

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК

H04N 19/12 (2024.01); H04N 19/13 (2024.01); H04N 19/176 (2024.01); H04N 19/18 (2024.01); H04N 19/186 (2024.01); H04N 19/70 (2024.01)

(21)(22) Заявка: 2023130834, 27.04.2022

(24) Дата начала отсчета срока действия патента:
27.04.2022Дата регистрации:
24.10.2024

Приоритет(ы):

(30) Конвенционный приоритет:
28.04.2021 US 63/181,110

(43) Дата публикации заявки: 08.12.2023 Бюл. № 34

(45) Опубликовано: 24.10.2024 Бюл. № 30

(85) Дата начала рассмотрения заявки РСТ на
национальной фазе: 28.11.2023(86) Заявка РСТ:
US 2022/026598 (27.04.2022)(87) Публикация заявки РСТ:
WO 2022/232317 (03.11.2022)

Адрес для переписки:

129090, Москва, ул. Б. Спасская, 25, стр. 3, ООО
"Юридическая фирма Городисский и
Партнеры"

(72) Автор(ы):

ЦЗХУ, Хун-Цзхэн (US),
СЮ, Сяюй (US),
ЧЭНЬ, И-Вэнь (US),
ЧЭНЬ, Вэй (US),
КО, Чэ-Вэй (CN),
ВАН, Сянлинь (US),
ЮЙ, Бин (CN)

(73) Патентообладатель(и):

БЭЙЦЗИН ДАЦЗЯ ИНТЕРНЕТ
ИНФОРМЕЙШН ТЕКНОЛОДЖИ КО.,
ЛТД. (CN)(56) Список документов, цитированных в отчете
о поиске: US 2021105491 A1, 2021.04.08. WO
2021067278 A1, 2021.04.08. WO 2020228673 A1,
2020.11.19. US 9918105 B2, 2018.03.13. CN
112118456 A, 2020.12.22. WO 2021040319 A1,
2021.03.04. US 2020404040 A1, 2020.12.24. US
9936200 B2, 2018.04.03. US 10021419 B2,
2018.07.10. RU 2643655 C2, 2018.02.02. US
2018115787 A1, 2018.04.26.(54) ОСТАТОЧНОЕ КОДИРОВАНИЕ И КОДИРОВАНИЕ КОЭФФИЦИЕНТОВ ДЛЯ КОДИРОВАНИЯ
ВИДЕО

(57) Реферат:

Изобретение относится к кодированию и сжатию видео. Техническим результатом является повышение эффективности кодирования видео. Результат достигается тем, что в ответ на определение, что пропуск преобразования деактивирован, выполняют деактивацию, посредством декодера, присутствия параметра

Райса для остаточного кодирования с пропуском преобразования. Способ дополнительно может включать в себя прием (внедрение) флага пропуска преобразования, `sps_transform_skip_enabled_flag`, чтобы обуславливать присутствие `sps_ts_residual_coding_rice_present_in_sh_flag`. 3 н. и 9 з.п. ф-лы, 41 ил., 70 табл.

В ответ на определение того, что пропуск преобразования деактивируется, деактивация, посредством декодера, присутствия параметра Райса для остаточного кодирования с пропуском преобразования

2702



Адаптация флага пропуска преобразования, `sps_transform_skip_enabled_flag`, чтобы преобразовывать и согласовывать присутствие `sps_ts_residual_coding_rice_present_in_sh_flag`

2704

ФИГ. 27



FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(12) **ABSTRACT OF INVENTION**

(52) CPC

H04N 19/12 (2024.01); H04N 19/13 (2024.01); H04N 19/176 (2024.01); H04N 19/18 (2024.01); H04N 19/186 (2024.01); H04N 19/70 (2024.01)

(21)(22) Application: **2023130834, 27.04.2022**(24) Effective date for property rights:
27.04.2022Registration date:
24.10.2024

Priority:

(30) Convention priority:
28.04.2021 US 63/181,110(43) Application published: **08.12.2023 Bull. № 34**(45) Date of publication: **24.10.2024 Bull. № 30**(85) Commencement of national phase: **28.11.2023**(86) PCT application:
US 2022/026598 (27.04.2022)(87) PCT publication:
WO 2022/232317 (03.11.2022)Mail address:
**129090, Moskva, ul. B. Spasskaya, 25, str. 3, OOO
"Yuridicheskaya firma Gorodisskij i Partnery"**

(72) Inventor(s):

**JHU, Hong-Jheng (US),
XIU, Xiaoyu (US),
CHEN, Yi-Wen (US),
CHEN, Wei (US),
KUO, Che-Wei (CN),
WANG, Xianglin (US),
YU, Bing (CN)**

(73) Proprietor(s):

**BEIJING DAJIA INTERNET INFORMATION
TECHNOLOGY CO., LTD. (CN)**

(54) **RESIDUAL CODING AND COEFFICIENT CODING FOR VIDEO CODING**

(57) Abstract:

FIELD: video encoding and compression.

SUBSTANCE: result is achieved by the fact that in response to determination that conversion skipping is deactivated, deactivation, by means of a decoder, of the presence of the Rice parameter for residual coding with conversion skipping is performed. Method may further

include receiving (embedding) a transformation skip flag, `sps_transform_skip_enabled_flag`, to cause presence `sps_ts_residual_coding_rice_present_in_sh_flag`.

EFFECT: high efficiency of video encoding.
12 cl, 41 dwg, 70 tbl

В ответ на определение того, что пропуск преобразования деактивируется, деактивация, посредством декодера, присутствия параметра Райса для остаточного кодирования с пропуском преобразования

2702



Адаптация флага пропуска преобразования, `sps_transform_skip_enabled_flag`, чтобы преобразовывать и согласовывать присутствие `sps_ts_residual_coding_rice_present_in_sh_flag`

2704

ФИГ. 27

Перекрестная ссылка на родственную заявку

[0001] Данная заявка основана на и испрашивает приоритет предварительной заявки на патент номер 63/181110, поданной 28 апреля 2021 года, содержимое которой полностью содержится в данном документе по ссылке для всех целей.

5 Область техники, к которой относится изобретение

[0002] Данное раскрытие относится к кодированию (кодировке) и сжатию видео. Более конкретно, данное раскрытие относится к улучшениям и упрощениям остаточного кодирования (кодировки) и (кодировке) кодирования коэффициентов для кодирования (кодировки) видео.

10 Уровень техники

[0003] Различные технологии кодирования (coding) видео могут использоваться для того, чтобы сжимать видеоданные. Кодирование видео выполняется согласно одному или более стандартов кодирования видео. Например, стандарты кодирования видео включают в себя стандарт универсального кодирования видео (VVC), стандарт объединенной исследовательской группы по тестовой модели (JEM), стандарт высокоэффективного кодирования видео (H.265/HEVC), стандарт усовершенствованного кодирования видео (H.264/AVC), кодирование по стандарту Экспертной группы по движущимся картинкам (MPEG) и т.п. Кодирование (coding) видео, в общем, использует способы прогнозирования (например, взаимное прогнозирование, внутреннее
15 прогнозирование и т.п.), которые используют преимущество избыточности, присутствующей в видеоизображениях или последовательностях. Важная цель технологий кодирования видео состоит в том, чтобы сжимать видеоданные в форму, которая использует более низкую скорость передачи битов при недопущении или минимизации ухудшения качества видео.

25 Сущность изобретения

[0004] Примеры настоящего раскрытия предоставляют способы и оборудование для кодирования видео.

[0005] Согласно первому аспекту настоящего раскрытия, предоставляется способ для кодирования видео. Способ может включать в себя: в ответ на определение того,
30 что пропуск преобразования деактивируется, деактивацию, посредством декодера, присутствия параметра Райса для остаточного кодирования с пропуском преобразования.

[0006] Согласно второму аспекту настоящего раскрытия, предоставляется способ для кодирования видео. Способ может включать в себя: прием, посредством декодера,
35 флага активации согласования с набором параметров последовательности (SPS), который указывает то, согласуется или нет индекс, `ivlCurrRange`, до обходного декодирования синтаксических элементов `sb_coded_flag`, `abs_remainder`, `dec_abs_level` и `coeff_sign_flag` на основе значения активации SPS-согласования.

[0007] Согласно третьему аспекту настоящего раскрытия, предоставляется способ
40 для кодирования видео. Способ может включать в себя: прием, посредством декодера, флага обработки с расширенной точностью, который указывает, принимается (внедряется) или нет расширенный динамический диапазон для коэффициентов преобразования и во время обработки преобразования на основе значения флага обработки с расширенной точностью.

45 [0008] Согласно четвертому аспекту настоящего раскрытия, предоставляется способ для кодирования видео. Способ может включать в себя: прием, посредством декодера, флага активации постоянной адаптации Райса, который указывает, инициализируется или нет извлечение параметра Райса для преобразования в двоичную форму `abs_remaining`

и `dec_abs_level` в начале каждого субблока, принимая зависимую от режима статистику, накапливаемую из предыдущих субблоков, на основе значения флага активации постоянной адаптации Райса.

[0009] Следует понимать, что вышеприведенные общие описания и нижеприведенные
5 подробные описания являются только примерными и пояснительными и не имеют намерение ограничивать настоящее раскрытие.

Краткое описание чертежей

[0010] Прилагаемые чертежи, которые содержатся и составляют часть этого
10 подробного описания, иллюстрируют примеры в соответствии с настоящим раскрытием и наряду с описанием служат для того, чтобы пояснять принципы раскрытия.

[0011] Фиг. 1 является блок-схемой кодера, согласно примеру настоящего раскрытия.

[0012] Фиг. 2 является блок-схемой декодера, согласно примеру настоящего раскрытия.

[0013] Фиг. 3А является схемой, иллюстрирующей блочные сегменты в многотипной
15 древовидной структуре, согласно примеру настоящего раскрытия.

[0014] Фиг. 3В является схемой, иллюстрирующей блочные сегменты в многотипной древовидной структуре, согласно примеру настоящего раскрытия.

[0015] Фиг. 3С является схемой, блочные сегменты в многотипной древовидной структуре, согласно примеру настоящего раскрытия.

[0016] Фиг. 3D является схемой, иллюстрирующей блочные сегменты в многотипной
20 древовидной структуре, согласно примеру настоящего раскрытия.

[0017] Фиг. 3Е является схемой, иллюстрирующей блочные сегменты в многотипной древовидной структуре, согласно примеру настоящего раскрытия.

[0018] Фиг. 4 является иллюстрацией на схеме картинки с СТУ сигналов яркости 18
25 на 12, согласно примеру настоящего раскрытия.

[0019] Фиг. 5 является иллюстрацией картинки с СТУ сигналов яркости 18 на 12, согласно примеру настоящего раскрытия.

[0020] Фиг. 6А является иллюстрацией примера запрещенной сегментации на троичное
30 дерево (ТТ) и двоичное дерево (ВТ) в VTM, согласно примеру настоящего раскрытия.

[0021] Фиг. 6В является иллюстрацией примера запрещенной ТТ- и ВТ-сегментации
в VTM, согласно примеру настоящего раскрытия.

[0022] Фиг. 6С является иллюстрацией примера запрещенной ТТ- и ВТ-сегментации
в VTM, согласно примеру настоящего раскрытия.

[0023] Фиг. 6D является иллюстрацией примера запрещенной ТТ- и ВТ-сегментации
35 в VTM, согласно примеру настоящего раскрытия.

[0024] Фиг. 6Е является иллюстрацией примера запрещенной ТТ- и ВТ-сегментации
в VTM, согласно примеру настоящего раскрытия.

[0025] Фиг. 6F является иллюстрацией примера запрещенной ТТ- и ВТ-сегментации
в VTM, согласно примеру настоящего раскрытия.

[0026] Фиг. 6G является иллюстрацией примера запрещенной ТТ- и ВТ-сегментации
40 в VTM, согласно примеру настоящего раскрытия.

[0027] Фиг. 6Н является иллюстрацией примера запрещенной ТТ- и ВТ-сегментации
в VTM, согласно примеру настоящего раскрытия.

[0028] Фиг. 7 является иллюстрацией структуры остаточного кодирования для блоков
45 преобразования, согласно примеру настоящего раскрытия.

[0029] Фиг. 8 является иллюстрацией структуры остаточного кодирования для блоков с пропуском преобразования, согласно примеру настоящего раскрытия.

[0030] Фиг. 9 является иллюстрацией двух скалярных квантователей, согласно примеру

настоящего раскрытия.

[0031] Фиг. 10А является иллюстрацией перехода состояния, согласно примеру настоящего раскрытия.

5 [0032] Фиг. 10В является иллюстрацией выбора квантователя, согласно примеру настоящего раскрытия.

[0033] Фиг. 11 является иллюстрацией эталона, используемого для выбора вероятностных моделей, согласно настоящему раскрытию.

[0034] Фиг. 12 является иллюстрацией примера блока, кодированного в палитровом режиме, согласно настоящему раскрытию.

10 [0035] Фиг. 13 является иллюстрацией использования предиктора палитры для того, чтобы сигнализировать записи палитры, согласно настоящему раскрытию.

[0036] Фиг. 14А является иллюстрацией горизонтального пересекающего сканирования, согласно настоящему раскрытию.

15 [0037] Фиг. 14В является иллюстрацией вертикального пересекающего сканирования, согласно настоящему раскрытию.

[0038] Фиг. 15А является иллюстрацией сканирования карты индексов на основе субблоков для палитры, согласно настоящему раскрытию.

[0039] Фиг. 15В является иллюстрацией сканирования карты индексов на основе субблоков для палитры, согласно настоящему раскрытию.

20 [0040] Фиг. 16 является способом для кодирования видеосигнала, согласно примеру настоящего раскрытия.

[0041] Фиг. 17 является способом для кодирования видеосигнала, согласно примеру настоящего раскрытия.

25 [0042] Фиг. 18 является схемой, иллюстрирующей вычислительное окружение, соединенное с пользовательским интерфейсом, согласно примеру настоящего раскрытия.

[0043] Фиг. 19 иллюстрирует способ для кодирования видео, согласно примеру настоящего раскрытия.

[0044] Фиг. 20 иллюстрирует способ для кодирования видео, согласно примеру настоящего раскрытия.

30 [0045] Фиг. 21 иллюстрирует способ для кодирования видео, согласно примеру настоящего раскрытия.

[0046] Фиг. 22 иллюстрирует способ для кодирования видео, согласно примеру настоящего раскрытия.

35 [0047] Фиг. 23 является блок-схемой, иллюстрирующей примерную систему для кодирования и декодирования видеоблоков согласно примеру настоящего раскрытия.

[0048] Фиг. 24 является блок-схемой, иллюстрирующей примерный видеокодер согласно примеру настоящего раскрытия.

[0049] Фиг. 25 является блок-схемой, иллюстрирующей примерный видеodeкодер согласно примеру настоящего раскрытия.

40 [0050] Фиг. 26 иллюстрирует способ остаточного кодирования с пропуском преобразования (TSRC) с низкой задержкой согласно примеру настоящего раскрытия.

[0051] Фиг. 27 иллюстрирует способ для кодирования видео, согласно примеру настоящего раскрытия.

Подробное описание изобретения

45 [0052] Далее приводится подробное описание примерных вариантов осуществления, примеры которых проиллюстрированы на прилагаемых чертежах. Нижеприведенное описание ссылается на прилагаемые чертежи, при этом одинаковые номера на различных чертежах представляют одинаковые или аналогичные элементы, если не указано иное.

Реализации, изложенные в нижеприведенном описании примерных вариантов осуществления, не представляют все реализации в соответствии с раскрытием. Вместо этого, они являются просто примерами устройств и способов в соответствии с аспектами, связанными с раскрытием, изложенным в прилагаемой формуле изобретения.

5 [0053] Терминология, используемая в настоящем раскрытии, служит только для целей описания конкретных вариантов осуществления и не имеет намерение ограничивать настоящее раскрытие. При использовании в настоящем раскрытии и в прилагаемой формуле изобретения, формы единственного числа "a", "an" и "the" имеют намерение включать в себя также формы множественного числа, если контекст явно
10 не указывает иное. Также следует понимать, что термин "и/или", используемый в данном документе, имеет намерение обозначать и включать в себя любые возможные комбинации одного или более ассоциированных перечисленных элементов.

[0054] Следует понимать, что, хотя термины "первый", "второй", "третий", и т.д. могут использоваться в данном документе, чтобы описывать различную информацию, информация не должна быть ограничена посредством этих терминов. Эти термины
15 используются только для того, чтобы отличать одну категорию информации от другой. Например, без отступления от объема настоящего раскрытия, первая информация может называться "второй информацией"; и аналогично, вторая информация также может называться "первой информацией". При использовании в данном документе,
20 термин "если" может пониматься как означающий "когда" или "при" либо "в ответ на проведение оценки", в зависимости от контекста.

[0055] Первая версия HEVC-стандарта завершена в октябре 2013, причем она предлагает приблизительно 50%-ю экономию по скорости передачи битов или эквивалентное перцепционное качество по сравнению с H.264/MPEG AVC-стандартом
25 кодирования видео предшествующего поколения. Хотя HEVC-стандарт предоставляет существенные улучшения в отношении кодирования по сравнению с предшественником, имеется доказательство того, что превосходная эффективность кодирования может достигаться с дополнительными средствами кодирования по сравнению с HEVC. На основе этого, как VCEG, так и MPEG начинают исследовательские работы новых
30 технологий кодирования для будущей стандартизации в области кодирования видео. Одна Объединенная исследовательская группа по видеостандартам (JVET) сформирована в октябре 2015 посредством ITU-T VCEG и ISO/IEC MPEG для того, чтобы начать важное исследование передовых технологий, которые могут обеспечивать существенное повышение эффективности кодирования. Одно опорное программное
35 обеспечение, называемое "стандартом объединенной исследовательской группы по тестовой модели (JEM)", поддерживается посредством JVET в силу интегрирования нескольких дополнительных средств кодирования поверх тестовой HEVC-модели (HM).

[0056] В октябре 2017 года, объединенный конкурс заявок (CfP) по сжатию видео с возможностями за рамками HEVC проведен посредством ITU-T и ISO/IEC. В апреле
40 2018 года, 23 CfP-ответа приняты и оценены на 10-й JVET-конференции, которые демонстрируют прирост эффективности сжатия по сравнению с HEVC приблизительно в 40%. На основе таких результатов оценки, JVET запускает новый проект по разработке стандарта кодирования видео нового поколения, который называется "стандартом универсального кодирования видео (VVC)". В этом же месяце, одна кодовая база
45 опорного программного обеспечения, называемая "тестовой VVC-моделью (VTM)", установлена для демонстрации опорной реализации VVC-стандарта.

[0057] Аналогично HEVC, VVC основывается на инфраструктуре гибридного кодирования видео на основе блоков.

[0058] Фиг. 1 показывает общую схему видеокодера на основе блоков для VVC. В частности, фиг. 1 показывает типичный кодер 100. Кодер 100 имеет видеоввод 110, компенсацию 112 движения, оценку 114 движения, решение 116 по выбору внутреннего/взаимного режима, модуль 140 блочного прогнозирования, сумматор 128,
 5 преобразование 130, квантование 132, связанную с прогнозированием информацию 142, внутреннее прогнозирование 118, буфер 120 картинок, обратное квантование 134, обратное преобразование 136, сумматор 126, запоминающее устройство 124, внутриконтурный фильтр 122, энтропийное кодирование 138 и поток 144 битов.

[0059] В кодере 100, видеокадр сегментируется на множество видеоблоков для
 10 обработки. Для каждого данного видеоблока, прогнозирование формируется на базе либо подхода на основе взаимного прогнозирования, либо подхода на основе внутреннего прогнозирования.

[0060] Остаток прогнозирования, представляющий разность между текущим видеоблоком, частью видеоввода 110 и его предиктором, частью модуля 140 блочного
 15 прогнозирования, отправляется в преобразование 130 из сумматора 128. Коэффициенты преобразования затем отправляются из преобразования 130 в квантование 132 для уменьшения энтропии. Квантованные коэффициенты затем подаются в энтропийное кодирование 138 для того, чтобы формировать сжатый поток видеобитов. Как показано на фиг. 1, связанная с прогнозированием информация 142 из решения 116 по выбору
 20 внутреннего/взаимного режима, такая как информация сегментации на видеоблоки, векторы движения (MV), индекс опорной картинке и режим внутреннего прогнозирования также подаются через энтропийное кодирование 138 и сохраняются в сжатый поток 144 битов. Сжатый поток 144 битов включает в себя поток видеобитов.

[0061] В кодере 100, связанные с декодером схемы также требуются для того, чтобы
 25 восстанавливать пиксели для целей прогнозирования. Во-первых, остаток прогнозирования восстанавливается через обратное квантование 134 и обратное преобразование 136. Этот восстановленный остаток прогнозирования комбинируется с модулем 140 блочного прогнозирования для того, чтобы формировать нефильтрованные восстановленные пиксели для текущего видеоблока.

[0062] Пространственное прогнозирование (или "внутреннее прогнозирование")
 30 использует пиксели из выборок уже кодированных соседних блоков (которые называются "опорными выборками") в одном и том же видеокадре с текущим видеоблоком для того, чтобы прогнозировать текущий видеоблок.

[0063] Временное прогнозирование (также называемое "взаимным
 35 прогнозированием") использует восстановленные пиксели из уже кодированных видеокартинок для того, чтобы прогнозировать текущий видеоблок. Временное прогнозирование уменьшает временную избыточность, внутренне присущую в видеосигнале. Сигнал временного прогнозирования для данной единицы кодирования (CU) или блока кодирования обычно сигнализируется посредством одного или более
 40 MV, которые указывают величину и направление движения между текущей CU и ее временным ориентиром. Дополнительно, если несколько опорных картинок поддерживаются, дополнительно отправляется один индекс опорной картинке, которая используется для того, чтобы идентифицировать, из какой опорной картинке в устройстве хранения опорных картинок поступает сигнал временного прогнозирования.

[0064] Оценка 114 движения впускает видеоввод 110 и сигнал из буфера 120 картинок и выводит, в компенсацию 112 движения, сигнал оценки движения. Компенсация 112
 45 движения впускает видеоввод 110, сигнал из буфера 120 картинок и сигнал оценки движения из оценки 114 движения и выводит, в решение 116 по выбору внутреннего/

взаимного режима, сигнал компенсации движения.

[0065] После того, как пространственное и/или временное прогнозирование выполняется, решение 116 по выбору внутреннего/взаимного режима в кодере 100 выбирает наилучший режим прогнозирования, например, на основе способа оптимизации искажения в зависимости от скорости передачи. Модуль 140 блочного прогнозирования затем вычитается из текущего видеоблока, и результирующий остаток прогнозирования декоррелируется с использованием преобразования 130 и квантования 132.

Результирующие квантованные остаточные коэффициенты обратно квантуются посредством обратного квантования 134 и обратно преобразуются посредством обратного преобразования 136, чтобы формировать восстановленный остаток, который затем суммируется обратно с блоком прогнозирования для того, чтобы формировать восстановленный сигнал CU. Дополнительная внутриконтурная фильтрация 122, такая как фильтр удаления блочности, дискретизированное адаптивное смещение (SAO) и/или адаптивный внутриконтурный фильтр (ALF), может применяться к восстановленной CU до того, как она помещается в устройство хранения опорных картинок буфера 120 картинок и используется для того, чтобы кодировать будущие видеоблоки. Чтобы формировать выходной поток 144 видеобитов, режим кодирования (взаимный или внутренний), информация режима прогнозирования, информация движения и квантованные остаточные коэффициенты отправляются в модуль 138 энтропийного кодирования с возможностью дополнительно сжиматься и пакетироваться для того, чтобы формировать поток битов.

[0066] Фиг. 1 приводит блок-схему общей гибридной системы кодирования видео на основе блоков. Входной видеосигнал обрабатывается поблочно (называются "единицами кодирования (CU)"). В VTM-1.0, CU может составлять вплоть до 128×128 пикселей.

Однако, в отличие от HEVC, который сегментирует блоки только на основе деревьев квадрантов, в VVC, одна единица дерева кодирования (CTU) разбивается на CU, чтобы адаптироваться к варьированию локальных характеристик на основе quadro-/двоичного/троичного дерева. По определению, блок дерева кодирования (CTB) представляет собой блок NxN выборок для некоторого значения N таким образом, что разделение компонента на CTB представляет собой сегментацию. CTU включает в себя CTB выборок сигналов яркости, два соответствующих CTB выборок сигналов цветности картинки, которая имеет три массива выборок, либо CTB выборок монохромной картинки или картинки, которая кодируется с использованием трех отдельных цветовых плоскостей и синтаксических структур, используемых для того, чтобы кодировать выборки.

Дополнительно, концепция типа с несколькими единицами сегментации в HEVC удаляется, т.е. разделение CU, единицы прогнозирования (PU) и единицы преобразования (TU) более не существует в VVC; вместо этого, каждая CU всегда используется в качестве базовой единицы для прогнозирования и преобразования без дополнительных сегментов. В многотипной древовидной структуре, одна CTU сначала сегментируется посредством структуры в виде дерева квадрантов. После этого, каждый узел-лист дерева квадрантов дополнительно может сегментироваться посредством структуры в виде двоичного и троичного дерева. Как показано на фиг. 3A, 3B, 3C, 3D и 3E, предусмотрено пять типов разбиения, четверичная сегментация, горизонтальная двоичная сегментация, вертикальная двоичная сегментация, горизонтальная троичная сегментация и вертикальная троичная сегментация.

[0067] Фиг. 3A показывает схему, иллюстрирующую блочный четверичный сегмент в многотипной древовидной структуре в соответствии с настоящим раскрытием.

[0068] Фиг. 3B показывает схему, иллюстрирующую блочный вертикальный двоичный

сегмент в многотипной древовидной структуре в соответствии с настоящим раскрытием.

[0069] Фиг. 3С показывает схему, иллюстрирующую блочный горизонтальный двоичный сегмент в многотипной древовидной структуре в соответствии с настоящим раскрытием.

5 [0070] Фиг. 3D показывает схему, иллюстрирующую блочный вертикальный троичный сегмент в многотипной древовидной структуре в соответствии с настоящим раскрытием.

[0071] Фиг. 3Е показывает схему, иллюстрирующую блочный горизонтальный троичный сегмент в многотипной древовидной структуре в соответствии с настоящим раскрытием.

10 [0072] На фиг. 1, пространственное прогнозирование и/или временное прогнозирование может выполняться. Пространственное прогнозирование (или "внутреннее прогнозирование") использует пиксели из выборок уже кодированных соседних блоков (которые называются "опорными выборками") в одном и том же видеокартинке/срезах (слайсе) для того, чтобы прогнозировать текущий видеоблок.

15 Пространственное прогнозирование уменьшает пространственную избыточность, внутренне присущую в видеосигнале. Временное прогнозирование (также называемое "взаимным прогнозированием" или "прогнозированием с компенсацией движения") использует восстановленные пиксели из уже кодированных видеокартинки для того, чтобы прогнозировать текущий видеоблок. Временное прогнозирование уменьшает

20 временную избыточность, внутренне присущую в видеосигнале. Сигнал временного прогнозирования для данной CU обычно сигнализируется посредством одного или более векторов движения (MV), которые указывают величину и направление движения между текущей CU и ее временным ориентиром. Кроме того, если несколько опорных картинок поддерживаются, дополнительно отправляется один индекс опорной картинки,

25 который используется для того, чтобы идентифицировать, из какой опорной картинки в хранилище опорных картинок поступает сигнал временного прогнозирования. После пространственного и/или временного прогнозирования, блок принятия решений по выбору режима в кодере выбирает наилучший режим прогнозирования, например, на основе способа оптимизации искажения в зависимости от скорости передачи. Блок

30 прогнозирования затем вычитается из текущего видеоблока; и остаток прогнозирования декоррелируется с использованием преобразования и квантованный. Квантованные остаточные коэффициенты обратно квантуются и обратно преобразуются, чтобы сформировать восстановленный остаток, который затем суммируется обратно с блоком прогнозирования для того, чтобы сформировать восстановленный сигнал CU.

35 Дополнительная внутриконтурная фильтрация, такая как фильтр удаления блочности, фильтр на основе дискретизированного адаптивного смещения (SAO) и адаптивный внутриконтурный фильтр (ALF), может применяться к восстановленной CU до того, как она помещается в хранилище опорных картинок и используется для того, чтобы кодировать будущие видеоблоки. Чтобы сформировать выходной поток видеобитов,

40 режим кодирования (взаимный или внутренний), информация режима прогнозирования, информация движения и квантованные остаточные коэффициенты отправляются в модуль энтропийного кодирования с возможностью дополнительно сжиматься и пакетироваться для того, чтобы сформировать поток битов.

[0073] Фиг. 2 показывает общую блок-схему видеодекодера для VVC. В частности, фиг. 2 показывает типичную блок-схему декодера 200. Декодер 200 имеет поток 210 битов, энтропийное декодирование 212, обратное квантование 214, обратное

преобразование 216, сумматор 218, выбор 220 внутреннего/взаимного режима, внутреннее прогнозирование 222, запоминающее устройство 230, внутриконтурный

фильтр на основе дискретизированного адаптивного смещения (SAO) и адаптивный внутриконтурный фильтр (ALF), может применяться к восстановленной CU до того, как она помещается в хранилище опорных картинок и используется для того, чтобы кодировать будущие видеоблоки. Чтобы сформировать выходной поток видеобитов,

40 режим кодирования (взаимный или внутренний), информация режима прогнозирования, информация движения и квантованные остаточные коэффициенты отправляются в модуль энтропийного кодирования с возможностью дополнительно сжиматься и пакетироваться для того, чтобы сформировать поток битов.

[0073] Фиг. 2 показывает общую блок-схему видеодекодера для VVC. В частности, фиг. 2 показывает типичную блок-схему декодера 200. Декодер 200 имеет поток 210 битов, энтропийное декодирование 212, обратное квантование 214, обратное преобразование 216, сумматор 218, выбор 220 внутреннего/взаимного режима, внутреннее прогнозирование 222, запоминающее устройство 230, внутриконтурный

фильтр 228, компенсацию 224 движения, буфер 226 картинок, связанную с прогнозированием информации 234 и видеовывод 232.

[0074] Декодер 200 является аналогичным связанной с восстановлением секции, размещающейся в кодере 100 по фиг. 1. В декодере 200, входящий поток видеобитов 210 сначала декодируется через энтропийное декодирование 212, чтобы извлекать уровни квантованных коэффициентов и связанную с прогнозированием информацию. Уровни квантованных коэффициентов затем обрабатываются через обратное квантование 214 и обратное преобразование 216, чтобы получать восстановленный остаток прогнозирования. Механизм блочного прогнозирования, реализованный в модуле 220 выбора внутреннего/взаимного режима, выполнен с возможностью выполнять внутреннее прогнозирование 222 или компенсацию 224 движения, на основе декодированной информации прогнозирования. Набор нефильТРованных восстановленных пикселей получается посредством суммирования восстановленного остатка прогнозирования из обратного преобразования 216 и прогнозирующего вывода, сформированного посредством механизма блочного прогнозирования, с использованием сумматора 218.

[0075] Восстановленный блок дополнительно может проходить через внутриконтурный фильтр 228 до того, как он сохраняется в буфере 226 картинок, который функционирует в качестве хранилища опорных картинок. Восстановленное видео в буфере 226 картинок может отправляться для того, чтобы активировать устройство отображения, а также использоваться для того, чтобы прогнозировать будущие видеоблоки. В ситуациях, когда внутриконтурный фильтр 228 включается, операция фильтрации выполняется для этих восстановленных пикселей, чтобы извлекать конечный восстановленный видеовывод 232.

[0076] Фиг. 2 приводит общую блок-схему видеodeкодера на основе блоков. Поток видеобитов сначала энтропийно декодируется в модуле энтропийного декодирования. Режим кодирования и информация прогнозирования отправляются либо в модуль пространственного прогнозирования (при внутреннем кодировании), либо в модуль временного прогнозирования (при взаимном кодировании) для того, чтобы формировать блок прогнозирования. Остаточные коэффициенты преобразования отправляются в модуль обратного квантования и модуль обратного преобразования для того, чтобы восстанавливать остаточный блок. Блок прогнозирования и остаточный блок затем суммируются. Восстановленный блок дополнительно может проходить через внутриконтурную фильтрацию до того, как он сохраняется в хранилище опорных картинок. Восстановленное видео в хранилище опорных картинок затем отправляется для того, чтобы активировать устройство отображения, а также используется для того, чтобы прогнозировать будущие видеоблоки.

[0077] В общем, базовая схема внутреннего прогнозирования, применяемая в VVC, сохраняется такой же как базовая схема HEVC, за исключением того, что несколько модулей дополнительно расширяются и/или улучшаются, например, режим кодирования на основе внутренних субсегментов (ISP), расширенное внутреннее прогнозирование с широкоугольными внутренними направлениями, позиционно-зависимое комбинирование с внутренним прогнозированием (PDPC) и 4-отводная внутренняя интерполяция.

[0078] *Сегментация картинок, групп плиток (тайлов), плиток и CTU в VVC*

[0079] В VVC, плитка задается как прямоугольная область CTU в конкретном столбце плиток и конкретной строке плиток в картинке. Группа плиток представляет собой группу из целого числа плиток картинки, которые содержатся исключительно в одной

NAL-единице. По существу, концепция группы плиток является такой же как срез, как задано в HEVC. Например, картинки разделяются на группы плиток и плитки. Плитка представляет собой последовательность STU, которые покрывают прямоугольную область картинки. Группа плиток содержит некоторое число плиток картинки.

- 5 Поддерживаются два режима групп плиток, а именно, режим групп плиток на основе растрового сканирования и режим прямоугольных групп плиток. В режиме групп плиток на основе растрового сканирования, группа плиток содержит последовательность плиток при растровом сканировании плиток картинки. В режиме
10 прямоугольных групп плиток, группа плиток содержит некоторое число плиток картинки, которые совместно формируют прямоугольную область картинки. Плитки в прямоугольной группе плиток находятся в порядке растрового сканирования плиток для группы плиток.

[0080] Фиг. 4 показывает пример сегментации на группы плиток на основе растрового сканирования картинки, при которой картинка разделяется на 12 плиток и 3 группы
15 плиток на основе растрового сканирования. Фиг. 4 включает в себя плитки 410, 412, 414, 416 и 418. Каждая плитка имеет 18 STU. Более конкретно, фиг. 4 показывает картинку с STU сигналов яркости 18 на 12, которая сегментируется на 12 плиток и 3 (информативные) группы плиток. Три группы плиток являются следующими (1): первая группа плиток включает в себя плитки 410 и 412, (2) вторая группа плиток включает в
20 себя плитки 414, 416, 418, 420 и 422, и (3) третья группа плиток включает в себя плитки 424, 426, 428, 430 и 432.

[0081] Фиг. 5 показывает пример сегментации на прямоугольные группы плиток картинки, при которой картинка разделяется на 24 плитки (6 столбцов плиток и строки из 4 плиток) и 9 прямоугольных групп плиток. Фиг. 5 включает в себя плитку 510, 512,
25 514, 516, 518, 520, 522, 524, 526, 528, 530, 532, 534, 536, 538, 540, 542, 544, 546, 548, 550, 552, 554 и 556. Более конкретно, фиг. 5 показывает картинку с STU сигналов яркости 18 на 12, которая сегментируется на 24 плитки и 9 групп плиток(информативно). Группа плиток содержит плитки, и плитка содержат STU. 9 прямоугольных групп плиток включают в себя (1) две плитки 510 и 512, (2) две плитки 514 и 516, (3) две плитки 518 и
30 520, (4) четыре плитки 522, 524, 534 и 536, (5) четыре плитки 526, 528, 538 и 540 (6) четыре плитки 530, 532, 542 и 544, (7) две плитки 546 и 548, (8) две плитки 550 и 552 и (9) две плитки 554 и 556.

[0082] *Преобразования с большими размерами блоков с высокочастотным обнулением в VVC*

- 35 [0083] В VTM4, преобразования с большими размерами блоков, вплоть до 64×64 по размеру, активируются, что является главным образом полезным для видео более высокого разрешения, например, для 1080p- и 4K-последовательностей.

Высокочастотные коэффициенты преобразования обнуляются для блоков преобразования с размером (с шириной или высотой либо с шириной и высотой), равным
40 64, так что только более низкие частотные коэффициенты сохраняются. Например, для блока преобразования M×N, с M в качестве ширины блока и N в качестве высоты блока, когда M равен 64, только левые 32 столбца коэффициентов преобразования сохраняются. Аналогично, когда N равен 64, только верхние 32 строки коэффициентов преобразования сохраняются. Когда режим пропуска преобразования используется для большого блока,
45 весь блок используется вообще без обнуления значений.

[0084] *Виртуальные конвейерные единицы данных (VPDU) в VVC*

[0085] Виртуальные конвейерные единицы данных (VPDU) задаются как неперекрывающиеся единицы в картинке. В аппаратных декодерах, последовательные

VPDU обрабатываются посредством нескольких ступеней конвейера одновременно. VPDU-размер является приблизительно пропорциональным размеру буфера в большинстве ступеней конвейера, так что важно сохранять VPDU-размер небольшим. В большинстве аппаратных декодеров, VPDU-размер может задаваться равным

5 максимальному размеру блока преобразования (ТВ). Однако, в VVC, сегмент троичного дерева (ТТ) и двоичного дерева (ВТ) может приводить к увеличению размера VPDU.

[0086] Чтобы сохранять VPDU-размер в качестве выборок сигналов яркости 64×64 , следующие нормативные ограничения по сегментации (с модификацией сигнализации синтаксиса) применяются в VTM5:

10 [0087] - ТТ-разбиение не разрешается для CU с шириной или с высотой либо как с шириной, так и с высотой, равными 128.

[0088] - Для CU $128 \times N$ с $N \leq 64$ (т.е. с шириной, равной 128, и высотой, меньшей 128), горизонтальное сегментирование ВТ не разрешается.

[0089] - Для CU $N \times 128$ с $N \leq 64$ (т.е. с высотой, равной 128, и шириной, меньшей 128),

15 вертикальное сегментирование ВТ не разрешается.

[0090] Фиг. 6А, 6В, 6С, 6D, 6Е, 6F, 6G и 6H показывают примеры запрещенной ТТ- и ВТ-сегментации в VTM.

[0091] *Кодирование коэффициентов преобразования в VVC*

[0092] Кодирование коэффициентов преобразования в VVC является аналогичным

20 HEVC в том смысле, что оба из них используют неперекрывающиеся группы коэффициентов (также называемые "CG" или "субблоками"). Однако, также имеются некоторые различия между ними. В HEVC, каждая CG коэффициентов имеет фиксированный размер 4×4 . В VVC-проекте 6, CG-размер становится зависимым от ТВ-размера. Как следствие, различные CG-размеры (1×16 , 2×8 , 8×2 , 2×4 , 4×2 и 16×1)

25 доступны в VVC. CG в блоке кодирования и коэффициенты преобразования в CG кодируются согласно предварительно заданным порядкам сканирования.

[0093] Чтобы ограничивать максимальное число контекстно-кодированных элементов разрешения (бинов) в расчете на пиксел, площадь ТВ и тип видеокompонента (например, компонент сигналов яркости в сравнении с компонентом сигналов цветности)

30 используются для того, чтобы извлекать максимальное число контекстно-кодированных элементов разрешения (ССВ) для ТВ. Максимальное число контекстно-кодированных элементов разрешения равно $TV_zsize * 1,75$. Здесь, TV_zsize указывает число выборок в ТВ после обнуления коэффициентов. Следует отметить, что `coded_sub_block_flag`, который представляет собой флаг, указывающий то, содержит CG ненулевой

35 коэффициент или нет, не рассматривается для ССВ-количества.

[0094] Обнуление коэффициентов представляет собой операцию, выполняемую для блока преобразования для того, чтобы принудительно задавать коэффициенты, расположенные в некоторой области блока преобразования, равными 0. Например, в текущем VVC, преобразование 64×64 имеет ассоциированную операцию обнуления.

40 Как результат, коэффициенты преобразования, расположенные за пределами левой верхней области 32×32 в блоке преобразования 64×64 , принудительно задаются равными 0. Фактически, в текущем VVC, для любого блока преобразования с размером больше 32 вдоль некоторой размерности, операция обнуления коэффициентов выполняется вдоль этой размерности, чтобы принудительно задавать коэффициенты, расположенные

45 за пределами левой верхней области 32×32 , равными 0.

[0095] При кодировании коэффициентов преобразования в VVC, переменная, `remBinsPass1`, сначала задается равной максимальному числу разрешенных контекстно-кодированных элементов разрешения (МССВ). В процессе кодирования, переменная

снижается на единицу каждый раз, когда контекстно-кодированный элемент разрешения сигнализируется. Хотя `remBinsPass1` превышает или равна четыре, коэффициент сначала сигнализируется через синтаксисы `sig_coeff_flag`, `abs_level_gt1_flag`, `par_level_flag` и `abs_level_gt3_flag`, все с использованием контекстно-кодированных элементов разрешения в первом проходе. Остальная часть информации уровня коэффициента кодируется с синтаксическим элементом `abs_remainder` с использованием кода Голомба-Райса и подвергнутых обходному кодированию элементов разрешения во втором проходе. Когда `remBinsPass1` становится меньше 4 при кодировании первого прохода, текущий коэффициент не кодируется в первом проходе, а непосредственно кодируется во втором проходе с синтаксическим элементом `dec_abs_level` с использованием кода Голомба-Райса и подвергнутых обходному кодированию элементов разрешения. Процесс извлечения параметра Райса для `dec_abs_level[]` извлекается так, как указано в таблице 3. После всего вышеуказанного кодирования по уровням, знаки (`sign_flag`) для всех позиций сканирования с `sig_coeff_flag`, равным 1, в конечном счете кодируются в качестве обходных элементов разрешения. Такой процесс проиллюстрирован на фиг. 7; `remBinsPass1` сбрасывается для каждого ТВ. Переход от использования контекстно-кодированных элементов разрешения для `sig_coeff_flag`, `abs_level_gt1_flag`, `par_level_flag` и `abs_level_gt3_flag` к использованию подвергнутых обходному кодированию элементов разрешения для остальных коэффициентов происходит самое большее только один раз в расчете на ТВ. Для субблока коэффициентов, если `remBinsPass1` меньше 4 до кодирования его самого первого коэффициента, весь субблок коэффициентов кодируется с использованием подвергнутых обходному кодированию элементов разрешения.

[0096] Фиг. 7 показывает иллюстрацию структуры остаточного кодирования для блоков преобразования.

[0097] Унифицированное (одинаковое) извлечение параметра Райса (`RicePara`) используется для сигнализации синтаксиса `abs_remainder` и `dec_abs_level`. Единственное различие заключается в том, что базовый уровень, *baseLevel*, задается равным 4 и 0 для кодирования `abs_remainder` и `dec_abs_level`, соответственно. Параметр Райса определяется на основе не только суммы абсолютных уровней соседних пяти коэффициентов преобразования в локальном эталоне, но также и соответствующего базового уровня, следующим образом:

[0098] `RicePara=RiceParTable[max(min(31, sumAbs-5*baseLevel), 0)]`

[0099] Синтаксис и ассоциированная семантика остаточного кодирования в текущем проекте VVC-спецификации проиллюстрированы в таблице 1 и в таблице 2, соответственно. То, как следует читать таблицу 1, проиллюстрировано в разделе "Приложения" этого настоящего раскрытия, который также содержится в VVC-спецификации.

Таблица 1. Синтаксис остаточного кодирования		
40	residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	Дескриптор
	if(sps_mts_enabled_flag && cu_sbt_flag && cIdx== 0 && log2TbWidth== 5 && log2TbHeight < 6)	
	log2ZoTbWidth=4	
	else	
	log2ZoTbWidth=Min(log2TbWidth, 5)	
45	if(sps_mts_enabled_flag && cu_sbt_flag && cIdx== 0 && log2TbWidth < 6 && log2TbHeight== 5)	
	log2ZoTbHeight=4	
	else	
	log2ZoTbHeight=Min(log2TbHeight, 5)	
	if(log2TbWidth > 0)	

	last_sig_coeff_x_prefix	ae(v)
	if(log2TbHeight > 0)	
	last_sig_coeff_y_prefix	ae(v)
	if(last_sig_coeff_x_prefix > 3)	
5	last_sig_coeff_x_suffix	ae(v)
	if(last_sig_coeff_y_prefix > 3)	
	last_sig_coeff_y_suffix	ae(v)
	log2TbWidth=log2ZoTbWidth	
	log2TbHeight=log2ZoTbHeight	
	remBinsPass1=((1 << (log2TbWidth+log2TbHeight)) * 7) >> 2	
10	log2SbW=(Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2)	
	log2SbH=log2SbW	
	if(log2TbWidth+log2TbHeight > 3)	
	if(log2TbWidth < 2) {	
	log2SbW=log2TbWidth	
	log2SbH=4 - log2SbW	
15	} else if(log2TbHeight < 2) {	
	log2SbH=log2TbHeight	
	log2SbW=4 - log2SbH	
	}	
	numSbCoeff=1 << (log2SbW+log2SbH)	
	lastScanPos=numSbCoeff	
20	lastSubBlock=(1 << (log2TbWidth+log2TbHeight - (log2SbW+log2SbH))) - 1	
	do {	
	if(lastScanPos== 0) {	
	lastScanPos=numSbCoeff	
	lastSubBlock- -	
	}	
25	lastScanPos- -	
	xS=DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH]	
	[lastSubBlock][0]	
	yS=DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH]	
	[lastSubBlock][1]	
	xC=(xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][lastScanPos][0]	
	yC=(yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][lastScanPos][1]	
30	} while((xC != LastSignificantCoeffX) (yC != LastSignificantCoeffY))	
	if(lastSubBlock== 0 && log2TbWidth >= 2 && log2TbHeight >= 2 &&	
	!transform_skip_flag[x0][y0][cIdx] && lastScanPos > 0)	
	LfstDcOnly=0	
	if((lastSubBlock > 0 && log2TbWidth >= 2 && log2TbHeight >= 2)	
35	(lastScanPos > 7 && (log2TbWidth== 2 log2TbWidth== 3) &&	
	log2TbWidth== log2TbHeight))	
	LfstZeroOutSigCoeffFlag=0	
	if((lastSubBlock > 0 lastScanPos > 0) && cIdx== 0)	
	MtsDcOnly=0	
	QState=0	
	for(i=lastSubBlock; i >= 0; i- -) {	
40	startQStateSb=QState	
	xS=DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH]	
	[i][0]	
	yS=DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH]	
	[i][1]	
	inferSbDcSigCoeffFlag=0	
	if(i < lastSubBlock && i > 0) {	
45	sb_coded_flag[xS][yS]	ae(v)
	inferSbDcSigCoeffFlag=1	
	}	
	if(sb_coded_flag[xS][yS] && (xS > 3 yS > 3) && cIdx== 0)	
	MtsZeroOutSigCoeffFlag=0	

	firstSigScanPosSb=numSbCoeff	
	lastSigScanPosSb=-1	
	firstPosMode0=(i== lastSubBlock ? lastScanPos : numSbCoeff - 1)	
	firstPosMode1=firstPosMode0	
5	for(n=firstPosMode0; n >= 0 && remBinsPass1 >= 4; n- -) {	
	xC=(xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
	yC=(yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
	if(sb_coded_flag[xS][yS] && (n > 0 !inferSbDcSigCoeffFlag) && (xC != LastSignificantCoeffX yC != LastSignificantCoeffY)) {	
	sig_coeff_flag[xC][yC]	ae(v)
	remBinsPass1- -	
10	if(sig_coeff_flag[xC][yC])	
	inferSbDcSigCoeffFlag=0	
	}	
	if(sig_coeff_flag[xC][yC]) {	
	abs_level_gtx_flag[n][0]	ae(v)
	remBinsPass1- -	
15	if(abs_level_gtx_flag[n][0]) {	
	par_level_flag[n]	ae(v)
	remBinsPass1- -	
	abs_level_gtx_flag[n][1]	ae(v)
	remBinsPass1- -	
	}	
20	if(lastSigScanPosSb== -1)	
	lastSigScanPosSb=n	
	firstSigScanPosSb=n	
	}	
	AbsLevelPass1[xC][yC]=sig_coeff_flag[xC][yC] + par_level_flag[n] + abs_level_gtx_flag[n][0] + 2 * abs_level_gtx_flag[n][1]	
25	if(sh_dep_quant_used_flag)	
	QState=QStateTransTable[QState][AbsLevelPass1[xC][yC] & 1]	
	firstPosMode1=n - 1	
	}	
	for(n=firstPosMode0; n > firstPosMode1; n- -) {	
30	xC=(xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
	yC=(yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
	if(abs_level_gtx_flag[n][1])	
	abs_remainder[n]	ae(v)
	AbsLevel[xC][yC]=AbsLevelPass1[xC][yC] + 2 * abs_remainder[n]	
	}	
	for(n=firstPosMode1; n >= 0; n- -) {	
35	xC=(xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
	yC=(yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
	if(sb_coded_flag[xS][yS])	
	dec_abs_level[n]	ae(v)
	if(AbsLevel[xC][yC] > 0) {	
40	if(lastSigScanPosSb== -1)	
	lastSigScanPosSb=n	
	firstSigScanPosSb=n	
	}	
	if(sh_dep_quant_used_flag)	
	QState=QStateTransTable[QState][AbsLevel[xC][yC] & 1]	
45	}	
	signHiddenFlag=sh_sign_data_hiding_used_flag && (lastSigScanPosSb - firstSigScanPosSb > 3 ? 1 : 0)	
	for(n=numSbCoeff - 1; n >= 0; n- -) {	
	xC=(xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
	yC=(yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	

	if((AbsLevel[xC][yC] > 0) && (!signHiddenFlag (n != firstSigScanPosSb)))	
	coeff_sign_flag[n]	ae(v)
	}	
	if(sh_dep_quant_used_flag) {	
5	QState=startQStateSb	
	for(n=numSbCoeff - 1; n >= 0; n--) {	
	xC=(xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
	yC=(yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
	if(AbsLevel[xC][yC] > 0)	
10	TransCoeffLevel[x0][y0][cIdx][xC][yC] = (2 * AbsLevel[xC][yC] - (QState > 1 ? 1 : 0)) * (1 - 2 * coeff_sign_flag[n])	
	QState=QStateTransTable[QState][AbsLevel[xC][yC] & 1]	
	} else {	
	sumAbsLevel=0	
	for(n=numSbCoeff - 1; n >= 0; n--) {	
15	xC=(xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
	yC=(yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
	if(AbsLevel[xC][yC] > 0) {	
	TransCoeffLevel[x0][y0][cIdx][xC][yC] = AbsLevel[xC][yC] * (1 - 2 * coeff_sign_flag[n])	
	if(signHiddenFlag) {	
20	sumAbsLevel += AbsLevel[xC][yC]	
	if((n== firstSigScanPosSb) && (sumAbsLevel % 2)== 1))	
	TransCoeffLevel[x0][y0][cIdx][xC][yC] = -TransCoeffLevel[x0][y0][cIdx][xC][yC]	
	}	
	}	
	}	
25	}	
	}	
	}	

Таблица 2. Семантика остаточного кодирования

30	Массив AbsLevel[xC][yC] представляет массив абсолютных значений уровней коэффициентов преобразования для текущего блока преобразования, и массив AbsLevelPass1[xC][yC] представляет массив частично восстановленных абсолютных значений уровней коэффициентов преобразования для текущего блока преобразования. Индексы xC и yC массива указывают местоположение (xC, yC) коэффициентов преобразования в текущем блоке преобразования. Когда значение AbsLevel[xC][yC] не указывается в подразделе 7.3.11.11, оно логически выводится равным 0. Когда значение AbsLevelPass1[xC][yC] не указывается в подразделе 7.3.11.11, оно логически выводится равным 0.
	Переменные CoeffMin и CoeffMax, указывающие минимальные и максимальные значения коэффициентов преобразования, извлекаются следующим образом: CoeffMin=(1<<15) (189) CoeffMax=(1<<15)-1 (190)
35	Массив QStateTransTable[] указывается следующим образом: QStateTransTable[]={{0, 2}, {2, 0}, {1, 3}, {3, 1}} (191) - last_sig_coeff_x_prefix указывает префикс позиции столбца последнего значимого коэффициента в порядке сканирования в блоке преобразования. Значения last_sig_coeff_x_prefix должны составлять в диапазоне от 0 до (log2ZoTbWidth<<1)-1, включительно.
	Когда last_sig_coeff_x_prefix не присутствует, он логически выводится равным 0.
40	- last_sig_coeff_y_prefix указывает префикс позиции строки последнего значимого коэффициента в порядке сканирования в блоке преобразования. Значения last_sig_coeff_y_prefix должны составлять в диапазоне от 0 до (log2ZoTbHeight<<1)-1, включительно.
	Когда last_sig_coeff_y_prefix не присутствует, он логически выводится равным 0.
	- last_sig_coeff_x_suffix указывает суффикс позиции столбца последнего значимого коэффициента в порядке сканирования в блоке преобразования. Значения last_sig_coeff_x_suffix должны составлять в диапазоне 0 в (1<<((last_sig_coeff_x_prefix>>1)-1))-1, включительно.
	Позиция столбца последнего значимого коэффициента в порядке сканирования в блоке LastSignificantCoeffX преобразования извлекается следующим образом: - Если last_sig_coeff_x_suffix не присутствует, применимо следующее: LastSignificantCoeffX=last_sig_coeff_x_prefix (192) - Иначе (last_sig_coeff_x_suffix присутствует), применимо следующее: LastSignificantCoeffX=(1<<((last_sig_coeff_x_prefix>>1)-1))* (193) (2+(last_sig_coeff_x_prefix и 1))+last_sig_coeff_x_suffix
45	- last_sig_coeff_y_suffix указывает суффикс позиции строки последнего значимого коэффициента в порядке сканирования в блоке преобразования. Значения last_sig_coeff_y_suffix должны составлять в диапазоне 0 в (1<<((last_sig_coeff_y_prefix>>1)-1))-1, включительно.
	Позиция строки последнего значимого коэффициента в порядке сканирования в блоке LastSignificantCoeffY преобразования извлекается следующим образом:

- Если last_sig_coeff_y_suffix не присутствует, применимо следующее:
LastSignificantCoeffY=last_sig_coeff_y_prefix (194)
- Иначе (last_sig_coeff_y_suffix присутствует), применимо следующее:
LastSignificantCoeffY=(1<<((last_sig_coeff_y_prefix>>1)-1))* (195)
(2+(last_sig_coeff_y_prefix и 1))+last_sig_coeff_y_suffix

5 - sb_coded_flag[xS][yS] указывает следующее для субблока в местоположении (xS, yS) в текущем блоке преобразования, причем субблок представляет собой массив уровней коэффициентов преобразования:
Когда sb_coded_flag[xS][yS] равен 0, все уровни коэффициентов преобразования субблока в местоположении (xS, yS) логически выводятся равными 0.
Когда sb_coded_flag[xS][yS] не присутствует, он логически выводится равным 1.

- sig_coeff_flag[xC][yC] указывает для местоположения (xC, yC) коэффициентов преобразования в текущем блоке преобразования то, является или нет соответствующий уровень коэффициентов преобразования в местоположении (xC, yC) ненулевым, следующим образом:
- Если sig_coeff_flag[xC][yC] равен 0, уровень коэффициентов преобразования в местоположении (xC, yC) задается равным 0.
- Иначе (sig_coeff_flag[xC][yC] равен 1), уровень коэффициентов преобразования в местоположении (xC, yC) имеет ненулевое значение.

10 - Когда sig_coeff_flag[xC][yC] не присутствует, он логически выводится следующим образом:
- Если transform_skip_flag[x0][y0][cIdx] равен 0, или sh_ts_residual_coding_disabled_flag равен 1, применимо следующее:
- if(xC, yC) представляет собой последнее значимое местоположение (LastSignificantCoeffX, LastSignificantCoeffY) в порядке сканирования, или все следующие условия представляют собой "истина", sig_coeff_flag[xC][yC] логически выводится равным 1:
- (xC и ((1<<log2SbW)-1), yC и ((1<<log2SbH)-1)) равно (0, 0).
- inferSbDcSigCoeffFlag равен 1.
- sb_coded_flag[xS][yS] равен 1.
- Иначе, sig_coeff_flag[xC][yC] логически выводится равным 0.

15 - Иначе (transform_skip_flag[x0][y0][cIdx] равен 1, и sh_ts_residual_coding_disabled_flag, равен 0), применимо следующее:
- Если все следующие условия представляют собой "истина", sig_coeff_flag[xC][yC] логически выводится равным 1:
- (xC и ((1<<log2SbW)-1), yC и ((1<<log2SbH)-1)) равно ((1<<log2SbW)-1, (1<<log2SbH)-1).
- inferSbSigCoeffFlag равен 1.
- sb_coded_flag[xS][yS] равен 1.
- Иначе, sig_coeff_flag[xC][yC] логически выводится равным 0.

- abs_level_gtx_flag[n][j] указывает то, больше абсолютное значение уровня коэффициентов преобразования (в позиции n сканирования) или нет (j <= 1) + 1. Когда abs_level_gtx_flag[n][j] не присутствует, он логически выводится равным 0.

20 - par_level_flag[n] указывает четность уровня коэффициентов преобразования в позиции n сканирования. Когда par_level_flag[n] не присутствует, он логически выводится равным 0.
- abs_remainder[n] является оставшимся абсолютным значением уровня коэффициентов преобразования, которое кодируется с кодом Голомба-Райса в позиции n сканирования. Когда abs_remainder[n] не присутствует, оно логически выводится равным 0. Требование соответствия потока битов заключается в том, что значение abs_remainder[n] должно ограничиваться таким образом, что соответствующее значение TransCoeffLevel[x0][y0][cIdx][xC][yC] составляет в диапазоне CoeffMin-CoeffMax, включительно.
- dec_abs_level[n] является промежуточным значением, которое кодируется с кодом Голомба-Райса в позиции n сканирования. С учетом ZeroPos[n], который извлекается в таблице 3 во время синтаксического анализа dec_abs_level[n], абсолютное значение уровня коэффициентов преобразования в местоположении (xC, yC) AbsLevel[xC][yC] извлекается следующим образом:
- Если dec_abs_level[n] не присутствует или равен ZeroPos[n], AbsLevel[xC][yC] задается равным 0.
- Иначе, если dec_abs_level[n] меньше ZeroPos[n], AbsLevel[xC][yC] задается равным dec_abs_level[n]+1;
- Иначе (dec_abs_level[n] больше ZeroPos[n]), AbsLevel[xC][yC] задается равным dec_abs_level[n].

25 - Требование соответствия потока битов заключается в том, что значение dec_abs_level[n] должно ограничиваться таким образом, что соответствующее значение TransCoeffLevel[x0][y0][cIdx][xC][yC] составляет в диапазоне CoeffMin-CoeffMax, включительно.
- coeff_sign_flag[n] указывает знак уровня коэффициентов преобразования для позиции n сканирования следующим образом:
- Если coeff_sign_flag[n] равен 0, соответствующий уровень коэффициентов преобразования имеет положительное значение.
- Иначе (coeff_sign_flag[n] равен 1), соответствующий уровень коэффициентов преобразования имеет отрицательное значение.
Когда coeff_sign_flag[n] не присутствует, он логически выводится равным 0.

30 - Значение CoeffSignLevel[xC][yC] указывает знак уровня коэффициентов преобразования в местоположении (xC, yC) следующим образом:
- Если CoeffSignLevel[xC][yC] равен 0, соответствующий уровень коэффициентов преобразования равен нулю
- Иначе, если CoeffSignLevel[xC][yC] равен 1, соответствующий уровень коэффициентов преобразования имеет положительное значение.
- Иначе (CoeffSignLevel[xC][yC] равен -1), соответствующий уровень коэффициентов преобразования имеет отрицательное значение.

35 - Таблица 3. Процесс извлечения параметра Райса для abs_remainder[] и dec_abs_level[]

Вводы в этот процесс представляют собой базовый уровень baseLevel, индекс cIdx цветового компонента, местоположение (x0, y0) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм log2TbWidth ширины блока преобразования и двоичный логарифм log2TbHeight высоты блока преобразования.
Вывод этого процесса представляет собой параметр cRiceParam Райса.
С учетом массива AbsLevel[x][y] для блока преобразования с индексом cIdx компонента и левым верхним местоположением (x0, y0) сигнала яркости, переменная locSumAbs извлекается так, как указано посредством следующего псевдокода:

40 locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
locSumAbs+=AbsLevel[xC+1][yC]
if(xC<(1<<log2TbWidth)-2)
locSumAbs+=AbsLevel[xC+2][yC]
if(yC<(1<<log2TbHeight)-1)
locSumAbs+=AbsLevel[xC+1][yC+1]
(1494)
}
if(yC<(1<<log2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]
if(yC<(1<<log2TbHeight)-2)
locSumAbs+=AbsLevel[xC][yC+2]
}
locSumAbs=Clip3(0, 31, locSumAbs-baseLevel*5)

45 - С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается так, как указано в нижеприведенной таблице 4.

Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:
 $ZeroPos[n] = (QState < 2^{?1:2}) < cRiceParam$

Таблица 4. Спецификация cRiceParam на основе locSumAbs

locSumAbs	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cRiceParam	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2
locSumAbs	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cRiceParam	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3

[00100] *Остаточное кодирование для режима пропуска преобразования в VVC*

[00101] В отличие от HEVC, в котором одна схема остаточного кодирования проектируется для кодирования коэффициентов преобразования и коэффициентов пропуска преобразования в VVC, две отдельных схемы остаточного кодирования используются для коэффициентов преобразования и коэффициентов пропуска преобразования (т.е. остатка), соответственно.

[00102] В режиме пропуска преобразования, статистические характеристики остаточного сигнала отличаются от статистических характеристик коэффициентов преобразования, и энергетическое уплотнение вокруг низкочастотных компонентов не наблюдается. Остаточное кодирование модифицируется для того, чтобы учитывать различные характеристики сигналов (пространственного) остатка пропуска преобразования, которые включают в себя:

[00103] - отсутствие сигнализации последней позиции по оси X/Y;

[00104] - coded_sub_block_flag, закодированный для каждого субблока за исключением DC-субблока, когда все предыдущие флаги равны 0;

[00105] - sig_coeff_flag, контекстно моделирующий с двумя соседними коэффициентами;

[00106] - par_level_flag с использованием только одной контекстной модели;

[00107] - дополнительные более чем 5, 7, 9 флагов;

[00108] - модифицированное извлечение параметра Райса для преобразования в двоичную форму оставшейся части;

[00109] - контекстное моделирование для флага знака определяется на основе значений левого и вышележащего соседнего коэффициента, и флаг знака синтаксически анализируется после sig_coeff_flag, чтобы поддерживать все контекстно-кодированные элементы разрешения вместе.

[00110] Как показано на фиг. 8, синтаксические элементы *sig_coeff_flag*, *coeff_sign_flag*, *abs_level_gt1_flag*, *par_level_flag* кодируются перемеженным способом по остаточным выборкам в первом проходе, с дальнейшим кодированием битовых плоскостей *abs_level_gtX_flag* во втором проходе и кодированием *abs_remainder* в третьем проходе.

[00111] Проход 1: sig_coeff_flag, coeff_sign_flag, abs_level_gt1_flag, par_level_flag

[00112] Проход 2: abs_level_gt3_flag, abs_level_gt5_flag, abs_level_gt7_flag, abs_level_gt9_flag

[00113] Проход 3: abs_remainder

[00114] Фиг. 8 показывает иллюстрацию структуры остаточного кодирования для блоков с пропуском преобразования.

[00115] Синтаксис и ассоциированная семантика остаточного кодирования для режима пропуска преобразования в текущем проекте VVC-спецификации проиллюстрированы в таблице 5 и в таблице 2, соответственно. То, как следует читать таблицу 5, проиллюстрировано в разделе "Приложения" этого настоящего раскрытия, который также содержится в VVC-спецификации.

Таблица 5. Синтаксис остаточного кодирования для режима пропуска преобразования

	residual_ts_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	Дескриптор
	log2SbW=(Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2)	
	log2SbH=log2SbW	
	if(log2TbWidth+log2TbHeight > 3)	
5	if(log2TbWidth < 2) {	
	log2SbW=log2TbWidth	
	log2SbH=4 - log2SbW	
	} else if(log2TbHeight < 2) {	
	log2SbH=log2TbHeight	
	log2SbW=4 - log2SbH	
10	}	
	numSbCoeff=1 << (log2SbW+log2SbH)	
	lastSubBlock=(1 << (log2TbWidth+log2TbHeight - (log2SbW+log2SbH))) - 1	
	inferSbCbf=1	
	RemCpbs=((1 << (log2TbWidth+log2TbHeight)) * 7) >> 2	
	for(i =0; i <= lastSubBlock; i++) {	
15	xS=DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH][i][0]	
	yS=DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH][i][1]	
	if(i != lastSubBlock !inferSbCbf)	
	sb_coded_flag[xS][yS]	ae(v)
	if(sb_coded_flag[xS][yS] && i < lastSubBlock)	
	inferSbCbf=0	
20	/* First scan pass */	
	inferSbSigCoeffFlag=1	
	lastScanPosPass1=-1	
	for(n=0; n <= numSbCoeff - 1 && RemCpbs >= 4; n++) {	
	xC=(xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
	yC=(yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
25	lastScanPosPass1=n	
	if(sb_coded_flag[xS][yS] && (n != numSbCoeff - 1 !inferSbSigCoeffFlag)) {	
	sig_coeff_flag[xC][yC]	ae(v)
	RemCpbs- -	
	if(sig_coeff_flag[xC][yC])	
30	inferSbSigCoeffFlag=0	
	}	
	CoeffSignLevel[xC][yC]=0	
	if(sig_coeff_flag[xC][yC]) {	
	coeff_sign_flag[n]	ae(v)
	RemCpbs- -	
35	CoeffSignLevel[xC][yC]=(coeff_sign_flag[n] > 0 ? -1 : 1)	
	abs_level_gtx_flag[n][0]	ae(v)
	RemCpbs- -	
	if(abs_level_gtx_flag[n][0]) {	
	par_level_flag[n]	ae(v)
	RemCpbs- -	
40	}	
	}	
	AbsLevelPass1[xC][yC] =	
	sig_coeff_flag[xC][yC] + par_level_flag[n] + abs_level_gtx_flag[n][0]	
	}	
	/* Greater than X scan pass (numGtXFlags=5) */	
45	lastScanPosPass2=-1	
	for(n=0; n <= numSbCoeff - 1 && RemCpbs >= 4; n++) {	
	xC=(xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
	yC=(yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
	AbsLevelPass2[xC][yC]=AbsLevelPass1[xC][yC]	

	for(j=1; j < 5; j++) {	
	if(abs_level_gtx_flag[n][j - 1]) {	
	abs_level_gtx_flag[n][j]	ae(v)
	RemCcbs- -	
	}	
5	AbsLevelPass2[xC][yC] += 2 * abs_level_gtx_flag[n][j]	
	}	
	lastScanPosPass2=n	
	}	
	/* remainder scan pass */	
10	for(n=0; n <= numSbCoeff - 1; n++) {	
	xC=(xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
	yC=(yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
	if((n <= lastScanPosPass2 && AbsLevelPass2[xC][yC] >= 10) (n > lastScanPosPass2 && n <= lastScanPosPass1 && AbsLevelPass1[xC][yC] >= 2) (n > lastScanPosPass1 && sb_coded_flag[xS][yS]))	
15	abs_remainder[n]	ae(v)
	if(n <= lastScanPosPass2)	
	AbsLevel[xC][yC]=AbsLevelPass2[xC][yC] + 2 * abs_remainder[n]	
	else if(n <= lastScanPosPass1)	
	AbsLevel[xC][yC]=AbsLevelPass1[xC][yC] + 2 * abs_remainder[n]	
	else { /* bypass */	
20	AbsLevel[xC][yC]=abs_remainder[n]	
	if(abs_remainder[n])	
	coeff_sign_flag[n]	ae(v)
	}	
	if(BdpcmFlag[x0][y0][cldx] == 0 && n <= lastScanPosPass1) {	
	absLeftCoeff=xC > 0 ? AbsLevel[xC - 1][yC] : 0	
25	absAboveCoeff=yC > 0 ? AbsLevel[xC][yC - 1] : 0	
	predCoeff=Max(absLeftCoeff, absAboveCoeff)	
	if(AbsLevel[xC][yC] == 1 && predCoeff > 0)	
	AbsLevel[xC][yC]=predCoeff	
	else if(AbsLevel[xC][yC] > 0 && AbsLevel[xC][yC] <= predCoeff)	
	AbsLevel[xC][yC]- -	
30	}	
	TransCoeffLevel[x0][y0][cldx][xC][yC]=(1 - 2 * coeff_sign_flag[n]) * AbsLevel[xC][yC]	
	}	
	}	
	}	

35 [00116] *Квантование*

[00117] В текущем VVC, максимальное QR-значение расширено от 51 до 63, и передача служебных сигналов начального QR изменяется, соответственно. Начальное значение SliceQpY может модифицироваться в слое сегментов срезов, когда ненулевое значение slice_qp_delta кодируется. Для блока с пропуском преобразования, минимальный разрешенный параметр квантования (QR) задается 4, поскольку размер шага квантования становится 1, когда QR равен 4.

40 [00118] Помимо этого, такое же скалярное HEVC-квантование используется с новой концепцией, называемой "зависимым скалярным квантованием". Зависимое скалярное квантование означает подход, при котором набор допустимых значений восстановления для коэффициента преобразования зависит от значений уровней коэффициентов преобразования, которые предшествуют текущему уровню коэффициентов преобразования в порядке восстановления. Основное преимущество этого подхода заключается в том, что, по сравнению с традиционным независимым скалярным

квантованием, используемым в HEVC, допустимые векторы восстановления пакетируются плотнее в N-мерном векторном пространстве (N представляет число коэффициентов преобразования в блоке преобразования). Это означает, для данного среднего числа допустимых векторов восстановления в расчете на N-мерный единичный объем, то, что среднее искажение между входным вектором и ближайшим вектором восстановления уменьшается. Подход зависимого скалярного квантования реализуется посредством следующего: (а) задание двух скалярных квантователей с различными уровнями восстановления и (b) задание процесса для переключения между двумя скалярными квантователями.

[00119] Два используемых скалярных квантователя, обозначаемые посредством Q0 и Q1, проиллюстрированы на фиг. 9. Местоположение доступных уровней восстановления уникально указывается посредством размера Δ шага квантования. Используемый скалярный квантователь (Q0 или Q1) явно не сигнализируется в потоке битов. Вместо этого, квантователь, используемый для текущего коэффициента преобразования, определяется посредством четностей уровней коэффициентов преобразования, которые предшествуют текущему коэффициенту преобразования в порядке кодирования/восстановления.

[00120] Фиг. 9 показывает иллюстрацию двух скалярных квантователей, используемых в предложенном подходе зависимого квантования.

[00121] Как проиллюстрировано на фиг. 10A и 10B, переключение между двумя скалярными квантователями (Q0 и Q1) реализуется через машину состояний с четырьмя состояниями (QState) квантователя. QState может принимать четыре различных значения: 0, 1, 2, 3. Оно то уникально определяется посредством четностей уровней коэффициентов преобразования, предшествующих текущему коэффициенту преобразования в порядке кодирования/восстановления. В начале обратного квантования для блока преобразования, состояние задается равным 0. Коэффициенты преобразования восстанавливаются в порядке сканирования (т.е. в таком же порядке, в котором они энтропийно декодируются). После того, как текущий коэффициент преобразования восстанавливается, состояние обновляется так, как показано на фиг. 10, где k обозначает значение уровня коэффициентов преобразования.

[00122] Фиг. 10A показывает схему переходов, иллюстрирующую переход состояния для предложенного зависимого квантования.

[00123] Фиг. 10B показывает таблицу, иллюстрирующую выбор квантователя для предложенного зависимого квантования.

[00124] Также поддерживается возможность сигнализировать задаваемые по умолчанию и определяемые пользователем масштабирующие матрицы. Масштабирующие матрицы в режиме по умолчанию являются плоскими, с элементами, равными 16 для всех ТВ-размеров. IBC-режимы и режимы внутреннего кодирования в данный момент совместно используют одинаковые масштабирующие матрицы. Таким образом, для случая определяемых пользователем матриц, число MatrixType и MatrixType_DC обновляется следующим образом:

[00125] MatrixType: $30=2$ (2 для intraandIBC/inter) $\times 3$ (Y/Cb/Cr-компоненты) $\times 5$ (квадратный ТВ-размер: от 4×4 до 64×64 для сигнала яркости, от 2×2 до 32×32 для сигнала цветности).

[00126] MatrixType_DC: $14=2$ (2 для intraandIBC/inter $\times 1$ для Y-компонента) $\times 3$ (ТВ-размер: 16×16 , 32×32 , 64×64) $+4$ (2 для intraandIBC/inter $\times 2$ для Cb/Cr-компонентов) $\times 2$ (ТВ-размер: 16×16 , 32×32).

[00127] DC-значения отдельно кодируются для следующих масштабирующих матриц:

16×16, 32×32 и 64×64. Для ТВ размера, меньшего 8×8, все элементы в одной масштабирующей матрице сигнализируются. Если ТВ имеют размер, больший или равный 8×8, только 64 элемента в одной масштабирующей матрице 8×8 сигнализируются в качестве базовой масштабирующей матрицы. Для получения квадратных матриц размера, большего 8×8, базовая масштабирующая матрица 8×8 повышающе дискретизируется (посредством дублирования элементов) до соответствующего квадратного размера (т.е. 16×16, 32×32, 64×64). Когда обнуление высокочастотных коэффициентов для 64-точечного преобразования применяется, соответствующие высокие частоты масштабирующих матриц также обнуляются. Таким образом, если ширина или высота ТВ больше или равна 32, только левая или верхняя половина коэффициентов сохраняется, и оставшиеся коэффициенты назначаются нулю. Кроме того, число элементов, сигнализируемых для масштабирующей матрицы 64×64, также уменьшается с 8×8 до трех субматриц 4×4, поскольку правые нижние элементы 4×4 никогда не используются.

[00128] *Контекстное моделирование для кодирования коэффициентов преобразования*

[00129] Выбор вероятностных моделей для синтаксических элементов, связанных с абсолютными значениями уровней коэффициентов преобразования, зависит от значений абсолютных уровней или частично восстановленных абсолютных уровней в локальном окружении. Используемый эталон проиллюстрирован на фиг. 11.

[00130] Фиг. 11 показывает иллюстрацию эталона, используемого для выбора вероятностных моделей. Черный квадрат указывает текущую позицию сканирования, и квадраты с "x" представляют используемое локальное окружение.

[00131] Выбранные вероятностные модели зависят от суммы абсолютных уровней (или частично восстановленных абсолютных уровней) в локальном окружении и числа абсолютных уровней, большего 0 (задаваемого посредством числа sig_coeff_flags, равного 1) в локальном окружении. Контекстное моделирование и преобразование в двоичную форму зависят от следующих показателей для локального окружения:

[00132] - numSig: число ненулевых уровней в локальном окружении;

[00133] - sumAbs1: сумма частично восстановленных абсолютных уровней (absLevel1)

после первого прохода в локальном окружении;

[00134] - sumAbs: сумма восстановленных абсолютных уровней в локальном окружении;

[00135] - diagonal position(d): сумма горизонтальных и вертикальных координат текущей позиции сканирования в блоке преобразования.

[00136] На основе значений numSig, sumAbs1 и d, выбираются вероятностные модели для кодирования sig_coeff_flag, abs_level_gt1_flag, par_level_flag и abs_level_gt3_flag. Параметр Райса для преобразования в двоичную форму abs_remainder и dec_abs_level выбирается на основе значений sumAbs и numSig.

[00137] В текущем VVC, уменьшенный 32-точечный MTS (также называемый "RMTS32") основан на пропуске высокочастотных коэффициентов и используется для того, чтобы уменьшать вычислительную сложность 32-точечного DST-7/DCT-8. Так же, он сопровождает изменения при кодировании коэффициентов, включающие в себя все типы обнуления (т.е. RMTS32 и существующее обнуление для высокочастотных компонентов в DCT2). В частности, преобразование в двоичную форму позиционного кодирования последних ненулевых коэффициентов кодируется на основе уменьшенного TU-размера, и выбор контекстной модели для позиционного кодирования последних ненулевых коэффициентов определяется посредством исходного TU-размера. Помимо этого, 60 контекстных моделей используются для того, чтобы кодировать sig_coeff_flag

коэффициентов преобразования. Выбор индекса контекстной модели основан на сумме максимума пяти ранее частично восстановленных абсолютных уровней, называемой "locSumAbsPass1", и на состоянии QState зависимого квантования следующим образом:

[00138] Если cIdx равен 0, ctxInc извлекается следующим образом:

ctxInc=12*Max(0, QState-1)+
Min((locSumAbsPass1+1)>>1, 3)+(d<2?8:(d<5?4:0))

[00139] Иначе (cIdx больше 0), ctxInc извлекается следующим образом:

ctxInc=36+8*Max(0, QState-1)+
Min((locSumAbsPass1+1)>>1, 3)+(d<2?4:0)

[00140] *Палитровый режим*

[00141] Базовая идея в основе палитрового режима состоит в том, что выборки в CU представляются посредством небольшого набора репрезентативных значений цветов. Этот набор называется "палитрой". Также можно указывать значение цвета, которое исключается из палитры, посредством его сигнализации в качестве цвета перехода, для которого значения трех цветовых компонентов непосредственно сигнализируются в потоке битов. Это проиллюстрировано на фиг. 12.

[00142] Фиг. 12 показывает пример блока, кодированного в палитровом режиме. Фиг. 12 включает в себя блок 1210, кодированный в палитровом режиме, и палитру 1220.

[00143] На фиг. 12, размер палитры равен 4. Первые 3 выборки используют записи 2, 0 и 3 палитры, соответственно, для восстановления. Выборка черного цвета представляет символ перехода. Флаг уровня CU, palette_escape_val_present_flag, указывает то, присутствуют или нет вообще символы перехода в CU. Если символы перехода присутствуют, размер палитры дополняется единицей, и последний индекс используется для того, чтобы указывать символ перехода. Таким образом, на фиг. 12, индекс 4 назначается символу перехода.

[00144] Для декодирования кодированного в палитровом режиме блока, декодер должен иметь следующую информацию:

[00145] - Таблица палитр;

[00146] - Индексы палитр.

[00147] Если индекс палитры соответствует символу перехода, дополнительный объем служебной информации сигнализируется для того, чтобы указывать соответствующие значения цветов выборки.

[00148] Помимо этого, на стороне кодера, необходимо извлекать соответствующую палитру, которая должна использоваться с этой CU.

[00149] Для извлечения палитры для кодирования с потерями, используется модифицированный алгоритм кластеризации методом k-средних. Первая выборка блока добавляется в палитру. После этого, для каждой последующей выборки из блока, сумма абсолютных разностей (SAD) между выборкой и каждым текущим цветом палитры вычисляется. Если искажение для каждого из компонентов меньше порогового значения для записи палитры, соответствующей минимальной SAD, выборка добавляется в кластер, принадлежащий записи палитры. В противном случае, выборка добавляется в качестве новой записи палитры. Когда число выборок, отображаемое в кластер, превышает пороговое значение, центроид для этого кластера обновляется и становится записью палитры этого кластера.

[00150] На следующем этапе, кластеры сортируются в порядке по убыванию использования. Затем, запись палитры, соответствующая каждой записи, обновляется. Нормально, центроид кластера используется в качестве записи палитры. Но анализ

искажения в зависимости от скорости передачи выполняется для того, чтобы анализировать то, может или нет какая-либо запись из предиктора палитры быть более подходящей для использования в качестве обновленной записи палитры вместо центроида, когда затраты кодирования записей палитры принимаются во внимание.

5 Этот процесс продолжается до тех пор, пока все кластеры не обрабатываются, или максимальный размер палитры не достигается. В завершение, если кластер имеет только одну выборку, и соответствующая запись палитры не находится в предикторе палитры, выборка преобразуется в символ перехода. Дополнительно, дублированные записи палитры удаляются, и их кластеры объединяются.

10 [00151] После извлечения палитры, каждой выборке в блоке назначается индекс ближайшей (в SAD) записи палитры. Затем выборки назначаются "индексному" режиму или режиму "копирования сверху". Для каждой выборки, для которой является возможным "индексный" режим или режим "копирования сверху". После этого, затраты кодирования режима вычисляются. Режим, для которого затраты являются более
15 низкими, выбирается.

[00152] Для кодирования записей палитры, предиктор палитры поддерживается. Максимальный размер палитры, а также предиктора палитры сигнализируется в SPS. Предиктор палитры инициализируется в начале каждой STU-строки, каждого среза и каждой плитки.

20 [00153] Для каждой записи в предикторе палитры, флаг многократного использования сигнализируется для того, чтобы указывать то, представляет он собой или нет часть текущей палитры. Это проиллюстрировано на фиг. 13.

[00154] Фиг. 13 показывает использование предиктора палитры для того, чтобы сигнализировать записи палитры. Фиг. 13 включает в себя предыдущую палитру 1310
25 и текущую палитру 1320.

[00155] Флаги многократного использования отправляются с использованием кодирования по длинам серий нулей. После этого, число новых записей палитры сигнализируется с использованием экспоненциального кода Голомба порядка 0. В завершение, значения компонентов для новых записей палитры сигнализируются.

30 [00156] Индексы палитр кодируются с использованием горизонтального и вертикального пересекающихся сканирований, как показано на фиг. 14A и 14B. Порядок сканирования явно сигнализируется в потоке битов с использованием `palette_transpose_flag`.

[00157] Фиг. 14A показывает горизонтальное пересекающее сканирование.

35 [00158] Фиг. 14B показывает вертикальное пересекающее сканирование.

[00159] Для кодирования индексов палитр, используется палитровый режим на основе групп коэффициентов (CG) линий, который разделяет CU на несколько сегментов с 16 выборками на основе режима пересекающего сканирования, как показано на фиг. 15A и 15B, при этом серии индексов, значения индексов палитр и квантованные цвета для
40 режима кодирования управляющим кодом кодируются/синтаксически анализируются последовательно для каждой CG.

[00160] Фиг. 15A показывает сканирование карты индексов на основе субблоков для палитры.

45 [00161] Фиг. 15B показывает сканирование карты индексов на основе субблоков для палитры.

[00162] Индексы палитр кодируются с использованием двух основных палитровых выборочных режимов: "индексный" и "копирование сверху". Как пояснено выше, символу перехода назначается индекс, равный максимальному размеру палитры. В

режиме "копирования сверху", индекс палитры выборки в строке выше копируется. В "индексном" режиме, индекс палитры явно сигнализируется. Порядок кодирования для кодирования по сериям палитр в каждом сегменте заключается в следующем:

[00163] Для каждого пиксела, 1 контекстно-кодированный элемент разрешения `run_cory_flag=0` сигнализируется, указывающий то, имеет или нет пиксел одинаковый режим с предыдущим пикселом, т.е. то, имеют или нет предыдущий отсканированный пиксел и текущий пиксел тип серии "копирование сверху", либо то, имеют или нет предыдущий отсканированный пиксел и текущий пиксел тип серии "индексный" и одинаковое значение индекса. В противном случае, сигнализируется `run_cory_flag=1`.

[00164] Если пиксел и предыдущий пиксел имеют различный режим, один контекстно-кодированный элемент `cory_above_palette_indices_flag` разрешения сигнализируется, указывающий тип серии, т.е. "индексный" или "копирование сверху", пиксела. Декодер не должен синтаксически анализировать тип серии, если выборка находится в первой строке (горизонтальное пересекающее сканирование) или в первом столбце (вертикальное пересекающее сканирование), поскольку индексный режим используется по умолчанию. Кроме того, декодер не должен синтаксически анализировать тип серии, если ранее синтаксически проанализированный тип серии представляет собой "копирование сверху".

После кодирования по сериям палитр пикселов в одном сегменте, значения индекса для индексного режима (`palette_idx_idc`) и квантованные цвета перехода (`palette_escape_val`) подвергаются обходному кодированию.

[00165] *Улучшения остаточного кодирования и кодирования коэффициентов*

[00166] В VVC, при кодировании коэффициентов преобразования, унифицированное (одно и то же) извлечение параметра Райса (RicePara) используется для сигнализации синтаксиса `abs_remainder` и `dec_abs_level`. Единственное различие заключается в том, что базовый уровень, *baseLevel*, задается равным 4 и 0 для кодирования `abs_remainder` и `dec_abs_level`, соответственно. Параметр Райса определяется на основе не только суммы абсолютных уровней соседних пяти коэффициентов преобразования в локальном эталоне, но также и соответствующего базового уровня, следующим образом:

[00167] $\text{RicePara} = \text{RiceParTable}[\max(\min(31, \text{sumAbs} - 5 * \text{baseLevel}), 0)]$

[00168] Другими словами, двоичные кодовые слова для синтаксических элементов `abs_remainder` и `dec_abs_level` определяются адаптивно согласно информации уровня соседних коэффициентов. Поскольку это определение кодовых слов выполняется для каждой выборки, оно требует дополнительных логик, чтобы обрабатывать эту адаптацию кодового слова для кодирования коэффициентов.

[00169] Аналогично, при кодировании остаточного блока в режиме пропуска преобразования, двоичные кодовые слова для синтаксических элементов *abs_remainder* определяются адаптивно согласно информации уровня соседних остаточных выборок.

[00170] Кроме того, при кодировании синтаксических элементов, связанных с остаточным кодированием или кодированием коэффициентов преобразования, выбор вероятностных моделей зависит от информации уровня соседних уровней, что требует дополнительных логик и дополнительных контекстных моделей.

[00171] В текущем проектном решении, преобразование в двоичную форму выборок перехода извлекается посредством активации процесса экспоненциального преобразования в двоичную форму Голомба третьего порядка. Имеется запас для дополнительного повышения его производительности.

[00172] В текущем VVC, две разных схемы отображения уровней доступны и применены к регулярному преобразованию и пропуску преобразования, соответственно.

Каждая схема отображения уровней ассоциирована с различными условиями, функцией отображения и позицией отображения. Для блоков, в которых регулярное преобразование применяется, схема отображения уровней используется после того, как число контекстно-кодированных элементов разрешения (CCB) превышает предел.

- 5 Позиция отображения, обозначаемая в качестве ZeroPos[n], и результат отображения, обозначаемый в качестве AbsLevel[xC][yC], извлекаются так, как указано в таблице 2. Для блоков, в которых пропуск преобразования применяется, другая схема отображения уровней используется до того, как число контекстно-кодированных элементов разрешения (CCB) превышает предел. Позиция отображения, обозначаемая в качестве predCoeff, и результат отображения, обозначаемый в качестве AbsLevel[xC][yC],
10 извлекаются так, как указано в таблице 5. Такое неунифицированное проектное решение может не быть оптимальным с точки зрения стандартизации.

- [00173] Для профилей выше 10-битовых в HEVC, extended_precision_processing_flag, равный 1, указывает то, что расширенный динамический диапазон используется для
15 синтаксического анализа коэффициентов и обработки обратного преобразования. В текущем VVC, остаточное кодирование для коэффициентов преобразования или кодирование с пропуском преобразования выше 10-битового сообщается в качестве причины значительного уменьшения производительности. Имеется запас для дополнительного повышения его производительности.

- 20 [00174] *Предложенные способы*

[00175] В этом раскрытии, несколько способов предлагаются для того, чтобы разрешать проблемы, упомянутые в разделе "Улучшения остаточного кодирования и кодирования коэффициентов". Следует отметить, что следующие способы могут применяться независимо или объединенно.

- 25 [00176] Согласно первому аспекту раскрытия, предлагается использовать фиксированный набор двоичных кодовых слов для кодирования некоторых синтаксических элементов, например, abs_remainder, при остаточном кодировании. Двоичные кодовые слова могут формироваться с использованием различных способов. Некоторые примерные способы перечисляются следующим образом.

- 30 [00177] Во-первых, используется процедура, такая же процедура для определения кодового слова для abs_remainder, как используемая в текущем VVC, но всегда с выбранным фиксированным параметром Райса (например, 1, 2 или 3).

[00178] Во-вторых, преобразование в двоичную форму кодом фиксированной длины.

[00179] В-третьих, преобразование в двоичную форму усеченным кодом Райса.

- 35 [00180] В-четвертых, процесс усеченного двоичного (ТВ) преобразования в двоичную форму.

[00181] В-пятых, процесс экспоненциального преобразования в двоичную форму Голомба k-ого порядка (EGk).

- 40 [00182] В-шестых, ограниченное экспоненциальное преобразование в двоичную форму Голомба k-ого порядка.

- [00183] Согласно второму аспекту раскрытия, предлагается использовать фиксированный набор кодовых слов для кодирования некоторых синтаксических элементов, например, abs_remainder и dec_abs_level, при кодировании коэффициентов преобразования. Двоичные кодовые слова могут формироваться с использованием
45 различных способов. Некоторые примерные способы перечисляются следующим образом.

[00184] Во-первых, используется такая же процедура для определения кодовых слов для abs_remainder и dec_abs_level, как используемая в текущем VVC, но с фиксированным

параметром Райса, например, 1, 2 или 3. Значение `baseLevel` по-прежнему может отличаться для `abs_remainder` и `dec_abs_level`, используемых в текущем VVC (например, `baseLevel` задается равным 4 и 0 для кодирования `abs_remainder` и `dec_abs_level`, соответственно).

5 [00185] Во-вторых, используется такая же процедура для определения кодовых слов для `abs_remainder` и `dec_abs_level`, как используемая в текущем VVC, но с фиксированным параметром Райса, например, 1, 2 или 3. Значение `baseLevel` для `abs_remainder` и `dec_abs_level` выбирается одинаковым, например, оба из них используют 0 либо оба из них используют 4.

10 [00186] В-третьих, преобразование в двоичную форму кодом фиксированной длины.

[00187] В-четвертых, преобразование в двоичную форму усеченным кодом Райса.

[00188] В-пятых, процесс усеченного двоичного (ТВ) преобразования в двоичную форму.

[00189] В-шестых, процесс экспоненциального преобразования в двоичную форму

15 Голомба k -ого порядка (EGk).

[00190] В-седьмых ограниченное экспоненциальное преобразование в двоичную форму Голомба k -ого порядка

[00191] Согласно третьему аспекту раскрытия, предлагается использовать один контекст для кодирования синтаксических элементов, связанных с остаточным

20 кодированием или кодированием коэффициентов (например, `abs_level_gtx_flag`), и выбор контекста на основе соседней декодированной информации уровня может удаляться.

[00192] Согласно четвертому аспекту раскрытия, предлагается использовать наборы переменных двоичных кодовых слов для кодирования некоторых синтаксических элементов, например, `abs_remainder`, при остаточном кодировании, и выбор набора

25 двоичных кодовых слов определяется согласно некоторой кодированной информации текущего блока, например, параметру квантования (QP), ассоциированному с ТВ/СВ и/или срезом, режимам прогнозирования CU (например, ИВС-режим или внутренний или взаимный) и/или типу среза (например, I-срез, P-срез или B-срез). Различные способы могут использоваться для того, чтобы извлекать наборы переменных двоичных кодовых слов, при этом некоторые примерные способы перечисляются следующим образом.

30 [00193] Во-первых, используется такая же процедура для определения кодового слова для `abs_remainder`, как используемая в текущем VVC, но с различными параметрами Райса.

[00194] Во-вторых, процесс экспоненциального преобразования в двоичную форму

35 Голомба k -ого порядка (EGk)

[00195] В-третьих, ограниченное экспоненциальное преобразование в двоичную форму Голомба k -ого порядка

Таблица 6. Определение параметра Райса на основе QP-значения

40	<pre> if(QP_{CU}<TH1) { rice parameter=K0 } else if(QP_{CU}<TH2) { rice parameter=K1 } </pre>
45	<pre> else if(QP_{CU}<TH3) { rice parameter=K2 } else if(QP_{CU}<TH4) { rice parameter=K3 } </pre>

```

    }
    else
    {
        rice parameter=K4
    }

```

[00196] Такие же способы, поясненные в четвертом аспекте, также являются применимыми к кодированию коэффициентов преобразования. Согласно пятому аспекту раскрытия, предлагается использовать наборы переменных двоичных кодовых слов для кодирования некоторых синтаксических элементов, например, `abs_remainder` и `dec_abs_level`, при кодировании коэффициентов преобразования, и выбор набора двоичных кодовых слов определяется некоторой закодированной информацией текущего блока, например, параметру квантования (QP), ассоциированному с ТВ/СВ и/или срезом, режимам прогнозирования CU (например, IBC-режим или внутренний или взаимный) и/или типу среза (например, I-срез, P-срез или B-срез). С другой стороны, различные способы могут использоваться для того, чтобы извлекать наборы переменных двоичных кодовых слов, при этом некоторые примерные способы перечисляются следующим образом.

[00197] Во-первых, используется такая же процедура для определения кодового слова для `abs_remainder`, как используемая в текущем VVC, но с различными параметрами Райса.

[00198] Во-вторых, процесс экспоненциального преобразования в двоичную форму Голомба k-ого порядка (EGk).

[00199] В-третьих, ограниченное экспоненциальное преобразование в двоичную форму Голомба k-ого порядка.

[00200] В этих вышеприведенных способах, различные параметры Райса могут использоваться для того, чтобы извлекать различный набор двоичных кодовых слов. Для данного блока остаточных выборок, используемые параметры Райса определяются согласно CU QP, обозначаемому в качестве QP_{CU} , вместо соседней информации уровня. Один конкретный пример проиллюстрирован так, как показано в таблице 6, где $TH1$ - $TH4$ являются предварительно заданными пороговыми значениями, удовлетворяющими ($TH1 < TH2 < TH3 < TH4$), и $K0$ - $K4$ представляют собой предварительно заданные параметры Райса. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса, как показано в таблице 6, из QP-значения текущей CU.

[00201] Согласно пятому аспекту раскрытия, набор параметров и/или пороговых значений, ассоциированных с определением кодовых слов для синтаксических элементов кодирования коэффициентов преобразования и/или остаточного кодирования с пропуском преобразования, сигнализируется в поток битов. Определенные кодовые слова используются в качестве кодовых слов преобразования в двоичную форму при кодировании синтаксических элементов через энтропийный кодер, например, через арифметическое кодирование.

[00202] Следует отметить, что набор параметров и/или пороговых значений может представлять собой полный набор или поднабор всех параметров и пороговых значений, ассоциированных с определением кодовых слов для синтаксических элементов. Набор параметров и/или пороговых значений может сигнализироваться на различных уровнях в потоке видеобитов. Например, они могут сигнализироваться на уровне последовательности (например, в наборе параметров последовательности), на уровне картинки (например, в наборе параметров картинки и/или в заголовке картинки), на

уровне среза (например, в заголовке среза), на уровне единицы дерева кодирования (CTU) или на уровне единицы кодирования (CU).

[00203] В одном примере, параметр Райса, используемый для того, чтобы определять кодовые слова для кодирования `abs_remainder` синтаксис при остаточном кодировании с пропуском преобразования, сигнализируется в заголовке среза, заголовке картинки, PPS и/или SPS. Сигнализируемый параметр Райса используется для того, чтобы определять кодовое слово для кодирования синтаксиса `abs_remainder`, когда CU кодируется в качестве режима пропуска преобразования, и CU ассоциирована с вышеуказанным заголовком среза, заголовком картинки, PPS и/или SPS и т.д.

[00204] Согласно шестому аспекту раскрытия, набор параметров и/или пороговых значений, ассоциированных с определением кодовых слов, как проиллюстрировано в первом и втором аспектах, используется для синтаксических элементов кодирования коэффициентов преобразования и/или остаточного кодирования с пропуском преобразования. Кроме того, различные наборы могут использоваться согласно тому, содержит либо нет текущий блок остаток/коэффициенты сигналов яркости или остаток/коэффициенты сигналов цветности. Определенные кодовые слова используются в качестве кодовых слов преобразования в двоичную форму при кодировании синтаксических элементов через энтропийный кодер, например, через арифметическое кодирование.

[00205] В одном примере, кодовое слово для `abs_remainder`, ассоциированного с остаточным кодированием с преобразованием, используемым в текущем VVC, используется для блоков сигналов яркости и сигналов цветности, но различные фиксированные параметры Райса используются посредством блока сигналов яркости и блока сигналов цветности, соответственно (например, K1 для блока сигналов яркости, K2 для блока сигналов цветности, где K1 и K2 являются целыми числами),

[00206] Согласно седьмому аспекту раскрытия, набор параметров и/или пороговых значений, ассоциированных с определением кодовых слов для синтаксических элементов кодирования коэффициентов преобразования и/или остаточного кодирования с пропуском преобразования, сигнализируется в поток битов. Кроме того, различные наборы могут сигнализироваться для блоков сигналов яркости и сигналов цветности. Определенные кодовые слова используются в качестве кодовых слов преобразования в двоичную форму при кодировании синтаксических элементов через энтропийный кодер, например, через арифметическое кодирование.

[00207] Такие же способы, поясненные в вышеприведенных аспектах, также являются применимыми к кодированию значений перехода в палитровом режиме, например, `palette_escape_val`.

[00208] Согласно восьмому аспекту раскрытия, различные k-ые порядки экспоненциального преобразования в двоичную форму Голомба могут использоваться для того, чтобы извлекать различный набор двоичных кодовых слов для кодирования значений перехода в палитровом режиме. В одном примере, для данного блока выборка перехода, используемый параметр экспоненциального преобразования в двоичную форму Голомба, т.е. значение k, определяется согласно QP-значению блока, обозначаемому в качестве QP_{CU} . Такой же пример, как проиллюстрировано в таблице 6, может использоваться при извлечении значения параметра k на основе данного QP-значения блока. Хотя в этом примере четыре различных пороговых значения (из TH1-TH4) перечисляются, и пять различных значений k (из K0-K4) могут извлекаться на основе этих пороговых значений и QP_{CU} , заслуживает внимание тот факт, что число пороговых значений служит только для иллюстрации. На практике, другое число

пороговых значений может использоваться для того, чтобы сегментировать целый диапазон QR-значений на другое число сегментов QR-значений, и для каждого сегмента QR-значений, различное значение k может использоваться для того, чтобы извлекать соответствующие двоичные кодовые слова для кодирования значений перехода блока, который кодируется в палитровом режиме. Также необходимо отметить, что одниковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

[00209] Согласно девятому аспекту раскрытия, набор параметров и/или пороговых значений, ассоциированных с определением кодовых слов для синтаксических элементов выборки перехода, сигнализируется в поток битов. Определенные кодовые слова используются в качестве кодовых слов преобразования в двоичную форму при кодировании синтаксических элементов выборок перехода через энтропийный кодер, например, через арифметическое кодирование.

[00210] Следует отметить, что набор параметров и/или пороговых значений может представлять собой полный набор или поднабор всех параметров и пороговых значений, ассоциированных с определением кодовых слов для синтаксических элементов. Набор параметров и/или пороговых значений может сигнализироваться на различных уровнях в потоке видеобитов. Например, они могут сигнализироваться на уровне последовательности (например, в наборе параметров последовательности), на уровне картинки (например, в наборе параметров картинки и/или в заголовке картинки), на уровне среза (например, в заголовке среза), на уровне единицы дерева кодирования (CTU) или на уровне единицы кодирования (CU).

[00211] В одном примере согласно аспекту, k -ые порядки экспоненциального преобразования в двоичную форму Голомба используются для того, чтобы определять кодовые слова для кодирования синтаксиса `palette_escape_val` в палитровом режиме, и значение k сигнализируется в потоке битов в декодер. Значение k может сигнализироваться на различных уровнях, например, оно может сигнализироваться в заголовке среза, заголовке картинки, PPS и/или SPS и т.д. Сигнализируемый параметр экспоненциального преобразования в двоичную форму Голомба используется для того, чтобы определять кодовое слово для кодирования синтаксиса `palette_escape_val`, когда CU кодируется в качестве палитрового режима, и CU ассоциирована с вышеуказанным заголовком среза, заголовком картинки, PPS и/или SPS и т.д.

[00212] *Гармонизация отображения уровней для режима пропуска преобразования и режима регулярного преобразования*

[00213] Согласно десятому аспекту раскрытия, одно и то же условие для применения отображения уровней используется как для режима пропуска преобразования, так и для режима регулярного преобразования. В одном примере, предлагается применять отображение уровней после того, как число контекстно-кодированных элементов разрешения (CCB) превышает предел как для режима пропуска преобразования, так и для режима регулярного преобразования. В другом примере, предлагается применять отображение уровней до того, как число контекстно-кодированных элементов разрешения (CCB) превышает предел как для режима пропуска преобразования, так и для режима регулярного преобразования.

[00214] Согласно одиннадцатому аспекту раскрытия, такой же способ для извлечения позиции отображения при отображении уровней используется как для режима пропуска преобразования, так и для режима регулярного преобразования. В одном примере, также предлагается применять способ извлечения позиции отображения при

отображении уровней, который используется в режиме пропуска преобразования, к режиму регулярного преобразования. В другом примере, также предлагается применять способ извлечения позиции отображения при отображении уровней, который используется в режиме регулярного преобразования, к режиму пропуска преобразования.

[00215] Согласно двенадцатому аспекту раскрытия, один и тот же способ отображения уровней применяется как к режиму пропуска преобразования, так и к режиму регулярного преобразования. В одном примере, также предлагается применять функцию отображения уровней, которая используется в режиме пропуска преобразования, к режиму регулярного преобразования. В другом примере, также предлагается применять функцию отображения уровней, которая используется в режиме регулярного преобразования, к режиму пропуска преобразования.

[00216] Упрощение извлечения параметра Райса при остаточном кодировании

[00217] Согласно тринадцатому аспекту раскрытия, предлагается использовать простую логику, такую как операция сдвига или деления, вместо таблицы поиска для извлечения параметра Райса при кодировании синтаксического элемента `abs_remainder/dec_abs_level` с использованием кода Голомба-Райса. Согласно настоящему раскрытию, таблица поиска, как указано в таблице 4, может удаляться. В одном примере, параметр `cRiceParam` Райса извлекается следующим образом: `cRiceParam=(locSumAbs>>n)`, где `n` является положительным числом, например, 3. Необходимо отметить, что на практике другие отличающиеся логики могут использоваться для того, чтобы достигать одинаковых результатов, например, операция деления на значение, равное 2 в степени `n`. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 7. Процесс извлечения параметра Райса

30	Входы в этот процесс представляют собой базовый уровень <code>baseLevel</code>, индекс <code>cIdx</code> цветового компонента, местоположение (<code>x0</code>, <code>y0</code>) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (<code>xC</code>, <code>yC</code>) сканирования коэффициентов, двоичный логарифм <code>log2TbWidth</code> ширины блока преобразования и двоичный логарифм <code>log2TbHeight</code> высоты блока преобразования.
	Вывод этого процесса представляет собой параметр <code>cRiceParam</code> Райса.
35	С учетом массива <code>AbsLevel[x][y]</code> для блока преобразования с индексом <code>cIdx</code> компонента и левым верхним местоположением (<code>x0</code>, <code>y0</code>) сигнала яркости, переменная <code>locSumAbs</code> извлекается так, как указано посредством следующего псевдокода:
40	<pre> locSumAbs=0 if(xC<(1<<log2TbWidth)-1) { locSumAbs+=AbsLevel[xC+1][yC] if(xC<(1<<log2TbWidth)-2) locSumAbs+=AbsLevel[xC+2][yC] if(yC<(1<<log2TbHeight)-1) locSumAbs+=AbsLevel[xC+1][yC+1] (1494) } if(yC<(1<<log2TbHeight)-1) { locSumAbs+=AbsLevel[xC][yC+1] if(yC<(1<<log2TbHeight)-2) locSumAbs+=AbsLevel[xC][yC+2] } locSumAbs=Clip3(0, 31, locSumAbs-baseLevel*5) </pre>
	С учетом переменной <code>locSumAbs</code>, параметр <code>cRiceParam</code> Райса извлекается следующим образом: <i>указано в таблице 4.</i>
	<i><code>cRiceParam=(locSumAbs>>3)</code></i>
	Когда <code>baseLevel</code> равен 0, переменная <code>ZeroPos[n]</code> извлекается следующим образом:
	<i><code>ZeroPos[n]=(QState<2?1:2)<<cRiceParam</code></i>

[00218] Согласно четырнадцатому аспекту раскрытия, предлагается использовать меньшее количество соседних позиций для извлечения параметра Райса при кодировании синтаксического элемента `abs_remainder/dec_abs_level` с использованием кода Голомба-Райса. В одном примере, предлагается использовать только 2 соседних позиции для извлечения параметра Райса при кодировании синтаксического элемента `abs_remainder/dec_abs_level`. Соответствующий процесс декодирования на основе VVC-проекта

проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 8. Процесс извлечения параметра Райса

Входы в этот процесс представляют собой базовый уровень *baseLevel*, индекс *cIdx* цветового компонента, местоположение (*x0*, *y0*) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (*xC*, *yC*) сканирования коэффициентов, двоичный логарифм *log2TbWidth* ширины блока преобразования и двоичный логарифм *log2TbHeight* высоты блока преобразования.

Вывод этого процесса представляет собой параметр *cRiceParam* Райса.

С учетом массива *AbsLevel[x][y]* для блока преобразования с индексом *cIdx* компонента и левым верхним местоположением (*x0*, *y0*) сигнала яркости, переменная *locSumAbs* извлекается так, как указано посредством следующего псевдокода:

```
locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
locSumAbs+=AbsLevel[xC+1][yC]
}
if(yC<(1<<log2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]
}
locSumAbs=Clip3(0, 31, locSumAbs-baseLevel*52)
```

С учетом переменной *locSumAbs*, параметр *cRiceParam* Райса извлекается так, как указано в таблице 4.

Когда *baseLevel* равен 0, переменная *ZeroPos[n]* извлекается следующим образом:

```
ZeroPos[n]=(QState<2?1:2)<<cRiceParam
```

[00219] В другом примере, предлагается использовать только одну соседнюю позицию для извлечения параметра Райса при кодировании синтаксического элемента *abs_remainder/dec_abs_level*. Соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 9. Процесс извлечения параметра Райса

Входы в этот процесс представляют собой базовый уровень *baseLevel*, индекс *cIdx* цветового компонента, местоположение (*x0*, *y0*) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (*xC*, *yC*) сканирования коэффициентов, двоичный логарифм *log2TbWidth* ширины блока преобразования и двоичный логарифм *log2TbHeight* высоты блока преобразования.

Вывод этого процесса представляет собой параметр *cRiceParam* Райса.

С учетом массива *AbsLevel[x][y]* для блока преобразования с индексом *cIdx* компонента и левым верхним местоположением (*x0*, *y0*) сигнала яркости, переменная *locSumAbs* извлекается так, как указано посредством следующего псевдокода:

```
locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
locSumAbs+=AbsLevel[xC+1][yC]
if(xC<(1<<log2TbWidth)-2)
}
locSumAbs=Clip3(0, 31, locSumAbs-baseLevel)
```

С учетом переменной *locSumAbs*, параметр *cRiceParam* Райса извлекается так, как указано в таблице 4.

Когда *baseLevel* равен 0, переменная *ZeroPos[n]* извлекается следующим образом:

```
ZeroPos[n]=(QState<2?1:2)<<cRiceParam
```

[00220] Согласно пятнадцатому аспекту раскрытия, предлагается использовать различные логики, чтобы регулировать значение *locSumAbs* на основе значения *baseLevel* для извлечения параметра Райса при кодировании синтаксического элемента *abs_remainder/dec_abs_level* с использованием кода Голомба-Райса. В одном примере, дополнительные операции масштабирования и смещения применяются в форме " $(locSumAbs-baseLevel*5)*\alpha+\beta$ ". Когда *alpha* принимает значение 1,5, и *beta* принимает значение 1, соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 10. Процесс извлечения параметра Райса

Входы в этот процесс представляют собой базовый уровень *baseLevel*, индекс *cIdx* цветового компонента, местоположение (*x0*, *y0*) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (*xC*, *yC*) сканирования коэффициентов, двоичный логарифм *log2TbWidth* ширины блока преобразования и двоичный логарифм *log2TbHeight* высоты блока преобразования.

Вывод этого процесса представляет собой параметр *cRiceParam* Райса.

С учетом массива *AbsLevel[x][y]* для блока преобразования с индексом *cIdx* компонента и левым верхним местоположением (*x0*, *y0*) сигнала яркости, переменная *locSumAbs* извлекается так, как указано посредством следующего псевдокода:

```
locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
locSumAbs+=AbsLevel[xC+1][yC]
if(xC<(1<<log2TbWidth)-2)
```

```

locSumAbs+=AbsLevel[xC+2][yC]
if(yC<(1<<log2TbHeight)-1)
locSumAbs+=AbsLevel[xC+1][yC+1]
(1494)
}
if(yC<(1<<log2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]
if(yC<(1<<log2TbHeight)-2)
locSumAbs+=AbsLevel[xC][yC+2]
}
locSumAbs=Clip3(0, 31, (locSumAbs-baseLevel*5)*1,5+1)

```

С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается так, как указано в таблице 4.
 Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:
 ZeroPos[n]=(QState<2?1:2)<<cRiceParam

[00221] Согласно шестнадцатому аспекту раскрытия, предлагается удалять операции отсечения для извлечения параметра Райса в синтаксическом элементе abs_remainder/dec_abs_level с использованием кода Голомба-Райса. Согласно настоящему раскрытию, пример процесса декодирования на VVC-проекте проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 11. Процесс извлечения параметра Райса

Вводы в этот процесс представляют собой базовый уровень baseLevel, индекс cIdx цветового компонента, местоположение (x0, y0) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм log2TbWidth ширины блока преобразования и двоичный логарифм log2TbHeight высоты блока преобразования.

Вывод этого процесса представляет собой параметр cRiceParam Райса.

С учетом массива AbsLevel[x][y] для блока преобразования с индексом cIdx компонента и левым верхним местоположением (x0, y0) сигнала яркости, переменная locSumAbs извлекается так, как указано посредством следующего псевдокода:

```

locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
locSumAbs+=AbsLevel[xC+1][yC]
if(xC<(1<<log2TbWidth)-2)
locSumAbs+=AbsLevel[xC+2][yC]
if(yC<(1<<log2TbHeight)-1)
locSumAbs+=AbsLevel[xC+1][yC+1]
(1494)
}
if(yC<(1<<log2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]
if(yC<(1<<log2TbHeight)-2)
locSumAbs+=AbsLevel[xC][yC+2]
}
locSumAbs=locSumAbs-baseLevel*5

```

С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается следующим образом: *указано в таблице 4.*
cRiceParam=(locSumAbs>>3)

Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:
 ZeroPos[n]=(QState<2?1:2)<<cRiceParam

[00222] Согласно настоящему раскрытию, пример процесса декодирования на VVC-проекте проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 12. Процесс извлечения параметра Райса

Вводы в этот процесс представляют собой базовый уровень baseLevel, индекс cIdx цветового компонента, местоположение (x0, y0) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм log2TbWidth ширины блока преобразования и двоичный логарифм log2TbHeight высоты блока преобразования.

Вывод этого процесса представляет собой параметр cRiceParam Райса.

С учетом массива AbsLevel[x][y] для блока преобразования с индексом cIdx компонента и левым верхним местоположением (x0, y0) сигнала яркости, переменная locSumAbs извлекается так, как указано посредством следующего псевдокода:

```

locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
locSumAbs+=AbsLevel[xC+1][yC]
if(xC<(1<<log2TbWidth)-2)
locSumAbs+=AbsLevel[xC+2][yC]
if(yC<(1<<log2TbHeight)-1)
locSumAbs+=AbsLevel[xC+1][yC+1]
(1494)
}
if(yC<(1<<log2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]
if(yC<(1<<log2TbHeight)-2)
locSumAbs+=AbsLevel[xC][yC+2]
}

```

```

    }
    locSumAbs=Min(31, locSumAbs-baseLevel*5)
    С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается так, как указано в таблице 4.
    Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:
    ZeroPos[n]=(QState<2?1:2)<<cRiceParam

```

[00223] Согласно семнадцатому аспекту раскрытия, предлагается изменять начальное значение locSumAbs с 0 на ненулевое целое число для извлечения параметра Райса при кодировании синтаксического элемента abs_remainder/dec_abs_level с использованием кода Голомба-Райса. В одном примере, начальное значение 1 назначается locSumAbs, и соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 13. Процесс извлечения параметра Райса

Вводы в этот процесс представляют собой базовый уровень baseLevel, индекс cIdx цветового компонента, местоположение (x0, y0) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм log2TbWidth ширины блока преобразования и двоичный логарифм log2TbHeight высоты блока преобразования.

Вывод этого процесса представляет собой параметр cRiceParam Райса.

С учетом массива AbsLevel[x][y] для блока преобразования с индексом cIdx компонента и левым верхним местоположением (x0, y0) сигнала яркости, переменная locSumAbs извлекается так, как указано посредством следующего псевдокода:

```

    locSumAbs=1
    if(xC<(1<<log2TbWidth)-1) {
    locSumAbs+=AbsLevel[xC+1][yC]
    if(xC<(1<<log2TbWidth)-2)
    locSumAbs+=AbsLevel[xC+2][yC]
    if(yC<(1<<log2TbHeight)-1)
    locSumAbs+=AbsLevel[xC+1][yC+1]
    (1494)
    }
    if(yC<(1<<log2TbHeight)-1) {
    locSumAbs+=AbsLevel[xC][yC+1]
    if(yC<(1<<log2TbHeight)-2)
    locSumAbs+=AbsLevel[xC][yC+2]
    }

```

```

    locSumAbs=Clip3(0, 31, locSumAbs-baseLevel*5)

```

С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается так, как указано в таблице 4.

Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:

```

    ZeroPos[n]=(QState<2?1:2)<<cRiceParam

```

[00224] Согласно восемнадцатому аспекту раскрытия, предлагается использовать максимальное значение значений уровня соседних позиций вместо их значения суммы для извлечения параметра Райса при кодировании синтаксического элемента abs_remainder/dec_abs_level с использованием кода Голомба-Райса. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 14. Процесс извлечения параметра Райса

Вводы в этот процесс представляют собой базовый уровень baseLevel, индекс cIdx цветового компонента, местоположение (x0, y0) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм log2TbWidth ширины блока преобразования и двоичный логарифм log2TbHeight высоты блока преобразования.

Вывод этого процесса представляет собой параметр cRiceParam Райса.

С учетом массива AbsLevel[x][y] для блока преобразования с индексом cIdx компонента и левым верхним местоположением (x0, y0) сигнала яркости, переменная locSumAbs извлекается так, как указано посредством следующего псевдокода:

```

    locSumAbs=0
    if(xC<(1<<log2TbWidth)-1) {
    locSumAbs+=Max(AbsLevel[xC+1][yC], locSumAbs)
    if(xC<(1<<log2TbWidth)-2)
    locSumAbs+=Max(AbsLevel[xC+2][yC], locSumAbs)
    if(yC<(1<<log2TbHeight)-1)
    locSumAbs+=Max(AbsLevel[xC+1][yC+1], locSumAbs) (1494)
    }
    if(yC<(1<<log2TbHeight)-1) {
    locSumAbs+=Max(AbsLevel[xC][yC+1], locSumAbs)
    if(yC<(1<<log2TbHeight)-2)
    locSumAbs+=Max(AbsLevel[xC][yC+2], locSumAbs)
    }
    locSumAbs=Clip3(0, 31, locSumAbs-baseLevel*5)

```

С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается так, как указано в таблице 4.
 Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:
 $ZeroPos[n] = (QState < 2^{?1:2}) < cRiceParam$

[00225] Согласно девятнадцатому аспекту раскрытия, предлагается извлекать параметр Райса на основе относительной амплитуды каждого значения AbsLevel в соседних позициях и значения базового уровня, при кодировании синтаксического элемента abs_remainder/dec_abs_level с использованием кода Голомба-Райса. В одном примере, параметр Райса извлекается на основе того, сколько из значений AbsLevel в соседних позициях больше базового уровня. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 15. Процесс извлечения параметра Райса

Вводы в этот процесс представляют собой базовый уровень baseLevel, индекс cIdx цветового компонента, местоположение (x0, y0) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм log2TbWidth ширины блока преобразования и двоичный логарифм log2TbHeight высоты блока преобразования.

Вывод этого процесса представляет собой параметр cRiceParam Райса.

С учетом массива AbsLevel[x][y] для блока преобразования с индексом cIdx компонента и левым верхним местоположением (x0, y0) сигнала яркости, переменная locSumAbs извлекается так, как указано посредством следующего псевдокода:

```
locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
  locSumAbs+=(AbsLevel[xC+1][yC]>baseLevel? 1:0)
if(xC<(1<<log2TbWidth)-2)
  locSumAbs+=(AbsLevel[xC+2][yC]>baseLevel? 1:0)
if(yC<(1<<log2TbHeight)-1)
  locSumAbs+=(AbsLevel[xC+1][yC+1]>baseLevel? 1:0) (1494)
}
if(yC<(1<<log2TbHeight)-1) {
  locSumAbs+=(AbsLevel[xC][yC+1]>baseLevel? 1:0)
if(yC<(1<<log2TbHeight)-2)
  locSumAbs+=(AbsLevel[xC][yC+2]>baseLevel? 1:0)
}
```

С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается следующим образом: *указано в таблице 4.*

cRiceParam=locSumAbs

Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:

$ZeroPos[n] = (QState < 2^{?1:2}) < cRiceParam$

[00226] В другом примере, параметр Райса извлекается на основе суммы значений (AbsLevel-baseLevel) для этих соседних позиций, значения AbsLevel которых больше базового уровня. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 16. Процесс извлечения параметра Райса

Вводы в этот процесс представляют собой базовый уровень baseLevel, индекс cIdx цветового компонента, местоположение (x0, y0) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм log2TbWidth ширины блока преобразования и двоичный логарифм log2TbHeight высоты блока преобразования.

Вывод этого процесса представляет собой параметр cRiceParam Райса.

С учетом массива AbsLevel[x][y] для блока преобразования с индексом cIdx компонента и левым верхним местоположением (x0, y0) сигнала яркости, переменная locSumAbs извлекается так, как указано посредством следующего псевдокода:

```
locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
  locSumAbs+=Max(0, AbsLevel[xC+1][yC]-baseLevel)
if(xC<(1<<log2TbWidth)-2)
  locSumAbs+=Max(0, AbsLevel[xC+2][yC]-baseLevel)
if(yC<(1<<log2TbHeight)-1)
  locSumAbs+=Max(0, AbsLevel[xC+1][yC+1]-baseLevel) (1494)
}
if(yC<(1<<log2TbHeight)-1) {
  locSumAbs+=Max(0, AbsLevel[xC][yC+1]-baseLevel)
if(yC<(1<<log2TbHeight)-2)
  locSumAbs+=Max(0, AbsLevel[xC][yC+2]-baseLevel)
}
locSumAbs=Min(31, locSumAbs)
```

С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается так, как указано в таблице 4.

Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:

$ZeroPos[n] = (QState < 2^{?1:2}) < cRiceParam$

[00227] Согласно настоящему раскрытию, один пример процесса декодирования на VVC-проекте проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 17. Процесс извлечения параметра Райса

Входы в этот процесс представляют собой базовый уровень *baseLevel*, индекс *cIdx* цветового компонента, местоположение (*x0*, *y0*) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (*xC*, *yC*) сканирования коэффициентов, двоичный логарифм *log2TbWidth* ширины блока преобразования и двоичный логарифм *log2TbHeight* высоты блока преобразования.

Вывод этого процесса представляет собой параметр *cRiceParam* Райса.

С учетом массива *AbsLevel[x][y]* для блока преобразования с индексом *cIdx* компонента и левым верхним местоположением (*x0*, *y0*) сигнала яркости, переменная *locSumAbs* извлекается так, как указано посредством следующего псевдокода:

```

locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
locSumAbs+=AbsLevel[xC+1][yC]-baseLevel
if(xC<(1<<log2TbWidth)-2)
locSumAbs+=AbsLevel[xC+2][yC]-baseLevel
if(yC<(1<<log2TbHeight)-1)
locSumAbs+=AbsLevel[xC+1][yC+1]-baseLevel (1494)
}
if(yC<(1<<log2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]-baseLevel
if(yC<(1<<log2TbHeight)-2)
locSumAbs+=AbsLevel[xC][yC+2]-baseLevel
}
locSumAbs=Clip3(0, 31, locSumAbs)

```

С учетом переменной *locSumAbs*, параметр *cRiceParam* Райса извлекается так, как указано в таблице 4.

Когда *baseLevel* равен 0, переменная *ZeroPos[n]* извлекается следующим образом:

```

ZeroPos[n]=(QState<2?1:2)<<cRiceParam

```

[00228] Упрощение извлечения позиций отображения уровней при остаточном кодировании

[00229] Согласно двадцатому аспекту раскрытия, предлагается удалять *QState* из извлечения *ZeroPos[n]*, так что *ZeroPos[n]* извлекается только из *cRiceParam*. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 18. Процесс извлечения параметра Райса

Входы в этот процесс представляют собой базовый уровень *baseLevel*, индекс *cIdx* цветового компонента, местоположение (*x0*, *y0*) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (*xC*, *yC*) сканирования коэффициентов, двоичный логарифм *log2TbWidth* ширины блока преобразования и двоичный логарифм *log2TbHeight* высоты блока преобразования.

Вывод этого процесса представляет собой параметр *cRiceParam* Райса.

С учетом массива *AbsLevel[x][y]* для блока преобразования с индексом *cIdx* компонента и левым верхним местоположением (*x0*, *y0*) сигнала яркости, переменная *locSumAbs* извлекается так, как указано посредством следующего псевдокода:

```

locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
locSumAbs+=AbsLevel[xC+1][yC]
if(xC<(1<<log2TbWidth)-2)
locSumAbs+=AbsLevel[xC+2][yC]
if(yC<(1<<log2TbHeight)-1)
locSumAbs+=AbsLevel[xC+1][yC+1]
(1494)
}
if(yC<(1<<log2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]
if(yC<(1<<log2TbHeight)-2)
locSumAbs+=AbsLevel[xC][yC+2]
}
locSumAbs=Clip3(0, 31, locSumAbs-baseLevel*5)

```

С учетом переменной *locSumAbs*, параметр *cRiceParam* Райса извлекается так, как указано в таблице 4.

Когда *baseLevel* равен 0, переменная *ZeroPos[n]* извлекается следующим образом:

```

ZeroPos[n]=2<<cRiceParam

```

[00230] Согласно двадцать первому аспекту раскрытия, предлагается извлекать *ZeroPos[n]* на основе значения *locSumAbs*. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 19. Процесс извлечения параметра Райса

Вводы в этот процесс представляют собой базовый уровень *baseLevel*, индекс *cIdx* цветового компонента, местоположение (*x0*, *y0*) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (*xC*, *yC*) сканирования коэффициентов, двоичный логарифм $\log_2 TbWidth$ ширины блока преобразования и двоичный логарифм $\log_2 TbHeight$ высоты блока преобразования.

Вывод этого процесса представляет собой параметр *cRiceParam* Райса.

С учетом массива *AbsLevel[x][y]* для блока преобразования с индексом *cIdx* компонента и левым верхним местоположением (*x0*, *y0*) сигнала яркости, переменная *locSumAbs* извлекается так, как указано посредством следующего псевдокода:

```
locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
locSumAbs+=AbsLevel[xC+1][yC]
if(xC<(1<<log2TbWidth)-2)
locSumAbs+=AbsLevel[xC+2][yC]
if(yC<(1<<log2TbHeight)-1)
locSumAbs+=AbsLevel[xC+1][yC+1]
(1494)
}
if(yC<(1<<log2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]
if(yC<(1<<log2TbHeight)-2)
locSumAbs+=AbsLevel[xC][yC+2]
}
```

```
locSumAbs=Clip3(0, 31, locSumAbs-baseLevel*5)
```

С учетом переменной *locSumAbs*, параметр *cRiceParam* Райса извлекается так, как указано в таблице 4.

Когда *baseLevel* равен 0, переменная *ZeroPos[n]* извлекается следующим образом:

```
ZeroPos[n]=locSumAbs
```

[00231] Согласно двадцать второму аспекту раскрытия, предлагается извлекать *ZeroPos[n]* на основе значения *AbsLevel* соседних позиций. В одном примере, *ZeroPos[n]* извлекается на основе максимального значения из *AbsLevel[xC+1][yC]* и *AbsLevel[xC]*
[yC+1]. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 20. Процесс извлечения параметра Райса

Вводы в этот процесс представляют собой базовый уровень *baseLevel*, индекс *cIdx* цветового компонента, местоположение (*x0*, *y0*) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (*xC*, *yC*) сканирования коэффициентов, двоичный логарифм $\log_2 TbWidth$ ширины блока преобразования и двоичный логарифм $\log_2 TbHeight$ высоты блока преобразования.

Вывод этого процесса представляет собой параметр *cRiceParam* Райса.

С учетом массива *AbsLevel[x][y]* для блока преобразования с индексом *cIdx* компонента и левым верхним местоположением (*x0*, *y0*) сигнала яркости, переменная *locSumAbs* извлекается так, как указано посредством следующего псевдокода:

```
locSumAbs=0
ZeroPos[n]=1
if(xC<(1<<log2TbWidth)-1) {
locSumAbs+=AbsLevel[xC+1][yC]
ZeroPos[n]=Max(AbsLevel[xC+1][yC], ZeroPos[n])
if(xC<(1<<log2TbWidth)-2)
locSumAbs+=AbsLevel[xC+2][yC]
if(yC<(1<<log2TbHeight)-1)
locSumAbs+=AbsLevel[xC+1][yC+1]
(1494)
}
if(yC<(1<<log2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]
ZeroPos[n]=Max(AbsLevel[xC][yC+1], ZeroPos[n])
if(yC<(1<<log2TbHeight)-2)
locSumAbs+=AbsLevel[xC][yC+2]
}
```

```
locSumAbs=Clip3(0, 31, locSumAbs-baseLevel*5)
```

С учетом переменной *locSumAbs*, параметр *cRiceParam* Райса извлекается так, как указано в таблице 4.

Когда *baseLevel* равен 0, переменная *ZeroPos[n]* извлекается следующим образом:

```
ZeroPos[n]=ZeroPos[n]*1,25+1
```

[00232] Согласно двадцать третьему аспекту раскрытия, предлагается извлекать как *cRiceParam*, так и *ZeroPos[n]* на основе максимального значения всех значений *AbsLevel* соседних позиций. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 21. Процесс извлечения параметра Райса

Вводы в этот процесс представляют собой базовый уровень *baseLevel*, индекс *cIdx* цветового компонента, местоположение (*x0*, *y0*) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней вы-

борки текущей картинке, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм $\log_2 \text{TbWidth}$ ширины блока преобразования и двоичный логарифм $\log_2 \text{TbHeight}$ высоты блока преобразования.

Вывод этого процесса представляет собой параметр cRiceParam Райса.

С учетом массива AbsLevel[x][y] для блока преобразования с индексом cIdx компонента и левым верхним местоположением (x0, y0) сигнала яркости, переменная locSumAbs извлекается так, как указано посредством следующего псевдокода:

```

locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
locSumAbs=Max(AbsLevel[xC+1][yC], locSumAbs)
if(xC<(1<<log2TbWidth)-2)
locSumAbs=Max(AbsLevel[xC+2][yC], locSumAbs)
if(yC<(1<<log2TbHeight)-1)
locSumAbs=Max(AbsLevel[xC+1][yC+1], locSumAbs) (1494)
}
if(yC<(1<<log2TbHeight)-1) {
locSumAbs=Max(AbsLevel[xC][yC+1], locSumAbs)
if(yC<(1<<log2TbHeight)-2)
locSumAbs=Max(AbsLevel[xC][yC+2], locSumAbs)
}
locSumAbs=Max(0, locSumAbs-baseLevel)

```

С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается следующим образом:
 $\text{cRiceParam} = \text{Min}((\text{locSumAbs} \gg 2), 3)$

Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:
 $\text{ZeroPos}[n] = \text{locSumAbs}$

[00233] Такие же способы, поясненные в вышеприведенных аспектах, также являются применимыми к извлечению predCoeff при остаточном кодировании для режима пропуска преобразования. В одном примере, переменная predCoeff извлекается следующим образом:

[00234] $\text{predCoeff} = \text{Max}(\text{absLeftCoeff}, \text{absAboveCoeff}) + 1$

[00235] *Остаточное кодирование для коэффициентов преобразования*

[00236] В этом раскрытии, чтобы разрешать проблемы, указанные в разделе "Улучшения остаточного кодирования и кодирования коэффициентов", предоставляются способы для того, чтобы упрощать и/или дополнительно улучшать существующее проектное решение по остаточному кодированию. В общем, основные признаки предложенных технологий в этом раскрытии обобщаются следующим образом.

[00237] Во-первых, регулирование извлечения параметра Райса, используемого при регулярном остаточном кодировании на основе текущего проектного решения.

[00238] Во-вторых, изменение двоичных способов, используемых при регулярном остаточном кодировании.

[00239] В-третьих, изменение извлечения параметра Райса, используемого при регулярном остаточном кодировании.

[00240] *Извлечение параметра Райса при остаточном кодировании на основе текущего проектного решения*

[00241] Согласно двадцать четвертому аспекту раскрытия, предлагается использовать переменные способы извлечений параметра Райса для кодирования некоторых синтаксических элементов, например, abs_remainder/dec_abs_level, при остаточном кодировании, и выбор определяется согласно некоторой кодированной информации текущего блока, например, параметру квантования или битовой глубине кодирования, ассоциированной с ТВ/СВ и/или со срезом/профилем, и/или согласно новому флагу, ассоциированному с уровнем ТВ/СВ/среза/картинки/последовательности, например, extended_precision_processing_flag. Различные способы могут использоваться для того, чтобы извлекать параметр Райса, при этом некоторые примерные способы перечисляются следующим образом.

[00242] Во-первых, $\text{cRiceParam} = (\text{cRiceParam} \ll a) + (\text{cRiceParam} \gg b) + c$, где a, b и c являются положительным числом, например, $\{a, b, c\} = \{1, 1, 0\}$. Необходимо отметить, что на практике другие отличающиеся логики могут использоваться для того, чтобы достигать одинаковых результатов, например, операция умножения на значение, равное 2 в степени n.

[00243] Во-вторых, $cRiceParam = (cRiceParam \ll a) + b$, где a и b являются положительным числом, например, $\{a, b\} = \{1, 1\}$. Необходимо отметить, что на практике другие отличающиеся логики могут использоваться для того, чтобы достигать одинаковых результатов, например, операция умножения на значение, равное 2 в степени n .

[00244] В-третьих, $cRiceParam = (cRiceParam * a) + b$, где a и b являются положительным числом, например, $\{a, b\} = \{1, 5, 0\}$. Необходимо отметить, что на практике другие отличающиеся логики могут использоваться для того, чтобы достигать одинаковых результатов, например, операция умножения на значение, равное 2 в степени n .

[00245] Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Изменения VVC-проекта показаны в таблице 22 полужирным и курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса, из значения $BitDepth$ текущей CU/последовательности.

Таблица 22. Процесс извлечения параметра Райса

20	Входы в этот процесс представляют собой базовый уровень $baseLevel$, индекс $cIdx$ цветового компонента, местоположение ($x0$, $y0$) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм $\log_2 TbWidth$ ширины блока преобразования и двоичный логарифм $\log_2 TbHeight$ высоты блока преобразования.
	Вывод этого процесса представляет собой параметр $cRiceParam$ Райса.
	С учетом массива $AbsLevel[x][y]$ для блока преобразования с индексом $cIdx$ компонента и левым верхним местоположением ($x0$, $y0$) сигнала яркости, переменная $locSumAbs$ извлекается так, как указано посредством следующего псевдокода:
25	<pre> locSumAbs=0 if(xC<(1<<log2TbWidth)-1) { locSumAbs+=AbsLevel[xC+1][yC] if(xC<(1<<log2TbWidth)-2) </pre>
30	<pre> locSumAbs+=AbsLevel[xC+2][yC] if(yC<(1<<log2TbHeight)-1) locSumAbs+=AbsLevel[xC+1][yC+1] (1494) } if(yC<(1<<log2TbHeight)-1) { locSumAbs+=AbsLevel[xC][yC+1] if(yC<(1<<log2TbHeight)-2) locSumAbs+=AbsLevel[xC][yC+2] } </pre>
	<pre> locSumAbs=Clip3(0, 31, locSumAbs-baseLevel*5) </pre>
	С учетом переменной $locSumAbs$, параметр $cRiceParam$ Райса извлекается так, как указано в таблице 4.
	Если <i>extended_precision_processing_flag</i> равен 1, параметр <i>cRiceParam</i> Райса указывается так, как пояснено ниже.
35	<pre> if(BitDepth<11) { cRiceParam=cRiceParam } else if(BitDepth<13) { cRiceParam=cRiceParam+(cRiceParam>>1) } else if(BitDepth<15) { cRiceParam=cRiceParam<<1 } else { cRiceParam=cRiceParam<<1+(cRiceParam>>1) } </pre>
40	Когда $baseLevel$ равен 0, переменная $ZeroPos[n]$ извлекается следующим образом:
	<pre> ZeroPos[n]=(QState<2?1:2)<<cRiceParam </pre>

[00246] В другом примере, когда $BitDepth$ больше или равен предварительно заданному пороговому значению (например, 10, 11, 12, 13, 14, 15 или 16), параметр $cRiceParam$ Райса извлекается следующим образом: $cRiceParam = (cRiceParam \ll a) + (cRiceParam \gg b) + c$, где a , b и c являются положительным числом, например, 1. Соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный

контент показан курсивным шрифтом. Изменения VVC-проекта показаны в таблице 23 полужирным и курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса, из значения BitDepth текущей CU/последовательности.

Таблица 23. Процесс извлечения параметра Райса

	Вводы в этот процесс представляют собой базовый уровень baseLevel, индекс cIdx цветового компонента, местоположение (x0, y0) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм log2TbWidth ширины блока преобразования и двоичный логарифм log2TbHeight высоты блока преобразования.
10	Вывод этого процесса представляет собой параметр cRiceParam Райса.
	С учетом массива AbsLevel[x][y] для блока преобразования с индексом cIdx компонента и левым верхним местоположением (x0, y0) сигнала яркости, переменная locSumAbs извлекается так, как указано посредством следующего псевдокода:
15	<pre> locSumAbs=0 if(xC<(1<<log2TbWidth)-1) { locSumAbs+=AbsLevel[xC+1][yC] if(xC<(1<<log2TbWidth)-2) locSumAbs+=AbsLevel[xC+2][yC] if(yC<(1<<log2TbHeight)-1) locSumAbs+=AbsLevel[xC+1][yC+1] (1494) } </pre>
	<pre> if(yC<(1<<log2TbHeight)-1) { locSumAbs+=AbsLevel[xC][yC+1] if(yC<(1<<log2TbHeight)-2) locSumAbs+=AbsLevel[xC][yC+2] } </pre>
20	<pre> locSumAbs=Clip3(0, 31, locSumAbs-baseLevel*5) </pre> С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается так, как указано в таблице 4.
	<i>Параметр cRiceParam Райса указывается так, как пояснено ниже.</i>
	<pre> if(BitDepth<11) { cRiceParam=cRiceParam } </pre>
25	<pre> else if(BitDepth<13) { cRiceParam=cRiceParam+(cRiceParam>>1) } </pre>
	<pre> else if(BitDepth<15) { cRiceParam=cRiceParam<<1 } </pre>
	<pre> else { cRiceParam=cRiceParam<<1+(cRiceParam>>1) } </pre>
30	Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:
	ZeroPos[n]=(QState<2?1:2)<<cRiceParam

[00247] *Двоичные способы при остаточном кодировании для профилей выше 10-битовых*

[00248] Согласно двадцать пятому аспекту раскрытия, предлагается использовать наборы переменных двоичных кодовых слов для кодирования некоторых синтаксических элементов, например, abs_remainder/dec_abs_level, при остаточном кодировании, и выбор определяется согласно некоторой кодированной информации текущего блока, например, параметру квантования или битовой глубине кодирования, ассоциированной с ТВ/СВ и/или со срезом/профилем, и/или согласно новому флагу, ассоциированному с уровнем ТВ/СВ/среза/картинки/последовательности, например, extended_precision_processing_flag. Различные способы могут использоваться для того, чтобы извлекать наборы переменных двоичных кодовых слов, при этом некоторые примерные способы перечисляются следующим образом.

[00249] Во-первых, используется такая же процедура для определения кодового слова для abs_remainder, как используемая в текущем VVC, но всегда с выбранным фиксированным параметром Райса (например, 2, 3, 4, 5, 6, 7 или 8). Фиксированное значение может отличаться при различном условии согласно некоторой кодированной информации текущего блока, например, параметру квантования или битовой глубине

кодирования, ассоциированной с ТВ/СВ и/или со срезом/профилем, и/или согласно синтаксическому элементу, ассоциированному с уровнем ТВ/СВ/среза/картинки/последовательности, например, `rice_parameter_value`. Один конкретный пример проиллюстрирован так, как показано в таблице 24, где TH1-TH4 являются

предварительно заданными пороговыми значениями, удовлетворяющими (TH1<TH2<TH3<TH4), и K0-K4 представляют собой предварительно заданные параметры Райса. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса, как показано в таблице 24, из значения BitDepth текущей CU/последовательности.

[00250] Во-вторых, преобразование в двоичную форму кодом фиксированной длины.

[00251] В-третьих, преобразование в двоичную форму усеченным кодом Райса.

[00252] В-четвертых, процесс усеченного двоичного (ТВ) преобразования в двоичную форму.

[00253] В-пятых, процесс экспоненциального преобразования в двоичную форму Голомба k-ого порядка (EGk).

[00254] В-шестых, ограниченное экспоненциальное преобразование в двоичную форму Голомба k-ого порядка

Таблица 24. Определение параметра Райса на основе битовой глубины

```

if(BitDepth<TH1)
{
rice parameter=K0
}
else if(BitDepth<TH2)
{
rice parameter=K1
}
else if(BitDepth<TH3)
{
rice parameter=K2
}
else if(BitDepth<TH4)
{
rice parameter=K3
}
else
{
rice parameter=K4
}

```

[00255] В одном примере, когда новый флаг, например, `extended_precision_processing_flag`, равен 1, параметр `cRiceParam` Райса является фиксированным в качестве n, где n является положительным числом (например, 2, 3, 4, 5, 6, 7 или 8). Фиксированное значение может отличаться при различном условии. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Изменения VVC-проекта показаны в таблице 25 полужирным и курсивным шрифтом.

Таблица 25. Процесс извлечения параметра Райса

Параметр `cRiceParam` Райса извлекается следующим образом:

- Если `transform_skip_flag[x0][y0][cIdx]` равен 1, и `sh_ts_residual_coding_disabled_flag` равен 0, параметр `cRiceParam` Райса задается равным 1.
- Если *`extended_precision_processing_flag` равен 1, параметр `cRiceParam` Райса задается равным 6.*
- Иначе, параметр `cRiceParam` Райса извлекается посредством активации процесса извлечения параметра Райса для `abs_remainder[]`, как указано в таблице 3, с переменной `baseLevel`, заданной равной 4, индексом `cIdx` цветового компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом `log2TbWidth` ширины блока преобразования и двоичным логарифмом `log2TbHeight` высоты блока преобразования в качестве вводов.

[00256] В другом примере, предлагается использовать только одно фиксированное

значение для параметра Райса при кодировании синтаксического элемента `abs_remainder/dec_abs_level`, когда новый флаг, например, `extended_precision_processing_flag`, равен 1. Соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Изменения VVC-проекта показаны в таблице 26 полужирным и курсивным шрифтом.

Таблица 26. Процесс извлечения параметра Райса	
Параметр <code>cRiceParam</code> Райса извлекается следующим образом: <i>- Если <code>extended_precision_processing_flag</code> равен 1, параметр <code>cRiceParam</code> Райса задается равным 7.</i>	
10	- Если <code>transform_skip_flag[x0][y0][cIdx]</code> равен 1, и <code>sh_ts_residual_coding_disabled_flag</code> равен 0, параметр <code>cRiceParam</code> Райса задается равным 1.
	- Иначе, параметр <code>cRiceParam</code> Райса извлекается посредством активации процесса извлечения параметра Райса для <code>abs_remainder[]</code> , как указано в таблице 3, с переменной <code>baseLevel</code> , заданной равной 4, индексом <code>cIdx</code> цветового компонента, местоположением $(x0, y0)$ сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом <code>log2TbWidth</code> ширины блока преобразования и двоичным логарифмом <code>log2TbHeight</code> высоты блока преобразования в качестве вводов.

[00257] В еще одном другом примере, когда `BitDepth` больше или равен предварительно заданному пороговому значению (например, 10, 11, 12, 13, 14, 15 или 16), параметр `cRiceParam` Райса является фиксированным в качестве `n`, где `n` является положительным числом, например, 4, 5, 6, 7 или 8. Фиксированное значение может отличаться при различном условии. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, где `TH` является предварительно заданным пороговым значением (например, 10, 11, 12, 13, 14, 15 или 16). Изменения VVC-проекта показаны в таблице 27 полужирным и курсивным шрифтом, и при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом.

Таблица 27. Процесс извлечения параметра Райса	
Параметр <code>cRiceParam</code> Райса извлекается следующим образом:	
25	- Если <code>transform_skip_flag[x0][y0][cIdx]</code> равен 1, и <code>sh_ts_residual_coding_disabled_flag</code> равен 0, параметр <code>cRiceParam</code> Райса задается равным 1.
	<i>- Если <code>BitDepth</code> больше <code>TH</code>, параметр <code>cRiceParam</code> Райса задается равным 6.</i>
30	- Иначе, параметр <code>cRiceParam</code> Райса извлекается посредством активации процесса извлечения параметра Райса для <code>abs_remainder[]</code> , как указано в таблице 3, с переменной <code>baseLevel</code> , заданной равной 4, индексом <code>cIdx</code> цветового компонента, местоположением $(x0, y0)$ сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом <code>log2TbWidth</code> ширины блока преобразования и двоичным логарифмом <code>log2TbHeight</code> высоты блока преобразования в качестве вводов.

[00258] В еще одном другом примере, предлагается использовать только одно фиксированное значение для параметра Райса при кодировании синтаксического элемента `abs_remainder/dec_abs_level`, когда `BitDepth` больше предварительно заданного порогового значения (например, 10, 11, 12, 13, 14, 15 или 16). Соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, где `TH` является предварительно заданным пороговым значением (например, 10, 11, 12, 13, 14, 15 или 16), и при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Изменения VVC-проекта показаны в таблице 28 полужирным и курсивным шрифтом.

Таблица 28. Процесс извлечения параметра Райса	
Параметр <code>cRiceParam</code> Райса извлекается следующим образом:	
<i>- Если <code>BitDepth</code> больше <code>TH</code>, параметр <code>cRiceParam</code> Райса задается равным 7.</i>	
40	- Если <code>transform_skip_flag[x0][y0][cIdx]</code> равен 1, и <code>sh_ts_residual_coding_disabled_flag</code> равен 0, параметр <code>cRiceParam</code> Райса задается равным 1.
45	- Иначе, параметр <code>cRiceParam</code> Райса извлекается посредством активации процесса извлечения параметра Райса для <code>abs_remainder[]</code> , как указано в таблице 3, с переменной <code>baseLevel</code> , заданной равной 4, индексом <code>cIdx</code> цветового компонента, местоположением $(x0, y0)$ сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом <code>log2TbWidth</code> ширины блока преобразования и двоичным логарифмом <code>log2TbHeight</code> высоты блока преобразования в качестве вводов.

[00259] *Извлечение параметра Райса при остаточном кодировании*

[00260] Согласно двадцать шестому аспекту раскрытия, предлагается использовать

переменные способы извлечений параметра Райса для кодирования некоторых синтаксических элементов, например, `abs_remainder/dec_abs_level`, при остаточном кодировании, и выбор определяется согласно некоторой кодированной информации текущего блока, например, параметру квантования или битовой глубине кодирования, ассоциированной с ТВ/СВ и/или со срезом/профилем, и/или согласно новому флагу, ассоциированному с уровнем ТВ/СВ/среза/картинки/последовательности, например, `extended_precision_processing_flag`. Различные способы могут использоваться для того, чтобы извлекать параметр Райса, при этом некоторые примерные способы перечисляются следующим образом.

[00261] Во-первых, предлагается использовать счетчики для того, чтобы извлекать параметр Райса. Счетчики определяются согласно значению кодированного коэффициента и некоторой кодированной информации текущего блока, например, `compent ID`. Один конкретный пример, `riceParameter=counter/a`, где `a` является положительным числом, например, 4, и оно поддерживает 2 счетчика (разбиение посредством сигнала яркости/сигнала цветности). Эти счетчики сбрасываются до 0 в начале каждого среза. После кодирования, счетчик обновляется, если он представляет собой первый коэффициент, кодированный в суб-TU следующим образом:

```
if (coeffValue>=(3<<rice)) counter++
if (((coeffValue<<1)<(1<<riceParameter)) andand (counter>0)) counter--;
```

[00262] Во-вторых, предлагается добавлять операцию сдвига при извлечении параметра Райса в VVC. Сдвиг определяется согласно значению кодированного коэффициента. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, сдвиг определяется согласно счетчикам способа 1, и при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Изменения VVC-проекта показаны в таблице 29 полужирным и курсивным шрифтом.

Таблица 29. Процесс извлечения параметра Райса

30	Вводы в этот процесс представляют собой базовый уровень <code>baseLevel</code>, индекс <code>clDx</code> цветового компонента, местоположение (<code>x0</code>, <code>y0</code>) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (<code>xC</code>, <code>yC</code>) сканирования коэффициентов, двоичный логарифм <code>log2TbWidth</code> ширины блока преобразования и двоичный логарифм <code>log2TbHeight</code> высоты блока преобразования. Вывод этого процесса представляет собой параметр <code>cRiceParam</code> Райса. С учетом массива <code>AbsLevel[x][y]</code> для блока преобразования с индексом <code>clDx</code> компонента и левым верхним местоположением (<code>x0</code>, <code>y0</code>) сигнала яркости, переменная <code>locSumAbs</code> извлекается так, как указано посредством следующего псевдокода:
35	<pre>Shift=max(0, counter/4-2) locSumAbs=0 if(xC<(1<<log2TbWidth)-1) { locSumAbs+=AbsLevel[xC+1][yC] if(xC<(1<<log2TbWidth)-2) locSumAbs+=AbsLevel[xC+2][yC] if(yC<(1<<log2TbHeight)-1) locSumAbs+=AbsLevel[xC+1][yC+1] (1494) } if(yC<(1<<log2TbHeight)-1) { locSumAbs+=AbsLevel[xC][yC+1] if(yC<(1<<log2TbHeight)-2) locSumAbs+=AbsLevel[xC][yC+2] }</pre>
40	<pre>locSumAbs=Clip3(0, 31, (locSumAbs-baseLevel*5)>>Shift) С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается так, как указано в таблице 4. cRiceParam=cRiceParam+Shift Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом: ZeroPos[n]=(QState<2?1:2)<<cRiceParam</pre>

[00263] Во-первых, предлагается добавлять операцию сдвига при извлечении параметра Райса в VVC. Сдвиг определяется согласно некоторой кодированной информации текущего блока, например, битовой глубине кодирования, ассоциированной с ТВ/СВ и/или с профилем среза (например, 14-битовый профиль или 16-битовый

профиль). Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, сдвиг определяется согласно счетчикам способа 1, и при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Изменения VVC-проекта показаны в таблице 30 полужирным и курсивным шрифтом.

Таблица 30. Процесс извлечения параметра Райса	
Вводы в этот процесс представляют собой базовый уровень <i>baseLevel</i> , индекс <i>cIdx</i> цветового компонента, местоположение (<i>x0</i> , <i>y0</i>) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (<i>xC</i> , <i>yC</i>) сканирования коэффициентов, двоичный логарифм <i>log2TbWidth</i> ширины блока преобразования и двоичный логарифм <i>log2TbHeight</i> высоты блока преобразования.	
Вывод этого процесса представляет собой параметр <i>cRiceParam</i> Райса.	
С учетом массива <i>AbsLevel[x][y]</i> для блока преобразования с индексом <i>cIdx</i> компонента и левым верхним местоположением (<i>x0</i> , <i>y0</i>) сигнала яркости, переменная <i>locSumAbs</i> извлекается так, как указано посредством следующего псевдокода:	
	<pre> Shift=max(0, (BitDepth-8)/2) locSumAbs=0 if(xC<(1<<log2TbWidth)-1) { locSumAbs+=AbsLevel[xC+1][yC] if(xC<(1<<log2TbWidth)-2) locSumAbs+=AbsLevel[xC+2][yC] if(yC<(1<<log2TbHeight)-1) locSumAbs+=AbsLevel[xC+1][yC+1] (1494) } if(yC<(1<<log2TbHeight)-1) { locSumAbs+=AbsLevel[xC][yC+1] if(yC<(1<<log2TbHeight)-2) locSumAbs+=AbsLevel[xC][yC+2] } locSumAbs=Clip3(0, 31, (locSumAbs-baseLevel*5)>>Shift) </pre>
С учетом переменной <i>locSumAbs</i> , параметр <i>cRiceParam</i> Райса извлекается так, как указано в таблице 4.	
	<i>cRiceParam=cRiceParam+Shift</i>
Когда <i>baseLevel</i> равен 0, переменная <i>ZeroPos[n]</i> извлекается следующим образом:	
	<i>ZeroPos[n]=(QState<2?1:2)<<cRiceParam</i>

[00264] *Остаточное кодирование для пропуска преобразования*

[00265] Согласно двадцать седьмому аспекту раскрытия, предлагается использовать наборы переменных двоичных кодовых слов для кодирования некоторых синтаксических элементов, например, *abs_remainder*, при остаточном кодировании с пропуском преобразования, и выбор определяется согласно некоторой кодированной информации текущего блока, например, параметру квантования или битовой глубине кодирования, ассоциированной с ТВ/СВ и/или со срезом/профилем, и/или согласно новому флагу, ассоциированному с уровнем ТВ/СВ/среза/картинки/последовательности, например, *extended_precision_processing_flag*. Различные способы могут использоваться для того, чтобы извлекать наборы переменных двоичных кодовых слов, при этом некоторые примерные способы перечисляются следующим образом.

[00266] Во-первых, используется такая же процедура для определения кодового слова для *abs_remainder*, как используемая в текущем VVC, но всегда с выбранным фиксированным параметром Райса (например, 2, 3, 4, 5, 6, 7 или 8). Фиксированное значение может отличаться при различном условии согласно некоторой кодированной информации текущего блока, например, параметру квантования, типу кадра (например, I, P или B), идентификатору компонента (например, сигнал яркости или сигнал цветности), цветовому формату (например, 420, 422 или 444) или битовой глубине кодирования, ассоциированной с ТВ/СВ и/или со срезом/профилем, и/или согласно синтаксическому элементу, ассоциированному с уровнем ТВ/СВ/среза/картинки/последовательности, например, *rice_parameter_value*. Один конкретный пример проиллюстрирован так, как показано в таблице 7, где ТН1-ТН4 являются предварительно заданными пороговыми значениями, удовлетворяющими ($ТН1 < ТН2 < ТН3 < ТН4$), и К0-К4 представляют собой предварительно заданные параметры Райса. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике.

Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса, как показано в таблице 7, из значения BitDepth текущей CU/последовательности.

[00267] Во-вторых, преобразование в двоичную форму кодом фиксированной длины.

[00268] В-третьих, преобразование в двоичную форму усеченным кодом Райса.

[00269] В-четвертых, процесс усеченного двоичного (ТВ) преобразования в двоичную форму.

[00270] В-пятых, процесс экспоненциального преобразования в двоичную форму Голомба k-ого порядка (EGk).

[00271] В-шестых, ограниченное экспоненциальное преобразование в двоичную форму Голомба k-ого порядка

[00272] Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, изменения VVC-проекта показаны в таблице 31 полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике.

Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

Таблица 31. Процесс извлечения параметра Райса	
Параметр cRiceParam Райса извлекается следующим образом:	
- Если <i>transform_skip_flag[x0][y0][cldx]</i> равен 1, и <i>sh_ts_residual_coding_disabled_flag</i> равен 0, процесс извлечения для параметра cRiceParam Райса указывается так, как пояснено ниже.	
	<i>if(BitDepth<11)</i>
	<i>{</i>
	<i>rice parameter=1</i>
	<i>}</i>
	<i>else if(BitDepth<13)</i>
	<i>{</i>
	<i>rice parameter=4</i>
	<i>}</i>
	<i>else if(BitDepth<15)</i>
	<i>{</i>
	<i>rice parameter=6</i>
	<i>}</i>
	<i>else</i>
	<i>{</i>
	<i>rice parameter=8</i>
	<i>}</i>
- Иначе, параметр cRiceParam Райса извлекается посредством активации процесса извлечения параметра Райса для <i>abs_remainder</i> [], как указано в таблице 3, с переменной <i>baseLevel</i> , заданной равной 4, индексом <i>cldx</i> цветового компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xС, yС) сканирования коэффициентов, двоичным логарифмом <i>log2TbWidth</i> ширины блока преобразования и двоичным логарифмом <i>log2TbHeight</i> высоты блока преобразования в качестве вводов.	

[00273] В другом примере, предлагается использовать только одно фиксированное значение для параметра Райса при кодировании синтаксического элемента *abs_remainder*, когда новый флаг, например, *extended_precision_processing_flag*, равен 1.

Соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Изменения VVC-проекта показаны в таблице 32 полужирным и курсивным шрифтом.

Таблица 32. Процесс извлечения параметра Райса	
Параметр cRiceParam Райса извлекается следующим образом:	
- Если <i>extended_precision_processing_flag</i> равен 1, <i>transform_skip_flag[x0][y0][cldx]</i> равен 1, и <i>sh_ts_residual_coding_disabled_flag</i> равен 0, процесс извлечения для параметра cRiceParam Райса указывается так, как пояснено ниже.	
	<i>if(BitDepth<11)</i>
	<i>{</i>
	<i>rice parameter=1</i>
	<i>}</i>
	<i>else if(BitDepth<13)</i>
	<i>{</i>
	<i>rice parameter=4</i>
	<i>}</i>
	<i>else if(BitDepth<15)</i>

```

    {
        rice parameter=6
    }
    else
    {
        rice parameter=8
    }

```

5 - Если `transform_skip_flag[x0][y0][cIdx]` равен 1, и `sh_ts_residual_coding_disabled_flag` равен 0, параметр `cRiceParam` Райса задается равным 1.
 - Иначе, параметр `cRiceParam` Райса извлекается посредством активации процесса извлечения параметра Райса для `abs_remainder` [], как указано в таблице 3, с переменной `baseLevel`, заданной равной 4, индексом `cIdx` цветового компонента, местоположением $(x0, y0)$ сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом `log2TbWidth` ширины блока преобразования и двоичным логарифмом `log2TbHeight` высоты блока преобразования в качестве вводов.

10 [00274] В еще одном другом примере, когда новый флаг, например, `extended_precision_processing_flag`, равен 1, параметр `cRiceParam` Райса является фиксированным в качестве `n`, где `n` является положительным числом (например, 2, 3, 4, 5, 6, 7 или 8).
 Фиксированное значение может отличаться при различном условии. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, при этом изменения показаны полужирным и курсивным шрифтом, а удаленный
 15 контент показан курсивным шрифтом. Изменения VVC-проекта показаны в таблице 33 полужирным и курсивным шрифтом.

Таблица 33. Процесс извлечения параметра Райса

20 Параметр `cRiceParam` Райса извлекается следующим образом:
 - Если ***`extended_precision_processing_flag`*** равен 1, ***`transform_skip_flag[x0][y0][cIdx]`*** равен 1, и ***`sh_ts_residual_coding_disabled_flag`*** равен 0, параметр `cRiceParam` Райса задается равным 7.
 - Если `transform_skip_flag[x0][y0][cIdx]` равен 1, и `sh_ts_residual_coding_disabled_flag` равен 0, параметр `cRiceParam` Райса задается равным 1.
 - Иначе, параметр `cRiceParam` Райса извлекается посредством активации процесса извлечения параметра Райса для `abs_remainder` [], как указано в таблице 3, с переменной `baseLevel`, заданной равной 4, индексом `cIdx` цветового компонента, местоположением $(x0, y0)$ сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом `log2TbWidth` ширины блока преобразования и двоичным логарифмом `log2TbHeight` высоты блока преобразования в качестве вводов.

25 [00275] В еще одном другом примере, когда `BitDepth` больше или равен предварительно заданному пороговому значению (например, 10, 11, 12, 13, 14, 15 или 16), параметр `cRiceParam` Райса является фиксированным в качестве `n`, где `n` является положительным числом, например, 4, 5, 6, 7 или 8. Фиксированное значение может отличаться при различном условии. Пример соответствующего процесса декодирования
 30 на основе VVC-проекта проиллюстрирован ниже, где `TH` является предварительно заданным пороговым значением (например, 10, 11, 12, 13, 14, 15 или 16), и при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Изменения VVC-проекта показаны в таблице 34 полужирным и курсивным шрифтом.

Таблица 34. Процесс извлечения параметра Райса

35 Параметр `cRiceParam` Райса извлекается следующим образом:
 - Если ***`BitDepth`*** больше ***`TH`***, ***`transform_skip_flag[x0][y0][cIdx]`*** равен 1, и ***`sh_ts_residual_coding_disabled_flag`*** равен 0, параметр ***`cRiceParam`*** Райса задается равным 7.
 - Если `transform_skip_flag[x0][y0][cIdx]` равен 1, и `sh_ts_residual_coding_disabled_flag` равен 0, параметр `cRiceParam` Райса задается равным 1.
 40 - Иначе, параметр `cRiceParam` Райса извлекается посредством активации процесса извлечения параметра Райса для `abs_remainder` [], как указано в таблице 3, с переменной `baseLevel`, заданной равной 4, индексом `cIdx` цветового компонента, местоположением $(x0, y0)$ сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом `log2TbWidth` ширины блока преобразования и двоичным логарифмом `log2TbHeight` высоты блока преобразования в качестве вводов.

[00276] В еще одном другом примере, один флаг управления сигнализируется в заголовке среза, чтобы указывать то, активируется или деактивируется сигнализация параметра Райса для блоков с пропуском преобразования. Когда флаг управления
 45 сигнализируется в качестве активированного, один синтаксический элемент дополнительно сигнализируется для каждого среза пропуска преобразования, чтобы указывать параметр Райса этого среза. Когда флаг управления сигнализируется в качестве деактивированного (например, заданного равным "0"), дополнительный

синтаксический элемент не сигнализируется на нижнем уровне для того, чтобы указывать параметр Райса для среза пропуска преобразования, и параметр Райса по умолчанию (например, 1) используется для всего среза пропуска преобразования. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, где ТН является предварительно заданным значением (например, 0, 1, 2), и при этом изменения показаны полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Изменения VVC-проекта показаны в таблице 35 полужирным и курсивным шрифтом. Необходимо отметить, что `sh_ts_residual_coding_rice_index` может кодироваться различными способами и/или может иметь максимальное значение. Например, `u(n)`, целое число без знака с использованием `n` битов, или `f(n)`, битовая строка с фиксированным шаблоном с использованием `n` битов, записываемая (слева направо) с левым битом сначала, также может использоваться для того, чтобы кодировать/декодировать одинаковый синтаксический элемент.

[00277] *Синтаксис заголовков срезов*

Таблица 35. Синтаксис остаточного кодирования	
<code>slice_header() {</code>	Дескриптор
<code>...</code>	
<code>if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)</code>	
<code>sh_ts_residual_coding_disabled_flag</code>	<code>u(1)</code>
<code>if(!sh_ts_residual_coding_disabled_flag) {</code>	
<code>sh_ts_residual_coding_rice_flag</code>	<code>u(1)</code>
<code>if(sh_ts_residual_coding_rice_flag)</code>	
<code>sh_ts_residual_coding_rice_index</code>	<code>ue(v)</code>
<code>}</code>	
<code>...</code>	

[00278] - `sh_ts_residual_coding_rice_flag`, равный 1, указывает то, что `sh_ts_residual_coding_rice_index` может присутствовать в текущем срезе; `sh_ts_residual_coding_rice_flag`, равный 0, указывает то, что `sh_ts_residual_coding_rice_index` не присутствует в текущем срезе. Когда `sh_ts_residual_coding_rice_flag` не присутствует, значение `sh_ts_residual_coding_rice_index` логически выводится равным 0.

- `sh_ts_residual_coding_rice_index` указывает параметр Райса, используемый для синтаксической структуры `residual_ts_coding()`.

Таблица 36. Процесс извлечения параметра Райса	
Параметр <code>cRiceParam</code> Райса извлекается следующим образом:	
- Если <code>sh_ts_residual_coding_rice_flag</code> равен 1, <code>transform_skip_flag[x0][y0][cIdx]</code> равен 1, и <code>sh_ts_residual_coding_disabled_flag</code> равен 0, параметр <code>cRiceParam</code> Райса задается равным <code>(sh_ts_residual_coding_rice_index+TH)</code> .	
- Если <code>transform_skip_flag[x0][y0][cIdx]</code> равен 1, и <code>sh_ts_residual_coding_disabled_flag</code> равен 0, параметр <code>cRiceParam</code> Райса задается равным 1.	
- Иначе, параметр <code>cRiceParam</code> Райса извлекается посредством активации процесса извлечения параметра Райса для <code>abs_remainder[]</code> , как указано в таблице 3, с переменной <code>baseLevel</code> , заданной равной 4, индексом <code>cIdx</code> цветового компонента, местоположением <code>(x0, y0)</code> сигнала яркости, текущим местоположением <code>(xC, yC)</code> сканирования коэффициентов, двоичным логарифмом <code>log2TbWidth</code> ширины блока преобразования и двоичным логарифмом <code>log2TbHeight</code> высоты блока преобразования в качестве вводов.	

[00279] В еще одном другом примере, один флаг управления сигнализируется в наборе параметров последовательности (или в синтаксисе расширений диапазона набора параметров последовательности), чтобы указывать то, активируется или деактивируется сигнализация параметра Райса для блоков с пропуском преобразования. Когда флаг управления сигнализируется в качестве активированного, один синтаксический элемент дополнительно сигнализируется для каждого среза пропуска преобразования, чтобы указывать параметр Райса этого среза. Когда флаг управления сигнализируется в качестве деактивированного (например, заданного равным "0"), дополнительный синтаксический элемент не сигнализируется на нижнем уровне для того, чтобы указывать

параметр Райса для среза пропуска преобразования, и параметр Райса по умолчанию (например, 1) используется для всего среза пропуска преобразования. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, где ТН является предварительно заданным значением (например, 0, 1, 2).

- 5 Изменения VVC-проекта показаны в таблице 37 полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Необходимо отметить, что *sh_ts_residual_coding_rice_idx* может кодироваться различными способами и/или может иметь максимальное значение. Например, *u(n)*, целое число без знака с использованием *n* битов, или *f(n)*, битовая строка с фиксированным шаблоном с использованием *n* битов, записываемая (слева направо) с левым битом сначала, также может использоваться
- 10 для того, чтобы кодировать/декодировать одинаковый синтаксический элемент.

[00280] *RBSP-синтаксис наборов параметров последовательности*

Таблица 37. Синтаксис остаточного кодирования	
seq_parameter_set_rbsp() {	Дескриптор
...	
sps_sign_data_hiding_enabled_flag	u(1)
<i>sps_ts_residual_coding_rice_present_in_sh_flag</i>	<i>u(1)</i>
sps_virtual_boundaries_enabled_flag	u(1)
...	
}	

- [00281] - *sps_ts_residual_coding_rice_present_in_sh_flag*, равный 1, указывает то, что *sh_ts_residual_coding_rice_idx* может присутствовать в синтаксических SH-структурах, ссылающихся на SPS; *sps_ts_residual_coding_rice_present_in_sh_flag*, равный 0, указывает то, что *sh_ts_residual_coding_rice_idx* не присутствует в синтаксических SH-структурах, ссылающихся на SPS. Когда *sps_ts_residual_coding_rice_present_in_sh_flag* не присутствует,
- 25 значение *sps_ts_residual_coding_rice_present_in_sh_flag* логически выводится равным 0.

[00282] *Синтаксис заголовков срезов*

Таблица 38. Синтаксис остаточного кодирования	
slice_header() {	Дескриптор
...	
if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)	
sh_ts_residual_coding_disabled_flag	u(1)
<i>if(!sh_ts_residual_coding_disabled_flag) && sps_ts_residual_coding_rice_enabled_flag</i>)	
<i>sh_ts_residual_coding_rice_idx</i>	<i>ue(v)</i>
...	
}	

[00283] - *sh_ts_residual_coding_rice_idx* указывает параметр Райса, используемый для синтаксической структуры *residual_ts_coding()*.

Таблица 39. Процесс извлечения параметра Райса	
Параметр <i>cRiceParam</i> Райса извлекается следующим образом:	
- Если <i>sps_ts_residual_coding_rice_flag</i> равен 1, <i>transform_skip_flag[x0][y0][cldx]</i> равен 1, и <i>sh_ts_residual_coding_disabled_flag</i> равен 0, параметр <i>cRiceParam</i> Райса задается равным (<i>sh_ts_residual_coding_rice_idx</i> +ТН).	
- Если <i>transform_skip_flag[x0][y0][cldx]</i> равен 1, и <i>sh_ts_residual_coding_disabled_flag</i> равен 0, параметр <i>cRiceParam</i> Райса задается равным 1.	
- Иначе, параметр <i>cRiceParam</i> Райса извлекается посредством активации процесса извлечения параметра Райса для <i>abs_remainder</i> [], как указано в таблице 3, с переменной <i>baseLevel</i> , заданной равной 4, индексом <i>cldx</i> цветового компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xС, yС) сканирования коэффициентов, двоичным логарифмом <i>log2TbWidth</i> ширины блока преобразования и двоичным логарифмом <i>log2TbHeight</i> высоты блока преобразования в качестве вводов.	

[00284] В одном или более примеров раскрытия, как проиллюстрировано на фиг. 27, предлагается деактивировать присутствие параметра Райса для остаточного

кодирования с пропуском преобразования, если пропуск преобразования деактивируется, на этапе 2702. В одном конкретном примере, чтобы удовлетворять такой проектной цели, предлагается использовать `sps_transform_skip_enabled_flag`, чтобы обуславливать присутствие `sps_ts_residual_coding_rice_present_in_sh_flag`, как показано на этапе 2704. Например, когда флаг `sps_transform_skip_enabled_flag` равен нулю (т.е. пропуск преобразования деактивируется в текущей картинке), `sps_ts_residual_coding_rice_present_in_sh_flag` не сигнализируется, а логически выводится равным 0. Когда флаг `sps_transform_skip_enabled_flag` равен единице, `sps_ts_residual_coding_rice_present_in_sh_flag` дополнительно сигнализируется. Изменение рабочего проекта текущего VVC показывается курсивным шрифтом ниже.

<i>if(sps_transform_skip_enabled_flag)</i>	
<i>sps_ts_residual_coding_rice_present_in_sh_flag</i>	<i>u(1)</i>

[00285] В еще одном другом примере, когда флаг пропуска преобразования (`sps_transform_skip_enabled_flag`) сигнализируется в качестве активированного, один флаг управления дополнительно сигнализируется в наборе параметров последовательности (или в синтаксисе расширений диапазона набора параметров последовательности), чтобы указывать то, активируется или деактивируется сигнализация параметра Райса для блоков с пропуском преобразования. Когда флаг управления сигнализируется в качестве активированного, один синтаксический элемент дополнительно сигнализируется для каждого среза пропуска преобразования, чтобы указывать параметр Райса этого среза. Когда флаг управления сигнализируется в качестве деактивированного (например, заданного равным "0"), дополнительный синтаксический элемент не сигнализируется на нижнем уровне для того, чтобы указывать параметр Райса для среза пропуска преобразования, и параметр Райса по умолчанию (например, 1) используется для всего среза пропуска преобразования. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже. Изменения VVC-проекта показаны курсивным шрифтом.

[00286] *RBSP-синтаксис наборов параметров последовательности*

<i>seq_parameter_set_rbsp() {</i>	Дескриптор
<i>...</i>	
<i>sps_sign_data_hiding_enabled_flag</i>	<i>u(1)</i>
<i>if(sps_transform_skip_enabled_flag)</i>	
<i>sps_ts_residual_coding_rice_present_in_sh_flag</i>	<i>u(1)</i>
<i>sps_virtual_boundaries_enabled_flag</i>	<i>u(1)</i>
<i>...</i>	
<i>}</i>	

[00287] - `sps_ts_residual_coding_rice_present_in_sh_flag`, равный 1, указывает то, что `sh_ts_residual_coding_rice_idx` может присутствовать в синтаксических SH-структурах, ссылающихся на SPS; `sps_ts_residual_coding_rice_present_in_sh_flag`, равный 0, указывает то, что `sh_ts_residual_coding_rice_idx_minus1` не присутствует в синтаксических SH-структурах, ссылающихся на SPS. Когда `sps_ts_residual_coding_rice_present_in_sh_flag` не присутствует, значение `sps_ts_residual_coding_rice_present_in_sh_flag` логически выводится равным 0.

[00288] *Синтаксис заголовков срезов*

<i>slice_header() {</i>	Дескриптор
<i>...</i>	
<i>if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag &&</i>	

!sh_sign_data_hiding_used_flag)	
sh_ts_residual_coding_disabled_flag	u(1)
if((!sh_ts_residual_coding_disabled_flag) && sps_ts_residual_coding_rice_present_in_sh_flag)	
sh_ts_residual_coding_rice_idx_minus1	u(3)
...	
}	

[00289] - sh_ts_residual_coding_rice_idx_minus1 плюс 1, указывает параметр Райса, используемый для синтаксической структуры residual_ts_coding(). Когда sh_ts_residual_coding_rice_idx_minus1 не присутствует, значение sh_ts_residual_coding_rice_idx_minus1 логически выводится равным 0.

[00290] 9.3.3.11. Процесс преобразования в двоичную форму для abs_remainder[]

[00291] Ввод в этот процесс представляет собой запрос на преобразование в двоичную форму для синтаксического элемента abs_remainder[n], цветовой компонент cIdx, индекс i текущего субблока и местоположение (x0, y0) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования сигналов яркости относительно левой верхней выборки сигналов яркости картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм log2TbWidth ширины блока преобразования и двоичный логарифм log2TbHeight высоты блока преобразования.

[00292] Вывод этого процесса представляет собой преобразование в двоичную форму синтаксического элемента.

[00293] Переменные lastAbsRemainder и lastRiceParam извлекаются следующим образом:

[00294] - Если этот процесс активируется в первый раз для индекса i текущего субблока, lastAbsRemainder и lastRiceParam задаются равными 0.

[00295] - Иначе (этот процесс не активируется в первый раз для индекса i текущего субблока), lastAbsRemainder и lastRiceParam задаются равными значениям abs_remainder[n] и cRiceParam, соответственно, которые извлечены во время последней активации процесса преобразования в двоичную форму для синтаксического элемента abs_remainder[n], как указано в этом разделе.

[00296] Параметр cRiceParam Райса извлекается следующим образом:

[00297] - Если transform_skip_flag[x0][y0][cIdx] равен 1, и sh_ts_residual_coding_disabled_flag равен 0, параметр cRiceParam Райса задается равным sh_ts_residual_coding_rice_idx_minus1+1.

[00298] - Иначе, параметр cRiceParam Райса извлекается посредством активации процесса извлечения параметра Райса для abs_remainder[], как указано в разделе 9.3.3.2, с переменной baseLevel, заданной равной 4, индексом cIdx цветовой компоненты, местоположением (x0, y0) сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом log2TbWidth ширины блока преобразования и двоичным логарифмом log2TbHeight высоты блока преобразования в качестве вводов.

[00299] В еще одном другом примере, один синтаксический элемент сигнализируется для каждого среза пропуска преобразования, чтобы указывать параметр Райса этого среза. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже. Изменения VVC-проекта показаны в таблице 40 полужирным и курсивным шрифтом. Необходимо отметить, что sh_ts_residual_coding_rice_idx может кодироваться различными способами и/или может иметь максимальное значение. Например, u(n), целое число без знака с использованием n битов, или f(n), битовая строка с фиксированным шаблоном с использованием n битов, записываемая (слева направо) с левым битом сначала, также может использоваться для того, чтобы кодировать/декодировать одинаковый синтаксический элемент.

[00300] Синтаксис заголовков срезов

Таблица 40. Синтаксис остаточного кодирования	
	Дескриптор
slice_header() {	
...	
if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)	
sh_ts_residual_coding_disabled_flag	u(1)
if(!sh_ts_residual_coding_disabled_flag)	
sh_ts_residual_coding_rice_idx	ue(v)
...	
}	

[00301] - sh_ts_residual_coding_rice_idx указывает параметр Райса, используемый для синтаксической структуры residual_ts_coding(). Когда sh_ts_residual_coding_rice_idx не присутствует, значение sh_ts_residual_coding_rice_idx логически выводится равным 0.

Таблица 41. Процесс извлечения параметра Райса	
Параметр cRiceParam Райса извлекается следующим образом:	
- Если transform_skip_flag[x0][y0][cIdx] равен 1, и sh_ts_residual_coding_disabled_flag равен 0, параметр cRiceParam Райса задается равным sh_ts_residual_coding_rice_idx+1.	
- Иначе, параметр cRiceParam Райса извлекается посредством активации процесса извлечения параметра Райса для abs_remainder [], как указано в таблице 3, с переменной baseLevel, заданной равной 4, индексом cIdx цветового компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом log2TbWidth ширины блока преобразования и двоичным логарифмом log2TbHeight высоты блока преобразования в качестве вводов.	

[00302] В еще одном другом примере, один флаг управления сигнализируется в синтаксисе расширений диапазона наборов параметров картинки, чтобы указывать то, активируется или деактивируется сигнализация параметра Райса для блоков с пропуском преобразования. Когда флаг управления сигнализируется в качестве активированного, один синтаксический элемент дополнительно сигнализируется для того, чтобы указывать параметр Райса этой картинки. Когда флаг управления сигнализируется в качестве деактивированного (например, заданного равным "0"), дополнительный синтаксический элемент не сигнализируется на нижнем уровне для того, чтобы указывать параметр Райса для среза пропуска преобразования, и параметр Райса по умолчанию (например, 1) используется для всего среза пропуска преобразования. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, где ТН является предварительно заданным значением (например, 0, 1, 2). Изменения VVC-проекта показаны в таблице 42 полужирным и курсивным шрифтом. Необходимо отметить, что pps_ts_residual_coding_rice_idx может кодироваться различными способами и/или может иметь максимальное значение. Например, u(n), целое число без знака с использованием n битов, или f(n), битовая строка с фиксированным шаблоном с использованием n битов, записываемая (слева направо) с левым битом сначала, также может использоваться для того, чтобы кодировать/декодировать одинаковый синтаксический элемент.

[00303] Синтаксис расширений диапазона наборов параметров картинки

Таблица 42. Синтаксис остаточного кодирования	
	Дескриптор
pps_range_extensions() {	
...	
<i>pps_ts_residual_coding_rice_flag</i>	<i>u(1)</i>
<i>if(pps_ts_residual_coding_rice_flag)</i>	
<i>pps_ts_residual_coding_rice_idx</i>	<i>ue(v)</i>
}	
...	

[00304] - pps_ts_residual_coding_rice_flag, равный 1, указывает то, что pps_ts_residual_coding_rice_idx может присутствовать в текущей картинке; pps_ts_residual_coding_rice_flag, равный 0, указывает то, что pps_ts_residual_coding_rice_idx не присутствует в текущей картинке. Когда pps_ts_residual_coding_rice_flag не присутствует, значение pps_ts_residual_coding_rice_flag логически выводится равным 0.

[00305] - pps_ts_residual_coding_rice_idx указывает параметр Райса, используемый для синтаксической структуры residual_ts_coding().

Таблица 43. Процесс извлечения параметра Райса

Параметр cRiceParam Райса извлекается следующим образом:

- Если pps_ts_residual_coding_rice_flag равен 1, transform_skip_flag[x0][y0][cIdx] равен 1, и sh_ts_residual_coding_disabled_flag равен 0, параметр cRiceParam Райса задается равным (pps_ts_residual_coding_rice_idx+TH).
- Если transform_skip_flag[x0][y0][cIdx] равен 1, и sh_ts_residual_coding_disabled_flag равен 0, параметр cRiceParam Райса задается равным 1.
- Иначе, параметр cRiceParam Райса извлекается посредством активации процесса извлечения параметра Райса для abs_remainder[], как указано в таблице 3, с переменной baseLevel, заданной равной 4, индексом cIdx цветного компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом log2TbWidth ширины блока преобразования и двоичным логарифмом log2TbHeight высоты блока преобразования в качестве вводов.

[00306] В еще одном другом примере, предлагается использовать только варьирующийся параметр Райса для кодирования синтаксического элемента abs_remainder. Значение применяемого параметра Райса может определяться согласно некоторой закодированной информации текущего блока, например, размеру блока, параметру квантования, битовой глубине, типу преобразования и т.д. В одном конкретном варианте осуществления, предлагается регулировать параметр Райса на основе битовой глубины кодирования и параметра квантования, который применяется к одной CU. Соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, изменения VVC-проекта показаны в таблице 44 полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

Таблица 44. Процесс извлечения параметра Райса

Параметр cRiceParam Райса извлекается следующим образом:

- Если transform_skip_flag[x0][y0][cIdx] равен 1, и sh_ts_residual_coding_disabled_flag равен 0, процесс извлечения для параметра cRiceParam Райса указывается так, как пояснено ниже.

```

if(BitDepth<11)
{
    rice parameter=1
}
else if(BitDepth<13)
{
    if(QPCU<-15)
    {
        rice parameter=6
    }
    else if(QPCU<-10)
    {
        rice parameter=5
    }
    else if(QPCU<0)
    {
        rice parameter=4
    }
    else if(QPCU<10)
    {
        rice parameter=3
    }
    else
    {
        rice parameter=2
    }
}
else if(BitDepth<15)
{

```

5	<pre> if(QP_{CU}<-25) { rice parameter=7 } else if(QP_{CU}<-15) { rice parameter=6 } else if(QP_{CU}<-10) { rice parameter=5 } else { rice parameter=4 } else { if(QP_{CU}<-30) { rice parameter=8 } else if(QP_{CU}<-25) { rice parameter=7 } else if(QP_{CU}<-15) { rice parameter=6 } else if(QP_{CU}<-10) { rice parameter=5 } else { rice parameter=4 } } </pre>
10	
15	
20	
25	- Иначе, параметр cRiceParam Райса извлекается посредством активации процесса извлечения параметра Райса для abs_remainder [], как указано в таблице 3, с переменной baseLevel, заданной равной 4, индексом cldx цветового компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xС, yС) сканирования коэффициентов, двоичным логарифмом log2TbWidth ширины блока преобразования и двоичным логарифмом log2TbHeight высоты блока преобразования в качестве вводов.

[00307] В еще одном другом примере, соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, где ТН является предварительно заданным пороговым значением (например, 33 или 34). Изменения VVC-проекта показаны в таблице 45 полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

	Таблица 45. Процесс извлечения параметра Райса
40	Параметр cRiceParam Райса извлекается следующим образом: - Если <i>transform_skip_flag[x0][y0][cldx]</i> равен 1, и <i>sh_ts_residual_coding_disabled_flag</i> равен 0, процесс извлечения для параметра cRiceParam Райса указывается так, как пояснено ниже. <i>rice parameter=Clip3(1, 8, Floor((TH-BitDepth-QP_{CU})/6))</i> - Иначе, параметр cRiceParam Райса извлекается посредством активации процесса извлечения параметра Райса для abs_remainder [], как указано в таблице 3, с переменной baseLevel, заданной равной 4, индексом cldx цветового компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xС, yС) сканирования коэффициентов, двоичным логарифмом log2TbWidth ширины блока преобразования и двоичным логарифмом log2TbHeight высоты блока преобразования в качестве вводов.

[00308] В еще одном другом примере, соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, где ТН_А и ТН_В являются предварительно заданными пороговыми значениями (например, ТН_А=8, ТН_В=33 или 34). Изменения VVC-проекта показаны в таблице 46 полужирным и курсивным шрифтом, а удаленный контент показан курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или

таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

Таблица 46. Процесс извлечения параметра Райса

5	Параметр cRiceParam Райса извлекается следующим образом: - Если <i>transform_skip_flag</i>[x0][y0][cIdx] равен 1, и <i>sh_ts_residual_coding_disabled_flag</i> равен 0, процесс извлечения для параметра cRiceParam Райса указывается так, как пояснено ниже. $rice\ parameter = Clip3(1, BitDepth - TH_A, Floor((TH_B - BitDepth - QP_{CU})/6))$ - Иначе, параметр cRiceParam Райса извлекается посредством активации процесса извлечения параметра Райса для <i>abs_remainder</i> [], как указано в таблице 3, с переменной <i>baseLevel</i>, заданной равной 4, индексом <i>cIdx</i> цветового компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xС, уС) сканирования коэффициентов, двоичным логарифмом <i>log2TbWidth</i> ширины блока преобразования и двоичным логарифмом <i>log2TbHeight</i> высоты блока преобразования в качестве вводов.
---	---

[00309] В еще одном другом примере, предлагается использовать варьирующийся параметр Райса для кодирования синтаксического элемента *abs_remainder* только тогда, когда новый флаг, например, *extended_precision_processing_flag*, равен 1. Варьирующееся значение может определяться согласно некоторой кодированной информации текущего блока, например, размеру блока, параметру квантования, битовой глубине, типам преобразования и т.д. В одном конкретном варианте осуществления, предлагается регулировать параметр Райса на основе битовой глубины кодирования и параметра квантования, который применяется к одной CU. Соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже. Изменения VVC-проекта показаны в таблице 47 полужирным и курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

Таблица 47. Процесс извлечения параметра Райса

25	Параметр cRiceParam Райса извлекается следующим образом: - Если <i>extended_precision_processing_flag</i> равен 1, <i>transform_skip_flag</i>[x0][y0][cIdx] равен 1, и <i>sh_ts_residual_coding_disabled_flag</i> равен 0, процесс извлечения для параметра cRiceParam Райса указывается так, как пояснено ниже. <pre> if(QP_{CU} < -30) { rice parameter = 8 } else if(QP_{CU} < -25) { rice parameter = 7 } else if(QP_{CU} < -15) { rice parameter = 6 } else if(QP_{CU} < -10) { rice parameter = 5 } else if(QP_{CU} < 0) { rice parameter = 4 } else if(QP_{CU} < 10) { rice parameter = 3 } else if(QP_{CU} < 15) { rice parameter = 2 } else { rice parameter = 1 } </pre> - Если <i>transform_skip_flag</i>[x0][y0][cIdx] равен 1, и <i>sh_ts_residual_coding_disabled_flag</i> равен 0, параметр cRiceParam Райса задается равным 1. - Иначе, параметр cRiceParam Райса извлекается посредством активации процесса извлечения параметра Райса для <i>abs_remainder</i> [], как указано в таблице 3, с переменной <i>baseLevel</i>, заданной равной 4, индексом <i>cIdx</i> цветового компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xС, уС) сканирования коэффициентов, двоичным логарифмом <i>log2TbWidth</i> ширины блока преобразования и двоичным логарифмом <i>log2TbHeight</i> высоты блока преобразования в качестве вводов.
----	--

[00310] В еще одном другом примере, соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, где TN является предварительно заданным пороговым значением (например, 18, 19). Изменения VVC-проекта показаны в таблице 48 полужирным и курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

Таблица 48. Процесс извлечения параметра Райса

Параметр $cRiceParam$ Райса извлекается следующим образом:	
- Если <i>$extended_precision_processing_flag$</i> равен 1, <i>$transform_skip_flag[x0][y0][cIdx]$</i> равен 1, и <i>$sh_ts_residual_coding_disabled_flag$</i> равен 0, процесс извлечения для параметра $cRiceParam$ Райса указывается так, как пояснено ниже.	
<i>$rice_parameter = Clip3(1, 8, (TH - QP)/6)$</i>	
- Если <i>$transform_skip_flag[x0][y0][cIdx]$</i> равен 1, и <i>$sh_ts_residual_coding_disabled_flag$</i> равен 0, параметр $cRiceParam$ Райса задается равным 1.	
- Иначе, параметр $cRiceParam$ Райса извлекается посредством активации процесса извлечения параметра Райса для $abs_remainder$ [], как указано в таблице 3, с переменной $baseLevel$, заданной равной 4, индексом $cIdx$ цветового компонента, местоположением $(x0, y0)$ сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом $log2TbWidth$ ширины блока преобразования и двоичным логарифмом $log2TbHeight$ высоты блока преобразования в качестве вводов.	

[00311] В еще одном другом примере, соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, где TN_A и TN_B являются предварительно заданными пороговыми значениями (например, $TN_A=8$, $TN_B=18$ или 19). Изменения VVC-проекта показаны в таблице 49 полужирным и курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

Таблица 49. Процесс извлечения параметра Райса

Параметр $cRiceParam$ Райса извлекается следующим образом:	
- Если <i>$extended_precision_processing_flag$</i> равен 1, <i>$transform_skip_flag[x0][y0][cIdx]$</i> равен 1, и <i>$sh_ts_residual_coding_disabled_flag$</i> равен 0, процесс извлечения для параметра $cRiceParam$ Райса указывается так, как пояснено ниже.	
<i>$rice_parameter = Clip3(1, BitDepth - TN_A, (TN_B - QP)/6)$</i>	
- Если <i>$transform_skip_flag[x0][y0][cIdx]$</i> равен 1, и <i>$sh_ts_residual_coding_disabled_flag$</i> равен 0, параметр $cRiceParam$ Райса задается равным 1.	
- Иначе, параметр $cRiceParam$ Райса извлекается посредством активации процесса извлечения параметра Райса для $abs_remainder$ [], как указано в таблице 3, с переменной $baseLevel$, заданной равной 4, индексом $cIdx$ цветового компонента, местоположением $(x0, y0)$ сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом $log2TbWidth$ ширины блока преобразования и двоичным логарифмом $log2TbHeight$ высоты блока преобразования в качестве вводов.	

[00312] Фиг. 16 показывает способ для кодирования видео. Способ, например, может применяться к кодеру. На этапе 1610, кодер может принимать видеоввод. Видеоввод, например, может представлять собой передаваемый вживую поток. На этапе 1612, кодер может получать параметр квантования на основе видеоввода. Параметр квантования, например, может вычисляться посредством модуля квантования в кодере. На этапе 1614, кодер может извлекать параметр Райса на основе, по меньшей мере, одного предварительно заданного порогового значения, битовой глубины кодирования и параметра квантования. Параметр Райса, например, используется для сигнализации синтаксиса $abs_remainder$ и dec_abs_level . На этапе 1616, кодер может энтропийно кодировать поток видеобитов на основе параметра Райса. Поток видеобитов, например, может энтропийно кодироваться, чтобы формировать сжатый поток видеобитов.

[00313] В еще одном другом примере, предлагается использовать фиксированное значение (например, 2, 3, 4, 5, 6, 7 или 8) для параметра Райса при кодировании синтаксического элемента $abs_remainder$ только тогда, когда $BitDepth$ больше 10. Фиксированное значение может отличаться при различном условии согласно некоторой закодированной информации текущего блока, например, параметру квантования. Соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован

ниже, где TN является предварительно заданным пороговым значением (например, 18, 19). Изменения VVC-проекта показаны в таблице 50 полужирным и курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

Таблица 50. Процесс извлечения параметра Райса	
	Параметр $cRiceParam$ Райса извлекается следующим образом:
10	- Если $BitDepth$ больше 10, $transform_skip_flag[x0][y0][cIdx]$ равен 1, и $sh_ts_residual_coding_disabled_flag$ равен 0, процесс извлечения для параметра $cRiceParam$ Райса указывается так, как пояснено ниже. <i>$rice_parameter = Clip3(1, 8, (TN - QP)/6)$</i>
	- Если $transform_skip_flag[x0][y0][cIdx]$ равен 1, и $sh_ts_residual_coding_disabled_flag$ равен 0, параметр $cRiceParam$ Райса задается равным 1.
	- Иначе, параметр $cRiceParam$ Райса извлекается посредством активации процесса извлечения параметра Райса для $abs_remainder[]$, как указано в таблице 3, с переменной $baseLevel$, заданной равной 4, индексом $cIdx$ цветового компонента, местоположением $(x0, y0)$ сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом $\log2TbWidth$ ширины блока преобразования и двоичным логарифмом $\log2TbHeight$ высоты блока преобразования в качестве вводов.

[00314] В еще одном другом примере, соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, где TN_A и TN_B являются предварительно заданными пороговыми значениями (например, $TN_A=8$, $TN_B=18$ или 19). Изменения VVC-проекта показаны в таблице 51 полужирным и курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

Таблица 51. Процесс извлечения параметра Райса	
	Параметр $cRiceParam$ Райса извлекается следующим образом:
25	- Если $BitDepth$ больше 10, $transform_skip_flag[x0][y0][cIdx]$ равен 1, и $sh_ts_residual_coding_disabled_flag$ равен 0, процесс извлечения для параметра $cRiceParam$ Райса указывается так, как пояснено ниже. <i>$rice_parameter = Clip3(1, BitDepth - TN_A, (TN_B - QP)/6)$</i>
	- Если $transform_skip_flag[x0][y0][cIdx]$ равен 1, и $sh_ts_residual_coding_disabled_flag$ равен 0, параметр $cRiceParam$ Райса задается равным 1.
	- Иначе, параметр $cRiceParam$ Райса извлекается посредством активации процесса извлечения параметра Райса для $abs_remainder[]$, как указано в таблице 3, с переменной $baseLevel$, заданной равной 4, индексом $cIdx$ цветового компонента, местоположением $(x0, y0)$ сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом $\log2TbWidth$ ширины блока преобразования и двоичным логарифмом $\log2TbHeight$ высоты блока преобразования в качестве вводов.

[00315] В еще одном другом примере, соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, где TN является предварительно заданным пороговым значением (например, 33 или 34). Изменения VVC-проекта показаны в таблице 52 полужирным и курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

Таблица 52. Процесс извлечения параметра Райса	
	Параметр $cRiceParam$ Райса извлекается следующим образом:
40	- Если $BitDepth$ больше 10, $transform_skip_flag[x0][y0][cIdx]$ равен 1, и $sh_ts_residual_coding_disabled_flag$ равен 0, процесс извлечения для параметра $cRiceParam$ Райса указывается так, как пояснено ниже. <i>$rice_parameter = Clip3(1, 8, (TN - BitDepth - QP_{CU})/6)$</i>
	- Если $transform_skip_flag[x0][y0][cIdx]$ равен 1, и $sh_ts_residual_coding_disabled_flag$ равен 0, параметр $cRiceParam$ Райса задается равным 1.
	- Иначе, параметр $cRiceParam$ Райса извлекается посредством активации процесса извлечения параметра Райса для $abs_remainder[]$, как указано в таблице 3, с переменной $baseLevel$, заданной равной 4, индексом $cIdx$ цветового компонента, местоположением $(x0, y0)$ сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом $\log2TbWidth$ ширины блока преобразования и двоичным логарифмом $\log2TbHeight$ высоты блока преобразования в качестве вводов.

[00316] В еще одном другом примере, соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, где TN_A и TN_B являются предварительно заданными пороговыми значениями (например, $TN_A=8$, $TN_B=33$ или 34). Изменения VVC-проекта показаны в таблице 53 полужирным и курсивным шрифтом. Необходимо

отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

5	Таблица 53. Процесс извлечения параметра Райса
10	Параметр cRiceParam Райса извлекается следующим образом: - Если <i>BitDepth</i> больше 10, <i>transform_skip_flag[x0][y0][cIdx]</i> равен 1, и <i>sh_ts_residual_coding_disabled_flag</i> равен 0, процесс извлечения для параметра <i>cRiceParam Райса</i> указывается так, как пояснено ниже. $rice\ parameter = Clip3(1, BitDepth - TH_A, (TH_B - BitDepth - QP_{CU}) / 6)$ - Если <i>transform_skip_flag[x0][y0][cIdx]</i> равен 1, и <i>sh_ts_residual_coding_disabled_flag</i> равен 0, параметр <i>cRiceParam Райса</i> задается равным 1. - Иначе, параметр <i>cRiceParam Райса</i> извлекается посредством активации процесса извлечения параметра Райса для <i>abs_remainder</i> [], как указано в таблице 3, с переменной <i>baseLevel</i>, заданной равной 4, индексом <i>cIdx</i> цветового компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом <i>log2TbWidth</i> ширины блока преобразования и двоичным логарифмом <i>log2TbHeight</i> высоты блока преобразования в качестве вводов.

[00317] Заслуживает упоминания тот факт, что в вышеприведенных иллюстрациях, уравнения, используемые для вычисления конкретных параметров Райса, используются только в качестве примеров, чтобы иллюстрировать предложенные идеи. Для специалистов в области современных технологий кодирования видео, другие функции отображения (или эквивалентно уравнения для отображения) являются уже применимыми к предложенной идее (т.е. определение параметра Райса режима пропуска преобразования на основе бита кодирования и применяемого параметра квантования). Между тем, также следует отметить, что в текущем проектном VVC-решении, значению применяемого параметра квантования разрешается изменяться на уровне групп блоков кодирования. Следовательно, предложенная схема регулирования параметра Райса может предоставлять гибкую адаптацию параметров Райса режима пропуска преобразования на уровне групп блоков кодирования.

[00318] *Сигнализация информации для регулярного остаточного кодирования и остаточного кодирования с пропуском преобразования*

[00319] Согласно двадцать восьмому аспекту раскрытия, предлагается сигнализировать параметр Райса двоичных кодовых слов для кодирования некоторых синтаксических элементов, например, *abs_remainder* при остаточном кодировании с пропуском преобразования, параметров сдвига и смещения для извлечения параметра Райса, используемого для *abs_remainder/dec_abs_level* при регулярном остаточном кодировании, и определять то, следует или нет сигнализировать, согласно некоторой закодированной информации текущего блока, например, параметру квантования или битовой глубине кодирования, ассоциированной с ТВ/СВ и/или со срезом/профилем, и/или согласно новому флагу, ассоциированному с уровнем ТВ/СВ/среза/картинки/последовательности, например, *sps_residual_coding_info_present_in_sh_flag*.

[00320] В одном примере, один флаг управления сигнализируется в заголовке среза, чтобы указывать то, активируется или деактивируется сигнализация параметра Райса для блоков с пропуском преобразования и сигнализация параметров сдвига и/или смещения для извлечения параметра Райса в блоках преобразования. Когда флаг управления сигнализируется в качестве активированного, один синтаксический элемент дополнительно сигнализируется для каждого среза пропуска преобразования, чтобы указывать параметр Райса этого среза, и два синтаксических элемента дополнительно сигнализируются для каждого среза преобразования, чтобы указывать параметры сдвига и/или смещения для извлечения параметра Райса этого среза. Когда флаг управления сигнализируется в качестве деактивированного (например, заданного равным "0"), дополнительный синтаксический элемент не сигнализируется на нижнем уровне для того, чтобы указывать параметр Райса для среза пропуска преобразования, и параметр Райса по умолчанию (например, 1) используется для всего среза пропуска

преобразования, и дополнительный синтаксический элемент не сигнализируется на нижнем уровне для того, чтобы указывать параметры сдвига и смещения для извлечения параметра Райса для среза преобразования, и параметры сдвига и/или смещения по умолчанию (например, 0) используются для всего среза преобразования. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован

ниже, где ТН является предварительно заданным значением (например, 0, 1, 2). Изменения VVC-проекта показаны в таблице 54 полужирным и курсивным шрифтом. Необходимо отметить, что *sh_residual_coding_rice_shift*, *sh_residual_coding_rice_offset* и *sh_ts_residual_coding_rice_index* могут кодироваться различными способами и/или могут иметь максимальное значение. Например, *u(n)*, целое число без знака с использованием *n* битов, или *f(n)*, битовая строка с фиксированным шаблоном с использованием *n* битов, записываемая (слева направо) с левым битом сначала, также может использоваться для того, чтобы кодировать/декодировать одинаковый синтаксический элемент.

[00321] Фиг. 17 показывает способ для декодирования видео. Способ, например, может применяться к кодеру. На этапе 1710, кодер может принимать видеоввод. На этапе 1712, кодер может сигнализировать параметр Райса двоичных кодовых слов для кодирования синтаксических элементов. Синтаксические элементы кодирования могут включать в себя *abs_remainder* при остаточном кодировании с пропуском преобразования. На этапе 1714, кодер может энтропийно кодировать поток видеобитов на основе параметра Райса и видеоввода.

[00322] *Синтаксис заголовков срезов*

Таблица 54. Синтаксис остаточного кодирования	
<i>slice_header()</i> {	Дескриптор
...	
<i>if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)</i>	
<i>sh_ts_residual_coding_disabled_flag</i>	<i>u(1)</i>
<i>sh_residual_coding_rice_flag</i>	<i>u(1)</i>
<i>if(sh_ts_residual_coding_rice_flag) {</i>	
<i>sh_residual_coding_rice_shift</i>	<i>ue(v)</i>
<i>sh_residual_coding_rice_offset</i>	<i>ue(v)</i>
<i>if(!sh_ts_residual_coding_disabled_flag)</i>	
<i>sh_ts_residual_coding_rice_index</i>	<i>ue(v)</i>
<i>}</i>	
...	

[00323] - *sh_residual_coding_rice_flag*, равный 1, указывает то, что *sh_residual_coding_rice_shift*, *sh_residual_coding_rice_offset* и *sh_residual_coding_rice_index* могут присутствовать в текущем срезе; *sh_residual_coding_rice_flag*, равный 0, указывает то, что *sh_residual_coding_rice_shift*, *sh_residual_coding_rice_offset* и *sh_residual_coding_rice_index* не присутствуют в текущем срезе.

[00324] - *sh_residual_coding_rice_shift* указывает параметр сдвига, используемый для процесса извлечения параметра Райса для *abs_remainder[]* и *dec_abs_level[]*. Когда *sh_residual_coding_rice_shift* не присутствует, значение *sh_residual_coding_rice_shift* логически выводится равным 0.

[00325] - *sh_residual_coding_rice_offset* указывает параметр смещения, используемый для процесса извлечения параметра Райса для *abs_remainder[]* и *dec_abs_level[]*. Когда *sh_residual_coding_rice_offset* не присутствует, значение *sh_residual_coding_rice_offset* логически выводится равным 0.

[00326] - *sh_ts_residual_coding_rice_index* указывает параметр Райса, используемый

для синтаксической структуры residual_ts_coding(). Когда sh_ts_residual_coding_rice_index не присутствует, значение sh_ts_residual_coding_rice_index логически выводится равным 0.

5

Таблица 55. Процесс извлечения параметра Райса

Процесс преобразования в двоичную форму для abs_remainder[]

...

Параметр cRiceParam Райса извлекается следующим образом:

- Если sh_residual_coding_rice_flag равен 1, transform_skip_flag[x0][y0][cIdx] равен 1, и sh_ts_residual_coding_disabled_flag равен 0, параметр cRiceParam Райса задается равным (sh_ts_residual_coding_rice_index+TH).
- Если transform_skip_flag[x0][y0][cIdx] равен 1, и sh_ts_residual_coding_disabled_flag равен 0, параметр cRiceParam Райса задается равным 1.

10

- Иначе, параметр cRiceParam Райса извлекается посредством активации процесса извлечения параметра Райса для abs_remainder[], как указано в таблице 3, с переменной baseLevel, заданной равной 4, индексом cIdx цветового компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом log2TbWidth ширины блока преобразования и двоичным логарифмом log2TbHeight высоты блока преобразования в качестве вводов.

...

Таблица 56. Процесс извлечения параметра Райса

Процесс извлечения параметра Райса для abs_remainder[] и dec_abs_level[]

15

Вводы в этот процесс представляют собой базовый уровень baseLevel, индекс cIdx цветового компонента, местоположение (x0, y0) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм log2TbWidth ширины блока преобразования, двоичный логарифм log2TbWidth ширины блока преобразования, двоичный логарифм log2TbHeight высоты блока преобразования, sh_residual_coding_rice_shift и sh_residual_coding_rice_offset.

Выход этого процесса представляет собой параметр cRiceParam Райса.

С учетом массива AbsLevel[x][y] для блока преобразования с индексом cIdx компонента и левым верхним местоположением (x0, y0) сигнала яркости, переменная locSumAbs извлекается так, как указано посредством следующего псевдокода:

20

```
locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
locSumAbs+=AbsLevel[xC+1][yC]
if(xC<(1<<log2TbWidth)-2)
locSumAbs+=AbsLevel[xC+2][yC]
if(yC<(1<<log2TbHeight)-1)
locSumAbs+=AbsLevel[xC+1][yC+1]
(1494)
}
```

25

```
if(yC<(1<<log2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]
if(yC<(1<<log2TbHeight)-2)
locSumAbs+=AbsLevel[xC][yC+2]
}
```

```
locSumAbs=Clip3(0, 31, ((locSumAbs+sh_residual_coding_rice_offset)>>sh_residual_coding_rice_shift)-baseLevel*5)
```

С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается так, как указано в таблице 4.

cRiceParam=cRiceParam+sh_residual_coding_rice_shift

30

Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:

ZeroPos[n]=(QState<2?1:2)<<cRiceParam

[00327] В другом примере, один флаг управления сигнализируется в наборе параметров последовательности (или в синтаксисе расширений диапазона набора параметров последовательности), чтобы указывать то, активируется или деактивируется сигнализация параметра Райса для блоков с пропуском преобразования и сигнализация параметров сдвига и/или смещения для извлечения параметра Райса в блоках преобразования. Когда флаг управления сигнализируется в качестве активированного, один синтаксический элемент дополнительно сигнализируется для каждого среза пропуска преобразования, чтобы указывать параметр Райса этого среза, и два синтаксических элемента дополнительно сигнализируются для каждого среза преобразования, чтобы указывать параметры сдвига и/или смещения для извлечения параметра Райса этого среза. Когда флаг управления сигнализируется в качестве деактивированного (например, заданного равным "0"), дополнительный синтаксический элемент не сигнализируется на нижнем уровне для того, чтобы указывать параметр Райса для среза пропуска преобразования, и параметр Райса по умолчанию (например, 1) используется для всего среза пропуска преобразования, и дополнительный синтаксический элемент не сигнализируется на нижнем уровне для того, чтобы указывать параметры сдвига и/или смещения для извлечения параметра Райса для среза преобразования, и параметры сдвига и/или смещения по умолчанию (например, 0)

используются для всего среза преобразования. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, где TH является

предварительно заданным значением (например, 0, 1, 2). Изменения VVC-проекта показаны в таблице 57 полужирным и курсивным шрифтом. Необходимо отметить, что `sh_residual_coding_rice_shift`, `sh_residual_coding_rice_offset` и `sh_ts_residual_coding_rice_idx` могут кодироваться различными способами и/или могут иметь максимальное значение. Например, `u(n)`, целое число без знака с использованием `n` битов, или `f(n)`, битовая строка с фиксированным шаблоном с использованием `n` битов, записываемая (слева направо) с левым битом сначала, также может использоваться для того, чтобы кодировать/декодировать одинаковый синтаксический элемент.

[00328] *RBSP-синтаксис наборов параметров последовательности*

Таблица 57. Синтаксис остаточного кодирования	
<code>seq_parameter_set_rbsp() {</code>	Дескриптор
<code>...</code>	
<code>sps_sign_data_hiding_enabled_flag</code>	<code>u(1)</code>
<i><code>sps_residual_coding_info_present_in_sh_flag</code></i>	<i><code>u(1)</code></i>
<code>sps_virtual_boundaries_enabled_flag</code>	<code>u(1)</code>
<code>...</code>	
<code>}</code>	

[00329] - `sps_residual_coding_info_present_in_sh_flag`, равный 1, указывает то, что `sh_residual_coding_rice_shift`, `sh_residual_coding_rice_offset` и `sh_ts_residual_coding_rice_idx` могут присутствовать в синтаксических SH-структурах, ссылающихся на SPS; `sps_residual_coding_info_present_in_sh_flag`, равный 0, указывает то, что `sh_residual_coding_rice_shift`, `sh_residual_coding_rice_offset` и `sh_ts_residual_coding_rice_idx` не присутствуют в синтаксических SH-структурах, ссылающихся на SPS. Когда `sps_residual_coding_info_present_in_sh_flag` не присутствует, значение `sps_residual_coding_info_present_in_sh_flag` логически выводится равным 0.

[00330] *Синтаксис заголовков срезов*

Таблица 58. Синтаксис остаточного кодирования	
<code>slice_header() {</code>	Дескриптор
<code>...</code>	
<code>if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)</code>	
<code>sh_ts_residual_coding_disabled_flag</code>	<code>u(1)</code>
<i><code>if(sps_ts_residual_coding_rice_enabled_flag) {</code></i>	
<i><code>sh_residual_coding_rice_shift</code></i>	<i><code>ue(v)</code></i>
<i><code>sh_residual_coding_rice_offset</code></i>	<i><code>ue(v)</code></i>
<i><code>if(!sh_ts_residual_coding_disabled_flag)</code></i>	
<i><code>sh_ts_residual_coding_rice_index</code></i>	<i><code>ue(v)</code></i>
<code>}</code>	
<code>...</code>	
<code>}</code>	

[00331] - `sh_residual_coding_rice_shift` указывает параметр сдвига, используемый для процесса извлечения параметра Райса для `abs_remainder[]` и `dec_abs_level[]`. Когда `sh_residual_coding_rice_shift` не присутствует, значение `sh_residual_coding_rice_shift` логически выводится равным 0.

[00332] - `sh_residual_coding_rice_offset` указывает параметр смещения, используемый для процесса извлечения параметра Райса для `abs_remainder[]` и `dec_abs_level[]`. Когда `sh_residual_coding_rice_offset` не присутствует, значение `sh_residual_coding_rice_offset`

логически выводится равным 0.

[00333] - `sh_ts_residual_coding_rice_idx` указывает параметр Райса, используемый для синтаксической структуры `residual_ts_coding()`. Когда `sh_ts_residual_coding_rice_index` не присутствует, значение `sh_ts_residual_coding_rice_index` логически выводится равным 0.

Таблица 59. Процесс извлечения параметра Райса

Процесс преобразования в двоичную форму для `abs_remainder[]`

...

Параметр `cRiceParam` Райса извлекается следующим образом:

- Если `sps_ts_residual_coding_info_flag` равен 1, `transform_skip_flag[x0][y0][cIdx]` равен 1, и `sh_ts_residual_coding_disabled_flag` равен 0, параметр `cRiceParam` Райса задается равным (`sh_ts_residual_coding_rice_idx+TH`).

- Если `transform_skip_flag[x0][y0][cIdx]` равен 1, и `sh_ts_residual_coding_disabled_flag` равен 0, параметр `cRiceParam` Райса задается равным 1.

- Иначе, параметр `cRiceParam` Райса извлекается посредством активации процесса извлечения параметра Райса для `abs_remainder[]`, как указано в таблице 3, с переменной `baseLevel`, заданной равной 4, индексом `cIdx` цветового компонента, местоположением (`x0`, `y0`) сигнала яркости, текущим местоположением (`xC`, `yC`) сканирования коэффициентов, двоичным логарифмом `log2TbWidth` ширины блока преобразования и двоичным логарифмом `log2TbHeight` высоты блока преобразования в качестве вводов.

...

Таблица 60. Процесс извлечения параметра Райса

Процесс извлечения параметра Райса для `abs_remainder[]` и `dec_abs_level[]`

Вводы в этот процесс представляют собой базовый уровень `baseLevel`, индекс `cIdx` цветового компонента, местоположение (`x0`, `y0`) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (`xC`, `yC`) сканирования коэффициентов, двоичный логарифм `log2TbWidth` ширины блока преобразования, двоичный логарифм `log2TbHeight` высоты блока преобразования, `sh_residual_coding_rice_shift` и `sh_residual_coding_rice_offset`.

Вывод этого процесса представляет собой параметр `cRiceParam` Райса.

С учетом массива `AbsLevel[x][y]` для блока преобразования с индексом `cIdx` компонента и левым верхним местоположением (`x0`, `y0`) сигнала яркости, переменная `locSumAbs` извлекается так, как указано посредством следующего псевдокода:

```
locSumAbs=0
if(xC<(1<<log2TbWidth)-1) {
locSumAbs+=AbsLevel[xC+1][yC]
if(xC<(1<<log2TbWidth)-2)
locSumAbs+=AbsLevel[xC+2][yC]
if(yC<(1<<log2TbHeight)-1)
locSumAbs+=AbsLevel[xC+1][yC+1]
(1494)
```

```
}
if(yC<(1<<log2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]
if(yC<(1<<log2TbHeight)-2)
locSumAbs+=AbsLevel[xC][yC+2]
}
```

```
locSumAbs=Clip3(0, 31, (locSumAbs+sh_residual_coding_rice_offset)>>sh_residual_coding_rice_shift)-baseLevel*5)
```

С учетом переменной `locSumAbs`, параметр `cRiceParam` Райса извлекается так, как указано в таблице 4.

```
cRiceParam=cRiceParam+sh_residual_coding_rice_shift
```

Когда `baseLevel` равен 0, переменная `ZeroPos[n]` извлекается следующим образом:

```
ZeroPos[n]=(QState<2?1:2)<<cRiceParam
```

[00334] В еще одном другом примере, один синтаксический элемент сигнализируется для каждого среза пропуска преобразования, чтобы указывать параметр Райса этого среза, и два синтаксических элемента сигнализируются для каждого среза

преобразования, чтобы указывать параметры сдвига и/или смещения для извлечения параметра Райса этого среза. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже. Изменения VVC-проекта показаны в таблице 61 полужирным и курсивным шрифтом. Необходимо отметить, что `sh_residual_coding_rice_shift`, `sh_residual_coding_rice_offset` и `sh_ts_residual_coding_rice_idx` могут кодироваться различными способами и/или могут иметь максимальное значение. Например, `u(n)`, целое число без знака с использованием `n` битов, или `f(n)`, битовая строка с фиксированным шаблоном с использованием `n` битов, записываемая (слева направо) с левым битом сначала, также может использоваться для того, чтобы кодировать/декодировать одинаковый синтаксический элемент.

[00335] *Синтаксис заголовков срезов*

Таблица 61. Синтаксис остаточного кодирования

<code>slice_header()</code> {	Дескриптор
...	

	if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)	
	sh_ts_residual_coding_disabled_flag	u(1)
	if(!sh_ts_residual_coding_disabled_flag)	
	sh_ts_residual_coding_rice_idx	ue(v)
5	sh_residual_coding_rice_shift	ue(v)
	sh_residual_coding_rice_offset	ue(v)
	...	
	}	

[00336] - sh_ts_residual_coding_rice_idx указывает параметр Райса, используемый для синтаксической структуры residual_ts_coding(). Когда sh_ts_residual_coding_rice_idx не присутствует, значение sh_ts_residual_coding_rice_idx логически выводится равным 0.

[00337] - sh_residual_coding_rice_offset указывает параметр смещения, используемый для процесса извлечения параметра Райса для abs_remainder[] и dec_abs_level[]. Когда sh_residual_coding_rice_offset не присутствует, значение sh_residual_coding_rice_offset логически выводится равным 0.

[00338] - sh_residual_coding_rice_shift указывает параметр сдвига, используемый для процесса выведения параметра Райса для abs_remainder[] и dec_abs_level[]. Когда sh_residual_coding_rice_shift не присутствует, значение sh_residual_coding_rice_shift логически выводится равным 0.

Таблица 62. Процесс извлечения параметра Райса

	Процесс преобразования в двоичную форму для abs_remainder[]
	...
	Параметр cRiceParam Райса извлекается следующим образом:
	- Если transform_skip_flag[x0][y0][cldx] равен 1, и sh_ts_residual_coding_disabled_flag равен 0, параметр cRiceParam Райса задается равным sh_ts_residual_coding_rice_idx+1.
25	- Иначе, параметр cRiceParam Райса извлекается посредством активации процесса извлечения параметра Райса для abs_remainder[], как указано в таблице 3, с переменной baseLevel, заданной равной 4, индексом cldx цветового компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом log2TbWidth ширины блока преобразования и двоичным логарифмом log2TbHeight высоты блока преобразования в качестве вводов.
	...

Таблица 63. Процесс извлечения параметра Райса

	Процесс извлечения параметра Райса для abs_remainder[] и dec_abs_level[]
30	Вводы в этот процесс представляют собой базовый уровень baseLevel, индекс cldx цветового компонента, местоположение (x0, y0) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм log2TbWidth ширины блока преобразования, двоичный логарифм log2TbHeight высоты блока преобразования, sh_residual_coding_rice_shift и sh_residual_coding_rice_offset.
	Вывод этого процесса представляет собой параметр cRiceParam Райса.
	С учетом массива AbsLevel[x][y] для блока преобразования с индексом cldx компонента и левым верхним местоположением (x0, y0) сигнала яркости, переменная locSumAbs извлекается так, как указано посредством следующего псевдокода:
35	locSumAbs=0
	if(xC<(1<<log2TbWidth)-1) {
	locSumAbs+=AbsLevel[xC+1][yC]
	if(xC<(1<<log2TbWidth)-2)
	locSumAbs+=AbsLevel[xC+2][yC]
	if(yC<(1<<log2TbHeight)-1)
	locSumAbs+=AbsLevel[xC+1][yC+1]
	(1494)
	}
40	if(yC<(1<<log2TbHeight)-1) {
	locSumAbs+=AbsLevel[xC][yC+1]
	if(yC<(1<<log2TbHeight)-2)
	locSumAbs+=AbsLevel[xC][yC+2]
	}
	locSumAbs=Clip3(0, 31, ((locSumAbs+sh_residual_coding_rice_offset)>>sh_residual_coding_rice_shift)-baseLevel*5)
	С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается так, как указано в таблице 4.
	cRiceParam=cRiceParam+sh_residual_coding_rice_shift
45	Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:
	ZeroPos[n]=(QState<2?1:2)<<cRiceParam

[00339] В еще одном другом примере, один флаг управления сигнализируется в синтаксисе расширений диапазона наборов параметров картинки, чтобы указывать то, активируется или деактивируется сигнализация параметра Райса для блоков с

пропуском преобразования и сигнализация параметров сдвига и/или смещения для извлечения параметра Райса в блоках преобразования. Когда флаг управления сигнализируется в качестве активированного, один синтаксический элемент дополнительно сигнализируется для того, чтобы указывать параметр Райса для остаточного кодирования с пропуском преобразования этой картинки, и два синтаксических элемента дополнительно сигнализируются для того, чтобы регулярное остаточное кодирование указывает параметры сдвига и/или смещения для извлечения параметра Райса этой картинки. Когда флаг управления сигнализируется в качестве деактивированного (например, заданного равным "0"), дополнительный синтаксический элемент не сигнализируется на нижнем уровне для того, чтобы указывать параметр Райса для остаточного кодирования с пропуском преобразования, и параметр Райса по умолчанию (например, 1) используется для всего остаточного кодирования с пропуском преобразования, и дополнительный синтаксический элемент не сигнализируется на нижнем уровне для того, чтобы указывать параметры сдвига и/или смещения для извлечения параметра Райса для регулярного остаточного кодирования, и параметры сдвига и/или смещения по умолчанию (например, 0) используются для всего регулярного остаточного кодирования. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, где ТН является предварительно заданным значением (например, 0, 1, 2). Изменения VVC-проекта показаны в таблице 64 полужирным и курсивным шрифтом. Необходимо отметить, что *pps_residual_coding_rice_shift*, *pps_residual_coding_rice_offset* и *pps_ts_residual_coding_rice_idx* могут кодироваться различными способами и/или могут иметь максимальное значение. Например, $u(n)$, целое число без знака с использованием n битов, или $f(n)$, битовая строка с фиксированным шаблоном с использованием n битов, записываемая (слева направо) с левым битом сначала, также может использоваться для того, чтобы кодировать/декодировать одинаковый синтаксический элемент.

[00340] *Синтаксис расширений диапазона наборов параметров картинки*

Таблица 64. Синтаксис остаточного кодирования		
30	<i>pps_range_extensions() {</i>	Дескриптор
	...	
	<i>pps_residual_coding_info_flag</i>	<i>u(1)</i>
	<i>if(pps_ts_residual_coding_rice_flag)</i>	
	<i>pps_residual_coding_rice_shift</i>	<i>ue(v)</i>
	<i>pps_residual_coding_rice_offset</i>	<i>ue(v)</i>
35	<i>pps_ts_residual_coding_rice_idx</i>	<i>ue(v)</i>
	<i>}</i>	
	...	

[00341] - *pps_residual_coding_info_flag*, равный 1, указывает, что *pps_residual_coding_rice_shift*, *pps_residual_coding_rice_offset* и *pps_ts_residual_coding_rice_idx* могут присутствовать в текущей картинке; *pps_residual_coding_info_flag*, равный 0, указывает, что *pps_residual_coding_rice_shift*, *pps_residual_coding_rice_offset* и *pps_ts_residual_coding_rice_idx* не присутствуют в текущей картинке. Когда *pps_residual_coding_info_flag* не присутствует, значение *pps_residual_coding_info_flag* логически выводится равным 0.

[00342] - *pps_residual_coding_rice_shift* указывает параметр сдвига, используемый для процесса извлечения параметра Райса для *abs_remainder[]* и *dec_abs_level[]*. Когда *pps_residual_coding_rice_shift* не присутствует, значение *pps_residual_coding_rice_shift* логически выводится равным 0.

[00343] - *pps_residual_coding_rice_offset* указывает параметр смещения, используемый

для процесса извлечения параметра Райса для `abs_remainder[]` и `dec_abs_level[]`. Когда `pps_residual_coding_rice_offset` не присутствует, значение `pps_residual_coding_rice_offset` логически выводится равным 0.

[00344] - `pps_ts_residual_coding_rice_idx` указывает параметр Райса, используемый для синтаксической структуры `residual_ts_coding()`. Когда `pps_ts_residual_coding_rice_idx` не присутствует, значение `pps_ts_residual_coding_rice_idx` логически выводится равным 0.

Таблица 65. Процесс извлечения параметра Райса

Процесс преобразования в двоичную форму для `abs_remainder[]`

...
Параметр `cRiceParam` Райса извлекается следующим образом:

- Если `pps_ts_residual_coding_rice_flag` равен 1, `transform_skip_flag[x0][y0][cIdx]` равен 1, и `sh_ts_residual_coding_disabled_flag` равен 0, параметр `cRiceParam` Райса задается равным (`pps_ts_residual_coding_rice_idx+TH`).

- Если `transform_skip_flag[x0][y0][cIdx]` равен 1, и `sh_ts_residual_coding_disabled_flag` равен 0, параметр `cRiceParam` Райса задается равным 1.

- Иначе, параметр `cRiceParam` Райса извлекается посредством активации процесса извлечения параметра Райса для `abs_remainder[]`, как указано в таблице 3, с переменной `baseLevel`, заданной равной 4, индексом `cIdx` цветового компонента, местоположением (`x0`, `y0`) сигнала яркости, текущим местоположением (`xC`, `yC`) сканирования коэффициентов, двоичным логарифмом `log2TbWidth` ширины блока преобразования и двоичным логарифмом `log2TbHeight` высоты блока преобразования в качестве вводов.

...

Таблица 66. Процесс извлечения параметра Райса

Процесс извлечения параметра Райса для `abs_remainder[]` и `dec_abs_level[]`

Вводы в этот процесс представляют собой базовый уровень `baseLevel`, индекс `cIdx` цветового компонента, местоположение (`x0`, `y0`) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (`xC`, `yC`) сканирования коэффициентов, двоичный логарифм `log2TbWidth` ширины блока преобразования, двоичный логарифм `log2TbHeight` высоты блока преобразования, `pps_residual_coding_rice_shift` и `pps_residual_coding_rice_offset`.

Вывод этого процесса представляет собой параметр `cRiceParam` Райса.

С учетом массива `AbsLevel[x][y]` для блока преобразования с индексом `cIdx` компонента и левым верхним местоположением (`x0`, `y0`) сигнала яркости, переменная `locSumAbs` извлекается так, как указано посредством следующего псевдокода:

`locSumAbs=0`

if(`xC`<(1<<`log2TbWidth`)-1) {

`locSumAbs+=AbsLevel[xC+1][yC]`

if(`xC`<(1<<`log2TbWidth`)-2)

`locSumAbs+=AbsLevel[xC+2][yC]`

if(`yC`<(1<<`log2TbHeight`)-1)

`locSumAbs+=AbsLevel[xC+1][yC+1]`

(1494)

}

if(`yC`<(1<<`log2TbHeight`)-1) {

`locSumAbs+=AbsLevel[xC][yC+1]`

if(`yC`<(1<<`log2TbHeight`)-2)

`locSumAbs+=AbsLevel[xC][yC+2]`

}

`locSumAbs=Clip3(0, 31, ((locSumAbs+pps_residual_coding_rice_offset)>>pps_residual_coding_rice_shift)-baseLevel*5)`

С учетом переменной `locSumAbs`, параметр `cRiceParam` Райса извлекается так, как указано в таблице 4.

`cRiceParam=cRiceParam+pps_residual_coding_rice_shift`

Когда `baseLevel` равен 0, переменная `ZeroPos[n]` извлекается следующим образом:

`ZeroPos[n]=(QState<2?1:2)<<cRiceParam`

[00345] Согласно двадцать девятому аспекту раскрытия, предлагается использовать различные параметры Райса для кодирования некоторых синтаксических элементов, например, `abs_remainder` при остаточном кодировании с пропуском преобразования, параметры сдвига и смещения для извлечения параметра Райса, используемые для `abs_remainder/dec_abs_level` при регулярном остаточном кодировании, и определять то, какой из них следует использовать, согласно некоторой кодированной информации текущего блока, например, параметру квантования или битовой глубине кодирования, ассоциированной с ТВ/СВ и/или со срезом/профилем, и/или согласно новому флагу, ассоциированному с уровнем ТВ/СВ/среза/картинки/последовательности, например, `pps_residual_coding_info_present_in_sh_flag`.

[00346] В одном примере, один флаг управления сигнализируется в заголовке среза, чтобы указывать то, активируются или деактивируются процесс извлечения параметра Райса для блоков с пропуском преобразования и процесс извлечения параметров сдвига и/или смещения для параметра Райса в блоках преобразования. Когда флаг управления сигнализируется в качестве активированного, параметр Райса может отличаться при

различном условии согласно некоторой кодированной информации текущего блока, например, параметру квантования и битовой глубине. Кроме того, параметры сдвига и/или смещения для извлечения параметра Райса при регулярном остаточном кодировании могут отличаться при различном условии согласно некоторой кодированной информации текущего блока, например, параметру квантования и битовой глубине. Когда флаг управления сигнализируется в качестве деактивированного (например, заданного равным "0"), параметр Райса по умолчанию (например, 1) используется для всего среза пропуска преобразования, и параметры сдвига и/или смещения по умолчанию (например, 0) используются для всего среза преобразования. Пример соответствующего процесса декодирования на основе VVC-проекта проиллюстрирован ниже, где TH_A и TH_B являются предварительно заданными пороговыми значениями (например, $TH_A=8$, $TH_B=18$ или 19). Изменения VVC-проекта показаны в таблице 67 полужирным и курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

[00347] *Синтаксис заголовков срезов*

Таблица 67. Синтаксис остаточного кодирования	
<code>slice_header () {</code>	Дескриптор
<code>...</code>	
<code>if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)</code>	
<code>sh_ts_residual_coding_disabled_flag</code>	<i><code>u(1)</code></i>
<i><code>sh_residual_coding_rice_flag</code></i>	<i><code>u(1)</code></i>
<code>...</code>	

[00348] - `sh_residual_coding_rice_flag`, равный 1, указывает то, что зависимый от битовой глубины процесс извлечения параметра Райса используется в текущем срезе; `sh_residual_coding_rice_flag`, равный 0, указывает то, что зависимый от битовой глубины процесс извлечения параметра Райса не используется в текущем срезе.

Таблица 68. Процесс извлечения параметра Райса	
Процесс преобразования в двоичную форму для <code>abs_remainder[]</code>	
...	
Параметр <code>cRiceParam</code> Райса извлекается следующим образом:	
- Если <i><code>sh_residual_coding_rice_flag</code> равен 1, <code>transform_skip_flag[x0][y0][cIdx]</code> равен 1, и <code>sh_ts_residual_coding_disabled_flag</code> равен 0, параметр <code>cRiceParam</code> Райса задается равным <code>Clip3(1, BitDepth-TH_A, (TH_B-QP)/6)</code>.</i>	
- Если <code>transform_skip_flag[x0][y0][cIdx]</code> равен 1, и <code>sh_ts_residual_coding_disabled_flag</code> равен 0, параметр <code>cRiceParam</code> Райса задается равным 1.	
- Иначе, параметр <code>cRiceParam</code> Райса извлекается посредством активации процесса извлечения параметра Райса для <code>abs_remainder[]</code> , как указано в таблице 3, с переменной <code>baseLevel</code> , заданной равной 4, индексом <code>cIdx</code> цветового компонента, местоположением (<code>x0</code> , <code>y0</code>) сигнала яркости, текущим местоположением (<code>xC</code> , <code>yC</code>) сканирования коэффициентов, двоичным логарифмом <code>log2TbWidth</code> ширины блока преобразования и двоичным логарифмом <code>log2TbHeight</code> высоты блока преобразования в качестве вводов.	
...	

Таблица 69. Процесс извлечения параметра Райса	
Процесс извлечения параметра Райса для <code>abs_remainder[]</code> и <code>dec_abs_level[]</code>	
Вводы в этот процесс представляют собой базовый уровень <code>baseLevel</code> , индекс <code>cIdx</code> цветового компонента, местоположение (<code>x0</code> , <code>y0</code>) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (<code>xC</code> , <code>yC</code>) сканирования коэффициентов, двоичный логарифм <code>log2TbHeight</code> высоты блока преобразования, <i><code>sh_residual_coding_rice_flag</code>.</i>	
Вывод этого процесса представляет собой параметр <code>cRiceParam</code> Райса.	
С учетом массива <code>AbsLevel[x][y]</code> для блока преобразования с индексом <code>cIdx</code> компонента и левым верхним местоположением (<code>x0</code> , <code>y0</code>) сигнала яркости, переменная <code>locSumAbs</code> извлекается так, как указано посредством следующего псевдокода:	
<pre> ShiftRice=<i><code>sh_residual_coding_rice_flag</code></i>?(<i><code>BitDepth</code></i>>10)?Floor(Log2(4*(<i><code>BitDepth</code></i>-10))):0:0 OffsetRice=<i><code>sh_residual_coding_rice_flag</code></i>?(<i><code>ShiftRice</code></i>>0)?(1<<(ShiftRice-1)):0:0 locSumAbs=0 if(xC<(1<<log2TbWidth)-1) { locSumAbs+=AbsLevel[xC+1][yC] if(xC<(1<<log2TbWidth)-2) </pre>	

```

locSumAbs+=AbsLevel[xC+2][yC]
if(yC<(1<<log2TbHeight)-1)
locSumAbs+=AbsLevel[xC+1][yC+1]
(1494)
}
if(yC<(1<<log2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]
if(yC<(1<<log2TbHeight)-2)
locSumAbs+=AbsLevel[xC][yC+2]
}
locSumAbs=Clip3(0, 31, ((locSumAbs+OffsetRice)>>ShiftRice)-baseLevel*5)
С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается так, как указано в таблице 4.
cRiceParam=cRiceParam+ShiftRice
Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом:
ZeroPos[n]=(QState<2?1:2)<<cRiceParam

```

[00349] В еще одном другом примере, соответствующий процесс декодирования на основе VVC-проекта проиллюстрирован ниже, где ТН является предварительно заданным пороговым значением (например, 18, 19). Изменения VVC-проекта показаны в таблице 70 полужирным и курсивным шрифтом. Необходимо отметить, что одинаковые логики могут реализовываться по-разному на практике. Например, некоторые уравнения или таблица поиска также могут использоваться для того, чтобы извлекать одинаковые параметры Райса.

Таблица 70. Процесс извлечения параметра Райса

	Процесс преобразования в двоичную форму для abs_remainder[]
20	... Параметр cRiceParam Райса извлекается следующим образом:
	<i>- Если BitDepth больше 10, transform_skip_flag[x0][y0][cIdx] равен 1, и sh_ts_residual_coding_disabled_flag равен 0, процесс извлечения для параметра cRiceParam Райса указывается так, как пояснено ниже.</i>
	<i>rice parameter=Clip3(1, 8, (ТН-QP)/6)</i>
	<i>- Если transform_skip_flag[x0][y0][cIdx] равен 1, и sh_ts_residual_coding_disabled_flag равен 0, параметр cRiceParam Райса задается равным 1.</i>
	<i>- Иначе, параметр cRiceParam Райса извлекается посредством активации процесса извлечения параметра Райса для abs_remainder [], как указано в таблице 3, с переменной baseLevel, заданной равной 4, индексом cIdx цветового компонента, местоположением (x0, y0) сигнала яркости, текущим местоположением (xC, yC) сканирования коэффициентов, двоичным логарифмом log2TbWidth ширины блока преобразования и двоичным логарифмом log2TbHeight высоты блока преобразования в качестве вводов.</i>
	Процесс извлечения параметра Райса для abs_remainder[] и dec_abs_level[]
	Вводы в этот процесс представляют собой базовый уровень baseLevel, индекс cIdx цветового компонента, местоположение (x0, y0) сигнала яркости, указывающее левую верхнюю выборку текущего блока преобразования относительно левой верхней выборки текущей картинки, текущее местоположение (xC, yC) сканирования коэффициентов, двоичный логарифм log2TbWidth ширины блока преобразования, двоичный логарифм log2TbHeight высоты блока преобразования.
	Вывод этого процесса представляет собой параметр cRiceParam Райса.
	С учетом массива AbsLevel[x][y] для блока преобразования с индексом cIdx компонента и левым верхним местоположением (x0, y0) сигнала яркости, переменная locSumAbs извлекается так, как указано посредством следующего псевдокода:
	<i>ShiftRice=(BitDepth>10)? Floor(Log2 (4*(Bitdepth-10))):0</i>
	<i>OffsetRice=(BitDepth>10)? (ShiftRice>0)? (1<<(ShiftRice-1)):0:0</i>
	<i>locSumAbs=0</i>
	if(xC<(1<<log2TbWidth)-1) {
	locSumAbs+=AbsLevel[xC+1][yC]
	if(xC<(1<<log2TbWidth)-2)
	locSumAbs+=AbsLevel[xC+2][yC]
	if(yC<(1<<log2TbHeight)-1)
	locSumAbs+=AbsLevel[xC+1][yC+1]
	(1494)
	}
	if(yC<(1<<log2TbHeight)-1) {
	locSumAbs+=AbsLevel[xC][yC+1]
	if(yC<(1<<log2TbHeight)-2)
	locSumAbs+=AbsLevel[xC][yC+2]
	}
	locSumAbs=Clip3(0, 31, ((locSumAbs+OffsetRice)>>ShiftRice)-baseLevel*5)
	С учетом переменной locSumAbs, параметр cRiceParam Райса извлекается так, как указано в таблице 4. cRiceParam=cRiceParam+ShiftRice Когда baseLevel равен 0, переменная ZeroPos[n] извлекается следующим образом: ZeroPos[n]=(QState<2?1:2)<<cRiceParam

[00350] Согласно другому аспекту раскрытия, предлагается добавлять такое ограничение, что значение этих вышеприведенных средств кодирования помечается флагом, чтобы предоставлять такие же общие средства управления ограничениями, что и другие в общей информации ограничений.

[00351] Например, sps_ts_residual_coding_rice_present_in_sh_flag равный 1 указывает то, что sh_ts_residual_coding_rice_idx может присутствовать в синтаксических SH-

структурах, ссылающихся на SPS; `sps_ts_residual_coding_rice_present_in_sh_flag`, равный 0, указывает то, что `sh_ts_residual_coding_rice_idx` не присутствует в синтаксических SH-структурах, ссылающихся на SPS. Согласно раскрытию, предлагается добавлять синтаксический элемент, `gci_no_ts_residual_coding_rice_constraint_flag`, в синтаксис общей информации ограничений, чтобы предоставлять такие же общие средства управления ограничениями, что и другие флаги. Пример процесса декодирования на VVC-проекте проиллюстрирован ниже. Изменения VVC-проекта выделяются. Добавленные части выделяются с помощью курсивных шрифтов.

10	<code>general_constraint_info() {</code>	Дескриптор
	<code>...</code>	
	<code><i>gci_no_ts_residual_coding_rice_constraint_flag</i></code>	<code><i>u(1)</i></code>
	<code>...</code>	
	<code>}</code>	
15	<i>gci_no_ts_residual_coding_rice_constraint_flag, равный 1, указывает то, что sps_ts_residual_coding_rice_present_in_sh_flag должен быть равным 0; gci_no_ts_residual_coding_rice_constraint_flag, равный 0, не налагает такое ограничение.</i>	

[00352] В другом примере, `pps_ts_residual_coding_rice_flag`, равный 1, указывает то, что `pps_ts_residual_coding_rice_idx` может присутствовать в текущей картинке; `pps_ts_residual_coding_rice_flag`, равный 0, указывает то, что `pps_ts_residual_coding_rice_idx` не присутствует в текущей картинке. Согласно раскрытию, предлагается добавлять синтаксический элемент, `gci_no_ts_residual_coding_rice_constraint_flag`, в синтаксис общей информации ограничений, чтобы предоставлять такие же общие средства управления ограничениями, что и другие флаги. Пример процесса декодирования на VVC-проекте проиллюстрирован ниже. Изменения VVC-проекта выделяются. Добавленные части выделяются с помощью курсивных шрифтов.

25	<code>general_constraint_info() {</code>	Дескриптор
	<code>...</code>	
	<code><i>gci_no_ts_residual_coding_rice_constraint_flag</i></code>	<code><i>u(1)</i></code>
	<code>...</code>	
	<code>}</code>	
30	<i>gci_no_ts_residual_coding_rice_constraint_flag, равный 1, указывает то, что pps_ts_residual_coding_rice_flag должен быть равным 0; gci_no_ts_residual_coding_rice_constraint_flag, равный 0, не налагает такое ограничение.</i>	

[00353] В еще одном другом примере, `sps_rice_adaptation_enabled_flag`, равный 1, указывает то, что параметр Райса для преобразования в двоичную форму `abs_remaining[]` и `dec_abs_level` может извлекаться посредством формулы.

[00354] Формула может включать в себя: `RiceParam=RiceParam+shiftVal`, и `shiftVal=(localSumAbs<Tx[0])?Rx[0]: ((localSumAbs<Tx[1])?Rx[1]: ((localSumAbs<Tx[2])?Rx[2]: ((localSumAbs<Tx[3])?Rx[3]: Rx[4])))`,

где списки `Tx[]` и `Rx[]` указываются следующим образом: `Tx[]={32, 128, 512, 2 048}>>(1523)`, `Rx[]={0, 2, 4, 6, 8}`

[00355] Согласно раскрытию, предлагается добавлять синтаксический элемент, `gci_no_rice_adaptation_constraint_flag`, в синтаксис общей информации ограничений, чтобы предоставлять такие же общие средства управления ограничениями, что и другие флаги. Пример процесса декодирования на VVC-проекте проиллюстрирован ниже. Изменения VVC-проекта выделяются. Добавленные части выделяются с помощью курсивных шрифтов.

	<code>general_constraint_info() {</code>	Дескриптор
	<code>...</code>	
	<code><i>gci_no_rice_adaptation_constraint_flag</i></code>	<code><i>u(1)</i></code>

...	
}	
<i>gci_no_rice_adaptation_constraint_flag, равный 1, указывает то, что sps_rice_adaptation_enabled_flag должен быть равным 0; gci_no_rice_adaptation_constraint_flag, равный 0, не налагает такое ограничение.</i>	

[00356] Поскольку предложенная схема адаптации параметра Райса используется только для остаточного кодирования с пропуском преобразования (TSRC), предложенный способ может становиться эффективным только тогда, когда TSRC активируется. Соответственно, в одном или более вариантов осуществления раскрытия, предлагается добавлять одно ограничение по потоку битов, которое требует от значения *gci_no_rice_adaptation_constraint_flag* то, что оно должно составлять единицу, когда режим пропуска преобразования деактивируется из уровня общей информации ограничений, например, когда значение *gci_no_transform_skip_constraint_flag* задается равным единице.

[00357] В еще одном другом примере, *sps_range_extension_flag*, равный 1, указывает то, что синтаксическая структура *sps_range_extension()* присутствует в синтаксической SPS RBSP-структуре; *sps_range_extension_flag*, равный 0, указывает то, что эта синтаксическая структура не присутствует. Согласно раскрытию, предлагается добавлять синтаксический элемент, *gci_no_range_extension_constraint_flag*, в синтаксис общей информации ограничений, чтобы предоставлять такие же общие средства управления ограничениями, что и другие флаги. Пример процесса декодирования на VVC-проекте проиллюстрирован ниже. Изменения VVC-проекта выделяются. Добавленные части выделяются с помощью курсивных шрифтов.

<i>general_constraint_info()</i> {	Дескриптор
...	
<i>gci_no_range_extension_constraint_flag</i>	<i>u(1)</i>
...	
}	
<i>gci_no_range_extension_constraint_flag, равный 1, указывает то, что sps_range_extension_flag должен быть равным 0; gci_no_range_extension_constraint_flag, равный 0, не налагает такое ограничение.</i>	

[00358] Фиг. 19 показывает способ для кодирования видео согласно одному примеру настоящего раскрытия. Способ, например, может применяться к декодеру. На этапе 1902, декодер может принимать флаг расширения диапазона набора параметров последовательности (SPS), который указывает то, присутствует или нет синтаксическая структура, *sps_range_extension*, в синтаксических структурах первичной байтовой последовательности данных (RBSP) заголовков срезов (SH) на основе значения флага расширения SPS-диапазона.

[00359] На этапе 1904, в ответ на определение того, что значение флага расширения SPS-диапазона равно 1, декодер может определять то, что *sps_range_extension* присутствует в синтаксических SH RBSP-структурах.

[00360] На этапе 1906, в ответ на определение того, что значение флага расширения диапазона равно 0, декодер может определять то, что *sps_range_extension* не присутствует в синтаксических SH RBSP-структурах.

[00361] В еще одном другом примере, *sps_cabac_bypass_alignment_enabled_flag*, равный 1, указывает то, что значение *ivlCurrRange* может согласовываться до обходного декодирования синтаксических элементов *sb_coded_flag[]*, *abs_remainder[]*, *dec_abs_level[n]* и *coeff_sign_flag[]*; *sps_cabac_bypass_alignment_enabled_flag*, равный 0, указывает то, что значение *ivlCurrRange* не согласуется до обходного декодирования. Согласно раскрытию, предлагается добавлять синтаксический элемент, *gci_no_cabac_bypass_alignment_constraint_flag*, в синтаксис общей информации ограничений, чтобы предоставлять

такие же общие средства управления ограничениями, что и другие флаги. Пример процесса декодирования на VVC-проекте проиллюстрирован ниже. Изменения VVC-проекта выделяются. Добавленные части выделяются с помощью курсивных шрифтов.

5	<code>general_constraint_info() {</code>	Дескриптор
	<code>...</code>	
	<code><i>gci_no_cabac_bypass_alignment_constraint_flag</i></code>	<i>u(1)</i>
	<code>...</code>	
	<code>}</code>	
10	<i>gci_no_cabac_bypass_alignment_constraint_flag, равный 1, указывает то, что sps_cabac_bypass_alignment_enabled_flag должен быть равным 0; gci_no_cabac_bypass_alignment_constraint_flag, равный 0, не налагает такое ограничение.</i>	

[00362] Фиг. 20 показывает способ для кодирования видео согласно одному примеру настоящего раскрытия. Способ, например, может применяться к декодеру. На этапе 2002, декодер может принимать флаг активации согласования с набором параметров последовательности (SPS), который указывает то, согласется или нет индекс, `ivlCurrRange`, до обходного декодирования синтаксических элементов `sb_coded_flag`, `abs_remainder`, `dec_abs_level` и `coeff_sign_flag` на основе значения активации SPS-согласования.

[00363] На этапе 2004, в ответ на определение того, что значение флага активации SPS-согласования равно 1, декодер может определять то, что `ivlCurrRange` согласуется до обходного декодирования.

[00364] На этапе 2006, в ответ на определение того, что значение флага активации SPS-согласования равно 0, декодер может определять то, что `ivlCurrRange` не согласуется до обходного декодирования.

[00365] В еще одном другом примере, `extended_precision_processing_flag`, равный 1, указывает то, что расширенный динамический диапазон может использоваться для коэффициентов преобразования и обработки преобразования; `extended_precision_processing_flag`, равный 0, указывает то, что расширенный динамический диапазон не используется. Согласно раскрытию, предлагается добавлять синтаксический элемент, `gci_no_extended_precision_processing_constraint_flag`, в синтаксис общей информации ограничений, чтобы предоставлять такие же общие средства управления ограничениями, что и другие флаги. Пример процесса декодирования на VVC-проекте проиллюстрирован ниже. Изменения VVC-проекта выделяются. Добавленные части выделяются с помощью курсивных шрифтов.

35	<code>general_constraint_info() {</code>	Дескриптор
	<code>...</code>	
	<code><i>gci_no_extended_precision_processing_constraint_flag</i></code>	<i>u(1)</i>
	<code>...</code>	
	<code>}</code>	
40	<i>gci_no_extended_precision_processing_constraint_flag, равный 1, указывает то, что extended_precision_processing_flag должен быть равным 0; gci_no_extended_precision_processing_constraint_flag, равный 0, не налагает такое ограничение.</i>	

[00366] Фиг. 21 показывает способ для кодирования видео согласно одному примеру настоящего раскрытия. Способ, например, может применяться к декодеру. На этапе 2102, декодер может принимать флаг обработки с расширенной точностью, который указывает то, принимается или нет расширенный динамический диапазон для коэффициентов преобразования и во время обработки преобразования на основе значения флага обработки с расширенной точностью.

[00367] На этапе 2104, в ответ на определение того, что значение флага обработки с расширенной точностью равно 1, декодер может определять то, что расширенный

динамический диапазон принимается для коэффициентов преобразования и во время обработки преобразования.

[00368] На этапе 2106, в ответ на определение того, что значение флага обработки с расширенной точностью в 0, декодер может определять то, что расширенный динамический диапазон не принимается для коэффициентов преобразования или во время обработки преобразования.

[00369] В еще одном другом примере, *persistent_rice_adaptation_enabled_flag*, равный 1, указывает то, что извлечение параметра Райса для преобразования в двоичную форму *abs_remaining[]* и *dec_abs_level* может инициализироваться в начале каждого субблока с использованием зависимой от режима статистики, накопленной из предыдущих субблоков; *persistent_rice_adaptation_enabled_flag*, равный 0, указывает то, что состояние предыдущих субблоков не используется при извлечении параметра Райса. Согласно раскрытию, предлагается добавлять синтаксический элемент, *gci_no_persistent_rice_adaptation_constraint_flag*, в синтаксис общей информации ограничений, чтобы предоставлять такие же общие средства управления ограничениями, что и другие флаги. Пример процесса декодирования на VVC-проекте проиллюстрирован ниже. Изменения VVC-проекта выделяются. Добавленные части выделяются с помощью курсивных шрифтов.

general_constraint_info() {	Дескриптор
...	
<i>gci_no_persistent_rice_adaptation_constraint_flag</i>	<i>u(1)</i>
...	
}	
<i>gci_no_persistent_rice_adaptation_constraint_flag</i> , равный 1, указывает то, что <i>persistent_rice_adaptation_enabled_flag</i> должен быть равным 0; <i>gci_no_persistent_rice_adaptation_constraint_flag</i> , равный 0, не налагает такое ограничение.	

[00370] Фиг. 22 показывает способ для кодирования видео согласно одному примеру настоящего раскрытия. Способ, например, может применяться к декодеру. На этапе 2202, декодер может принимать флаг активации постоянной адаптации Райса, который указывает то, инициализируется или нет извлечение параметра Райса для преобразования в двоичную форму *abs_remaining* и *dec_abs_level* в начале каждого субблока, принимая зависимую от режима статистику, накапливаемую из предыдущих субблоков, на основе значения флага активации постоянной адаптации Райса.

[00371] На этапе 2204, в ответ на определение того, что значение флага активации постоянной адаптации Райса равно 1, декодер может определять то, что извлечение параметра Райса для преобразования в двоичную форму инициализируется в начале каждого субблока, принимая зависимую от режима статистику, накапливаемую из предыдущих субблоков.

[00372] На этапе 2206, в ответ на определение того, что значение флага активации постоянной адаптации Райса равно 0, декодер может определять то, что состояние предыдущих субблоков не принимается при извлечении параметра Райса.

[00373] Вышеописанные способы могут реализовываться с использованием оборудования, которое включает в себя одну или более схем, которые включают в себя специализированные интегральные схемы (ASIC), процессоры цифровых сигналов (DSP), устройства обработки цифровых сигналов (DSPD), программируемые логические устройства (PLD), программируемые пользователем вентильные матрицы (FPGA), контроллеры, микроконтроллеры, микропроцессоры или другие электронные компоненты. Оборудование может использовать схемы в комбинации с другими аппаратными или программными компонентами для осуществления вышеописанных

способов. Каждый модуль, submodule, блок или subblock, раскрытый выше, может реализовываться, по меньшей мере, частично с использованием одной или более схем.

[00374] *Решение по выбору параметра Райса*

[00375] На стороне кодера, TSRC-кодирование может требовать нескольких проходов кодирования, чтобы извлекать наилучший параметр Райса. Это многопроходное кодирование может не быть подходящим для практических проектных решений по аппаратному кодированию. Чтобы разрешать эту проблему, также предлагается способ TSRC-кодирования с низкой задержкой. Согласно тридцатому аспекту раскрытия, предлагается извлекать параметр Райса согласно некоторой кодированной информации текущего среза, например, параметру квантования и/или битовой глубине кодирования, ассоциированной со срезом/картинкой/последовательностью, и/или согласно хэш-отношению, ассоциированному с уровнем среза/картинки/последовательности. Различные способы могут использоваться для того, чтобы извлекать параметр Райса, при этом некоторые примерные способы перечисляются следующим образом. Следует отметить, что следующие способы могут применяться независимо или объединенно.

[00376] 1. Параметр Райса, упомянутый в вышеприведенных вариантах осуществления, дополнительно может зависеть от разрешения видео, включающего в себя как временное разрешение (например, частоту кадров), так и пространственное разрешение (например, ширину и высоту картинки) видео.

[00377] 2. Параметр Райса может варьироваться на уровне последовательности, на уровне картинки, на уровне среза и/или в любой предварительно заданной области. В одном конкретном примере, различные значения Райса используются для картинок с различными идентификаторами временных слоев (которые связаны с `nuh_temporal_id_plus1`, указываемым в VVC-спецификации). Альтернативно, параметр Райса может включать в себя значение, определенное на основе QR-значений, используемых на уровне последовательности, на уровне картинки, на уровне среза и/или в любой предварительно заданной области. Например, `rice parameter=Clip3(1, 8, (TH-QR)/6)`, где TH является предварительно заданным пороговым значением (например, 18, 19).

[00378] 3. Параметр Райса может задаваться в качестве значения по умолчанию, например, 1, согласно изменению кодированной информации между текущим срезом и предыдущим срезом. В одном конкретном примере, значение Райса по умолчанию используется для картинок, когда его идентификатор временного слоя изменяется по сравнению с предыдущей картинкой. Альтернативно, значение Райса по умолчанию используется для картинок, когда ΔQ больше TH, где ΔQ вычисляется как $\text{abs}(QR_{\text{current}} - QR_{\text{previous}})$, и TH является предварительно заданным пороговым значением. Параметр Райса (например, 0, 5). Например, `Rice parameter=1`, когда хэш-отношение из режима внутриблочного копирования в текущем срезе больше TH, где TH является предварительно заданным пороговым значением, например, $\text{Max}(41 * (\text{число STU}), 4200)$.

[00379] 4. Параметр Райса для каждого среза основан на значениях `abs_remainder`, которые кодируются в его предшествующем срезе согласно порядку кодирования. В одном конкретном примере, после того, как один срез кодируется, вычисляется число элементов разрешения для преобразования в двоичную форму `abs_remainder` с использованием различных параметров Райса, которые затем используются для того, чтобы определять параметр Райса следующего среза. Например, параметр Райса, который достигает минимального номера элемента разрешения в предшествующем срезе, должен выбираться для текущего среза. В качестве другого примера, если текущий срез и его предшествующий срез используют один одинаковый QR, то параметр Райса,

который достигает минимального номера элемента разрешения в предшествующем срезе, должен выбираться для текущего среза; иначе, число элементов разрешения, сформированных с использованием параметра Райса по умолчанию (т.е. 1) в предшествующем срезе, масштабируется посредством ТН перед сравнением с другими параметрами Райса, и параметр Райса, который приводит к минимальному числу элементов разрешения, должен выбираться для текущего среза, где ТН является предварительно заданным пороговым значением, например, 0,9.

[00380] 5. Параметр Райса для каждого среза основан на значениях `abs_remainder`, которые кодируются в его предшествующем срезе согласно порядку кодирования, и параметр Райса может регулироваться согласно изменению закодированной информации между текущим срезом и предыдущим срезом. В одном конкретном примере, параметр Райса, который достигает минимального номера элемента разрешения в предшествующем срезе, должен выбираться для текущего среза. Кроме того, значение Райса может регулироваться, когда ΔQ больше ТН, где ΔQ вычисляется как `abs(QPcurrent - QPprevious)`, и ТН является предварительно заданным пороговым значением. Параметр Райса (например, 0, 5). Регулирование может представлять собой суммирование предварительно заданного смещения (например, +1, -1) или масштабирование посредством предварительно заданного значения.

[00381] Фиг. 26 показывает блок-схему последовательности операций способа остаточного кодирования с пропуском преобразования (TSRC) с низкой задержкой согласно одному примеру настоящего раскрытия. Способ, например, может применяться к кодеру. На этапе 2602, кодер может извлекать параметр Райса на основе закодированной информации текущего среза видео. Информация может включать в себя одно или более из следующего: параметр квантования или битовая глубина кодирования, ассоциированная со срезом, картинкой или последовательностью видео; либо хэш-отношение, ассоциированное со срезом, картинкой или последовательностью видео.

[00382] Следует отметить, что вышеуказанные способы кодера могут применяться на стороне декодера. В одном конкретном примере, параметр Райса не должен обязательно сигнализироваться в декодер, и кодер/декодер используют такой же способ для того, чтобы извлекать параметр Райса.

[00383] Фиг. 18 показывает вычислительное окружение 1810, соединенное с пользовательским интерфейсом 1860. Вычислительное окружение 1810 может представлять собой часть сервера обработки данных. Вычислительное окружение 1810 включает в себя процессор 1820, запоминающее устройство 1840 и интерфейс 1850 ввода-вывода.

[00384] Процессор 1820 типично полностью управляет операциями вычислительного окружения 1810, к примеру, операциями, ассоциированными с отображением, получением данных, обменом данными и обработкой изображений. Процессор 1820 может включать в себя один или более процессоров, которые выполняют инструкции для того, чтобы выполнять все или некоторые этапы в вышеописанных способах. Кроме того, процессор 1820 может включать в себя один или более модулей, которые упрощают взаимодействие между процессором 1820 и другими компонентами. Процессор может представлять собой центральный процессор (CPU), микропроцессор, одномикросхемную машину, GPU и т.п.

[00385] Запоминающее устройство 1840 выполнено с возможностью сохранять различные типы данных, чтобы поддерживать работу вычислительного окружения 1810. Запоминающее устройство 1840 может включать в себя предварительно определенное программное обеспечение 1842. Примеры таких данных включают в себя

инструкции для любых приложений или способов, работающих в вычислительном окружении 1810, наборы видеоданных, данные изображений и т.д. Запоминающее устройство 1840 может реализовываться посредством использования любого типа энергозависимых или энергонезависимых запоминающих устройств либо их комбинации, к примеру, как статическое оперативное запоминающее устройство (SRAM), электрически стираемое программируемое постоянное запоминающее устройство (EEPROM), стираемое программируемое постоянное запоминающее устройство (EPROM), программируемое постоянное запоминающее устройство (PROM), постоянное запоминающее устройство (ROM), магнитное запоминающее устройство, флэш-память, магнитный или оптический диск.

[00386] Интерфейс 1850 ввода-вывода предоставляет интерфейс между процессором 1820 и периферийными интерфейсными модулями, такими как клавиатура, колесико с кнопками, кнопки и т.п. Кнопки могут включать в себя, но не только, кнопку перехода на начальный экран, кнопку запуска сканирования и кнопку остановки сканирования.

Интерфейс 1850 ввода-вывода может соединяться с кодером и декодером.

[00387] В некоторых вариантах осуществления, также предоставляется энергонезависимый машиночитаемый носитель хранения данных, содержащий множество программ, к примеру, содержащихся в запоминающем устройстве 1840, выполняемых посредством процессора 1820 в вычислительном окружении 1810, для осуществления вышеописанных способов. Например, энергонезависимый машиночитаемый носитель хранения данных может представлять собой ROM, RAM, CD-ROM, магнитную ленту, гибкий диск и оптическое устройство хранения данных и т.п.

[00388] Энергонезависимый машиночитаемый носитель хранения данных имеет сохраненными множество программ для выполнения посредством вычислительного устройства, имеющего один или более процессоров, при этом множество программ, при выполнении посредством одного или более процессоров, инструктируют вычислительному устройству осуществлять вышеописанный способ для прогнозирования движения.

[00389] В некоторых вариантах осуществления, вычислительное окружение 1810 может реализовываться с помощью одной или более специализированных интегральных схем (ASIC), процессоров цифровых сигналов (DSP), устройств обработки цифровых сигналов (DSPD), программируемых логических устройств (PLD), программируемых пользователем вентильных матриц (FPGA), графических процессоров (GPU), контроллеров, микроконтроллеров, микропроцессоров или других электронных компонентов, для осуществления вышеописанных способов.

[00390] Фиг. 23 является блок-схемой, иллюстрирующей примерную систему 10 для кодирования и декодирования видеоблоков параллельно в соответствии с некоторыми реализациями настоящего раскрытия. Как показано на фиг. 23, система 10 включает в себя исходное устройство 12, которое формирует и кодирует видеоданные, которые должны декодироваться в дальнейшем посредством целевого устройства 14. Исходное устройство 12 и целевое устройство 14 могут содержать любые из широкого спектра электронных устройств, включающих в себя настольные или переносные компьютеры, планшетные компьютеры, смартфоны, абонентские приставки, цифровые телевизоры, камеры, устройства отображения, цифровые мультимедийные проигрыватели, консоли для видеоигр, устройство потоковой передачи видео и т.п. В некоторых реализациях, исходное устройство 12 и целевое устройство 14 оснащаются возможностями беспроводной связи.

[00391] В некоторых реализациях, целевое устройство 14 может принимать кодированные видеоданные, которые должны декодироваться, через линию 16 связи. Линия 16 связи может содержать любой тип среды связи или устройства, допускающего перемещение кодированных видеоданных из исходного устройства 12 в целевое устройство 14. В одном примере, линия 16 связи может содержать среду связи, чтобы обеспечивать возможность исходному устройству 12 передавать кодированные видеоданные непосредственно в целевое устройство 14 в реальном времени.

Кодированные видеоданные могут модулироваться согласно стандарту связи, такому как протокол беспроводной связи, и передаваться в целевое устройство 14. Среда связи может содержать любую беспроводную или проводную среду связи, такую как радиочастотный (RF) спектр или одна или более физических линий передачи. Среда связи может формировать часть сети с коммутацией пакетов, такой как локальная вычислительная сеть, глобальная вычислительная сеть либо глобальная сеть, такая как Интернет. Среда связи может включать в себя маршрутизаторы, коммутаторы, базовые станции или любое другое оборудование, которое может быть полезным для того, чтобы упрощать передачу из исходного устройства 12 в целевое устройство 14.

[00392] В некоторых других реализациях, кодированные видеоданные могут передаваться из интерфейса 22 вывода в устройство 32 хранения данных. Затем, к кодированным видеоданным в устройстве 32 хранения данных может осуществляться доступ посредством целевого устройства 14 через интерфейс 28 ввода. Устройство 32 хранения данных может включать в себя любые из множества распределенных или локально доступных носителей хранения данных, к примеру, жесткий диск, Blu-Ray-диски, универсальные цифровые диски (DVD), постоянные запоминающие устройства на компакт-дисках (CD-ROM), флэш-память, энергозависимое или энергонезависимое запоминающее устройство либо любые другие подходящие цифровые носители хранения данных для сохранения кодированных видеоданных. В дополнительном примере, устройство 32 хранения данных может соответствовать файловому серверу или другому промежуточному устройству хранения данных, которое может запоминать кодированные видеоданные, сформированные посредством исходного устройства 12. Целевое устройство 14 может осуществлять доступ к сохраненным видеоданным из устройства 32 хранения данных через потоковую передачу или загрузку. Файловый сервер может представлять собой любой тип компьютера, допускающего сохранение кодированных видеоданных и передачу кодированных видеоданных в целевое устройство 14. Примерные файловые серверы включают в себя веб-сервер (например, для веб-узла), сервер по протоколу передачи файлов (FTP), устройства по протоколу системы хранения данных с подключением по сети (NAS) или локальный накопитель на дисках. Целевое устройство 14 может осуществлять доступ к кодированным видеоданным через любое стандартное соединение для передачи данных, включающее в себя беспроводной канал (например, соединение по стандарту высококачественной беспроводной связи (Wi-Fi)), проводное соединение (например, цифровую абонентскую линию (DSL), кабельный модем и т.д.) либо комбинацию означенного, которая является подходящей для осуществления доступа к кодированным видеоданным, сохраненным на файловом сервере. Передача кодированных видеоданных из устройства 32 хранения данных может представлять собой потоковую передачу, передачу на основе загрузки либо комбинацию вышеозначенного.

[00393] Как показано на фиг. 23, исходное устройство 12 включает в себя видеоисточник 18, видеокодер 20 и интерфейс 22 вывода. Видеоисточник 18 может включать в себя источник, к примеру, устройство видеозахвата, например, видеокамеру,

видеоархив, содержащий ранее захваченное видео, интерфейс прямых видеотрансляций для того, чтобы принимать видео от поставщика видеоконтента, и/или компьютерную графическую систему для формирования компьютерных графических данных в качестве исходного видео либо комбинацию таких источников. В качестве одного примера, если видеоисточник 18 представляет собой видеокамеру системы наблюдения для обеспечения безопасности, исходное устройство 12 и целевое устройство 14 могут формировать камерофоны или видеотелефоны. Однако, реализации, описанные в настоящей заявке, могут быть применимыми к кодированию видео в целом и могут применяться к беспроводным и/или проводным вариантам применения.

[00394] Захваченное, предварительно захваченное или машиногенерируемое видео может кодироваться посредством видеокодера 20. Кодированные видеоданные могут передаваться непосредственно в целевое устройство 14 через интерфейс 22 вывода исходного устройства 12. Кодированные видеоданные также (или альтернативно) могут сохраняться на устройстве 32 хранения данных для последующего доступа посредством целевого устройства 14 или других устройств, для декодирования и/или воспроизведения. Интерфейс 22 вывода дополнительно может включать в себя модем и/или передатчик.

[00395] Целевое устройство 14 включает в себя интерфейс 28 ввода, видеodeкодер 30 и устройство 34 отображения. Интерфейс 28 ввода может включать в себя приемник и/или модем и принимать кодированные видеоданные по линии 16 связи. Кодированные видеоданные, передаваемые по линии 16 связи или предоставленные на устройстве 32 хранения данных, могут включать в себя множество синтаксических элементов, сформированных посредством видеокодера 20 для использования посредством видеodeкодера 30 при декодировании видеоданных. Такие синтаксические элементы могут быть включены в кодированные видеоданные, передаваемые по среде связи, сохраненные на носителе хранения данных или сохраненные на файловом сервере.

[00396] В некоторых реализациях, целевое устройство 14 может включать в себя устройство 34 отображения, которое может представлять собой интегрированное устройство отображения и внешнее устройство отображения, которое выполнено с возможностью обмениваться данными с целевым устройством 14. Устройство 34 отображения отображает декодированные видеоданные пользователю и может содержать любое из множества устройств отображения, к примеру, жидкокристаллический дисплей (ЖК-дисплей), плазменный дисплей, дисплей на органических светодиодах (OLED) или другой тип устройства отображения.

[00397] Видеокодер 20 и видеodeкодер 30 могут работать согласно собственным или отраслевым стандартам, таким как VVC, HEVC, MPEG-4 часть 10, AVC или расширения таких стандартов. Следует понимать, что настоящая заявка не ограничена конкретным стандартом кодирования/декодирования видео и может быть применимой к другим стандартам кодирования/декодирования видео. В общем, предполагается, что видеокодер 20 исходного устройства 12 может быть выполнен с возможностью кодировать видеоданные согласно любому из этих текущих или будущих стандартов. Аналогично, в общем, также предполагается, что видеodeкодер 30 целевого устройства 14 может быть выполнен с возможностью декодировать видеоданные согласно любому из этих текущих или будущих стандартов.

[00398] Видеокодер 20 и видеodeкодер 30 могут реализовываться как любая из множества надлежащих схем кодера и/или декодера, к примеру, как один или более микропроцессоров, процессоров цифровых сигналов (DSP), специализированных интегральных схем (ASIC), программируемых пользователем вентильных матриц (FPGA), как дискретная логика, программное обеспечение, аппаратные средства,

микропрограммное обеспечение либо как любые комбинации вышеозначенного. При частичной реализации в программном обеспечении, электронное устройство может сохранять инструкции для программного обеспечения на подходящем, энергонезависимом машиночитаемом носителе и выполнять инструкции в аппаратных средствах с использованием одного или более процессоров, чтобы выполнять операции кодирования/декодирования видео, раскрытые в настоящем раскрытии. Каждый из видеокодера 20 и декодера 30 может быть включен в один или более кодеров или декодеров, любой из которых может интегрироваться как часть комбинированного кодера/декодера (кодека) в соответствующем устройстве.

[00399] Фиг. 24 является блок-схемой, иллюстрирующей примерный видеокодер 20, в соответствии с некоторыми реализациями, описанными в настоящей заявке. Видеокодер 20 может выполнять внутреннее и взаимное прогнозирующее кодирование видеоблоков в видеокдрах. Внутреннее прогнозирующее кодирование основано на пространственном прогнозировании для того, чтобы уменьшать или удалять пространственную избыточность в видеоданных в данном видеокдре или картинке. Взаимное прогнозирующее кодирование основано на временном прогнозировании для того, чтобы уменьшать или удалять временную избыточность в видеоданных в смежных видеокдрах или картинках видеопоследовательности. Следует отметить, что термин "кадр" может использоваться в качестве синонимов для термина "изображение" или "картинка" в области техники кодирования видео.

[00400] Как показано на фиг. 24, видеокодер 20 включает в себя запоминающее устройство 40 видеоданных, модуль 41 обработки прогнозирования, буфер 64 декодированных картинок (DPB), сумматор 50, модуль 52 обработки преобразования, модуль 54 квантования и модуль 56 энтропийного кодирования. Модуль 41 обработки прогнозирования дополнительно включает в себя модуль 42 оценки движения, модуль 44 компенсации движения, модуль 45 сегментации, модуль 46 обработки внутреннего прогнозирования и модуль 48 внутриблочного копирования (BC). В некоторых реализациях, видеокодер 20 также включает в себя модуль 58 обратного квантования, модуль 60 обработки обратного преобразования и сумматор 62 для восстановления видеоблоков. Внутриконтурный фильтр 63, такой как фильтр удаления блочности, может позиционироваться между сумматором 62 и DPB 64, чтобы фильтровать границы блоков, чтобы удалять артефакты блочности из восстановленного видео. Другой внутриконтурный фильтр, такой как фильтр на основе дискретизированного адаптивного смещения (SAO) и/или адаптивный внутриконтурный фильтр (ALF), также может использоваться в дополнение к фильтру удаления блочности для того, чтобы фильтровать вывод сумматора 62. В некоторых примерах, внутриконтурные фильтры могут опускаться, и декодированный видеоблок может непосредственно предоставляться посредством сумматора 62 в DPB 64. Видеокодер 20 может принимать форму фиксированного или программируемого аппаратного модуля либо может разделяться между одним или более проиллюстрированных фиксированных или программируемых аппаратных модулей.

[00401] Запоминающее устройство 40 видеоданных может сохранять видеоданные, которые должны кодироваться посредством компонентов видеокодера 20. Видеоданные в запоминающем устройстве 40 видеоданных могут получаться, например, из видеисточника 18, как показано на фиг. 23. DPB 64 представляет собой буфер, который сохраняет опорные видеоданные (например, опорные кадры или картинки) для использования при кодировании видеоданных посредством видеокодера 20 (например, в режимах внутреннего или взаимного прогнозирующего кодирования). Запоминающее

устройство 40 видеоданных и DPB 64 могут формироваться посредством любых из множества запоминающих устройств. В различных примерах, запоминающее устройство 40 видеоданных может быть внутримикросхемным с другими компонентами видеокодера 20 или внешним относительно этих компонентов.

5 [00402] Как показано на фиг. 24, после приема видеоданных, модуль 45 сегментации в модуле 41 обработки прогнозирования сегментирует видеоданные на видеоблоки. Эта сегментация также может включать в себя сегментацию видеокадра на срезы, плитки (например, наборы видеоблоков), или другие большие единицы кодирования (CU) согласно предварительно заданным структурам разбиения, таким как структура 10 в виде дерева квадрантов (QT), ассоциированная с видеоданными. Видеокадр представляет собой либо может рассматриваться в качестве двумерного массива или матрицы выборок с выборочными значениями. Выборка в массиве также может называться "пикселом" или "пелом". Число выборок в горизонтальном и вертикальном направлениях (или осях) массива или картинки задает размер и/или разрешение 15 видеокадра. Видеокадр может разделяться на несколько видеоблоков, например, посредством использования QT-сегментации. Видеоблок снова представляет собой или может рассматриваться в качестве двумерного массива или матрицы выборок с выборочными значениями, хотя и меньшей размерности, чем видеокадр. Число выборок в горизонтальном и вертикальном направлениях (или осях) видеоблока задает размер 20 видеоблока. Видеоблок дополнительно может сегментироваться на один или более или блочных сегментов или субблоков (которые снова могут формировать блоки), например, посредством итеративного использования QT-сегментации, сегментации на двоичное дерево (BT) или сегментации на троичное дерево (TT) либо любой комбинации вышеозначенного. Следует отметить, что термин "блок" или "видеоблок" при 25 использовании в данном документе может представлять собой часть, в частности, прямоугольную (квадратную или неквадратную) часть кадра или картинки. Со ссылкой, например, на HEVC и VVC, блок или видеоблок может представлять собой либо соответствовать единице дерева кодирования (CTU), CU, единице прогнозирования (PU) или единице преобразования (TU) и/или может представлять собой или 30 соответствовать надлежащему блоку, например, блоку дерева кодирования (CTB), блоку кодирования (CB), блоку прогнозирования (PB) или блоку преобразования (TB) и/или субблоку.

[00403] Модуль 41 обработки прогнозирования может выбирать один из множества возможных режимов прогнозирующего кодирования, к примеру, один из множества 35 режимов внутреннего прогнозирующего кодирования или один из множества режимов взаимного прогнозирующего кодирования, для текущего видеоблока на основе результатов ошибки (например, скорости кодирования и уровня искажения). Модуль 41 обработки прогнозирования может предоставлять результирующий кодированный блок внутреннего или взаимного прогнозирования в сумматор 50, чтобы формировать 40 остаточный блок, и в сумматор 62, чтобы восстанавливать кодированный блок для использования в качестве части опорного кадра в дальнейшем. Модуль 41 выбора режима также предоставляет синтаксические элементы, такие как векторы движения, индикаторы внутреннего режима, информация сегментов и другая такая синтаксическая информация, в модуль 56 энтропийного кодирования.

45 [00404] Чтобы выбирать соответствующий режим внутреннего прогнозирующего кодирования для текущего видеоблока, модуль 46 обработки внутреннего прогнозирования в модуле 41 обработки прогнозирования может выполнять внутреннее прогнозирующее кодирование текущего видеоблока относительно одного или более

соседних блоков в одном и том же кадре с текущим блоком, который должен кодироваться, чтобы предоставлять пространственное прогнозирование. Модуль 42 оценки движения и модуль 44 компенсации движения в модуле 41 обработки прогнозирования выполняют взаимное прогнозирующее кодирование текущего видеоблока относительно одного или более прогнозирующих блоков в одном или более опорных картинок для того, чтобы предоставлять временное сжатие. Видеокодер 20 может выполнять несколько проходов кодирования, например, для того чтобы выбирать надлежащий режим кодирования для каждого блока видеоданных.

[00405] В некоторых реализациях, модуль 42 оценки движения определяет режим взаимного прогнозирования для текущего видеокadra посредством формирования вектора движения, который указывает смещение видеоблока в текущем видеокadre относительно прогнозирующего блока в опорном видеокadre, согласно предварительно определенному шаблону в последовательности видеокadров. Оценка движения, выполняемая посредством модуля 42 оценки движения, представляет собой процесс формирования векторов движения, которые оценивают движение для видеоблоков. Вектор движения, например, может указывать смещение видеоблока в текущем видеокadre или картинке относительно прогнозирующего блока в опорном кадре относительно текущего блока, кодируемого в текущем кадре. Предварительно определенный шаблон может обозначать видеокadры в последовательности в качестве Р-кадров или В-кадров. Модуль 48 внутреннего ВС может определять векторы, например, блочные векторы, для внутреннего ВС-кодирования способом, аналогичным определению векторов движения посредством модуля 42 оценки движения для взаимного прогнозирования либо может использовать модуль 42 оценки движения для того, чтобы определять блочный вектор.

[00406] Прогнозирующий блок для видеоблока может представлять собой или может соответствовать блоку или опорному блоку опорного кадра, который считается тесно совпадающим с видеоблоком, который должен кодироваться, с точки зрения пиксельной разности, которая может определяться посредством суммы абсолютных разностей (SAD), суммы квадратов разностей (SSD) или других разностных показателей. В некоторых реализациях, видеокодер 20 может вычислять значения для субцелочисленнопиксельных позиций опорных кадров, сохраненных в DPB 64. Например, видеокодер 20 может интерполировать значения позиций в одну четверть пиксела, позиций в одну восьмую пиксела или других дробнопиксельных позиций опорного кадра. Следовательно, модуль 42 оценки движения может выполнять поиск движения относительно полнопиксельных позиций и дробнопиксельных позиций и выводить вектор движения с дробнопиксельной точностью.

[00407] Модуль 42 оценки движения вычисляет вектор движения для видеоблока во взаимном прогнозном кодированном кадре посредством сравнения позиции видеоблока с позицией прогнозирующего блока опорного кадра, выбранного из первого списка опорных кадров (список 0) или второго списка опорных кадров (список 1), каждый из которых идентифицирует один или более опорных кадров, сохраненных в DPB 64. Модуль 42 оценки движения отправляет вычисленный вектор движения в модуль 44 компенсации движения и затем в модуль 56 энтропийного кодирования.

[00408] Компенсация движения, выполняемая посредством модуля 44 компенсации движения, может заключать в себе осуществление выборки или формирование прогнозирующего блока на основе вектора движения, определенного посредством модуля 42 оценки движения. При приеме вектора движения для текущего видеоблока, модуль 44 компенсации движения может находить прогнозирующий блок, на который

указывает вектор движения в одном из списков опорных кадров, извлекать прогнозирующий блок из DPB 64 и перенаправлять в прогнозирующий блок в сумматор 50. Сумматор 50 затем формирует остаточный видеоблок пиксельных разностных значений посредством вычитания пиксельных значений прогнозирующего блока, предоставленного посредством модуля 44 компенсации движения, из пиксельных значений текущего кодируемого видеоблока. Пиксельные разностные значения, формирующие остаточный видеоблок, могут включать в себя разностные компоненты сигналов яркости или сигналов цветности либо и то, и другое. Модуль 44 компенсации движения также может формировать синтаксические элементы, ассоциированные с видеоблоками видеокadra, для использования посредством видеodeкодера 30 при декодировании видеоблоков видеокadra. Синтаксические элементы могут включать в себя, например, синтаксические элементы, задающие вектор движения, используемый для того, чтобы идентифицировать прогнозирующий блок, любые флаги, указывающие режим прогнозирования, либо любую другую синтаксическую информацию, описанную в данном документе. Следует отметить, что модуль 42 оценки движения и модуль 44 компенсации движения могут иметь высокую степень интеграции, но проиллюстрированы отдельно в концептуальных целях.

[00409] В некоторых реализациях, модуль 48 внутреннего ВС может формировать векторы и осуществлять выборку прогнозирующих блоков способом, аналогичным способу, описанному выше в связи с модулем 42 оценки движения и модулем 44 компенсации движения, но при том, что прогнозирующие блоки находятся в одном и том же кадре с кодируемым текущим блоком, и при том, что векторы называются "блочными векторами", в отличие от векторов движения. В частности, модуль 48 внутреннего прогнозирования может определять режим внутреннего прогнозирования для использования для того, чтобы кодировать текущий блок. В некоторых примерах, модуль 48 внутреннего ВС может кодировать текущий блок с использованием различных режимов внутреннего прогнозирования, например, во время отдельных проходов кодирования и тестировать их производительность через анализ искажения в зависимости от скорости передачи. Затем, модуль 48 внутреннего ВС может выбирать, из различных протестированных режимов внутреннего прогнозирования, соответствующий режим внутреннего прогнозирования, который следует использовать, и формировать индикатор внутреннего режима, соответственно. Например, модуль 48 внутреннего ВС может вычислять значения искажения в зависимости от скорости передачи с использованием анализа искажения в зависимости от скорости передачи для различных протестированных режимов внутреннего прогнозирования и выбирать режим внутреннего прогнозирования, имеющий наилучшие характеристики искажения в зависимости от скорости передачи из протестированных режимов, в качестве соответствующего режима внутреннего прогнозирования, который следует использовать. Анализ искажения в зависимости от скорости передачи, в общем, определяет величину искажения (или ошибки) между кодированным блоком и исходным некодированным блоком, который кодирован для того, чтобы формировать кодированный блок, а также скорость передачи битов (т.е. число битов), используемую для того, чтобы формировать кодированный блок. Модуль 48 внутреннего ВС может вычислять отношения из искажений и скоростей для различных кодированных блоков, чтобы определять то, какой режим внутреннего прогнозирования демонстрирует наилучшее значение искажения в зависимости от скорости передачи для блока.

[00410] В других примерах, модуль 48 внутреннего ВС может использовать модуль 42 оценки движения и модуль 44 компенсации движения, полностью или частично,

чтобы выполнять такие функции для внутреннего ВС-прогнозирования согласно реализациям, описанным в данном документе. В любом случае, для внутриблочного копирования, прогнозирующий блок может представлять собой блок, который считается тесно совпадающим с блоком, который должен кодироваться, с точки зрения пиксельной разности, которая может определяться посредством SAD, SSD или других разностных показателей, и идентификация прогнозирующего блока может включать в себя вычисление значений для субцелочисленнопиксельных позиций.

[00411] Независимо от того, исходит прогнозирующий блок из одного и того же кадра согласно внутреннему прогнозированию или из другого кадра согласно взаимному прогнозированию, видеокодер 20 может формировать остаточный видеоблок посредством вычитания пиксельных значений прогнозирующего блока из пиксельных значений текущего кодируемого видеоблока, с формированием пиксельных разностных значений. Пиксельные разностные значения, формирующие остаточный видеоблок, могут включать в себя разности компонентов сигналов яркости и сигналов цветности.

[00412] Модуль 46 обработки внутреннего прогнозирования может внутренне прогнозировать текущий видеоблок, в качестве альтернативы взаимному прогнозированию, выполняемому посредством модуля 42 оценки движения и модуля 44 компенсации движения, либо прогнозированию с внутриблочным копированием, выполняемому посредством модуля 48 внутреннего ВС, как описано выше. В частности, модуль 41 обработки внутреннего прогнозирования может определять режим внутреннего прогнозирования для использования для того, чтобы кодировать текущий блок. Для этого, модуль 46 обработки внутреннего прогнозирования может кодировать текущий блок с использованием различных режимов внутреннего прогнозирования, например, во время отдельных проходов кодирования, и модуль 46 обработки внутреннего прогнозирования (или модуль выбора режима, в некоторых примерах) может выбирать соответствующий режим внутреннего прогнозирования, который следует использовать, из протестированных режимов внутреннего прогнозирования. Модуль 46 обработки внутреннего прогнозирования может предоставлять информацию, указывающую выбранный режим внутреннего прогнозирования для блока, в модуль 56 энтропийного кодирования. Модуль 56 энтропийного кодирования может кодировать информацию, указывающую выбранный режим внутреннего прогнозирования, в потоке битов.

[00413] После того, как модуль 41 обработки прогнозирования определяет прогнозирующий блок для текущего видеоблока через взаимное прогнозирование или внутреннее прогнозирование, сумматор 50 формирует остаточный видеоблок посредством вычитания прогнозирующего блока из текущего видеоблока. Остаточные видеоданные в остаточном блоке могут включаться в одну или более TU и предоставляться в модуль 52 обработки преобразования. Модуль 52 обработки преобразования преобразует остаточные видеоданные в остаточные коэффициенты преобразования с использованием преобразования, такого как дискретное косинусное преобразование (DCT) или концептуально аналогичное преобразование.

[00414] Модуль 52 обработки преобразования может отправлять результирующие коэффициенты преобразования в модуль 54 квантования. Модуль 54 квантования квантует коэффициенты преобразования для того, чтобы дополнительно уменьшать скорость передачи битов. Процесс квантования также может уменьшать битовую глубину, ассоциированную с некоторыми или всеми коэффициентами. Степень квантования может модифицироваться посредством регулирования параметра квантования. В некоторых примерах, модуль 54 квантования затем может выполнять

сканирование матрицы, включающей в себя квантованные коэффициенты преобразования. Альтернативно, модуль 56 энтропийного кодирования может выполнять сканирование.

[00415] После квантования, модуль 56 энтропийного кодирования энтропийно кодирует квантованные коэффициенты преобразования в поток видеобитов с использованием, например, контекстно-адаптивного кодирования переменной длины (CAVLC), контекстно-адаптивного двоичного арифметического кодирования (CABAC), синтаксического контекстно-адаптивного двоичного арифметического кодирования (SBAC), энтропийного кодирования на основе сегментации на интервалы вероятности (PIPE) либо другой технологии энтропийного кодирования. Кодированный поток битов затем может передаваться в видеодекодер 30, как показано на фиг. 23, или архивироваться в устройстве 32 хранения данных, как показано на фиг. 23, для последующей передачи в или извлечения посредством видеодекодера 30. Модуль 56 энтропийного кодирования также может энтропийно кодировать векторы движения и другие синтаксические элементы для кодируемого текущего видеокадра.

[00416] Модуль 58 обратного квантования и модуль 60 обработки обратного преобразования применяют обратное квантование и обратное преобразование, соответственно, чтобы восстанавливать остаточный видеоблок в пиксельной области для формирования опорного блока для прогнозирования других видеоблоков. Как отмечено выше, модуль 44 компенсации движения может формировать прогнозирующий блок с компенсацией движения из одного или более опорных блоков кадров, сохраненных в DPB 64. Модуль 44 компенсации движения также может применять один или более интерполяционных фильтров к прогнозирующему блоку, чтобы вычислять субцелочисленнопиксельные значения для использования в оценке движения.

[00417] Сумматор 62 суммирует восстановленный остаточный блок с прогнозирующим блоком с компенсацией движения, сформированным посредством модуля 44 компенсации движения, чтобы формировать опорный блок для хранения в DPB 64. Опорный блок затем может использоваться посредством модуля 48 внутреннего ВС, модуля 42 оценки движения и модуля 44 компенсации движения в качестве прогнозирующего блока для того, чтобы взаимно прогнозировать другой видеоблок в последующем видеокадре.

[00418] Фиг. 25 является блок-схемой, иллюстрирующей примерный видеодекодер 30 в соответствии с некоторыми реализациями настоящей заявки. Видеодекодер 30 включает в себя запоминающее устройство 79 видеоданных, модуль 80 энтропийного декодирования, модуль 81 обработки прогнозирования, модуль 86 обратного квантования, модуль 88 обработки обратного преобразования, сумматор 90 и DPB 92. Модуль 81 обработки прогнозирования дополнительно включает в себя модуль 82 компенсации движения, модуль 84 внутреннего прогнозирования и модуль 85 внутреннего ВС. Видеодекодер 30 может выполнять процесс декодирования, в общем, взаимнообратный по отношению к процессу кодирования, описанному выше относительно видеокодера 20 в связи с фиг. 24. Например, модуль 82 компенсации движения может формировать прогнозирующие данные на основе векторов движения, принимаемых из модуля 80 энтропийного декодирования, в то время как модуль 84 внутреннего прогнозирования может формировать прогнозирующие данные на основе индикаторов режима внутреннего прогнозирования, принимаемых из модуля 80 энтропийного декодирования.

[00419] В некоторых примерах, для модуля видеодекодера 30 можно определять задачу, чтобы выполнять реализации настоящей заявки. Кроме того, в некоторых

примерах, реализации настоящего раскрытия могут разделяться между одним или более модулей видеodeкодера 30. Например, модуль 85 внутреннего ВС может выполнять реализации настоящей заявки, отдельно или в комбинации с другими модулями видеodeкодера 30, такими как модуль 82 компенсации движения, модуль 84 внутреннего прогнозирования и модуль 80 энтропийного декодирования. В некоторых примерах, видеodeкодер 30 может не включать в себя модуль 85 внутреннего ВС, и функциональность модуля 85 внутреннего ВС может выполняться посредством других компонентов модуля 81 обработки прогнозирования, таких как модуль 82 компенсации движения.

[00420] Запоминающее устройство 79 видеоданных может сохранять видеоданные, такие как кодированный поток видеобитов, который должен декодироваться посредством других компонентов видеodeкодера 30. Видеоданные, сохраненные в запоминающем устройстве 79 видеоданных, могут получаться, например, из устройства 32 хранения данных, из локального видеоисточника, к примеру, камеры, через передачу видеоданных по проводной или беспроводной сети либо посредством осуществления доступа к физическим носителям хранения данных (например, к флэш-накопителю или жесткому диску). Запоминающее устройство 79 видеоданных может включать в себя буфер кодированных картинок (CPB), который сохраняет кодированные видеоданные из кодированного потока видеобитов. DPB 92 видеodeкодера 30 сохраняет опорные видеоданные для использования при декодировании видеоданных посредством видеodeкодера 30 (например, в режимах внутреннего или взаимного прогнозирующего кодирования). Запоминающее устройство 79 видеоданных и буфер 92 декодированных картинок могут формироваться посредством любых из множества запоминающих устройств, к примеру, как динамическое оперативное запоминающее устройство (DRAM), включающее в себя синхронное DRAM (SDRAM), магниторезистивное RAM (MRAM), резистивное RAM (RRAM) или другие типы запоминающих устройств. В качестве иллюстрации, запоминающее устройство 79 видеоданных и DPB 92 проиллюстрированы как два различных компонента видеodeкодера 30 на фиг. 25. Но специалистам в данной области техники должно быть очевидным, что запоминающее устройство 79 видеоданных и DPB 92 могут предоставляться посредством одного и того же запоминающего устройства или отдельных запоминающих устройств. В некоторых примерах, запоминающее устройство 79 видеоданных может быть внутримикросхемным с другими компонентами видеodeкодера 30 или внешним относительно этих компонентов.

[00421] В ходе процесса декодирования, видеodeкодер 30 принимает кодированный поток видеобитов, который представляет видеоблоки кодированного видеокadra и ассоциированные синтаксические элементы. Видеodeкодер 30 может принимать синтаксические элементы на уровне видеокadra и/или на уровне видеоблока. Модуль 80 энтропийного декодирования видеodeкодера 30 энтропийно декодирует поток битов, чтобы формировать квантованные коэффициенты, векторы движения или индикаторы режима внутреннего прогнозирования и другие синтаксические элементы. Модуль 80 энтропийного декодирования затем перенаправляет векторы движения или индикаторы режима внутреннего прогнозирования и другие синтаксические элементы в модуль 81 обработки прогнозирования.

[00422] Когда видеокادر кодируется в качестве внутреннего прогнозирующего кодированного (I-) кадра или для внутренне кодированных прогнозирующих блоков в других типах кадров, модуль 84 внутреннего прогнозирования модуля 81 обработки прогнозирования может формировать прогнозирующие данные для видеоблока

текущего видеокадра на основе сигнализируемого режима внутреннего прогнозирования и опорных данных из ранее декодированных блоков текущего кадра.

[00423] Когда видеокадр кодируется в качестве взаимного прогнозирующего кодированного (т.е. В- или Р-) кадра, модуль 82 компенсации движения модуля 81 обработки прогнозирования формирует один или более прогнозирующих блоков для видеоблока текущего видеокадра на основе векторов движения и других синтаксических элементов, принимаемых из модуля 80 энтропийного декодирования. Каждый из прогнозирующих блоков может формироваться из опорного кадра в одном из списков опорных кадров. Вicodeкодер 30 может конструировать списки опорных кадров, список 0 и список 1, с использованием технологий конструирования по умолчанию на основе опорных кадров, сохраненных в DPB 92.

[00424] В некоторых примерах, когда видеоблок кодируется согласно режиму внутреннего ВС, описанному в данном документе, модуль 85 внутреннего ВС модуля 81 обработки прогнозирования формирует прогнозирующие блоки для текущего видеоблока на основе блочных векторов и других синтаксических элементов, принимаемых из модуля 80 энтропийного декодирования. Прогнозирующие блоки могут находиться в восстановленной области той же картинке, что и текущий видеоблок, заданной посредством видеокодера 20.

[00425] Модуль 82 компенсации движения и/или модуль 85 внутреннего ВС определяют информацию прогнозирования для видеоблока текущего видеокадра посредством синтаксического анализа векторов движения и других синтаксических элементов и затем используют информацию прогнозирования для того, чтобы формировать прогнозирующие блоки для текущего декодируемого видеоблока. Например, модуль 82 компенсации движения использует некоторые принимаемые синтаксические элементы для того, чтобы определять режим прогнозирования (например, внутреннее или взаимное прогнозирование), используемый для того, чтобы кодировать видеоблоки видеокадра, тип взаимного прогнозного кадра (например, В или Р), информацию конструирования для одного или более списков опорных кадров для кадра, векторы движения для каждого взаимного прогнозирующего кодированного видеоблока кадра, состояние взаимного прогнозирования для каждого взаимного прогнозирующего кодированного видеоблока кадра и другую информацию, чтобы декодировать видеоблоки в текущем видеокадре.

[00426] Аналогично, модуль 85 внутреннего ВС может использовать некоторые принимаемые синтаксические элементы, например, флаг, чтобы определять то, что текущий видеоблок прогнозирован с использованием режима внутреннего ВС, информацию конструирования в отношении того, какие видеоблоки кадра находятся в восстановленной области и должны сохраняться в DPB 92, блочные векторы для каждого внутреннего ВС-прогнозированного видеоблока кадра, состояние внутреннего ВС-прогнозирования для каждого внутреннего ВС-прогнозированного видеоблока кадра и другую информацию, чтобы декодировать видеоблоки в текущем видеокадре.

[00427] Модуль 82 компенсации движения также может выполнять интерполяцию с использованием интерполяционных фильтров, используемую посредством видеокодера 20 во время кодирования видеоблоков, чтобы вычислять интерполированные значения для субцелочисленных пикселей опорных блоков. В этом случае, модуль 82 компенсации движения может определять интерполяционные фильтры, используемые посредством видеокодера 20, из принимаемых синтаксических элементов и использовать интерполяционные фильтры для того, чтобы формировать прогнозирующие блоки.

[00428] Модуль 86 обратного квантования обратно квантует квантованные

коэффициенты преобразования, предоставленные в потоке битов и энтропийно декодированные посредством модуля 80 энтропийного декодирования, с использованием одинакового параметра квантования, вычисленного посредством видеокодера 20 для каждого видеоблока в видеокадре, чтобы определять степень квантования. Модуль 88

5 обработки обратного преобразования применяет обратное преобразование, например, обратное DCT, обратное целочисленное преобразование или концептуально аналогичный процесс обратного преобразования, к коэффициентам преобразования для того, чтобы восстанавливать остаточные блоки в пиксельной области.

[00429] После того, как модуль 82 компенсации движения или модуль 85 внутреннего

10 ВС формирует прогнозирующий блок для текущего видеоблока на основе векторов и других синтаксических элементов, сумматор 90 восстанавливает декодированный видеоблок для текущего видеоблока посредством суммирования остаточного блока из модуля 88 обработки обратного преобразования и соответствующего прогнозирующего блока, сформированного посредством модуля 82 компенсации

15 движения и модуля 85 внутреннего ВС. Внутриконтурный фильтр 91, такой как фильтр удаления блочности, SAO-фильтр и/или ALF, может позиционироваться между сумматором 90 и DPB 92, чтобы дополнительно обрабатывать декодированный видеоблок. В некоторых примерах, контурный фильтр 91 может опускаться, и декодированный видеоблок может непосредственно предоставляться посредством

20 сумматора 90 в DPB 92. Декодированные видеоблоки в данном кадре затем сохраняются в DPB 92, который сохраняет опорные кадры, используемые для последующей компенсации движения следующих видеоблоков. DPB 92 или запоминающее устройство, отдельное от DPB 92, также может сохранять декодированное видео для последующего представления на устройстве отображения, к примеру, на устройстве 34 отображения

25 по фиг. 23.

[00430] Описание настоящего раскрытия представлено для целей иллюстрации и не имеет намерение быть исчерпывающим или ограниченным настоящим раскрытием. Множество модификаций, варьирований и альтернативных реализаций должны становиться очевидными для специалистов в данной области техники с использованием

30 преимущества идей, представленных в вышеприведенных описаниях и на ассоциированных чертежах.

[00431] Примеры выбраны и описаны для того, чтобы пояснять принципы раскрытия и обеспечивать возможность специалистам в данной области техники понимать раскрытие для различных реализаций и лучше всего использовать базовые принципы

35 и различные реализации с различными модификациями, которые подходят для конкретного предполагаемого использования. Следовательно, следует понимать, что объем раскрытия не должен ограничиваться конкретными примерами раскрытых реализаций, и что модификации и другие реализации имеют намерение включаться в пределы объема настоящего раскрытия.

(57) Формула изобретения

1. Способ для декодирования видео, содержащий этап, на котором:

- в ответ на определение, что пропуск преобразования деактивирован, определяют посредством декодера, что синтаксический элемент, относящийся к параметру Райса,

45 для остаточного кодирования с пропуском преобразования не присутствует.

2. Способ для декодирования видео по п. 1, дополнительно содержащий этап, на котором:

- принимают флаг пропуска преобразования, `sps_transform_skip_enabled_flag`, чтобы

обуславливать присутствие `sps_ts_residual_coding_rice_present_in_sh_flag`.

3. Способ для декодирования видео по п. 2, дополнительно содержащий этап, на котором:

- в ответ на определение того, что флаг, `sps_transform_skip_enabled_flag`, равен 0, указывая, что пропуск преобразования деактивирован, логически выводят `sps_ts_residual_coding_rice_present_in_sh_flag` равным 0 и не сигнализируемым.

4. Способ для декодирования видео по п. 2, дополнительно содержащий этап, на котором:

- в ответ на определение, что флаг, `sps_transform_skip_enabled_flag`, равен 1, указывая, что пропуск преобразования не деактивирован, принимают `sps_ts_residual_coding_rice_present_in_sh_flag`.

5. Способ для декодирования видео по п. 1, дополнительно содержащий этап, на котором:

- в ответ на определение, что флаг пропуска преобразования, `sps_transform_skip_enabled_flag`, сигнализируется в качестве активированного, принимают флаг управления, в наборе параметров последовательности или в синтаксисе расширения диапазона набора параметров последовательности, чтобы указывать, присутствует или нет синтаксический элемент, относящийся к параметру Райса, для остаточного кодирования с пропуском преобразования.

6. Способ для декодирования видео по п. 5, дополнительно содержащий этап, на котором:

- в ответ на определение, что флаг управления равен 1, принимают один синтаксический элемент для некоторого среза, чтобы определять параметр Райса для остаточного кодирования с пропуском преобразования.

7. Способ для декодирования видео по п. 5, дополнительно содержащий этап, на котором:

- в ответ на определение того, что флаг управления равен 0, принимают параметр Райса по умолчанию для некоторого среза, и дополнительный синтаксический элемент не сигнализируется для того, чтобы указывать параметр Райса для остаточного кодирования с пропуском преобразования.

8. Способ для декодирования видео по п. 2, дополнительно содержащий этап, на котором:

- в ответ на определение, что `sps_ts_residual_coding_rice_present_in_sh_flag` равен 1, определяют, что индекс, `sh_ts_residual_coding_rice_idx_minus1`, присутствует в синтаксических структурах заголовков срезов (SH), ссылающихся на набор параметров последовательности (SPS).

9. Способ для декодирования видео по п. 2, дополнительно содержащий этап, на котором:

- в ответ на определение, что `sps_ts_residual_coding_rice_present_in_sh_flag` равен 0, определяют, что индекс, `sh_ts_residual_coding_rice_idx_minus1`, не присутствует в синтаксических структурах заголовков срезов (SH), ссылающихся на набор параметров последовательности (SPS).

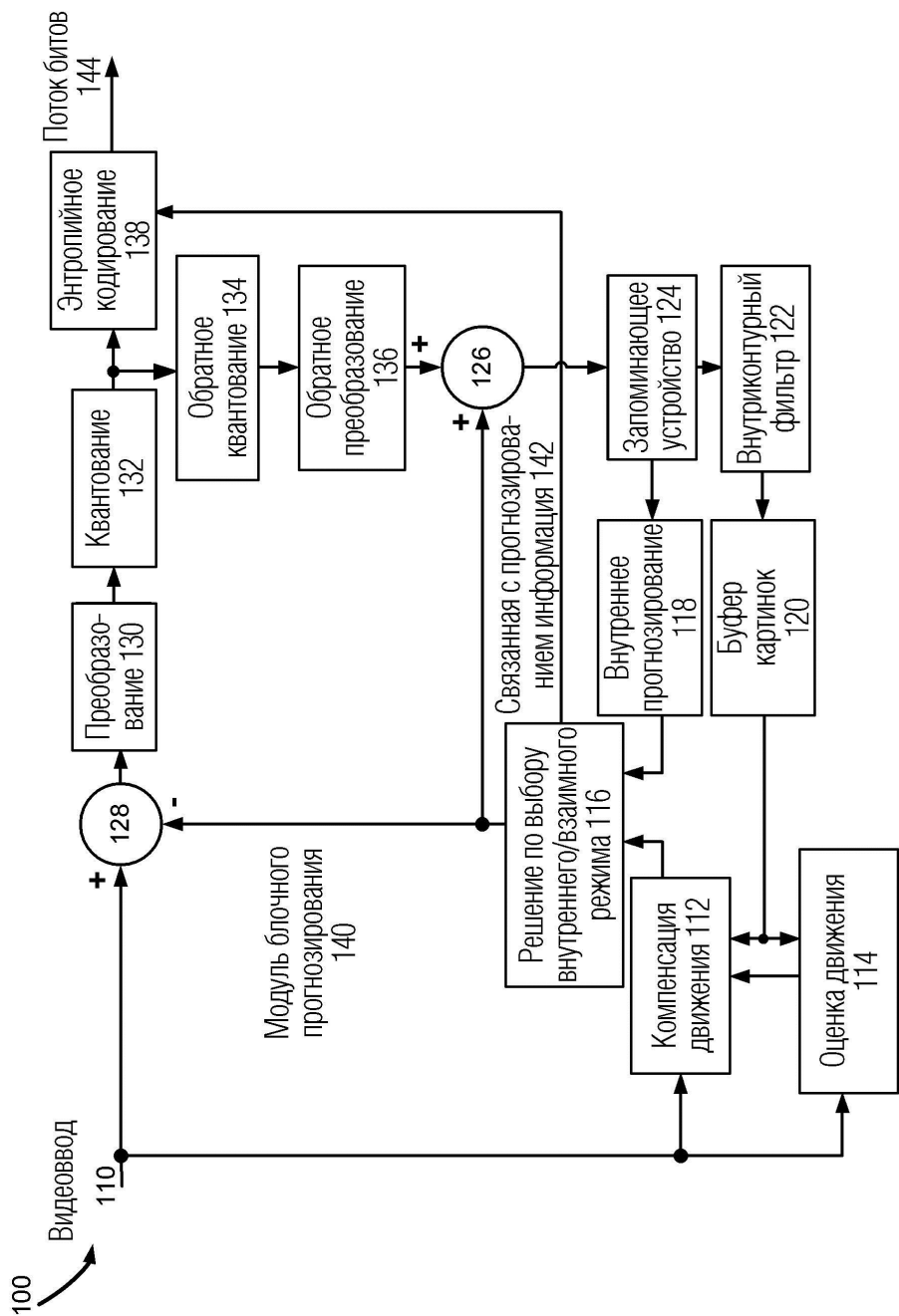
10. Способ для декодирования видео по п. 2, дополнительно содержащий этап, на котором:

- в ответ на определение, что `sps_ts_residual_coding_rice_present_in_sh_flag` не присутствует, логически выводят значение `sps_ts_residual_coding_rice_present_in_sh_flag` равным 0.

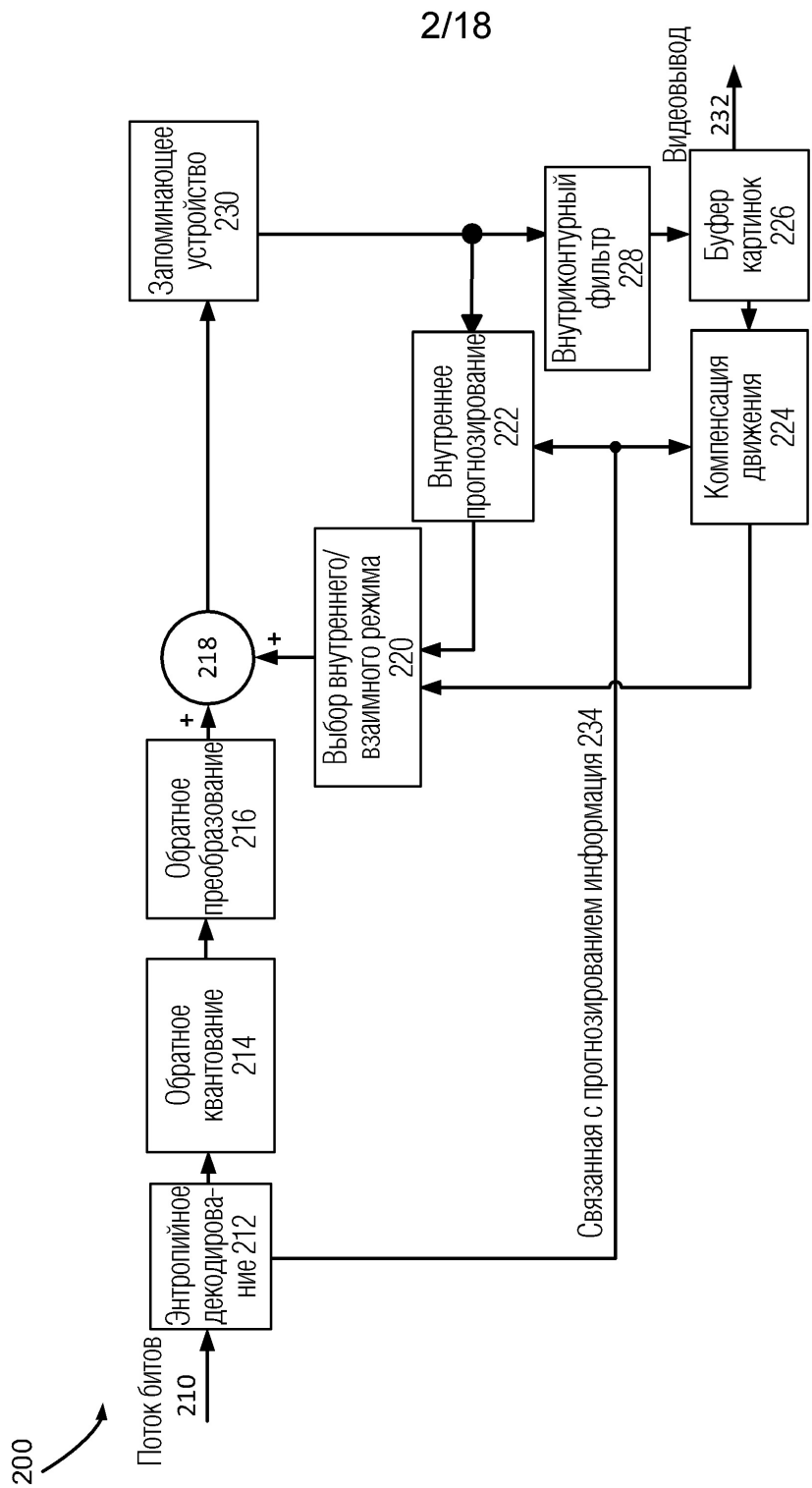
11. Оборудование для декодирования видео, содержащее:

- один или более процессоров; и
- запоминающее устройство, выполненное с возможностью сохранять инструкции, выполняемые посредством одного или более процессоров; при этом один или более процессоров при выполнении инструкций выполнены с возможностью осуществлять способ по любому из пп. 1-10.

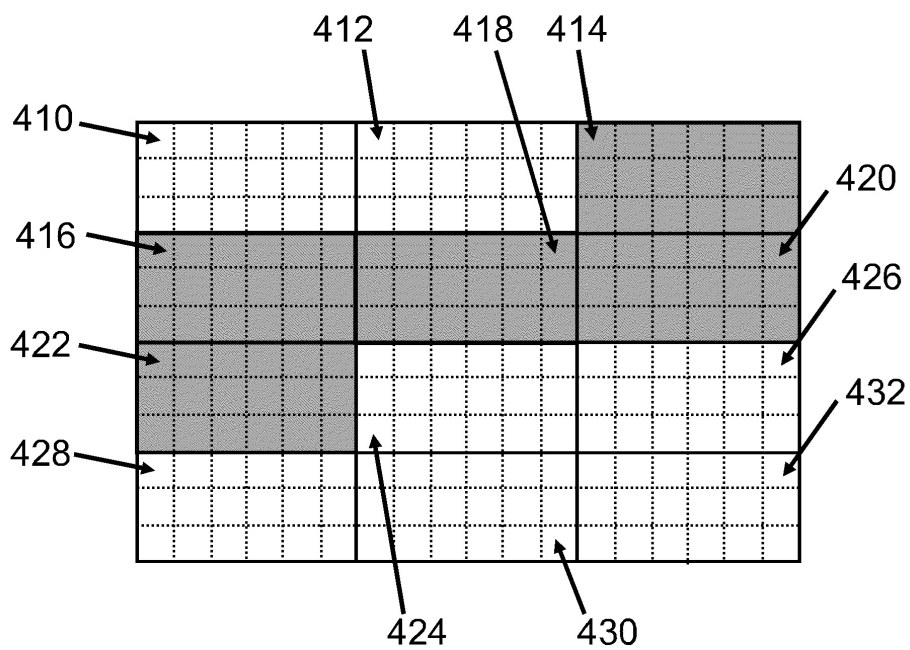
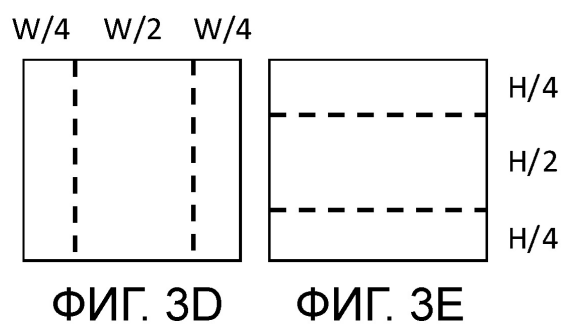
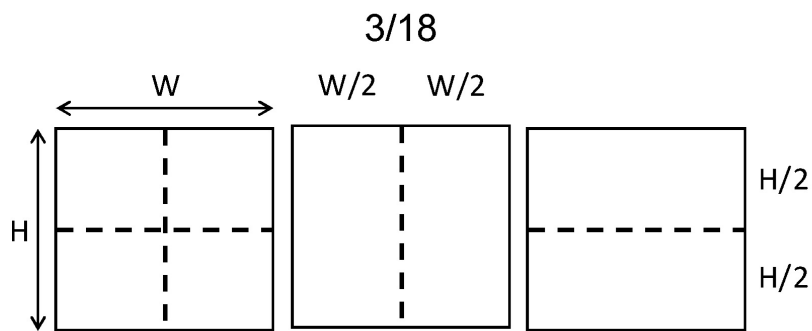
12. Энергонезависимый машиночитаемый носитель хранения данных для декодирования видео, сохраняющий машиноисполняемые инструкции, которые при выполнении посредством одного или более компьютерных процессоров инструктируют одному или более компьютерных процессоров осуществлять способ по любому из пп. 1-10, чтобы декодировать поток битов и сохранять декодированный поток битов в упомянутом энергонезависимом машиночитаемом носителе хранения данных.



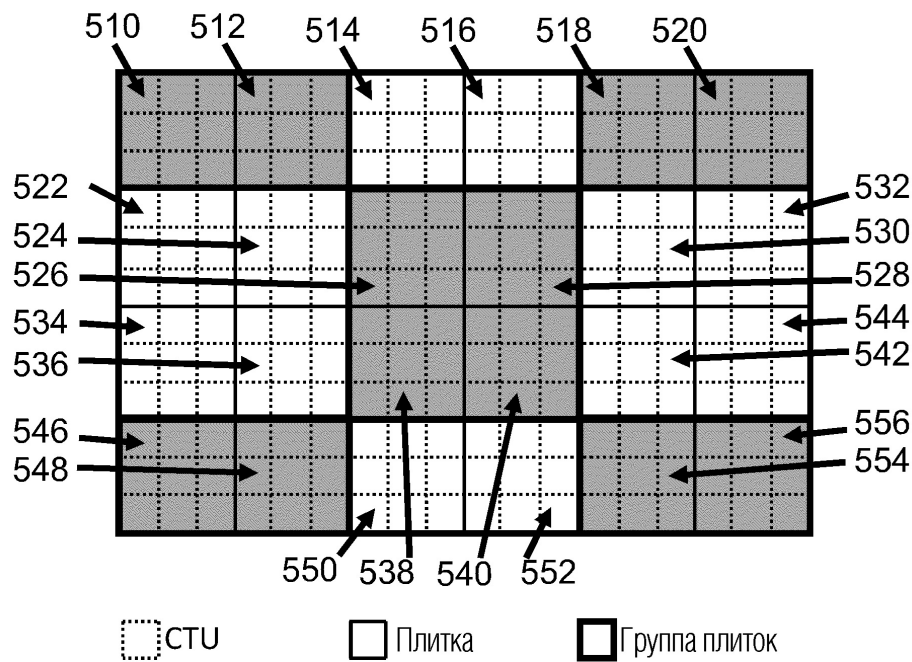
ФИГ. 1



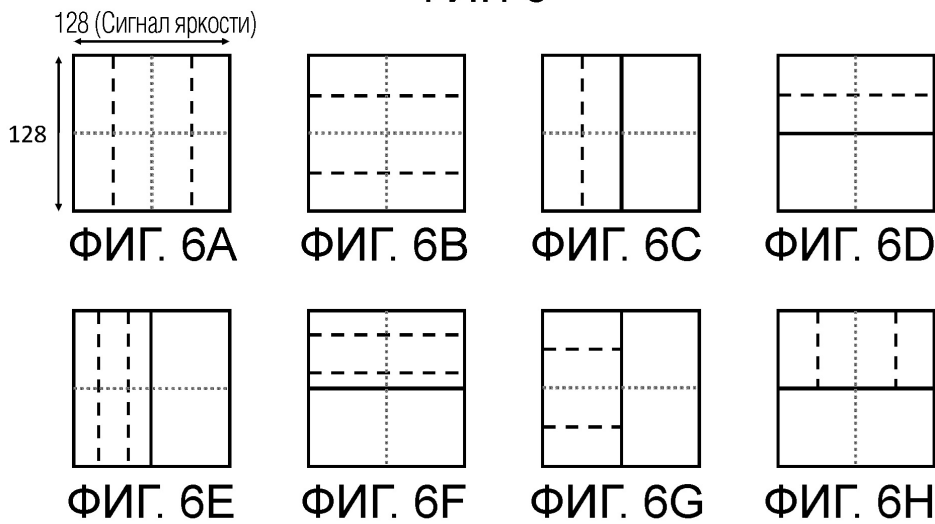
ФИГ. 2

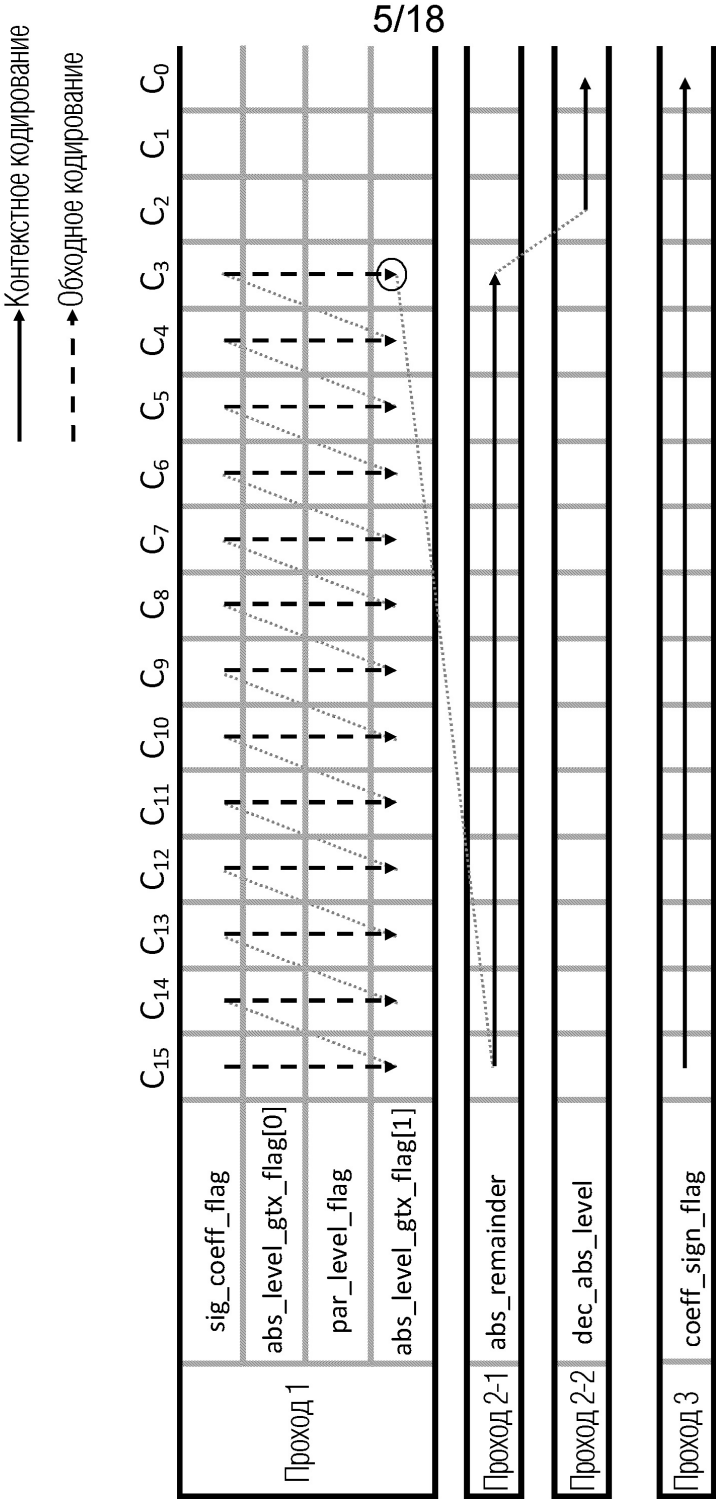


4/18

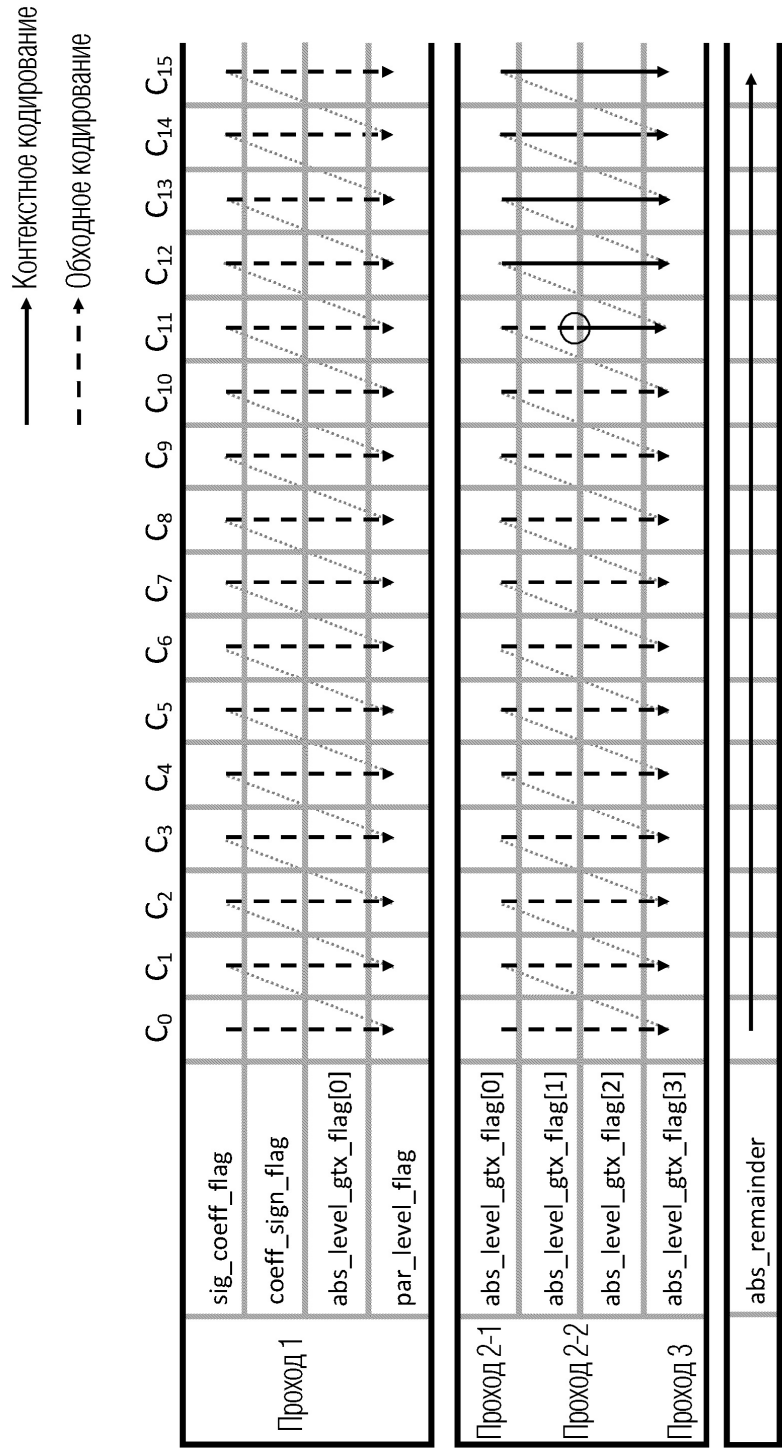


ФИГ. 5



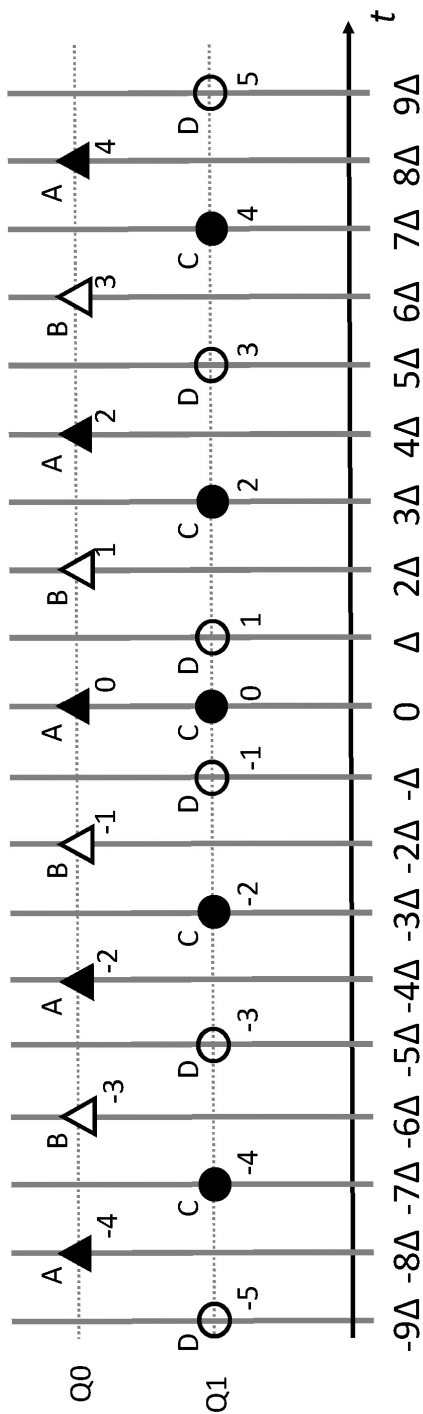


ФИГ. 7

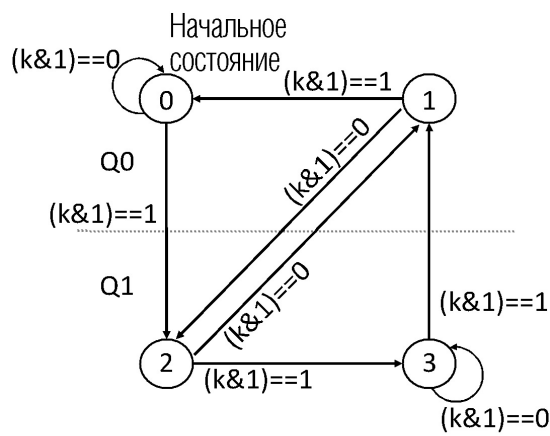


ФИГ. 8

7/18



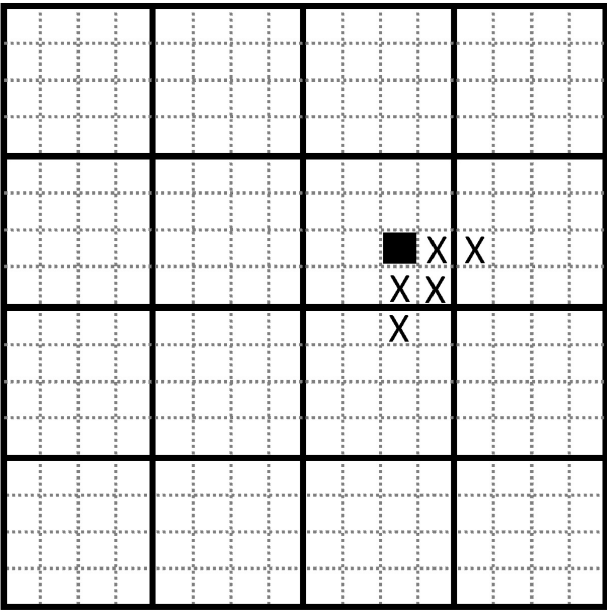
ФИГ. 9



ФИГ. 10А

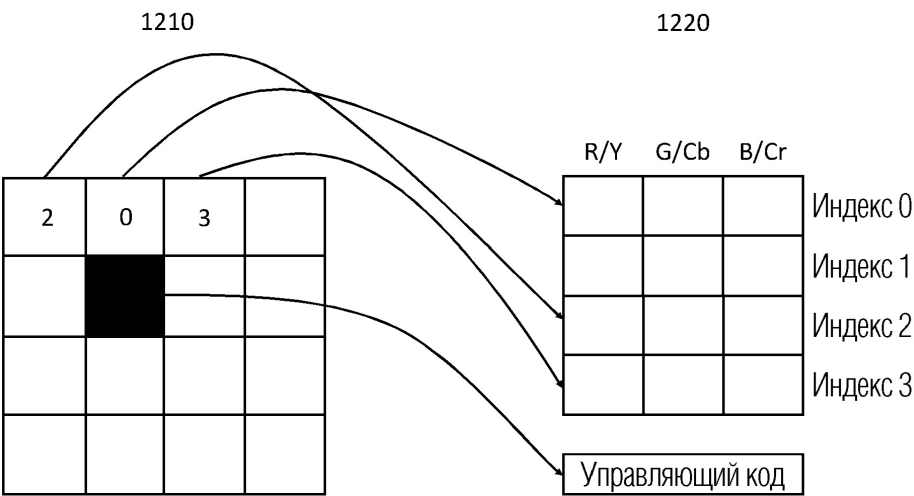
Текущее состояние	Следующее состояние для...	
	$(k \& 1) == 0$	$(k \& 1) == 1$
0	0	2
1	2	0
2	1	3
3	3	1

ФИГ. 10В

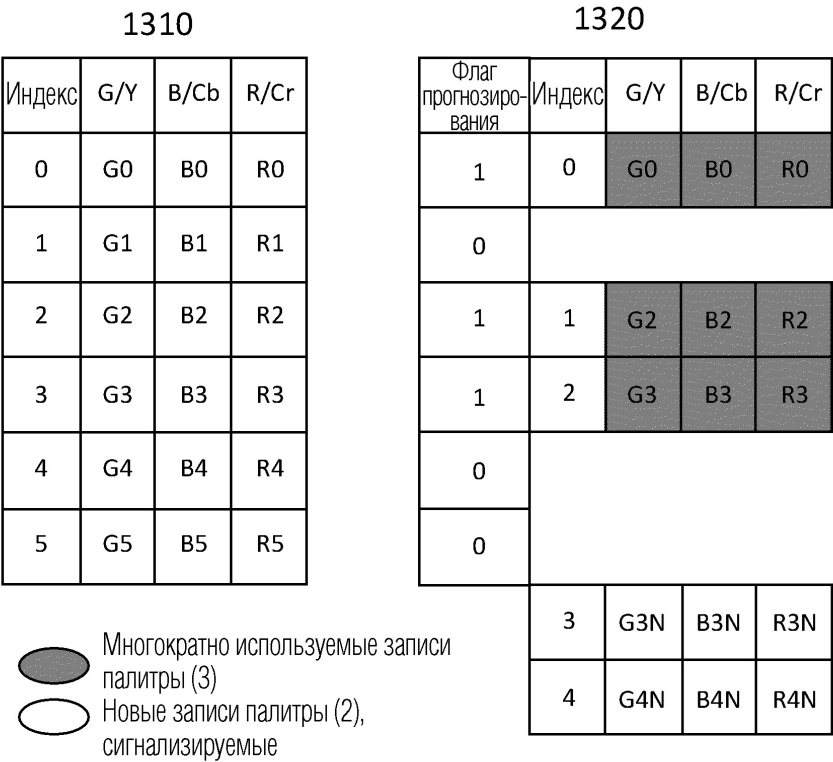


ФИГ. 11

9/18

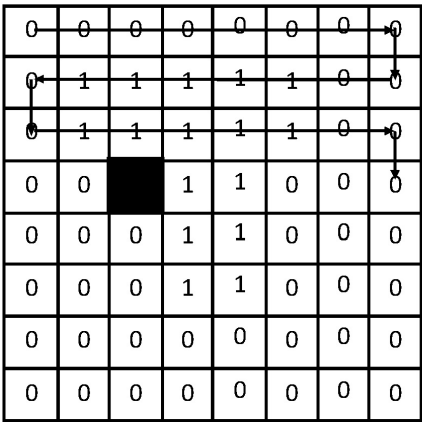


ФИГ. 12

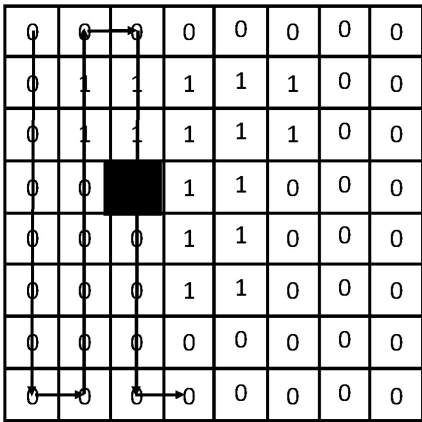


ФИГ. 13

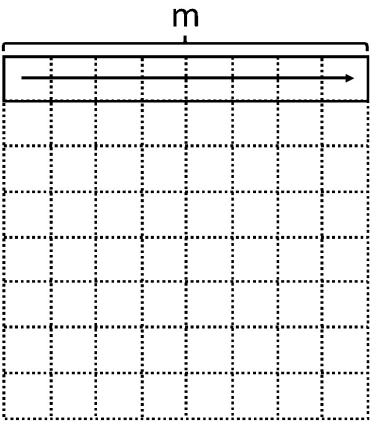
10/18



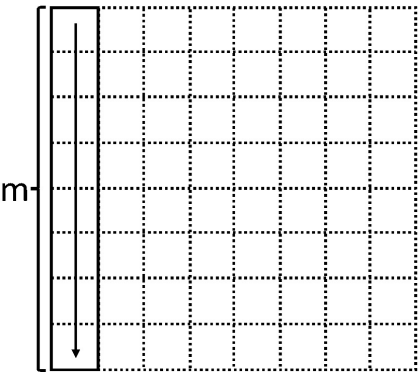
ФИГ. 14А



ФИГ. 14В

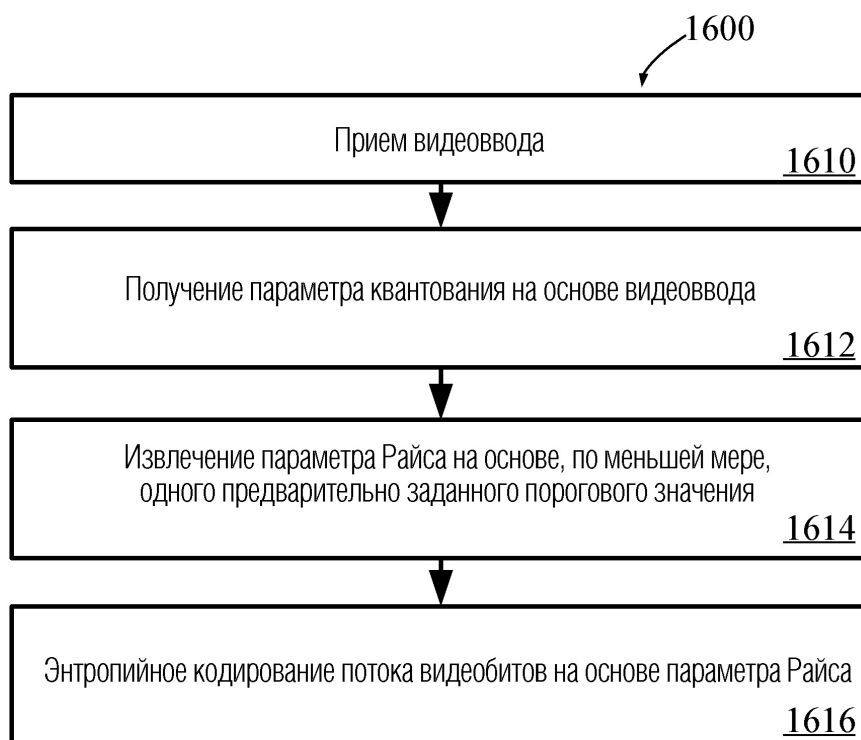


ФИГ. 15А



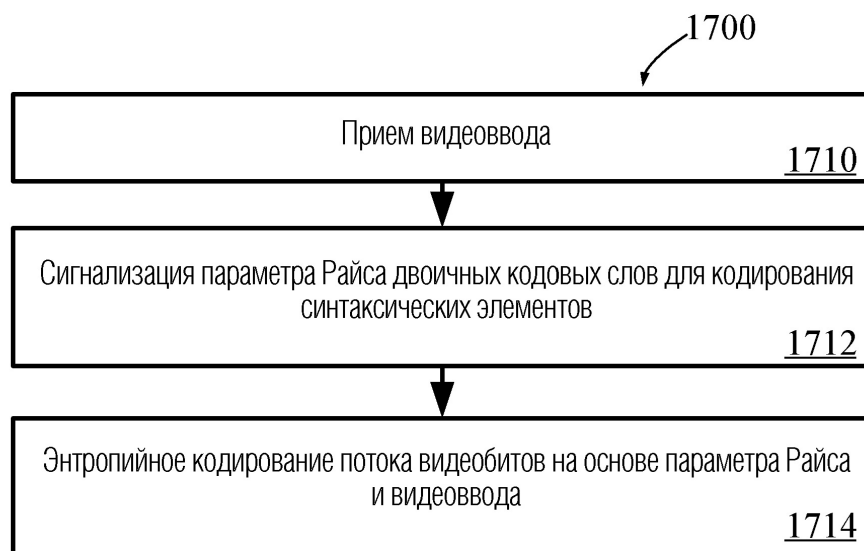
ФИГ. 15В

11/18



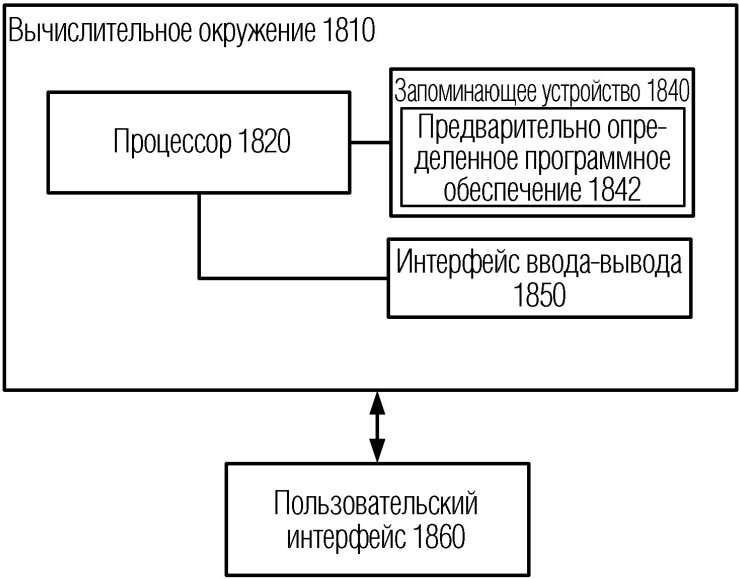
ФИГ. 16

12/18

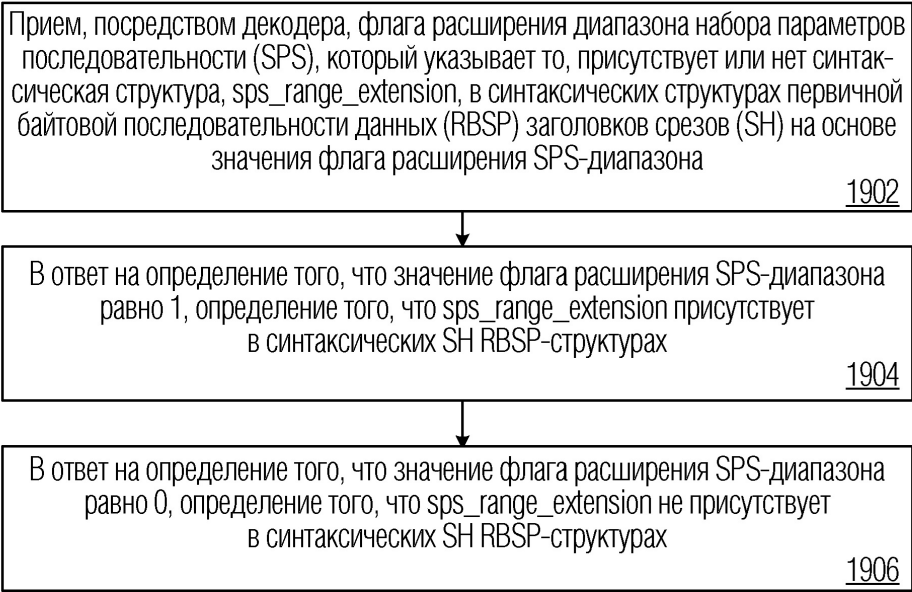


ФИГ. 17

13/18

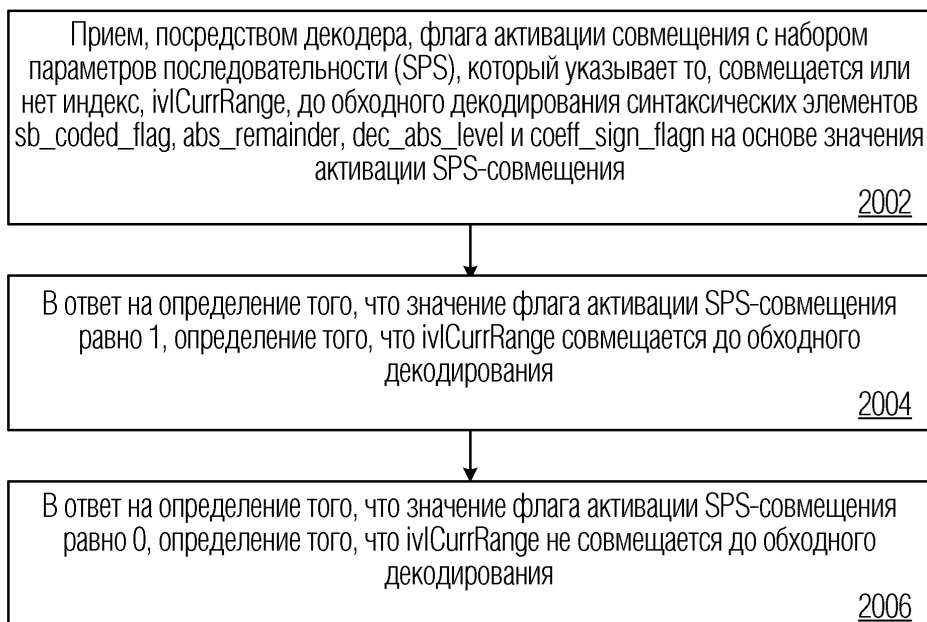


ФИГ. 18

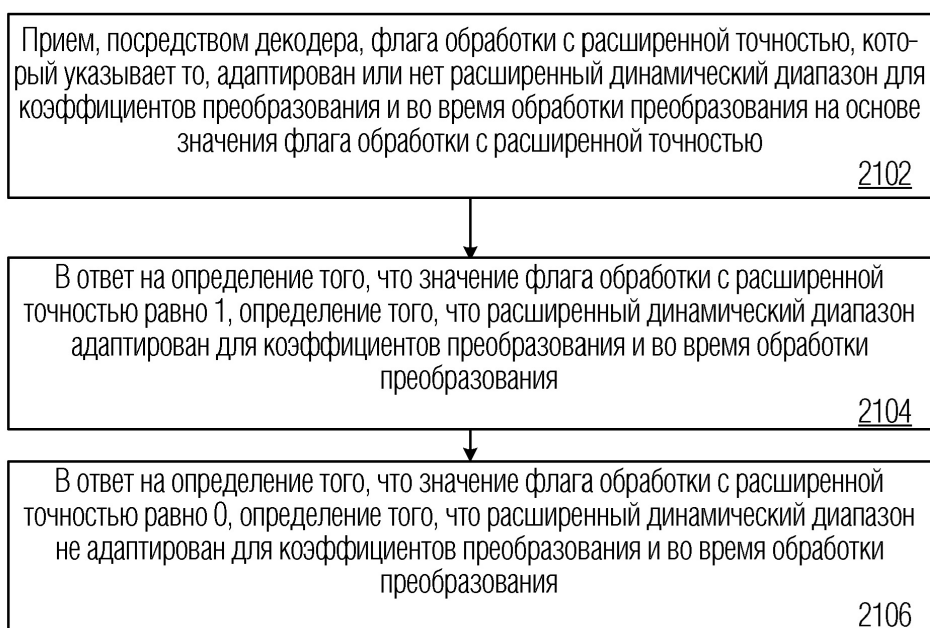


ФИГ. 19

14/18

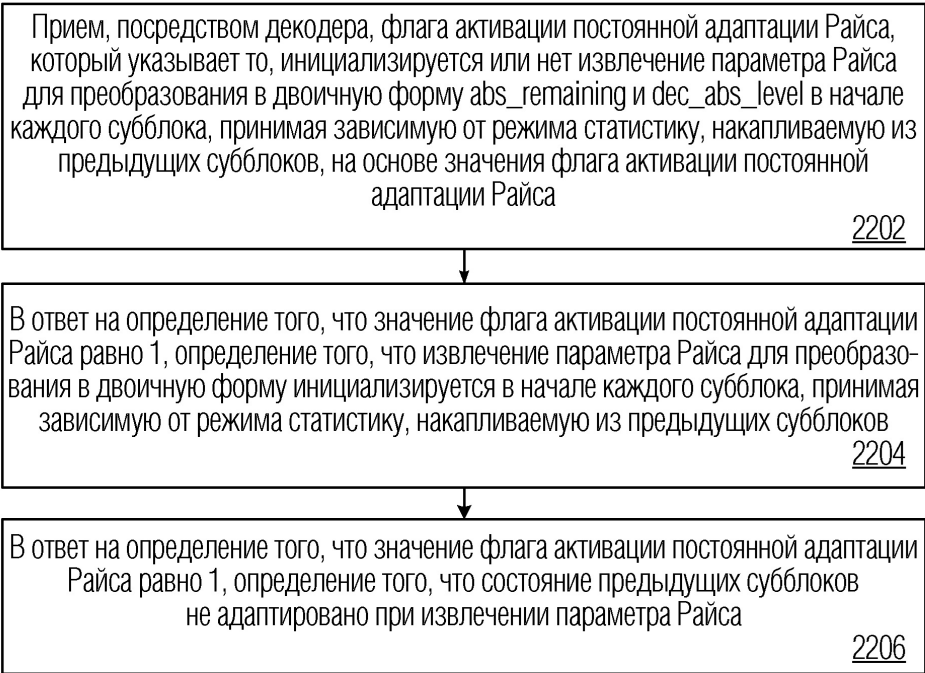


ФИГ. 20

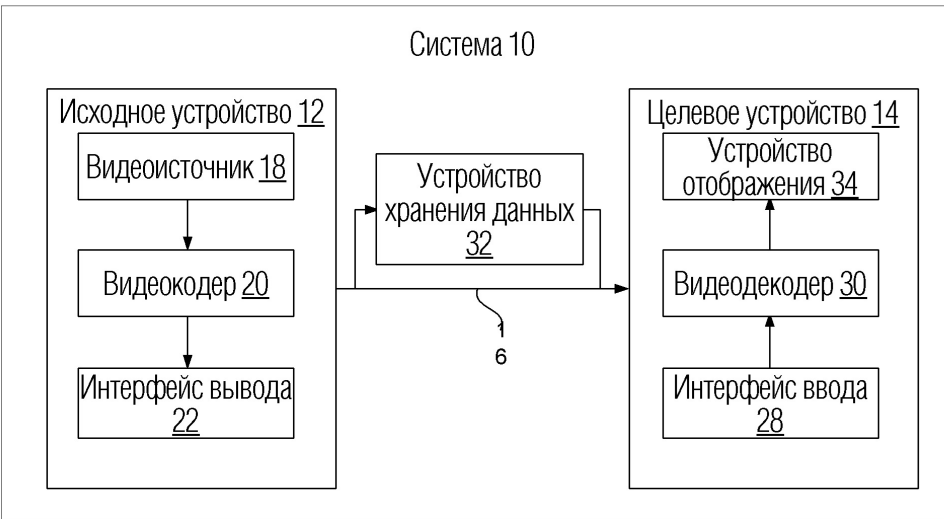


ФИГ. 21

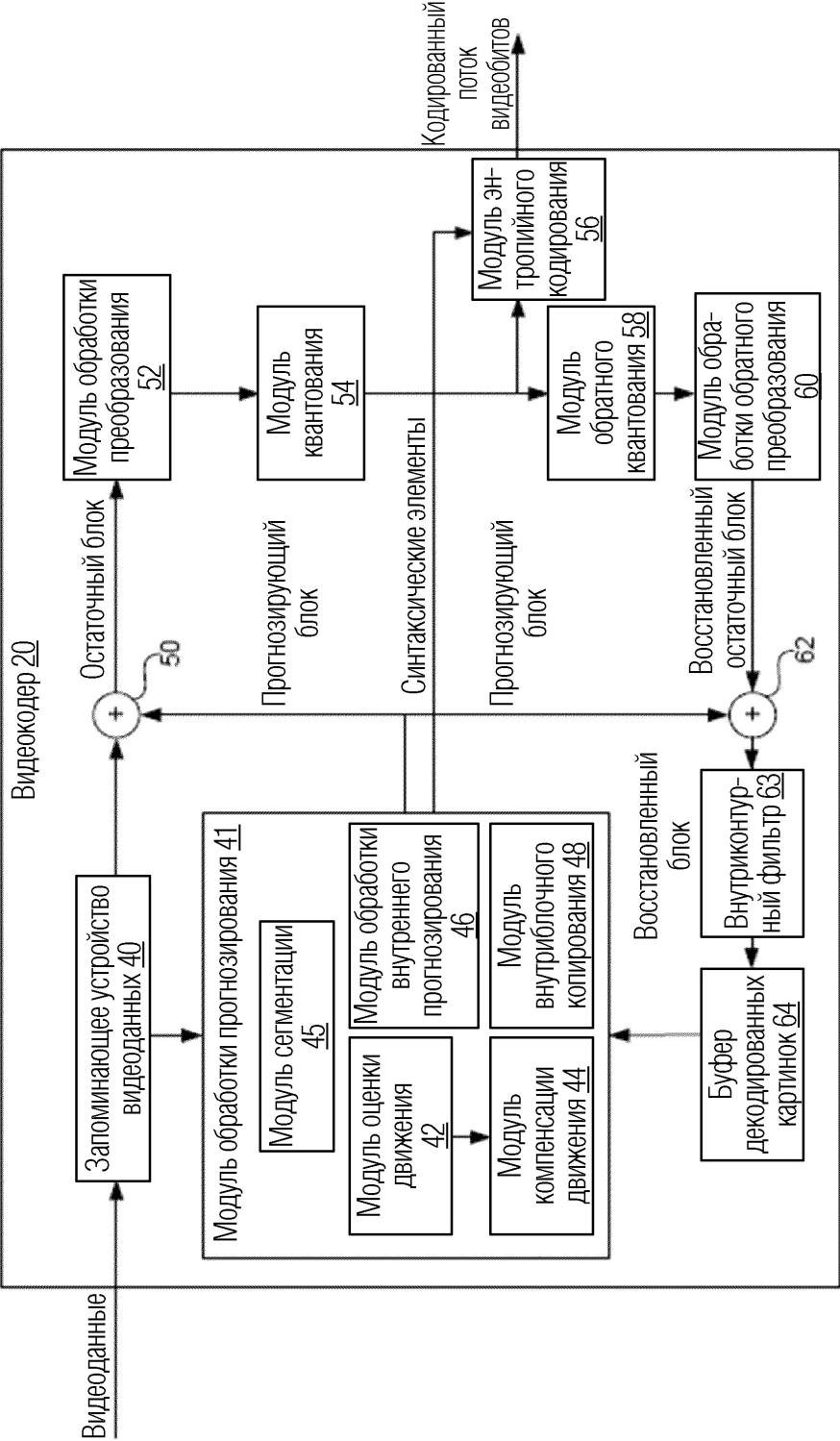
15/18



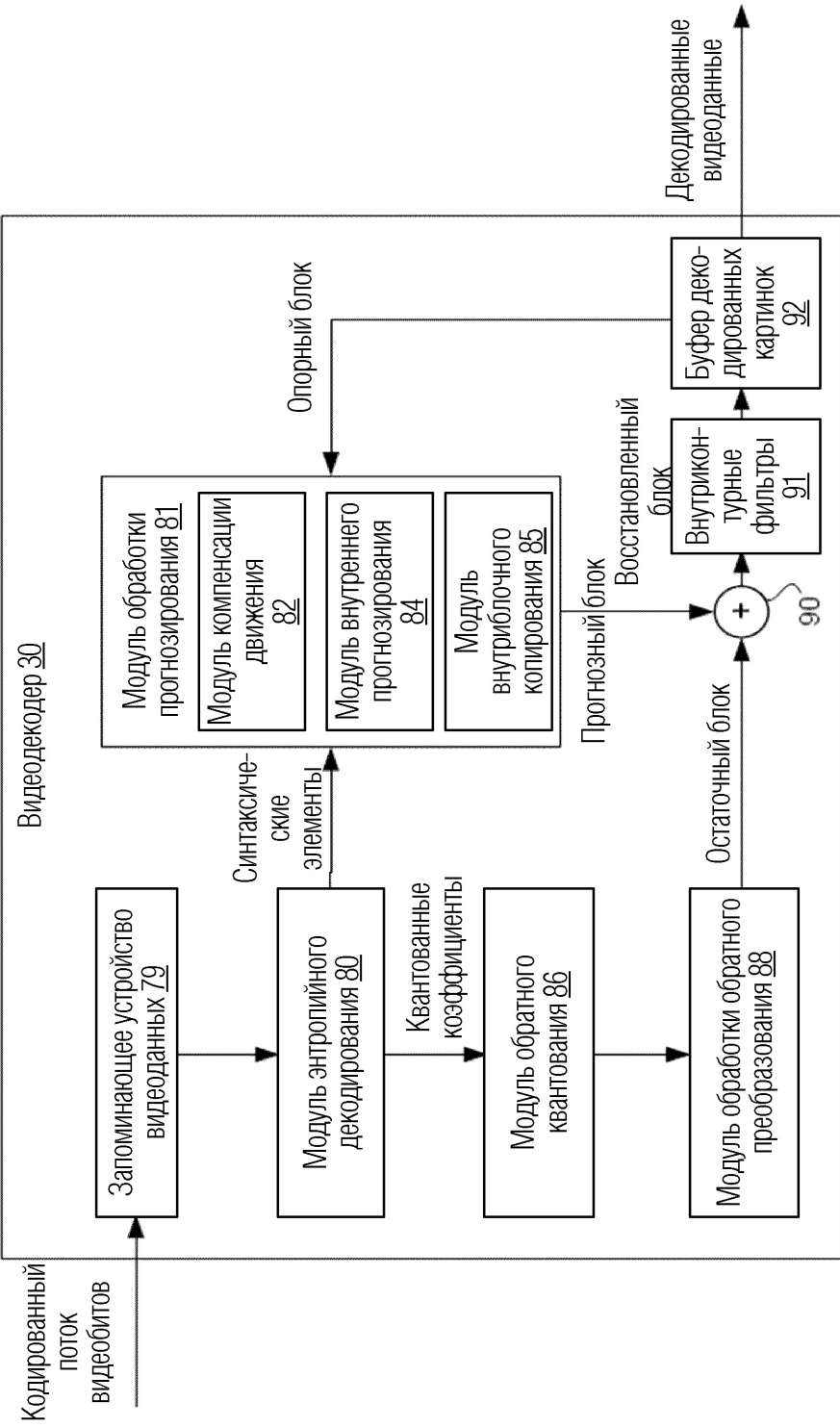
ФИГ. 22



ФИГ. 23

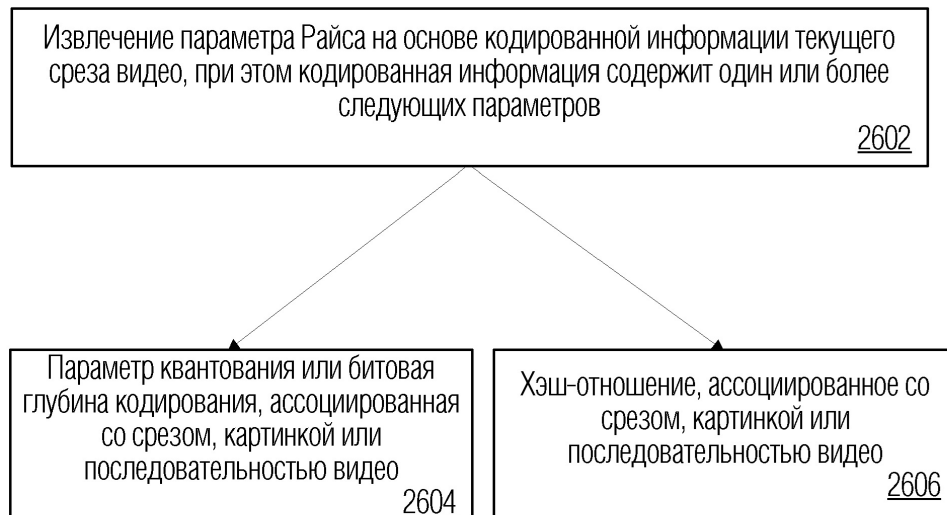


ФИГ. 24

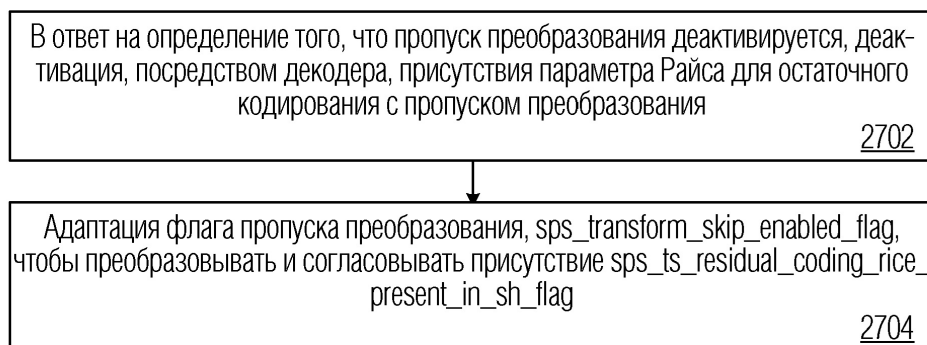


ФИГ. 25

18/18



ФИГ. 26



ФИГ. 27