РОССИЙСКАЯ ФЕДЕРАЦИЯ

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(19) **RU** (11) **2 746 981** (13) **C2**

(51) МПК
H04N 19/61 (2014.01)
H04N 19/44 (2014.01)

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК
H04N 19/61 (2021.02); H04N 19/44 (2021.02)

(21)(22) Заявка: **2020122372, 29.06.2018**

(24) Дата начала отсчета срока действия патента:
29.06.2018

Дата регистрации:
22.04.2021

Приоритет(ы):
(30) Конвенционный приоритет:
29.06.2017 US 62/526,577;
21.09.2017 US 62/561,561;
12.02.2018 US 62/629,313;
05.06.2018 US 62/680,710;
19.06.2018 US 62/686,738
Номер и дата приоритета первоначальной заявки, из которой данная заявка выделена:
2019125713 29.06.2017

(43) Дата публикации заявки: **24.09.2020** Бюл. № 27

(45) Опубликовано: **22.04.2021** Бюл. № 12

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

(72) Автор(ы):
ЛУ, Таожань (US),
ПУ, Фанцзюнь (US),
ИНЬ, Пэн (US),
ЧЭНЬ, Тао (US),
ХЬЮСЭК, Уолтер Дж. (US)

(73) Патентообладатель(и):
ДОЛБИ ЛЭБОРЕТЕРИЗ ЛАЙСЕНСИНГ КОРПОРЕЙШН (US)

(56) Список документов, цитированных в отчете о поиске: **WO 2016/164235 A1, 13.10.2016. WO 2012/122425 A1, 13.09.2012. US 2006/0268991 A1, 30.11.2006. US 2013/0208998 A1, 15.08.2013. RU 2370908 C2, 20.10.2009.**

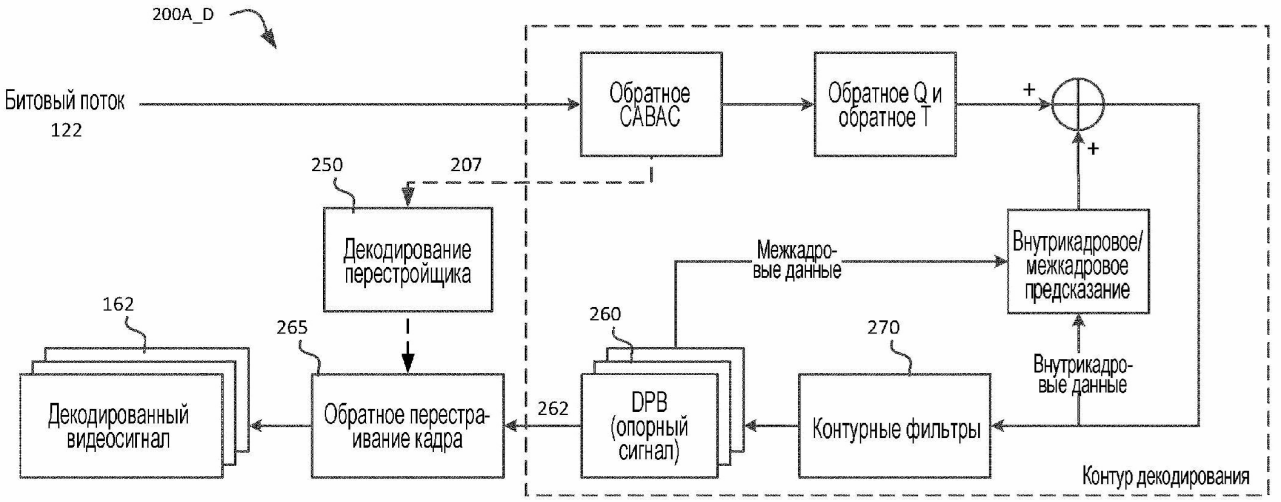
(54) СОВМЕСТНОЕ ПЕРЕСТРАИВАНИЕ ИЗОБРАЖЕНИЯ И КОДИРОВАНИЕ ВИДЕОСИГНАЛА

(57) Реферат:

Изобретение относится к вычислительной технике. Технический результат заключается в повышении эффективности кодирования при перестраивании. Способ декодирования кодированного битового потока для формирования выходного изображения, в котором принимают кодированный битовый поток, содержащий параметры отображения для отображения кодированных пикселей из перестроенного представления кодовой комбинацией в выходное представление кодовой комбинацией; осуществляют доступ к кодированному изображению в кодированном битовом потоке посредством пикселей в перестроенном представлении кодовой комбинацией; извлекают упомянутые параметры отображения, чтобы сформировать функцию прямого перестраивания, причем функция прямого перестраивания отображает пиксели из выходного представления кодовой комбинацией в перестроенное представление кодовой комбинацией, и функцию обратного перестраивания, причем функция обратного перестраивания отображает пиксели из перестроенного представления кодовой комбинацией в выходное представление кодовой

RU 2746981 C2
RU 2746981 C2

комбинацией; и декодируют кодированное изображение, чтобы сформировать выходное изображение в выходном представлении кодовой комбинацией. 3 н. и 7 з.п. ф-лы, 23 ил., 21 табл.



ФИГ. 2B

RU 2746981 C2

RU 2746981 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 746 981** (13) **C2**

(51) Int. Cl.
H04N 19/61 (2014.01)
H04N 19/44 (2014.01)

(12) ABSTRACT OF INVENTION

(52) CPC
H04N 19/61 (2021.02); H04N 19/44 (2021.02)

(21)(22) Application: **2020122372, 29.06.2018**

(24) Effective date for property rights:
29.06.2018

Registration date:
22.04.2021

Priority:
(30) Convention priority:
29.06.2017 US 62/526,577;
21.09.2017 US 62/561,561;
12.02.2018 US 62/629,313;
05.06.2018 US 62/680,710;
19.06.2018 US 62/686,738
Number and date of priority of the initial application, from which the given application is allocated:
2019125713 29.06.2017

(43) Application published: **24.09.2020** Bull. № **27**

(45) Date of publication: **22.04.2021** Bull. № **12**

Mail address:
129090, Moskva, ul. B. Spasskaya, 25, str. 3, OOO "Yuridicheskaya firma Gorodisskij i Partnery"

(72) Inventor(s):
LU, Taoran (US),
PU, Fangjun (US),
YIN, Peng (US),
CHEN, Tao (US),
HUSAK, Walter J. (US)

(73) Proprietor(s):
DOLBY LABORATORIES LICENSING CORPORATION (US)

RU 2746981 C2

RU 2746981 C2

(54) JOINT IMAGE REARRANGEMENT AND VIDEO ENCODING

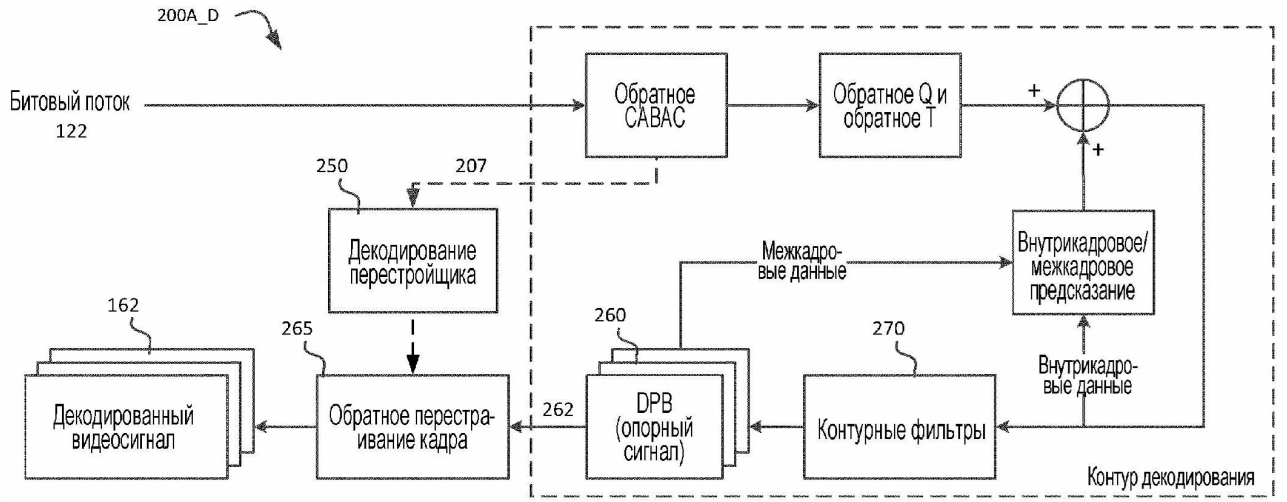
(57) Abstract:

FIELD: computing.

SUBSTANCE: invention relates to computing. Disclosed is a method for decoding an encoded bitstream to generate an output image. An encoded bitstream is received. It contains display parameters displaying the encoded pixels from the rearranged code combination representation in the output code combination representation. After that one should gain access to the encoded image in the bitstream encoded by means of pixels in the rearranged code combination representation. It's necessary to extract the display parameters to generate a forward rearranging function. The forward rearranging function displays pixels from the output code combination representation to the rearranged code combination representation and an inverse rearranging function. The inverse rearranging function displays pixels from the rearranged code combination representation to the output code combination representation. After that the encoded picture is decoded to generate an output picture in the output code combination representation.

EFFECT: improving coding efficiency in the process of rearrangement.

10 cl, 23 dwg, 21 tbl



ФИГ. 2B

RU 2746981 C2

RU 2746981 C2

ПЕРЕКРЕСТНАЯ ССЫЛКА НА РОДСТВЕННЫЕ ЗАЯВКИ

[0001] Данная заявка испрашивает приоритет по предварительной заявке на выдачу патента США под порядковым номером 62/686,738, поданной 19 июня 2018 года; под порядковым номером 62/680,710, поданной 5 июня 2018 года; под порядковым номером 62/629,313, поданной 12 февраля 2018 года; под порядковым номером 62/561,561, поданной 21 сентября 2017 года; под порядковым номером 62/526,577, поданной 29 июня 2017 года, каждая из которых включена в материалы настоящей заявки посредством ссылки во всей своей полноте.

ТЕХНОЛОГИЯ

[0002] Настоящее изобретение в целом относится к кодированию изображений и видеосигнала. Конкретнее, вариант осуществления настоящей заявки относится к совместному перестраиванию изображения и кодированию видеосигнала.

УРОВЕНЬ ТЕХНИКИ

[0003] В 2013, группа MPEG в Международной организации по стандартизации (ISO), совместно с Международным союзом телекоммуникаций (ITU), выпустили первый проект документа стандарта кодирования видеосигнала HEVC (также известного как H.265, высокоэффективное кодирование видеосигналов). Позже та же самая группа выпустила требование данных для поддержки развития стандарта кодирования следующего поколения, который дает улучшенные эксплуатационные качества кодирования над существующими технологиями кодирования видеосигнала.

[0004] В качестве используемого в материалах настоящей заявки, термин 'битовая глубина' обозначает количество пикселей, используемых для представления одной из цветовых компонент изображения. Традиционно изображения кодировались в 8 битах, на цветовую компоненту, для каждого пикселя (например, 24 бита на каждый пиксель); однако, современные архитектуры сейчас могут поддерживать большие битовые глубины, такие как 10 битов, 12 битов или более.

[0005] В традиционном конвейере изображений, захваченные изображения квантуются с использованием линейной оптоэлектронной функции (OETF), которая преобразует линейное освещение места действия в нелинейный видеосигнал (например, кодированный степенью контрастности RGB или YCbCr). Затем, на приемнике, перед отображением на устройстве отображения, сигнал обрабатывается функцией электрооптической передачи (EOTF), которая преобразует значения видеосигнала в интенсивности цвета экрана вывода. Такие нелинейные функции включают в себя традиционную кривую «степени контрастности», документированную в протоколах BT.709 и BT.2020 ITU-R, и кривую «PQ» (перцепционного квантования), описанную в ST 2084 SMPTE и протоколе BT.2100 ITU-R.

[0006] В качестве используемого в материалах настоящей заявки, термин «прямое перестраивание» обозначает процесс отображения из отсчета в отсчет или из кодовой комбинации в кодовую комбинацию цифрового изображения из его исходной битовой глубины и исходных классификации или представления (например, степени контрастности или PQ, и тому подобного) кодовых комбинаций в изображение с той же самой или другой битовой глубиной и другими классификацией или представлением кодовыми комбинациями. Перестраивание предусматривает улучшенную возможность сжатия или улучшенное качество изображения при постоянной битовой скорости (битрейте). Например, без ограничения, перестраивание может применяться к кодированному 10-битным или 12-битным PQ видеосигналу HDR для улучшения эффективности кодирования в архитектуре кодирования 10-битного видеосигнала. В приемнике, после восстановления от сжатия перестроенного сигнала, приемник может

применять «функцию обратного перестраивания» для восстановления сигнала в его исходную классификацию кодовой комбинацией. Как в данном документе принимается во внимание изобретателями, в то время как начинается развертывание для следующего поколения стандарта кодирования видеосигнала, требуются улучшенные технологии для совместного перестраивания и кодирования изображений. Способы по данному изобретению могут быть применимы к многообразию видеоконтента, в том числе, но не в качестве ограничения, к контенту в стандартном динамическом диапазоне (SDR) и/или расширенном динамическом диапазоне (HDR).

[0007] Подходы, описанные в данном разделе, являются подходами, которые могли бы быть осуществлены, но не обязательно подходами, которые были задуманы или осуществлены ранее. Поэтому, если не указано иное, не должно предполагаться, что какой бы то ни было из подходов, описанных в данном разделе, квалифицируется в качестве прототипа всего лишь на основе своего включения в данный раздел. Подобным образом, обсуждаемые темы в отношении одного или более подходов не должны восприниматься распознанными в каком бы то ни было прототипе на основе этого раздела, если не указано иное.

КРАТКОЕ ОПИСАНИЕ ЧЕРТЕЖЕЙ

[0008] Вариант осуществления настоящего изобретения проиллюстрирован в качестве примера, а не в качестве ограничения, на фигурах прилагаемых чертежей, и на которых одинаковые номера позиций указывают ссылкой на сходные элементы, и на которых:

[0009] Фиг. 1A изображает примерный процесс для конвейера поставки видеоданных;

[0010] фиг. 1B изображает примерный процесс для сжатия данных с использованием перестраивания сигнала согласно предшествующему уровню техники;

[0011] Фиг. 2A изображает примерную архитектуру для кодировщика, использующего нормативное внеконтурное перестраивание согласно варианту осуществления данного изобретения;

[0012] фиг. 2B изображает примерную архитектуру для декодера, использующего нормативное внеконтурное перестраивание, согласно варианту осуществления данного изобретения;

[0013] фиг. 2C изображает примерную архитектуру для кодировщика, использующего нормативное внутриконтурное перестраивание только с внутрикадровым предсказанием, согласно варианту осуществления данного изобретения;

[0014] фиг. 2D изображает примерную архитектуру для декодера, использующего нормативное внутриконтурное перестраивание только с внутрикадровым предсказанием, согласно варианту осуществления данного изобретения;

[0015] фиг. 2E изображает примерную архитектуру для кодировщика, использующего внутриконтурное перестраивание для остаточных значений предсказания, согласно варианту осуществления данного изобретения;

[0016] фиг. 2F изображает примерную архитектуру для декодера, использующего внутриконтурное перестраивание для остаточных значений предсказания, согласно варианту осуществления данного изобретения;

[0017] фиг. 2G изображает примерную архитектуру для кодировщика, использующего гибридное внутриконтурное перестраивание, согласно варианту осуществления данного изобретения;

[0018] фиг. 2H изображает примерную архитектуру для декодера, использующего гибридное внутриконтурное перестраивание, согласно варианту осуществления данного изобретения;

[0019] фиг. 3A изображает примерный процесс для кодирования видеосигнала с

использованием архитектуры внеконтурного перестраивания согласно варианту осуществления данного изобретения;

[0020] фиг. 3B изображает примерный процесс для декодирования видеосигнала с использованием архитектуры внеконтурного перестраивания согласно варианту осуществления данного изобретения;

[0021] фиг. 3C изображает примерный процесс для кодирования видеосигнала с использованием архитектуры внутриконтурного перестраивания только с внутрикадровым предсказанием согласно варианту осуществления данного изобретения;

[0022] фиг. 3D изображает примерный процесс для декодирования видеосигнала с использованием архитектуры внутриконтурного перестраивания только с внутрикадровым предсказанием согласно варианту осуществления данного изобретения;

[0023] фиг. 3E изображает примерный процесс для кодирования видеосигнала с использованием архитектуры внутриконтурного перестраивания для остаточных значений предсказания согласно варианту осуществления данного изобретения;

[0024] фиг. 3F изображает примерный процесс для декодирования видеосигнала с использованием архитектуры внутриконтурного перестраивания для предсказания остаточных значений согласно варианту осуществления данного изобретения;

[0025] фиг. 4A изображает примерный процесс для кодирования видеосигнала с использованием комбинации из трех основанных на перестраивании архитектур согласно варианту осуществления данного изобретения;

[0026] фиг. 4B изображает примерный процесс для декодирования видеосигнала с использованием комбинации из трех основанных на перестраивании архитектур согласно варианту осуществления данного изобретения;

[0027] фиг. 5A и фиг. 5B изображают процесс реконструкции функции перестраивания в видеодекодере согласно варианту осуществления данного изобретения;

[0028] фиг. 6A и фиг. 6B изображают примеры того, каким образом значения сдвига QP цветности изменяются согласно параметру квантования (QP) яркости для кодированных с помощью PQ и HLG сигналов согласно варианту осуществления данного изобретения; и

[0029] фиг. 7 изображает пример основанного на повороте представления функции перестраивания согласно варианту осуществления данного изобретения.

ОПИСАНИЕ ПРИМЕРНЫХ ВАРИАНТОВ ОСУЩЕСТВЛЕНИЯ

[0030] В материалах настоящей заявки описаны нормативные технологии внеконтурного и внутриконтурного совмещенного перестраивания и кодирования сигнала для сжатия изображений. В нижеследующем описании, в целях пояснения, многие конкретные детали изложены для того, чтобы обеспечить исчерпывающее понимание настоящего изобретения. Однако, будет очевидно, что настоящее изобретение может быть осуществлено на практике без этих конкретных деталей. В других случаях, широко известные конструкции и устройства не описаны с исчерпывающей подробностью, для того чтобы избежать ненужного закрывания, затенения или запутывания настоящего изобретения.

Обзор

[0031] Примерные варианты осуществления, описанные в материалах настоящей заявки, относятся к совместному перестраиванию и кодированию сигнала для видеосигнала. В кодировщике, процессор принимает входное изображение в первом представлении кодовой комбинацией, представленном входной битовой глубиной N и входным отображением кодовой комбинации (например, степенью контрастности, PQ, и тому подобным). Процессор выбирает архитектуру кодировщика (с перестройщиком,

являющимся неотъемлемой частью кодировщика) из двух или более потенциально подходящих архитектур кодировщика для сжатия входного изображения с использованием второго представления кодовой комбинацией, предусматривающего более эффективное сжатие, чем первое представление кодовой комбинацией, при этом, две или более потенциально подходящих архитектур кодировщика содержат архитектуру внеконтурного перестраивания, архитектуру внутриконтурного перестраивания только для кадров с внутрикадровым предсказанием или внутриконтурную архитектуру для остаточных значений предсказания, и процессор сжимает входное изображение согласно выбранной архитектуре кодировщика.

[0032] В еще одном варианте осуществления, декодер для формирования выходных изображений в первом представлении кодовой комбинацией принимает кодированный битовый поток с по меньшей мере частью кодированных изображений, являющихся сжатыми во втором представлении кодовой комбинацией. Он также принимает связанную информацию о перестраивании. Процессор принимает сигнализацию, указывающую архитектуру декодера из двух или более потенциально подходящих архитектур декодера для восстановления от сжатия входного кодированного битового потока, при этом две или более потенциально подходящих архитектуры декодера содержат архитектуру внеконтурного перестраивания, архитектуру внутриконтурного перестраивания только для кадров с внутрикадровым предсказанием или внутриконтурную архитектуру для остаточных значений предсказания, и он восстанавливает от сжатия кодированное изображение для формирования выходного изображения согласно принятой архитектуре перестраивания.

[0033] В еще одном варианте осуществления, в кодировщике для сжатия изображений согласно внутриконтурной архитектуре для остаточных значений предсказания, процессор осуществляет доступ к входному изображению в первом представлении кодовой комбинацией и формирует функцию прямого перестраивания, отображающую пиксели входного изображения из первого представления кодовой комбинацией во второе представление кодовой комбинацией. Он формирует функцию обратного перестраивания на основе функции прямого перестраивания, отображающую пиксели из второго представления кодовой комбинацией в пиксели в первом представлении кодовой комбинацией. Затем, применительно к области входных пикселей во входном изображении: он

вычисляет по меньшей мере одну предсказанную область на основе пиксельных данных в буфере опорного кадра или кодированных ранее пространственно соседних элементов;

формирует перестроенную область остаточных значений на основе области входных пикселей, предсказанной области и функции прямого перестраивания;

формирует кодированную (преобразованную и квантованную) область остаточных значений на основе перестроенной области остаточных значений;

формирует декодированную (обратно квантованную и обратно преобразованную) область остаточных значений на основе кодированной области остаточных значений;

формирует область восстановленных пикселей на основе декодированной области остаточных значений, предсказанной области, функции прямого перестраивания и функции обратного перестраивания; и

формирует область опорных пикселей, подлежащую сохранению в буфере опорного кадра, на основе области восстановленных пикселей.

[0034] В еще одном варианте осуществления, в декодере для формирования выходных изображений в первом представлении кодовой комбинацией согласно внутриконтурной

архитектуре для остаточных значений предсказания, процессор принимает кодированный битовый поток, частично кодированный во втором представлении кодовой комбинацией. Он также принимает связанную информацию о перестраивании. Процессор формирует, на основе информации о перестраивании, функцию прямого перестраивания, которая отображает пиксели из первого представления кодовой комбинацией во второе представление кодовой комбинацией, и функцию обратного перестраивания, при этом функция обратного перестраивания отображает пиксели из второго представления кодовой комбинацией в первое представление кодовой комбинацией. Применительно к области кодированного изображения, процессор:

формирует декодированную область остаточных значений на основе кодированного изображения;

формирует предсказанную область на основе пикселей в буфере опорных пикселей или декодированных ранее пространственно соседних элементов;

формирует область восстановленных пикселей на основе декодированной перестроенной области остаточных значений, предсказанной области, функции прямого перестраивания и функции обратного перестраивания;

формирует область выходных пикселей на основе области восстановленных пикселей; и сохраняет область выходных пикселей в буфере опорных пикселей.

Примерный конвейер обработки для поставки видеосигнала

[0035] Фиг. 1A изображает примерный процесс традиционного конвейера (100) поставки видеоданных, показывающий различные стадии от захвата видеоданных до отображения видеоконтента. Последовательность видеокадров (102) захватывается или формируется с использованием блока (105) формирования изображений. Видеокадры (102) могут захватываться в цифровом виде (например, цифровой камерой) или формироваться компьютером (например, с использованием компьютерной анимации) для выдачи видеоданных (107). В качестве альтернативы (видеокадры (102) могут сниматься на пленку пленочной камерой. Пленка преобразуется в цифровой формат для выдачи видеоданных (107). На фазе (110) производства, видеоданные (107) монтируются для выдачи потока (112) видеопродукции.

[0036] Видеоданные потока (112) продукции затем выдаются в процессор на блок (115) для редакционного окончательного монтажа. Блок (115) редакционного окончательного монтажа может включать в себя коррекцию или модификацию цветов или яркости в конкретных зонах изображения для улучшения качества изображения или достижения конкретного внешнего вида для изображения в соответствии творческим намерением автора видеоматериала. Это иногда называется «синхронизацией цветовой поднесущей» или «цветокоррекцией». Другой монтаж (например, выбор и упорядочение сцены, обрезка изображения, добавление сформированных компьютером специальных визуальных эффектов, и т. д.) может выполняться в блоке (115), чтобы давать окончательный вариант (117) продукции для классификации. Во время редакционного окончательного монтажа, видеоизображения просматриваются на эталонном устройстве (115) отображения.

[0037] Вслед за окончательным монтажом (115), видеоданные конечной продукции могут доставляться в кодирующий блок (120) для поставки битового потока в устройства декодирования и воспроизведения, такие как телевизоры, телевизионные абонентские приставки, кинотеатры, и тому подобное. В некоторых вариантах осуществления, кодирующий блок (120) может включать в себя кодировщики звукового сигнала и видеосигнала, такие как определенные форматами ATSC, DVB, DVD, Blu-Ray и другими форматами поставки, для формирования кодированного битового потока (122). В

приемнике, кодированный битовый поток (122) декодируется декодирующим блоком (130) для формирования декодированного сигнала (132), представляющего собой идентичный или приемлемое приближение сигнала (117). Приемник может быть прикреплен к целевому устройству (140) отображения, которое может иметь совершенно другие характеристики, нежели эталонное устройство (125) отображения. В таком случае, блок (135) управления отображением может использоваться для приведения динамического диапазона декодированного сигнала (132) в соответствие характеристикам целевого устройства (140) отображения посредством формирования приведенного в соответствие устройству отображения сигнала (137).

Перестраивание сигнала

[0038] Фиг. 1B изображает примерный процесс для перестраивания сигнала согласно справочному материалу [1] предшествующего уровня техники. При данных входных кадрах (117), блок (150) прямого перестраивания анализирует ограничения ввода и кодирования и формирует функции отображения кодовых комбинаций, которые отображают входные кадры (117) в переквантованные выходные кадры (152). Например, входные кадры (117) могут быть закодированы согласно определенной функции электрооптической передачи (EOTF) (например, степени контрастности). В некоторых вариантах осуществления, информация о процессе перестраивания может передаваться на находящиеся ниже по потоку устройства (такие как декодеры) с использованием метаданных. В качестве используемого в материалах настоящей заявки, термин «метаданные» относится к любой вспомогательной информации, которая передается в качестве части кодированного битового потока и помогает декодеру воспроизводить декодированное изображение. Такие метаданные могут включать в себя, но не в качестве ограничения, информацию о цветовом пространстве или гамме, параметры эталонного устройства отображения и вспомогательные параметры сигнала, как описанные в материалах настоящей заявки.

[0039] Вслед за кодированием (120) и декодированием (130), декодированные кадры (132) могут обрабатываться функцией (160) обращенного (или обратного) перестраивания, которая преобразует переквантованные кадры (132) обратно в исходную область (например, степень контрастности) EOTF, для дальнейшей низовой обработки, такой как процесс (135) управления отображением, обсужденный раньше. В некоторых вариантах осуществления, функция (160) обращенного перестраивания может быть объединена с деквантователем в декодере (130), например, в качестве части деквантователя в декодере видеосигнала AVC или HEVC.

[0040] В качестве используемого в материалах настоящей заявки, термин «перестройщик» может обозначать функцию прямого или обратного перестраивания, подлежащую использованию при кодировании и/или декодировании цифровых изображений. Примеры функций перестраивания обсуждены в справочных материалах [1] и [2]. В целях данного изобретения, предполагается, что специалист в данной области техники может получить пригодные функции прямого и обратного перестраивания согласно характеристикам входного видеосигнала и имеющейся в распоряжении битовой глубины архитектур кодирования и декодирования.

[0041] В справочном материале [1], был предложен способ внутриконтурного основанного на блоках перестраивания изображения для кодирования видеосигнала с расширенным динамическим диапазоном. Такое исполнение предоставляет возможность основанного на блоках перестраивания в пределах контура кодирования, но за счет повышенной сложности. Чтобы быть более точным, исполнение требует поддержки двух наборов буферов декодированного изображения: одного набора для

подвергнутых обратному перестраиванию (или неперестроенных) декодированных кадров, которые могут использоваться как для предсказания без перестраивания, так и для вывода на устройство отображения, и другого набора для подвергнутых прямому перестраиванию декодированных кадров, которые используются только для предсказания с перестраиванием. Хотя подвергнутые прямому перестраиванию декодированные кадры могут вычисляться на лету, стоимость сложности очень высока, особенно для межкадрового предсказания (компенсации движения с интерполяцией подпикселями). Вообще, управление буфером кадров отображения (DPB) усложняется и требует очень пристального внимания, таким образом, как понимается изобретателями, требуются упрощенные способы для кодирования видеосигнала.

[0042] Варианты осуществления архитектур основанного на перестраивании кодека, представленные в материалах настоящей заявки, могут быть разделены, как изложено ниже: архитектура с внешним внеконтурным перестройщиком, архитектура с внутриконтурным перестройщиком только с внутрикадровым предсказанием и архитектура с внутриконтурным перестройщиком для остаточных значений предсказания, также подлежащая указанию ссылкой, ради краткости, как 'внутриконтурный перестройщик для остаточных значений'. Кодировщик или декодер видеосигнала могут поддерживать любую одну из этих архитектур или их комбинацию. Каждая из этих архитектур также может применяться сама по себе или в комбинации с любой одной из остальных. Каждая архитектура может применяться к компоненте яркости, компоненте цветности или комбинации компоненты яркости и одной или более компонент цветности.

[0043] В дополнение к этим трем архитектурам, дополнительные варианты осуществления описывают эффективные способы сигнализации для метаданных, связанных с перестраиванием, и несколько основанных на кодировщике средств оптимизации для улучшения эффективности кодирования, когда применяется перестраивание.

Нормативный внеконтурный перестройщик

[0044] Фиг. 2A и фиг. 2B изображают архитектуры для кодировщика (200A_E) видеосигнала и соответствующего декодера (200A_D) видеосигнала с «нормативным» внеконтурным перестройщиком. Термин «нормативный» обозначает, что, в отличие от предыдущих исполнений, где перестраивание считалось этапом предварительной обработки, таким образом находясь вне нормативного описания стандарта кодирования, такого как AVC, HEVC, и тому подобное, в данном варианте осуществления, прямое и обратное перестраивание являются часть нормативных требований. В отличие от архитектуры по фиг. 1B, где соответствие битового потока требованиям согласно стандарту, испытывается после декодирования (130), на фиг. 2B, соответствие требованиям проверяется после блока (265) обратного перестраивания (например, на выходе 162 на фиг. 1B).

[0045] В кодировщике (200A_E), два новых блока добавлены в традиционный основанный на блоках кодировщик (например, HEVC): блок (205) для оценки функции прямого перестраивания и блок (210) прямого перестраивания кадра, который применяет прямое перестраивание к одной или более из цветовых компонент входного видеосигнала (117). В некоторых вариантах осуществления, эти две операции могут выполняться в качестве части единого блока перестраивания изображений. Параметры (207), связанные с определением функции обратного перестраивания в декодере, могут пересылаться в блок кодировщика без потерь кодировщика видеосигнала (например, CABAC 220), так что они могут быть встроены в кодированный битовый поток (122). Все операции,

связанные с внутрикадровым или межкадровым предсказанием (225), преобразованием и квантованием (T & Q), обратным преобразованием и деквантованием (Q^{-1} & T^{-1}), а также контурной фильтрацией, выполняются с использованием перестроенных кадров, хранимых в DPB (215).

[0046] В декодере (200A_D), два новых нормативных блока добавлены в традиционный основанный на блоках декодер: блок (250) для реконструкции функции обратного перестраивания на основе кодированных параметров (207) функции перестраивания, и блок (265) для применения функции обратного перестраивания к декодированным данным (262), чтобы формировать декодированный видеосигнал (162). В некоторых вариантах осуществления, операции, связанные с блоками 250 и 265 могут быть объединены в единый обрабатывающий блок.

[0047] Фиг. 3A изображает примерный процесс (300A_E) для кодирования видеосигнала с использованием архитектуры (200A_E) внеконтурного перестраивания согласно варианту осуществления данного изобретения. Если перестраивание не активировано (ветвь 305), то кодирование продолжает движение, как известно в кодировщиках предшествующего уровня техники (например, HEVC). Если перестраивание активировано (ветвь 310), то кодировщик может иметь варианты выбора применить предварительно заданную (установленную по умолчанию) функцию (315) перестраивания или адаптивно определять новую функцию (325) перестраивания на основе анализа (320) кадра (например, как описано в справочных материалах [1]-[3]). Вслед за прямым перестраиванием (330), оставшаяся часть кодирования придерживается традиционного конвейера (335) кодирования. Если применяется адаптивное перестраивание (312), метаданные, связанные с функцией обратного перестраивания, формируются в качестве части этапа (327) «Кодировать перестройщик».

[0048] Фиг. 3B изображает примерный процесс (300A_D) для декодирования видеосигнала с использованием архитектуры (200A_D) внеконтурного перестраивания согласно варианту осуществления данного изобретения. Если перестраивание не активировано (ветвь 355), то, после декодирования кадра (350), выходные кадры формируются (390), как в традиционном конвейере декодирования. Если перестраивание активировано (ветвь 360), то, на этапе (370), декодер определяет, применять ли предварительно заданную (установленную по умолчанию функцию (375) перестраивания или дополнительно определить функцию (380) обратного перестраивания на основе принятых параметров (например, 207). Вслед за обратным перестраиванием (385), оставшаяся часть декодирования придерживается традиционного конвейера декодирования.

Нормативный внутриконтурный перестройщик только с внутрикадровым предсказанием

[0049] Фиг. 2C изображает примерную архитектуру для кодировщика (200B_E), использующего нормативное внутриконтурное перестраивание только с внутрикадровым предсказанием, согласно варианту осуществления данного изобретения. Исполнение полностью аналогично исполнению, предложенному в справочном материале [1]; однако, для снижения сложности, особенно в том, что относится к использованию памяти DPB (215 и 260), только кадры с внутрикадровым предсказанием кодируются с использованием данной архитектуры.

[0050] По сравнению с внеконтурным перестраиванием (200A_E), основное отличие кодировщика 200B_E состоит в том, что DPB (215) хранит обратно перестроенные кадры вместо перестроенных кадров. Другими словами, необходимо, чтобы декодированные кадры с внутрикадровым предсказанием подвергались обратному

перестраиванию (блоком 265 обратного перестраивания) перед сохранением в DPB. Аргументация за этим подходом состоит в том, что, если кадры с внутрикадровым предсказанием кодируются с помощью перестраивания, улучшенные эксплуатационные качества кодирования кадров с внутрикадровым предсказанием будут репродуцироваться, чтобы также (косвенным образом) улучшать кодирование кадров с межкадровым предсказанием, даже если кадры с межкадровым предсказанием кодируются без перестраивания. Таким образом, можно использовать в своих интересах перестраивание, не имея дела со сложностью внуктриконтурного перестраивания для кадров с межкадровым предсказанием. Поскольку обратное перестраивание (265) является частью внутреннего контура, оно может осуществляться до внутриконтурного фильтра (270). Преимущество добавления обратного перестраивания до внутриконтурного фильтра состоит в том, что, в этом случае, исполнение внутриконтурного фильтра может быть оптимизировано на основе характеристик исходных кадров вместо подвергнутых прямому перестраиванию кадров.

[0051] Фиг. 2D изображает примерную архитектуру для декодера (200B_D), использующего нормативное внутриконтурное перестраивание только с внутрикадровым предсказанием, согласно варианту осуществления данного изобретения. Как изображено на фиг. 2D, определение функции (250) обратного перестраивания и применение обратного перестраивания (265) теперь выполняются раньше внутриконтурной фильтрации (270).

[0052] Фиг. 3C изображает примерный процесс (300B_E) для кодирования видеосигнала с использованием архитектуры внутриконтурного перестраивания только с внутрикадровым предсказанием согласно варианту осуществления данного изобретения. Как изображено, поток операций на фиг. 3C совместно использует многие элементы с потоком операций на фиг. 3A. Далее, по умолчанию, перестраивание не применяется для кодирования с межкадровым предсказанием. Что касается кодированных с внутрикадровым предсказанием кадров, если перестраивание активировано, кодировщик вновь имеет вариант выбора использовать установленную по умолчанию кривую перестраивания ли применять адаптивное перестраивание (312). Если кадр перестроен, обратное перестраивание (385) является частью процесса, и связанные параметры кодируются на этапе (327). Соответствующий процесс (300B_D) декодирования изображен на фиг. 3D.

[0053] Как изображено на фиг. 3D, связанные с перестраиванием операции задействуются только для принятых кадров с внутрикадровым предсказанием, и только если перестраивание с внутрикадровым предсказанием применялось в кодировщике.

Внутриконтурный перестройщик для остаточных значений предсказания

[0054] При кодировании, термин 'остаточные значения' обозначает несовпадение между предсказанием отсчета или элемента данных и его исходным или декодированным значением. Например, при заданном исходном отсчете из входного видеосигнала (117), обозначенном как *Orig_sample*, внутрикадровое или межкадровое предсказание (225) может формировать соответствующий предсказанный отсчет (227), обозначенный как *Pred_sample*. Если перестраивания нет, неперестроенное остаточное значение (*Res_u*) может быть определено в виде

$$Res\_u = Orig\_sample - Pred\_sample. \quad (1)$$

[0055] В некоторых вариантах осуществления, может быть полезно применять перестраивание к области остаточных значений. Фиг. 2E изображает примерную архитектуру для кодировщика (200C_E), использующего внутриконтурное перестраивание для предсказанных остаточных значений, согласно варианту

осуществления данного изобретения. Пусть *Fwd*() обозначает функцию прямого перестраивания, и пусть *Inv*() обозначает соответствующую функцию обратного перестраивания. В варианте осуществления, перестроенное остаточное значение (232) может быть определено как

$$Res\_r = Fwd(Orig\_sample) - Fwd(Pred\_sample). \quad (2)$$

[0056] Соответственно, на выходе (267) обращенного перестройщика (265), восстановленный отсчет, обозначенный как *Reco_sample* (267), может быть выражен в виде

$$Reco\_sample = Inv(Res\_d + Fwd(Pred\_sample)), \quad (3)$$

где *Res_d* представляет собой остаточное значение (234), приемлемое приближение *Res_r*, после внутриконтурного кодирования и декодирования в 200C_E.

[0057] Отметим, что, хотя перестраивание применяется к остаточным значениям, фактические входные пиксели видеоданных не перестроены. Фиг. 2F изображает соответствующий декодер (200C_D). Отметим, что, как изображено на фиг. 2F, и на основе уравнения (3), декодеру требуется доступ к обеим функциям, прямого и обратного перестраивания, которые могут извлекаться с использованием принятых метаданных (207) и блока (250) «Декодирование перестройщика».

[0058] В варианте осуществления, для снижения сложности, уравнения (2) и (3) могут быть упрощены. Например, при условии, что функция прямого перестраивания может быть аппроксимирована кусочно-линейной функцией, и что абсолютная разность между *Pred_sample* и *Orig_sample* относительно мала, в таком случае, уравнение (2) могло бы быть приближенно выражено в виде

$$Res\_r = a(Pred\_sample) * (Orig\_sample - Pred\_sample), \quad (4)$$

где *a*(*Pred_sample*) обозначает масштабный коэффициент, основанный на значении *Pred_sample.* Из уравнений (3) и (4), уравнение (3) может быть приближенно выражено в виде

$$Reco\_sample = Pred\_sample + (1/a(Pred\_sample)) * Res\_r, \quad (5)$$

Таким образом, в варианте осуществления, нужно передавать в декодер только масштабные коэффициенты *a*(*Pred_sample*) для кусочно-линейной модели.

[0059] Фиг. 3E и фиг. 3F изображают примерные потоки операций процесса для кодирования (300C_E) и декодирования (300C_D) видеосигнала с использованием внутриконтурного перестраивания остаточных значений предсказания. Процессы полностью аналогичны описанным на фиг. 3A и 3B и, таким образом, не требуют пояснений.

[0060] Таблица обобщает ключевые признаки трех предложенных архитектур.

Таблица 1: Ключевые признаки для рассматриваемых архитектур перестраивания

Архитектура	Внеконтурное	Внутриконтурное только с внутрикадровым предсказанием	Внутриконтурное для остаточных значений
Хранение DPB	перестроенные кадры	режим внутрикадрового предсказания: обратно перестроенные кадры режим межкадрового предсказания: нет перестраивания	неперестроенные кадры
Внутрикадровое предсказание, выполняемое над	перестроенные кадры	перестроенные кадры	неперестроенные кадры
Межкадровое предсказание (оценка движения) выполняемое над	перестроенные кадры	неперестроенные кадры	неперестроенные кадры
Необходим дополнительный буфер кадров	да (необходим буфер для хранения перестроенных кадров в DPB и неперестроенных кадров для вывода)	нет (замена отсчетов кадра на лету)	нет (замена отсчетов остаточных значений на лету)

Место/частота адаптивной оценки перестраивания	не ограничены (могут быть только с внутрикадровым предсказанием, основанными на сцене или конфигурируемыми)	только над кадрами с внутрикадровым предсказанием	не ограничены (могут быть только с внутрикадровым предсказанием, основанными на сцене или конфигурируемыми)
Сложность для модификации (перестраивания) отсчетов	обрабатывать все кадры	обрабатывать только кадры с внутрикадровым предсказанием (низшая сложность)	обрабатывать остаточные значения, внутрикадровое или межкадровое предсказание безразлично
Взаимодействие с контурным фильтром	оптимизация с использованием перестроенного кадра в качестве опорного	оптимизация с использованием исходного кадра в качестве опорного	оптимизация с использованием исходного кадра в качестве опорного
Возможно адаптивное перестраивание на уровне блока/области	нет	нет	да
Другие аспекты	эксплуатационные качества внутрикадрового предсказания могут страдать, если опорные кадры имеют разные функции перестраивания		межкадровое предсказание может пользоваться перестройщиком для текущего кадра для обработки остаточных значений по отношению к опорным кадрам (сами которые могут иметь разные перестройщики)
	стороне декодера необходимо всего лишь обратить функцию перестраивания	стороне декодера нужна только функция обратного перестраивания	декодеру нужны обе функции, прямого и обратного перестраивания

[0061] Фиг. 4A и фиг. 4B изображают примерные потоки обработки кодирования и декодирования для кодирования и декодирования с использованием комбинации трех предложенных архитектур. Как изображено на фиг. 4A, если перестраивание не активировано, входной видеосигнал кодируются согласно известным технологиям кодирования видеосигнала (например, HEVC, и тому подобному) без использования какого бы то ни было перестраивания. Иначе, кодировщик может выбирать любой один из трех основных предложенных способов в зависимости от возможностей целевого приемника и/или входных характеристик. Например, в варианте осуществления, кодировщик мог бы переключаться между этими способами на уровне сцены, где 'сцена' обозначена в виде последовательности непрерывных кадров с аналогичными характеристиками яркости. В еще одном варианте осуществления, высокоуровневые параметры определены на уровне набора параметров последовательности (SPS).

[0062] Как изображено на фиг. 4B, декодер, в зависимости от принимаемой сигнализации информации о перестраивании, может активизировать один из соответствующих процессов декодирования для декодирования входящего кодированного битового потока.

Гибридное внутриконтурное перестраивание

[0063] Фиг. 2G изображает примерную архитектуру (200D_E) для кодировщика, использующего архитектуру гибридного внутриконтурного перестраивания. Эта архитектура комбинирует элементы обеих архитектур, внутриконтурного перестраивания (200B_E) только с внутрикадровым предсказанием и внутриконтурной для остаточных значений (200C_E), обсужденных ранее. При данной архитектуре, секции с внутрикадровым предсказанием кодируются согласно архитектуре кодирования с внутриконтурным перестраиванием и внутрикадровым предсказанием (например, 200B_E на фиг. 2C), за исключением одного отличия: что касается секций, обратное перестраивание (265-1) кадров выполняется после контурной фильтрации (270-1). В еще одном варианте осуществления, внутриконтурная фильтрация для секций с внутрикадровым предсказанием может выполняться после обратного перестраивания; однако, экспериментальные результаты показали, что такое устройство может давать худшую эффективность кодирования, чем когда обратное перестраивание выполняется после контурной фильтрации. Остальные операции остаются такими же, как обсужденные ранее.

[0064] Секции (слайсы) с межкадровым предсказанием кодируются согласно

архитектуре внутриконтурного кодирования остаточных значений (например, 200C_E на фиг. 2E), как обсуждено раньше. Как изображено на фиг. 2G, переключатель секций с внутрикадровым/межкадровым предсказанием предоставляет возможность переключения между двумя архитектурами в зависимости от типа секции, подлежащего кодированию.

[0065] Фиг. 2H изображает примерную архитектуру (200D_D) для декодера, использующего гибридное внутрикадровое перестраивание. Вновь, секции с внутрикадровым предсказанием согласно архитектуре декодера с внуриконтурным перестраиванием и внутрикадровым предсказанием (например, 200B_D на фиг. 2D), где вновь, что касается секций с внутрикадровым предсказанием, контурная фильтрация (270-1) предшествует обратному перестраиванию (265-1) кадров. Секции с межкадровым предсказанием декодируются согласно архитектуре внутриконтурного декодирования для остаточных значений (например, 200C_D на фиг. 2F). Как изображено на фиг. 2H, переключатель секций с внутрикадровым/межкадровым предсказанием предоставляет возможность переключения между двумя архитектурами в зависимости от типов секций в кодированных кадрах видеоданных.

[0066] Фиг. 4A может быть без труда расширена, чтобы также включать в себя способ кодирования с гибридным внутриконтурным перестраиванием, посредством активизации процесса 300D-E кодирования, изображенного на фиг. 2G. Подобным образом, фиг. 4B может быть без труда расширена, чтобы также включать в себя способ декодирования с гибридным внутриконтурным перестраиванием, посредством активизации процесса 300D-D декодирования, изображенного на фиг. 2H.

Перестраивание на уровне секции

[0067] Варианты осуществления настоящего изобретения предусматривают многообразия адаптаций уровня секции. Например, для сокращения вычислений, перестраивание может быть активировано только для секций с внутрикадровым предсказанием или только для секций только с межкадровым предсказанием. В еще одном варианте осуществления, перестраивание может разрешаться на основе значения временного идентификатора (например, переменной TemporalId из HEVC (справочный материал [11]), где TemporalId=**nuh_temporal_id_plus1**-1). Например, если TemporalId для текущей секции является меньшим, чем или равным предварительно заданному значению, то **slice_reshaper_enable_flag** для текущей секции может быть установлен в 1, иначе, **slice_reshaper_enable_flag** будет иметь значение 0. Чтобы избегать отправки параметра **slice_reshaper_enable_flag** для каждой секции, можно задавать параметр **sps_reshaper_temporal_id** на уровне SPS, таким образом, его значение может подразумеваться.

[0068] Что касается секций, где активировано перестраивание, необходимо, чтобы декодер знал какая модель перестраивания подлежит использованию. В одном из вариантов осуществления, он может всегда пользоваться моделью перестраивания, определенной на уровне SPS. В еще одном варианте осуществления, он может всегда пользоваться моделью перестраивания, определенной в заголовке секции. Если модель перестраивания не определена в текущей секции, то он может применять модель перестраивания, используемую в самой последней декодированной секции, которая использовала перестраивание. В еще одном варианте осуществления, модель перестраивания может всегда задаваться в секциях с внутрикадровым предсказанием безотносительно того, используется или нет перестаривание для секции с внутрикадровым предсказанием. В такой реализации, необходимо, чтобы параметры **slice_reshaper_enable_flag** и **slice_reshaper_model_present_flag** были разъединены. Пример

такого синтаксиса секции изображен в таблице 5.

Сигнализация информации о перестраивании

[0069] Информация, имеющая отношение к прямому и/или обратному перестраиванию может присутствовать на разных информационных уровнях, например, в наборе параметров видеосигнала (VPS), наборе параметров последовательности (SPS), наборе параметров кадра (PPS), заголовке секции (SEI) или любом другом высокоуровневом синтаксисе. В качестве примера и без ограничения, таблица 2 предоставляет пример высокоуровневого синтаксиса в SPS для сигнализации того, активировано ли перестраивание, является или нет перестраивание адаптивным, и какая из трех архитектур используется.

Таблица 2: Пример информации о перестраивании в SPS

SPS()	Descriptor
……	
sps_reshaper_enable_flag /*1: перестраивание включено, иначе отключено */	u(1)
if (sps_reshaper_enable_flag) {	
sps_reshaper_adaptive_flag /* 1: адаптивное перестраивание включено, иначе выключено */	u(1)
sps_reshaper_architecture /*например: 0: внеконтурная, 1: внутриконтурная с внутрикадровым предсказанием, 2: внеконтурная для остаточных значений*/	ue(v)
}	
……	

[0070] Дополнительная информация также может переноситься на некотором другом уровне, скажем, в заголовке секции. Функции перестраивания могут описываться справочными таблицами (LUT), сплайнами или другими видами параметрических моделей. Тип модели перестраивания, используемой для сообщения функций перестраивания, может сигнализироваться дополнительными синтаксическими элементами например, флагом **reshaping_model_type**. Например, рассмотрим системы, которая использует два отдельных представления: **model_A** (например, reshaping_model_type=0) представляет собой функцию перестраивания в виде заданной сплайнами (например, смотрите справочный материал [4]), тогда как в модели **model_B** (например, reshaping_model_type=1) функция перестраивания выводится адаптивно посредством назначения кодовых комбинаций на разные полосы яркости на основе характеристик яркости кадра и зрительной важности (например, смотрите справочный материал [3]). Таблица 3 приводит примерные синтаксические элементы в заголовке секции кадра для помощи декодеру определять надлежащую используемую модель перестраивания.

Таблица 3: Примерный синтаксис для сигнализации перестраивания в заголовке секции

slice_segment_header()	Descriptor
……	
if (sps_reshaper_adaptive_flag) {	
reshaping_model_type	ue(v)
if (reshaping_model_type == model_A) {	
reshaping_sliceheader_table_model_A()	
}	
else if (reshaping_model_type == model_B) {	
reshaping_sliceheader_table_model_B()	
}	
else …	
}	
……	

[0071] Нижеследующие три таблицы описывают альтернативные примеры синтаксиса битового потока для перестраивания сигнала на уровнях последовательности, секции или узла кодового дерева (CTU).

Таблица 4: Пример информации о перестраивании в SPS

SPS()	Descriptor
……	
sps_reshaper_enable_flag /*1: перестраивание включено, иначе отключено */	u(1)
if (sps_reshaper_enable_flag) {	
sps_reshaper_signal_type /* 0:SDR, 1:PQ, 2:HLG */	u(2)
sps_reshaper_ILF_opt /* контурный фильтр в котором область: 2 бита с внешним/внутренним кодированием */	u(2)
sps_reshaper_chromaAdj /* 1: chromaDQP; 2: масштабирование цветности/	u(2)
sps_reshaper_model_present_flag /*1: присутствует*/	u(1)
if (sps_reshaper_model_present_flag)	
sps_reshaper_model ()	
}	

Таблица 5: Примерный синтаксис для сигнализации перестраивания в заголовке секции

slice_header()	Descriptor
……	
slice_reshaper_model_present_flag	u(1)
if (slice_reshaper_model_present_flag)	
slice_reshaper_model ( )	
slice_reshaper_enable_flag	u(1)
if (slice_reshaper_enable_flag) {	
reshaper_CTU_control_flag /*1: включено, флаг включения/выключения на уровне CTU*/_	u(1)
}	
……	

Таблица 6: Примерная сигнализация для сигнализации перестраивания в CTU

coding_tree_unit()	Descriptor
……	
if (reshape_CTU_control_flag) {	
reshaper_CTU_flag	ae(v)
}	
……	

[0072] Применительно к таблицам 4-6, примерная семантика может быть обозначена в виде:

sps_reshaper_enable_flag, равный 1, задает, что перестройщик используется в кодированной видеопоследовательности (CVS). sps_reshaper_enabled_flag, равный 0, задает, что перестройщик не используется в CVS.

slice_reshaper_enable_flag, равный 1, задает, что перестройщик активирован для текущей секции. **slice_reshaper_enable_flag**, равный 0, задает, что перестройщик не активирован для текущей секции.

sps_reshaper_signal_type указывает исходные классификацию или представление кодовыми комбинациями. В качестве примера и без ограничения, sps_reshaper_signal_type, равный 0, задает SDR (степень контрастности); sps_reshaper_signal_type, равный 1, задает PQ; а sps_reshaper_signal_type, равный 2 задает HLG.

reshaper_CTU_control_flag, равный 1, указывает, что перестройщику предоставлена возможность адаптироваться для каждого CTU. **reshaper_CTU_control_flag**, равный 0,

указывает, что перестройщику не разрешено адаптироваться для каждого CTU. Когда reshaper_CUT_control_flag отсутствует, значение будет подразумеваться 0.

reshaper_CTU_flag, равный 1, задает, что перестройщик используется для текущего CTU. reshaper_CUT_flag, равный 0, задает, что перестройщик не используется для текущего CTU. Когда reshaper_CTU_flag отсутствует, значение будет подразумеваться равным slice_reshaper_enabled_flag.

sps_reshaper_model_present_flag, равный 1, указывает, что sps_reshaper_model() присутствует в SPS. **sps_reshaper_model_present_flag**, равный 0, указывает, что sps_reshaper_model() отсутствует в SPS.

slice_reshaper_model_present_flag, равный 1, указывает, что slice_reshaper_model() присутствует в заголовке секции. slice_reshaper_model_present_flag, равный 0, указывает, что slice_reshaper_model() отсутствует в SPS.

sps_reshaper_chromaAdj, равный 1, указывает, что коррекция QP цветности выполняется с использованием chromaDQP. sps_reshaper_chromaAdj, равный 2, указывает, что коррекция QP цветности выполняется с использованием масштабирования цветности.

sps_reshaper_ILF_opt указывает, должен ли внутриконтурный фильтр применяться в исходной области или перестроенной области для секций с внутрикадровым или межкадровым предсказанием. Например, с использованием двухбитного синтаксиса, где младший бит указывает ссылкой на секции с внутрикадровым предсказанием:

sps_reshaper_ILF_opt	Операции входного контурного фильтра
0 0	В исходной области как для внутрикадрового, так и для межкадрового предсказания
0 1	В исходной области для внутрикадрового предсказания, в перестроенной области для внутрикадрового предсказания
1 0	В перестроенной области для внутрикадрового предсказания, в исходной области для внутрикадрового предсказания
1 1	В перестроенной области как для внутрикадрового, так и для межкадрового предсказания

[0073] В некоторых вариантах осуществления, данный параметр может настраиваться на уровне секции Например, в варианте осуществления, секция может включать в себя **slice_reshape_ILFOPT_flag**, когда **slice_reshaper_enable_flag** установлен в 1. В еще одном варианте осуществления, в SPS, можно включать в состав параметр **sps_reshaper_ILF_Tid**, если активирован **sps_reshaper_ILF_opt**. Если TemporalID для текущей секции <= **sps_reshaper_ILF_Tid**, и **slice_reshaper_enable_flag** установлен в 1, то внутриконтурный фильтр применяется в области перестраивания. Иначе, он применяется в неперестроенной области.

[0074] В таблице 4, коррекция QP цветности управляется на уровне SPS. В варианте осуществления, коррекция QP цветности также может управляться на уровне секции. Например, в каждой секции можно добавлять синтаксический элемент **slice_reshape_chromaAdj_flag**, когда **slice_reshaper_enable_flag** установлен в 1. В еще одном варианте осуществления, в SPS можно добавлять синтаксический элемент sps_**reshaper_ChromaAdj_Tid**, если активирован **sps_reshaper_chromaAdj**. Если TemporalID для текущей секции <= **sps_reshaper_ChromaAdj_Tid**, и **slice_reshaper_enable_flag** установлен в 1, то применяется коррекция цветности. Иначе, коррекция цветности не применяется. Таблица 4B изображает примерный вариант таблицы 4 с использованием синтаксиса, описанного ранее.

Таблица 4B: Примерный синтаксис для сигнализации перестраивания в SPS с использованием временных идентификаторов

SPS()	Descriptor
……	

sps_reshaper_enable_flag /*1: перестраивание включено, иначе отключено */	u(1)
if (sps_reshaper_enable_flag) {	
sps_reshaper_signal_type /* 0:HDR, 1:PQ, 2:HLG */	u(2)
sps_reshaper_ILF_opt /* контурный фильтр в котором область: 2 бита с внешним/внутренним кодированием */	u(2)
if (sps_reshaper_ILF_opt == 3)	
sps_reshaper_ILF_Tid	u(3)
sps_reshaper_chromaAdj /* 1: chromaDQP; 2: chromaScaling/	u(2)
if (sps_reshaper_chromaAdj)	
sps_reshaper_chromaAdj_Tid	u(3)
sps_reshaper_model_present_flag /*1: присутствует*/	u(1)
if (sps_reshaper_model_present_flag)	
sps_reshaper_model ()	
}	

sps_reshaper_ILF_Tid задает наивысший TemporalID там, где внутриконтурный фильтр применяется для перестроенной секции в перестроенной области.

sps_reshaper_chromaAdj_Tid задает наивысший TemporalID, применительно к которому коррекция цветности применяется для перестроенной секции.

[0075] В еще одном варианте осуществления, модель перестраивания может быть определена с использованием идентификатора модели перестраивания, например, **reshape_model_id,** в качестве части функции slice_reshape_model(). Модель перестраивания может сигнализироваться на уровнях SPS, PPS или заголовка секции. Если сигнализируется в SPS или PPS, значение reshape_model_id также может подразумеваться из **sps_seq_parameter_set_id** или **pps_pic_parameter_set_id**. Пример того, каким образом использовать **reshape_model_id** для секций, которые не несут slice_reshape_model() (например, slice_reshaper_model_present_flag, равный 0) показан ниже в таблице 5B, варианте таблицы 5.

Таблица 5B: Примерный синтаксис для сигнализации перестраивания в заголовке секции с использованием reshape_model_id

slice_header()	Descriptor
……	
slice_reshaper_model_present_flag	u(1)
if (slice_reshaper_model_present_flag)	
slice_reshaper_model ( )	
else	
reshape_model_id	ue(v)
slice_reshaper_enable_flag	u(1)
if (slice_reshaper_enable_flag) {	
reshaper_CTU_control_flag /*1: включено, флаг включения/выключение на уровне CTU*/	u(1)
}	
……	

В примерном синтаксисе, параметр **reshape_model_id** задает значение для используемой reshape_model. Значение reshape_model_id будет находиться в диапазоне от 0 до 15.

[0076] В качестве примера использования предложенного синтаксиса, рассмотрим сигнал HDR, закодированный с использованием EOTF PQ, где перестраивание используется на уровне SPS, специальное перестраивание не используется на уровне секции (перестраивание используется для всех секций), и адаптация CTU разрешена только для секций с внутрикадровым предсказанием. В таком случае:

sps_reshaper_signal_type=1 (PQ);

sps_reshaper_model_present_flag=1;
// Примечание: Можно манипулировать slice_reshaper_enable_flag для активации и деактивации перестройщика.
slice_reshaper_enable_flag=1;
if (CTUAdp)
{
if (I_slice)
slice_reshaper_model_present_flag=0;
reshaper_CTU_control_flag=0;
else
slice_reshaper_model_present_flag=0;
reshaper_CTU_control_flag=1;
}
else
{
slice_reshaper_model_present_flag=0;
reshaper_CTU_control_flag=0;
}

[0077] В еще одном примере, рассмотрим сигнал SDR, где перестраивание применяется только на уровне секции и только для секций с внутрикадровым предсказанием. Адаптация перестраивания CTU предусмотрена только для секций с внутрикадровым предсказанием. В таком случае:

sps_reshaper_signal_type=0 (SDR);
sps_reshaper_model_present_flag=0;
slice_reshaper_enable_flag=1;
if (I_slice)
{
slice_reshaper_model_present_flag=1;
reshaper_CTU_control_flag=0;
}
else
{
slice_reshaper_model_present_flag=0;
if (CTUAdp)
reshape_CTU_control_flag=1;
else
reshaper_CTU_control_flag=0;
}

[0078] На уровне CTU, в варианте осуществления, перестраивание на уровне CTU может активироваться на основе характеристик яркости CTU. Например, применительно к каждому CTU, можно вычислять среднюю яркость (например, CTU_avg_lum_value), сравнивать ее с одним или более пороговых значений и решать, следует или нет включить или отключить перестраивание, на основе результатов таких сравнений. Например,

если CTU_avg_lum_value < THR1, или
если CTU_avg_lum_value > THR2, или
если THR3<CTU_avg_lum_value<THR4,
то reshaper_CTU_Flag=1 для этого CTU.

В варианте осуществления, вместо использования средней яркости, можно

пользоваться некоторой другой характеристикой яркости CTU, такой как минимальная, максимальная или средняя яркость, изменчивость, и тому подобное. Также можно применять основанные на цветности характеристики CTU, или можно комбинировать характеристики яркости и цветности и пороговые значения.

[0079] Как описано раньше (например, в отношении этапов по фиг. 3A, 3B и 3C), варианты осуществления могут поддерживать как установленную по умолчанию или статическую функцию перестраивания, так и адаптивное перестраивание. «Установленный по умолчанию перестройщик» может использоваться для выполнения предварительно заданной функции перестраивания, по этой причине понижая сложность для анализа каждого кадра или сцены при получении кривой перестраивания. В этом случае, нет необходимости в сигнализации функции обратного перестраивания на уровне сцены, кадра или секции. Установленный по умолчанию перестройщик может быть реализован посредством использования постоянной кривой отображения, хранимой в декодере, для избегания какой бы то ни было сигнализации, или он может сигнализироваться один раз в виде части набора параметров уровня последовательности. В еще одном варианте осуществления, декодированная ранее функция адаптивного перестраивания могла бы использоваться повторно для более поздних кадров в порядке кодирования. В еще одном варианте осуществления, кривые перестраивания могут сигнализироваться разностным методом по отношению к декодированным ранее. В других вариантах осуществления (например, применительно к внутриконтурному перестраиванию остаточных значений, где обе функции *Inv*() и *Fwd*() нужны для выполнения обратного перестраивания) можно было бы сигнализировать в битовом потоке только об одной из функций *Inv*() или *Fwd*(), либо, в качестве альтернативы, об обеих, для уменьшения сложности декодера. Таблицы 7 и 8 приводят два примера для сигнализации информации о перестраивании.

[0080] В таблице 7, функция перестраивания сообщается в виде набора полиномов второго порядка. Это упрощенный синтаксис пробной опытной модели (ETM) (справочный материал [5]). Более ранний вариант также может быть найден в справочном материале [4].

Таблица 7: Примерный синтаксис для пофрагментного представления функции перестраивания (модели A)

reshaping_sliceheader_table_model_A() {	Descriptor
reshape_input_luma_bit_depth_minus8	ue(v)
coeff_log2_offset_minus2	ue(v)
reshape_num_ranges_minus1	ue(v)
reshape_equal_ranges_flag	u(1)
reshape_global_offset_val	u(v)
if(!reshape_equal_ranges_flag)	
for (i=0; i < reshape_num_ranges_minus1+ 1; i++)	
reshape_range_val[i]	u(v)
reshape_continuity_flag	u(1)
for(i=0; i < reshape_num_ranges_minus1+2; i++) {	
reshape_poly_coeff_order0_int[i]	ue(v)
reshape_poly_coeff_order0_frac[i]	u(v)
}	
if(reshape_continuity_flag== 1) {	
reshape_poly_coeff_order1_int	se(v)
reshape_poly_coeff_order1_frac	u(v)
}	
}	

reshape_input_luma_bit_depth_minus8 задает битовую глубину отсчета входной компоненты яркости процесса перестраивания.

coeff_log2_offset_minus2 задает количество дробных битов для связанных с перестраиванием расчетов коэффициентов применительно к компоненте яркости. Значение coeff_log2_offset_minus2 будет находиться в диапазоне от 0 до 3 включительно.

reshape_num_ranges_minus1 плюс 1 задает количество диапазонов в кусочной функции перестраивания. Когда отсутствует, значение reshape_num_ranges_minus1 подразумевается 0. reshape_num_ranges_minus1 будет находиться в диапазоне от 0 до 7 включительно применительно к компоненте яркости.

reshape_equal_ranges_flag, равный 1, задает, что кусочная функция перестраивания подразделена на NumberRanges фрагментов с приблизительно равной длиной, и длина каждого диапазона не сигнализируется в явном виде. reshape_equal_ranges_flag, равный 0, задает, что длина каждого диапазона сигнализируется в явном виде.

reshape_global_offset_val используется для вывода значения сдвига, которое используется, чтобы задавать начальную точку 0-ого диапазона.

reshape_range_val[i] используется для вывода длины i-ого диапазона компоненты яркости.

reshape_continuity_flag задает свойства непрерывности функции перестраивания для компоненты яркости. Если reshape_continuity_flag равен 0, непрерывность нулевого порядка применяется к кусочно-линейным функциям обратного перестраивания между следующими друг за другом точками поворота. Если reshape_continuity_flag равен 1, гладкость первого порядка используется для получения полиномиальных функций обратного перестраивания полного второго порядка между следующими друг за другом точками поворота.

reshape_poly_coeff_order0_int [i] задает целочисленное значение коэффициента полинома нулевого порядка i-ого фрагмента для компоненты яркости.

reshape_poly_coeff_order0_frac [i] задает дробное значение коэффициента полинома нулевого порядка i-ого фрагмента для компоненты яркости.

reshape_poly_coeff_order1_int задает целочисленное значение коэффициента полинома первого порядка для компоненты яркости.

reshape_poly_coeff_order1_frac задает дробное значение коэффициента полинома первого порядка для компоненты яркости.

[0081] Таблица 8 описывает примерный вариант осуществления альтернативного параметрического представления согласно модели B, описанной ранее (справочный материал [3]).

Таблица 8: Примерный синтаксис для параметрического представления функции перестраивания (модели B)

reshaping_sliceheader_table_model_B() {	Descriptor
reshape_model_profile_type	ue(v)
reshape_model_scale_idx	u(2)
reshape_model_min_bin_idx	u(5)
reshape_model_max_bin_idx	u(5)
reshape_model_num_band	u(4)
for (i=0; i < reshape_model_num_band; i++) {	
reshape_model_band_profile_delta [i]	u(1)
}	
}	

[0082] В таблице 8, в варианте осуществления, синтаксические параметры могут быть

определены в виде:

reshape_model_profile_type задает тип профиля, подлежащий использованию в процессе построения перестройщика.

reshape_model_scale_idx задает значение индекса масштабного коэффициента (обозначенного как ScaleFactor), подлежащего использованию в процессе построения перестройщика. Значение ScaleFactor предусматривает улучшенное управление функцией перестраивания для улучшенной общей эффективности кодирования. Дополнительные подробности об использовании этого ScaleFactor приведены в отношении обсуждения касательно процесса реконструкции функции перестраивания (например, как изображено на фиг. 5A и фиг. 5B). В качестве примера и без ограничения, значение reshape_model_scale_idx будет находиться в диапазоне от 0 до 3 включительно. В варианте осуществления, отношение соответствия между scale_idx и ScaleFactor, как показано в таблице, приведенной ниже, заданы посредством:

ScaleFactor=1.0-0.05* reshape_model_scale_idx.

reshape_model_scale_idx	ScaleFactor
0	1,0
1	0,95
2	0,9
3	0,85

В еще одном примере, ради более эффективной реализации с фиксированной точкой, ScaleFactor=1-1/16* reshape_model_scale_idx.

reshape_model_scale_idx	ScaleFactor
0	1,0
1	0,9375
2	0,875
3	0,8125

reshape_model_min_bin_idx задает минимальный индекс элемента разрешения, подлежащий использованию в процессе построения перестройщика. Значение reshape_model_min_bin_idx будет находиться в диапазоне от 0 до 31 включительно.

reshape_model_max_bin_idx задает максимальный индекс элемента разрешения, подлежащий использованию в процессе построения перестройщика. Значение reshape_model_max_bin_idx будет находиться в диапазоне от 0 до 31 включительно.

reshape_model_num_band задает количество полос, подлежащих использованию в процессе построения перестройщика. Значение reshape_model_num_band будет находиться в диапазоне от 0 до 15 включительно.

reshape_model_band_profile_delta [i] задает значение приращения, подлежащее использованию для коррекции профиля i-ой полосы в процессе построения перестройщика. Значение reshape_model_band_profile_delta [i] будет находиться в диапазоне от 0 до 1 включительно.

[0083] По сравнению со справочным материалом [3], синтаксис в таблице 8 гораздо более эффективен посредством определения набора «установленных по умолчанию типов профиля», скажем, наиболее ярких участков изображения, средних тонов и темных участков. В варианте осуществления, каждый тип имеет предварительно заданный профиль зрительной важности полосы. Предварительно заданные полосы и соответствующие профили могут быть реализованы в виде постоянных значений в декодере, или они также могут сигнализироваться с использованием высокоуровневого синтаксиса (такого как набор параметров последовательности). В кодировщике, каждое

изображение сначала анализируется и категоризируется одним из типов профиля. Тип профиля сигнализируется синтаксическим элементом **«reshape_model_profile_type»**. При адаптивном перестраивании, для того чтобы захватывать полный диапазон динамики изображений, профилирование по умолчанию дополнительно корректируется приращением для каждого или подмножества диапазонов яркости. Значения приращения выводятся на основе зрительной важности полос яркости и сигнализируются синтаксическими элементами **«reshape_model_band_profile_delta»**

[0084] В одном из вариантов осуществления, значение приращения может принимать значения только 0 или 1. В кодировщике, зрительная важность определяется посредством сравнения процента пикселей полосы во всем изображении с процентом пикселей полосы в пределах «преобладающих полос», где преобладающие полосы могут выявляться с использованием локальной гистограммы. Если пиксели в пределах полосы сосредотачиваются в небольшом локальном блоке, полоса наиболее вероятно является зрительно важной в блоке. Итоговые суммы для преобладающих полос суммируются и нормируются для формирования значимого сравнения, чтобы получить значения приращения для каждой полосы.

[0085] В декодере, процесс реконструкции функции перестройщика должен вызываться для получения LUT перестраивания на основе способов, описанных в справочном материале [3]. Поэтому, сложность является более высокой по сравнению с более простой моделью кусочной аппроксимации, которой необходимо оценивать только сплайны для вычисления LUT. Преимущество использования синтаксиса параметрической модели состоит в том, что оно значительно снижает битовую скорость использования перестройщика. Например, на основе типичного испытательного контента, модели, изображенной в таблице 7, необходимо 200-300 битов для сигнализации перестройщика, тогда как параметрическая модель (как в таблице 8) использует всего лишь около 40 бит.

[0086] В еще одном варианте осуществления, как изображено в таблице 9, справочная таблица прямого перестраивания может выводиться согласно параметрической модели для значений dQP. Например, в варианте осуществления,

dQP=clip3(*min*, *max*, *scale***X*+*offset*),

при этом *min* и *max* обозначают границы dQP, *scale* и *offset* - два параметра модели, а *X* обозначает параметр, выведенный на основе яркости сигнала (например, значения яркости пикселя или, применительно к блокам, показателя яркости блока, например, ее минимума, максимума, среднего значения, изменчивости, среднеквадратического отклонения, и тому подобного). Например, без ограничения,

dQP=clip3(-3, 6, 0.015**X* - 7.5).

Таблица 9: Примерный синтаксис для параметрического представления функции перестраивания (модели C)

sps_reshaper_model_C() {	descriptor
full_range_input_flag	u(1)
dQP_model_scale_int_prec	ue(v)
if (dQP_model_scale_int_prec > 0) {	
dQP_model_scale_int	u(v)
}	
dQP_model_scale_frac_prec_minus16	ue(v)
dQP_model_scale_frac	u(v)
if (dQPModelScaleAbs) {	
dQP_model_scale_sign	u(1)
}	

dQP_model_offset_int_prec_minus3	ue(v)
dQP_model_offset_int	u(v)
dQP_model_offset_frac_prec_minus1	ue(v)
dQP_model_offset_frac	u(v)
if (dQPModelOffsetAbs) {	
dQP_model_offset_sign	u(1)
}	
dQP_model_abs_prec_minus3	ue(v)
dQP_model_max_abs	u(v)
if (dQP_model_max_abs) {	
dQP_model_max_sign	u(1)
}	
dQP_model_min_abs	u(v)
if (dQP_model_min_abs) {	
dQP_model_min_sign	u(1)
}	
}	

[0087] В варианте осуществления, параметры в таблице 9 могут быть определены, как изложено ниже:

full_range_input_flag задает диапазон входного видеосигнала. full_range_input_flag со значением 0 соответствует входному видеосигналу со стандартным динамическим диапазоном. full_range_input_flag со значением 1 соответствует входному видеосигналу с полным диапазоном. Когда отсутствует, full_range_input_flag подразумевается имеющим значение 0.

Примечание: В качестве используемого в материалах настоящей заявки, термин «видео с полным диапазоном» обозначает, что действительные кодовые комбинации в видеосигнале «не ограничены». Например, применительно к 10-битному видеосигналу с полным диапазоном, действительные кодовые комбинации находятся между 0 и 1023, где 0 отображается в наименьший уровень яркости. В противоположность, применительно к 10-битному «видеосигналу со стандартным диапазоном», действительные кодовые комбинации находятся между 64 и 940, и 64 отображается в наименьший уровень яркости.

Например, расчет «полного диапазона» и «стандартного диапазона» может вычисляться, как изложено ниже:

для нормирования значений Ey' яркости в пределах [0 1], чтобы кодировать в битах BD (например, BD=10, 12, и тому подобному):

полный диапазон: Y=clip3(0, (1<<BD) -1, Ey'* ((1<<BD) -1)))

стандартный диапазон: Y=clip3(0, (1<<BD) -1, round(1<<(BD-8)*(219*Ey'+16)))

Этот синтаксис аналогичен синтаксису **«video_full_range_flag»** в параметрах VUI HEVC, как описано в разделе E.2.1 технических условий HEVC (H.265) (справочный материал [11]).

dQP_model_scale_int_prec задает количество битов, используемых для представления dQP_model_scale_int. dQP_model_scale_int_prec, равный 0, указывает, что dQP_model_scale_int не сигнализируется и подразумевается имеющим значение 0.

dQP_model_scale_int задает целочисленное значение шкалы модели dQP.

dQP_model_scale_frac_prec_minus16 плюс 16 задает количество битов, используемых для представления dQP_model_scale_frac.

dQP_model_scale_frac задает дробное значение шкалы модели dQP.

Переменная dQPModelScaleAbs выводится в виде:

dQPModelScaleAbs=dQP_model_scale_int << (dQP_model_scale_frac_prec_minus16+16)+

dQP_model_scale_frac

dQP_model_scale_sign задает знак шкалы модели dQP. Когда dQPModelScaleAbs равно 0, dQP_model_scale_sign не сигнализируется, и он подразумевается имеющим значение 0.

dQP_model_offset_int_prec_minus3 плюс 3 задает количество битов, используемых для представления dQP_model_offset_int.

dQP_model_offset_int задает целочисленное значение сдвига модели dQP.

dQP_model_ offset _frac_prec_minus1 плюс 1 задает количество битов, используемых для представления dQP_model_offset_frac.

dQP_model_ offset_frac задает дробное значение сдвига модели dQP.

Переменная dQPModelOffsetAbs выводится в виде:

dQPModelOffsetAbs=dQP_model_offset_int << (dQP_model_offset_frac_prec_minus1+1)+ dQP_model_offset_frac

dQP_model_offset_sign задает знак сдвига модели dQP. Когда dQPModelOffsetAbs равен 0, dQP_model_offset_sign не сигнализируется и подразумевается имеющим значение 0.

dQP_model_abs_prec_minus3 плюс 3 задает количество битов, используемых для представления dQP_model_max_abs и dQP_model_min_abs.

dQP_model_max_abs задает целочисленное значение максимума модели dQP.

dQP_model_max_sign задает знак максимума модели dQP model max. Когда dQP_model_max_abs равен 0, dQP_model_max_sign не сигнализируется и подразумевается имеющим значение 0.

dQP_model_min_abs задает целочисленное значение минимума модели dQP.

dQP_model_min_sign задает знак минимума модели dQP. Когда dQP_model_min_abs равен 0, dQP_model_min_sign не сигнализируется и подразумевается имеющим значение 0.

Процесс декодирования для модели C

[0088] При данных синтаксических элементах таблицы 9, LUT перестраивания может быть выведена, как изложено ниже.

Переменная dQPModelScaleFP выводится в виде:

dQPModelScaleFP=((1- 2*dQP_model_scale_sign) * dQPModelScaleAbs) << (dQP_model_offset_frac_prec_minus1+1).

Переменная dQPModelOffsetFP выводится в виде:

dQPModelOffsetFP=((1-2* dQP_model_offset_sign) * dQPModelOffsetAbs) << (dQP_model_scale_frac_prec_minus16+16).

Переменная dQPModelShift выводится в виде:

dQPModelShift=(dQP_model_offset_frac_prec_minus1+1)+ (dQP_model_scale_frac_prec_minus16+16).

Переменная dQPModelMaxFP выводится в виде:

dQPModelMaxFP=((1- 2*dQP_model_max_sign) * dQP_model_max_abs) << dQPModelShift.

Переменная dQPModelMinFP выводится в виде:

dQPModelMinFP=((1- 2*dQP_model_min_sign) * dQP_model_min_abs) << dQPModelShift.

for Y =0: maxY // Например, применительно к 10-битным видеоданным, maxY=1023

{

dQP[Y]=clip3(dQPModelMinFP, dQPModelMaxFP, dQPModelScaleFP*Y+ dQPModelOffsetFP);

slope[Y]=exp2((dQP[Y]+3)/6); // реализация exp2 с фиксированной точкой, где exp2(x) =2^(x);

}

If (full_range_input_flag == 0) // если входным сигналом является видеосигнал со стандартным диапазоном

Применительно к Y вне стандартного диапазона (то есть, Y=[0:63] и [940:1023]), устанавливаем slope[Y]=0;

CDF[0]=slope[0];

for Y =0: maxY-1

{

CDF[Y+1]=CDF[Y]+slope[Y]; // CDF[Y] - интеграл углового коэффициента [Y]

}

for Y=0: maxY

{

FwdLUT[Y]=round(CDF[Y]*maxY/CDF[maxY]); // округление и нормирование для получения FwdLUT

}

[0089] В еще одном варианте осуществления, как изображено в таблице 10, функция прямого перестраивания может быть представлена в виде совокупности точек (In_Y) поворота яркости и их соответствующих кодовых комбинаций (Out_Y). Для упрощения кодирования, диапазон яркости входного сигнала описан в показателях начального поворота и последовательности равноразнесенных последующих поворотов с использованием кусочно-линейного представления. Пример представления функции прямого перестраивания для 10-битных входных данных изображен на фиг. 7.

Таблица 10: Примерный синтаксис для основанном на повороте представлении функции перестраивания (модели D)

sps_reshaper_model_D() {	descriptor
full_range_input_flag	u(1)
bin_pivot_start	u(v)
bin_cw_start	u(v)
log2_num_equal_bins_minus3	ue(v)
equal_bin_pivot_delta	u(v)
bin_cw_in_first_equal_bin	u(v)
bin_cw_delta_abs_prec_minus4	ue(v)
for(i=0 ; i < NumEqualBins - 1 ; i++) {	
bin_cw_delta_abs[i]	u(v)
if (bin_cw_delta_abs[i]) {	
bin_cw_delta_sign[i]	u(1)
}	
}	
}	

[0090] В варианте осуществления, параметры в таблице 10 могут быть определены, как изложено ниже:

full_range_input_flag задает диапазон входного видеосигнала. full_range_input_flag со значением 0 соответствует входному видеосигналу со стандартным диапазоном. full_range_input_flag со значением 1 соответствует входному видеосигналу с полным диапазоном. Когда отсутствует, full_range_input_flag подразумевается имеющим значение 0.

bin_pivot_start задает значение поворота первого элемента (710) разрешения равной длины. Когда full_range_input_flag равен 0, bin_pivot_start будет большим, чем или равным наименьшему входному сигналу со стандартным диапазоном и будет меньшим, чем

наибольший входной сигнал со стандартным диапазоном. (Например, применительно к 10-битному входному сигналу SDR, bin_pivot_start (710) будет находиться между 64 и 940).

bin_cw_start задает приведенное в соответствие значение (715) bin_pivot_start (710) (например, bin_cw_start=FwdLUT[bin_pivot_start]).

log2_num_equal_bins_minus3 плюс 3 задает количество элементов разрешения равной длины, следующих за начальным поворотом (710). Переменные NumEqualBins и NumTotalBins определены согласно:

NumEqualBins=1<<(log2_num_equal_bins_minus3+3)

if full_range_input_flag == 0

NumTotalBins=NumEqualBins+4

else

NumTotalBins=NumEqualBins+2

Примечание: Экспериментальные результаты показывают, что большинство функций прямого перестраивания могут быть представлены с использованием восьми отрезков равной длины; однако, сложные функции перестраивания могут требовать большего количества отрезков (например, 16 или более).

equal_bin_pivot_delta задает длину элементов разрешения равной длины (например, 720-1, 720-N). NumEqualBins *, равный bin_pivot_delta будет меньшим, чем или равным действительному входному диапазону. (Например, если full_range_input_flag имеет значение 0, действительный входной диапазон должен иметь значение 940-64=876 для 10-битных входных сигналов; если full_range_input_flag имеет значение 1, действительный входной диапазон должен находиться от 0 до 1023 для 10-битных входных сигналов).

bin_cw_in_first_equal_bin задает количество приведенных в соответствие кодовых комбинаций (725) в первом элементе разрешения равной длины (720-1).

bin_cw_delta_abs_prec_minus4 плюс 4 задает количество битов, используемых для представления bin_cw_delta_abs[i] применительно к каждому последующему равному элементу разрешения.

bin_cw_delta_abs[i] задает значение bin_cw_delta_abs[i] для каждого последующего элемента разрешения равной длины. bin_cw_delta[i] (например, 735) является разностью кодовых комбинаций (например, 740) в текущем элементе i разрешения равной длины (например, 720-N) по сравнению с кодовыми комбинациями (например, 730) в предыдущем элементе i-1 разрешения равной длины.

bin_cw_delta_sign[i] задает знак bin_cw_delta_abs[i]. Когда bin_cw_delta_abs[i] равен 0, bin_cw_delta_sign[i] не сигнализируется и подразумевается имеющим значение 0.

Переменная bin_cw_delta[i]=(1- 2*bin_cw_delta_sign[i])* bin_cw_delta_abs[i]

Процесс декодирования для модели D

[0091] При данных синтаксических элементах таблицы 10, LUT перестраивания может быть выведена, как изложено ниже, для 10-битного входного сигнала:

Определим ограничения:

minIN=minOUT=0;

maxIN=maxOUT=2^BD - 1=1023 for 10-bit //BD=битовая глубина

minStdIN=64 для 10 бит

minStdIN=940 для 10 бит

Этап 1: вывести значение поворота In_Y [j] для j=0: NumTotalBins

In_Y [0]=0;

In_Y [NumTotalBins]=maxIN;

if (full_range_input_flag == 0)

```
{
In_Y [ 1 ]=minStdIN;
In_Y [ 2 ]=bin_pivot_start;
for ( j=3: NumTotalBins - 2 )
In_Y [ j ]=In_Y [ j - 1 ]+equal_bin_pivot_delta;
In_Y [ NumTotalBins - 1 ]=maxStdIN;
}
else
{
In_Y [ 1 ]=bin_pivot_start;
for j=2: NumTotalBins - 1
In_Y [ j ]=In_Y [ j - 1 ]+equal_bin_pivot_delta;
}
```

Этап 2: вывести отображаемое значение Out_Y [j] для j=0: NumTotalBins

```
Out_Y [ 0 ]=0;
Out_Y [ NumTotalBins ]=maxOUT;
if ( full_range_input_flag == 0 )
{
Out_Y [ 1 ]=0;
Out_Y [ 2 ]=bin_cw_start;
Out_Y [ 3 ]=bin_cw_start+bin_cw_in_first_equal_bin;
bin_cw [ 3 ]=bin_cw_in_first_equal_bin;
for j=( 4: NumTotalBins - 2 )
bin_cw [ j ]=bin_cw [ j - 1 ]+bin_cw_delta [ j - 4 ]; // bin_cw_delta[ i ] start from idx 0
for j=( 4: NumTotalBins - 2 )
Out_Y [ j ]=Out_Y [ j - 1 ]+bin_cw [ j ];
Out_Y [ NumTotalBins - 1 ]=maxOUT;
}
else
{
Out_Y [ 1 ]=bin_cw_start;
Out_Y [ 2 ]=bin_cw_start+bin_cw_in_first_equal_bin;
bin_cw [ 2 ]=bin_cw_in_first_equal_bin;
for j=( 3: NumTotalBins - 1 )
bin_cw [ j ]=bin_cw [ j - 1 ]+bin_cw_delta [ j - 3 ]; // bin_cw_delta[ i ] start from idx 0
for j=3: NumTotalBins - 1
Out_Y [ j ]=Out_Y [ j - 1 ]+bin_cw [ j ];
}
```

Этап 3: Линейная интерполяция для получения всей записи LUT

```
Init: FwdLUT[ ]
for ( j=0: NumTotalBins - 1 )
{
InS=In_Y [ j ];
InE=In_Y [ j+1 ];
OutS=Out_Y [ j ];
OutE=Out_Y [ j+1 ];
for i=In_Y[ j ]: In_Y[ j+1] - 1
{
```

FwdLUT [i]=OutS+round ((OutE - OutS) * (i - InS)/(InE - InS));
}
}
FwdLUT [In_Y [NumTotalBins]]=Out_Y [NumTotalBins];

[0092] Вообще, перестраивание может включаться или отключаться для каждой секции. Например, можно активировать перестраивание только для секций с внутрикадровым предсказанием и деактивировать для секций с межкадровым предсказанием. В еще одном примере, можно деактивировать перестраивание для секций с межкадровым предсказанием, которые имеют наивысший временный уровень. (Примечание: в качестве примера, в качестве используемого в материалах настоящей заявки, временные подуровни могут соответствовать определению временных подуровней в HEVC.) При определении модели перестройщика, в одном из примеров, можно сигнализировать о модели перестройщика только в SPS, но, в другом примере, можно сигнализировать о модели перестройщика секции в секциях с внутрикадровым предсказанием. В качестве альтернативы, можно сигнализировать о модели перестройщика в SPS и предоставлять модели перестройщика секции возможность обновлять модель перестройщика SPS применительно ко всем секциям, или можно предоставлять модели перестройщика секции возможность обновлять модель перестройщика SPS только применительно к секциям с внутрикадровым предсказанием. Что касается секций с межкадровым предсказанием, которые поддерживают секцию с внутрикадровым предсказанием, можно применять модель перестройщика SPS или модель перестройщика секции с внутрикадровым предсказанием.

[0093] В качестве еще одного примера, фиг. 5A и 5B изображают процесс реконструкции функции перестраивания в декодере согласно варианту осуществления. Процесс использует способы, описанные в материалах настоящей заявки и в справочном материале [3], с диапазоном видимости в пределах [0 5].

[0094] Как изображено на фиг. 5A, прежде всего (этап 510), декодер извлекает переменную reshape_model_profile_type и устанавливает (этапы 515, 520 и 525), для каждого элемента разрешения, надлежащий начальный профиль полосы. Например, в псевдокоде:

if (reshape_model_profile_type == 0) $R[b_i]=R_{bright}[b_i]$;

elseif (reshape_model_profile_type == 1) $R[b_i]=R_{dark}[b_i]$;

else $R[b_i]=R_{mid}[b_i]$.

[0095] На этапе 530, декодер корректирует каждый профиль полосы с использованием принятых значений reshape_model_band_profile_delta[b_i], как в

for (i=0: reshape_model_num_band-1)
{$R[b_i]=R[b_i]$+reshape_model_band_profile_delta [b_i]}.

[0096] На этапе 535, декодер распространяет скорректированные значения на каждый профиль элемента разрешения, как в

если bin[j] принадлежит полосе b_i, R_bin[j]=R[b_i].

[0097] На этапе 540, профили элементов разрешения модифицируются, как в

if (j > reshape_model_max_bin_idx) or (j < reshape_model_min_bin_idx)
{R_bin[j]=0}.

[0098] Параллельно, на этапах 545 и 550, декодер может извлекать параметры для вычисления значения масштабного коэффициента и потенциально подходящих кодовых комбинаций для каждого bin[j], как в

ScaleFactor=1.0-0.05* reshape_model_scale_idx

CW_dft[j]=кодовые комбинации в элементе разрешения, когда используется перестраивание по умолчанию

CW_PQ[j]=TotalCW/TotalNumBins.

[0099] При вычислении значения ScaleFactor для реализации с фиксированной точкой, вместо использования пересчетного коэффициента 0,05, взамен можно использовать 1/16=0,0625.

[0100] Продолжая по фиг. 5B, на этапе 560, декодер начинает предварительное назначение кодовых комбинаций (CW) для каждого элемента разрешения на основе профиля элемента разрешения, как в

If R_bin[j] == 0, CW[j]=0

If R_bin[j] == 1, CW[j]=CW_dft[j]/2;

If R_bin[j] == 2, CW[j]=min(CW_PQ[j], CW_dft[j]);

If R_bin[j] == 3, CW[j]=(CW_PQ[j]+CW_dft[j])/2;

If R_bin[j] >=4, CW[j]=max(CW_PQ[j], CW_dft[j]);

[0101] На этапе 565, он вычисляет все используемые кодовые комбинации и уточняет/завершает назначения кодовых комбинаций (CW), как в

CW_{used}=Sum(CW[j]):

if CW_{used} > TotalCW, rebalance CW[j]=CW[j]/(CW_{used}/TotalCW);

else

{

CW_remain=TotalCW - CW_{used} ;

CW_remain назначается элементам разрешения с наибольшим R_bin[j]);

}

[0102] В заключение, на этапе 565, декодер a) формирует функцию прямого перестраивания (например, FwdLUT), накапливая значения CW[j], b) перемножает значение ScaleFactor со значениями FwdLUT, чтобы сформировать окончательное FwdLUT (FFwdLUT), и c) он формирует функцию InvLUT обратного перестраивания на основе FFwdLUT.

[0103] В реализации с фиксированной точкой, вычисление ScaleFactor и FFwdLUT может быть выражено в виде:

ScaleFactor=(1<< SF_PREC) - reshape_model_scale_idx

FFwdLUT=(FwdLUT * ScaleFactor+(1 << (FP_PREC+SF_PREC - 1))) >> (FP_PREC+SF_PREC),

где SF_PREC и FP_PREC - предварительно заданные связанные с точностью переменные (например, SF_PREC=4, а FP_PREC=14), *«c=a << n»* операцию двоичного сдвига влево *a* на *n* бит (или $c=a*(2^n)$), а *«c=a >> n»* обозначает операцию двоичного сдвига вправо *a* на *n* бит (или $c=a/(2^n)$).

Производные QP цветности

[0104] Эксплуатационные качества кодирования цветности тесно связаны с эксплуатационными качествами кодирования яркости. Например, в AVC и HEVC, определена таблица для задания зависимости между параметрами квантования (QP) для компонент яркости и цветности или между яркостью и цветностью. Технические условия также предоставляют возможность использовать один или более сдвигов QP цветности для дополнительной гибкости при определении зависимости QP между яркостью и цветностью. Когда используется перестраивание, значение яркости модифицируется, отсюда, зависимость между яркостью и цветностью также могла бы быть модифицирована. Для сохранения и дополнительного улучшения эффективности

кодирования при перестраивании, в варианте осуществления, на уровне элемента кодирования (CU), сдвиг QP цветности выводится на основе кривой перестраивания. Необходимо, чтобы эта операция выполнялась как в декодере, так и в кодировщике.

[0105] В качестве используемого в материалах настоящей заявки, термин «элемент кодирования» (CU) обозначает кодированный блок (например, макроблок, и тому подобное). Например, без ограничения, в HEVC, CU определен как «блок кодирования отсчетов яркости, два соответствующих блока кодирования отсчетов цветности кадра, который имеет три массива отсчетов, или блок кодирования отсчетов монохромного кадра или кадра, который кодирован с использованием трех отдельных цветовых плоскостей и синтаксических структур, используемых для кодирования отсчетов».

[0106] В варианте осуществления, значение параметра квантования (QP) цветности (chromaQP) может быть получено, как изложено ниже:

1) На основе кривой перестраивания, получаем эквивалентное отображение dQP яркости, dQPLUT:

для CW=0: MAX_CW_VALUE-1

dQPLUT [CW]= -6*log2(slope[CW]);

где slope[CW] обозначает угловой коэффициент кривой прямого перестраивания в каждой точке CW (кодовой комбинации), а MAX_CW_VALUE - максимальное значение кодовой комбинации для данной битовой глубины, например, для 10-битного сигнала, MAX_CW_VALUE=1024 (2^{10}).

Затем, для каждого элемента кодирования (CU):

2) вычисляем среднюю яркость элемента кодирования, обозначенную как AvgY:

3) вычисляем значение chromaDQP на основе dQPLUT[], AvgY, архитектуры перестраивания, функции Inv() обратного перестраивания и типа секции, как показано в таблице 11, приведенной ниже:

Таблица 11: Примерные значения согласно архитектуре перестраивания

Архитектура перестраивания	Секция с внутрикадровым предсказанием	Секция с межкадровым предсказанием
Внеконтурный	dQPLUT[Inv(AvgY)]	dQPLUT[Inv(AvgY)]
Внутриконтурный перестройщик только с внутрикадровым предсказанием	dQPLUT[Inv(AvgY)]	0
Внутриконтурный перестройщик для остаточных значений	dQPLUT[AvgY]	dQPLUT[AvgY]
Гибридное внутриконтурное перестраивание	dQPLUT[Inv(AvgY)]	dQPLUT[AvgY]

4) вычисляем chromaQP в виде:

chromaQP=QP_luma+chromaQPOffset+chromaDQP;

где chromaQPOffset обозначает сдвиг QP цветности, а QP_luma обозначает QP яркости для элемента кодирования. Отметим, что значение сдвига QP цветности может быть разным для каждой цветовой компоненты (скажем, Cb и Cr) и значения сдвига QP цветности передаются в декодер в виде части кодированного битового потока.

[0107] В варианте осуществления, dQPLUT[] может быть реализован в качестве предварительно заданной LUT. Допустим, все кодовые комбинации разделяются на N элементов разрешения (например, N=32) и каждый элемент разрешения содержит M= MAX_CW_VALUE/N кодовых комбинаций (например, M=1024/32=32). Когда назначаются новые кодовые комбинации на каждый элемент разрешения, они могут ограничивать количество кодовых комбинаций, чтобы находилось от 1 до 2*M, так что они могут предварительно вычислять dQPLUT[1 …2*M] и сохранять его в виде LUT. Этот подход может избегать каких бы то ни было вычислений с плавающей точкой

или аппроксимации вычислений с фиксированной точкой. Он также может экономить время кодирования/декодирования. Применительно к каждому элементу разрешения, один неизменный chromaQPOffset используется для всех кодовых комбинаций в этом элементе разрешения. Значение DQP устанавливается равным dQPLUT[L], где L - количество кодовых комбинаций для этого элемента разрешения, где $1 \leq L \leq 2*M$.

Значения dQPLUT могут предварительно вычисляться, как изложено ниже:

```
for i=1:2*M
slope[i]=i/M;
dQPLUT[i]=-6*log2(slope[i]);
end
```

Разные схемы квантования могут использоваться для получения целочисленного значения QP при вычислении dQPLUT[x], такие как: round(), ceil(), floor() или их комбинация. Например, можно устанавливать пороговое значение TH, и, если Y<TH, использовать floor() для квантования значения dQP, иначе, когда $Y \geq TH$, использовать ceil() для квантования значения dQP. Использование таких схем квантования и соответствующих параметров может быть предварительно задано в кодеке или может сигнализироваться в битовом потоке для адаптации. Примерный синтаксис, который предоставляет возможность комбинирования схем квантования с одним пороговым значением, как обсуждено ранее, показан, как изложено ниже:

quant_scheme_signal_table() {	Descriptor
if (sps_reshaper_chromaAdj > 0) {	
quant_scheme_idc // 0: round(), 1: ceil(), 2: floor(), 3: mix	u(2)
if (quant_scheme_idc == 3) { //mix	
quant_change_threshold	u(v)
first_quant_scheme_idc	u(2)
second_quant_scheme_idc	u(2)
}	
}	
}	

Функция quant_scheme_signal_table() может быть определена на разных уровнях синтаксиса перестраивания (например, уровне последовательности, уровне секции, и тому подобном) в зависимости от необходимости адаптации.

[0108] В еще одном варианте осуществления, значения chromaDQP могут вычисляться посредством применения масштабного коэффициента к остаточному сигналу в каждом элементе кодирования (или элементе преобразования, чтобы быть более точным). Этот масштабный коэффициент может быть зависящим от яркости значением и может вычисляться: a) численно, например, в виде производной первого порядка (углового коэффициента) LUT прямого перестраивания (например, смотрите уравнение (6) в следующем разделе), или b) в виде:

$$Slope(x)=2^{(\frac{dQP(x)}{6})}.$$

При вычислении $Slope(x)$ с использованием $dQP(x)$, dQP может поддерживаться на точности с плавающей точкой без целочисленного квантования. В качестве альтернативы, можно вычислять квантованные целочисленные значения dQP с использованием многообразия разных схем квантования. В некоторых вариантах осуществления, такое масштабирование может выполняться на уровне пикселей вместо уровня блока, где каждое остаточное значение цветности может масштабироваться разным масштабным коэффициентом, выведенным с использованием так называемого

яркостного значения предсказания такого отсчета цветности. Таким образом,

Таблица 12: Примерные значения dQP цветности, использующие масштабирование для архитектуры гибридного внутриконтурного перестраивания

Обновление	Секция с внутрикадровым предсказанием	Секция с межкадровым предсказанием
Основанное на CU масштабирование цветности (один и тот же перестройщик S_cu совместно используется всеми отсчетами в CU)	Scu=SlopeLUT[Inv(AvgY)] C_Res_scaled=C_Res * S_cu	Scu=SlopeLUT[AvgY] C_Res_scaled=C_Res * S_cu
Основанное на пикселях масштабирование цветности (разный преобразователь масштаба S_px в каждом отсчете)	Spx=SlopeLUT[Inv(ColPredY)] C_Res_scaled =C_Res* S_px	Spx=SlopeLUT[ColPredY] C_Res_scaled=C_Res * S_px

Например, если, if CSCALE_FP_PREC=16

Упреждающее масштабирование: после того, как сформировано остаточное значение, перед преобразованием и квантованием:

C_Res=C_orig - C_pred

C_Res_scaled=C_Res * S+(1 << (CSCALE_FP_PREC - 1))) >> CSCALE_FP_PREC

Обратное масштабирование: после обратного квантования цветности и обратного преобразования, но до восстановления:

C_Res_inv=(C_Res_scaled << CSCALE_FP_PREC)/S

C_Reco=C_Pred+C_Res_inv;

где S не является ни S_cu. ни S_px.

Примечание: В таблице 12, при вычислении Scu, средняя яркость блока (AvgY) рассчитывается перед применением обратного перестраивания. В качестве альтернативы, можно применять обратное перестраивание перед вычислением средней яркости, например, Scu=SlopeLUT[Avg(Inv[Y])]. Этот альтернативный порядок вычислений также применяется к вычислительным значениям в таблице 11; то есть, вычисление Inv(AvgY) могло бы быть заменено вычислением значений Avg(Inv[Y]). Последний подход может считаться более точным, но имеет повышенную вычислительную сложность.

Оптимизации кодировщика в отношении перестраивания

[0109] Этот раздел обсуждает некоторое количество технологий для улучшения эффективности кодирования в кодировщике посредством совместной оптимизации и перестраивания, и параметры кодировщика при перестраивании являются частью нормативного процесса декодирования (как описано в одной из трех потенциально подходящих архитектур). Вообще, оптимизация и перестраивание кодировщика пытаются решать задачи кодирования в разных местах со своими собственными ограничениями. В традиционной системе формирования изображений и кодирования, есть два типа квантования: a) квантование отсчетов (например, кодирование степени контрастности или PQ) в сигнале основной полосы и b) связанное с преобразование квантование (часть сжатия). Перестраивание расположено между ними Основанное на кадре перестраивание вообще обновляется на основе кадра и предоставляет возможность отображения значений отсчетов только на основе их уровня яркости, не учитывая никакой пространственной информации. В основанном на блоках кодеке (таком как HEVC), квантование преобразования (например, для яркости) применяется в пределах пространственного блока и может пространственно настраиваться, поэтому, способы оптимизации кодировщика должны применять один и тот же набор параметров для всего блока, содержащего в себе отсчеты с разными значениями яркости. Как понимается изобретателями и описано в материалах настоящей заявки, совместные перестраивание и оптимизация кодировщика могут дополнительно улучшать эффективность кодирования.

Выбор режима внутрикадрового/межкадрового предсказания

[0110] В традиционном кодировании, решения с межкадровым/внутрикадровым предсказанием основаны на вычислении функции искажения (*dfunc*()) между исходными отсчетами и предсказанными отсчетами. Примеры таких функций включают в себя сумму квадратичных ошибок (SSE) и сумму абсолютных разностей (SAD), а также другие. В варианте осуществления, такие показатели искажения могут применяться с использованием перестроенных значений пикселей. Например, если исходная *dfunct*() использует *Orig_sample*(*i*) и *Pred_sample*(*i*), когда применяется перестраивание, *dfunct*() может пользоваться своими соответствующими перестроенными значениями, *Fwd*(*Orig_sample*(*i*)) и *Fwd*(*Pred_sample*(*i*)). Этот подход предусматривает более точный выбор режима межкадрового/внутрикадрового предсказания, таким образом, улучшая эффективность кодирования.

LumaDQP с перестраиванием

[0111] В документе по обычным условиям испытания (CTC) HDR JCTVC (справочный материал [6]), lumaDQP и chromaQPoffsets являются двумя настройками кодировщика, используемыми для модификации параметров квантования (QP) применительно к компонентам яркости и цветности, чтобы улучшать эффективность кодирования HDR. В этом изобретении, предложено несколько новых алгоритмов кодировщика, чтобы дополнительно улучшить первоначальное предложение. Что касается каждого элемента адаптации lumaDQP (например, CTU 64×64), значение *dQP* вычисляется на основе среднего входного значения яркости элемента (как в таблице 3 из справочного материала [6]). Окончательный параметр QP квантования, используемый для каждого элемента кодирования в пределах этого элемента адаптации lumaDQP, должен корректироваться вычитанием этого *dQP*. Таблица отображения *dQP* является конфигурируемой во входной конфигурации кодировщика. Эта входная конфигурация обозначена как dQP_{inp}.

[0112] Как обсуждено в справочном материале [6] и [7], в существующих схемах кодирования, одна и та же dQP_{inp}LUT lumaDQP и используется для обоих кадров, с внутрикадровым предсказанием и межкадровым предсказанием. Кадр с внутрикадровым предсказанием и кадр с межкадровым предсказанием может иметь разные свойства и качественные характеристики. В данном изобретении предлагается адаптировать настройки lumaDQP на основе типа кодирования кадра. Поэтому, две таблицы отображения dQP являются конфигурируемыми во входной конфигурации кодировщика и обозначены как $dQP_{inpIntra}$и $dQP_{inpInter}$.

[0113] Как обсуждено раньше, при использовании способа внутриконтурного перестраивания с внутрикадровым предсказанием, так как перестраивание не выполняется над кадрами с межкадровым предсказанием, важно, чтобы некоторая настройка lumaDQP применялась к кадрам с межкадровым предсказанием для достижения аналогичного качества, как если бы кадры с межкадровым предсказанием перестраивались тем же самым перестройщиком, используемым для кадра с внутрикадровым предсказанием. В одном из вариантов осуществления, настройка lumaDQP для кадров с межкадровым предсказанием должна соответствовать характеристикам кривой перестраивания, используемой в кадрах с внутрикадровым предсказанием. Пусть

$$Slope(x)=Fwd'(x)=(Fwd(x+dx)- Fwd(x-dx))/(2dx), \quad (6)$$

обозначает первую производную функции прямого перестраивания, тогда, в варианте осуществления, обозначает автоматически выводимые значения $dQP_{auto}(x)$, которые могут вычисляться, как изложено ниже:

Если $Slope(x)=0$, то $dQP_{auto}(x)=0$, иначе

$dQP_{auto}(x)=6\log2(Slope(x))$, (7)

где $dQP_{auto}(x)$ может быть усечен приемлемым диапазоном, например, [-6 6].

[0114] Если lumaDQP активирован для кадров с внутрикадровым предсказанием с перестраиванием (то есть, установлен внешний $dQP_{inpIntra}$), lumaDQP для кадров с межкадровым предсказанием должен это учитывать. В варианте осуществления, окончательный dQP_{final} с межкадровым предсказанием может вычисляться посредством сложения dQP_{auto}, выведенного из перестройщика (уравнения (7)), и настройки $dQP_{inpIntra}$ для кадров с межкадровым предсказанием. В еще одном варианте осуществления, для использования в своих интересах классификации качества внутрикадрового предсказания, dQP_{final} для кадров с межкадровым предсказанием может быть установлен в dQP_{auto} или просто с небольшим приращением (посредством настройки $dQP_{inpInter}$) и прибавлен dQP_{auto}.

[0115] В варианте осуществления, когда перестраивание активировано, могут применяться нижеследующие общие правила для настройки значений *dQP* яркости:

1) Таблицы отображения *dQP* яркости могут устанавливаться независимо для кадров с внутрикадровым предсказанием и межкадровым предсказанием (на основе типа кодирования кадра);

2) Если кадра внутри контура кодирования находится в перестроенной области (например, кадры с внутрикадровым кодированием в архитектуре внутриконтурного перестраивания с внутрикадровым предсказанием или все кадры в архитектуре внеконтурного перестраивания), необходимо, чтобы отображение входной яркости в приращение QP, dQP_{inp}, также было преобразовано в dQP_{rsp} перестроенной области. То есть,

$dQP_{rsp}(x)=dQP_{inp}[Inv(x)]$. (8)

3) Если кадр внутри контура кодирования находится в неперестроенной области (например, подвергнутой обратному перестраиванию или неперестроенной, например, кадры с межкадровым предсказанием в архитектуре внутриконтурного перестраивания с внутрикадровым предсказанием или все кадры в архитектуре внутриконтурного перестраивания остаточных значений), отображение входной яркости в приращение QP не должно преобразовываться и может использоваться непосредственно.

4) Автоматический вывод deltaQP с межкадровым предсказанием действителен только применительно к архитектуре внутриконтурного перестраивания с внутрикадровым предсказанием. Фактическое приращение QP, используемое для кадров с межкадровым предсказанием, в таком случае является суммой автоматически выведенного и входного значений:

$dQP_{final}[x]=dQP_{inp}[x]+dQP_{auto}[x]$, (9)

и $dQP_{final}[x]$ может быть усечен приемлемым диапазоном, например [-12 12];

5) Таблица отображения яркости в *dQP* может обновляться в каждом кадре, или когда есть изменение LUT перестраивания. Фактическая адаптация *dQP* (из среднего значения яркости блока получаем соответствующий *dQP* для квантования этого блока) может происходить на уровне CU (конфигурируемом кодировщиком).

[0116] Таблица 13 обобщает настройки *dQP* для каждой одной из трех предложенных архитектур.

Таблица 13: настройка dQP

Архитектура	Внеконтурное	Внутриконтурное только с внутрикадровым предсказанием	Внутриконтурное для остаточных значений
Кадр с внутрикадровым предсказанием dQP	$dQP_{final}(x)=dQP_{rsp}(x)=dQP_{inpIntra}[Inv(x)]$	$dQP_{final}(x)=dQP_{rsp}(x)=dQP_{inpIntra}[Inv(x)]$	$dQP_{final}(x)=dQP_{inpIntra}(x)$
Кадр с межкадровым предсказанием dQP	$dQP_{final}(x)=dQP_{rsp}(x)=dQP_{inpInter}[Inv(x)]$	$dQP_{final}(x)=dQP_{auto}(x)+dQP_{inpInter}(x)$	$dQP_{final}(x)=dQP_{inpInter}(x)$

Rate Distortion Optimization (RDO)

[0117] В программном обеспечении JEM6.0 (справочный материал [8]), основанное на пикселях взвешенное искажение RDO (оптимизации скорости к искажениям) используется, когда активирован lumaDQP. Таблица весов фиксируется на основе значений яркости. В варианте осуществления, таблица весов будет адаптивно настраиваться на основе настройки lumaDQP, вычисляемой, как предложено в предыдущем разделе. Два веса, для суммы квадратичных ошибок (SSE) и суммы абсолютных разностей (SAD), предложены, как изложено ниже:

$$weight\_SSE[x]=2^{\left(\frac{dQP[x]}{3}\right)}, \quad (10a)$$

$$weight\_SAD[x]=2^{\left(\frac{dQP[x]}{6}\right)}. \quad (10b)$$

[0118] Весом, вычисленным посредством уравнения (10a) или уравнения (10b), является общий вес, основанный на окончательном dQP, который содержит как входной lumaDQP, так и dQP, выведенный из функции прямого перестраивания. Например, на основе уравнения (9), уравнение (10a) может быть записано в виде

$$weight_{SSE}[x] = 2^{\left(\frac{dQP_{final}[x]}{3}\right)} = 2^{\left(\frac{dQP_{inp}[x]+\ dQP_{auto}[x]}{3}\right)} = 2^{\left(\frac{dQP_{inp}[x]}{3}\right)} * 2^{\left(\frac{dQP_{auto}[x]}{3}\right)}.$$

Общий вес может быть разделен на вес, вычисленный по входному lumaDQP:

$$weight_{SSE}[x]\_inp = 2^{\left(\frac{dQP_{inp}[x]}{3}\right)},$$

и вес от перестраивания:

$$weight_{SSE}[x]_{reshape} = 2^{\left(\frac{dQP_{auto}[x]}{3}\right)} = 2^{\left(\frac{6\log2(Slope(x))}{3}\right)} = (Slope(x))^2.$$

Когда общий вес вычисляется с использованием полного *dQP* посредством вычисления первым веса от перестраивания, он теряет точность из-за операции усечения для получения целочисленного dQP_{auto}. Взамен, непосредственное использование функции углового коэффициента для расчета веса из перестраивания, может сохранять более высокую точность веса, а потому, является более благоприятным.

[0119] Обозначим в качестве W_{dQP} вес, выведенный из входного lumaDQP. Пусть $f'(x)$ обозначает первую производную (или угловой коэффициент) кривой прямого перестраивания. В варианте осуществления, общий вес учитывает как значения dQP, так и форму кривой перестраивания, таким образом, значение общего веса может быть выражено в виде:

$$weight_{total} = Clip3(0.0, 30.0, W_{dQP} * f'(x)^2). \quad (11)$$

[0120] Аналогичный подход также может быть применен к компонентам цветности. Например, в варианте осуществления, применительно к цветности, *dQP*[*x*] может быть определен согласно таблице 13.

Взаимодействие с другими средствами кодирования

[0121] Когда активировано перестраивание, этот раздел приводит несколько примеров предложенных изменений, необходимых в других средствах кодирования. Взаимодействия могли бы существовать для любых возможных существующих или будущих средств кодирования, подлежащих включению в стандарт кодирования видеосигнала следующего поколения. Примеры, приведенные ниже, не являются ограничивающими. Вообще, необходимо идентифицировать область видеосигнала (перестроенную, неперестроенную, подвергнутую обратному перестраиванию) во время этапов кодирования, и необходимо учитывать операции, имеющие дело с видеосигналом на каждом этапе.

Межкомпонентное предсказание по линейной модели

[0122] В CCLM (межкомпонентном предсказании по линейной модели) (справочный материал [8]) предсказанные отсчеты $pred_C(i, j)$ цветности могут выводиться с использованием сигнала $rec_L'(i, j)$ реконструкции яркости:

$$pred_C(i, j) = \alpha \cdot rec_L'(i, j) + \beta. \quad (12)$$

[0123] Когда перестраивание активировано, в варианте осуществления, может быть необходимо различать, находится ли восстановленный сигнал яркости в перестроенной области (например, внеконтурном перестройщике или внутриконтурном перестройщике с внутрикадровым предсказанием) или в неперестроенной области (например. внутриконтурном перестройщике остаточных значений). В одном из вариантов осуществления, можно в явном виде использовать сигнал яркости реконструкции как есть без каких бы то ни было дополнительных сигнализации или операций. В других вариантах осуществления, если восстановленный сигнал находится в неперестроенной области, можно преобразовывать сигнал яркости реконструкции, чтобы также находился в неперестроенной области, как в:

$$pred_C(i, j) = \alpha \cdot Inv(rec_L'(i, j)) + \beta. \quad (13)$$

[0124] В других вариантах осуществления, можно добавлять синтаксические элементы битового потока для сигнализации, какая область требуется (перестроенная или неперестроенная), что может решаться процессом RDO, или можно получать решение на основе декодированной информации, таким образом, экономя накладные расходы, необходимые при явной сигнализации. Можно выполнять соответствующие операции над восстановленным сигналом на основе решения.

Перестройщик со средством предсказания остаточных значений

[0125] В профиль расширения диапазона HEVC включено средство предсказания остаточных значений. Остаточный сигнал цветности предсказывается по остаточному сигналу яркости на стороне кодировщика в виде:

$$\Delta r_C(x, y) = r_C(x, y) - \left(\alpha \times r_L'(x, y)\right) >> 3, \quad (14)$$

и он компенсируется на стороне декодера в виде:

$$r_C'(x, y) = \Delta r_C'(x, y) + \left(\alpha \times r_L'(x, y)\right) >> 3, \quad (15)$$

где r_C обозначает остаточный отсчет цветности в положении (x, y), r_L' обозначает восстановленный остаточный отсчет компоненты яркости, Δr_C обозначает предсказанный сигнал, использующий межцветное предсказание, $\Delta r_C'$ обозначает восстановленный сигнал после кодирования и декодирования Δr_C, а r_C' обозначает восстановленный остаточный сигнал цветности.

[0126] Когда перестраивание активировано, может быть необходимо учитывать, какой остаточное значение яркости использовать для предсказания остаточных значений цветности. В одном из вариантов осуществления, можно использовать «остаточное значение» как есть (которое может быть перестроенным или неперестроенным на основе архитектуры перестройщика). В еще одном варианте осуществления, можно вынуждать остаточное значение яркости находиться в одной области (такой как неперестроенная область) и выполнять надлежащие отображения. В еще одном варианте осуществления, надлежащая обработка может выводиться декодером, может сигнализироваться в явном виде, как описано ранее.

Перестройщик с адаптивным усечением

[0127] Адаптивное усечение (справочный материал [8]) является новым средством, введенным для сигнализации исходного диапазона данных в отношении динамических характеристик контента и для выполнения адаптивного усечения вместо неизменного усечения (на основе внутренней информации о битовой глубине) на каждом этапе в рабочем процессе (например, при преобразовании/квантовании, контурной фильтрации, выводе), где происходит усечение. Пусть

$$T_{clip} = Clip_{BD}(T, bitdepth, C) = Clip3\left(min_C, max_C, T\right), \quad (16)$$

где $x=Clip3(min, max, c)$ обозначает:

$$x = \begin{cases} min, if\ c \le min \\ max, if\ c \ge max, \\ c, otherwise \end{cases}$$

и

C является идентификатором компоненты (типично Y, Cb или Cr)

min_C - нижняя граница усечения, используемая в текущей секции, для идентификатора компоненты. C

max_C - верхняя граница усечения, используемая в текущей секции для идентификатора компоненты. C

[0128] Когда перестраивание активировано, в варианте осуществления, может быть необходимо понимать, область, в которой на данный момент находится поток данных, и правильно выполнять усечение. Например, если имеем дело с усечением данных в перестроенной области, необходимо, чтобы исходные границы усечения были преобразованы в перестроенную область:

$$T_{clip} = Clip_{BD}(T, bitdepth, C) =$$

$$= Clip3\left(Fwd(min_C), Fwd(max_C), T\right). \quad (17)$$

Вообще, необходимо обрабатывать каждый этап усечения надлежащим образом в отношении архитектуры перестраивания.

Перестройщик и контурная фильтрация

[0129] В программном обеспечении HEVC и JEM 6.0, необходимо, чтобы контурные фильтры, такие как ALF и SAO оценивали оптимальные параметры фильтра с использованием восстановленных отсчетов яркости и несжатых «исходных» отсчетов яркости. Когда перестраивание активировано, в варианте осуществления, можно задавать (явно или неявно) область, где желательно выполнять оптимизацию фильтра. В одном из вариантов осуществления, можно оценивать параметры фильтра в перестроенной области (когда реконструкция происходит в перестроенной области по отношению к перестроенному оригиналу). В других вариантах осуществления, можно оценивать параметры фильтра в неперестроенной области (когда реконструкция

происходит в неперестроенной области или подвергнутой обратному перестраиванию по отношению к оригиналу).

Например, в зависимости от архитектуры внутриконтурного перестраивания, варианты выбора и операции оптимизации внутриконтурного фильтра (ILFOPT) могут быть описаны таблицами 14 и 15.

Таблица 14. Оптимизация внутриконтурной фильтрации в архитектуре внутриконтурного перестраивания только с внутрикадровым предсказанием и при внутриконтурном гибридном перестраивании

	Внутри кадра	Внутри кадра
Вариант 1 выбора: оценивать и выполнять контурную фильтрацию (LF) в перестроенной области;	**Сторона кодировщика:** Использует уже перестроенное исходное изображение с внутрикадровым предсказанием в качестве опорного сигнала LF Использует восстановленное изображение с внутрикадровым кодированием (в перестроенной области) и опорный сигнал LF для оценки параметров LF; два случая в зависимости от положения модуля LF по отношению к модулю обратного перестраивания (блока 265 и 270 на фиг. 2C): если обратное перестраивание выполняется раньше LF, необходимо применять прямое перестраивание к восстановленному изображению с внутрикадровым предсказанием если обратное перестраивание должно выполняться после LF, использует непосредственно восстановленное изображение с внутрикадровым предсказанием Применяет LF к восстановленному изображению с внутрикадровым предсказанием в перестроенной области Осуществляет обратное перестраивание всего восстановленного изображения с внутрикадровым предсказанием Сохраняет в DPB **Сторона декодера:** Применяет LF к восстановленному изображению с внутрикадровым предсказанием в перестроенной области; два случая в зависимости от положения модуля LF по отношению к модулю обратного перестраивания (блока 265 и 270 на фиг. 2D): если обратное перестраивание выполняется раньше LF, необходимо осуществлять прямое перестраивание восстановленного изображения с внутрикадровым предсказанием до применения LF если обратное перестраивание должно выполняться после LF, применяет LF непосредственно к восстановленному изображению с внутрикадровым предсказанием Осуществляет обратное перестраивание всего восстановленного изображения с внутрикадровым предсказанием Сохраняет в DPB	**Сторона кодировщика:** Осуществляет прямое перестраивание исходного изображения с внутрикадровым предсказанием в качестве опорного сигнала LF Осуществляет прямое перестраивание восстановленного изображения с межкадровым предсказанием Оценивает параметры LF Применяет LF к восстановленному изображению с межкадровым предсказанием в перестроенной области Осуществляет обратное перестраивание всего восстановленного изображения с межкадровым предсказанием Сохраняет в DPB **Сторона декодера:** Осуществляет прямое перестраивание восстановленного изображения с межкадровым предсказанием Применяет LF к восстановленному изображению с межкадровым предсказанием в перестроенной области Осуществляет обратное перестраивание всего восстановленного изображения с межкадровым предсказанием Сохраняет в DPB
Вариант 2 выбора: оценивает или выполняет LF в неперестроенной области	**Сторона кодировщика:** Осуществляет обратное перестраивание исходного изображения в качестве опорного сигнала LF, если перестраивание на месте было выполнено над буфером исходного изображения; или выбирает неперестроенное исходное изображение в качестве опорного сигнала LF Осуществляет обратное перестраивание восстановленного изображения с внутрикадровым предсказанием Оценивает параметры LF Осуществляет обратное перестраивание всего восстановленного изображения Применяет LF к восстановленному изображению с внутрикадровым предсказанием Сохраняет в DPB **Сторона декодера:** Осуществляет обратное перестраивание всего восстановленного изображения Применяет LF к восстановленному изображе-	**(В точности традиционный рабочий процесс LF)** **Сторона кодировщика:** Использует восстановленное изображение с межкадровым предсказанием и исходное изображение в качестве опорного значения LF для оценки параметров LF Применяет LF к восстановленному изображению с межкадровым предсказанием Сохраняет в DPB **Сторона декодера:** Применяет LF к восстановленному изображению с межкадровым предсказанием Сохраняет в DPB

	нию с внутрикадровым предсказанием Сохраняет в DPB	

Таблица 15. Оптимизация внутриконтурной фильтрации в архитектуре внутриконтурного перестраивания для остаточных значений

Внутриконтурное перестраивание для остаточных значений	С внутрикадровым предсказанием и межкадровым предсказанием (такое же как операции с межкадровым предсказанием в таблице 14)

[0130] Несмотря на то, что большинство подробных обсуждений в материалах настоящей заявки обращаются к способам, выполняемым над компонентой яркости, специалист в данной области техники будет принимать во внимание, что аналогичные способы могут выполняться в пределах цветовых компонентов цветности и связанных с цветностью параметрах, таких как chromaQPOffset (например, смотрите справочный материал [9]).

Внутриконтурное перестраивание и интересующие области (ROI)

[0131] При данном изображении, в качестве используемого в материалах настоящей заявки, термин 'интересующая область' (ROI) обозначает область изображения, которая считается особенно интересной. В этом разделе, представлены новейшие варианты осуществления, которые поддерживают внутриконтурное перестраивание только для интересующей области. То есть, в варианте осуществления, перестраивание может применяться только внутри ROI и не применяться вне. В еще одном варианте осуществления, можно применять разные кривые перестраивания в интересующей области и вне интересующей области.

[0132] Использование ROI мотивировано необходимостью сохранять равновесие битовой скорости и качества изображения. Например, рассмотрим видеопоследовательность захода солнца. В верхней половине изображений, можно получать солнце на небе относительно ровного цвета (так, пиксели на небесном фоне могут иметь очень низкую изменчивость). В противоположность, нижняя половина изображения может изображать движущиеся волны. С ракурса наблюдателя, верхняя часть может считаться гораздо более важной, чем нижняя. С другой стороны, движущиеся волны, вследствие высокой изменчивости их пикселей, труднее сжимать, требуется большее количество битов на каждый пиксель; однако, можно пожелать выделить большее количество битов на часть с солнцем, чем на часть с волнами. В этом случае, верхняя половина могла бы быть обозначена как интересующая область.

Описание ROI

[0133] В наши дни, большинство кодеков (например, AVC, HEVC, и тому подобные) основаны на блоках. Для упрощения реализации, можно задавать область в элементах или блоках. Без ограничения, с использованием HEVC в качестве примера, область может быть определена в виде множества элементов кодирования (CU) или узлов кодового дерева (CTU). Можно задавать один ROI или многочисленные ROI. Многочисленные ROI могут быть раздельными или перекрываться. ROI не обязательно должны быть прямоугольными. Синтаксис для ROI может быть предусмотрен на любом интересующем уровне, таком как уровень секции, уровень кадра, уровень видеопотока, и тому подобного. В варианте осуществления, ROI задана первой в наборе параметров последовательности (SPS). В таком случае, в заголовке секции можно предоставить возможность небольших изменений ROI. Таблица 16 изображает пример синтаксиса, где одна ROI задана в качестве многочисленных CTU в прямоугольной области. Таблица 17 описывает синтаксис модифицированного ROI на уровне секции.

Таблица 16: Синтаксис SPS для ROI

SPS() {	Descriptor
...	
sps_reshaper_enable_flag	u(1)
if (sps_reshaper_enable_flag) {	
…..	
sps_reshaper_active_ROI_flag	u(1)
if (sps_reshaper_active_ROI_flag) {	
reshaper_active_ROI_in_CTUsize_left	ue(v)
reshaper_active_ROI_in_CTUsize_right	ue(v)
reshaper_active_ROI_in_CTUsize_top	ue(v)
reshaper_active_ROI_in_CTUsize_bottom	ue(v)
}	
}	
….	
}	

Таблица 17: Синтаксис секции для ROI

reshaping_sliceheader_table() {	Descriptor
...	
if (sps_reshaper_active_ROI_flag) {	
reshape_model_ROI_modification_flag	u(1)
if (reshape_model_ROI_modification_flag) {	
reshaper_ROI_mod_offset_left	se(v)
reshaper_ROI_mod_offset_right	se(v)
reshaper_ROI_mod_offset_top	se(v)
reshaper_ROI_mod_offset_bottom	se(v)
}	
}	
….	
}	

sps_reshaper_active_ROI_flag, равный 1, задает, что ROI существует в кодированной видеопоследовательности (CVS). sps_reshaper_active_ROI_flag, равный 0, задает, что ROI не существует в CVS.

Каждый из **reshaper_active_ROI_in_CTUsize_left, reshaper_active_ROI_in_CTUsize_right, reshaper_active_ROI_in_CTUsize_top** и **reshaper_active_ROI_in_CTUsize_bottom** задает отсчет кадров в ROI в показателях прямоугольной области, заданной в координатах кадра. Координаты равны offset*CTUsize для левого верхнего угла и offset*CTUsize-1 для правого нижнего угла.

reshape_model_ROI_modification_flag, равный 1, задает, что ROI модифицируется в текущей секции. reshape_model_ROI_modification_flag, равный 0, задает, что ROI не модифицируется в текущей секции.

Каждый из **reshaper_ROI_mod_offset_left, reshaper_ROI_mod_offset_right, reshaper_ROI_mod_offset_top** и **reshaper_ROI_mod_offset_bottom** задает значение сдвига влево/вправо/вверх/вниз от reshaper_active_ROI_in_CTUsize_left, reshaper_active_ROI_in_CTUsize_right, reshaper_active_ROI_in_CTUsize_top и reshaper_active_ROI_in_CTUsize_bottom.

Что касается многочисленных ROI, примерный синтаксис по таблицам 16 и 17 для одиночной ROI мог бы быть расширен с использованием индекса (или идентификатора) для каждой ROI, аналогично схеме, используемой в HEVC для определения, с использованием сообщений SEI, многочисленных прямоугольников панорамирования и сканирования (смотрите технические условия HEVC, справочный материал [11], раздел D.2.4).

Обработка ROI при внутриконтурном перестраивании только с внутрикадровым предсказанием

Что касается перестраивания только с внутрикадровым предсказанием, сначала перестраивается часть ROI кадра, затем применяется кодирование. Так как перестраивание применяется только к ROI, могла бы быть видна граница между частями кадра с ROI и без ROI. Поскольку контурный фильтр (например, 270 на фиг. 2C или фиг. 2D) может проходить по границам, особое внимание должно быть уделено для ROI применительно к оптимизации контурного фильтра (ILFOPT). В варианте осуществления, предполагается, что контурный фильтр применяется только там, где весь декодированный кадр находится в одной и той же области. То есть, весь кадр полностью находится в перестроенной области, либо полностью находится в неперестроенной области. В одном из вариантов осуществления, на стороне декодера, если контурная фильтрация применяется в неперестроенной области, сначала будет применяться обратное перестраивание к секции ROI декодированного кадра, а затем применяться контурный фильтр. Затем, декодированный кадр сохраняется в DPB. В еще одном варианте осуществления, если контурный фильтр применяется к перестроенной области, сначала будет применяться перестраивание к части без ROI декодированного кадра, затем применяться контурный фильтра, а затем обратное перестраивание всего кадра. Затем, декодированный кадр сохраняется в DPB. В еще одном другом варианте осуществления, если контурная фильтрация применяется в перестроенной области, сначала можно осуществлять обратное перестраивание части ROI декодированного кадра, затем перестроить весь кадр, затем применять контурный фильтр, затем осуществлять обратное перестраивание всего кадра. Затем, декодированный кадр сохраняется в DPB. Три подхода обобщены в таблице 18. С точки зрения вычислений, способ «A» является более простым. В варианте осуществления, задействование ROI может использоваться для задания порядка выполнения обратного перестраивания по отношению к контурной фильтрации (LF). Например, если ROI активно используется (например, флаг синтаксиса SPS=истина), то LF (блок 270 на фиг. 2C и фиг. 2D) выполняется после обратного перестраивания (блок 265 на фиг. 2C и фиг. 2D). Если ROI активно не используется, то LF выполняется до обратного перестраивания.

Таблица 18. Варианты контурной фильтрации (LF) с использованием ROI

Способ A	Способ B	Способ C
Осуществить обратное перестраивание ROI декодированного кадра Применить контурную фильтрацию (LF) ко всему кадру Сохраняет в DPB	Применить перестраивание к не связанной с ROI част декодированного кадра. Применить LF ко всему кадру Осуществить обратное перестраивание всего кадра Сохраняет в DPB	Осуществить обратное перестраивание ROI декодированного кадра Перестроить весь кадр Применить LF ко всему кадру Осуществить обратное перестраивание всего кадра Сохраняет в DPB

Обработка ROI при внутриконтурном перестраивании остаточных значений предсказания

[0134] Применительно к архитектуре внутриконтурного перестраивания остаточных значений (предсказания) (например, смотрите 200C_D на фиг. 2F), в декодере, с использованием уравнения (3), обработка может быть выражена в виде:

Если (currentCTU принадлежит ROI),

$Reco\_sample = Inv(Res\_d + Fwd(Pred\_sample))$, (смотрите уравнение (3))

иначе

$Reco\_sample = Res\_d + Pred\_sample$

конец

ROI и соображения касательно кодировщика

[0135] В кодировщике, необходимо, чтобы каждый CTU проверялся, принадлежит ли он ROI или нет. Например, применительно к внутриконтурному перестраиванию остаточных значений предсказания, на основе уравнения (3), может выполняться простая проверка:

Если (currentCTU принадлежит ROI),

применять взвешенное искажение в RDO применительно к яркости. Вес выводится на основе уравнения (10).

иначе

применять невзвешенное искажение в RDO применительно к яркости.

конец

[0136] Примерный рабочий процесс кодирования, который принимает во внимание ROI во время перестраивания, может содержать следующие этапы:

Для кадра с внутрикадровым предсказанием:

Применить прямое перестраивание к зоне ROI исходного кадра

Кодировать кадр с внутрикадровым предсказанием

Применить обратное перестраивание к зоне ROI восстановленного кадра до контурного фильтра (LF).

Выполнить контурную фильтрацию в неперестроенной области, как изложено ниже (например, смотрите способ «C» в таблице 18), что включает в себя следующие этапы:

Применить прямое перестраивание к зоне без ROI исходного кадра (для того чтобы сделать весь исходный кадр перестроенным для опорного сигнала контурного фильтра)

Применить прямое перестраивание ко всей площади изображения восстановленного кадра

Вывести параметры контурного фильтра и применить контурную фильтрацию

Применить обратное перестраивание ко всей площади изображения восстановленного кадра и сохранить в DPB

На стороне кодировщика, поскольку необходимо, чтобы LF имел несжатое опорное изображение для оценки параметров фильтра, обработка опорного сигнала LF для каждого способа является такой, как в таблице 19:

Таблица 19. Обработка опорного изображения LF касательно ROI

Способ A	Способ B	Способ C
Использовать неперестроенный исходный входной кадр для опорного значения LF;	Применять перестраивание по всему (как на части без ROI, так и на части ROI) исходному входному кадру для опорного значения LF;	Применять перестраивание по всему (как на части без ROI, так и на части ROI) исходному входному кадру для опорного значения LF;

Что касается кадра с межкадровым предсказанием:

При кодировании кадра с межкадровым предсказанием, что касается каждого CU внутри ROI, применить перестраивание остаточных значений предсказания и взвешенное искажение по яркости; для каждого CU вне ROI, не применять никакого перестраивания

Оптимизация контурной фильтрации (вариант 1 выбора) выполняется как раньше (как если бы ROI не использовалась):

Осуществить прямое перестраивание всей площади изображения исходного кадра

Осуществить прямое перестраивание всей площади изображения восстановленного кадра

Вывести параметры контурного фильтра и применить контурную фильтрацию

Применить обратное перестраивание ко всей площади изображения восстановленного кадра и сохранить в DPB

Перестраивание кодированного с помощью HLG контента

[0137] Термин HybridLog-Gamma или HLG обозначает еще одну передаточную

функцию, определенную в протоколе BT. 2100 для отображения сигналов с расширенным динамическим диапазоном. HLG был разработан, чтобы поддерживать обратную совместимость с традиционными сигналами со стандартным динамическим диапазоном, кодированными с использованием традиционной функции степени контрастности. При сравнении классификации кодовой комбинацией между кодированным с помощью PQ контентом и кодированным с помощью HLG контентом, отображение PQ имеет тенденцию выделять больше кодовых комбинаций в темных и ярких зонах, тогда как большинство контента HLG оказывается распределенным в средний диапазон. Два подхода могут использоваться для перестраивания яркости HLG. В одном из вариантов осуществления, можно просто преобразовывать контент HLG в контент PQ, а затем, применять все связанные с PQ технологии перестраивания, обсужденные ранее. Например, могли бы применяться следующие этапы:

1) Отобразить яркость HLG (например, *Y*) в яркость PQ. Пусть функция или LUT преобразования будет обозначена как HLG2PQLUT(Y)

2) Анализировать значения яркости PQ и вывести основанную на PQ функцию или LUT прямого перестраивания. Обозначим ее как PQAdpFLUT(Y)

3) Объединить две функции или LUT в единую функцию или LUT: HLGAdpFLUT[i]= PQAdpFLUT[HLG2PQLUT[i]].

[0138] Поскольку классификация кодовой комбинацией HLG совершенно не такая как классификация кодовой комбинацией PQ, такой подход может давать субоптимальные результаты перестраивания. В еще одном варианте осуществления, функция перестраивания HLG выводится непосредственно из отсчетов HLG. Можно применять такую же инфраструктуру, как используется для сигналов PQ, но изменить таблицу CW_Bins_Dft, чтобы отражала характеристики сигнала HLG. В варианте осуществления, с использованием профиля средних тонов для сигналов HLG, несколько таблиц CW_Bins_Dft конструируются согласно предпочтениям пользователя. Например, когда предпочтительно сохранять яркие участки изображения, для альфа=1,4,

g_DftHLGCWBin0={ 8, 14, 17, 19, 21, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 36, 37, 38, 39, 39, 40, 41, 41, 42, 43, 43, 44, 44, 30 }.

Когда предпочтительно сохранять средние тоны (или средний диапазон):

g_DftHLGCWBin1= { 12, 16, 16, 20, 24, 28, 32, 32, 32, 32, 36, 36, 40, 44, 48, 52, 56, 52, 48, 44, 40, 36, 36, 32, 32, 32, 26, 26, 20, 16, 16, 12 }.

Когда предпочтительно сохранять цвет кожи:

g_DftHLGCWBin2= {12, 16, 16, 24, 28, 32, 56, 64, 64, 64, 64, 56, 48, 40, 32, 32, 32, 32, 32, 32, 28, 28, 24, 24, 20, 20, 20, 20, 20, 16, 16, 12};

[0139] С точки зрения синтаксиса битового потока, для проведения различия между основанным на PQ и HLG перестраиванием, добавлен новый параметр, обозначенный как **sps_reshaper_signal_type**, где значение **sps_reshaper_signal_type** указывает тип сигнала, который перестраивался (например, 0 для основанных на степени контрастности сигналов SDR, 1 для кодированных с помощью PQ сигналов, а 2 для кодированных с помощью HLG сигналов).

[0140] Примеры таблиц синтаксиса для перестраивания HDR в SPS и заголовке секции как для PQ, так и для HLG, со всеми признаками, обсужденными ранее (например, ROI, оптимизацией внутриконтурного фильтра (ILFOPT), и ChromaDQPAdjustment), показаны в таблицах 20 и 21.

Таблица 20: Пример синтаксиса SPS для перестраивания

SPS()	Descriptor
……	

sps_reshaper_enable_flag /*1: перестраивание включено, иначе отключено */	u(1)
if (sps_reshaper_enable_flag) {	
sps_reshaper_adaptive_flag /* 1: адаптивное перестраивание включено, иначе выключено */	u(1)
sps_reshaper_signal_type /* например: 0: SDR, 1:PQ, 2: HLG */	u(2)
sps_in_loop_filter_opt_flag /* флаг ILFOPT */	u(1)
sps_luma_based_chroma_qp_offset_flag /* флаг chromaDQPAjustment */	u(1)
sps_reshaper_active_ROI_flag	u(1)
if (sps_reshaper_active_ROI_flag) {	
reshaper_active_ROI_in_CTUsize_left	ue(v)
reshaper_active_ROI_in_CTUsize_right	ue(v)
reshaper_active_ROI_in_CTUsize_top	ue(v)
reshaper_active_ROI_in_CTUsize_bottom	ue(v)
}	
}	

sps_in_loop_filter_opt_flag, равный 1, задает, что оптимизация внутриконтурного фильтра должна выполняться в перестроенной области в кодированной видеопоследовательности (CVS). sps_in_loop_filter_opt_flag, равный 0, задает, что оптимизация внутриконтурного фильтра должна выполняться в неперестроенной области в CVS.

sps_luma_based_chroma_qp_offset_flag, равный 1, задает, что основанный на яркости сдвиг QP цветности выводится (например, согласно таблице 11 или 12) и применяется к кодированию цветности каждого CU в кодированной видеопоследовательности (CVS). sps_luma_based_chroma_qp_offset_flag, равный 0, задает, что основанный на яркости сдвиг QP цветности, не активирован в CVS.

Таблица 21: Примерный синтаксис для перестраивания на уровне секции

reshaping_sliceheader_table_model() {	Descriptor
reshape_model_profile_type	ue(v)
reshape_model_scale_idx	u(2)
reshape_model_min_bin_idx	u(5)
reshape_model_max_bin_idx	u(5)
reshape_model_num_band	u(4)
for (i=0; i < reshape_model_num_band; i++) {	
reshape_model_band_profile_delta [i]	u(1)
}	
if (sps_reshaper_active_ROI_flag) {	
reshape_model_ROI_modification_flag	u(1)
if (reshape_model_ROI_modification_flag) {	
reshaper_ROI_mod_offset_left	se(v)
reshaper_ROI_mod_offset_right	se(v)
reshaper_ROI_mod_offset_top	se(v)
reshaper_ROI_mod_offset_bottom	se(v)
}	
}	
}	

Улучшение качества цветности

[0141] Сторонники основанного на HLG кодирования утверждают, что оно дает лучшую обратную совместимость с сигнализацией SDR. Поэтому, теоретически, основанные на HLG сигналы могли бы применять такие же настройки кодирования, как унаследованные сигналы SDR. Но, при просмотре кодированных с помощью HLG сигналов в режиме HDR, по-прежнему могут наблюдаться некоторые цветные артефакты, особенно в бесцветных областях (таких как белого и серого цвета). В

варианте осуществления, такие артефакты могут быть ослаблены посредством коррекции значений chromaQPOffset во время кодирования. Предполагается, что, для контента HLG, применяется менее агрессивная корректировка chromaQP, чем та, что используется при кодировании сигналов PQ. Например, в справочном материале [10], модель для назначения сдвигов QP для Cb и Cr на основе яркости QP и коэффициента, основанного на основных цветах видеоввода и представления, описана в виде:

$$\mathrm{QPoffsetCb} = \mathrm{Clip3}\big(-12, 0, \mathrm{Round}\big(c_{cb} * (k * \mathrm{QP} + l)\big)\big), \quad (18a)$$

$$\mathrm{QPoffsetCr} = \mathrm{Clip3}\big(-12, 0, \mathrm{Round}\big(c_{cr} * (k * \mathrm{QP} + l)\big)\big), \quad (18b)$$

где c_{cb}=1, если основные цвета видеоввода идентичны основным цветам представления, c_{cb}=1,0, если основные цвета видеоввода равны основным цветам P3D65, и основные цвета представления равны основным цветам протокола BT.2020 ITU-R, и c_{cb}=1,14, если основные цвета видеоввода равны основным цветам протокола BT.709 ITU-R, а основные цвета представления равны основным цветам протокола BT.2020 ITU-R. Подобным образом, c_{cb}=1, если основные цвета видеоввода идентичны основным цветам представления, c_{cb}=1,0, если основные цвета видеоввода равны основным цветам P3D65, и основные цвета представления равны основным цветам протокола BT.2020 ITU-R, и c_{cb}=1,78, если основные цвета видеоввода равны основным цветам протокола BT.709 ITU-R, а основные цвета представления равны основным цветам протокола BT.2020 ITU-R. В заключение, $k = -0.46$ и $l = 0.26$.

[0142] В варианте осуществления, предложено использовать ту же самую модель, но с другими параметрами, которые дают менее агрессивное изменение chromaQPOffset. Например, без ограничения, в варианте осуществления, применительно к Cb в уравнении (18a), c_{cb}=1, *k*=-0,2, и *l*=7, а применительно к Cr в уравнении (18b), c_{cr}=1, *k*=-0,2, и *l*=7. Фиг. 6A и фиг. 6B изображают примеры того, как изменяются значения chromaQPOffset согласно параметру (QP) квантования яркости для PQ (протокол 709) и HLG. Связанные с PQ значения изменяются значительнее, чем связанные с HLG значения. Фиг. 6A соответствует Cb (уравнение (18a)), тогда как фиг. 6B соответствует Cr (уравнение (18b)).

Библиографический список

Каждый из справочных материалов, перечисленных в материалах настоящей заявки, включен в состав посредством ссылки во всей своей полноте.

[1] Заявка PCT под номером PCT/US2016/025082, *In-Loop Block-Based Image Reshaping in High Dynamic Range Video Coding* (*Внутриконтурное основанное на блоках перестраивание изображения при кодировании видео с расширенным динамическим диапазоном*), поданная 30 марта 2016 года, также опубликованная как WO 2016/164235, на Г-М Су.

[2] Д. Бэйлон, З. Гу, А. Лютра, К. Майну, П. Инь, Ф. Пу, Т. Лю, Т. Чен, В. Хьюсак, И. Хе, Л. Керовски, И. Йе, Б. И, «Response to Call for Evidence for HDR and WCG Video Coding: Arris, Dolby and InterDigital» («Ответ на требование данных касательно кодирования видеосигнала HDR и WCG: Arris, Dolby и InterDigital»), документ m36264, июль (2015), Варшава, Польша.

[3] Заявка 15/410,563 на выдачу патента США, *Content-Adaptive Reshaping for High Codeword representation Images* (*Адаптирующееся к контенту перестраивание изображений в представлении большой кодовой комбинацией*), поданная 19 января 2017 года Т. Лю и другими.

[4] Заявка PCT под номером PCT/US2016/042229, *Signal Reshaping and Coding for HDR and Wide Color Gamut Signals* (*Перестраивание и кодирование сигнала для сигналов HDR*

и широкой цветовой гаммы), поданная 14 июля 2016 года, также опубликованная как WO 2017/011636, П. Инем и другими.

[5] *«Exploratory Test Model for HDR extension of HEVC»* (*«Экспериментальный опытный образец для расширения HDR extension в HEVC»),* К. Мину и другие, выходной документ MPEG, JCTVC-W0092 (m37732), 2016 год, Сан-Диего, США.

[6] Е. Франкойс, Дж. Соул, Дж Штрем, П. Инь, *«Common Test Conditions for HDR/WCG video coding experiments»* (*«Общие режимы испытания для экспериментов с кодированием видеосигнала HDR/WCG »),* документ Z1020 JCTVC, Женева, январь 2017 года.

[7] А. Сегал, Е. Франкойс и Д. Русановский, «JVET common test conditions and evaluation procedures for HDR/WCG Video» («Общие режимы испытания и процедуры оценки JVET для видеосигнала HDR/WCG»), JVET-E1020, конференция ITU-T, Женева, январь 2017 года.

[8] Программное обеспечение JEM 6.0: https://jvet.hhi.fraunhofer.de/svn/svn_HMJEMSoftware/tags/HM-16.6-JEM-6.0

[9] Предварительная заявка на выдачу патента США под порядковым № 62/406,483, поданная 11 октября 2016 года, *«Adaptive Chroma Quantization in Video Coding for Multiple Color Imaging Formats»* (*«Адптивное квантование цветности при кодировании видеосигнала для многочисленных форматорв формирования цветного изображения»),* Т. Лю и другие, также поданную в виде заявки на выдачу патента США под порядковым номером 15/728,939, опубликованной в качестве публикации US 2018/0103253 заявки на выдачу патента США.

[10] Дж. Самюельсон и другие (Эдз), «Conversion and coding practices for HDR/WCG Y'CbCr 4:2:0 Video with PQ Transfer Characteristics» («Инструкции по преобразованию и кодированию для видеосигнала HDR/WCG Y'CbCr 4:2:0 с характеристиками передачи PQ») JCTVC-Y1017, конференция ITU-T/ISO, Ченду, октябрь 2016 года.

[11] ITU-T H.265, «*High efficiency video coding*» («Высокоэффективное кодирование видеосигнала») ITU, версия 4.0, (12/2016).

Примерная реализация компьютерной системы

[0143] Варианты осуществления настоящего изобретения могут быть реализованы компьютерной системой, системами, сконфигурированными в электронной схеме и компонентах, устройством на интегральных схемах (ИС, IC), такими как микроконтроллер, программируемая пользователем вентильная матрица (FPGA) или другое конфигурируемое или программируемое логическое устройство (PLD), процессор дискретного времени или сигнальный процессор (DSP), специализированная ИС (ASIC), и/или устройством, которое включает в себя одно или более из таких систем, устройств или компонентов. Компьютер и/или ИС могут выполнять, управлять или приводить в исполнение команды, относящиеся к совместному перестраиванию и кодированию сигналов изображений, таким как описанные в материалах настоящей заявки. Компьютер и/или ИС может вычислять любой из многообразия параметров или значений, которые относятся к процессам перестраивания и кодирования сигналов, описанным в материалах настоящей заявки. Варианты осуществления изображения и видеосигнала могут быть реализованы в аппаратных средствах, программном обеспечении, программно-аппаратных средствах и различных их комбинациях.

[0144] Некоторые реализации изобретения содержат компьютерные процессоры, которые исполняют команды программного обеспечения, которые побуждают процессоры выполнять способ по изобретению. Например, один или более процессоров в устройстве отображения, кодировщике, телевизионной абонентской приставке,

транскодере, или тому подобном, могут реализовывать способы, имеющие отношение к перестраиванию и кодированию сигналов изображений, как описано выше, приводя в исполнение команды программного обеспечения в памяти программ, доступной процессорам. Изобретение также может быть предоставлено в форме программного продукта. Программный продукт может содержать любой долговременный носитель, который несет набор машиночитаемых сигналов, содержащих команды, которые, когда исполняются процессором данных, побуждают процессор данных исполнять способ по изобретению. Программные продукты согласно изобретению могут быть в любом широком многообразии форм. Программный продукт, например, может содержать физические носители, такие как магнитные запоминающие носители данных, в том числе, гибкие диски, накопители на жестком диске, оптические запоминающие носители данных, в том числе, CD-ROM (ПЗУ на компакт-диске), DVD (цифровые многофункциональные диски, электронные запоминающие носители данных, в том числе, ПЗУ (постоянные запоминающие носители, ROM), ОЗУ (оперативное запоминающее устройство, RAM) на флэш-памяти, или тому подобное. Машиночитаемый сигналы в программном продукте по выбору могут быть сжаты или зашифрованы.

[0145] В тех случаях, когда компонент (например, модуль программного обеспечения, процессор, узел, устройство, схема, и т. д.) упоминается выше, если не указано иное, ссылка на такой компонент (в том числе, ссылка на «средство») должна интерпретироваться в качестве включающей в себя, в виде эквивалентов такого компонента, любой компонент, который выполняет функцию описанного компонента (например, который является функционально эквивалентным), в том числе, компоненты, которые конструктивно не эквивалентны раскрытой конструкции, которая выполняет функцию в проиллюстрированных примерных вариантах осуществления изобретения.

Эквиваленты, расширения, альтернативные варианты и прочее

[0146] Таким образом, описаны примерные варианты осуществления, которые относятся к действенному совместному перестраиванию и кодированию сигнала изображений. В вышеизложенном описании изобретения, варианты осуществления настоящего изобретения были описаны со ссылкой на многочисленные конкретные детали, которые могут меняться от реализации к реализации. Таким образом, единственным и исключительным признаком того, что является изобретением и подразумевается изобретением заявителями, является набор пунктов формулы изобретения, которыми кончается данная заявка, в конкретной форме, в которой публикуется такая формула изобретения, в том числе, любые последующие правки. Любые определения, изложенные в материалах настоящей заявки в прямой форме, применительно к терминам, содержащимся в таких пунктах формулы изобретения, будут обуславливать смысл таких терминов, как используемые в формуле изобретения. Отсюда, никакие ограничения, элементы, свойства, признаки, преимущества или свойства, которые не упомянуты в пункте формулы изобретения в прямой форме, ни коим образом не будут ограничивать объем такого пункта формулы изобретения. Описание изобретения и чертежи, соответственно, должны рассматриваться в скорее в иллюстративном, чем ограничительном смысле.

(57) Формула изобретения

1. Способ декодирования, с помощью процессора, кодированного битового потока для формирования выходного изображения, при этом способ содержит этапы, на которых:

принимают кодированный битовый поток, содержащий параметры отображения для отображения кодированных пикселей из перестроенного представления кодовой комбинацией в выходное представление кодовой комбинацией;

осуществляют доступ к кодированному изображению в кодированном битовом потоке посредством пикселей в перестроенном представлении кодовой комбинацией;

извлекают упомянутые параметры отображения, чтобы сформировать функцию прямого перестраивания, причем функция прямого перестраивания отображает пиксели из выходного представления кодовой комбинацией в перестроенное представление кодовой комбинацией, и функцию обратного перестраивания, причем функция обратного перестраивания отображает пиксели из перестроенного представления кодовой комбинацией в выходное представление кодовой комбинацией; и

декодируют кодированное изображение, чтобы сформировать выходное изображение в выходном представлении кодовой комбинацией, при этом декодирование кодированного изображения содержит этапы, на которых:

для области кодированного изображения,

формируют декодированную область остаточных значений в перестроенном представлении кодовой комбинацией,

формируют предсказанную область на основе пикселей в буфере опорных пикселей в выходном представлении кодовой комбинацией,

формируют область восстановленных пикселей в перестроенном представлении кодовой комбинацией на основе декодированной области остаточных значений, предсказанной области и функции прямого перестраивания и

формируют первую выходную область в выходном представлении кодовой комбинацией на основе области восстановленных пикселей и функции обратного перестраивания.

2. Способ по п.1, в котором формирование области восстановленных пикселей содержит этап, на котором вычисляют

$Rec\_sample = Res\_d + \mathrm{Fwd}(Pred\_sample)$,

где *Rec_sample* обозначает пиксель из области восстановленных пикселей, *Res_d* обозначает пиксель из декодированной области остаточных значений, *Pred_sample* обозначает пиксель из предсказанной области, и Fwd() обозначает функцию прямого перестраивания.

3. Способ по п.2, в котором формирование первой выходной области содержит этап, на котором вычисляют

$Reco\_sample = \mathrm{Inv}(Rec\_sample)$,

где *Reco_sample* обозначает пиксель из первой выходной области, и Inv() обозначает функцию обратного перестраивания.

4. Способ по п.1, дополнительно содержащий этап, на котором формируют фильтрованную область выходных пикселей выходного изображения посредством применения внутриконтурной фильтрации к пикселям первой выходной области.

5. Способ по п.1, в котором параметры отображения содержат:

флаг, указывающий, активировано ли перестраивание в кодированном битовом потоке,

один или более флагов, указывающих, активировано ли перестраивание для конкретной кодированной области кодированного изображения,

флаг, указывающий, активирована ли коррекция цветности в кодированном битовом потоке, и

набор синтаксических элементов, представляющих функцию прямого перестраивания.

6. Способ по п.5, в котором упомянутый набор синтаксических элементов, представляющих функцию прямого перестраивания, содержит значение минимального индекса элемента разрешения и значение максимального индекса элемента разрешения.

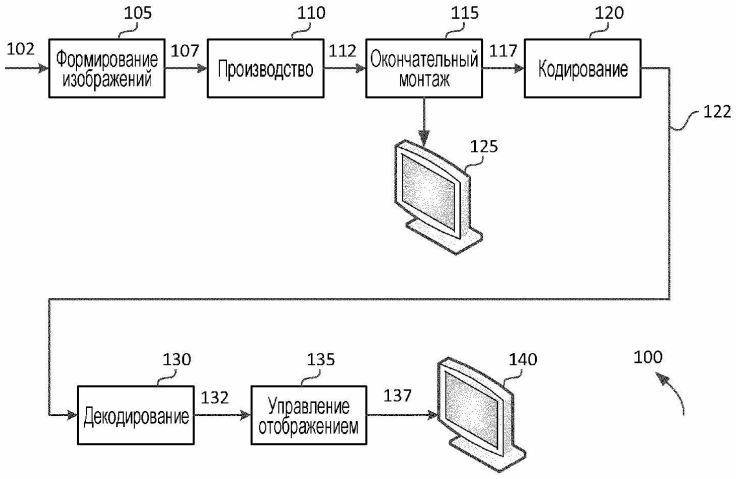
7. Способ по п.1, дополнительно содержащий этап, на котором применяют масштабный коэффициент к значениям цветности пикселей декодированной области остаточных значений.

8. Способ по п.7, в котором масштабный коэффициент основывается на функции прямого перестраивания и параметре, относящемся к яркости.

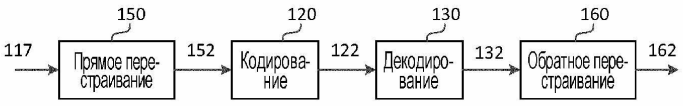
9. Устройство для декодирования перестроенных изображений, содержащее процессор и выполненное с возможностью осуществления любого одного из способов согласно пп.1-8.

10. Долговременный машиночитаемый носитель для декодирования перестроенных изображений, на котором сохранены машиноисполняемые команды для выполнения способа в соответствии с любым из пп.1-8.

1



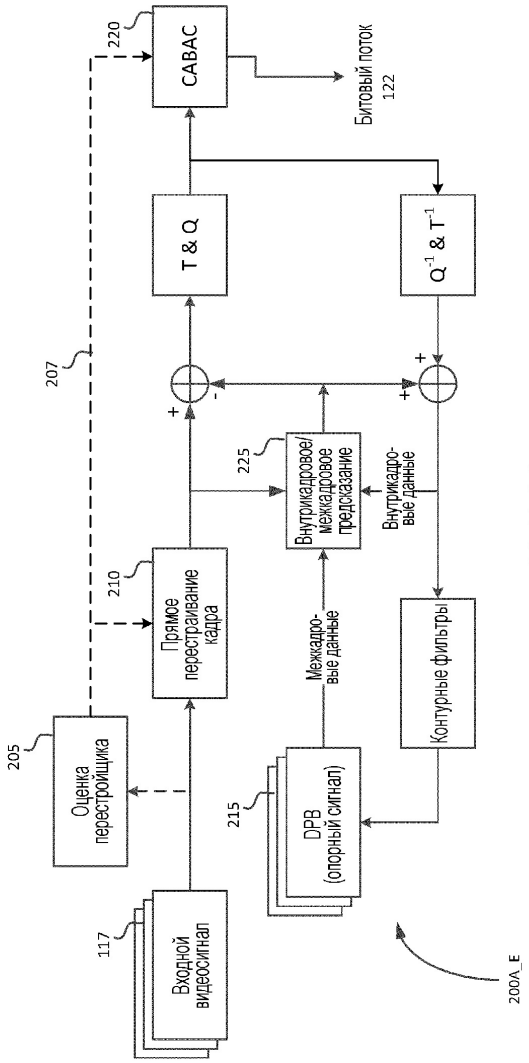
ФИГ. 1A



ФИГ. 1B

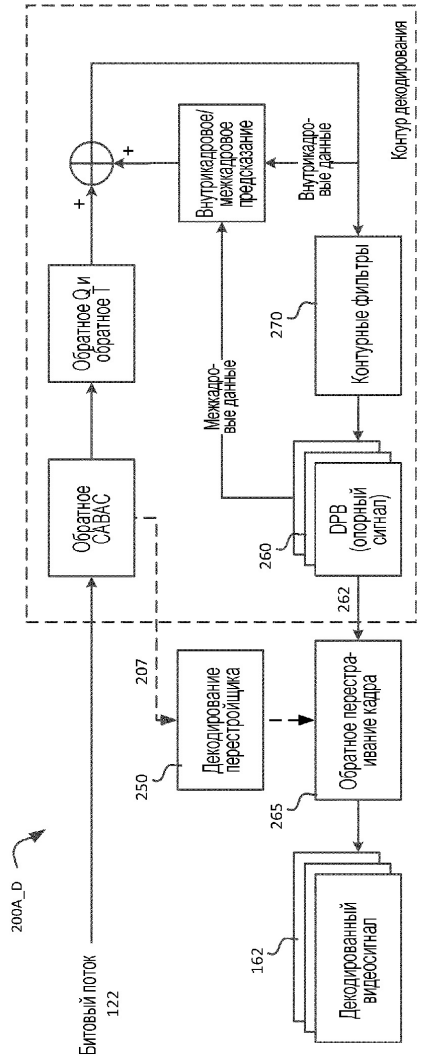
(Прототип)

2

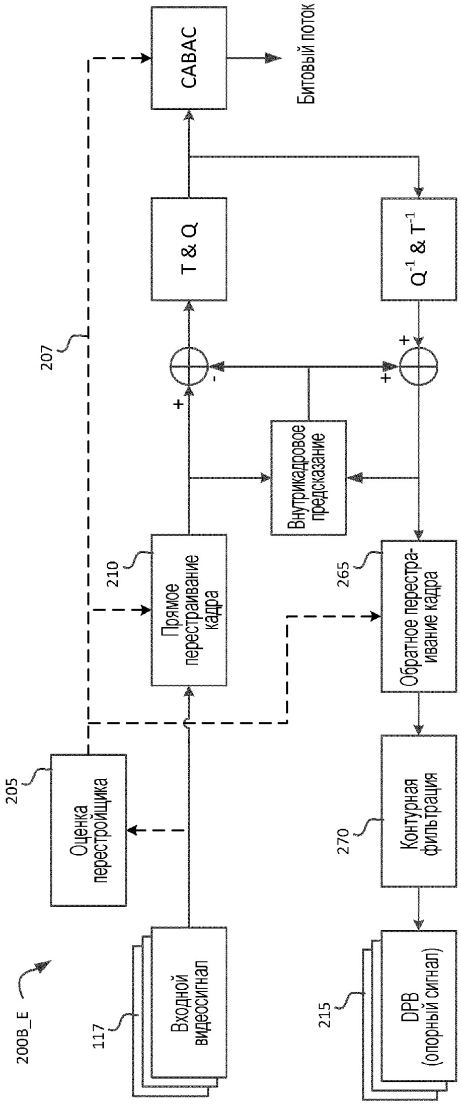

220
CABAC
Битовый поток
122
T & Q
Q^{-1} & T^{-1}
207
225
Внутрикадровое/
межкадровое
предсказание
Внутрикадро-
вые данные
210
Прямое
перестраивание
кадра
Межкадро-
вые данные
Контурные фильтры
205
Оценка
перестройщика
215
DPB
(опорный сигнал)
117
Входной
видеосигнал
200A_E

ФИГ. 2A

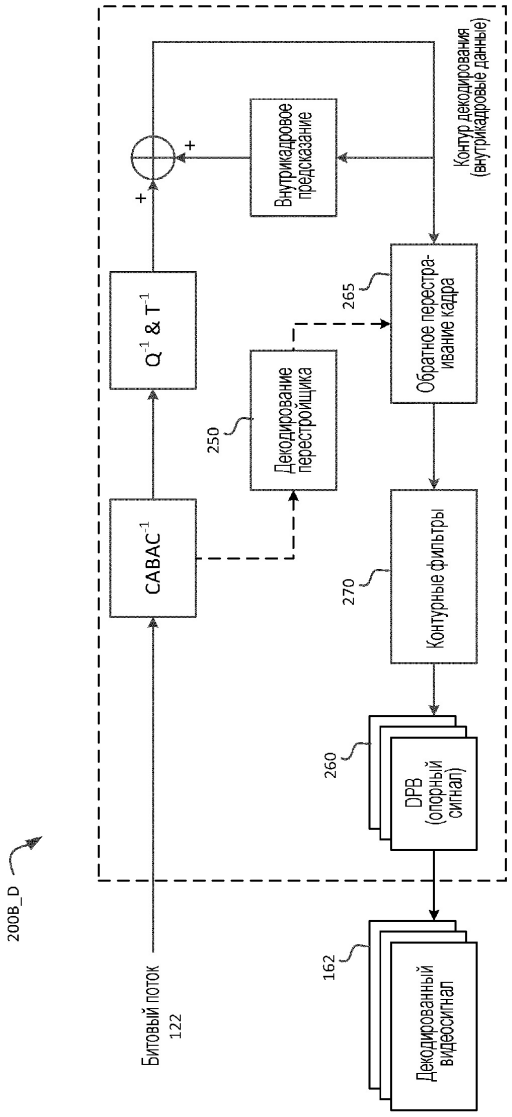
3


Битовый поток
122
200A_D
Обратное CAVAC
Обратное Q и обратное T
Внутрикадровое/ межкадровое предсказание
Межкадро-вые данные
Внутрикадро-вые данные
Контур декодирования
270
Контурные фильтры
260
DPB (опорный сигнал)
262
207
250
Декодирование перестройщика
265
Обратное перестра-ивание кадра
162
Декодированный видеосигнал

ФИГ.2В

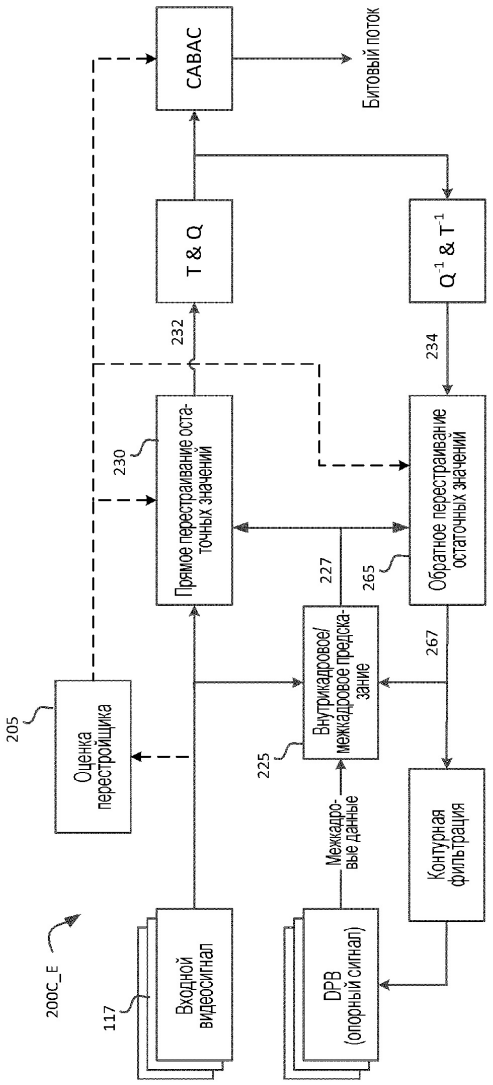

CABAC
Битовый поток
T & Q
Q^{-1} & T^{-1}
207
Внутрикадровое предсказание
210
Прямое перестраивание кадра
265
Обратное перестраивание кадра
205
Оценка перестройщика
270
Контурная фильтрация
200B_E
117
Входной видеосигнал
215
DPB (опорный сигнал)

ФИГ. 2C

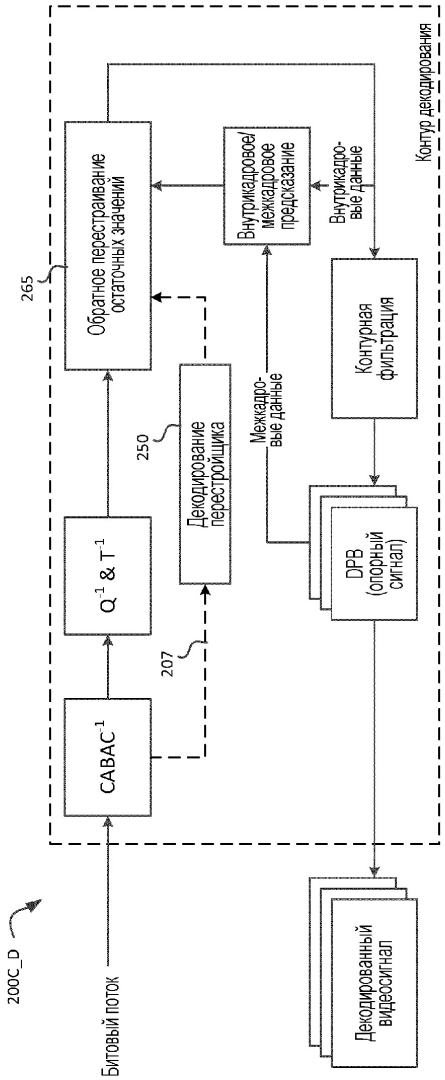

200B_D
Битовый поток
122
CABAC^{-1}
Q^{-1} & T^{-1}
Декодирование перестроящика
250
Внутрикадровое предсказание
Контур декодирования (внутрикадровые данные)
265
Обратное перестраивание кадра
270
Контурные фильтры
260
DPB (опорный сигнал)
162
Декодированный видеосигнал

ФИГ. 2D

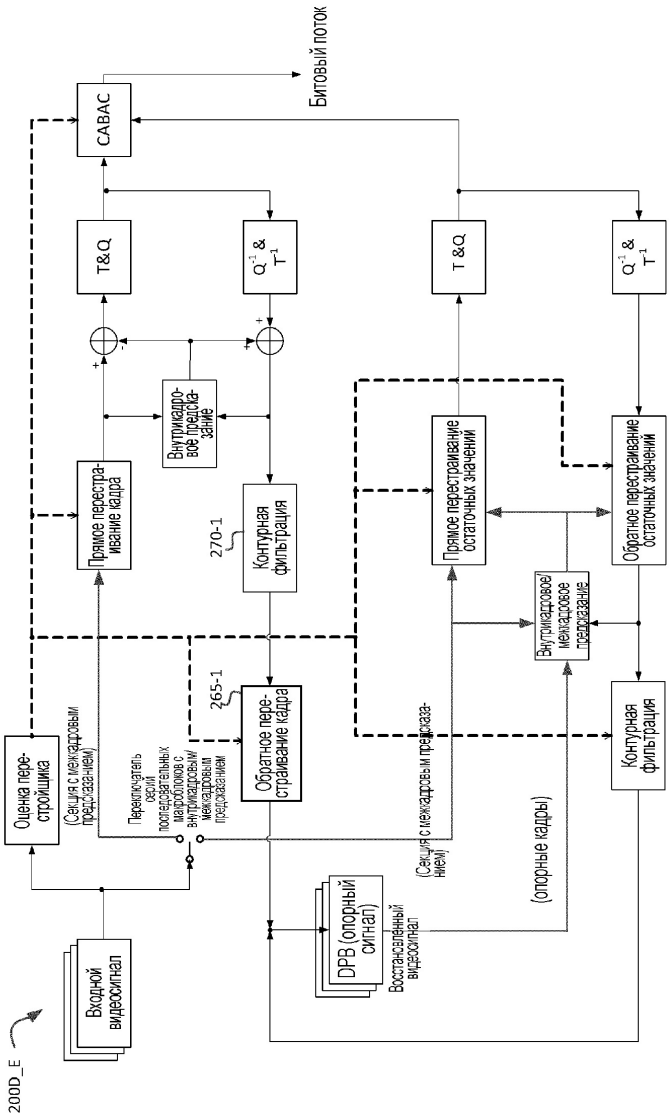
6


CABAC
Битовый поток
T & Q
Q^{-1} & T^{-1}
232
234
230
Прямое перестраивание остаточных значений
Обратное перестраивание остаточных значений
227
265
267
205
Оценка перестройщика
Внутрикадровое/межкадровое предсказание
225
Межкадровые данные
Контурная фильтрация
200c_E
117
Входной видеосигнал
DPB (опорный сигнал)

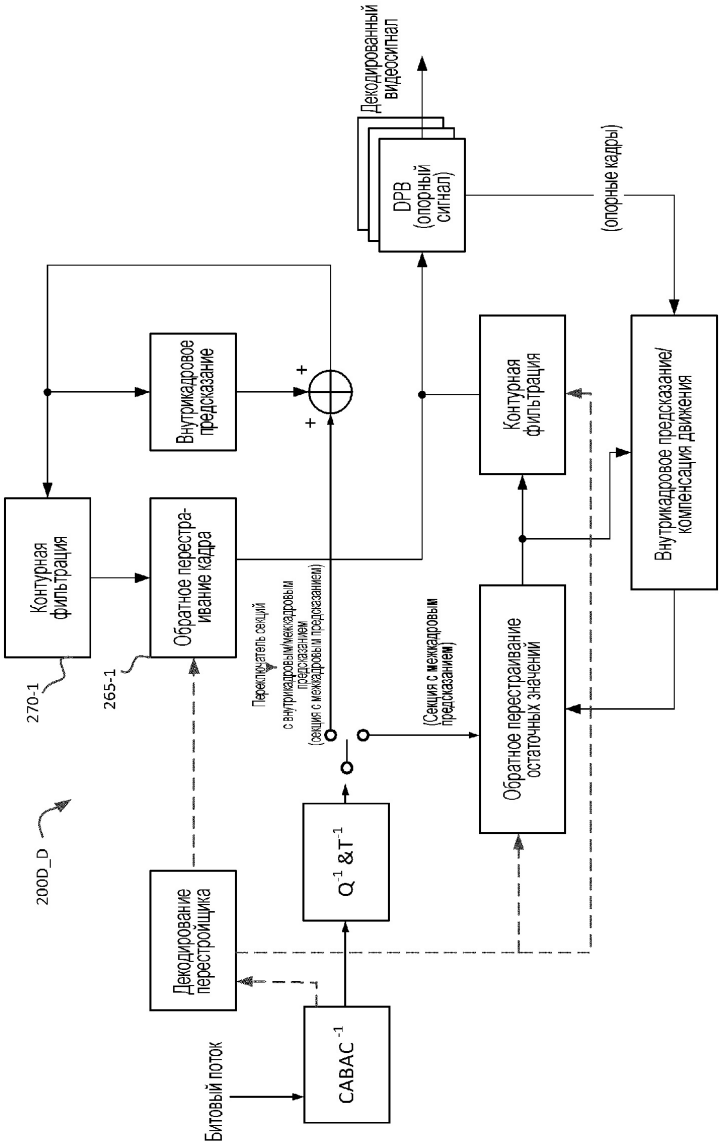
ФИГ. 2E


Контур декодирования
Обратное перестраивание остаточных значений
265
Внутрикадровое/ межкадровое предсказание
Внутрикадро-вые данные
Контурная фильтрация
Декодирование перестройщика
250
Межкадро-вые данные
Q^{-1} & T^{-1}
DPB (опорный сигнал)
207
CABAC^{-1}
200C_D
Битовый поток
Декодированный видеосигнал

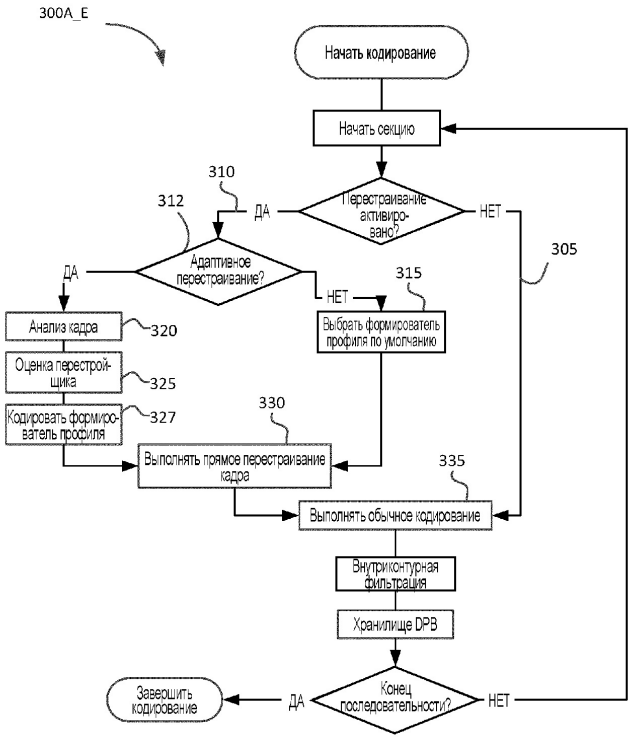
ФИГ.2F


Битовый поток
CABAC
T&Q
Q^{-1} & T^{-1}
T &Q
Q^{-1} & T^{-1}
Внутрикадровое предсказание
Прямое перестраивание кадра
270-1
Контурная фильтрация
Прямое перестраивание остаточных значений
Обратное перестраивание остаточных значений
Внутрикадровое/ межкадровое предсказание
265-1
Обратное перестраивание кадра
Контурная фильтрация
Оценка перестройщика
(Секция с межкадровым предсказанием)
Переключатель сессий повторяющихся макроблоков с внутрикадровым/ межкадровым предсказанием
(Секция с межкадровым предсказанием)
(опорные кадры)
DPB (опорный сигнал)
Восстановленный видеосигнал
Входной видеосигнал
200D_E

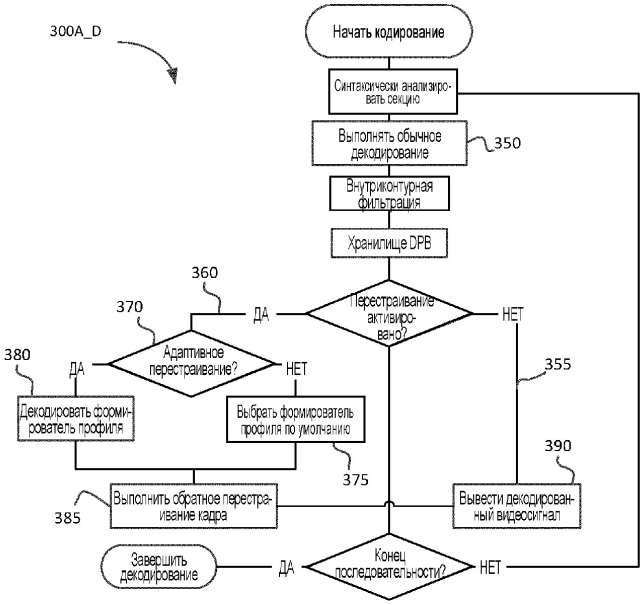
ФИГ.2G


Декодированный видеосигнал
DPB (опорный сигнал)
(опорные кадры)
Внутрикадровое предсказание
Контурная фильтрация
Внутрикадровое предсказание/ компенсация движения
Контурная фильтрация
Обратное перестра-ивание кадра
Переключатель секций с внутрикадровым/межкадровым предсказанием (секция с межкадровым предсказанием)
(Секция с межкадровым предсказанием)
Обратное перестраивание остаточных значений
270-1
265-1
200D_D
Декодирование перестройщика
Q^{-1} &T^{-1}
CABAC $^{-1}$
Битовый поток

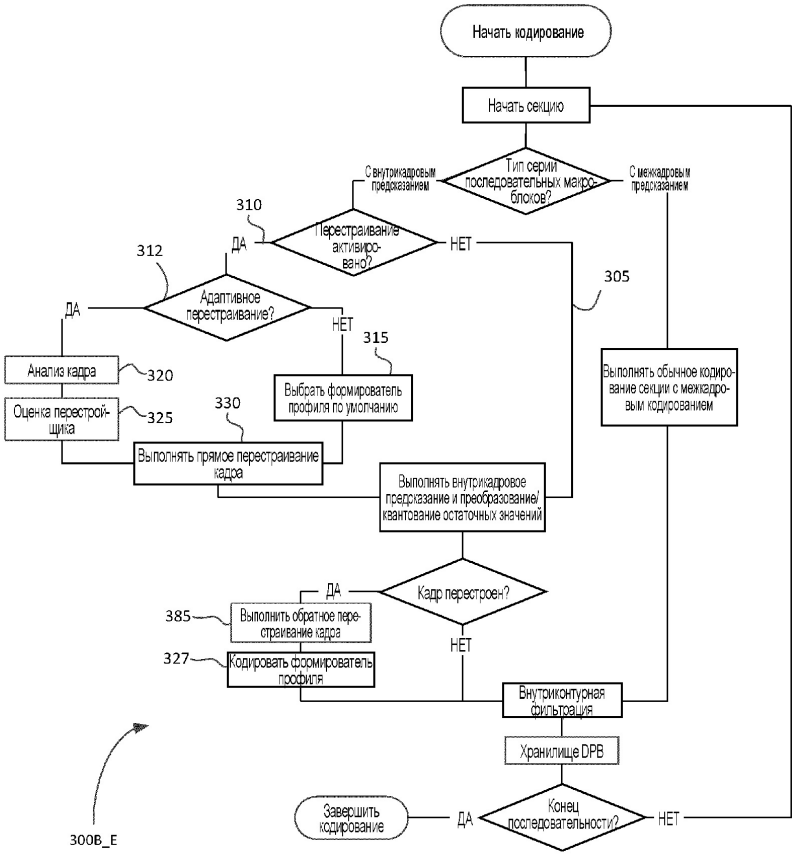
ФИГ.2Н


300A_E
Начать кодирование
Начать секцию
310
312
ДА
Перестраивание активировано?
НЕТ
305
ДА
Адаптивное перестраивание?
НЕТ
315
Анализ кадра
320
Выбрать формирователь профиля по умолчанию
Оценка перестройщика
325
Кодировать формирователь профиля
327
330
Выполнять прямое перестраивание кадра
335
Выполнять обычное кодирование
Внутриконтурная фильтрация
Хранилище DPB
Завершить кодирование
ДА
Конец последовательности?
НЕТ

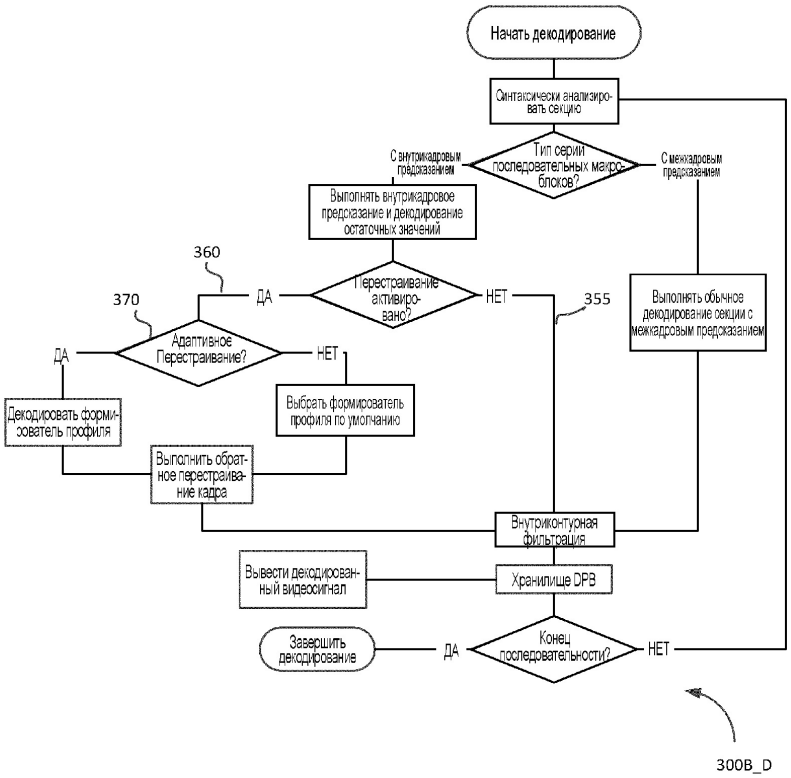
ФИГ. 3А


300A_D
Начать кодирование
Синтаксически анализировать секцию
Выполнять обычное декодирование
350
Внутриконтурная фильтрация
Хранилище DPB
360
Перестраивание активировано?
ДА
НЕТ
370
Адаптивное перестраивание?
ДА
НЕТ
380
Декодировать формирователь профиля
Выбрать формирователь профиля по умолчанию
375
355
390
Выполнить обратное перестраивание кадра
385
Вывести декодированный видеосигнал
Завершить декодирование
ДА
Конец последовательности?
НЕТ

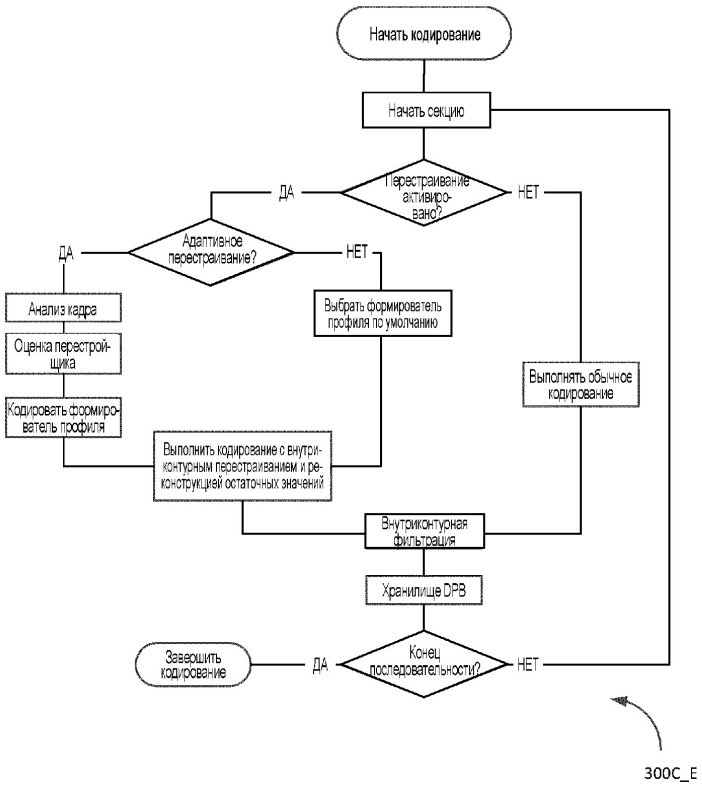
ФИГ. 3В


Начать кодирование
Начать секцию
Тип серии последовательных макроблоков?
С внутрикадровым предсказанием
С межкадровым предсказанием
310
Перестраивание активировано?
ДА
НЕТ
312
Адаптивное перестраивание?
ДА
НЕТ
305
315
Анализ кадра
320
Оценка перестройщика
325
330
Выбрать формирователь профиля по умолчанию
Выполнять прямое перестраивание кадра
Выполнять обычное кодирование секции с межкадровым кодированием
Выполнять внутрикадровое предсказание и преобразование/квантование остаточных значений
Кадр перестроен?
ДА
НЕТ
385
Выполнить обратное перестраивание кадра
327
Кодировать формирователь профиля
Внутриконтурная фильтрация
Хранилище DPB
Конец последовательности?
ДА
НЕТ
Завершить кодирование
300B_E

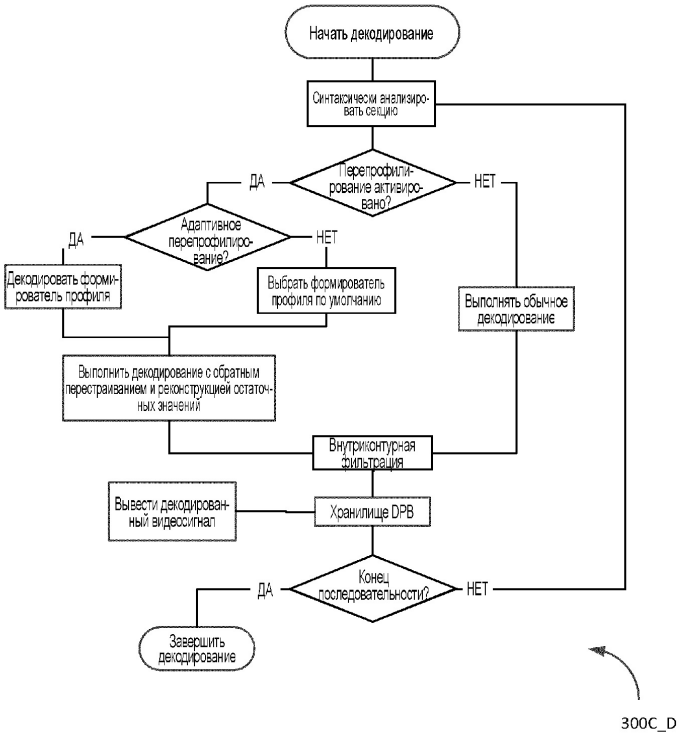
ФИГ. 3C


Начать декодирование
Синтаксически анализировать секцию
Тип серии последовательных макроблоков?
С внутрикадровым предсказанием
С межкадровым предсказанием
Выполнять внутрикадровое предсказание и декодирование остаточных значений
360
Перестраивание активировано?
ДА
НЕТ
355
370
Адаптивное Перестраивание?
ДА
НЕТ
Выполнять обычное декодирование секции с межкадровым предсказанием
Декодировать формирователь профиля
Выбрать формирователь профиля по умолчанию
Выполнить обратное перестраивание кадра
Внутриконтурная фильтрация
Вывести декодированный видеосигнал
Хранилище DPB
Завершить декодирование
ДА
Конец последовательности?
НЕТ
300B_D

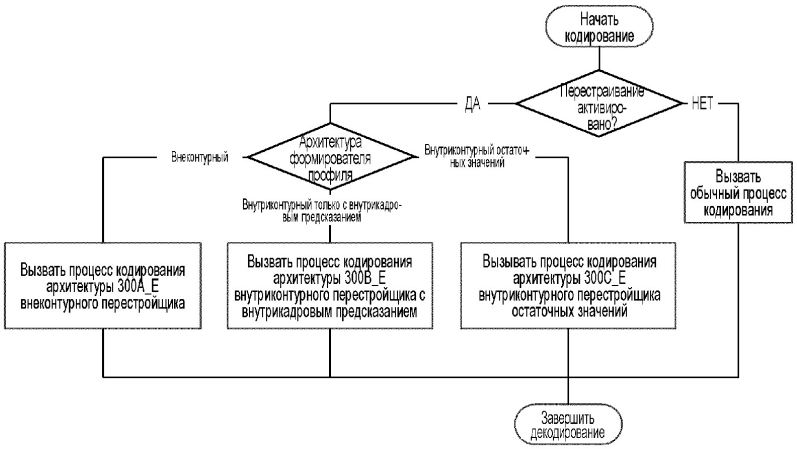
ФИГ. 3D


Начать кодирование
Начать секцию
Перестраивание активировано?
ДА
НЕТ
Адаптивное перестраивание?
ДА
НЕТ
Анализ кадра
Оценка перестройщика
Кодировать формирователь профиля
Выбрать формирователь профиля по умолчанию
Выполнить кодирование с внутриконтурным перестраиванием и реконструкцией остаточных значений
Выполнять обычное кодирование
Внутриконтурная фильтрация
Хранилище DPB
Завершить кодирование
ДА
Конец последовательности?
НЕТ
300C_E

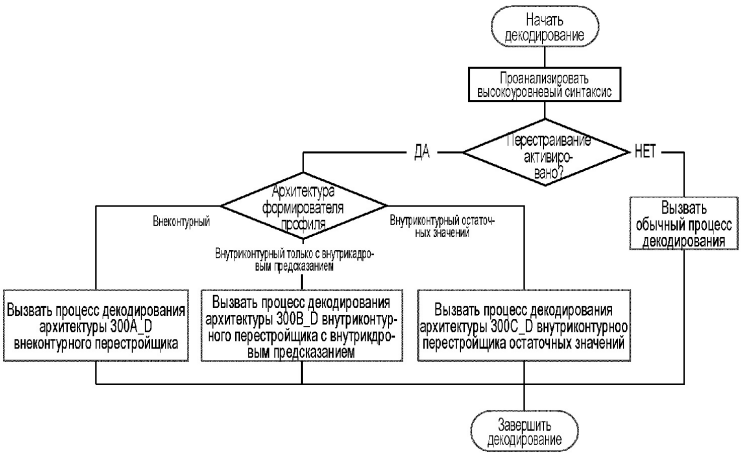
ФИГ. 3E


Начать декодирование
Синтаксически анализировать секцию
Перепрофилирование активировано?
ДА
НЕТ
Адаптивное перепрофилирование?
ДА
НЕТ
Декодировать формирователь профиля
Выбрать формирователь профиля по умолчанию
Выполнять обычное декодирование
Выполнить декодирование с обратным переставлением и реконструкцией остаточных значений
Внутриконтурная фильтрация
Вывести декодированный видеосигнал
Хранилище DPB
Конец последовательности?
ДА
НЕТ
Завершить декодирование
300C_D

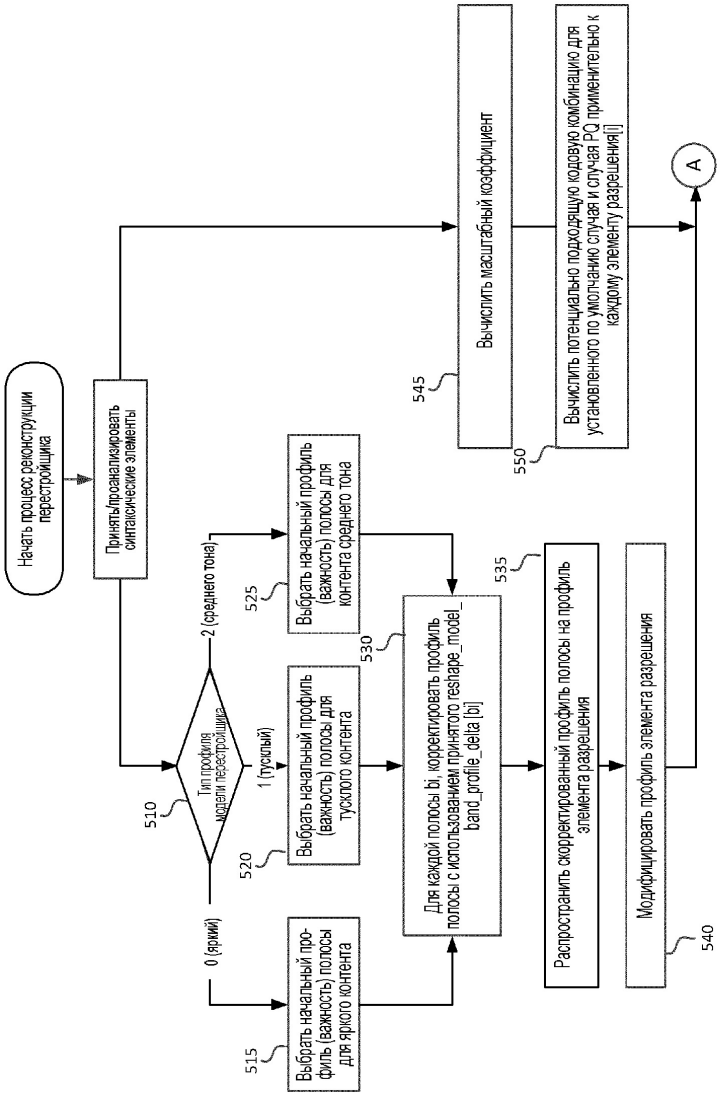
ФИГ. 3F


Начать кодирование
Перестраивание активировано?
ДА
НЕТ
Архитектура формирователя профиля
Внеконтурный
Внутриконтурный остаточных значений
Внутриконтурный только с внутрикадровым предсказанием
Вызвать обычный процесс кодирования
Вызвать процесс кодирования архитектуры 300A_E внеконтурного перестройщика
Вызвать процесс кодирования архитектуры 300B_E внутриконтурного перестройщика с внутрикадровым предсказанием
Вызывать процесс кодирования архитектуры 300C_E внутриконтурного перестройщика остаточных значений
Завершить декодирование

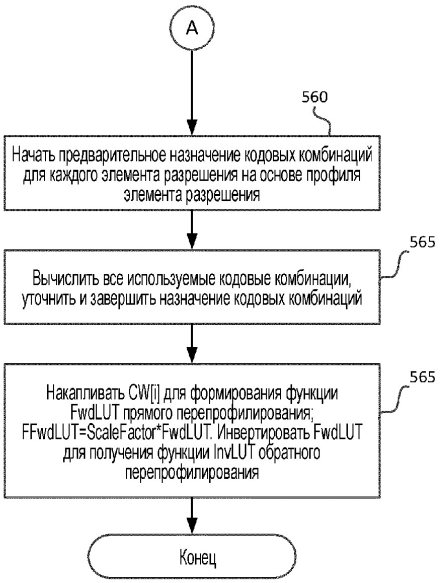
ФИГ. 4A


Начать декодирование
Проанализировать высокоуровневый синтаксис
Перестраивание активировано?
ДА
НЕТ
Архитектура формирователя профиля
Внеконтурный
Внутриконтурный остаточных значений
Внутриконтурный только с внутрикадровым предсказанием
Вызвать обычный процесс декодирования
Вызвать процесс декодирования архитектуры 300A_D внеконтурного перестройщика
Вызвать процесс декодирования архитектуры 300B_D внутриконтурного перестройщика с внутрикадровым предсказанием
Вызвать процесс декодирования архитектуры 300C_D внутриконтурно перестройщика остаточных значений
Завершить декодирование

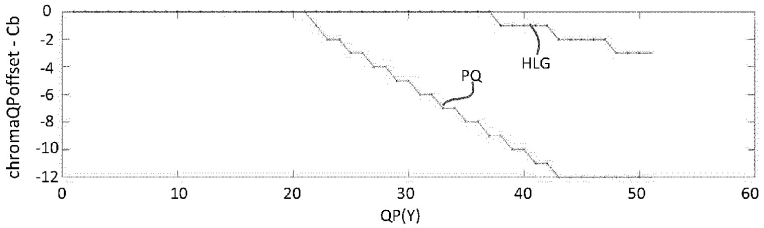
ФИГ. 4B


Начать процесс реконструкции перестроишика
Принять/проанализировать синтаксические элементы
Тип профиля модели перестроишика
0 (яркий)
1 (тусклый)
2 (среднего тона)
510
515
Выбрать начальный профиль (важность) полосы для яркого контента
520
Выбрать начальный профиль (важность) полосы для тусклого контента
525
Выбрать начальный профиль (важность) полосы для контента среднего тона
530
Для каждой полосы bi, корректировать профиль полосы с использованием принятого reshape_model_ band_profile_delta [bi]
535
Распространить скорректированный профиль полосы на профиль элемента разрешения
540
Модифицировать профиль элемента разрешения
545
Вычислить масштабный коэффициент
550
Вычислить потенциально подходящую кодовую комбинацию для установленного по умолчанию случая и случая PQ применительно к каждому элементу разрешения[i]
A

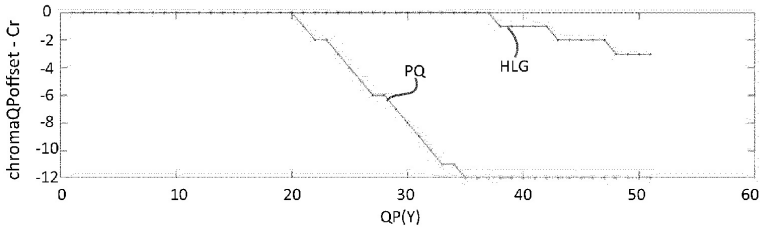
ФИГ. 5А


A
560
Начать предварительное назначение кодовых комбинаций для каждого элемента разрешения на основе профиля элемента разрешения
565
Вычислить все используемые кодовые комбинации, уточнить и завершить назначение кодовых комбинаций
565
Накапливать CW[i] для формирования функции FwdLUT прямого перепрофилирования; FFwdLUT=ScaleFactor*FwdLUT. Инвертировать FwdLUT для получения функции InvLUT обратного перепрофилирования
Конец

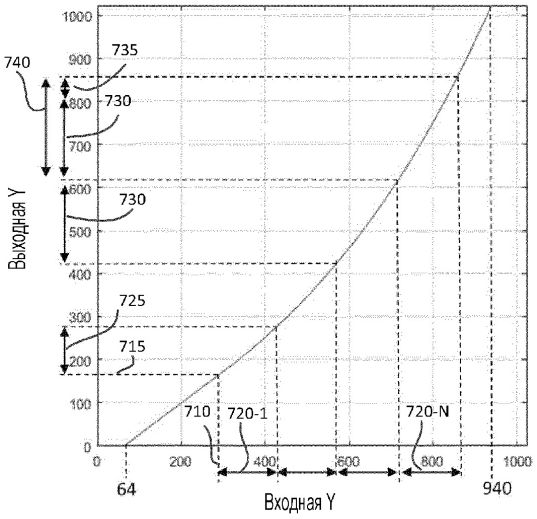
ФИГ. 5B


chromaQPoffset - Cb
0
-2
-4
-6
-8
-10
-12
0
10
20
30
40
50
60
QP(Y)
HLG
PQ

ФИГ. 6A


chromaQPoffset - Cr
0
-2
-4
-6
-8
-10
-12
0
10
20
30
40
50
60
QP(Y)
HLG
PQ

ФИГ. 6B


740
735
730
730
725
715
710
720-1
720-N
Выходная Y
Входная Y
64
940
0
100
200
300
400
500
600
700
800
900
1000
0
200
400
600
800
1000

ФИГ. 7