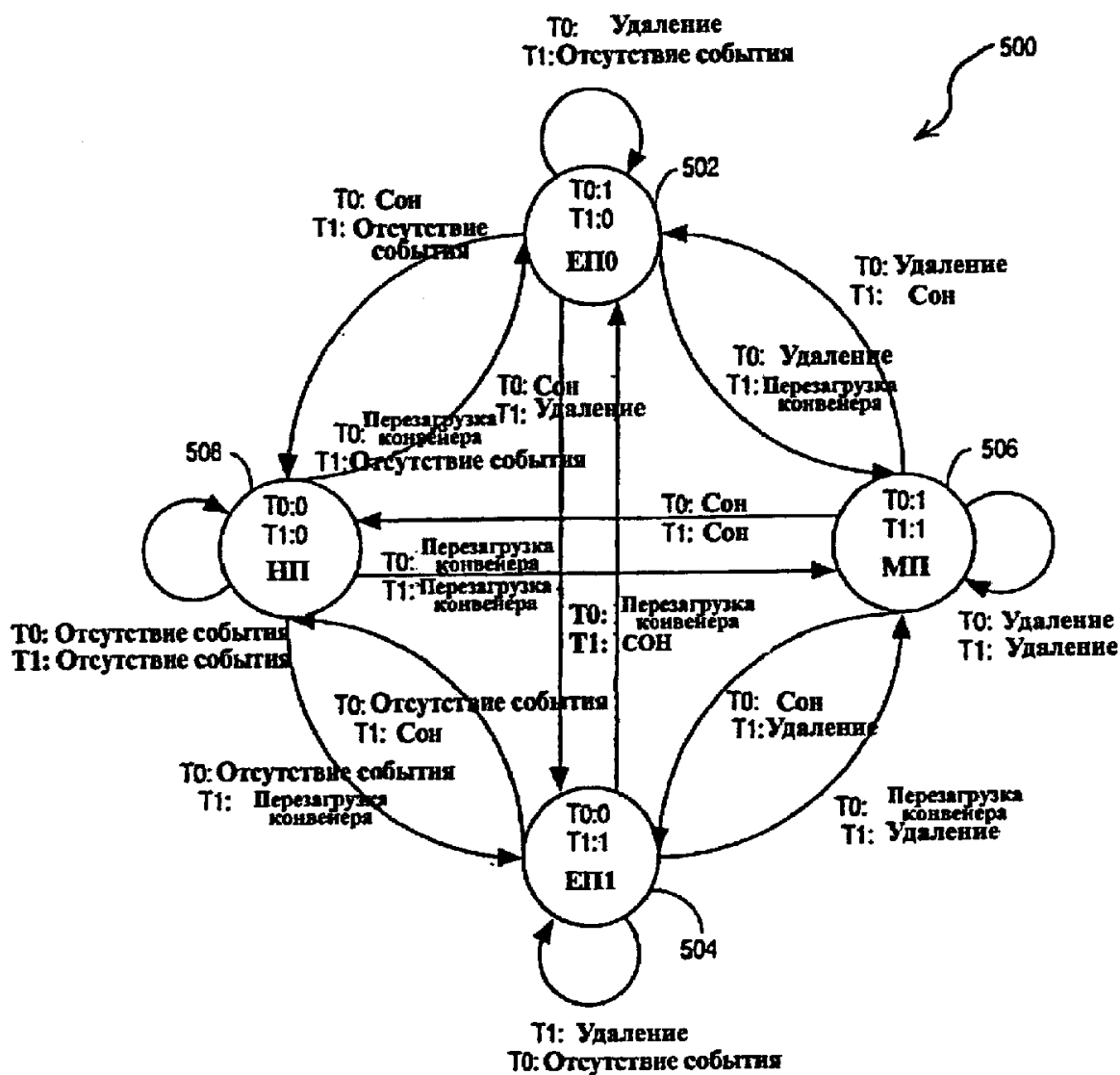
Фиг. 12



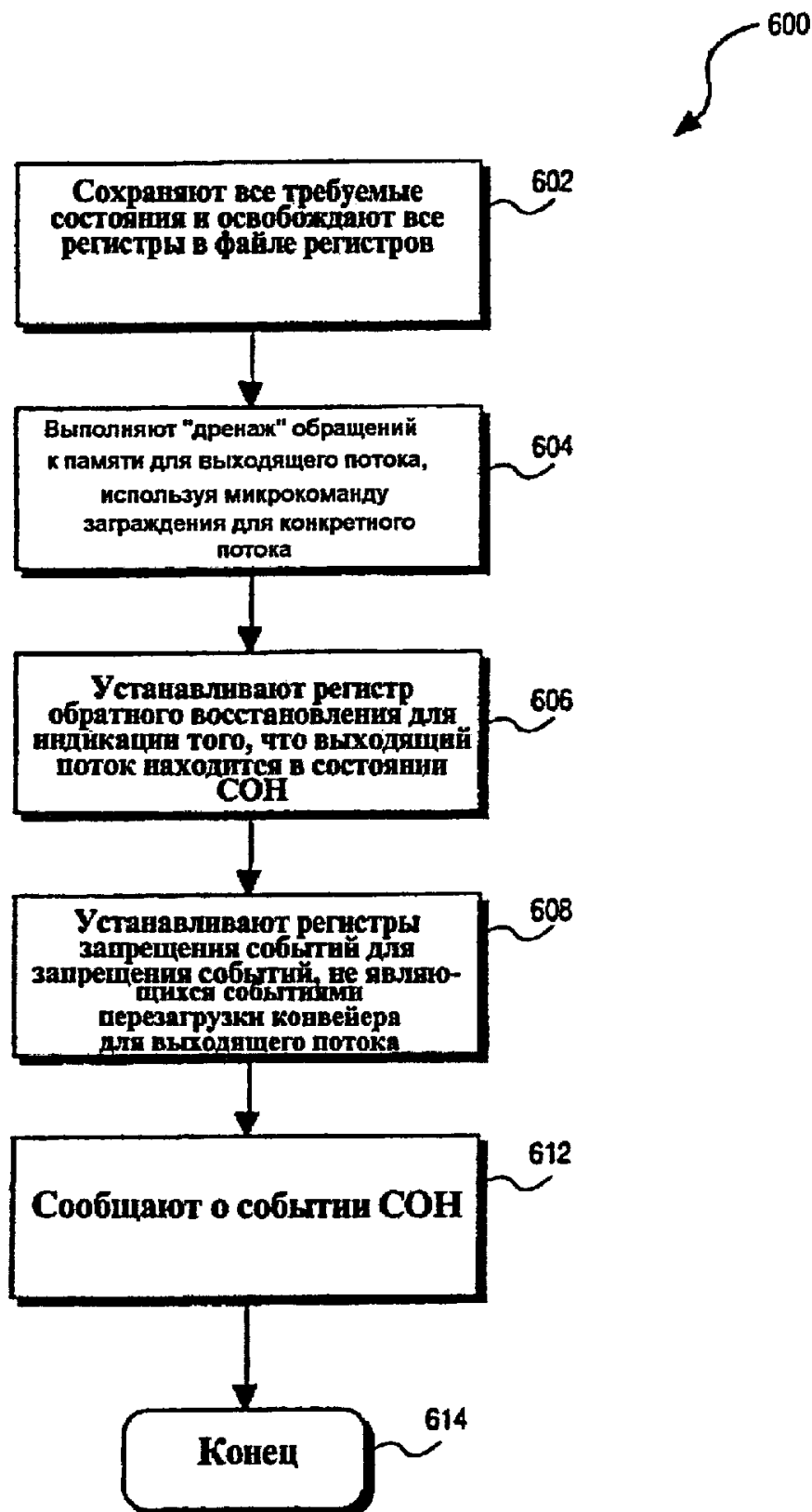
Фиг. 13



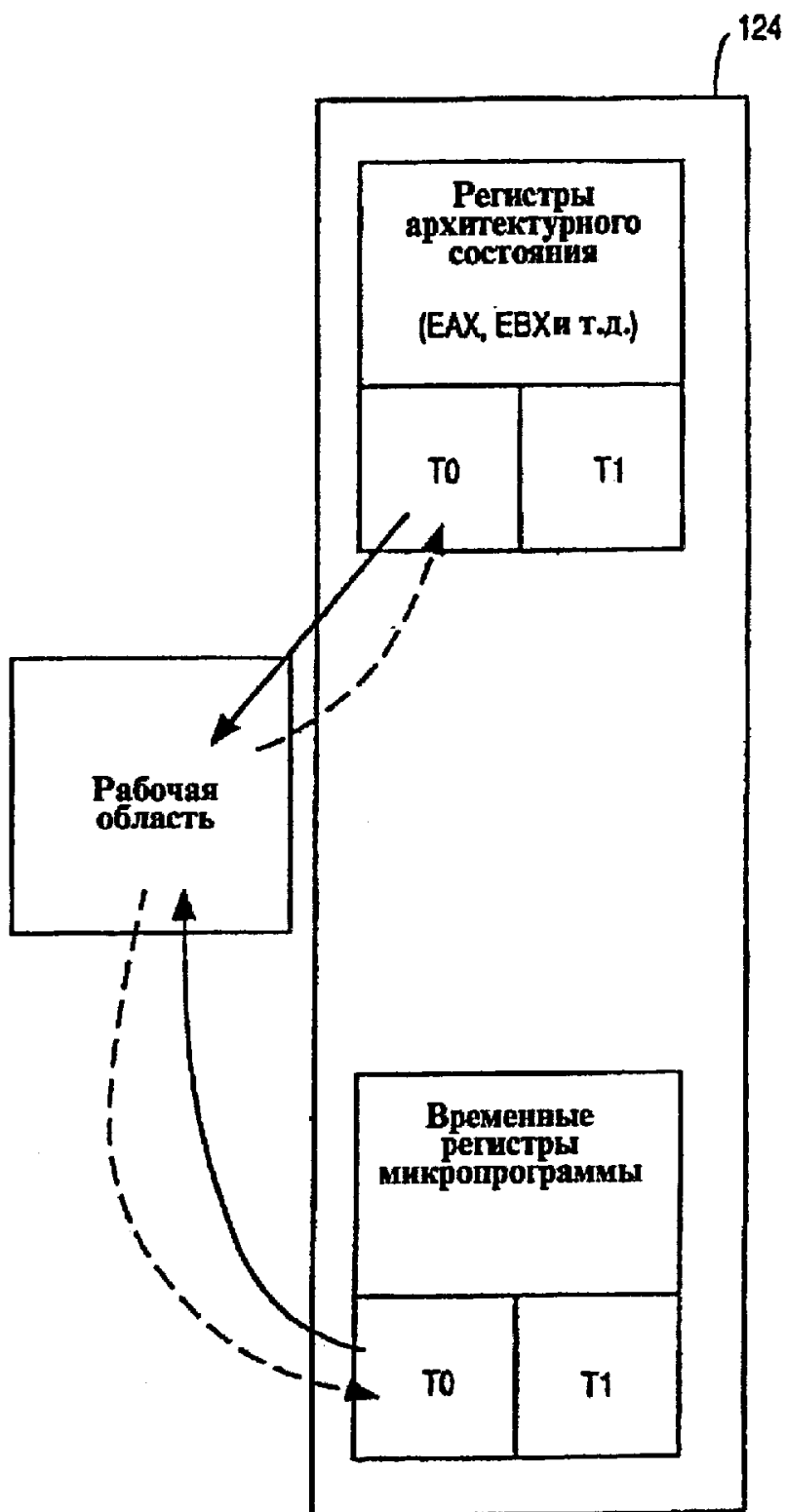
Фиг. 14



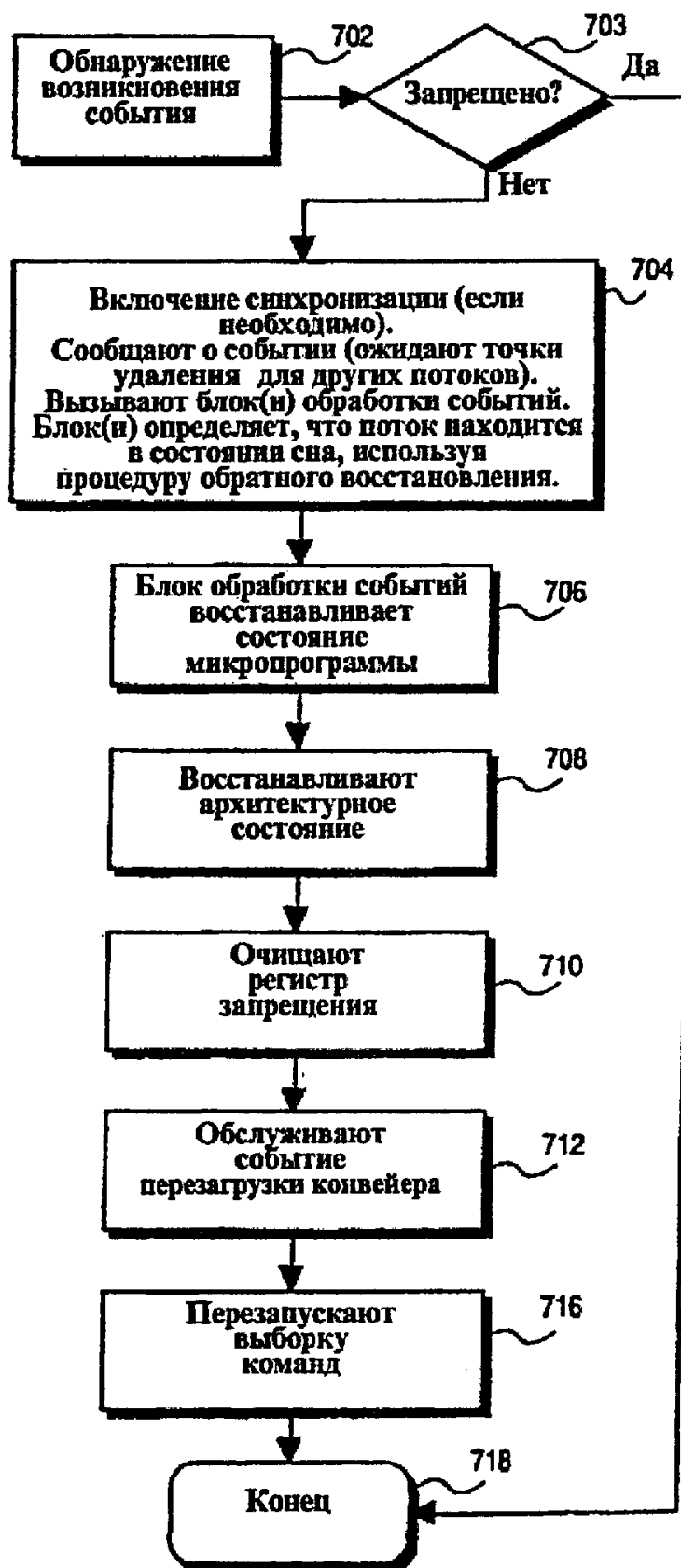
Фиг. 15



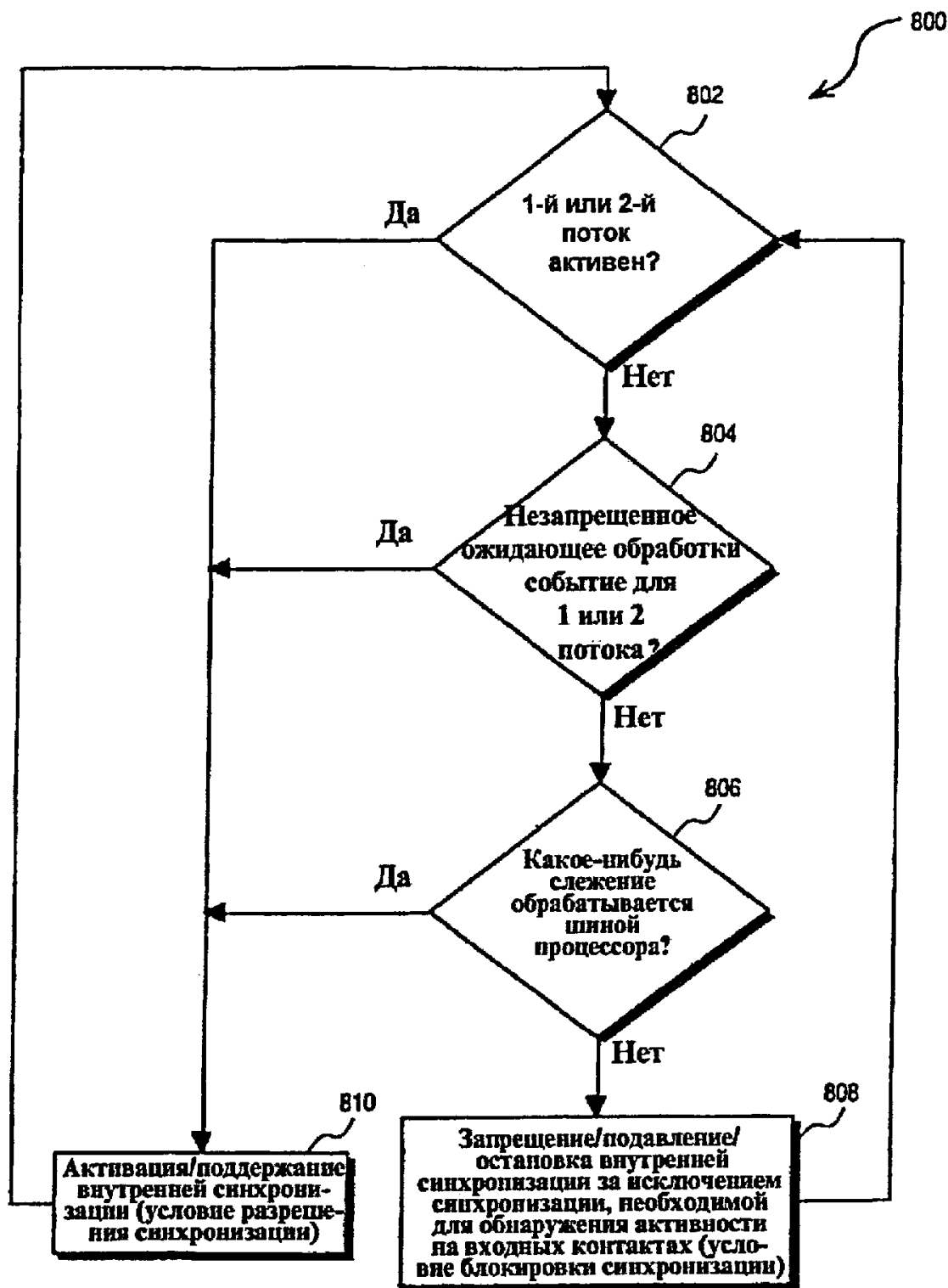
Фиг. 16А



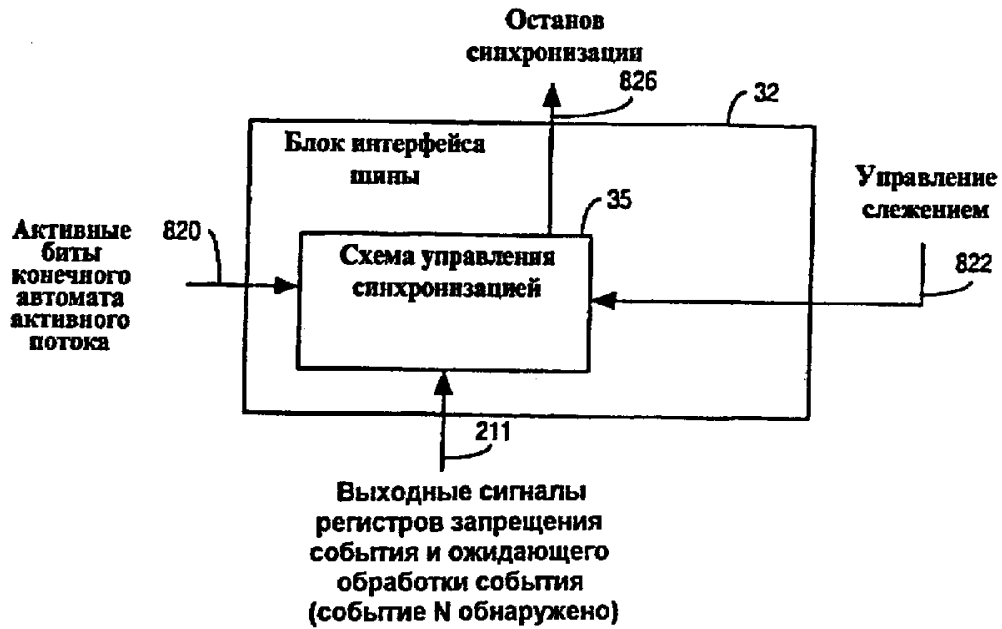
Фиг. 16В



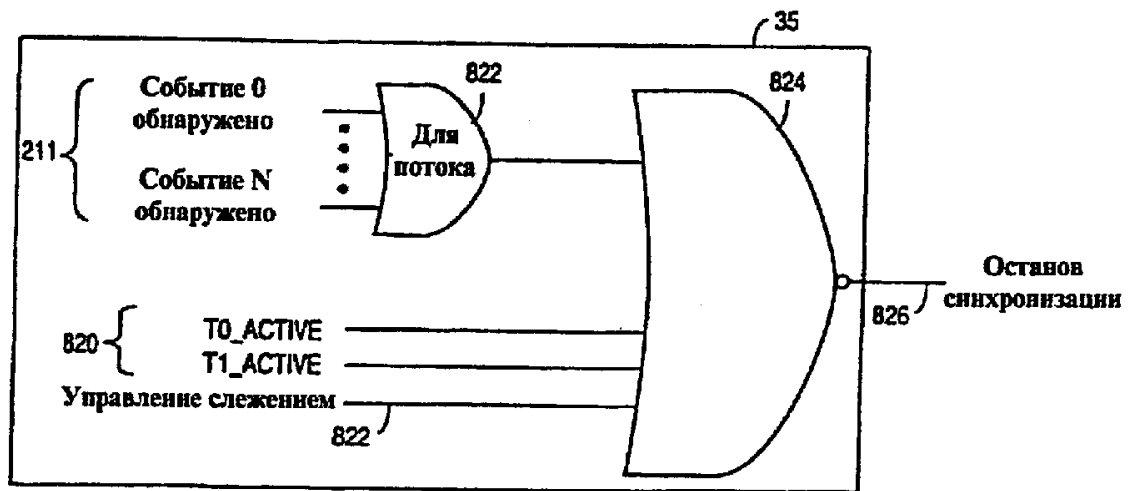
Фиг. 17



Фиг. 18



Фиг. 19А



Фиг. 19В