



ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК

H04N 19/13 (2019.08); H04N 19/91 (2019.08); H04N 19/70 (2019.08); H04N 19/00 (2013.01); H04N 17/00 (2013.01); H04N 19/96 (2013.01); H04N 19/463 (2019.08); H04N 19/174 (2019.08); H04N 19/436 (2019.08); H04N 19/169 (2019.08)

(21)(22) Заявка: 2016141337, 15.04.2013

(24) Дата начала отсчета срока действия патента:
15.04.2013

Дата регистрации:
14.01.2020

Приоритет(ы):

(30) Конвенционный приоритет:
13.04.2012 US 61/624,098;
29.06.2012 US 61/666,185

Номер и дата приоритета первоначальной заявки,
из которой данная заявка выделена:
2014145559 13.04.2012

(43) Дата публикации заявки: 18.12.2018 Бюл. № 35

(45) Опубликовано: 14.01.2020 Бюл. № 2

Адрес для переписки:

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

(72) Автор(ы):

ШИРЛЬ Томас (DE),
ГЕОРГЕ Валери (DE),
ХЕНКЕЛЬ Анастасия (DE),
МАРПЕ Детлеф (DE),
ГРЮНЕБЕРГ Карстен (DE),
СКУПИН Роберт (DE)

(73) Патентообладатель(и):

ДжиИ Видео Компреши, ЭлЭлСи (US)

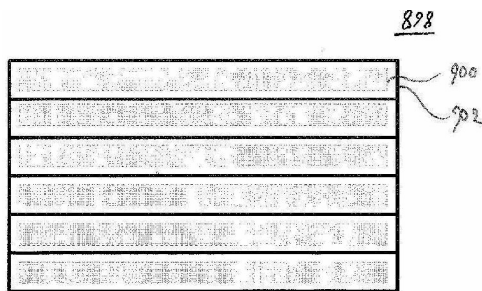
(56) Список документов, цитированных в отчете
о поиске: US 5786858 A, 28.07.1998. US 7782947
B2, 24.08.2010. WO 2010/050157 A1, 06.05.2010.
US 2008/0285657 A1, 20.11.2008. WO 2011/100456
A1, 18.08.2011. US 7738560 B2, 15.06.2010. US
2012/0014454 A1, 19.01.2012. US 2012/0082235
A1, 05.04.2012. RU 2384970 C1, 20.03.2010.

(54) КОДИРОВАНИЕ ИЗОБРАЖЕНИЙ С МАЛОЙ ЗАДЕРЖКОЙ

(57) Реферат:

Изобретение относится к вычислительной технике. Технический результат – повышение эффективности кодирования посредством уменьшения расходуемых издержек кодирования. Декодер для восстановления изображения из потока данных, в который изображение кодировано в единицах слайсов, на которые изображение разделено, используя WPP-обработку, при этом декодер выполнен с возможностью декодирования слайсов из потока данных в соответствии с порядком слайсов, при этом каждый слайс содержит начальный участок

синтаксиса, указывающий позицию начала декодирования соответствующего слайса в изображении, и декодер выполнен с возможностью: идентификации точек входа подпотоков WPP, в которые слайсы сгруппированы, посредством идентификации, используя начальные участки синтаксиса слайсов, слайсов, начинающихся на левой стороне изображения, параллельного декодирования подпотоков WPP ступенчатым образом с последовательным началом декодирования. 7 н. и 3 з.п. ф-лы, 32 ил., 4 табл.



Фиг.1

RU 2710908 C2

RU 2710908 C2



FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(51) Int. Cl.

H04N 19/13 (2014.01)*H04N 19/91* (2014.01)*H04N 19/70* (2014.01)(12) **ABSTRACT OF INVENTION**

(52) CPC

H04N 19/13 (2019.08); *H04N 19/91* (2019.08); *H04N 19/70* (2019.08); *H04N 19/00* (2013.01); *H04N 17/00* (2013.01); *H04N 19/96* (2013.01); *H04N 19/463* (2019.08); *H04N 19/174* (2019.08); *H04N 19/436* (2019.08); *H04N 19/169* (2019.08)

(21)(22) Application: **2016141337, 15.04.2013**(24) Effective date for property rights:
15.04.2013Registration date:
14.01.2020

Priority:

(30) Convention priority:
13.04.2012 US 61/624,098;
29.06.2012 US 61/666,185Number and date of priority of the initial application,
from which the given application is allocated:
2014145559 13.04.2012(43) Application published: **18.12.2018 Bull. № 35**(45) Date of publication: **14.01.2020 Bull. № 2**Mail address:
129090, Moskva, ul. B.Spasskaya, 25, stroenie 3,
OOO "Yuridicheskaya firma Gorodisskij i
Partnery"

(72) Inventor(s):

SCHIERL, Thomas (DE),
GEORGE, Valeri (DE),
HENKEL, Anastasia (DE),
MARPE, Detlev (DE),
GRUENEBERG, Karsten (DE),
SKUPIN, Robert (DE)

(73) Proprietor(s):

GE Video Compression, LLC (US)(54) **LOW DELAY PICTURE CODING**

(57) Abstract:

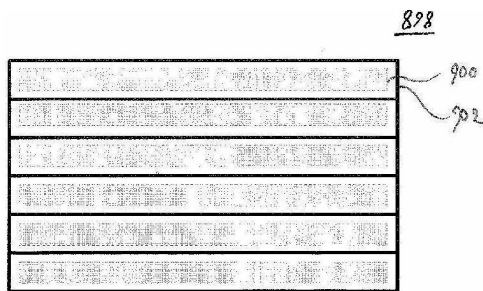
FIELD: physics.

SUBSTANCE: invention relates to the computer equipment. Decoder for recovering an image from a data stream into which an image is encoded in units of slices to which the image is divided using WPP processing, wherein the decoder is configured to decode the slices from the data stream in accordance with the slices order, wherein each slice has an initial syntax portion indicating the decoding start position of the corresponding slice in the image, and the decoder is

configured to: identify WPP sub-stream entry points, wherein slices are grouped, by identification, using initial sections of syntax of slices, slices starting on left side of image, parallel decoding of WPP sub-streams in stepwise manner with successive beginning of decoding.

EFFECT: technical result is high efficiency of encoding by reducing consumed coding costs.

10 cl, 32 dwg, 4 tbl



Фиг.1

RU 2710908 C2

RU 2710908 C2

Настоящее изобретение относится к кодированию изображений с малой задержкой.

В действующей схеме HEVC (высокоэффективное видеокодирование) в качестве инструментальных средств для распараллеливания содержатся слайсы, энтропийные слайсы (прежние облегченные слайсы), тайлы и волновая параллельная обработка (WPP).

Для распараллеливания видеокодеров и декодеров разбиение на уровне изображений имеет несколько преимуществ по сравнению с другими подходами. В предыдущих видеокодеках, подобных H.264/AVC [1], разбиения изображения были возможны только с регулярными слайсами с высокой ценой в отношении эффективности кодирования.

Для масштабируемого параллельного декодирования H.264/AVC необходимо объединять параллелизм на уровне макроблоков для восстановления изображения и параллелизм на уровне кадров для энтропийного декодирования. Этот подход, однако, обеспечивает ограниченное снижение латентности изображений и большое использование памяти. Чтобы преодолеть эти ограничения в кодек HEVC были включены новые стратегии разбиения изображения. Текущая эталонная версия программного обеспечения (HM-6) содержит 4 разных подхода: регулярные или нормальные слайсы, энтропийные слайсы, подпотоки волновой параллельной обработки (WPP) и тайлы (неперекрывающиеся изображения). Обычно эти разбиения изображения содержат набор наибольших единиц кодирования (LCU) или, в синонимической формулировке, единиц дерева кодирования (CTU), как определено в HEVC, или даже поднабор их.

Фиг.1 изображает изображение 898, расположенное в качестве примера в регулярном слайсе 900 на строку 902 LCU или макроблоков в видеокадре. Регулярные или нормальные слайсы (как определено в H.264 [1]) имеют наибольшее ухудшение кодирования, так как они нарушают зависимости энтропийного декодирования и предсказания.

Энтропийные слайсы, подобно слайсам, нарушают зависимости энтропийного декодирования, но делают возможным предсказание (и фильтрацию) для пересечения границ слайсов.

В WPP разбиения изображения перемежаются по строке, и предсказанию разрешено использовать данные из блоков в других разбиениях. Таким образом минимизируются потери кодирования, тогда как одновременно может применяться волновой параллелизм. Перемежение, однако, нарушает причинность битового потока, так как предшествующему разбиению необходимо следующее разбиение для декодирования.

Фиг.2, в качестве примера, изображает изображение 898, разделенное на две строки 904, 904b тайлов 906 горизонтального разбиения. Тайлы определяют горизонтальную 908 и вертикальную границы 910, которые разбивают изображение 898 на столбцы 912a, b, c и строки 904a, b тайлов. Подобно регулярным слайсам 900, тайлы 906 нарушают зависимости энтропийного декодирования и предсказания, но не требуют заголовка для каждого тайла.

Для каждого из этих методов количество разбиений может свободно выбираться кодером. Обычно, существование большего количества разбиений приводит к большим потерям сжатия. Однако при WPP распространение потерь не такое высокое, и поэтому количество разбиений изображения даже может быть фиксированным на один на строку. Это также приводит к нескольким преимуществам. Во-первых, для WPP гарантируется причинность битового потока. Во-вторых, реализации декодера могут принимать, что является доступной некоторая величина параллелизма, что также повышает разрешающую способность. И, наконец, ни одна из зависимостей выбора

контекста и предсказания не должна нарушаться при декодировании в волновом порядке, приводя к относительно низким потерям кодирования.

Однако до настоящего времени все принципы параллельного кодирования в преобразовании не обеспечивают достижение высокой эффективности сжатия в комбинации с сохранением малой задержки. Это также верно для принципа WPP. Слайсы представляют собой наименьшие единицы транспортировки в конвейере кодирования, и несколько подпотоков WPP все же должны транспортироваться последовательно.

Следовательно, задачей настоящего изобретения является обеспечение принципа кодирования изображений, который делает возможным параллельное декодирование согласно, например, волновой параллельной обработке с повышенной эффективностью, как например, с еще большим уменьшением сквозной задержки или повышением эффективности кодирования посредством уменьшения расходуемых издержек кодирования.

Эта задача достигается предметом независимых пунктов формулы изобретения.

Основным открытием настоящего изобретения является то, что принципы параллельной обработки, такие как волновая параллельная обработка, могут быть реализованы с уменьшенной сквозной задержкой, если отказаться от принципа обычных слайсов, согласно которому слайсы или кодируются/декодируются полностью независимо от областей изображения вне соответствующего слайса, или по меньшей мере независимо от областей вне соответствующего слайса, что касается энтропийного кодирования, в пользу слайсов разных видов, а именно слайсов, называемых зависимыми слайсами, которые принимают во внимание взаимозависимости через границы слайсов, и других слайсов, которые не принимают во внимание, называемых, например, нормальными слайсами.

Другим основным открытием настоящего изобретения, которое может быть объединено с первым или использоваться индивидуально, является то, что принцип обработки WPP может быть сделан более эффективным, если используются синтаксические участки начала слайсов для определения расположения точек входа WPP.

Предпочтительные варианты осуществления настоящей заявки описаны ниже в отношении фигур, причем полезные варианты осуществления являются объектом зависимых пунктов формулы изобретения. На фигурах:

фиг.1 изображает изображение, разбитое, в качестве примера, на регулярный слайс на строку LCU или макроблоков в изображении;

фиг.2 изображает изображение, разделенное, в качестве примера, на две строки горизонтально разбитых тайлов;

фиг.3 изображает, в качестве примера, назначение параллельно кодированных разбиений слайсу или сегменту сетевой транспортировки;

фиг.4 изображает схематическое представление, иллюстрирующее обобщенную фрагментацию кадра посредством подхода кодирования тайлов для минимальной сквозной задержки;

фиг.5 изображает схематическое представление, иллюстрирующее примерную фрагментацию кадра посредством подхода кодирования WPP для минимальной сквозной задержки;

фиг.6 изображает схематическую блок-схему, иллюстрирующую сценарий преобразования с использованием видеослуг;

фиг.7 иллюстрирует схематически возможное временное планирование кодирования, передачи и декодирования для тайлов с обобщенными поднаборами с минимальной

сквозной задержкой;

фиг.8 схематически изображает временное планирование, которое обычно достигает сквозной задержки;

фиг.9 иллюстрирует изображение, имеющее в качестве примера блоки дерева кодирования 11×9, которое разделено на два слайса;

фиг.10 иллюстрирует изображение, имеющее в качестве примера блоки дерева кодирования 13×8, которое разбито на три тайла;

фиг.11 изображает пример синтаксиса набора параметров последовательности;

фиг.12 изображает пример синтаксиса набора параметров изображения;

фиг.13 изображает пример синтаксиса заголовка слайса;

фиг.14 приводит в качестве примера разбиение изображения для обработки WPP на регулярный слайс и, для обработки с малой задержкой, на зависимые слайсы;

фиг.15 изображает пример для участка в синтаксисе набора параметров изображения;

фиг.16 изображает возможный синтаксис заголовка слайса;

фиг.17 схематически иллюстрирует взаимозависимости кодирования для нормальных слайсов (и зависимых слайсов);

фиг.18 изображает схематическое представление, сравнивающее кодирование для транспортировки с малой задержкой тайлов (волновая параллельная обработка, использующая зависимые слайсы);

фиг.19 иллюстрирует временное планирование, иллюстрирующее примерное кодирование WPP с конвейерной передачей с малой задержкой при использовании волновой параллельной обработки, использующей зависимые слайсы, как показано на правой стороне фиг.18;

фиг.20 изображает схематическое представление, иллюстрирующее улучшение робастности посредством использования регулярных слайсов в качестве привязок;

фиг.21 изображает другой вариант осуществления для синтаксиса заголовка слайса;

фиг.22 изображает другой вариант осуществления для синтаксиса набора параметров изображения;

фиг.23 изображает схематическое представление, иллюстрирующее процесс инициализации вероятностей символа для зависимого слайса в случае начала на левой границе изображения;

фиг.24 изображает схематическое представление декодера;

фиг.25 схематически изображает блок-схему декодера вместе со схематической иллюстрацией разбиения изображения на блоки кодирования и слайсы;

фиг.26 схематически изображает блок-схему кодера;

фиг.27 схематически изображает изображение, разбитое на нормальные и зависимые слайсы, названные здесь сегментами слайса;

фиг.28a и 28b схематически изображают изображение, разбитое на нормальные и зависимые слайсы, названные здесь сегментами слайса, с одной стороны, и тайлы, с другой стороны;

фиг.29 изображает блок-схему последовательности операций, иллюстрирующую процесс инициализации контекста, использующий зависимые слайсы;

фиг.30 изображает блок-схему последовательности операций, иллюстрирующую процесс сохранения контекста для использования зависимых слайсов; и

фиг.31 схематически изображает разные возможности сигнализации точек входа WPP.

В нижеследующем описание начинается с описания современных принципов предоставления возможности параллельной обработки изображений и кодирования с

малой задержкой соответственно. Кратко изложены проблемы, имеющие место при желании иметь обе возможности. В частности, как оказывается из последующего описания, принцип подпотоков WPP, как предлагается до настоящего времени, тем или иным образом вступает в противоречие с желанием иметь малую задержку из-за необходимости передавать подпотоки WPP посредством их группирования в один слайс. Нижеследующие варианты осуществления представляют принципы параллельной обработки, такие как принцип WPP, применяемые в применениях, требующих даже меньшую задержку посредством расширения принципа слайса, а именно введением другого типа слайса, позже названных зависимыми слайсами.

Минимизация сквозной задержки видео от захвата до отображения является одной из главных целей в приложениях, таких как видеоконференция и т.п.

Цепочка обработки сигнала для передачи цифрового видео состоит из камеры, устройства захвата, кодера, инкапсуляции, передачи, демультимплексора, декодера, рендера и дисплея. Каждый из этих этапов вносит свой вклад в сквозную задержку посредством буферизации данных изображения перед их последовательной передачей на последующий этап.

Некоторые приложения требуют минимизации такой задержки, например, удаленное манипулирование объектами в опасных зонах, без прямой видимости манипулируемого объекта, или минимально инвазивная хирургия. Даже короткая задержка может приводить к серьезным затруднениям надлежащего манипулирования или даже приводить к катастрофическим ошибкам.

Во многих случаях весь видеокادر буферизуется на этапе обработки, например, чтобы сделать возможной обработку интра-кадра. Некоторые этапы собирают данные, чтобы формировать пакеты, которые направляются на следующий этап. Обычно, существует нижняя граница для задержки, которая вытекает из требований к локальной обработке. Это анализируется для каждого индивидуального этапа более подробно ниже.

Обработка внутри камеры необязательно требует обработки сигнала интра-кадра, поэтому минимальная задержка определяется временем интегрирования датчика, которое ограничивается частотой кадров и некоторыми проектными решениями производителя аппаратных средств. Выходной сигнал камеры обычно связан с порядком сканирования, который обычно начинает обработку в верхнем левом углу, перемещается в верхний правый угол и продолжает построчно до нижнего правого угла. Следовательно, оно занимает длительность около одного кадра до тех пор, пока все данные не будут перенесены с датчика на выход камеры.

Устройство захвата может направлять данные камеры непосредственно после приема; однако оно обычно буферизует некоторые данные и генерирует пакеты, чтобы оптимизировать доступ к данным для памяти или запоминающего устройства. Кроме того, соединение между камерой/устройством захвата и памятью компьютера обычно ограничивает скорость передачи данных для направления захваченных данных изображения в память для дальнейшей обработки (кодирования). Обычно камеры подключаются посредством универсальной последовательной шины (USB) 2.0 или в скором времени USB 3.0, которая всегда включает в себя частичную транспортировку данных изображения на кодер. Это ограничивает возможность распараллеливания на стороне кодера в сценариях с предельной малой задержкой, т.е. кодер пытается начать кодирование как можно быстрее, когда данные становятся доступными от камеры, например, в порядке сканирования раstra от верха до низа изображения.

В кодере имеются некоторые степени свободы, которые допускают принять

компромиссное решение по эффективности кодирования, в отношении скорости передачи данных, необходимой для некоторой верности воспроизведения видео, для снижения задержки обработки.

Кодер использует данные, которые уже были посланы для предсказания изображения, подлежащего последующему кодированию. Обычно, разность между фактическим изображением и предсказанием может кодироваться с меньшим количеством битов, чем требовалось бы без предсказания. Это значения предсказания должны быть доступны на декодере, таким образом предсказание основывается на ранее декодированных участках этого же изображения (внутрикадровое предсказание) или на других изображениях (межкадровое предсказание), которые были обработаны ранее. Стандарты видеокодирования перед HEVC используют только часть изображения над этой же строкой или на ней, но слева - которые были ранее кодированы - для внутрикадрового предсказания, предсказания вектора движения и энтропийного кодирования (контекстно-адаптивное двоичное арифметическое кодирование (САВАС)).

В дополнение к оптимизации структуры предсказания может учитываться влияние параллельной обработки. Параллельная обработка требует идентификации областей изображения, которые могут быть обработаны независимо. По практическим причинам выбираются прилегающие области, такие как горизонтальные или вертикальные прямоугольники, которые часто называют «тайлами». В случае ограничений на малую задержку эти области должны предоставлять возможность выполнения параллелизованного кодирования данных, поступающих от устройства захвата в память, как можно скорее. Предполагая пересылку содержимого памяти сканирования раstra, имеют смысл вертикальные разбиения необработанных данных, чтобы начать кодирование немедленно. Внутри таких тайлов, которые делят изображение на вертикальные разбиения (ср. фигуру ниже), внутреннее предсказание, предсказание вектора движения и энтропийное кодирование (САВАС) могут приводить к приемлемой эффективности кодирования. Чтобы минимизировать задержку, только часть изображения, начиная сверху, переносится в память кадров кодера, и параллельная обработка должна начинаться в вертикальных тайлах.

Другим путем разрешения параллельной обработки является использование WPP в регулярном слайсе, который будет сравниваться с тайлами, «строкой» тайлов, включенных в единственный слайс. Данные в этом слайсе также могут кодироваться параллельным образом, используя подпотоки WPP, в пределах слайса. Разделение изображения на слайсы 900 и тайлы/подпотоки 914 WPP показано в виде примеров фиг.3/1.

Фиг.3, таким образом, изображает назначение параллельно кодированных разбиений, таких как 906 или 914, слайсу или сегменту сетевой транспортировки (единственный сетевой пакет или многочисленные сетевые 900 пакеты).

Инкапсуляция кодированных данных в единицы уровня сетевой абстракции (NAL), как определено в H.264 или HEVC, перед передачей или во время процесса кодирования добавляет некоторый заголовок к блокам данных, который допускает идентификацию каждого блока и переупорядочение блоков, если это применимо. В стандартном случае, не требуется дополнительная сигнализация, так как порядок элементов кодирования всегда находится в порядке декодирования, то есть определяется явное назначение положения тайла или общего фрагмента кодирования.

Если параллельная обработка рассматривается с дополнительным транспортным уровнем для параллельной транспортировки с малой задержкой, т.е. транспортный уровень может переупорядочивать разбиения изображения для тайлов, чтобы допускать

передачу с малой задержкой, подразумевая рассылку фрагментов, как показано на фиг.4, когда они кодируются. Эти фрагменты также могут быть не полностью кодированными слайсами, они могут быть поднабором слайса, или могут содержаться в зависимом слайсе.

5 В случае создания дополнительных фрагментов существует компромисс между эффективностью, которая является наибольшей с большими блоками данных, так как информация заголовка добавляет постоянное количество байтов, и задержкой, так как большие блоки данных параллельных кодеров необходимо буферизовать перед передачей. Общая задержка может быть уменьшена, если кодированное представление
10 вертикальных тайлов 906 разделяется на ряд фрагментов 916, которые передаются, как только будет полностью кодирован фрагмент. Размер каждого фрагмента может определяться на основе фиксированной области изображения, такой как макроблоки, LCU или на основе максимальных данных, как показано на фиг.4.

Фиг.4, таким образом, изображает обобщенную фрагментацию кадра посредством
15 подхода кодирования тайлов для минимальной сквозной задержки.

Аналогично, фиг.5 изображает фрагментацию кадра с подходом кодирования WPP для минимальной сквозной задержки.

Передача может добавлять дополнительную задержку, например, если применяется дополнительная ориентированная на блоки обработка, такая как коды прямой
20 коррекции ошибок, повышающие робастность передачи. Кроме того, сетевая инфраструктура (маршрутизаторы и т.д.) или физический уровень могут добавлять задержку, она обычно известна как латентность для соединения. В дополнение к латентности скорость передачи битов при передаче определяет время (задержку) для пересылки данных от Стороны а к Стороне b при разговоре, как показано на фиг.6,
25 который использует видеослуги.

Если кодированные блоки данных передаются вне порядка, необходимо учитывать задержку на переупорядочение. Декодирование может начинаться, как только поступит единица данных, предполагая, что являются доступными другие единицы данных, которые должны декодироваться перед этой.

30 В случае тайлов не существует зависимостей между тайлами, таким образом, тайл может декодироваться немедленно. Если были созданы фрагменты тайла, такие как отдельные слайсы на каждый фрагмент, как показано на фиг.4, фрагменты могут непосредственно транспортироваться, как только они будут кодированы соответственно тому, как были кодированы их содержащиеся LCU или единицы кодирования (CU).

35 Рендерер собирает выходные сигналы механизмов параллельного декодирования и направляет объединенное изображение построчно на дисплей.

Дисплей не добавляет обязательно никакой задержки, но на практике может выполнять некоторую внутрикадровую обработку перед тем, как данные изображения фактически отображаются. Это зависит от проектных решений производителя
40 аппаратных средств.

Суммируя, изобретатели могут оказывать влияние на этапы кодирования, инкапсуляции, передачи и декодирования, чтобы достичь минимальной сквозной задержки. Если изобретатели используют параллельную обработку, тайлы и фрагментацию в тайлах, общая задержка может быть значительно уменьшена, как
45 показано на фиг.7, по сравнению с обычно используемой цепочкой обработки, которая добавляет задержку около одного кадра на каждом из этих этапов, как показано на фиг.8.

В частности, в то время как фиг.7 изображает кодирование, передачу и декодирование

для тайлов с обобщенными поднаборами с минимальной сквозной задержкой, фиг.8 иллюстрирует обычно достигаемую сквозную задержку.

HEVC позволяет использовать разбиение слайса, разбиение тайла и следующим образом.

5 тайл: Целочисленное количество блоков дерева, совместно встречающихся в одном столбце и одной строке, упорядоченное последовательно в сканировании раstra блоков дерева тайла. Деление каждого изображения на тайлы представляет собой разбиение. Тайлы в изображении упорядочиваются последовательно в сканировании раstra тайла изображения. Хотя слайс содержит блоки дерева, которые являются последовательными
10 в сканировании раstra блоков дерева тайла, эти блоки дерева не являются обязательно последовательными в сканировании раstra блоков дерева изображения.

слайс: Целочисленное количество блоков дерева, упорядоченное последовательно в сканировании раstra. Деление каждого изображения на слайсы представляет собой разбиение. Адреса блоков дерева выводятся из адреса первого блока дерева в слайсе
15 (как представлено в заголовке слайса).

сканирование раstra: Отображение прямоугольного двумерного образца на одномерный образец, так что первые вводы в одномерном образце выполняются из первой верхней строки двумерного образца, сканируемого слева направо, за которым следуют аналогично вторая, третья и т.д. строки образца (идя вниз), причем каждая
20 сканируется слева направо.

блок дерева: блок $N \times N$ отсчетов яркости и два соответствующих блока отсчетов цветности изображения, которое имеет три массива отсчетов, или блок $N \times N$ отсчетов монохромного изображения или изображения, которое кодируется с использованием трех отдельных цветовых плоскостей. Деление слайса на блоки дерева представляет
25 собой разбиение.

Разбиение: Деление набора на поднаборы, так что каждый элемент набора находится точно в одном из поднаборов.

Квадродерево: Дерево, в котором родительский узел может быть разбит на четыре дочерних узла. Дочерний узел может стать родительским узлом для другого разбиения
30 на четыре дочерних узла.

В нижеследующем объясняется пространственное подразделение изображений, слайсов и тайлов. В частности, следующее описание задает то, как изображение разбивается на слайсы, тайлы и блоки дерева кодирования. Изображения делятся на слайсы и тайлы. Слайс представляет собой последовательность блоков дерева
35 кодирования. Аналогично, тайл представляет собой последовательность блоков дерева кодирования.

Отсчеты обрабатываются в единицах блоков дерева кодирования. Размером массива яркости для каждого блока дерева в отсчетах как по ширине, так и по высоте является CtbSize. Шириной и высотой массивов цветности для каждого блока дерева кодирования
40 являются CtbWidthC и CtbHeightC соответственно. Например, изображение может делиться на два слайса, как показано на следующей фигуре. В качестве другого примера, изображение может делиться на три тайла, как показано на второй последующей фигуре.

В отличие от слайсов, тайлы всегда являются прямоугольными и всегда содержат целочисленное количество блоков дерева кодирования в сканировании раstra блоков
45 дерева кодирования. Тайл может состоять из блоков дерева кодирования, содержащихся в более, чем одном слайсе. Аналогично, слайс может содержать блоки дерева кодирования, содержащиеся в более, чем одном тайле.

Фиг.9 иллюстрирует изображение 898 с 11 на 9 блоками 918 дерева кодирования,

которое разбивается на два слайса 900a, b.

Фиг.10 иллюстрирует изображение с 13 на 8 блоками 918 дерева кодирования, которое разбивается на три тайла.

Каждому блоку 918 дерева кодирования 898 назначается разбиение, сигнализирующее для идентификации размеров блока для внутреннего или внешнего предсказания и для кодирования с преобразованием. Разбиение представляет собой рекурсивное разбиение методом квадродерева. Корень квадродерева ассоциируется с блоком дерева кодирования. Квадродерево разделяется до тех пор, пока не будет достигнут лист, который упоминается как блок кодирования. Блок кодирования представляет собой корневой узел двух деревьев, дерева предсказания и дерева преобразования.

Дерево предсказания задает положение и размер блоков предсказания. Блоки предсказания и ассоциированные данные предсказания упоминаются как единица предсказания.

Фиг.11 изображает примерный синтаксис RBSP набора параметров последовательности.

Дерево преобразования задает положение и размер блоков преобразования. Блоки преобразования и ассоциированные данные преобразования упоминаются как единица преобразования.

Информация о разбиении для яркости и цветности является идентичной для дерева предсказания и может быть идентичной или может не быть идентичной для дерева преобразования.

Блок кодирования, ассоциированные данные кодирования и ассоциированные единицы предсказания и преобразования вместе образуют единицу кодирования.

Процесс для преобразования адреса блока дерева кодирования в порядке раstra блока дерева кодирования в порядок сканирования тайла может быть следующим:

Выходами данного процесса являются

- массив CtbAddrTS[ctbAddrRS] с ctbAddrRS в диапазоне от 0 до PicHeightInCtbs * PicWidthInCtbs-1 включительно.

- массив TileId[ctbAddrTS] с ctbAddrTS в диапазоне от 0 до PicHeightInCtbs * PicWidthInCtbs-1 включительно.

Массив CtbAddrTS[] выводится следующим образом:

```

for( ctbAddrRS = 0; ctbAddrRS < PicHeightInCtbs * PicWidthInCtbs; ctbAddrRS++ ) {
    tbX = ctbAddrRS % PicWidthInCtbs
    tbY = ctbAddrRS / PicWidthInCtbs
    for( j = 0; j <= num_tile_columns_minus1; j++ )
        if( tbX < ColBd[ j + 1 ] )
            tileX = j
    for( i = 0; i <= num_tile_rows_minus1; i++ )
        if( tbY < RowBd[ i + 1 ] )
            tileY = i
    CtbAddrTS[ ctbAddrRS ] = ctbAddrRS - tbX
    for( i = 0; i < tileX; i++ )
        ctbAddrTS += RowHeight[ tileY ] * ColumnWidth[ i ]
    CtbAddrTS[ ctbAddrRS ] += ( tbY - RowBd[ tileY ] ) * ColumnWidth[ tileY ] + tbX - ColBd[ tileX ]
}

```

Массив TileId[] выводится следующим образом:

```

for( j = 0, tileId = 0; j <= num_tile_columns_minus1; j++ )
  for( i = 0; i <= num_tile_rows_minus1; i++, tileId++ )
    for( y = RowBd[ j ]; y < RowBd[ j + 1 ]; y++ )
      for( x = ColBd[ i ]; x < ColBd[ i + 1 ]; x++ )
5         TileId[ CtbAddrTS[ y*PicWidthInCtbs + x ] ] = tileId

```

Соответствующий примерный синтаксис показан на фиг.11, 12 и 13, причем фиг.12 имеет примерный синтаксис RBSP набора параметров изображений. Фиг.13 изображает примерный синтаксис заголовка слайса.

В примере синтаксиса может применяться следующая семантика:

10 entropy_slice_flag равный 1 задает, что подразумевается, что значение неприсутствующих синтаксических элементов заголовка слайса равно значению синтаксических элементов заголовка слайса в предшествующем слайсе, где предшествующий слайс определяется как слайс, содержащий блок дерева кодирования с расположением (SliceCtbAddrRS-1). entropy_slice_flag должен быть равен 0, когда

15 SliceCtbAddrRS равен 0.
tiles_or_entropy_coding_sync_idc равный 0 задает, что существует только один тайл в каждом изображении в кодированной видеопоследовательности, и не вызывается конкретный процесс синхронизации для контекстных переменных перед декодированием первого блока дерева кодирования строки блоков дерева кодирования.

20 tiles_or_entropy_coding_sync_idc равный 1 задает, что может быть более одного тайла в каждом изображении в кодированной видеопоследовательности, и не вызывается конкретный процесс синхронизации для контекстных переменных перед декодированием первого блока дерева кодирования строки блоков дерева кодирования.

25 tiles_or_entropy_coding_sync_idc равный 2 задает, что существует только один тайл в каждом изображении в кодированной видеопоследовательности, конкретный процесс синхронизации для контекстных переменных вызывается перед декодированием первого блока дерева кодирования строки блоков дерева кодирования, и конкретный процесс запоминания для контекстных переменных вызывается после декодирования двух блоков дерева кодирования строки блоков дерева кодирования.

30 Значение tiles_or_entropy_coding_sync_idc должно быть в диапазоне от 0 до 2 включительно.

num_tile_columns_minus1 плюс 1 задает количество столбцов тайла, разбивающих изображение.

35 num_tile_rows_minus1 плюс 1 задает количество строк тайла, разбивающих изображение.

Когда num_tile_columns_minus1 равен 0, num_tile_rows_minus1 не должен быть равен 0.

Одно или оба из следующих условий должны выполняться для каждого слайса и тайла:

- 40 - Все кодированные блоки в слайсе принадлежат одному и тому же тайлу.
- Все кодированные блоки в тайле принадлежат одному и тому же слайсу.

ПРИМЕЧАНИЕ - В пределах одного и того же изображения могут быть как слайсы, которые содержат многочисленные тайлы, так и тайлы, которые содержат многочисленные слайсы.

45 uniform_spacing_flag равный 1 задает, что границы столбцов и, аналогично, границы строк распределяются равномерно по изображению. uniform_spacing_flag равный 0 задает, что границы столбцов и, аналогично, границы строк не распределяются равномерно по изображению, но сигнализируются однозначно, используя

синтаксические элементы `column_width[i]` и `row_height[i]`.

`column_width[i]` задает ширину *i*-ого столбца тайла в единицах блоков дерева кодирования.

`row_height[i]` задает высоту *i*-ой строки тайла в единицах блоков дерева кодирования.

5 Значения `ColumnWidth[i]`, задающие ширину *i*-ого столбца тайла в единицах блоков дерева кодирования, и значения `ColumnWidthInLumaSamples[i]`, задающие ширину *i*-ого столбца тайла в единицах отсчетов яркости, выводятся следующим образом:

```

    for( i = 0; i <= num_tile_columns_minus1; i++ ) {
        if( uniform_spacing_flag )
10         ColumnWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbs ) / ( num_tile_columns_minus1 + 1 ) -
            ( i * PicWidthInCtbs ) / ( num_tile_columns_minus1 + 1 )
        else
            ColumnWidth[ i ] = column_width[ i ]
            ColumnWidthInLumaSamples[ i ] = ColumnWidth[ i ] << Log2CtbSize
    }

```

15 Значения `RowHeight[i]`, задающие высоту *i*-ой строки тайла в единицах блоков дерева кодирования, выводятся следующим образом:

```

    for( i = 0; i <= num_tile_rows_minus1; i++ )
        if( uniform_spacing_flag )
20         RowHeight[ i ] = ( ( i + 1 ) * PicHeightInCtbs ) / ( num_tile_rows_minus1 + 1 ) -
            ( i * PicHeightInCtbs ) / ( num_tile_rows_minus1 + 1 )
        else
            RowHeight[ i ] = row_height[ i ]

```

Значения `ColBd[i]`, задающие расположение левой границы столбца *i*-ого столбца тайла в единицах блоков дерева кодирования, выводятся следующим образом:

```

    for( ColBd[ 0 ] = 0, i = 0; i <= num_tile_columns_minus1; i++ )
        ColBd[ i + 1 ] = ColBd[ i ] + ColumnWidth[ i ]

```

Значения `RowBd[i]`, задающие расположение верхней границы строки *i*-ой строки тайла в единицах блоков дерева кодирования, выводятся следующим образом:

```

30     for( RowBd[ 0 ] = 0, i = 0; i <= num_tile_rows_minus1; i++ )
        RowBd[ i + 1 ] = RowBd[ i ] + RowHeight[ i ]

```

`num_substreams_minus1` плюс 1 задает максимальное количество поднаборов, включенных в слайс, когда `tiles_or_entropy_coding_sync_idc` равен 2. Когда он не присутствует, делается заключение, что значение `num_substreams_minus1` равно 0.

35 `num_entry_point_offsets` задает количество синтаксических элементов `entry_point_offset[i]` в заголовке слайса. Когда `tiles_or_entropy_coding_sync_idc` равен 1, значение `num_entry_point_offsets` должно быть в диапазоне от 0 до $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1) - 1$ включительно. Когда `tiles_or_entropy_coding_sync_idc` равен 2, значение `num_entry_point_offsets` должно быть в диапазоне от 0 до `num_substreams_minus1` включительно. Когда не присутствует, подразумевается, что значение `num_entry_point_offsets` равно 0.

`offset_len_minus1` плюс 1 задает длину в битах синтаксических элементов `entry_point_offset[i]`.

45 `entry_point_offset[i]` задает смещение *i*-ой точки входа в байтах и должно быть представлено посредством `offset_len_minus1` плюс 1 битов. Кодированная единица NAL слайса состоит из `num_entry_point_offsets` + 1 поднаборов, причем значения индекса поднабора находятся в диапазоне от 0 до `num_entry_point_offsets` включительно. Поднабор 0 состоит из байтов от 0 до `entry_point_offset[0]` - 1 включительно,

кодированной единицы NAL слайса, поднабор k , с k в диапазоне от 1 до $\text{num_entry_point_offsets} - 1$ включительно, состоит из байтов от $\text{entry_point_offset}[k-1]$ до $\text{entry_point_offset}[k] + \text{entry_point_offset}[k-1] - 1$ включительно, кодированной единицы NAL слайса, и последний поднабор (с индексом поднабора равным $\text{num_entry_point_offsets}$) состоит из оставшихся байтов кодированной единицы NAL слайса.

ПРИМЕЧАНИЕ - Заголовок единицы NAL и заголовок слайса кодированной единицы NAL слайса всегда включены в поднабор 0.

Когда $\text{tiles_or_entropy_coding_sync_idc}$ равен 1 и $\text{num_entry_point_offsets}$ больше 0, каждый поднабор должен содержать все кодированные биты одного или многих завершенных тайлов, и количество поднаборов должно быть равно или меньше количеству тайлов в слайсе.

Когда $\text{tiles_or_entropy_coding_sync_idc}$ равен 2 и $\text{num_entry_point_offsets}$ больше 0, поднабор k для каждого из всех возможных значений k должен содержать все биты, подлежащие использованию во время процесса инициализации для текущего указателя k битового потока.

Что касается семантики данных слайса, то может применяться следующее.

end_of_slice_flag равный 0 задает, что следует другой макроблок в слайсе.

end_of_slice_flag равный 1 задает окончание слайса и что не следует другой макроблок.

$\text{entry_point_marker_two_3bytes}$ представляет собой последовательность фиксированного значения из 3 байтов равную $0x000002$. Этот синтаксический элемент называется префиксом маркера входа.

tile_idx_minus_1 задает TileID в порядке сканирования раstra. Первый тайл в изображении должен иметь TileID равный 0. Значение tile_idx_minus_1 должно быть в диапазоне от 0 до $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1) - 1$.

Процесс синтаксического анализа CABAC для данных слайса может быть следующим:

Этот процесс вызывается при синтаксическом анализе синтаксических элементов с дескриптором $\text{ae}(v)$.

Входами для этого процесса являются запрос значения синтаксического элемента и значения синтаксически анализированных синтаксических элементов.

Выходом этого процесса является значение синтаксического элемента.

В начале синтаксического анализа данных слайса в слайсе вызывается процесс инициализации процесса синтаксического анализа CABAC. Когда $\text{tiles_or_entropy_coding_sync_idc}$ равен 2 и $\text{num_substreams_minus1}$ больше 0, таблица отображения BitStreamTable с $\text{num_substreams_minus1} + 1$ элементами, задающими таблицу указателей битового потока для использования для последующего вывода текущего указателя битового потока, выводится следующим образом.

- BitStreamTable[0] инициализируется на содержание указателя битового потока.

- Для всех индексов i больше 0 и меньше $\text{num_substreams_minus1} + 1$, BitStreamTable[i] содержит указатель битового потока на $\text{entry_point_offset}[i]$ байтов после BitStreamTable[i-1].

Текущий указатель битового потока устанавливается на BitStreamTable[0].

Адрес минимального блока кодирования блока дерева кодирования, содержащего пространственный соседний блок T, ctbMinCbAddrT , выводится с использованием расположения $(x0, y0)$ верхнего левого отсчета яркости текущего блока дерева кодирования, например, следующим образом.

$x = x0 + 2 \ll \text{Log2CtbSize} - 1$

$y = y0 - 1$

$\text{ctbMinCbAddrT} = \text{MinCbAddrZS}[x \gg \text{Log2MinCbSize}][y \gg \text{Log2MinCbSize}]$

Переменная `availableFlagT` получается посредством вызова соответствующего процесса вывода доступности блока кодирования с `ctbMinCbAddrT` в качестве входа.

В начале синтаксического анализа дерева кодирования и `tiles_or_entropy_coding_sync_idc` равным 2, и `num_substreams_minus1` больше 0, применяется следующее.

- Если `CtbAddrRS % PicWidthInCtbs` равняется 0, применяется следующее.

- Когда `availableFlagT` равен 1, вызывается процесс синхронизации процесса синтаксического анализа CABAC, как определено в подпункте «Процесс синхронизации для контекстных переменных».

- Вызывается процесс декодирования для двоичных решений перед завершением, за которым следует процесс инициализации для механизма арифметического декодирования.

- Текущий указатель битового потока устанавливается на указание `BitStreamTable[i]` с индексом `i`, выводимым следующим образом.

$i = (\text{CtbAddrRS} / \text{PicWidthInCtbs}) \% (\text{num_substreams_minus1} + 1)$

- Иначе, если `CtbAddrRS % PicWidthInCtbs` равен 2, процесс запоминания процесса синтаксического анализа CABAC вызывается так, как задается в подпункте «Процесс запоминания для контекстных переменных».

Процесс инициализации может быть следующим:

Выходы этого процесса инициализируются внутренними переменными CABAC.

Его специальные процессы вызываются при начале синтаксического анализа данных слайса в слайсе, или при начале синтаксического анализа данных дерева кодирования, и деревом кодирования является первое дерево кодирования в тайле.

Процесс запоминания для контекстных переменных может быть следующим:

Входами этого процесса являются контекстные переменные CABAC, индексированные посредством `ctxIdx`.

Выходом этого процесса являются переменные `TableStateSync` и `TableMPSSync`, содержащие значения переменных `m` и `n`, используемых в процессе инициализации контекстных переменных, которые назначаются синтаксическим элементам за исключением флага окончания слайса.

Для каждой контекстной переменной соответствующие элементы `n` и `m` таблиц `TableStateSync` и `TableMPSSync` инициализируются до соответствующих `pStateIdx` и `valMPS`.

Процесс синхронизации для контекстных переменных может быть следующим:

Входами этого процесса являются переменные `TableStateSync` и `TableMPSSync`, содержащие значения переменных `n` и `m`, используемых в процессе запоминания контекстных переменных, которые назначаются синтаксическим элементам за исключением флага окончания слайса.

Выходами этого процесса являются контекстные переменные CABAC, индексированные посредством `ctxIdx`.

Для каждой контекстной переменной соответствующие контекстные переменные `pStateIdx` и `valMPS` инициализируются до соответствующих элементов `n` и `m` таблиц `TableStateSync` и `TableMPSSync`.

Ниже объясняется кодирование и транспортировка с малой задержкой с использование WPP. В частности, описание ниже показывает, как транспортировка с малой задержкой, как описано на фиг.7, также может быть применена к WPP.

Сначала, важно, что может быть послан поднабор изображения, перед завершением всего изображения. Нормально, что это достигается с использованием слайсов, как уже показано на фиг.5.

Чтобы уменьшить задержку по сравнению с тайлами, как показано на следующих фигурах, необходимо применить единственный подпоток WPP на строку LCU и дополнительно предоставить возможность отдельной передачи каждой из этих строк. Чтобы сохранить высокой эффективностью кодирования, не могут использоваться слайсы на каждую строку/подпоток. Поэтому, ниже вводятся так называемый зависимый слайс, определенный в следующем разделе. Этот слайс, например, не имеет все поля полного заголовка слайса HEVC, но поля, используемые для энтропийных слайсов. Кроме того, может быть переключатель для отключения разрыва CABAC между строками. В случае WPP, использование контекста CABAC (стрелки на фиг.14) и предсказание строк должно быть разрешено, чтобы сохранить выигрыш в эффективности кодирования WPP над тайлами.

В частности, фиг.14 приводит в качестве примера изображение 10 для WPP в регулярный слайс 900 (reg. SL), и для обработки с малой задержкой в зависимые слайсы (OS) 920.

В настоящее время предстоящий стандарт HEVC предлагает два типа разбиения в отношении слайсов. Имеется регулярный (нормальный) слайс и энтропийный слайс. Регулярный слайс представляет собой полностью независимое разбиение изображения за исключением некоторых зависимостей, которые могут быть доступны из-за процесса деблокирующего фильтра на границах слайса. Энтропийный слайс также является независимым, но только в отношении энтропийного кодирования. Идеей фиг.14 является обобщение принципа формирования слайсов. Таким образом, предстоящий стандарт HEVC должен предлагать два общих типа слайсов: независимый (регулярный) или зависимый. Поэтому, вводится новый тип слайса, зависимый слайс.

Зависимый слайс представляет собой слайс, который имеет зависимости с предыдущим слайсом. Зависимости представляют собой конкретные данные, которые могут использоваться между слайсами в процессе энтропийного декодирования и/или в процессе восстановления пикселей.

На фиг.14 в качестве примера представлен принцип зависимых слайсов. Изображение начинается, например, всегда с регулярного слайса. Отметим, что в данном принципе незначительно изменено поведение регулярного слайса. Обычно, в стандартах, подобных H.264/AVC или HEVC регулярный слайс представляет собой полностью независимое разбиение и нет необходимости сохранять никакие данные после декодирования за исключением некоторых данных для процесса деблокирующего фильтра. Но обработка предстоящего зависимого слайса 920 возможна только посредством ссылки на данные слайса выше, в данном случае в первой строке: регулярного слайса 900. Для установления этого регулярные слайсы 900 должны сохранять данные последней CU-строки.

Эти данные содержат:

- данные механизма кодирования CABAC (состояния контекстной модели одной CU, из которой может инициализироваться процесс энтропийного декодирования зависимого слайса),
- все декодируемые синтаксические элементы CU для регулярного процесса декодирования CABAC зависимых CU,
- данные внутреннего предсказания и предсказания вектора движения.

Следовательно, каждый зависимый слайс 920 должен выполнять одну и ту же

процедуру - сохранять данные для предстоящего зависимого слайса в этом же изображении.

На практике эти дополнительные этапы не должны быть проблемой, так как процесс декодирования, обычно, всегда вынужден сохранять некоторые данные, подобные синтаксическим элементам.

В разделах ниже представлены возможные изменения для синтаксиса стандарта HEVC, которые необходимы, чтобы сделать возможным принцип зависимых слайсов.

Фиг.5, например, иллюстрирует возможные изменения в синтаксисе RBSP набора параметров изображения.

Семантика набора параметров изображения для зависимых слайсов может быть следующей:

dependent_slices_present_flag равный 1 задает, что изображение содержит зависимые слайсы, и процесс декодирования каждого (регулярного или зависимого) слайса должен сохранять состояния энтропийного декодирования и данные внутреннего предсказания и предсказания вектора движения для следующего слайса, который может быть зависимым слайсом, который также может следовать за регулярным слайсом. Следующий зависимый слайс может ссылаться на эти сохраненные данные.

Фиг.16 изображает возможный синтаксис **slice_header** с изменениями относительно текущего состояния HEVC.

dependent_slice_flag равный 1 задает, что подразумевается, что значение не присутствующих синтаксических элементов заголовка слайса равно значению синтаксических элементов заголовка слайса в предшествующем (регулярном) слайсе, где предшествующий слайс определяется как слайс, содержащий блок дерева кодирования с расположением (**SliceCtbAddrRS** - 1). **dependent_slice_flag** должен быть равен 0, когда **SliceCtbAddrRS** равен 0.

no_cabac_reset_flag равный 1 задает инициализацию CABAC из сохраненного состояния ранее декодированного слайса (и без первоначальных значений). Иначе, т.е. если равен 0, инициализация CABAC независима от любого состояния ранее декодированного слайса, т.е. с первоначальными значениями.

last_ctb_cabac_init_flag равный 1 задает инициализацию CABAC из сохраненного состояния последнего кодированного блока дерева ранее декодированного слайса (например, для тайлов всегда равных 1). Иначе, (равный 0) на данные инициализации ссылаются из сохраненного состояния второго кодированного блока дерева последнего (соседнего) ctb-row ранее декодированного слайса, если первым кодированным блоком дерева текущего слайса является первый кодированный блок дерева в строке (т.е. режим WPP), иначе инициализация CABAC выполняется из сохраненного состояния последнего кодированного блока дерева ранее декодированного слайса.

Ниже представлено сравнение зависимых слайсов и других схем (информационных) разбиения.

На фиг.17 показана разность между нормальным и зависимым слайсами.

Возможное кодирование и передача подпотоков WPP в зависимых слайсах (DS), как показано в отношении фиг.18, сравнивает кодирование для транспортировки с малой задержкой тайлов (слева) и WPP/DS (справа). Жирные непрерывно начерченные кресты на фиг.18 изображают один и тот же момент времени для двух способов, предполагая, что кодирование строки WPP занимает такое же время, что и кодирование единственного тайла. Вследствие зависимостей кодирования только первая строка WPP готова, после того как все тайлы были кодированы. Но используя подход зависимого слайса позволяет подходу WPP отсылать первую строку, если она кодирована. Это отличается от более

ранних назначений подпотока WPP, «подпоток» определяется для WPP как сцепление строк CU слайса для декодирования WPP одним и тем же потоком декодера, т.е. одним и тем же ядром/процессором. Хотя подпоток на строку и на энтропийный слайс также был возможен ранее, энтропийный слайс нарушает зависимости энтропийного кодирования и имеет, поэтому, более низкую эффективность кодирования, т.е. теряется выигрыш эффективности WPP.

Дополнительно, разность задержек между обоими подходами может быть реально низкой, предполагая передачу, как показано на фиг.19. В частности, фиг.19 изображает кодирование WPP с конвейерной передачей с малой задержкой.

Предполагая, что кодирование последних двух CU из DS #1.1 в подходе WPP на фиг. 18 не занимает больше времени, чем передача первой строки SL#1, нет различия между тайлами и WPP в случае малой задержки. Но эффективность кодирования WP/DS превосходит принцип тайлов.

Чтобы повысить робастность для режима WPP с малой задержкой, фиг.20 иллюстрирует, что повышение робастности достигается посредством использования регулярных слайсов (RS) в качестве привязок. В изображении, показанном на фиг.20, за (регулярным) слайсом (RS) следуют зависимые слайсы (DS). В данном случае, (регулярный) слайс действует в качестве привязки для нарушения зависимостей для предшествующих слайсов, следовательно, обеспечивается большая робастность в такой точке вставки (регулярного) слайса. В принципе, это не отличается от вставки (регулярного) слайса в любом случае.

Принцип зависимых слайсов также может быть реализован следующим образом.

В данном случае, фиг.21 изображает возможный синтаксис заголовка слайса.

Семантика заголовка слайса является следующей:

dependent_slice_flag равный 1 задает, что подразумевается, что значение каждого не присутствующего синтаксического элемента заголовка слайса равно значению соответствующего синтаксического элемента заголовка слайса в предшествующем слайсе, содержащем блок дерева кодирования, для которого адресом блока дерева кодирования является $\text{SliceCtbAddrRS} - 1$. Когда не присутствует, подразумевается, что значение **dependent_slice_flag** равно 0. Значение **dependent_slice_flag** должно быть равно 0, когда SliceCtbAddrRS равно 0.

slice_address задает адрес в разрешении степени детализации слайса, в котором начинается слайс. Длина синтаксического элемента **slice_address** равна $(\text{Ceil}(\text{Log2}(\text{PicWidthInCtbs} * \text{PicHeightInCtbs})) + \text{SliceGranularity})$ битам.

Переменная SliceCtbAddrRS , задающая блок дерева кодирования, в котором начинается слайс, в порядке сканирования раstra блока дерева кодирования, выводится следующим образом.

$\text{SliceCtbAddrRS} = (\text{slice_address} \gg \text{SliceGranularity})$

Переменная SliceCbAddrZS , задающая адрес первого блока кодирования в слайсе с минимальной степенью детализации блока кодирования в порядке z-образного сканирования, выводится следующим образом.

$\text{SliceCbAddrZS} = \text{slice_address}$

$\ll ((\log2_diff_max_min_coding_block_size - \text{SliceGranularity}) \ll 1)$

Декодирование слайса начинается с наибольшей возможной единицы кодирования, или, иначе, CTU, на координате начала слайса.

first_slice_in_pic_flag указывает, является ли слайс первым слайсом изображения. Если **first_slice_in_pic_flag** равен 1, переменные SliceCbAddrZS и SliceCtbAddrRS обе устанавливаются на 0, и декодирование начинается с первого блока дерева кодирования

в изображении.

pic_parameter_set_id задает используемый набор параметров изображения. Значение **pic_parameter_set_id** должно быть в диапазоне от 0 до 255 включительно.

num_entry_point_offsets задает количество синтаксических элементов **entry_point_offset** [i] в заголовке слайса. Когда **tiles_or_entropy_coding_sync_idc** равен 1, значение **num_entry_point_offsets** должно быть в диапазоне от 0 до $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1) - 1$ включительно. Когда **tiles_or_entropy_coding_sync_idc** равен 2, значение **num_entry_point_offsets** должно быть в диапазоне от 0 до **PicHeightInCtbs** - 1 включительно. Когда не присутствует, подразумевается, что значение **num_entry_point_offsets** равно 0.

offset_len_minus1 плюс 1 задает длину в битах синтаксических элементов **entry_point_offset[i]**.

entry_point_offset[i] задает i-ое смещение точки входа в байтах и должно быть представлено посредством **offset_len_minus1** плюс 1 битов. Кодированные данные слайса после заголовка слайса состоят из **num_entry_point_offsets** + 1 поднаборов, причем значения индекса поднабора находятся в диапазоне от 0 до **num_entry_point_offsets** включительно. Поднабор 0 состоит из байтов 0 - **entry_point_offset[0]** - 1 включительно кодированных данных слайса, поднабор k, причем k в диапазоне от 1 до **num_entry_point_offsets** - 1 включительно, состоит из байтов с **entry_point_offset[k-1]** до **entry_point_offset[k] + entry_point_offset[k-1] - 1** включительно, кодированных данных слайса, и последний поднабор (с индексом поднабора равным **num_entry_point_offsets**) состоит из оставшихся байтов кодированных данных слайса.

Когда **tiles_or_entropy_coding_sync_idc** равен 1, и **num_entry_point_offsets** больше 0, каждый поднабор должен содержать все кодированные биты точно одного тайла, и количество поднаборов (т.е. значение **num_entry_point_offsets** + 1) должно быть равно или меньше количества тайлов в слайсе.

ПРИМЕЧАНИЕ - Когда **tiles_or_entropy_coding_sync_idc** равен 1, каждый слайс должен включать в себя или поднабор из одного тайла (в этом случае является необязательной сигнализацией точек входа) или целочисленное количество полных тайлов.

Когда **tiles_or_entropy_coding_sync_idc** равен 2, и **num_entry_point_offsets** больше 0, каждый поднабор k с k в диапазоне от 0 до **num_entry_point_offsets** - 1 включительно, должен включать в себя все кодированные биты точно одной строки блоков дерева кодирования, последний поднабор (с индексом поднабора равным **num_entry_point_offsets**) должен содержать все кодированные биты остальных блоков кодирования, включенных в слайс, в котором остальные блоки кодирования состоят или из точно одной строки блоков дерева кодирования, или из поднабора одной строки блоков дерева кодирования, и количество поднаборов (т.е. значение **num_entry_point_offsets** + 1) должно быть равно количеству строк блоков дерева кодирования в слайсе, в котором также подсчитывается поднабор из одной строки блоков дерева кодирования в слайсе.

ПРИМЕЧАНИЕ Когда **tiles_or_entropy_coding_sync_idc** равен 2, слайс может включать в себя количество строк блоков дерева кодирования и поднабор строки блоков дерева кодирования. Например, если слайс включает в себя две с половиной строки блоков дерева кодирования, количество поднаборов (т.е. значение **num_entry_point_offsets** + 1) должно быть равно 3.

Соответствующий синтаксис RBSP набора параметров изображения может выбираться так, как показано на фиг.22.

Семантика Rbsp набора параметров изображения может быть следующей:

dependent_slice_enabled_flag равный 1 задает присутствие синтаксического элемента **dependent_slice_flag** в заголовке слайса для кодируемых изображений, ссылающихся на набор параметров изображения. **dependent_slice_enabled_flag** равный 0 задает отсутствие синтаксического элемента **dependent_slice_flag** в заголовке слайса для кодируемых изображений, ссылающихся на набор параметров изображения. Когда **tiles_or_entropy_coding_sync_idc** равен 3, значение **dependent_slice_enabled_flag** должно быть равно 1.

tiles_or_entropy_coding_sync_idc равный 0 задает, что должен быть только один тайл в каждом изображении, ссылающемся на набор параметров изображения, не должно быть конкретного процесса синхронизации для контекстных переменных, вызываемых перед декодированием первого блока дерева кодирования строки блоков дерева кодирования в каждом видеокadre, ссылающемся на набор параметров изображения, и значения **cabac_independent_flag** и **dependent_slice_flag** для кодируемых изображений, ссылающихся на набор параметров изображения, не должны быть оба равны 1.

Отметьте, что, когда **cabac_independent_flag** и **dependent_slice_flag** оба равны 1 для слайса, слайсом является энтропийный слайс.

tiles_or_entropy_coding_sync_idc равный 1 задает, что может быть более одного тайла в каждом изображении, ссылающемся на набор параметров изображения, не должно быть конкретного процесса синхронизации для контекстных переменных, вызываемых перед декодированием первого блока дерева кодирования строки блоков дерева кодирования в каждом изображении, ссылающемся на набор параметров изображения, и значения **cabac_independent_flag** и **dependent_slice_flag** для кодируемых изображений, ссылающихся на набор параметров изображения, не должны оба быть равны 1.

tiles_or_entropy_coding_sync_idc равный 2 задает, что должен быть только один тайл в каждом изображении, ссылающемся на набор параметров изображения, конкретный процесс синхронизации для контекстных переменных должен вызываться перед декодированием первого блока дерева кодирования строки блоков дерева кодирования в каждом изображении, ссылающемся на набор параметров изображения, и конкретный процесс запоминания для контекстных переменных должен вызываться после декодирования двух блоков дерева кодирования строки блоков дерева кодирования в каждом изображении, ссылающемся на набор параметров изображения, и значения **cabac_independent_flag** и **dependent_slice_flag** для кодируемых изображений, ссылающихся на набор параметров изображения, не должны быть оба равными 1.

tiles_or_entropy_coding_sync_idc равный 3 задает, что должен быть только один тайл в каждом изображении, ссылающемся на набор параметров изображения, не должно быть конкретного процесса синхронизации для контекстных переменных, вызываемого перед декодированием первого блока дерева кодирования строки блоков дерева кодирования в каждом изображении, ссылающемся на набор параметров изображения, и значения **cabac_independent_flag** и **dependent_slice_flag** для кодируемых изображений, ссылающихся на набор параметров изображения, оба могут быть равны 1.

Когда **dependent_slice_enabled_flag** должен быть равен 0, **tiles_or_entropy_coding_sync_idc** не должен быть равен 3.

Требованием соответствия битовых потоков является то, что значение **tiles_or_entropy_coding_sync_idc** должно быть одинаковым для всех наборов параметров изображения, которые активизируются в кодированной видеопоследовательности.

Для каждого слайса, ссылающегося на набор параметров изображения, когда **tiles_or_entropy_coding_sync_idc** равен 2, и первым блоком кодирования в слайсе не является первый блок кодирования в первом блоке дерева кодирования строки блоков дерева

кодирования, последний блок кодирования в слайсе должен принадлежать этой же строке блоков дерева кодирования, что и первый блок кодирования в слайсе.

num_tile_columns_minus1 плюс 1 задает количество столбцов тайла, разбивающих изображение.

5 **num_tile_rows_minus1** плюс 1 задает количество строк тайла, разбивающих изображение.

Когда **num_tile_columns_minus1** равен 0, **num_tile_rows_minus1** не должен быть равен 0. **uniform_spacing_flag** равный 1 задает, что границы столбцов и, аналогично, границы строк распределяются равномерно по изображению, **uniform_spacing_flag** равный 0
10 задает, что границы столбцов и, аналогично, границы строк не распределяются равномерно по изображению, но сигнализируются явно с использованием синтаксических элементов **column_width[i]** и **row_height[i]**.

column_width[i] задает ширину *i*-ого столбца тайла в единицах блоков дерева кодирования.

15 **row_height[i]** задает высоту *i*-ой строки тайла в единицах блоков дерева кодирования.

Вектор **colWidth[i]** задает ширину *i*-ого столбца тайла в единицах блока дерева кодирования (CTB), при этом столбец *i* находится в диапазоне от 0 до **num_tile_columns_minus1** включительно.

Вектор **CtbAddrRStoTS[ctbAddrRS]** задает диалоговый режим от адреса CTB в порядке
20 сканирования раstra к адресу CTB в порядке сканирования тайла, при этом индекс **ctbAddrRS** находится в диапазоне от 0 до $(\text{picHeightInCtbs} * \text{picWidthInCtbs}) - 1$ включительно.

Вектор **CtbAddrTStoRS[ctbAddrTS]** задает диалоговый режим от адреса CTB в порядке
25 сканирования тайла к адресу CTB в порядке сканирования раstra, при этом индекс **ctbAddrRS** находится в диапазоне от 0 до $(\text{picHeightInCtbs} * \text{picWidthInCtbs}) - 1$ включительно.

Вектор **TileId[ctbAddrTS]** задает диалоговый режим от адреса CTB в порядке сканирования тайла к *id* тайла, при этом **ctbAddrTS** находится в диапазоне от 0 до $(\text{picHeightInCtbs} * \text{picWidthInCtbs}) - 1$ включительно.

30 Значения **colWidth**, **CtbAddrRStoTS**, **CtbAddrTStoRS** и **TileId** выводятся посредством вызова процесса диалогового режима сканирования раstra и тайла CTB с **PicHeightInCtbs** и **PicWidthInCtbs** в качестве входов, и выход назначается **colWidth**, **CtbAddrRStoTS** и **TileId**.

Значения **ColumnWidthInLumaSamples[i]**, задающие ширину *i*-ого столбца тайла в
35 единицах отсчетов яркости, устанавливаются равными $\text{colWidth}[i] \ll \text{Log2CtbSize}$.

Массив **MinCbAddrZS[x][y]**, задающий диалоговый режим от расположения (*x*,*y*) в единицах минимальных блоков кодирования (CB) к адресу минимального CB в порядке *z*-образного сканирования, при этом *x* находится в диапазоне от 0 до **picWidthInMinCbs** - 1 включительно, и *y* находится в диапазоне от 0 до **picHeightInMinCbs** - 1 включительно,
40 выводится посредством вызова процесса инициализации массива порядка *Z*-образного сканирования с **Log2MinCbSize**, **Log2CtbSize**, **PicHeightInCtbs**, **PicWidthInCtbs** и вектором **CtbAddrRStoTS** в качестве входов, и выход назначается **MinCbAddrZS**.

loop_filter_across_tiles_enabled_flag равный 1 задает, что операции петлевой фильтрации выполняются через границы тайлов. **loop_filter_across_tiles_enabled_flag** равный 0 задает,
45 что операции петлевой фильтрации не выполняются через границы тайлов. Операции петлевой фильтрации включают в себя операции деблокирующего фильтра, адаптивного смещения отсчетов и адаптивного петлевого фильтра. Когда не присутствует, подразумевается, что значение **loop_filter_across_tiles_enabled_flag** равно 1.

cabac_independent_flag равный 1 задает, что декодирование CABAC блоков кодирования в слайсе является независимым от любого состояния ранее декодированного слайса. **cabac_independent_flag** равный 0 задает, что декодирование CABAC блоков кодирования в слайсе является зависимым от состояний ранее декодированного слайса. Когда не присутствует, подразумевается, что значение **cabac_independent_flag** равно 0.

Процесс вывода для доступности блока кодирования с адресом минимального блока кодирования может быть следующим:

Входами для этого процесса являются

- адрес минимального блока кодирования **minCbAddrZS** в порядке z-образного сканирования
- адрес текущего минимального блока кодирования **currMinCbAddrZS** в порядке z-образного сканирования

Выходом этого процесса является доступность блока кодирования с адресом минимального блока кодирования **cbAddrZS** в порядке z-образного сканирования **cbAvailable**.

ПРИМЕЧАНИЕ 1 - Значение доступности определяется тогда, когда вызывается этот процесс.

ПРИМЕЧАНИЕ 2 - Любой блок кодирования, независимо от его размера, ассоциируется с адресом минимального блока кодирования, который представляет собой адрес блока кодирования с минимальным размером блока кодирования в порядке z-образного сканирования.

- Если одно или несколько из следующих условий являются верными, **cbAvailable** устанавливается на FALSE.

- **minCbAddrZS** меньше 0
- **minCbAddrZS** больше **currMinCbAddrZS**
- блок кодирования с адресом минимального блока кодирования **minCbAddrZS** принадлежит другому слайсу, чем блок кодирования с адресом текущего минимального блока кодирования **currMinCbAddrZS**, и **dependent_slice_flag** слайса, содержащего блок кодирования с адресом текущего минимального блока кодирования **currMinCbAddrZS**, равен 0.
- блок кодирования с адресом минимального блока кодирования **minCbAddrZS** содержится в другом тайле, чем блок кодирования с адресом текущего минимального блока кодирования **currMinCbAddrZS**.

- Иначе, **cbAvailable** устанавливается на TRUE.

Процесс синтаксического анализа CABAC для данных слайса может быть следующим: Этот процесс вызывается при синтаксическом анализе некоторых синтаксических элементов с дескриптором **ae(v)**.

Входами для этого процесса являются запрос значения синтаксического элемента и значений ранее синтаксически анализированных синтаксических элементов.

Выходом этого процесса является значение синтаксического элемента.

Во время начала синтаксического анализа данных слайса в слайсе вызывается процесс инициализации процесса синтаксического анализа CABAC.

Фиг.23 иллюстрирует, как пространственный соседний блок T используется для вызова процесса вывода доступности блока дерева кодирования относительно текущего блока дерева кодирования (информативный).

Адрес минимального блока кодирования блока дерева кодирования, содержащего пространственный соседний блок T (фиг.23), **ctbMinCbAddrT**, выводится с использованием

расположения (x0,y0) верхнего-левого отсчета яркости текущего блока дерева кодирования следующим образом.

$$x = x0 + 2 \ll \text{Log2CtbSize} - 1$$

$$y = y0 - 1$$

$$\text{ctbMinCbAddrT} = \text{MinCbAddrZS}[x \gg \text{Log2MinCbSize}][y \gg \text{Log2MinCbSize}]$$

Переменная availableFlagT получается посредством вызова процесса вывода доступности блока кодирования с ctbMinCbAddrT в качестве входа.

В начале синтаксического анализа заданного дерева кодирования применяются следующие упорядоченные этапы.

Механизм арифметического декодирования инициализируется следующим образом.

Если CtbAddrRS равен slice_address, dependent_slice_flag равен 1, и entropy_coding_reset_flag равен 0, применяется следующее.

Вызывается процесс синхронизации процесса синтаксического анализа CABAC с TableStateIdxDS и TableMPSValDS в качестве входа.

Вызывается процесс декодирования для двоичных решений перед завершением, за которым следует процесс инициализации для механизма арифметического декодирования.

Иначе, если tiles_or_entropy_coding_sync_idc равен 2, и CtbAddrRS % PicWidthInCtbs равен 0, применяется следующее.

Когда availableFlagT равен 1, вызывается процесс синхронизации процесса синтаксического анализа CABAC с TableStateIdxWPP и TableMPSValWPP в качестве входа.

Вызывается процесс декодирования для двоичных решений перед завершением, за которым следует процесс для механизма арифметического декодирования.

Когда cabac_independent_flag равен 0, и dependent_slice_flag равен 1, или когда tiles_or_entropy_coding_sync_idc равен 2, процесс запоминания применяется следующим образом.

Когда tiles_or_entropy_coding_sync_idc равен 2, и CtbAddrRS % PicWidthInCtbs равен 2, процесс запоминания процесса синтаксического анализа CABAC вызывается с TableStateIdxWPP и TableMPSValWPP в качестве выхода.

Когда cabac_independent_flag равен 0, dependent_slice_flag равен 1, и end_of_slice_flag равен 1, процесс запоминания процесса синтаксического анализа CABAC вызывается с TableStateIdxDS и TableMPSValDS в качестве выхода.

Синтаксический анализ синтаксических элементов выполняется следующим образом:

Для каждого запрошенного значения синтаксического элемента выводится бинаризация.

Бинаризация для синтаксического элемента и последовательности синтаксически анализированных бинов определяет поток процесса декодирования.

Для каждого бина бинаризации синтаксического элемента, который индексируется переменной binIdx, выводится индекс контекста ctxIdx.

Для каждого ctxIdx вызывается процесс арифметического декодирования.

Результирующая последовательность (b0...bbinIdx) синтаксически анализированных бинов сравнивается с набором строк бинов, определенных процессом бинаризации после декодирования каждого бина. Когда последовательность совпадает со строкой бинов в данном наборе, соответствующее значение назначается синтаксическому элементу.

В случае, если запрос значения синтаксического элемента обрабатывается для синтаксического элемента rcm_flag, и декодированное значение rcm_flag равно 1,

инициализируется механизм декодирования после декодирования любого `pcm_alignment_zero_bit`, `num_subsequent_pcm` и всех данных `pcm_sample_luma` и `pcm_sample_chroma`.

Таким образом, вышеприведенное описание показывает декодер, изображенный на 5 фиг.24. Этот декодер, который в целом обозначен ссылочной позицией 5, восстанавливает изображение 10 из потока 12 данных, в который изображение 10 кодировано в единицах слайсов 14, на которые разбито изображение 10, причем декодер 5 выполнен с возможностью декодирования слайсов 14 из потока 12 данных в соответствии с порядком 16 слайсов. Конечно, декодер 5 не ограничивается 10 последовательным декодированием слайсов 14. Скорее, декодер 5 может использовать волновую параллельную обработку, чтобы декодировать слайсы 14, при условии, что разбиение изображения 10 на слайсы 14 соответствует волновой параллельной обработке. Следовательно, декодер 5, например, может представлять собой декодер, который способен декодировать слайсы 14 параллельно ступенчатым образом, начиная 15 декодирование слайсов 14, принимая во внимание порядок 16 слайсов, чтобы сделать возможной волновую обработку, как было описано выше, и также описано ниже.

Декодер 5 реагирует на участок 18 синтаксического элемента в текущем слайсе из слайсов 14, чтобы декодировать текущий слайс в соответствии с одним из по меньшей мере двух режимов 20 и 22. В соответствии с первым из упомянутых по меньшей мере 20 двух режимов, а именно режимом 20, текущий слайс декодируется из потока 12 данных, используя контекстно-адаптивное энтропийное декодирование, включающее в себя вывод контекста через границы слайсов, т.е. через пунктирные линии на фиг.24, т.е. посредством использования информации, происходящей из кодирования/декодирования других «предшествующих слайсов в порядке 16 слайсов». Кроме того, декодирование 25 текущего слайса из потока 12 данных, используя первый режим 20, содержит непрерывное обновление вероятностей символа кодека и инициализацию вероятностей символа в начале декодирования текущего слайса, который зависит от сохраненных состояний вероятностей символа ранее декодированного слайса. Такая зависимость была описана выше, например, в связи с «процессом синхронизации для переменных 30 кодека». Наконец, первый режим 20 также включает в себя декодирование с предсказанием через границы слайсов. Такое декодирование с предсказанием через границы слайсов, например, может включать в себя внутреннее предсказание через границы слайсов, т.е. предсказание значений отсчетов в текущем слайсе на основе уже восстановленных значений отсчетов «порядка 16 в слайсе», предшествующего слайса 35 или предсказания параметров кодирования через границы слайсов, такого как предсказание векторов движения, режимов предсказания, режимов кодирования или т.п.

В соответствии со вторым режимом 22 декодер 5 декодирует текущий слайс, т.е. слайс, в настоящий момент подлежащий декодированию, из потока 12 данных, используя 40 контекстно-адаптивное энтропийное декодирование с ограничением, однако, вывода контекстов, чтобы не пересекать границы слайсов. Если когда-либо, например, шаблон соседних положений, используемый для вывода контекста для некоторого синтаксического элемента, относящегося к блоку в текущем слайсе, простирается в соседний слайс, таким образом пересекая границу слайсов текущего слайса, 45 соответствующий атрибут соответствующего участка соседнего слайса, такой как значение соответствующего синтаксического элемента этого соседнего участка соседнего слайса, устанавливается на значение по умолчанию, чтобы блокировать взаимозависимости между текущим слайсом и соседними слайсами. Хотя непрерывное

обновление вероятностей символа контекстов может происходить точно так, как это имеет место в первом режиме 20, инициализация вероятностей символа во втором режиме 22 является независимой от любого раннего декодированного слайса. Кроме того, декодирование с предсказанием выполняется с ограничением декодирования с предсказанием, чтобы не пересекать границы слайсов.

Чтобы облегчить понимание описания фиг.24 и последующего описания, ссылка делается на фиг.25, которая изображает возможную реализацию декодера 5 в более конструктивном смысле, чем по сравнению с фиг.24. Как это имеет место на фиг.24, декодер 5 представляет собой декодер с предсказанием, использующий контекстно-адаптивное энтропийное декодирование для декодирования потока данных, чтобы получить, например, остаток предсказания и параметры предсказания.

Как показано на фиг.25, декодер 5 может содержать энтропийный декодер 24, модуль 26 деквантования и обратного преобразования, объединитель 28, реализованный, как показано на фиг.25, например, в виде сумматора, и предсказатель 28. Энтропийный декодер 24, модуль 26 и сумматор 27 последовательно подсоединены между входом и выходом декодера 5 в порядке их упоминания, и предсказатель 28 подсоединен между выходом сумматора 28 и другим его входом, чтобы формировать петлю предсказания с объединителем 27. Следовательно, декодер 24 имеет свой выход, дополнительно подсоединенный к входу параметров кодирования предсказателя 28.

Хотя фиг.25 создает впечатление, что декодер последовательно декодирует текущее изображение, декодер 5 может, например, быть реализован так, чтобы декодировать изображение 10 параллельно. Декодер 5, например, может содержать многочисленные ядра, при этом каждый работает в соответствии с элементами 24-28 на фиг.25.

Параллельная обработка, однако, является необязательной, и последовательно работающий декодер 5 также способен декодировать поток данных, поступающий на вход энтропийного декодера 24.

Чтобы эффективно достичь вышеупомянутой способности или последовательного, или параллельного декодирования текущего изображения 10, декодер 5 работает в единицах блоков 30 кодирования, чтобы декодировать изображение 10. Блоками 30 кодирования, например, являются листовые блоки, на которые разбиваются блоки дерева кодирования или наибольшие блоки 32 кодирования посредством рекурсивного разбиения методом мультидерева, таким как разбиение методом квадродерева. Блоки 32 дерева кодирования, в свою очередь, могут регулярно располагаться столбцами и строками, чтобы формировать регулярное разбиение изображения 10 на эти блоки 32 дерева кодирования. На фиг.25 блоки 32 дерева кодирования показаны непрерывными линиями, тогда как блоки 30 кодирования показаны пунктирными линиями. С целью иллюстрации только один блок 32 дерева кодирования показан дополнительно разбитым на блоки 30 кодирования, тогда как другие блоки 32 дерева кодирования показаны без дополнительного разбиения, чтобы, вместо этого, непосредственно формировать блок кодирования. Поток 12 данных может содержать сигнализацию синтаксического участка, как изображение 10 разбито на блоки 30 кодирования.

Поток 12 данных передает, для каждого блока 30 кодирования, синтаксические элементы, которые показывают, как модули 24-28 должны восстанавливать содержимое изображения в этом блоке 30 кодирования. Например, эти синтаксические элементы содержат:

1) необязательно, данные разбиения, дополнительно разбивающие блок 30 кодирования на блоки предсказания,

2) необязательно, данные разбиения, дополнительно разбивающие блок 30

кодирования на блоки остатка и/или преобразования,

3) сигнализацию режима предсказания, какой режим предсказания должен использоваться для вывода сигнала предсказания для блока 30 кодирования, причем степень детализации, с которой этот режим предсказания сигнализируется, может

зависеть от блоков 30 кодирования и/или блока предсказания.

4) параметры предсказания могут сигнализироваться на блок кодирования или, если присутствует, на блок предсказания с типом параметров предсказания, посылаемых в зависимости, например, от режима предсказания. Возможные режимы предсказания, например, могут содержать внутреннее предсказание и/или внешнее предсказание.

5) Другие синтаксические элементы также могут присутствовать, такие как информация о фильтрации для фильтрации изображения 10 в блоке 30 кодирования, чтобы получить сигнал предсказания и/или восстановленный сигнал, подлежащий воспроизведению.

6) Наконец, информация об остатке в виде, между прочим, коэффициентов преобразования может содержаться в потоке данных для блока 30 кодирования; в единицах блоков остатка, остаточные данные могут сигнализироваться; на блок остатка, спектральное разложение, например, может выполняться в единицах вышеупомянутых блоков преобразования, если они присутствуют.

Энтропийный декодер 24 отвечает за получение вышеупомянутых синтаксических элементов из потока данных. С этой целью, энтропийный декодер 24 использует контекстно-адаптивное энтропийное декодирование. Т.е. энтропийный декодер 24 обеспечивает несколько видов контекста. Чтобы вывести некоторый синтаксический элемент из потока 12 данных, энтропийный декодер 24 выбирает некоторый контекст из числа возможных контекстов. Выбор из числа возможных контекстов выполняется в зависимости от атрибута соседства участка изображения 10, к которому принадлежит текущий синтаксический контекст. Для каждого из возможных контекстов энтропийный декодер 24 управляет вероятностями символа, т.е. оценкой вероятности для каждого возможного символа из алфавита символов, основываясь на том, на котором работает энтропийный декодер 24. «Управление» включает в себя вышеупомянутые непрерывные обновления вероятностей символа контекстов, чтобы адаптировать вероятности символа, ассоциированные с каждым контекстом, к фактическому содержанию изображения. Посредством этой меры вероятности символа адаптируются к фактической статистике вероятностей символа.

Другим случаем, когда атрибуты соседства оказывают влияние на восстановление текущего участка изображения 10, такого как текущий блок 30 кодирования, является декодирование с предсказанием в предсказателе 28. Предсказание ограничивается не только содержимым предсказания в текущем блоке 30 кодирования, но также может охватывать предсказание параметров, содержащихся в потоке 12 данных для текущего блока 30 кодирования, таких как параметры предсказания, данные разбиения или даже коэффициенты преобразования. Т.е. предсказатель 28 может предсказывать содержимое изображения или такие параметры из вышеупомянутого соседства, чтобы получать записанный сигнал, который затем объединяется с остатком предсказания, полученным модулем 26 из потока 12 данных. В случае параметров предсказания предсказатель 28 может использовать синтаксические элементы, содержащиеся в потоке данных, в качестве остатков предсказания, чтобы получать фактическое значение параметра предсказания. Предсказатель 28 использует последнее значение параметра предсказания, чтобы получать только что упомянутый сигнал предсказания, подлежащий объединению с остатком предсказания в объединителе 27.

Вышеупомянутое «соседство» охватывает, главным образом, верхнюю левую часть окружности текущего участка, к которому принадлежит синтаксический элемент, подлежащий в настоящий момент энтропийному декодированию, или синтаксический элемент, подлежащий в настоящий момент предсказанию. На фиг.25 такое соседство

изображается позицией 34 в качестве примера для одного блока 30 кодирования. Порядок кодирования/декодирования определяется среди блоков 30 кодирования: на самом грубом уровне, блоки 32 дерева кодирования изображения 10 сканируются в порядке 36 сканирования, в данном случае изображен как сканирование раstra, происходящее построчно сверху вниз. В каждом блоке дерева кодирования блоки 30 кодирования сканируются в порядке обхода вглубь, так что, на каждом иерархическом уровне, блок 32 дерева кодирования сканируется, по существу, также сканированием раstra, происходящим построчно сверху вниз.

Порядок сканирования, определенный среди блоков 30 кодирования, согласовывается с определением соседства 34, используемого для вывода атрибута в соседстве, чтобы выбирать контексты и/или выполнять пространственное предсказание, тем, что соседство 34, главным образом, охватывает участки изображения 10, которые уже были подвергнуты декодированию в соответствии с порядком кодирования. Всякий раз, когда участок соседства 34 охватывает недоступные участки изображения 10, вместо этого используются данные по умолчанию. Например, шаблон 34 соседства может простираться за пределы изображения 10. Другой возможностью, однако, является то, что соседство 34 простирается в соседний слайс.

Слайсы делят, например, изображение 10 в соответствии с порядком кодирования/декодирования, определенным по блокам 30 кодирования, т.е. каждый слайс представляет собой непрерывную непрерываемую последовательность блоков 30 кодирования в соответствии с вышеупомянутым порядком блоков кодирования. На фиг.25 слайсы обозначены пунктирными линиями 14. Порядок, определенный среди слайсов 14, является результатом их расположения выполнений последовательных блоков 30 кодирования, как кратко изложено выше. Если участок 18 синтаксического элемента некоторого слайса 14 указывает, что он должен декодироваться в первом режиме, тогда энтропийный декодер 24 делает возможным контекстно-адаптивное энтропийное декодирование для вывода контекстов через границы слайсов. Т.е. используется пространственное соседство 34, чтобы выбрать контексты при энтропийном декодировании данных, касающихся текущего слайса 14. В случае фиг.25, например, слайсом номер 3 может быть декодируемый в настоящий момент слайс, и при энтропийном декодировании синтаксического элемента, касающегося блока 30 кодирования или некоторого участка, содержащего в нем, энтропийный декодер 24 может использовать атрибуты, происходящие из декодирования участков в соседнем слайсе, таком как слайс номер 1. Предсказатель 28 ведет себя также: для слайсов в первом режиме 20 предсказатель 28 использует пространственное предсказание через границу слайсов, охватывающую текущий слайс.

Для слайсов, однако, имеющих второй режим 22, ассоциированный с ними, т.е. для которых участок 18 синтаксического элемента указывает второй режим 22, энтропийный декодер 24 и предсказатель 28 ограничивают вывод энтропийных контекстов и декодирование с предсказанием тем, что они зависят от атрибутов, относящихся к участкам, лежащим только в пределах текущего слайса. Очевидно, что эффективность кодирования страдает от этого ограничения. С другой стороны, слайсы второго режима 22 принимают во внимание нарушение взаимозависимостей между последовательностью слайсов. Следовательно, слайсы второго режима 22 могут распределяться в пределах

изображения 10 или в пределах видео, к которому принадлежит изображение 10, чтобы принимать во внимание точки повторной синхронизации. Не является необходимым, однако, чтобы каждое изображение 10 имело по меньшей мере один слайс во втором режиме 22.

5 Как уже упомянуто выше, первый и второй режим 20 и 22 также отличаются своей инициализацией вероятностей символа. Слайсы, кодированные по второму режиму 22, приводят к тому, что энтропийный декодер 24 повторно инициализирует вероятности независимо от любого ранее декодированного слайса, т.е. ранее декодированного в смысле порядка, определенного среди слайсов. Вероятности символа, например,
10 устанавливаются на значения по умолчанию, известные как кодирующей, так и декодирующей стороне, или значения инициализации содержатся в слайсах, кодированных во втором режиме 22.

Т.е. для слайсов, кодируемых/декодируемых во втором режиме 22, адаптация вероятностей символа всегда начинается немедленно с началом этих слайсов.

15 Следовательно, точность адаптации является плохой для этих слайсов в начале этих слайсов.

Дело обстоит иначе в слайсах, кодируемых/декодируемых в первом режиме 20. Для последних слайсов инициализация вероятностей символа, выполняемая энтропийным декодером 24, зависит от сохраненных состояний вероятностей символа ранее
20 декодированного слайса. Всякий раз, когда слайс, кодируемый/декодируемый в первом режиме 20, имеет свое начало, например, расположенное кроме левой стороны изображения 10, т.е. не на стороне, откуда сканирование 36 раstra начинает выполнение построчно перед выполнением шага на следующую строку вниз, принимаются вероятности символа, получающиеся в конце энтропийного декодирования
25 непосредственно предшествующего слайса. Это изображено, например, на фиг.2 стрелкой 38 для слайса №4. Слайс №4 имеет свое начало где-то между правой стороной и левой стороной изображения 10, и, следовательно, при инициализации вероятностей символа энтропийный декодер 24 принимает, при инициализации вероятностей символа, вероятности символа, которые были получены при энтропийном декодировании
30 непосредственно предшествующего слайса, т.е. слайса №3, до его окончания, т.е. включая непрерывное обновление вероятностей символа во время энтропийного декодирования слайса №3 до его окончания.

Слайсы, имеющие второй режим 22, ассоциированный с ними, которые, однако, имеют начало на левой стороне изображения 10, такой как, например, слайс №5, не
35 принимают вероятности символа, полученные после окончания энтропийного декодирования непосредственно предшествующего слайса №4, так как это бы предотвратило бы параллельное декодирование декодером 5 изображения 10 посредством использования волновой обработки. Скорее, как кратко изложено выше, энтропийный декодер 24 адаптирует вероятности символа, полученные после окончания
40 энтропийного декодирования второго - в порядке 36 кодирования/декодирования - блока 32 дерева кодирования в непосредственно предшествующем - в порядке 36 кодирования/декодирования - блоке дерева кодирования, как показано стрелкой 40.

На фиг.25, например, изображение 10, в качестве примера, было разбито на три строки блоков дерева кодирования и четыре столбца блоков 32 корня дерева
45 кодирования, и каждая строка блоков дерева кодирования была подразделена на два слайса 14, так что начало каждого второго слайса совпадает с первой единицей кодирования в порядке единиц кодирования соответствующей строки блоков корня дерева кодирования. Энтропийный декодер 24, следовательно, имеет возможность

использовать волновую обработку при декодировании изображения 10 посредством декодирования параллельно каждой строки блоков корня дерева кодирования, при этом декодирование этих строк блоков корня дерева кодирования начинается ступенчатым образом, начиная с первой или самой верхней строки блоков корня дерева кодирования, затем второй, и затем третьей.

Конечно, разбиение блоков 32 рекурсивным образом на дополнительные блоки 30 кодирования является необязательным, и, следовательно, в более общем смысле, блоки 32 также могут называться «блоками кодирования». Т.е. вообще говоря, изображение 10 может разбиваться на блоки 32 кодирования, расположенные по строкам и столбцам и имеющие порядок 36 сканирования растра, определенный между ними, и декодер 5 может рассматриваться как ассоциирующий каждый слайс 14 с непрерывным поднабором блоков 32 кодирования в порядке 36 сканирования растра, так что поднаборы следуют друг за другом в соответствии с порядком 36 сканирования растра в соответствии с порядком слайсов.

Как также становится ясно из вышеприведенного описания, декодер 5, или более конкретно, энтропийный декодер 24 может быть выполнен с возможностью сохранения вероятностей символа, полученных при контекстно-адаптированном энтропийном декодировании любого слайса до второго блока кодирования в строке блоков кодирования в соответствии с порядком 36 сканирования растра. При инициализации вероятностей символа для контекстно-адаптивного энтропийного декодирования текущего слайса, имеющего первый режим 20, ассоциированный с ним, декодер 5, или, более конкретно, энтропийный декодер 24, проверяет, является ли первый блок 32 кодирования непрерывного поднабора блоков 32 кодирования, ассоциированных с текущим слайсом, первым блоком 32 кодирования в строке блоков кодирования в соответствии с порядком 36 сканирования растра. Если это так, вероятности символа для контекстно-адаптивного энтропийного декодирования текущего слайса инициализируются как объяснено в отношении стрелки 40, а именно, в зависимости от сохраненных вероятностей символа, полученных при контекстном энтропийном декодировании ранее декодированного слайса до второго блока кодирования в строке блоков кодирования в соответствии с порядком 36 сканирования растра. Если нет, инициализация вероятностей символа для контекстно-адаптивного энтропийного декодирования текущего слайса выполняется в зависимости от вероятностей символа, полученных при контекстно-адаптивном энтропийном декодировании ранее декодированного слайса до конца ранее декодированного слайса, т.е. в соответствии со стрелкой 38. Снова, в случае инициализации согласно позиции 38, подразумевается сохраненное состояние в конце энтропийного декодирования непосредственно предшествующего слайса в порядке 36 слайсов, тогда как в случае инициализации 40 им является этот ранее декодированный слайс, содержащий конец второго блока непосредственно предшествующей строки блока 32 в порядке 36 блоков.

Как показано пунктирными линиями на фиг.24, декодер может быть выполнен с возможностью реагирования на участок 18 синтаксического элемента в текущем слайсе из слайсов 14, чтобы декодировать текущий слайс в соответствии с одним из по меньшей мере трех режимов. Т.е. может быть третий режим 42 кроме других режимов 20 и 22. Третий режим 42 может отличаться от второго режима 22 тем, что разрешается предсказание через границы слайсов, тогда как энтропийное кодирование/декодирование все еще ограничивается, чтобы не пересекать границы слайсов.

Выше были представлены два варианта осуществления, касающиеся участка 18 синтаксического элемента. Таблица ниже суммирует эти два варианта осуществления.

	вариант 1 осуществления	вариант 2 осуществления
участок синтаксического элемента	dependent_slice_flag, no_cabac_reset_flag	dependent_slice_flag
режим 1	dependent_slice_flag = 1, no_cabac_reset_flag = 1	dependent_slice_flag = 1,
режим 2	dependent_slice_flag = 0	dependent_slice_flag = 0
режим 3	dependent_slice_flag = 1, no_cabac_reset_flag = 0	dependent_slice_flag = 1, cabac_independent_flag = 1, tiles_or_entropy_coding_sync_idc = 3

В одном варианте осуществления участок 18 синтаксического элемента образован индивидуально посредством dependent_slice_flag, тогда как в другом варианте осуществления комбинация dependent_slice_flag и no_cabac_reset_flag образует участок синтаксического элемента. Ссылка делается на процесс синхронизации для контекстных переменных, что касается инициализации вероятностей символа в зависимости от сохраненных состояний вероятностей символа ранее декодированного слайса. В частности, декодер может быть выполнен с возможностью, если last_ctb_cabac_init_flag=0 и tiles_or_entropy_coding_sync_idc=2, сохранения вероятностей символа, полученных при контекстно-адаптивном энтропийном декодировании ранее декодированного слайса до второго блока кодирования в строке в соответствии с порядком сканирования раstra, и, при инициализации вероятностей символа для контекстно-адаптивного энтропийного декодирования текущего слайса в соответствии с первым режимом, проверки, является ли первый блок кодирования непрерывного поднабора блоков кодирования, ассоциированных с текущим слайсом, первым блоком кодирования в строке в соответствии с порядком сканирования раstra, и, если это так, инициализации вероятностей символа для контекстно-адаптивного энтропийного декодирования текущего слайса в зависимости от сохраненных вероятностей символа, полученных при контекстно-адаптивном энтропийном декодировании ранее декодированного слайса до второго блока кодирования в строке в соответствии с порядком сканирования раstra, и, если нет, инициализации вероятностей символа для контекстно-адаптивного энтропийного декодирования текущего слайса в зависимости от вероятностей символа, полученных при контекстно-адаптивном энтропийном декодировании ранее декодированного слайса до конца ранее декодированного слайса.

Таким образом, другими словами, в соответствии со вторым вариантом осуществления для синтаксиса, декодер восстанавливает изображение 10 из потока 12 данных, в который изображение кодировано в единицах слайсов 14, на которые изображение (10) разбито, причем декодер выполнен с возможностью декодирования слайсов 14 из потока 12 данных в соответствии с порядком 16 слайсов, и декодер реагирует на участок 18 синтаксического элемента, а именно dependent_slice_flag в текущем слайсе из слайсов, чтобы декодировать текущий слайс в соответствии с одним из по меньшей мере двух режимов 20, 22. В соответствии с первым 20 из упомянутых по меньшей мере двух режимов, а именно, если dependent_slice_flag=1, декодер декодирует текущий слайс из потока 12 данных, используя контекстно-адаптивное энтропийное декодирование 24, включающее в себя вывод контекстов через границы слайсов, непрерывное обновление вероятностей символа контекстов и инициализацию 38, 40

вероятностей символа в зависимости от сохраненных состояний вероятностей символа ранее декодированного слайса, и декодирование с предсказанием через границы слайсов, и в соответствии со вторым 22 из упомянутых по меньшей мере двух режимов, а именно, если `dependent_slice_flag=0`, декодер декодирует текущий слайс из потока 12 данных, используя контекстно-адаптивное энтропийное декодирование с ограничением вывода контекстов, чтобы не пересекать границы слайсов, непрерывное обновление вероятностей символа контекстов и инициализацию вероятностей символа независимо от любого ранее декодированного слайса, и декодирование с предсказанием с ограничением декодирования с предсказанием, чтобы не пересекать границы слайсов.

Изображение 10 может быть разбито на блоки 32 кодирования, расположенные по строкам и столбцам и имеющие порядок 36 сканирования раstra, определенный между ними, и декодер выполнен с возможностью ассоциирования каждого слайса 14 с непрерывным поднабором блоков 32 кодирования в порядке 36 сканирования раstra, так что поднаборы следуют друг за другом в соответствии с порядком 36 сканирования раstra в соответствии с порядком слайсов. Декодер может быть выполнен с возможностью, а именно реагируя на `tiles_or_entropy_coding_sync_idc=2`, сохранения вероятностей символа, полученных при контекстно-адаптивном энтропийном декодировании ранее декодированного слайса до второго блока 32 кодирования в строке в соответствии с порядком 36 сканирования раstra, и, при инициализации вероятностей символа для контекстно-адаптивного энтропийного декодирования текущего слайса в соответствии с первым режимом, проверки, является ли первый блок кодирования непрерывного поднабора блоков 32 кодирования, ассоциированных с текущим слайсом, первым блоком 32 кодирования в строке в соответствии с порядком сканирования раstra, и, если это так, инициализации 40 вероятностей символа для контекстно-адаптивного энтропийного декодирования текущего слайса в зависимости от сохраненных вероятностей символа, полученных при контекстно-адаптивном энтропийном декодировании ранее декодированного слайса до второго блока кодирования в строке в соответствии с порядком 36 сканирования раstra, и, если нет, инициализации 38 вероятностей символа для контекстно-адаптивного энтропийного декодирования текущего слайса в зависимости от вероятностей символа, полученных при контекстно-адаптивном энтропийном декодировании ранее декодированного слайса до конца ранее декодированного слайса. Декодер может быть выполнен с возможностью реагирования на участок (18) синтаксического элемента в текущем слайсе из слайсов 14, чтобы декодировать текущий слайс в соответствии с одним из по меньшей мере трех режимов, а именно, в одном из первого 20 и третьего режима 42 или второго режима 22, причем декодер выполнен с возможностью, в соответствии с третьим режимом 42, а именно, если `dependent_slice_flag=1` и `tiles_or_entropy_coding_sync_idc=3`, декодирования текущего слайса из потока данных, используя контекстно-адаптивное энтропийное декодирование с ограничением вывода контекстов, чтобы не пересекать границы слайсов, непрерывное обновление вероятностей символа контекстов и инициализацию вероятностей символа независимо от любого ранее декодированного слайса, и декодирования с предсказанием через границы слайсов, причем один из первого и третьего режимов выбирается в зависимости от синтаксического элемента, а именно `cabac_independent_flag`. Декодер дополнительно может быть выполнен с возможностью, а именно, если `tiles_or_entropy_coding_sync_idc=0,1` и 3 («3», когда `cabac_independent_flag=0`), сохранения вероятностей символа, полученных при контекстно-адаптивном энтропийном декодировании ранее декодированного слайса до конца ранее декодированного слайса, и, при инициализации вероятностей символа для контекстно-

адаптивного энтропийного декодирования текущего слайса в соответствии с первым режимом, инициализации вероятностей символа для контекстно-адаптивного энтропийного декодирования текущего слайса в зависимости от сохраненных вероятностей символа. Декодер может быть выполнен с возможностью, а именно, если
 5 tiles_or_entropy_coding_sync_idc=1, в первом и втором режиме, ограничения декодирования с предсказанием в тайлах, на которые изображение подразделено.

Конечно, кодер способен устанавливать представленный выше синтаксис соответствующим образом, чтобы дать возможность декодеру получить вышеприведенные преимущества. Кодером может быть кодер, такой как многоядерный,
 10 параллельной обработки, но в этом нет необходимости. Для кодирования изображения 10 в поток 12 данных в единицах слайсов 14 кодер выполнен с возможностью кодирования слайсов 14 в поток 12 данных в соответствии с порядком 16 слайсов. Кодер определяет участок 18 синтаксического элемента для текущего слайса из слайсов, и кодирует его в него, так что участок синтаксического элемента сигнализирует, что текущий слайс
 15 должен быть кодирован в соответствии с одним из упомянутых по меньшей мере двух режимов 20, 22, и, если текущий слайс должен быть кодирован в соответствии с первым 20 из упомянутых по меньшей мере двух режимов, кодирует текущий слайс в поток 12 данных, используя контекстно-адаптивное энтропийное кодирование 24, включающее в себя вывод контекстов через границы слайсов, непрерывное обновление вероятностей
 20 символа контекстов и инициализацию 38, 40 вероятностей символа в зависимости от сохраненных состояний вероятностей символа ранее кодированного слайса, и кодирование с предсказанием через границы слайсов, и, если текущий слайс должен быть кодирован в соответствии со вторым 22 из упомянутых по меньшей мере двух режимов, кодирует текущий слайс в поток 12 данных, используя контекстно-адаптивное
 25 энтропийное кодирование с ограничением вывода контекстов, чтобы не пересекать границы слайсов, непрерывное обновление вероятностей символа контекстов и инициализацию вероятностей символа независимо от любого ранее кодированного слайса, и кодирование с предсказанием с ограничением кодирования с предсказанием, чтобы не пересекать границы слайсов. Хотя изображение 10 может разбиваться на
 30 блоки 32 кодирования, расположенные по строкам и столбцам и имеющие порядок 36 сканирования раstra, определенный между ними, кодер может быть выполнен с возможностью ассоциирования каждого слайса 14 с непрерывным поднабором блоков 32 кодирования в порядке 36 сканирования раstra, так что поднаборы следуют один за другим в соответствии с порядком 36 сканирования раstra в соответствии с порядком
 35 слайсов. Кодер может быть выполнен с возможностью сохранения вероятностей символа, полученных при контекстно-адаптивном энтропийном кодировании ранее кодированного слайса до второго блока 32 кодирования в строке в соответствии с порядком 36 сканирования раstra, и, при инициализации вероятностей символа для контекстно-адаптивного энтропийного кодирования текущего слайса в соответствии
 40 с первым режимом, проверки, является ли первый блок кодирования непрерывного поднабора блоков 32 кодирования, ассоциированных с текущим слайсом, первым блоком 32 кодирования в строке в соответствии с порядком сканирования раstra, и, если это так, инициализации 40 вероятностей символа для контекстно-адаптивного энтропийного кодирования текущего слайса в зависимости от сохраненных вероятностей
 45 символа, полученных при контекстно-адаптивном энтропийном кодировании ранее кодированного слайса до второго блока кодирования в строке в соответствии с порядком 36 сканирования раstra, и, если это не так, инициализации 38 вероятностей символа для контекстно-адаптивного энтропийного кодирования текущего слайса в

зависимости от вероятностей символа, полученных при контекстно-адаптивном энтропийном кодировании ранее декодированного слайса до конца ранее кодированного слайса. Кодер может быть выполнен с возможностью кодирования участка (18) синтаксического элемента в текущий слайс из слайсов (14), так что сигнализируется, что текущий слайс должен быть кодирован в него в соответствии с одним из по меньшей мере трех режимов, а именно в одном из первого (20) и третьего режима (42) или второго режима (22), причем кодер выполнен с возможностью, ** в соответствии с третьим режимом (42), кодирования текущего слайса в поток данных, используя контекстно-адаптивное энтропийное кодирование с ограничением вывода контекстов, чтобы не пересекать границы слайсов, непрерывное обновление вероятностей символа контекстов и инициализацию вероятностей символа независимо от любого ранее кодированного слайса, и кодирование с предсказанием через границы слайсов, причем кодер различает один из первого и третьего режимов, используя, например, синтаксический элемент, а именно `cabac_independent_flag`. Кодер может быть выполнен с возможностью определения обобщенного синтаксического элемента, такого как `dependent_slices_present_flag` и записи его в поток данных, работая в одном из по меньшей мере двух обобщенных режимов работы в зависимости от обобщенного синтаксического элемента, а именно, с, согласно первому обобщенному режиму работы, выполнением кодирования участка синтаксического элемента для каждого слайса, и, согласно второму обобщенному режиму работы, неизбежного использования другого одного из упомянутых по меньшей мере двух режимов, кроме первого режима. Кодер может быть выполнен с возможностью, согласно первому и второму режимам, неизбежно и непрерывно продолжать непрерывное обновление вероятностей символа с начала до конца текущего слайса. Кодер может быть выполнен с возможностью сохранения вероятностей символа, полученных при контекстно-адаптивном энтропийном кодировании ранее кодированного слайса до конца ранее кодированного слайса, и, при инициализации вероятностей символа для контекстно-адаптивного энтропийного кодирования текущего слайса в соответствии с первым режимом, инициализации вероятностей символа для контекстно-адаптивного энтропийного кодирования текущего слайса в зависимости от сохраненных вероятностей символа. И кодер может, в первом и втором режиме, ограничивать кодирование с предсказанием в тайлах, на которые изображение подразделено.

Возможная конструкция кодера изображена на фиг.26 для полноты. Предсказатель 70 работает почти также как и предсказатель 28, т.е. выполняет предсказание, но также определяет посредством оптимизации, например, параметры кодирования, включающие в себя параметры и режимы предсказания. Модули 26 и 27 также находятся в декодере. Вычитатель 72 определяет остаток предсказания без потерь, который затем - посредством использования квантования и, необязательно, использования преобразования со спектральным разложением - кодируется с потерями в модуле 74 преобразования и квантования. Энтропийный кодер 76 выполняет контекстно-адаптивное энтропийное кодирование.

В дополнение к вышеупомянутым конкретным примерам синтаксиса ниже кратко описывается другой пример с представлением согласованности между терминами, используемыми в данном документе ниже, и терминами, используемыми выше.

В частности, без конкретного краткого описания выше, зависимые слайсы являются не только «зависимыми» в том, что они учитывают применение сведений, известных за пределами их границ, как например, как упомянуто выше, энтропийные контексты которых адаптируются быстрее, или достигают лучшего пространственного

предсказания из-за разрешения пересечения их границ. Скорее, чтобы сэкономить тарифные затраты, необходимые потратить для определения заголовков слайсов посредством разбиения изображения на слайсы, зависимые слайсы принимают часть синтаксиса заголовка слайса из предыдущих слайсов, т.е. эта часть заголовка синтаксиса слайса не передается снова для зависимых слайсов. Это показано, например, на фиг.16

5 позицией 100 и на фиг.21 позицией 102, согласно которым тип слайса, например, принимается от предыдущего слайса. Посредством этой меры подразделение изображения на слайсы, такие как независимый слайс и зависимые слайсы, является менее дорогостоящим в отношении расхода битов.

10 Именно вышеупомянутая зависимость приводит, в кратко описанном ниже примере, к слегка другой формулировке: слайсы определяются как единичные участки изображения, в которых синтаксис заголовка слайса является индивидуально устанавливаемым. Следовательно, слайсы состоят из одного - используя систему обозначений выше - независимого/регулярного/нормального слайса, теперь называемого

15 независимым сегментом слайса, и без одного или нескольких - используя систему обозначений выше - зависимых слайсов, теперь называемых зависимыми сегментами слайса.

Фиг.27, например, изображает изображение, подлежащее разбиению на два слайса, один образован сегментами 14₁-14₃ слайса, и другой образован исключительно

20 сегментом 14₄ слайса. Индексы 1-4 показывают порядок слайсов в порядке кодирования. Фиг.28а и b изображают другой пример в случае подразделения изображения 10 на два тайла, причем в случае фиг.28а один слайс образован всеми пятью сегментами 14 слайса, охватывающими оба тайла 50₁ и 50₂ - индекс снова повышается в порядке кодирования

25 - и, в случае фиг.28а, два слайса образованы сегментами 14₁ и 14₂ и 14₃ и 14₄ слайса соответственно, подразделяя тайл 50₁, и другой слайс образован сегментами 14₅-14₆ слайса, охватывающими тайл 50₂.

Определениями могут быть следующие:

30 **зависимый сегмент слайса:** *Сегмент слайса*, для которого значения некоторых синтаксических элементов заголовка сегмента слайса подразумеваются из значений для предыдущего независимого сегмента слайса в порядке декодирования - раньше, в вышеупомянутых вариантах осуществления - называемых зависимым слайсом.

независимый сегмент слайса: *Сегмент слайса*, для которого значения синтаксических элементов заголовка сегмента слайса не подразумеваются из значений для предыдущего

35 сегмента слайса - раньше, в вышеупомянутых вариантах осуществления, называемого нормальным слайсом.

слайс: Целочисленное количество единиц дерева кодирования, содержащихся в одном независимом сегменте слайса и всех последующих зависимых сегментах слайса (если

40 есть какие-либо), которые предшествуют следующему независимому сегменту слайса (если есть какой-либо) в одной и той же единице доступа/изображении.

заголовок слайса: *Заголовок сегмента слайса независимого сегмента слайса*, который представляет собой текущий сегмент слайса или представляет собой независимый сегмент слайса, который предшествует текущему зависимому сегменту слайса.

сегмент слайса: Целочисленное количество единиц дерева кодирования,

45 упорядоченных последовательно в сканировании тайла и содержащихся в единственной единице NAL; деление каждого изображения на сегменты слайса представляет собой разбиение.

заголовок сегмента слайса: Часть кодированного сегмента слайса, содержащего

элементы данных, относящиеся к первому или всем *единицам дерева кодирования*, представленным в *сегменте слайса*.

Сигнализация «режимов» 20 и 22, т.е. «зависимого сегмента слайса» и «независимого сегмента слайса» может быть следующей:

- 5 В некоторых дополнительных единицах NAL, таких как набор параметров изображения (PPS), синтаксический элемент может использоваться для сигнализации, выполняется ли или нет использование зависимых слайсов для некоторого изображения даже последовательности для некоторых изображений:

- 10 **dependent_slice_segments_enabled_flag** равный 1 задает присутствие синтаксического элемента **dependent_slice_segment_flag** в заголовках сегмента слайса. **dependent_slice_segments_enabled_flag** равный 0 задает отсутствие синтаксического элемента **dependent_slice_segment_flag** в заголовках сегмента слайса.

dependent_slice_segments_enabled_flag подобен по объему ранее описанному **dependent_slices_present_flag**.

- 15 Аналогично, **dependent_slice_flag** может называться **dependent_slice_segment_flag**, чтобы принимать во внимание разные системы обозначений в отношении слайсов.

- dependent_slice_segment_flag** равный 1 задает, что подразумевается, что значение каждого синтаксического элемента заголовка сегмента слайса, который не присутствует в текущем заголовке сегмента слайса, равно значению соответствующего синтаксического элемента заголовка сегмента слайса в заголовке слайса, т.е. заголовка сегмента слайса предшествующего независимого сегмента слайса.

На этом же уровне, таком как уровень изображения, может быть включен следующий синтаксический элемент:

- 25 **entropy_coding_sync_enabled_flag** равный 1 задает, что вызывается конкретный процесс синхронизации для контекстных переменных перед декодированием единицы дерева кодирования, которая включает в себя первый блок дерева кодирования строки блоков дерева кодирования в каждом тайле в каждом изображении, ссылающемся на PPS, и конкретный процесс сохранения для контекстных переменных вызывается после декодирования единицы дерева кодирования, которая включает в себя второй блок дерева кодирования строки блоков дерева кодирования в каждом тайле в каждом изображении, ссылающемся на PPS. **entropy_coding_sync_enabled_flag** равный 0 задает, что не требуется вызов конкретного процесса синхронизации для контекстных переменных перед декодированием единицы дерева кодирования, которая включает в себя первый блок дерева кодирования строки блоков дерева кодирования в каждом тайле в каждом изображении, ссылающемся на PPS, и не требуется вызов конкретного процесса сохранения для контекстных переменных после декодирования единицы дерева кодирования, которая включает в себя второй блок дерева кодирования строки блоков дерева кодирования в каждом тайле в каждом изображении, ссылающемся на PPS.

- 40 Требованием соответствия битовых потоков является то, что значение **entropy_coding_sync_enabled_flag** должно быть одинаковым для всех PPS, которые активизируются в CVS.

- 45 Когда **entropy_coding_sync_enabled_flag** равен 1, и первый блок дерева кодирования в слайсе не является первым блоком дерева кодирования строки блоков дерева кодирования в тайле, требованием соответствия битовых потоков является то, что последний блок дерева кодирования в слайсе должен принадлежать этой же строке блоков дерева кодирования, что и первый блок дерева кодирования в слайсе. Когда **entropy_coding_sync_enabled_flag** равен 1, и первый блок дерева кодирования в сегменте слайса не является первым блоком дерева кодирования строки блоков дерева кодирования

кодирования в тайле, требованием соответствия битовых потоков является то, что последний блок дерева кодирования в сегменте слайса должен принадлежать этой же строке блоков дерева кодирования, что и первый блок дерева кодирования в сегменте слайса.

5 Как уже было описано, порядок кодирования/декодирования среди СТВ 30 приводит к построчной форме растра сверху вниз, начинающейся со сканирования первого тайла, затем переходя к следующему тайлу, если в изображении присутствует более одного тайла.

Декодер 5 - и кодер соответственно - действует следующим образом при энтропийном декодировании (кодировании) сегментов 14 слайса изображения:

A1) Всякий раз когда декодируемый/кодируемый в данный момент синтаксический элемент synEl является первым синтаксическим элементом тайла 50, сегментом 14 слайса или строкой СТВ, начинается процесс инициализации по фиг.29.

A2) Иначе, происходит декодирование этого синтаксического элемента, используя 15 текущие энтропийные контексты.

A3) Если текущим синтаксическим элементом был последний синтаксический элемент в СТВ 30, тогда начинается процесс сохранения энтропийного контекста, как показано на фиг.30.

A4) Процесс продолжается на A1) со следующим синтаксическим элементом.

20 В процессе инициализации проверяется 200, является ли synEl первым синтаксическим элементом сегмента 14 слайса или тайла 50. Если да, инициализируются контексты независимо от любого предыдущего сегмента слайса на этапе 202. Если нет, проверяется 204, является ли synEl первым синтаксическим элементом строки СТВ 30, и равняется ли **entropy_coding_sync_enabled_flag** единице. Если да, проверяется 206, доступен ли 25 второй СТВ 30 в предыдущей строке СТВ 30 идентичного тайла (см. фиг.23). Если да, на этапе 210 выполняется принятие контекста в соответствии с позицией 40, используя сохраненные в данный момент вероятности контекста для принятия типа 40. Если нет, контексты инициализируются независимо от любого предыдущего сегмента слайса на этапе 202. Если проверка 204 показывает Нет, тогда на этапе 212 проверяется, является 30 ли synEl первым синтаксическим элементом в первом СТВ зависимого сегмента 14 слайса, и равен ли единице **dependent_slice_segment_flag**, и, если да, на этапе 214 выполняется принятие контекста в соответствии с позицией 38, используя сохраненные в данный момент вероятности контекста для принятий типа 38. После любого из этапов 214, 212, 210 и 202, фактически начинается декодирование/кодирование.

35 Зависимые сегменты слайса с **dependent_slice_segment_flag**, равняющимся единице, способствуют, таким образом, дальнейшему уменьшению задержки кодирования/декодирования почти без ухудшения эффективности кодирования.

В процессе сохранения по фиг.30 проверяется на этапе 300, является ли кодируемый/декодируемый synEl последним синтаксическим элементом второго СТВ 30 строки СТВ 40 30, и равняется ли единице **entropy_coding_sync_enabled_flag**. Если да, текущие энтропийные контексты сохраняются на этапе 302, т.е. вероятности энтропийного кодирования контекстов, в запоминающем устройстве, которое является специальным для принятий по типу 40. Аналогично, на этапе 304 проверяется, в дополнение к этапам 300 или 302, является кодируемый/декодируемый synEl последним синтаксическим 45 элементом сегмента 14 слайса, и равняется ли единице **dependent_slice_segment_flag**. Если да, текущие энтропийные контексты сохраняются на этапе 306, т.е. вероятности энтропийного кодирования контекстов, в запоминающем устройстве, которое является специальным для принятий по типу 38.

Отмечается, что любая проверка, запрашивающая, является ли синтаксический элемент первым `synEl` строки СТВ, применяет, например, синтаксический элемент `slice_address 400` в заголовках сегмента слайса, т.е. синтаксический элемент начала, показывающий положения начала соответствующего сегмента слайса в соответствии с порядком декодирования.

При восстановлении изображения 10 из потока 12 данных, используя обработку WPP, декодер способен применять точно последний синтаксический участок 400 начала, чтобы показывать точки входа подпотоков WPP. Так как каждый сегмент слайса содержит синтаксический участок 400 начала, указывающий положение начала декодирования соответствующего сегмента слайса в изображении 10, декодер способен идентифицировать точки входа подпотоков WPP, в которые сгруппированы сегменты слайса, посредством идентификации, используя синтаксические участки 400 начала сегментов слайса, сегментов слайса, начинающихся на левой стороне изображения. Декодер тогда может параллельно декодировать подпотоки WPP ступенчатым образом с последовательным началом декодирования подпотоков WPP в соответствии с порядком слайсов. Сегменты слайса даже могут быть меньше ширины одного изображения, т.е. одной строки СТВ, так что их передача может чередоваться среди подпотоков WPP, чтобы дополнительно уменьшить общую сквозную задержку передачи. Кодер обеспечивает каждый слайс (14) с синтаксическим участком (400) начала, указывающим положение начала кодирования соответствующего слайса в изображении (10) и группирует слайсы в подпотоки WPP, так что для каждого подпотока WPP первый слайс в порядке слайсов начинается на левой стороне изображения. Кодер даже может, сам собой, использовать обработку WPP при кодировании изображения: кодер параллельно кодирует подпотоки WPP ступенчатым образом, последовательно начиная кодирование подпотоков WPP в соответствии с порядком слайсов.

Между прочим, последний аспект использования синтаксических участков начала сегмента слайса в качестве средства для определения расположения точек входа подпотоков WPP, может использоваться без принципа зависимых слайсов.

Является допустимой для всех для параллельной обработки изображения 10 установка вышеупомянутых переменных следующим образом:

Таблица 1

	<code>tiles_enabled_flag == 0</code>	<code>tiles_enabled_flag == 1</code>	<code>entropy_coding_sync_enable_flag == 1</code>	<code>tiles_enabled_flag == 1</code>	<code>entropy_coding_sync_enable_flag == 1</code>
dependent slice segment flag	0/1	0/1	0/1	0/1	0/1
подразделение изображения на тайлы	нет	да	нет	да	нет
подразделение слайса с точками входа для указания начала подпотока WPP или тайла	нет	да	да	нет	нет
параллельная обработка возможна	в слайсах	в тайлах	WPP	в тайлах	WPP

Даже является допустимым смешивать WPP с разбиением тайлов. В этом случае, можно рассматривать тайлы в качестве индивидуальных изображений; каждый, который использует WPP, состоит из слайса, имеющего один или несколько зависимых сегментов

слайса, и проверка на этапе 300 и 208 ссылается на второй СТВ в вышеупомянутой строке СТВ в этом же тайле, точно как этапы 204 и A1 ссылаются на первый СТВ 30 в строке СТВ 30 текущего тайла! В этом случае вышеупомянутая таблица может быть расширена:

	tiles_e nabled _flag	tiles_e nabled _flag= =1	entrop y_coo ding_s ync_en able_fl	tiles_e nabled _flag= =1	entrop y_coo ding_s ync_en able_fl	entrop y_coo ding_s ync_en able_fl	entrop y_coo ding_s ync_en able_fl
			ag ==1		ag ==1	ag ==1	ag ==1
dependent_slice_segmen t_flag	0/1	0/1	0/1	0/1	0/1	0/1	0/1
Подразделение изображения на тайлы	Нет	Да	Нет	Да	Нет	Да	Да
Подразделение слайса с точками входа для указания начала подпотока WPP или тайла	Нет	Да	Да	Нет	Нет	Нет (каждый подпоток тайла в его соб- ственном зависи- мом слайсе)	Да (каждый тайл в отдель- ном слайсе и точки входа для под- потоков WPP в слайсе тайла)
Параллельная обработка возможна	в слайсах	в тайлах	WPP	в тайлах	WPP	под- потоки WPP в тайлах	под- потоки WPP в тайлах

В качестве краткого примечания, последнее расширение также возможно с вариантом 2 осуществления. Вариант 2 осуществления делает возможным следующую обработку:

Таблица 3

	уровень синтаксиса				
tiles_or_entropy_coding_sync_idc	на изображение	0	1	2	3

	cabac_independent_flag	на слайс	не присут- ствует	не присут- ствует	не присут- ствует	0/1
	dependent_slice_flag	на слайс	0/1	0/1	0/1	0/1
5	Подразделение изображения на тайлы		Нет	Да	Нет	Нет
	Подразделение слайса с точками входа для указания начала подпотока WPP или тайла		Нет	Да	Да	Нет
10	Параллельная обработка возможна		в слайсах	в тайлах	WPP	Энтропий- ное коди- рование слайсов

Но со следующими расширениями таблица ниже имеет результатом:

Добавить к семантике набора параметров изображения:

Если `tiles_or_entropy_coding_sync_idc` равен 4, каждый, но первая строка СТВ должна содержаться в другом слайсе с флагом зависимого слайса, установленным на 1. СТВ других строк не должны присутствовать в одном и том же слайсе. Может быть более одного слайса, присутствующего на строку СТВ.

Если `tiles_or_entropy_coding_sync_idc` равен 5, СТВ каждого, кроме первого тайла, должны содержаться в другом слайсе. СТВ других тайлов не должны присутствовать в одном и том же слайсе. Может быть более одного слайса, присутствующего на тайл.

См. фиг.31 для дополнительного объяснения.

Т.е. вышеупомянутая таблица может быть расширена:

		Уровень синтаксиса								
5	tiles_or_entropy_coding_sync_idc	на изображение	0	1	2	3	5	4	6	7
10	cabac_independent_flag	на слайс	не присутствует	не присутствует	не присутствует	0/1	не присутствует	не присутствует	не присутствует	не присутствует
	dependent slice flag	на слайс	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
15	Подразделение изображения на тайлы		Нет	Да	Нет	Нет	Да	Нет	Да	Да
20	Подразделение слайса с точками входа для указания начала подпотока WPP или тайла		Нет	Да	Да	Нет	Нет	Нет	Нет (каждый подпоток тайла в своем собственном зависимом слайсе)	Да (каждый тайл в отдельном слайсе и точки входа для подпотоков WPP в слайсе тайла)
25	Параллельная обработка возможна		в слайсах	в тайлах	WPP	энтропийное кодирование слайсов	в тайлах	WPP	подпотоки WPP в тайлах	подпотоки WPP в тайлах
30										

35 Что касается вышеописанных вариантов осуществления, необходимо отметить, что декодер может быть выполнен с возможностью, например, реагирования на tiles_or_entropy_coding_sync_idc=1,2, в первом и втором режиме, считывания информации из
 40 текущего слайса, показывая подразделение текущего слайса на параллельные подразделы, причем параллельные подразделы могут быть подпотоками WPP или тайлами, прекращения контекстно-адаптивного энтропийного декодирования в конце
 45 первого параллельного подраздела и возобновления вновь контекстно-адаптивного энтропийного декодирования в начале любого последующего параллельного подраздела, в том числе, в первом режиме, инициализацию вероятностей символа в зависимости от сохраненных состояний вероятностей символа предшествующего
 параллельного подраздела, и, во втором режиме, инициализацию вероятностей символа независимо от любого ранее декодированного слайса и любого ранее декодированного параллельного подраздела.

Таким образом, вышеприведенное описание показывает способы кодирования, декодирования, инкапсуляции и передачи с малой задержкой структурированных

видеоданных, предоставляемых новым стандартом кодирования HEVC, таких как структурированные в тайлах, подпотоках волновой параллельной обработки (WPP), слайсах или энтропийных слайсах.

Особенно, было определено, как транспортировать параллельные кодированные данные в разговорном сценарии, чтобы получить минимальную латентность в процессе кодирования, декодирования и передачи. Поэтому, было описан подход конвейерного параллельного кодирования, передачи и декодирования, чтобы сделать возможными приложения с минимальной задержкой, подобные играм, дистанционной хирургии и т.д.

Кроме того, вышеупомянутые варианты осуществления закрыли пробел волновой параллельной обработки (WPP), сделав ее пригодной для применения в сценариях передачи с малой задержкой. Поэтому, был представлен новый формат инкапсуляции для подпотоков 0 WPP, зависимый слайс. Этот зависимый слайс может содержать данные энтропийного слайса, подпоток WPP, полную строку LCU, только фрагмент слайса, где ранее переданный заголовок слайса также применяется к содержащимся данным фрагмента. Содержащиеся данные сигнализируются во вспомогательном заголовке слайса.

Наконец отмечается, что именованием для новых слайсов также может быть «поднабор/легковесные слайсы», но название «зависимый слайс» было найдено лучшим.

Была представлена сигнализация, которая описывает уровень распараллеливания при кодировании и транспортировке.

Хотя некоторые аспекты были описаны в контексте устройства, ясно, что эти аспекты также представляют описание соответствующего способа, где блок или устройство соответствует этапу способа или признаку этапа способа. Аналогично, аспекты, описанные в контексте этапа способа, также представляют описание соответствующего блока или элемента или признака соответствующего устройства. Некоторые или все из этапов способа могут выполняться аппаратным устройством (или использовать его), подобным, например, микропроцессору, программируемому компьютеру или электронной схеме. В некоторых вариантах осуществления некоторый один или несколько из наиболее важных этапов способа могут исполняться таким устройством.

В зависимости от некоторых требований к реализации, варианты осуществления изобретения могут быть реализованы аппаратными или программными средствами. Реализация может выполняться с использованием среды для запоминания цифровых данных, например, дискета, цифровой многофункциональный диск (DVD), диск Blu-Ray, компакт диск (CD), постоянное запоминающее устройство (ROM), программируемое ROM (PROM), стираемое программируемое ROM (EPROM), электрически стираемое программируемое ROM (EEPROM) или флэш-память, имеющие считываемые электронным образом сигналы управления, сохраненные на них, которые взаимодействуют (или способны взаимодействовать) с программируемой компьютерной системой, так что выполняется соответствующий способ. Поэтому, среда для запоминания цифровых данных может быть считываемой компьютером.

Некоторые варианты осуществления согласно изобретению содержат носитель данных, имеющий считываемые электронным образом сигналы управления, которые способны взаимодействовать с программируемой компьютерной системой, так что выполняется один из способов, описанных в данном документе.

Обычно, варианты осуществления настоящего изобретения могут быть реализованы в виде продукта компьютерной программы с программным кодом, причем программный код действует для выполнения одного из способов, когда продукт компьютерной

программы выполняется на компьютере. Программный код, например, может сохраняться на машиночитываемом носителе.

Другие варианты осуществления содержат компьютерную программу для выполнения одного из способов, описанных в данном документе, хранимую на машиночитываемом носителе.

Другими словами, вариант осуществления обладающими признаками изобретения способа, поэтому, представляет собой компьютерную программу, имеющую программный код для выполнения одного из способов, описанных в данном документе, когда компьютерная программа выполняется на компьютере.

Другим вариантом осуществления обладающих признаками изобретения способов, поэтому, является носитель данных (или среда для запоминания цифровых данных или считываемая компьютером среда), содержащий записанную на нем компьютерную программу для выполнения одного из способов, описанных в данном документе. Носитель данных, среда для запоминания цифровых данных или записанная среда обычно являются материальными и/или долговременными.

Другим вариантом осуществления обладающими признаками изобретения способа, поэтому, является поток данных или последовательность сигналов, представляющих компьютерную программу для выполнения одного из способов, описанных в данном документе. Поток данных или последовательность сигналов, например, может быть выполнена с возможностью передачи по соединению для передачи данных, например, по Интернету.

Другой вариант осуществления содержит средство обработки, например, компьютер, или программируемое логическое устройство, выполненное с возможностью или адаптированное для выполнения одного из способов, описанных в данном документе.

Другой вариант осуществления содержит компьютер, имеющий установленную на нем компьютерную программу для выполнения одного из способов, описанных в данном документе.

Другой вариант осуществления согласно изобретению содержит устройство или систему, выполненную с возможностью пересылки (например, электронным или оптическим образом) компьютерной программы для выполнения одного из способов, описанных в данном документе, на приемник. Приемником, например, может быть компьютер, мобильное устройство, устройство памяти или т.п. Устройство или система, например, может содержать файловый сервер для пересылки компьютерной программы на приемник.

В некоторых вариантах осуществления программируемое логическое устройство (например, программируемая вентиляционная матрица) может использоваться для выполнения некоторых или всех функциональных возможностей способов, описанных в данном документе. В некоторых вариантах осуществления программируемая вентиляционная матрица может взаимодействовать с микропроцессором, чтобы выполнять один из способов, описанных в данном документе. Обычно, способы, предпочтительно, выполняются любым аппаратным устройством.

Вышеописанные варианты осуществления являются просто иллюстративными для принципов настоящего изобретения. Понятно, что для специалистов в данной области техники очевидны модификации и варианты устройств и подробности, описанные в данном документе. Поэтому, намерением является то, что ограничивается только объемом находящейся на рассмотрении формулы изобретения и не конкретными подробностями, представленными посредством описания и объяснения вариантов осуществления в данном документе.

Список литературы

- [1] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard", IEEE Trans. Circuits Syst. Video Technol., vol. 13, N7, July 2003.
- [2] JCT-VC, "High-Efficiency Video Coding (HEVC) text specification Working Draft 6", JCTVC-H1003, February 2012.
- [3] ISO/IEC 13818-1: MPEG-2 Systems specification.

(57) Формула изобретения

1. Декодер для восстановления изображения (10) из потока (12) данных, в который изображение кодировано в единицах слайсов (14), на которые изображение (10) разделено, используя WPP-обработку, при этом декодер выполнен с возможностью декодирования слайсов (14) из потока (12) данных в соответствии с порядком (16) слайсов, при этом каждый слайс (14) содержит начальный участок (400) синтаксиса, указывающий позицию начала декодирования соответствующего слайса в изображении (10), и декодер выполнен с возможностью

идентификации точек входа подпотоков WPP, в которые слайсы сгруппированы, посредством идентификации, используя начальные участки (400) синтаксиса слайсов, слайсов, начинающихся на левой стороне изображения,

параллельного декодирования подпотоков WPP ступенчатым образом с последовательным началом декодирования подпотоков WPP в соответствии с порядком слайсов.

2. Декодер по п. 1, при этом поток данных имеет информацию разбиения, которая является различной для яркости и цветности.

3. Кодер для кодирования изображения (10) в поток (12) данных, в который изображение кодируется в единицах слайсов (14), на которые разделяется изображение (10), используя WPP-обработку, при этом кодер выполнен с возможностью кодирования слайсов (14) в поток (12) данных в соответствии с порядком (16) слайсов, при этом кодер выполнен с возможностью обеспечения каждого слайса (14) начальным участком (400) синтаксиса, указывающим позицию начала кодирования соответствующего слайса в изображении (10), и кодер выполнен с возможностью

группировки слайсов в подпотоки WPP так, чтобы для каждого подпотока WPP, первый слайс в порядке слайсов начинался с левой стороны изображения, и

параллельного кодирования подпотоков WPP ступенчатым образом с последовательным началом кодирования подпотоков WPP в соответствии с порядком слайсов.

4. Кодер по п. 3, при этом поток данных имеет информацию разбиения, которая является различной для яркости и цветности.

5. Способ восстановления изображения (10) из потока (12) данных, в который изображение кодировано в единицах слайсов (14), на которые изображение (10) разделено, используя WPP-обработку, при этом способ содержит декодирование слайсов (14) из потока (12) данных в соответствии с порядком (16) слайсов, при этом каждый слайс (14) содержит начальный участок (400) синтаксиса, указывающий позицию начала декодирования соответствующего слайса в изображении (10), и способ дополнительно содержит

идентификацию точек входа подпотоков WPP, в которые слайсы сгруппированы, посредством идентификации, используя начальные участки (400) синтаксиса слайсов,

слайсов, начинающихся на левой стороне изображения,

параллельное декодирование подпотоков WPP ступенчатым образом с последовательным началом декодирования подпотоков WPP в соответствии с порядком слайсов.

- 5 6. Способ кодирования изображения (10) в поток (12) данных, в который изображение кодируется в единицах слайсов (14), на которые разделяется изображение (10), используя WPP-обработку, при этом способ содержит кодирование слайсов (14) в поток (12) данных в соответствии с порядком (16) слайсов, и обеспечение каждого слайса (14) начальным участком (400) синтаксиса, указывающим позицию начала кодирования
- 10 соответствующего слайса в изображении (10), и дополнительно

группировку слайсов в подпотоки WPP так, чтобы для каждого подпотока WPP первый слайс в порядке слайсов начинался с левой стороны изображения, и

- 15 параллельное кодирование подпотоков WPP ступенчатым образом с последовательным началом кодирования подпотоков WPP в соответствии с порядком слайсов.

7. Считываемый компьютером носитель, хранящий компьютерную программу, имеющую программный код для выполнения, при исполнении на компьютере, способа по п. 5.

8. Считываемый компьютером носитель, хранящий компьютерную программу, имеющую программный код для выполнения, при исполнении на компьютере, способа
- 20 по п. 6.

9. Цифровой запоминающий носитель, хранящий поток данных, в который изображение (10) кодируется в единицах слайсов (14), на которые изображение (10) разделяется, при этом слайсы (14) являются декодируемыми, используя WPP-обработку,
- 25 из потока (12) данных в соответствии с порядком (16) слайсов, при этом каждый слайс (14) содержит начальный участок (400) синтаксиса, указывающий позицию начала декодирования соответствующего слайса в изображении (10), так что декодер имеет возможность

- идентификации точек входа подпотоков WPP, в которые слайсы сгруппированы, посредством идентификации, используя начальные участки (400) синтаксиса слайсов, слайсов, начинающихся на левой стороне изображения,
- 30

параллельного декодирования подпотоков WPP ступенчатым образом с последовательным началом декодирования подпотоков WPP в соответствии с порядком слайсов.

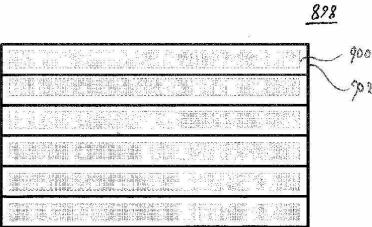
- 35 10. Цифровой запоминающий носитель, хранящий поток данных по п. 9, при этом поток данных имеет информацию разбиения, которая является различной для яркости и цветности.

40

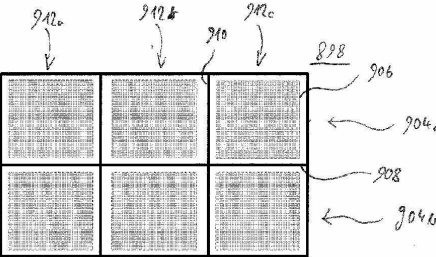
45

1

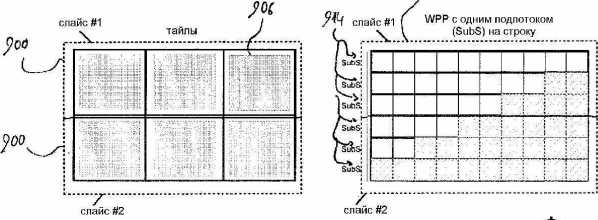
1/21



Фиг.1

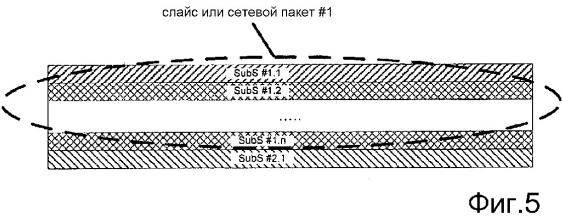
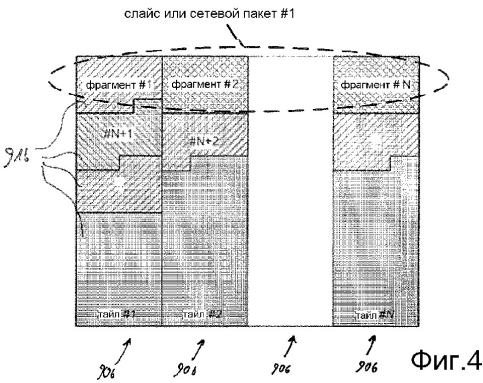


Фиг.2

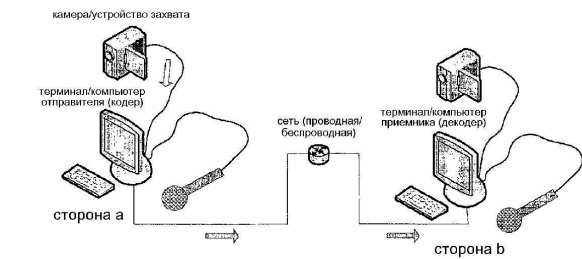


Фиг.3

2



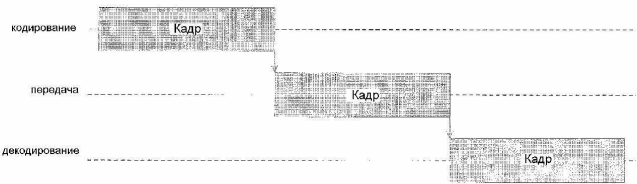
3/21



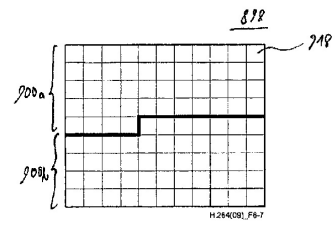
Фиг.6



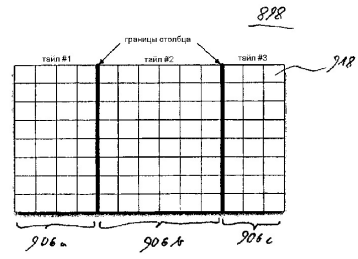
Фиг.7



Фиг.8



Фиг.9



Фиг.10

[illegible]

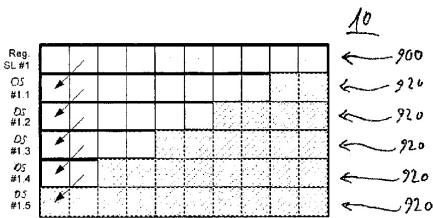
6/21

pic_parameter_set_rbsp() {	дескриптор
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
sign_data_hiding_flag	u(1)
if(sign_data_hiding_flag)	
sign_hiding_threshold	u(4)
cabac_init_present_flag	u(1)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
pic_init_qp_minus26	se(v)
constrained_intra_pred_flag	u(1)
enable_temporal_mvp_flag	u(1)
slice_granularity	u(2)
max_cu_qp_delta_depth	ue(v)
cb_qp_offset	se(v)
cr_qp_offset	se(v)
weighted_pred_flag	u(1)
weighted_bipred_idc	u(2)
output_flag_present_flag	u(1)
if(tiles_or_entropy_coding_sync_idc == 1) {	
tile_info_present_flag	u(1)
tile_control_present_flag	u(1)
if(tile_info_present_flag) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
if(tile_control_present_flag)	
loop_filter_across_tiles_enabled_flag	u(1)
} else if(tiles_or_entropy_coding_sync_idc == 2)	
num_substreams_minus1	ue(v)
deblocking_filter_control_present_flag	u(1)
if(slice_type == P slice_type == B)	
log2_parallel_merge_level_minus2	ue(v)
pps_extension_flag	u(1)
if(pps_extension_flag)	
while(more_rbsp_data())	
pps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	

Фиг.12

Стр.: 52

8/21



Фиг.14

pic_parameter_set_rbsp() {	дескриптор
.....	
dependent_quantisation_flag	0(1)
last_cabac_init_flag	0(1)
}	

Фиг.15

10/21
400
100

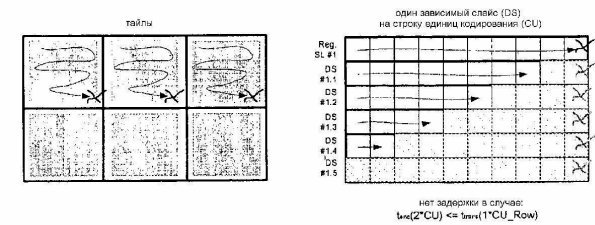
slice_header() {	дескриптор
first_slice_in_pic_flag	u(1)
if (first_slice_in_pic_flag == 0)	
slice_address	u(v)
slice_type	ue(v)
entropy_slice_flag	u(1)
if (entropy_slice_flag)	u(1)
if (dependent_slice_flag)	
if (dependent_slice_flag) {	
...	
} else {	
...	
if (cabac_init_present_flag && slice_type != I && !cabac_reset_flag)	
cabac_init_flag	u(1)
...	
if (slice_type == P slice_type == B)	
five_minus_max_num_merge_cand	ue(v)
if (adaptive_loop_filter_enabled_flag) {	
slice_adaptive_loop_filter_flag	u(1)
if (slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
alf_param()	
if (slice_adaptive_loop_filter_flag && talf_coef_in_slice_flag)	
alf_cu_control_param()	
}	
if (seq_loop_filter_across_slices_enabled_flag && (slice_adaptive_loop_filter_flag slice_sample_adaptive_offset_flag !disable_deblocking_filter_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
if (tiles_or_entropy_coding_sync_idc > 0) {	
num_entry_point_offsets	ue(v)
if (num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for (i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	

Фиг.16

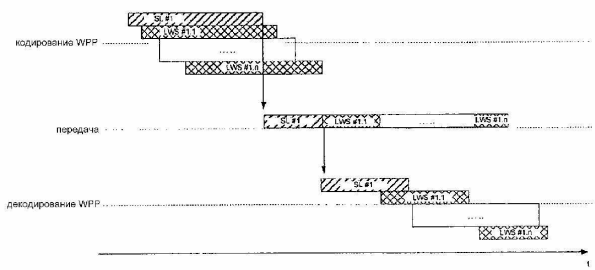


Фиг.17

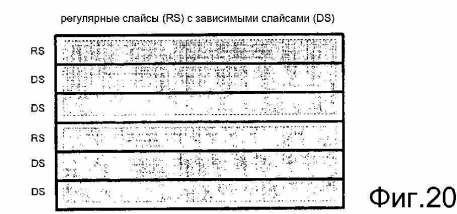
12/21



Фиг.18



Фиг.19



Фиг.20

13/21

4 06 102

slice_header() {	дескриптор
first_slice_in_pic_flag	u(1)
pic_parameter_set_id	ue(v)
if (first_slice_in_pic_flag)	
slice_address	u(v)
if (dependent_slice_enabled_flag && !first_slice_in_pic_flag)	
dependent_slice_flag	ue(1)
dependent_slice_flag	ue(1)
slice_type	ue(v)
if (output_flag_present_flag)	
pic_output_flag	u(1)
if (separate_colour_plane_flag == 1)	
colour_plane_id	u(2)
if (RapPicFlag) {	
rap_pic_id	ue(v)
no_output_of_prior_pics_flag	u(1)
}	
if (!IdrPicFlag) {	
if (tiles_or_entropy_coding_sync_idc == 1 	
tiles_or_entropy_coding_sync_idc == 2) {	
num_entry_point_offsets	ue(v)
if (num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for (i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	
if (slice_header_extension_present_flag) {	
slice_header_extension_length	ue(v)
for (i = 0; i < slice_header_extension_length; i++)	
slice_header_extension_data_byte	u(8)
}	
byte_alignment()	
}	

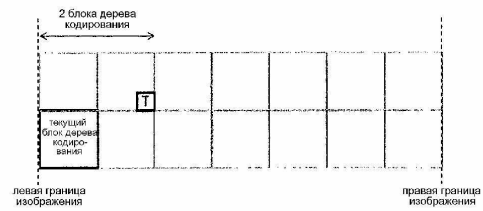
Фиг.21

14/21

pic_parameter_set_rbsp() {		дескриптор
pic_parameter_set_id	ue(v)	
seq_parameter_set_id	ue(v)	
sign_data_hiding_flag	u(1)	
if sign_data_hiding_flag {		
sign_hiding_threshold	u(4)	
cabac_init_present_flag	u(1)	
num_ref_idx_l0_default_active_minus1	ue(v)	
num_ref_idx_l1_default_active_minus1	ue(v)	
[Ed. (BD): not present in HM software]		
pic_init_qp_minus26	se(v)	
constrained_intra_pred_flag	u(1)	
slice_granularity	u(2)	
diff_cu_qp_delta_depth	ue(v)	
cb_qp_offset	se(v)	
cr_qp_offset	se(v)	
weighted_pred_flag	u(1)	
weighted_bipred_idc	u(2)	
output_flag_present_flag	u(1)	
transquant_bypass_enable_flag	u(1)	
dependent_residual_scaling_flag	u(1)	
tiles_enabled_flag	u(1)	
tile_rows_minus1	u(1)	
tile_cols_minus1	u(1)	
t_mvp_enabled_flag	u(1)	
num_tile_columns_max	u(8)	
num_tile_rows_max	u(8)	
uniform_spacing_flag	u(1)	
chroma_qp_offset_list	u(4)	
loop_filter_params_disabled_flag	u(1)	
luma_filter_param_present_flag	u(1)	
luma_filter_kernel_size	u(8)	
luma_filter_coefficients	u(16)	
row_height	ue(v)	
col_width	ue(v)	
block_interpolation_chroma_filters	u(4)	
disable_cabac_reinit_flag	u(1)	
enable_intra_block_copy	u(1)	
...		

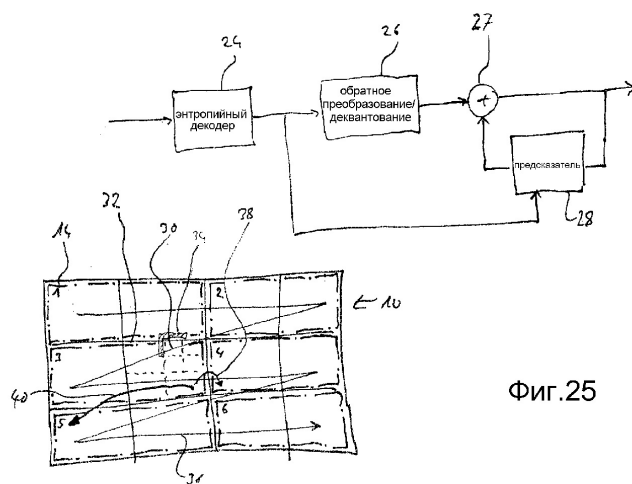
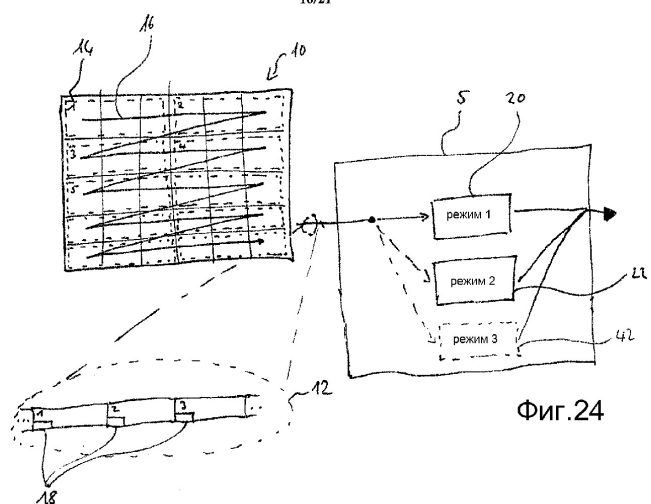
Фиг.22

15/21

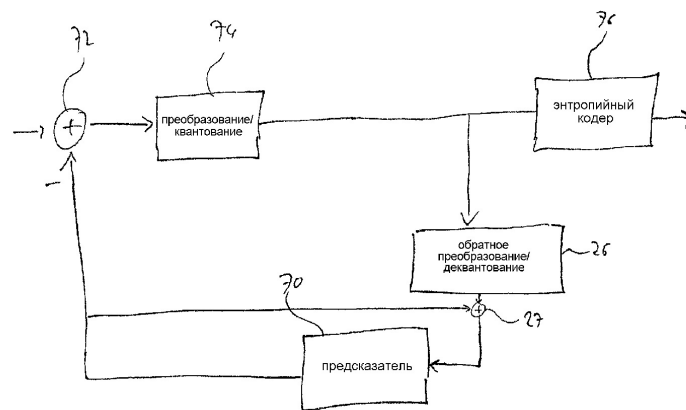


Фиг.23

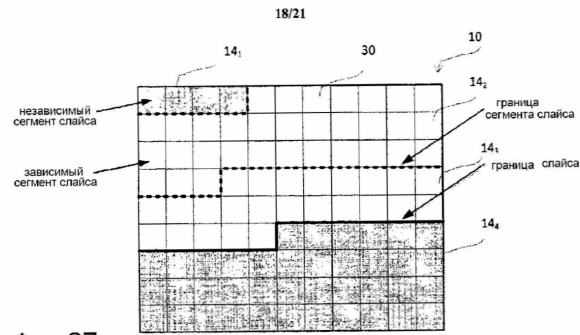
16/21



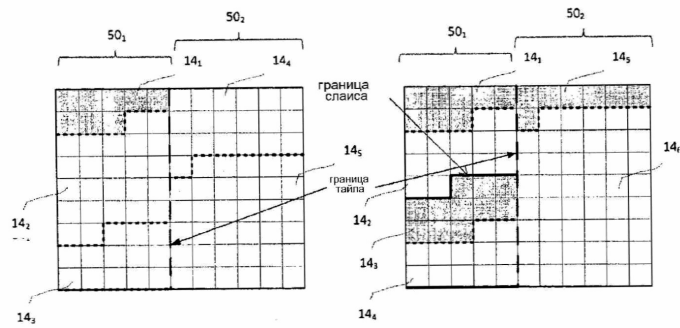
17/21



Фиг.26



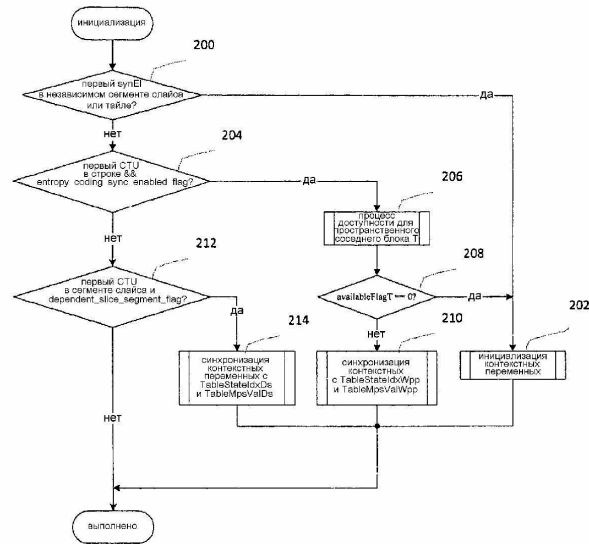
Фиг.27



Фиг.28a

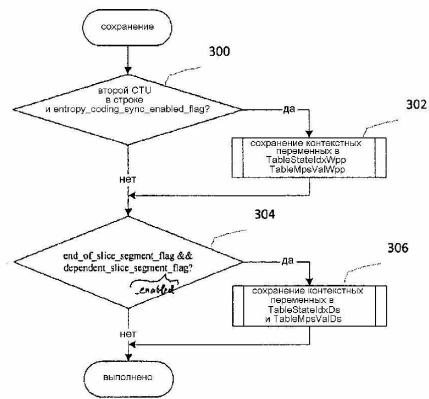
Фиг.28b

19/21



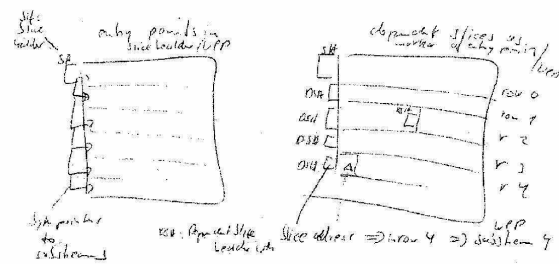
Фиг.29

20/21



Фиг.30

21/21



Фиг.31