УКРАЇНА

(19) **UA** (11) **124569** (13) **C2**

(51) МПК (2021.01)
H04N 7/00
H04N 19/107 (2014.01)
H04N 19/13 (2014.01)
H04N 19/169 (2014.01)
H04N 19/65 (2014.01)

НАЦІОНАЛЬНИЙ ОРГАН ІНТЕЛЕКТУАЛЬНОЇ ВЛАСНОСТІ ДЕРЖАВНЕ ПІДПРИЄМСТВО "УКРАЇНСЬКИЙ ІНСТИТУТ ІНТЕЛЕКТУАЛЬНОЇ ВЛАСНОСТІ"

(12) ОПИС ДО ПАТЕНТУ НА ВИНАХІД

(21) Номер заявки: **a 2017 02254**

(22) Дата подання заявки: **21.01.2013**

(24) Дата, з якої є чинними права інтелектуальної власності: **14.10.2021**

(31) Номер попередньої заявки відповідно до Паризької конвенції: **61/588,849**

(32) Дата подання попередньої заявки відповідно до Паризької конвенції: **20.01.2012**

(33) Код держави-учасниці Паризької конвенції, до якої подано попередню заявку: **US**

(41) Публікація відомостей про заявку: **11.09.2017, Бюл.№ 17**

(46) Публікація відомостей про державну реєстрацію: **13.10.2021, Бюл.№ 41**

(62) Номер та дата подання попередньої заявки, з якої виділено заявку, позначену кодом (21): **, a201409286, 21.01.2013**

(72) Винахідник(и):
Шірль Томас (DE),
Георге Валері (DE),
Грюнеберг Карстен (DE),
Кірххоффер Хайнер (DE),
Хенкель Анастасія (DE),
Марпе Детлеф (DE)

(73) Володілець (володільці):
ДЖ.І. ВІДІЕУ КЕМПРЕШН, ЛЛСІ,
8 Southwoods Boulevard, Albany, New York 12211, USA (US)

(74) Представник:
Пахаренко Антоніна Павлівна, реєстр. №4

(56) Перелік документів, взятих до уваги експертизою:
EP 2266319 A1, 29.12.2010
WO 2007077942 A1, 12.07.2007
EP 1758399 A1, 28.02.2007
Gordon C., Henry F., Pateux S. Wavefront Parallel Processing for HEVC Encoding and Decoding. 6. JCT-VC meeting. 97. MPEG meeting; 14.07.2011 – 22.07.2011; Torino; (Joint Collaborative Team on Video Coding of ISO/IEC JTC1/SC29/WG11 and ITU-T SG.16), JCTVC-F274, XP 030009297

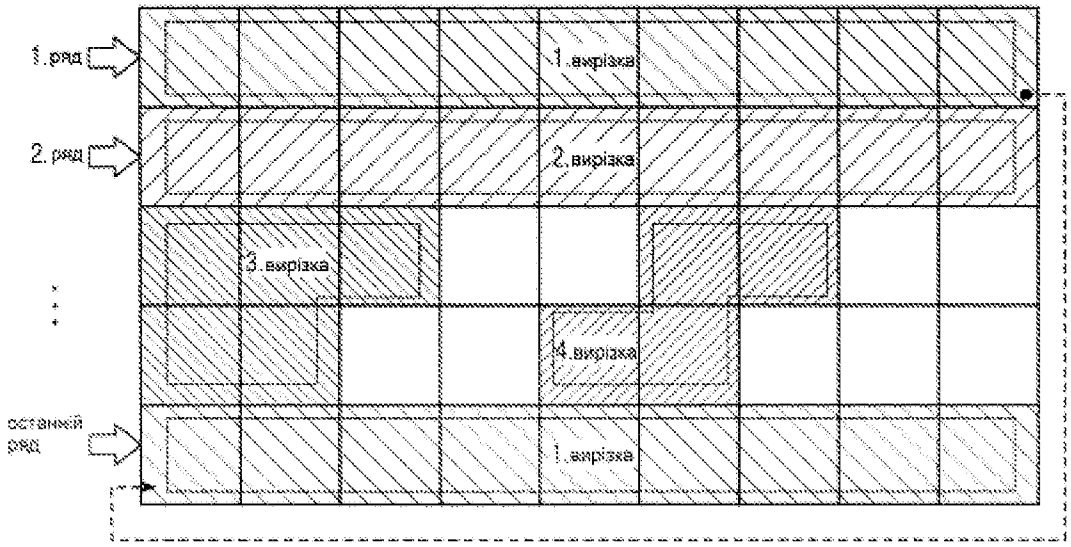
(54) КОНЦЕПЦІЯ КОДУВАННЯ, ЯКА ДОЗВОЛЯЄ ПАРАЛЕЛЬНУ ОБРОБКУ ДАНИХ, ТРАНСПОРТНИЙ ДЕМУЛЬТИПЛЕКСОР І ВІДЕОБІТОВИЙ ПОТІК

(57) Реферат:

Корисну інформацію необробленої послідовності байтів, яка описує картинку у вирізках, вкладених потоках WPP або мозаїках і кодована з використанням адаптивного до контексту арифметичного кодування, підрозбивають або розрізають на транші з продовженням адаптації ймовірності адаптивного до контексту арифметичного кодування крізь границі траншів. Завдяки цьому заходу границі траншів, додатково введені у вирізках, вкладених потоках WPP або мозаїках, не приводять до зниження ефективності ентропійного кодування цих елементів. Однак, з іншого боку, транші менші за первинні вирізки, вкладені потоки WPP або мозаїки і, відповідно, вони можуть передаватися раніше, тобто з малою затримкою, ніж нерозрізані первинні об'єкти, тобто вирізки, вкладені потоки WPP або мозаїки. Згідно з іншим аспектом, який здатен поєднуватися з першим аспектом, блоки NAL маркерів вкладених потоків використовуються в послідовності блоків NAL відеобітового потоку для надання можливості

UA 124569 C2

транспортному демультиплексору присвоювати дані вирізок в блоках NAL відповідним вкладеним потокам або мозаїкам для надання можливості паралельно живити декодери з багатьма послідовностями команд відповідними вкладеними потоками або мозаїками.



Фіг. 1

Представлений винахід стосується концепцій кодування, які дозволяють паралельну обробку даних, таку як у розроблюваному стандарті HEVC, транспортного демультиплексора і відеобітового потоку.

Розпаралелювання роботи кодера і декодера є дуже важливим внаслідок зростаючих вимог стандарту HEVC до обробки даних, а також очікуваного збільшення роздільної здатності відеозображення. Багатоядерні архітектури стають доступними в широкому різновиді сучасних електронних пристроїв. Тому, ефективні способи надають можливість використання необхідних багатоядерних архітектур.

Кодування або декодування LCUs (найбільша кодувальна чарунка) відбувається в растровій розгортці, за допомогою якої ймовірності CABAC (контекстно-адаптивне бінарне арифметичне кодування) адаптуються до специфічних ознак кожного зображення. Між сусідніми LCUs існують просторові залежності. Кожна LCU залежить від своїх лівих, верхніх, верхніх лівих і верхніх правих сусідніх LCUs, завдяки різним компонентам, наприклад, вектору руху, прогнозу, інтрапрогнозу та інших. Для надання можливості розпаралелювання в процесі декодування, ці залежності типово потребують переривання або перериваються у застосуваннях рівня техніки.

Були запропоновані деякі концепції розпаралелювання, зокрема хвильова обробка даних з використанням ентропійних вирізок [3], операції хвильової паралельної обробки даних (WPP) з використанням вкладених потоків [2], [4], [11], або мозаїк [5]. Останнє не обов'язково потребує об'єднання з хвильовою обробкою даних для надання можливості розпаралелювання роботи декодера або кодера. З цієї точки зору, мозаїки подібні до вкладених потоків WPP. Нашим початковим мотиватором для подальшого вивчення концепції ентропійної вирізки є використання технологій, які знижують втрату ефективності кодування і, таким чином, зменшують навантаження на потік бітів для наближень розпаралелювання операцій в кодері та декодері.

Для забезпечення кращого розуміння, зокрема використання LCUs, можна спершу розглянути структуру стандарту H.264/AVC [1].

Кодована відеопослідовність в стандарті H.264/AVC складається з ряду блоків доступу, які збираються в потоці блоків NAL, і вони використовують тільки один набір параметрів послідовності. Кожна відеопослідовність може незалежно декодуватися. Кодована послідовність складається з послідовності кодованих картинок. Кодований кадр може бути цілим кадром або єдиним полем. Кожна картинка розбивається на макроблоки фіксованого розміру (в стандарті HEVC [5]: LCUs). Декілька макроблоків або LCUs можуть об'єднуватися між собою в одну вирізку. Тому, картинка є сукупністю однієї або більшої кількості вирізок. Ціллю цього розділення даних є надання можливості незалежного декодування зразків на ділянці картинки, яка представлена вирізкою, без використання даних від інших вирізок.

Технологія, яка часто називається "ентропійними вирізками" [3], полягає у розбитті традиційної вирізки на додаткові субвирізки. Точніше, це означає фрагментацію ентропійно кодованих даних єдиної вирізки. Розташування ентропійних вирізок у вирізці може мати різні варіанти. Найпростішим варіантом є використання кожного рядка LCUs/макроблоків в кадрі як одну ентропійну вирізку. Альтернативно, стовпчики або окремі ділянки можуть використовуватися як ентропійні вирізки, які навіть можуть перериватися і з'єднуватися між собою, наприклад вирізка 1 на Фіг. 1.

Очевидною ціллю концепції ентропійної вирізки є надання можливості паралельного використання CPU(Центральний Процесор)/GPU(Графічний Процесор) і багатоядерних архітектур для покращення тривалості процесу декодування, тобто для прискорення процесу. Поточна вирізка може ділитися на частини, які можуть піддаватися синтаксичному аналізу і відновлюватися без посилання на інші дані вирізки. Хоча може досягатися пару переваг з використанням наближення ентропійної вирізки, таким чином створюючи деякі проблеми.

Концепція ентропійної вирізки була додатково поширена на хвильову обробку вкладеного потоку даних (WPP), як запропоновано в документах [2], [10], [11], і частково включено в роботу [5]. Тут визначається схема повторення вкладених потоків, яка має кращу ініціалізацію ентропійного стану на рядок порівняно з ентропійними вирізками.

Концепція мозаїки дозволяє розділення інформації картинки, яка кодується, тоді як кожна мозаїка має свій власний порядок растрової розгортки. Мозаїка визначається спільною структурою, яка повторюється в кадрі. Мозаїка може також мати певну ширину стовпчика і висоту рядка в термінах LCUs або CUs. Мозаїки можуть також незалежно кодуватися і можуть також кодуватися у такий спосіб, що вони не вимагають сумісної обробки з іншими мозаїками так, що послідовності команд декодера можуть обробляти мозаїки Блока Доступу повністю або принаймні для деяких етапів операції кодування незалежним чином, тобто ентропійно кодувати і кодувати з перетворенням.

Тому, мозаїка в значній мірі дозволяє запускати кодери мозаїки, а також декодери повністю або частково незалежно паралельним чином - в останньому випадку, наприклад, до етапу фільтрування кодека HEVC.

Для повного використання технологій розпаралелювання в уловлюванні, кодуванні, передачі, декодуванні і презентаційному ланцюгу системи відеозв'язку або подібних систем, передача і доступ до даних між учасниками зв'язку є важливим і довготривалим етапом для введення наскрізної затримки. Це головним чином є проблемою, якщо використовувати технології розпаралелювання, такі як мозаїки, вкладені потоки або ентропійні вирізки.

Наближення даних вкладених потоків WPP припускають, що кодовані дані частин, якщо вони обробляються, не мають локалізації даних, тобто єдина послідовність команд, яка декодує Блок Доступу, потребує перестрибувати через потенційно великі частини пам'яті для доступу до даних наступного вкладеного потоку WPP. Декодувальна система з багатьма послідовностями команд потребує очікування на передачу певних даних, тобто вкладених потоків WPP, для роботи повністю розпаралеленим чином з використанням хвильової обробки даних.

В потоковому відео надання можливості використання вищих роздільних здатностей (Full-HD, QUAD-HD і так далі) приводить до більшої кількості даних, які повинні передаватися. Для чутливих до часу сценаріїв, так званих випадків використання Малої Затримки, таких як відеоконференція (<145 мс) або ігри (<40 мс), потребуються наскрізні затримки. Тому, тривалість передачі даних стає критичним фактором. Дивіться завантажувальний лінк ADSL для проведення відеоконференцій. Тут, так звані точки довільного доступу потоку (зазвичай це належить до I-інтракадрів) будуть кандидатами для появи критичного параметра під час передачі даних.

Стандарт HEVC передбачає так звану хвильову обробку даних, а також мозаїчну обробку даних в кодері, а також в декодері. Це дозволяється використанням ентропійних вирізок, вкладених потоків WPP або навіть їх комбінації. Паралельна обробка також дозволена паралельним кодуванням і декодуванням мозаїки.

У випадку "нерозпаралелювального позиціювання", дані усієї вирізки повинні надаватися за раз, таким чином, остання CU вирізок доступна декодеру, якщо вона була передана. Це не є проблемою, якщо є декодер з єдиною послідовністю команд.

У випадку багатьох послідовностей команд, якщо може використовуватися багато CPUs або ядер, процес декодування повинен, однак, розпочинатися зразу при надходженні кодованих даних до декодера хвильової обробки даних або до послідовностей команд декодера мозаїки.

Таким чином, повинно бути вигідним мати під рукою концепції, які надають можливість зменшувати затримку кодування в середовищах з паралельною обробкою даних з менш різкими зниженнями ефективності кодування.

Відповідно, задачею представленого винаходу є надання концепції кодування, концепції транспортного демультиплексування і відеобітових потоків, яка дозволяє таке ефективніше кодування з малою затримкою в середовищах з паралельною обробкою даних.

Ця задача вирішується об'єктом доданих незалежних пунктів формули винаходу.

Згідно з першим аспектом представленого винаходу корисна інформація необробленої послідовності байтів, яка описує картинку у вирізках, вкладених потоках WPP або мозаїках і кодована з використанням адаптивного до контексту бінарного арифметичного кодування, підрозбивається або розрізається на транші з продовженням адаптивного до контексту бінарного арифметичного кодування з адаптацією ймовірності крізь границі траншів. За допомогою цього заходу границі траншів, додатково введені у вирізках, вкладених потоках WPP або мозаїках, не приводять до зниження ефективності ентропійного кодування цих елементів. З іншого боку, однак, транші менші за первинні вирізки, вкладені потоки WPP або мозаїки і, відповідно, вони можуть передаватися раніше, тобто з малою затримкою, ніж нерозрізані первинні об'єкти, тобто вирізки, вкладені потоки WPP або мозаїки.

Згідно з іншим аспектом, який може поєднуватися з першим аспектом, блоки NAL маркера вкладеного потоку використовуються в послідовності блоків NAL відеобітового потоку для надання можливості транспортному демультиплексору присвоювати дані вирізок в блоках NAL відповідним вкладеним потокам або мозаїкам для надання можливості паралельно подавати відповідні вкладені потоки або мозаїки декодеру з багатьма послідовностями команд.

Переважні втілення є об'єктом залежних пунктів формули винаходу. Окрім того, переважні варіанти виконання представленого винаходу пояснюються детальніше нижче стосовно фігур, серед яких

Фіг. 1 зображає схематичну ілюстрацію можливих структур ентропійних вирізок;

Фіг. 2 зображає схематичну ілюстрацію трьох мозаїк, розподілених серед трьох вирізок;

Фіг. 3 зображає схематичну ілюстрацію прикладу чергування траншів циклічної схеми перемежовування чотирьох траншів змінної довжини;

Фіг. 4 зображає схематичну ілюстрацію кодування, сегментації, перемежовування і декодування даних ентропійної вирізки;

Фіг. 5 зображає схематичну ілюстрацію прикладу перемежовування траншів циклічної схеми перемежовування чотирьох траншів змінної довжини, яка завжди використовує маркерні коди і розподіл реальних даних вирізки по багатьох блоках NAL. Маркерні коди використовуються, навіть якщо границя відсутня. Це може додатково покращуватися з використанням ідентифікатора траншу, який відповідає маркеру, який вказує номер траншу. Це завжди усуває потребу надсилання маркера, як це вимагається для циклічного режиму.

Фіг. 6 зображає таблицю псевдокодів, яка показує синтаксис блока NAL

Фіг. 7 зображає таблицю псевдокодів, яка показує синтаксис множини параметрів послідовності;

Фіг. 8 зображає таблицю псевдокодів, яка показує синтаксис RBSP шару Вирізки з Малою Затримкою;

Фіг. 9 зображає таблицю псевдокоду, яка показує синтаксис заголовка вирізки;

Фіг. 10 зображає таблицю псевдокоду, яка показує синтаксис маркера вкладеного потоку;

Фіг. 11 зображає схематичну ілюстрацію прикладу простої інкапсуляцій даних ентропійної вирізки (AF є Полем Адаптації MPEG-2 TS);

Фіг. 12 зображає схематичну ілюстрацію іншого прикладу інкапсуляції єдиного елементарного потоку (ES) даних ентропійної вирізки;

Фіг. 13 зображає схематичну ілюстрацію іншого прикладу інкапсуляції багатьох елементарних потоків (ES) даних ентропійної вирізки;

Фіг. 14 зображає блок-схему, яка показує транспортний демультиплексор для єдиного елементарного потоку (ES); і

Фіг. 15 зображає блок-схему, яка показує транспортний демультиплексор для багатьох елементарних потоків (ES).

Фіг. 16 зображає схему, яка показує кодер;

Фіг. 17 зображає схему, яка показує декодер;

Фіг. 18 зображає послідовність етапів, виконуваних декодером; і

Фіг. 19 зображає схематичну ілюстрацію прикладу багатьох елементарних потоків (ES), який використовує RTP (транспортний протокол реального часу).

Для зменшення тривалості, з якою послідовність команд паралельного декодера може починати і закінчувати обробку своїх даних кадру, нижченаведені варіанти виконання використовують сегментацію даних, структурованих для розпаралелювання, таких як дані однієї або більшої кількості мозаїк або дані одного або більшої кількості вкладених потоків WPP, на малі транші шляхом наближення перемежовування з малою затримкою.

Тому, кодер може надавати дані, які відповідають конкретному набору LCUs або принаймні байт-синхронізованій частині вкладеного потоку або його мозаїці або частинам у формі траншу, до декодера за допомогою каналу передачі даних від кодера до декодера.

Оскільки транші менші за весь вкладений потік WPP або мозаїку і/або можуть адаптуватися до реального максимального передавального блока (MTU) передавального каналу так, що транші багатьох вкладених потоків WPP або мозаїк можуть розташовуватися в передавальному блоці між кодером і декодером перед завершенням усього блока доступу, декодування на декодувальній стороні може починатися значно раніше, ніж при використанні послідовної передачі усіх вкладених потоків WPP або мозаїк Блока Доступу.

Це очевидно приводить до швидшої передачі траншів і ранішого початку процесу паралельного декодування в декодері. Наближення може також застосовуватися на границях кадру, у випадку, якщо вирізка(и) або ентропійна вирізка(и) наступного кадру можуть вже декодуватися, наприклад хвильовим чином, на основі знання, що необхідна інформація для декодування ентропійної вирізки наступного кадру доступна внаслідок доступності міжкадрових кореляцій. Такі вже здатні до декодування дані кадру, які слідують в порядку декодування, можуть одержуватися з максимальної дозволеної/сигналізованої довжини вектора руху або додаткової інформації в потоці, який вказує залежності частин даних від попереднього(іх) кадру(ів) або фіксованої корелюючої схеми, яка вказує положення, сигналізоване у фіксованій послідовності, такій як множина значень параметрів.

Картинка може кодуватися з однією ентропійною вирізкою на найбільшу кодувальну чарунку (LCU) або з використанням вкладеного потоку WPP або навіть як один вкладений потік WPP на рядок, який може додатково міститися в окремій ентропійній вирізці. Такі структури даних

необхідні для використання технології хвильової обробки даних на декодувальній стороні. Або для надання можливості паралельної обробки можуть використовуватися мозаїки.

Під час процесу кодування потік бітів кожної вирізки, який містить дані потоків WPP або мозаїки, можуть ділитися на транші змінного розміру для узгодження з максимальним розміром передавального блока між кодером і декодером. Потім одержані транші перемежовуються і можуть подаватися для передачі і вставляння в пакети максимального розміру.

Для надання можливості обробки на декодувальній стороні, до або після прийому кожного траншу може вставлятися маркерний код. Належний маркерний код для HEVC може бути "0 × 00 00 02", який повинен навіть підходити для запобігання емуляції початкового коду. Після прийому пакета, який містить багато траншів, приймач або декодер може проводити синтаксичний аналіз реального вміщеного потоку бітів під час процесу запобігання емуляції початкового коду для усунення додаткового етапу синтаксичного аналізу. Тут може бути, наприклад, два режими для ідентифікації траншу. Тут може завжди бути циклічне розташування траншів, починаючи з траншу tranche_id (ідентифікатор траншу), який дорівнює 1, до траншу tranche_id, який дорівнює n. Це може захищати сигнальні дані для другого головного способу. Альтернативним способом може бути спеціальний заголовок, який відповідає маркеру, який вказує tranche_id, наприклад як 8-бітову величину.

Усунення перемежовування даних перемежованого траншу може застосовуватися на основі знання кількості траншів на пакет, який може бути пакетом блока NAL. Тому, тут додатково може бути перетворення вкладених потоків WPP або мозаїк на транші. Це перетворення може неявно одержуватися з ряду мозаїк/ряду вкладених потоків WPP або може сигналізуватися безпосередньо в SPS. Перетворення важливе для процесу усунення перемежовування так, що дані певних вкладених потоків WPP або мозаїки можуть ідентифікуватися і служити для хвильової або паралельної обробки даних послідовністю команд декодера, який декодує розглядуваний вкладений потік WPP або мозаїку.

Для інформування декодера про використання схеми перемежовування для інкапсуляції з малою затримкою, тут може бути low_delay_flag в заголовку блока NAL.

Іншим способом може бути перемежовування і усунення перемежовування на передавальному шарі, тобто зовні процес декодування може відбуватися в RTP (обробка даних в реальному часі) [8] [9] 0 або в шарі транспортного потоку MPEG-2 [7]:

Тому, заголовок може вставлятися перед пакетом, вказуючи присутність траншу за допомогою прапорця, який вказує інформацію про розмір в байтах на даний транш. Оскільки транспортний шар видаляється з процесу декодування, може бути відсутня потреба в інтегруванні маркерного коду, оскільки додаткову інформацію транспортного шару потрібно видаляти в будь-якому випадку перед подачею таких даних до декодера. Потім транспортний шар також реорганізує дані для подачі потоку бітів до декодера.

Заголовок змінної довжини може використовуватися на екстрамультиплексуючому шарі. Цей мультиплексуючий шар може бути також частиною кодека і може вводитися перед реальним доступом до Даних Необробленої Послідовності Байтів (RBSP) в декодері. Одну схему заголовка можна знайти на Фіг. 3. Проте тут може бути також заголовок безпосередньо перед кожним траншем, який вказує довжину, а також його індикатор. Все ще існує потреба у перетворенні індикатора на структури потоку бітів, як вже зазначено вище.

Розмір траншу може також бути сталим, наприклад x байтів на транш. Це приводить до простої схеми мультиплексування, такої як та, що зображена на Фіг. 4.

Сталий розмір сегментів може створити проблему в кінці потоку бітів внаслідок їх змінної довжини.

Можливі два головні рішення. Першим є генерування циклічних x-байтових сегментів (зазвичай представлення бітового потоку вирізки є байт-синхронізованим) і контроль використання байтів кожним декодером, тобто, декодер виявляє завершення ентропійної вирізки або містить маркерний код.

Другим способом є довжини сигнального траншу, якщо транші мають змінну довжину в заголовку, як зображено на фігурі.

Розмір сегмента і режим перемежовування можуть сигналізуватися або в одному SEI-Повідомленні або в SPS.

Схема передачі даних зображена на Фіг. 4.

Іншим цікавим способом є використання кодів завершення або маркерних кодів в кінці набору траншів в пакеті, такому як NAL або пакет вирізки. У цьому випадку, можливі сегменти змінної довжини, таким чином, вимагається повний синтаксичний аналіз потоку бітів. Для обмеження тут доступу до пам'яті, цей додатковий синтаксичний аналіз для мультиплексування може поєднуватися з синтаксичним аналізом запобігання емуляції початкового коду, необхідним

як перший етап перед доступом до даних RBSP, які містяться в блоці NAL. Така маркерна схема зображена на Фіг. 5.

Тут ідея полягає в розбитті з перемежовуванням структури вищого рівня, такої як реальна вирізка, ентропійна вирізка або подібне, на її вміщувану структуру даних нижчого рівня, таку як вкладені потоки WPP або мозаїки, з одночасним перемежовуванням даних в транші. Ці транші, кожен з яких належить структурі нижчого рівня, наприклад спеціальному вкладеному потоку WPP або мозаїці, перемежовуються в пакеті з малою затримкою, який може бути спеціальним блоком NAL, блоком NAL з додатковою сигналізацією за допомогою прапорця перемежовування з малою затримкою або навіть заголовка вирізки або заголовка вирізки з малою вагою, який вказує наближення перемежовування з малою затримкою за допомогою прапорця або типу вирізки, як зображено для "NAL unit #1" на фігурі, таким чином декодер інформують для застосування функції переупорядковування для декодера з "єдиним" потоком команд, який використовує послідовну обробку траншів в первинному порядку/порядку з усуненим перемежовуванням. Для розбиття даних реальної вирізки як перемежованих траншів на багато пакетів для досягання малої затримки, транспортний шар може фрагментувати блок NAL, який містить перемежовані з малою затримкою дані, на мережеві пакети максимального розміру. Фрагментація даних реальної вирізки на багато блоків NAL може також безпосередньо використовуватися кодувальним шаром, таким чином, існує потреба у сигналізації такого типу блока NAL, який містить подовження вирізки, як зображено на Фіг. 5 для "NAL unit #2". Для виявлення завершення перемежованих даних в багатьох пакетах, таких як блоки NAL, може існувати потреба в спеціальному коді завершення, як також зображено для "NAL unit #2" на фігурі, або в прапорці, який вказує завершення у вирізці або заголовку NAL.

У випадку втрати пакетів NAL також існує потреба у виявленні втрат. Це може використовуватися додатковою інформацією в заголовку, наприклад заголовку вирізки з малою вагою, такому як перші мегабіти вміщених траншів, або тільки спеціального траншу #1. Маючи інформацію, таку як зміщення для вкладених потоків WPP або реальний розмір траншу, дехто може також використовувати ці значення розмірів (величини зміщення для спеціального вкладеного потоку WPP або мозаїки) для виконання контролю коректності після прийому блока NAL з кодом завершення і попередніх блоків NAL.

Тобто, як описано, транші можуть пакетуватися в пакети 300 у такий спосіб, що кожен пакет 300 містить один транш T# кожного вкладеного потоку WPP або мозаїки картинки, або підмножину вкладених потоків WPP або мозаїки картинки (оскільки, наприклад, певний вкладений потік WPP або мозаїка вже повністю переданий(а) попередніми пакетами), розташованих в порядку #, визначеному серед вкладених потоків WPP або мозаїк, при цьому кожен пакет містить заголовок 302, який містить інформацію, яка вказує положення і/або довжини траншів T#, запакованих у відповідний пакет 300, або маркери 304, які відділяють транші T# у відповідному пакеті 300 один від іншого, при цьому декодер може конфігуруватися для використання інформації, яка міститься в заголовках 302 або маркерах 304, для доступу до траншів в пакетах при прийомі корисної інформації необробленої послідовності байтів. Пакети 300a, які містять перші, у відповідності з порядком, визначеним серед вкладених потоків WPP або мозаїк, транші вкладених потоків WPP або мозаїк картинки, можуть містити індикатор 306 малої затримки, а пакети 300b, які містять другі або наступні, у відповідності з порядком, визначеним серед вкладених потоків WPP або мозаїк, транші T# вкладених потоків WPP або мозаїк картинки, можуть містити індикатор 308 подовження. Пакети 300 можуть бути блоками NAL або вирізками.

Далі надається приклад для сигналізації синтаксису і семантики для перемежовування з малою затримкою у траншах.

Тим не менше, розбиття даних траншу, таких як дані вкладеного потоку WPP або мозаїки, може також використовуватися на рівні вирізки або нижче, як вказано вище.

Тепер, зображується наближення, яке може поєднуватися з синтаксичним аналізом для запобігання емуляції початкового коду для зменшення додаткових етапів обробки. Тому, перемежовування застосовується на рівні RBSP кодека HEVC.

Транш можна розглядати як розбиття даних RBSP на частини, які перемежовуються в корисній інформації блока NAL для доступу до даних з малою затримкою. Завершення траншу може вказуватися кодом 0 × 000002 і за ним може слідувати 8-бітовий ідентифікатор траншу tranche_id. Транші можуть перемежовуватися циклічним чином так, що за кодом кінця траншу не слідує tranche_id, який одержується неявно. Дані RBSP в єдиному транші відповідають або даним мозаїки, даним вкладеного потоку, даним вирізки або даним будь-якої ентропійної вирізки.

В синтаксисі блока NAL можуть дозволятися два режими для перемежовування з малою затримкою, як вказано "low delay encapsulation_flag", тобто циклічне розташування траншів, а також індикація траншу додатковим ідентифікатором "tranche_id", який відповідає маркерному коду за допомогою прапорця, такого як "low delay cyclic_flag" в заголовку блока NAL. Ці два прапорці можуть бути також присутніми в Множинах Параметрів Послідовності (SPS) або навіть APS. Для циклічних розташувань траншів все ще може бути необхідним знання кількості траншів під час синтаксичного аналізу, такого як той, що наведений в SPS як "num_low_delay_tranches".

В блоці NAL перемежовані "LD_rbsp_byte's зчитуються елементом для виконання синтаксичного аналізу в реальному послідовному порядку RBSP в останньому для циклу синтаксисі NAL:

```
for (i= 0, i++, i < num_low_delay_tranches){
for (j=0, j++, j < NumBytesInRBSP[i]){
rbsp_byte [NumBytesInRBSP++] = LD_rbsp_byte[j][i]
}
```

Тут також може бути явна сигналізація в SPS або APS для фіксованого розміру циклічно розташованих траншів, як вказано в "low_delay_tranche_lenght_minus1". Останній не використовувався в прикладі синтаксису блока NAL, але є простим, якщо мати на увазі пакетування, як зображено на Фіг. 4. В синтаксисі блока NAL з Фіг. 6 пакетування, як зображено на Фіг. 5 і обговорено вище, було основним.

Для надання можливості цього перемежовування траншів в багатьох пакетах, таким як вирізки і/або блоки NAL, може потребуватися глобальний буфер, такий як масив LD_rbsp_byte для траншів, для повторного доступу до даних RBSP вже прийнятих блоків NAL.

Для забезпечення протистояння похибкам, після прийому коду завершення або якщо сума кількостей прийнятих байтів для траншу дорівнює розміру траншу, який може одержуватися з величин зміщення, як передбачено для вміщених даних траншу, наприклад з даних, які стосуються відповідного вкладеного потоку WPP або мозаїки, частиною якого є розглядуваний траншів, важлива вимога для вкладених потоків WPP, розташованих в перемежованих з малою затримкою траншах, полягає в тому, що до даних з траншу n можна одержати доступ тільки траншем n+1, які вже надані в транші n і вже збережені або доступні в декодері.

Синтаксис шару RBSP Вирізки. з Малою Затримкою для переупорядковування/усунення перемежовування на рівні вирізки може розроблятися наступним чином. Зокрема, синтаксис повинен у такому випадку мати майже ту ж поведінку що й в шарі блока NAL, але переупорядковування повинне визначатися на рівні вирізки. Фіг. 8 зображає синтаксис шару RBSP Вирізки з Малою Затримкою.

У випадку використання заголовка вирізки для пакетування перемежованих траншів, може вимагатися на рівні кодека, при прийомі нової вирізки, не скидати стан CABAC, оскільки ентропійне кодування траншів, наприклад вкладеного потоку WPP, не повинно перериватися. Нескидання CABAC вказується як "no_cabac_reset_flag" в заголовку вирізки. Зображений заголовок вирізки придатний для вирізок з малою затримкою, таким чином, також повинні бути присутніми ознаки ентропійної вирізки. Відповідний синтаксис заголовка вирізки зображений на Фіг. 9.

Передавальний шар дозволяє оптимізацію диспетчеризації даних, направлених до декодера(ів), на основі того факту, що, якщо кількість вкладених потоків/мозаїк/траншів (на транспортному шарі припускаємо абстрактний об'єкт, який може представлятися вкладеним потоком, мозаїкою, частиною вкладеного потоку або мозаїки, або частиною потоку бітів, яка має подібну функцію, тобто вона дозволяє паралельне декодування або поступове оновлення декодера) в кодувальному шарі можуть оброблятися незалежно одне від іншого, то єдиною можливістю є початок надсилання траншів паралельно декільком декодувальним блокам з мінімальною затримкою. Потік бітів складається з послідовності блоків NAL, які є найменшими об'єктами, якими можна індивідуально оперувати на транспортному шарі. Тому, наступні способи оперування на транспортному шарі базуються на вкладених потоках/мозаїках/траншах, які містяться в окремій вирізці або блоках NAL ентропійної вирізки.

Передавальний шар повинен також оптимізувати робочі характеристики декодера і протистояння похибкам на основі того факту, що, якщо кодувальний шар використовує поступове оновлення декодера, то одним варіантом є скидування несуттєвих частин потоку бітів, якщо попередні частини потоку бітів не були вірно прийняті, наприклад внаслідок помилок передачі даних, або не були прийняті взагалі, наприклад внаслідок перемикання між передавальними каналами.

Для надання можливості такої експлуатації/оптимізації, на транспортному шарі сигналізується різна інформація.

Загальна побічна інформація сигналізується з використанням дескрипторів:

- кількість вкладених потоків/мозаїк, де "1" означає, що існує тільки один потік/мозаїка, який містить увесь відеокадр;
- інформація, спільна для усіх вкладених потоків/мозаїк, наприклад, якщо усі вкладені потоки/мозаїки мають однаковий розмір або вимоги до буфера є однаковими;
- індивідуальна інформація про кожен вкладений потік/мозаїку, наприклад, якщо вкладені потоки/мозаїки мають різний розмір або їх вимоги до буфера є різними;
- кількість етапів поступового оновлення декодера, де "1" означає, що поступове оновлення декодера не використовується;
- прапорець, який вказує, чи дозволяють ці вкладені потоки/мозаїки паралельну обробку з малою затримкою.

Якщо кількість вкладених потоків/мозаїк >1, то синтаксичні елементи вставляються в потік перед кожним блоком даних, який містить певний вкладений потік/мозаїку. Ці синтаксичні елементи відповідають синтаксису блока NAL, але використовують унікальний тип блока NAL, який не використовується кодувальним шаром (наприклад, nal_unit_type=0 × 19 або nal_unit_type=0 × 1F), і далі називаються маркерами вкладеного потоку.

Ці синтаксичні елементи використовуються як маркери і містять інформацію про блок даних, який відповідає принаймні полю даних, яке ідентифікує вкладений потік/мозаїку.

Якщо кількість етапів поступового оновлення декодера >1, то ці синтаксичні елементи також містять прапорець, який вказує, чи інтракодований вкладений потік/мозаїка (дозволяє поступове оновлення декодера).

Відповідний синтаксис зображений на Фіг. 10. Можуть застосовуватися наступні обмеження:

forbidden_zero_bit повинен дорівнювати 0;

nal_ref_flag повинен дорівнювати 0;

nal_unit_type повинен дорівнювати 0 × 19;

substream_ID: лічильник величин, який, починаючи з 0 для першої вирізки, що належить картинці, збільшується на інкремент з кожною додатковою вирізкою або ентропійною вирізкою, яка належить ті й же картинці;

is_intra: якщо "1", то наступний блок NAL містить інтракодовану вирізку або інтракодовану ентропійну вирізку.

Спосіб інкапсуляції потоку відеоданих в транспортному мультиплексорі зображений на Фіг. 11, де кожна вирізка або ентропійна вирізка передається окремо в цілій кількості пакетів транспортного потоку. Якщо розмір корисної інформації точно не відповідає доступним байтам в TS пакетах фіксованого розміру, то останній TS пакет містить поле адаптації.

Слід відзначити, що подібна поведінка Елементарного Потоку Транспортного Потоку MPEG-2 може також забезпечуватися RTP Сесією або потоком Транспортного Протоколу в Реальному часі (RTP), як зображено на Фіг. 19. В RTP 0 потік RTP (ідентифікований типом середовища і типом корисної інформації, як вказано в SDP 0) може міститися у своїй власній RTP сесії, де RTP Сесія ідентифікується (IP) мережевою адресою, (UDP) портом, а також ідентифікатором джерела (SSRC). Інформаційна сесія, як вказано в SDP, може містити багато RTP сесій, кожна з яких містить різний тип інформації. Але також можна передавати той же потік інформації (наприклад, відеоінформацію) в різних RTP потоках, де RTP потоки можуть міститися в однаковій RTP сесії (аналогічно до випадку 1, наведеного нижче) або може міститися у своїх власних RTP сесіях (аналогічно нижченаведеному випадку 2.). Фіг. 19 зображає випадок 2.

Формати корисної інформації RTP 0 [13] мають номер порядку декодування (DON), який дозволяє відновлювати порядок декодування блоків NAL в приймачі у випадку, коли вони навмисне передаються не в порядку декодування для протистояння похибкам, як описано в документах 00. Тому, додаткові маркери MKR не потрібні. У випадку передачі траншів вкладених потоків WPP або мозаїк в порядку, коли вони стають доступними з процесів кодування, DON може також використовуватися для відновлення порядку декодування траншів перед надсиланням їх до єдиного декодера. Але у цьому випадку, в декодер повинна вводитися додаткова затримка внаслідок окремого процесу усунення перемежовування перед процесом декодування. Описана тут система може надавати кодовані транші безпосередньо до процесів декодування різних WPP вкладених потоків або мозаїк з одночасним надходженням даних до приймача. Ідентифікація траншів, пов'язана з вкладеним потоком WPP або мозаїкою, може отримуватися згідно з адресою вирізки в заголовку сегмента вирізки і порядком передачі пакетів, як вказано номером RTP послідовності в RTP заголовку. В цьому сценарії DON використовується тільки для зворотної сумісності, тобто, для декодерів, які не забезпечують

покращеної сумісності декодування траншів вкладених потоків WPP або мозаїк, які не відповідають порядку декодування при надходженні. Надсилання даних траншу з порушенням порядку декодування застосовується тільки стосовно вкладеного потоку WPP і рівня мозаїк, тобто, у переданих даних, при цьому транші єдиного вкладеного потоку WPP або мозаїка передаються в порядку декодування, де дані різних вкладених потоків WPP або мозаїки перемежовуються.

Існує два можливі варіанти:

1. Усі вирізки і ентропійні вирізки містяться в однаковому елементарному потоці, тобто однаковий PID(ідентифікатор пакета) присвоюється усім TS пакетам такого відеопотоку; в подальшому тексті цей спосіб називається інкапсуляцією єдиного елементарного потоку (ES).

2. Різні PIDs присвоюються вирізкам і ентропійним вирізкам однакового потоку відеобітів; в подальшому тексті цей спосіб називається інкапсуляцією багатьох елементарних потоків (ES).

Фіг. 11 дійсна для обох варіантів, якщо перший варіант стосується спеціального випадку більш загальної структури встановленням однакового PID для усіх ES.

Ефективніший спосіб інкапсуляції в єдиному ES зображений на Фіг. 12. Тут, вимагається щонайбільше одне поле адаптації на структуру.

Ефективніший спосіб інкапсуляції в багатьох ES зображений на Фіг. 13. Тут, адаптаційні поля уникаються; замість цього, інша вирізка, наприклад, сумісна мозаїка наступної картинки, починається безпосередньо у тому ж пакеті транспортного потоку.

Можлива структура передавального демультиплексора для інкапсуляції з одним єдиним елементарним потоком (ES), який націлює декодер з багатьма послідовностями команд, зображена на Фіг. 14. Ентропійна вирізка на фігурі може містити дані спеціального вкладеного потоку WPP або мозаїки.

Передавальний Буфер (TB) збирає дані, які належать до транспортного пакета і направляє їх до Мультиплексного Буфера (MB). На виході MB оцінюються заголовки блоків NAL і скидаються маркери вкладених потоків, тоді як дані, які переносяться в маркері вкладеного потоку, зберігаються. Дані кожної вирізки або ентропійної вирізки зберігаються в окремому Буфері Вирізок (SB), звідки вони беруться декодером з багатьма послідовностями команд, як тільки він стає доступним.

Можлива структура транспортного демультиплексора для інкапсуляції з багатьма елементарними потоками, які націлюють декодер з багатьма послідовностями команд, зображена на Фіг. 15.

Вищенаведені концепції знову описуються нижче іншими словами. Тому, нижченаведений опис може поєднуватися з додатковими деталями окремо наведеного вище опису.

Фіг. 16 зображає загальну структуру кодера згідно з варіантом виконання представленого винаходу. Кодер 10 може втілюватися здатним працювати в багатопотоковому режимі або ні, тобто просто в однопотоковому режимі. Тобто, кодер 10 може, наприклад, втілюватися з використанням багатоядерного центрального процесора (CPU). Іншими словами, кодер 10 може підтримувати паралельну обробку, але не повинен це робити. Концепція кодування представленого винаходу дозволяє кодерам з паралельною обробкою ефективно здійснювати паралельну обробку, однак, без порушення ефективності стискання. Що стосується здатності до паралельної обробки, то подібні твердження дійсні для декодера, який описується далі стосовно Фіг. 17.

Кодер 10 є відеокодером, але загалом кодер 10 може бути також кодером картинки. Картинка 12 відеозображення 14 показана як та, що надходить до кодера 10 на вхід 16.

Кодер 10 є гібридним кодером, тобто картинка 12 прогнозується в предикторі 18, а залишок прогнозування 20, як він одержаний елементом 22 для визначення залишку прогнозування, таким як блок віднімання, піддається перетворенню, такому як спектральний розклад, такий як Дискретне Косинусне Перетворення (DCT), і дискретизації в модулі 24 перетворення/дискретизації. Дискретизований таким чином одержаний залишок 26 піддається ентропійному кодуванню в ентропійному кодері 28, зокрема адаптивному до контексту бінарному арифметичному кодуванню. Здатний до відтворення залишок, як він доступний для декодера, тобто деквантизований і повторно перетворений залишковий сигнал 30, відновлюється модулем 31 повторного перетворення і повторної дискретизації і поєднуватися з прогнозувальним сигналом 32 предиктора 18 об'єднувальним елементом 33, таким чином забезпечуючи відновлення 34 картинки 12. Однак, кодер 10 працює на блочній основі. Відповідно, відновлений сигнал 34 страждає від розривів на границях блоків і, відповідно, фільтр 36 може застосовуватися до відновленого сигналу 34 для надання еталонної картинки 38, на основі якої предиктор 18 прогнозує послідовно кодовані картинки. Як зображено пунктирними лініями на Фіг. 16, предиктор 18 може, однак, також використовувати відновлений

сигнал 34 безпосередньо без фільтра 36 або проміжного варіанта. У випадку кодування картинки, фільтр 36 може усуватися.

Предиктор 18 може вибирати різні режими прогнозування для прогнозування певних блоків картинки 12. Тут може бути тимчасовий режим прогнозування, згідно з яким блок прогнозується на основі попередньо кодованих картинок, режим просторового прогнозування, згідно з яким блок прогнозується на основі попередньо кодованих блоків однієї і тієї ж картинки, режими міжшарового прогнозування, згідно з якими блок картинки, який показує сцену на вищому шарі, як, наприклад, при вищій просторовій роздільній здатності або з додаткової точки огляду, прогнозується на основі відповідної картинки, яка показує цю сцену на нижчому шарі, як, наприклад, з меншою просторовою роздільною здатністю або з іншої точки огляду.

Певний синтаксис використовується для збору дискретизованих залишкових даних 26, тобто рівнів коефіцієнтів перетворення та інших залишкових даних, а також дані режиму кодування, які містять, наприклад, режими прогнозування і параметри прогнозування для окремих блоків картинки 12, як визначено предиктором 18, і ці синтаксичні елементи піддаються ентропійному кодуванню ентропійним кодером 28. Таким чином одержаний потік даних як вихід ентропійного кодера 28 називається корисною інформацією 40 необробленої послідовності байтів.

Елементи кодера 10 з Фіг. 16 з'єднані між собою, як зображено на Фіг. 16.

Фіг. 17 зображає декодер, який відповідає кодеру з Фіг. 16, тобто здатний декодувати корисну інформацію необробленої послідовності байтів. Декодер з Фіг. 17 головним чином вказується позицією 50 і містить ентропійний декодер 52, модуль 54 повторного перетворення/деквантизації, об'єднувальний елемент 56, фільтр 58 і предиктор 60. Ентропійний декодер 42 приймає корисну інформацію 40 необробленої послідовності байтів і виконує ентропійне декодування з використанням адаптивного до контексту бінарного арифметичного декодування для відновлення залишкового сигналу 62 і кодувальних параметрів 64. Модуль 54 повторного перетворення/деквантизації деквантизує і повторно перетворює залишкові дані 62 і направляє таким чином одержаний залишковий сигнал до об'єднувального елемента 56. Об'єднувальний елемент 56 також приймає прогнозувальний сигнал 66 від предиктора 60, який, у свою чергу, формує прогнозувальний сигнал 66 з використанням кодувальних параметрів 64 на основі відновленого сигналу 68, визначеного об'єднувальним елементом 56 шляхом об'єднання прогнозувального сигналу 66 і залишкового сигналу 65. Як вже пояснювалося вище стосовно Фіг. 16, предиктор 60 може використовувати відфільтрований варіант відновленого сигналу 68 або деякий його проміжний варіант, альтернативно або додатково. Картинка, яка, врешті решт, відтворюється і видається на вихід 70 декодера 50, може подібним чином визначатися на нефільтрованому варіанті комбінованого сигналу 68 або деякому його відфільтрованому варіанті.

У відповідності з концепцією мозаїки картинку 12 підрозбивають на мозаїки і принаймні прогнози блоків в цих мозаїках обмежуються використанням просто даними як основи для просторового прогнозування, які відносяться до однієї і тієї ж мозаїки. Завдяки цьому заходу, принаймні прогнозування може виконуватися паралельно індивідуально для кожної мозаїки. Тільки для ілюстрації, Фіг. 16 зображає картинку 12, підрозбиту на дев'ять мозаїк. Підрозбиття кожної мозаїки на дев'ять блоків, як зображено на Фіг. 16, також просто служить як приклад. Окрім того, для повноти, відзначається, що спосіб окремого кодування мозаїк може не обмежуватися просторовим прогнозуванням (інтрапрогнозуванням). Скоріше, будь-яке прогнозування кодувальних параметрів відповідної мозаїки крізь її границі і будь-яка залежність вибору контексту в ентропійному кодуванні відповідної мозаїки крізь її відповідні границі може також заборонятися для обмеження тільки залежністю від даних однієї і тієї ж мозаїки. Таким чином, декодер здатен виконувати паралельно тільки що згадані операції, зокрема в блоках мозаїк.

Для передачі по деякому каналу передачі даних, синтаксичні елементи повинні ентропійно кодуватися ентропійним кодером 28 по вирізках. Для цього, ентропійний кодер 28 проходить блоки мозаїк, проходячи блоки спершу першої мозаїки, потім блоки наступної мозаїки в порядку розташування мозаїк і так далі. Порядок растрової розгортки може, наприклад, використовуватися для проходження блоків в мозаїках і, відповідно, мозаїк. Вирізки потім пакуються в блоки NAL, які є найменшими блоками для передачі. Перед ентропійним кодуванням вирізки ентропійний кодер 28 ініціалізує свої ймовірності CABAC, тобто ймовірності, використовувані для арифметичного кодування синтаксичного елемента такої вирізки. Ентропійний декодер 52 робить те ж саме, тобто ініціалізує свої ймовірності на початках вирізок. Однак, кожна ініціалізація негативно впливає на ефективність ентропійного кодування, оскільки ймовірності безперервно адаптуються до реальної статистики ймовірності символів різних контекстів і, відповідно, скидування ймовірностей CABAC представляє відхилення від

адаптованого стану. Як відомо фахівцю у цій галузі, ентропійне кодування приводить до оптимального стискання тільки, якщо ймовірності відповідають реальній статистиці ймовірності символів.

Відповідно, декодер у відповідності з варіантом виконання представленого винаходу працює, як зображено на Фіг. 18. Декодер приймає на етапі 80 корисну інформацію необробленої послідовності байтів, яка описує картинку 12 в мозаїках 82, в траншах мозаїк. На Фіг. 18 перша мозаїка 82 в порядку 84 розташування мозаїк ілюстративно зображена розрізаною або поділеною на два транші 86a і 86b, кожен з яких ілюстративно покриває підпослідовність послідовності блоків в такій мозаїці. Потім, на етапі 82 транші 86a і 86b ентропійно кодуються. Однак, в ентропійному декодуванні траншів 86a і 86b адаптація ймовірності CABAC продовжується крізь границі траншів. Тобто, під час декодування траншу 86a, ймовірності CABAC безперервно адаптуються до реальної символьної статистики і стан в кінці ентропійного декодування траншу 86a адаптується в початковому ентропійному декодуванні траншу 86b. На етапі 90 таким чином ентропійно декодована корисна інформація необробленої послідовності байтів декодується для одержання картинки 12.

Внаслідок продовження адаптації ймовірності CABAC через границі 92 траншів, розташованих всередині мозаїк 82, ці границі траншів не впливають негативно на ефективність ентропійного кодування після підрозбиття картинки 12 на мозаїки 82. З іншого боку, все ще можлива паралельна обробка. Після цього, можна індивідуально передавати транші і, коли транші менші за мозаїки 82, можна починати етап 90 декодування кожної мозаїки, як тільки прийнятий і ентропійно декодований перший транш відповідної мозаїки.

Опис Фіг. 16-18 головним чином стосується використання мозаїк. Як описано вище, мозаїки одержуються з просторового розбиття картинки. Подібно до мозаїк, вирізки також просторово підрозбивають картинку. Вирізки, відповідно, також є засобами для надання можливості паралельного кодування/декодуваня. Подібно до мозаїк, прогнозування і так далі заборонено таким чином, що вирізки здатні декодуватися окремо. Відповідно, опис Фіг. 16-18 також дійсний для розбиття вирізок на транші.

Те ж саме застосовується при використанні вкладених потоків WPP. Вкладені потоки WPP також представляють просторове розбиття картинки 12, зокрема на вкладені потоки WPP. На противагу до мозаїк і вирізок, вкладені потоки WPP не накладають обмеження на прогнози і на вибори вкладених потоків WPP. Вкладені потоки WPP проходять вздовж рядів блоків, таких як ряди LCU, як зображено на Фіг. 4, і для надання можливості паралельної обробки просто робиться один компроміс відносно ентропійного кодування CABAC в порядку, визначеному серед вкладених потоків WPP (дивіться Фіг. 4) 92 і для кожного з вкладених потоків WPP 92, за виключенням першого вкладеного потоку WPP, ймовірності CABAC не повністю скидаються, а адаптуються, або встановлюється рівними ймовірностям CABAC, які одержуються після ентропійного декодування безпосередньо попереднього вкладеного потоку WPP, його другого LCU 94, з порядком LCU, який починається для кожного вкладеного потоку WPP на однаковій стороні картинки 12, як, наприклад, на лівій стороні, як зображено на Фіг. 4. Відповідно, дотримуючись деякої затримки в кодуванні між послідовністю вкладених потоків WPP, ці вкладені потоки WPP 92 здатні декодуватися паралельно так, що частини, на яких паралельно декодується картинка 12, тобто, одночасно, формують фронт хвилі 96, який похило рухається по картинці зліва направо.

Тобто, при перенесенні опису Фіг. 16-18 на вкладені потоки WPP, будь-який вкладений потік WPP 92 (Фіг. 4) може також підрозбиватися на транші 98a і 98b без переривання адаптації ймовірності CABAC на границі 100 між цими траншами 98a і 98b всередині відповідного вкладеного потоку WPP 92, таким чином, уникаючи проблем стосовно ефективності ентропійного кодування внаслідок індивідуальної здатності до передавання обох траншів 98a і 98b, але зберігаючи здатність використовувати паралельну хвильову обробку і здатність починати цю паралельну хвильову обробку раніше, оскільки транші менші за усі вкладені потоки WPP 92.

Як описано вище стосовно Фіг. 1-15, існує декілька можливостей передачі траншів, пакетованих в блоки NAL. Посилання робиться на Фіг. 3, де мозаїки або вкладені потоки або вирізки таких траншів, або вкладені потоки розділені на транші в арифметично кодованій області із заголовком, який передує n-му траншу кожного вкладеного потоку або мозаїці, і представляє інформацію, яка дозволяє локалізацію границь траншу. Інший варіант виконання представлений на Фіг. 9. Тут, підрозбиття мозаїк або вкладених потоків WPP на транші виконувалося незначною зміною структури вирізки: вирізки, які починаються на мозаїці або границі вкладеного потоку WPP, тобто, які починаються на початку мозаїки або вкладеного потоку WPP, не мають no_cabac_reset_flag, встановленого в нуль, таким чином виконуючи

звичайну ініціалізацію/скидування ймовірності CABAC. Однак, вирізки, які містять транші, які починаються всередині мозаїки або вкладеного потоку WPP, мають no_cabac_reset_flag, встановлений в одиницю, таким чином виконуючи вищеописане продовження адаптації ймовірності CABAC.

Настільки, наскільки це стосується усунення перемежовування, яке має місце на етапі 80 прийому, для кожного траншу визначається, якому вкладеному потоку WPP або мозаїці належить відповідний транш. Вище були описані різні можливості, такі як, наприклад, карусельне циклювання крізь ряд вкладених потоків WPP або мозаїки поточної картинки. Альтернативно, у випадку використання заголовків вирізок для передачі траншів, заголовки вирізок можуть містити індикатор, який дозволяє локалізацію початку відповідної вирізки в поточній картинці 12.

З цього приводу, відзначається, що розбиття вирізок, вкладених потоків WPP або мозаїк на транші виконується в порядку декодування, визначеному в кожній вирізці, вкладеному потоці WPP або мозаїці: тобто, в кожній вирізці, вкладеному потоці WPP або мозаїці частина картинки, просторово покрита відповідною вирізкою, вкладеним потоком WPP або мозаїкою, кодується в або декодується з відповідної вирізки, вкладеного потоку WPP або мозаїки в такому порядку декодування і кожен транш відповідної вирізки, вкладеного потоку WPP або мозаїки покриває суцільну частину відповідної вирізки, вкладений потік WPP або мозаїку в такому порядку декодування. У цей спосіб, порядок визначається серед траншів, які належать одній і тій же вирізці, вкладеному потоку WPP або мозаїці, зокрема порядок кодування/декодування і кожен транш має ранг в такому порядку. Оскільки підрозбиття картинки на вкладені потоки WPP або мозаїки сигналізується декодеру, то декодер знає про підрозбиття. Відповідно, для зв'язування кожного траншу з відповідним вкладеним потоком WPP або мозаїкою, наприклад, повинно бути достатнім, щоб кожен транш мав початкову адресу, яка ідентифікує початкове положення, з якого відповідний транш неперервно покриває картинку з використанням порядку кодування/декодування мозаїки/вкладених потоків WPP, частиною яких є відповідний транш. Навіть порядок серед траншів, які належать певній мозаїці або вкладеному потоку WPP, наприклад, може відновлюватися в транспортному демультиплексорі або декодером з використанням початкових положень. Однак, для повторного упорядковування, може також використовуватися інформація заголовків транспортних пакетів нижніх OSI шарів, як описано вище стосовно RTP передачі, така як номер порядку декодування, тобто DON'S. Транспортний демультиплексор тільки що згаданого типу може конфігуруватися подібно до вищеописаного транспортного демультиплексора для зберігання даних траншів однакового вкладеного потоку WPP або мозаїки в одному буфері вирізок і даних траншів різних вкладених потоків WPP або мозаїк в різних буферах вирізок. Як згадано вище, структура вирізки, тобто заголовки вирізок, можуть використовуватися для подачі траншів.

Далі, посилання робиться на варіанти виконання з Фіг. 11-15 для повторного їх опису іншими словами. Як зображено на цих фігурах, вирізки Si пакетуються в блоки NAL, при цьому кожен блок 110 NAL (дивіться Фіг. 11) містить заголовок 112. Слід відзначити, що вирізки Si можуть бути нормальними вирізками або вирізками, які містять транші, згідно з Фіг. 9. Відповідно, ці вирізки містять тільки дані, які стосуються одного вкладеного потоку WPP або мозаїки поточної картинки, зокрема і-го вкладеного потоку WPP або, відповідно, мозаїки. За допомогою фрагментації блоки 110 NAL передаються пакетами 114 транспортного потоку (TS), зокрема їх корисною інформацією 116. Роблячи це, кожен блок 110 NAL і відповідна вирізка Si передується відповідним маркером MKR вкладеного потоку, який вказує і, тобто, вкладений потік WPP або мозаїку, який безпосередньо слідує за вирізкою безпосередньо наступного блока 110 NAL, який їй належить.

Блоки 110 NAL, які містять вирізки, які належать різним вкладеним потокам WPP або мозаїкам, можуть розподілятися по більше ніж одному елементарному потоку (ES) або по одному і тому ж елементарному потоку, як пояснюється на Фіг. 11-13. Як згадується вище, "елементарний потік" може також ідентифікувати окремий RTP потік у своїй власній RTP сесії.

Як пояснюється стосовно Фіг. 14 і 15, транспортний демультиплексор може містити мультиплексний буфер MB, буфер вирізок SB і транспортний буфер TB. Буфери вирізок SB беруться декодером MTD з багатьма послідовностями команд, який дозволяє паралельне декодування картинки у вкладених потоках WPP або мозаїках. Транспортний буфер TB сконфігурований для збирання даних, які належать до TS пакета наперед визначеного елементарного потоку відеобітів, і направлення даних до мультиплексного буфера MB. Транспортний демультиплексор потім конфігурується для оцінки заголовків блоків NAL послідовності блоків NAL, пакетованих в TS пакети на виході мультиплескного буфера MB, для скидування маркера MKR блоків NAL вкладеного потоку із збереженням даних маркера

вкладеного потоку, які переносяться в блоках NAL маркера вкладеного потоку, і зберігання даних вирізок вкладених потоків або мозаїк в блоках NAL, які відповідають блокам NAL маркера вкладеного потоку, поле даних яких ідентифікує однаковий вкладений потік WPP або мозаїку в одному, тобто одному і тому ж буфері вирізок SB, і дані вирізок вкладених потоків WPP або мозаїк в блоках NAL, які відповідають блокам NAL маркера вкладеного потоку, поле даних яких ідентифікує різні вкладені потоки WPP або мозаїки в різних буферах вирізок SB. Як зображено на Фіг. 15, транспортний демультиплексор може включати демультиплексор, названий TS demux на Фіг. 15, і сконфігурований для прийому потоку відеобітів і розбиття TS пакетів потоку відеобітів на різні елементарні потоки, тобто, для розподілення TS пакета потоку відеобітів по різних елементарних потоках. Демультиплексор виконує це розбиття або розподіл згідно з PIDs, які містяться в заголовках TS пакета, так, що кожен елементарний потік складається з TS пакетів з PID, відмінного від PIDs TS пакетів інших елементарних потоків.

Тобто, якщо вирізки відповідають траншам в сенсі варіанта виконання з Фіг. 9, то MTD, тобто декодер з багатьма послідовностями команд, здатен починати обробку більше, ніж одного вкладеного потоку WPP або мозаїки поточної картинки як тільки відповідний буфер вирізок SB відповідного вкладеного потоку WPP або мозаїки буде містити дані, таким чином зменшуючи затримку.

Хоча були описані деякі аспекти в контексті пристрою, зрозуміло, що ці аспекти також представляють опис відповідного способу, де блок або пристрій відповідають етапу способу або ознаці етапу способу. Аналогічно, аспекти, описані в контексті етапу способу, також представляють опис відповідного блока або деталі або ознаки відповідного пристрою. Деякі або усі етапи способу можуть виконуватися (або використовуючи) апаратний засіб, такий як, наприклад, мікропроцесор, програмований комп'ютер або електронна схема. В деяких варіантах виконання деякий один або більша кількість найважливіших етапів способу може виконуватися таким пристроєм.

Кодований потік бітів винаходу можна зберігати на середовищі для збереження цифрових даних або можна передавати по передавальному середовищі, такому як середовище безпровідної передачі даних або середовище проводової передачі даних, таке як Інтернет.

Ці вищезгадані досягнення, між тим, описують способи інкапсуляції з малою затримкою і передачі структурованих відеоданих, наданих новим стандартам кодування HEVC, таких як структуровані в мозаїках вкладені потоки хвильової обробки даних (WPP), вирізки або ентропійні вирізки. Між тим, були представлені технології, які дозволяють передачу даних з малою затримкою в розпаралеленому середовищі кодер-передавач-приймач-декодер за допомогою перемежованої передачі ентропійних вирізок/вирізок/мозаїк/вкладених потоків. Для вирішення проблем критичного параметра, описаних у вступній частині опису, і для мінімізації часу затримки в передачі і декодуванні даних, тобто наскрізної затримки, була представлена технологія для перемежованої схеми ентропійної вирізки для паралельної передачі і обробки даних.

В залежності від певних вимог до втілення, варіанти виконання винаходу можуть втілюватися в апаратних засобах або в програмних засобах. Втілення може здійснюватися з використанням середовища для зберігання цифрових даних, наприклад дискети, DVD, Blu-Ray, CD, ROM, PROM, EPROM, EEPROM або флеш-пам'яті, яке містить збережені в собі здатні до електронного читання керувальні сигнали, які взаємодіють (або здатні взаємодіяти) з програмованою комп'ютерною системою так, що виконується відповідний спосіб. Тому, середовище для зберігання цифрових даних може читатися комп'ютером.

Деякі варіанти виконання згідно з винаходом включають носій даних, який має здатні до електронного читання керувальні сигнали, які здатні взаємодіяти з програмованою комп'ютерною системою так, що виконується один із описаних тут способів.

Головним чином, варіанти виконання представленого винаходу можуть втілюватися як комп'ютерна програма з програмним кодом, який виконується для виконання одного із способів, коли програма виконується на комп'ютері. Програмний код може, наприклад, зберігатися на носії, який здатен читатися машиною.

Інші варіанти виконання включають комп'ютерну програму для виконання одного з описаних тут способів, збережену на носії, який здатен читатися машиною.

Іншими словами, варіант виконання способу винаходу є, тому, комп'ютерною програмою, яка має програмний код для виконання одного з описаних тут способів, коли комп'ютерна програма виконується на комп'ютері.

Тому, подальший варіант виконання способів винаходу є носієм даних (або середовищем для зберігання цифрових даних або здатним до читання комп'ютером середовищем), який містить записану в собі комп'ютерну програму для виконання одного з описаних тут способів.

Носій даних, середовище для зберігання цифрових даних або середовище запису є типово матеріальними і/або неперехідними.

Подальший варіант виконання способу винаходу є, тому, потік даних або послідовність сигналів, які представляють комп'ютерну програму для виконання одного з описаних тут способів. Потік даних або послідовність сигналів може, наприклад, конфігуруватися для передачі по з'єднанню для передачі даних, наприклад по Інтернету.

Подальший варіант виконання включає засоби обробки, наприклад комп'ютер або програмований логічний пристрій, сконфігуровані або адаптовані для виконання одного з описаних тут способів.

Подальший варіант виконання включає комп'ютер, який має встановлену на ньому комп'ютерну програму для виконання одного з описаних тут способів.

Подальший варіант виконання згідно з винаходом включає пристрій або систему, сконфігуровану для передачі до приймача (наприклад, електронно або оптично) комп'ютерної програми для виконання одного з описаних тут способів. Приймач може, наприклад, бути комп'ютером, мобільним пристроєм, запам'ятовуючим пристроєм або подібним. Пристрій або система може, наприклад, включати файловий сервер для передачі комп'ютерної програми до приймача.

В деяких варіантах виконання програмований логічний пристрій (наприклад, програмована логічна інтегральна схема) може використовуватися для виконання деяких або усіх функцій описаних тут способів. В деяких варіантах виконання програмована логічна інтегральна схема може об'єднуватися з мікропроцесором для виконання одного з описаних тут способів. Головним чином, способи переважно виконуються будь-яким апаратним засобом.

Вищеописані варіанти виконання є просто ілюстративними для принципів представленого винаходу. Зрозуміло, що модифікації і різні описані тут варіанти схем та деталі стануть очевидними для фахівців у цій галузі. Тому, наміром є обмежитися тільки об'ємом правового захисту доданої формули винаходу, а не спеціальними деталями, представленими у вигляді опису і пояснень варіантів виконання.

Джерела інформації:

[1] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard", IEEE Trans. Circuits Syst. Video Technol., vol. 13, N7, July 2003.

[2] JCTVC-E196, "Wavefront Parallel Processing", 5th JCT-VC Meeting, Geneva 2011.

[3] JCTVC-D070, "Lightweight slicing for entropy coding", 4th Meeting, Daegu, 2011.

[4] JCTVC-D073, "Periodic initialization for wavefront coding functionality", 4th Meeting, Daegu, 2011.

[5] HEVC WD5: Working Draft 5 °F High-Efficiency Video Coding JTCVC-G1103, 5th JCTVC Meeting, Geneva Meeting November 2011.

[6] JTCVC-D243, "Analysis of entropy slices approaches", 4th Meeting, Daegu, 2011

[7] ISO/IEC 13818-1/2011, MPEG-2 Transport Stream including AMDs 1-6.

[8] IETF Real-time transport protocol, RTP RFC 3550.

[9] IETF RTP Payload Format, IETF RFC 6184.

[10] JCTVC-F275, Wavefront and Cabac Flush: Different Degrees of Parallelism Without Transcoding, , Torino Meeting

[11] JCT-VC-F724, Wavefront Parallel Processing for HEVC Encoding and Decoding, Torino Meeting** at end of description

[12] IETF Session Description Protocol (SDP), RFC 4566

[13] IETF RTP Payload Format for High Efficiency Video Coding, draft-schierl-payload-h265

ФОРМУЛА ВИНАХОДУ

1. Декодер, сконфігурований для прийому корисної інформації необробленої послідовності байтів, яка описує картинку у вкладених потоках WPP з одним вкладеним потоком WPP на рядок LCU картинки, і кодованої з використанням CABAC, від кодера в траншах, на які розбиті вкладені потоки WPP, таким чином, вводячи в них межі траншів;

для ентропійного декодування траншів з продовженням адаптації ймовірності CABAC крізь межі траншів, введених у вкладені потоки WPP; і

для декодування корисної інформації необробленої послідовності байтів для одержання картинки, використовуючи повторне перетворення залишкових даних передбачення.

2. Декодер за п. 1, який **відрізняється** тим, що транші пакетують з використанням заголовків слайсів і, при прийомі траншів, декодер сконфігурований так, щоб реагувати після прийому

нового слайса на прапорець в заголовку нового слайса, slice-type нового слайса або тип блока тип блока NAL, який містить новий слайс, або для переривання адаптації ймовірності CABAC скиданням ймовірностей CABAC, або для продовження адаптації ймовірності CABAC.

3. Декодер за п. 1 або 2, який **відрізняється** тим, що виконаний з можливістю зворотного перемежовування при прийомі траншів шляхом ідентифікації для кожного траншу вкладеного потоку WPP, якому належить відповідний транш.

4. Декодер за п. 1 або 2, який **відрізняється** тим, що транші пакетуються в пакети таким чином, що кожен пакет містить один транш кожного вкладеного потоку WPP картинки або піднабір вкладених потоків WPP картинки, розташованих в порядку, визначеному серед вкладених потоків WPP, де кожен пакет містить заголовок, що містить розкриття позицій або довжин траншів, упакованих у відповідний пакет, або маркерів, що відокремлюють транші у відповідному пакеті один від одного, причому декодер налаштований для прийому корисної інформації необробленої послідовності байтів, використовують інформацію, що міститься в заголовках або маркерах, для доступу до траншів в пакетах.

5. Декодер за п. 4, який **відрізняється** тим, що пакети, що містять перші відповідно до порядку, визначеного серед вкладених потоків або мозаїк WPP, транші вкладених потоків WPP або мозаїк картинки, містять індикатор функції з низькою затримкою, та при цьому пакети, які містять другі або наступні відповідно до порядку, визначеного серед вкладених потоків або мозаїк WPP, транші вкладених потоків або мозаїк WPP картинки, містять індикатор продовження.

6. Декодер за п. 4, який **відрізняється** тим, що пакети є блоками NAL або слайсами.

7. Декодер, сконфігурований для:

прийому корисної інформації необробленої послідовності байтів, що описує картинки вкладених потоків WPP і кодованих з використанням CABAC з кодера в транші, на які розбиті вкладені потоки WPP, таким чином, вводячи в них межі траншів;

ентропійного декодування траншів з постійною адаптацією ймовірності CABAC через межі траншів, введених у вкладені потоки WPP, шляхом приймання, на початку ентропійного декодування одного траншу вкладеного потоку WPP, ймовірностей CABAC в кінці ентропійного декодування іншого траншу вкладеного потоку WPP; і

декодування корисної інформації необробленої послідовності байтів для отримання картинки, з використанням повторного перетворення залишкових даних передбачення.

8. Декодер за п. 1 або 7, який **відрізняється** тим, що корисна інформація необробленої послідовності байтів кодує сцену в шарах, що відповідають різним точкам огляду.

9. Декодер за п. 1 або 7, який **відрізняється** тим, що корисна інформація необробленої послідовності байтів картинки закодована в шарах.

10. Кодер, сконфігурований для:

формування корисної інформації необробленої послідовності байтів, шляхом кодування картинки для опису картинки у вкладених потоках WPP з одним вкладеним потоком WPP на рядок LCU картинки з ентропійним кодуванням необробленої послідовності байтів з використанням CABAC,

передавання необробленої послідовності байтів в траншах, на які розбиті вкладені потоки WPP, таким чином, вводячи в них межі траншів, і постійна адаптація ймовірності CABAC в ентропійному кодуванні через межі траншу, введені у вкладені потоки WPP, причому кодер використовує перетворення залишкових даних прогнозування для формування корисної інформації необробленої послідовності байтів.

11. Кодер за п. 10, який **відрізняється** тим, що кодер налаштований так, щоб формувати необроблену послідовність байтів таким чином, щоб транші відповідали максимальному розміру одиниці передачі.

12. Кодер за п. 10, який **відрізняється** тим, що корисна інформація необробленої послідовності байтів кодує сцену в шарах, що відповідають різним точкам огляду.

13. Кодер за п. 10, який **відрізняється** тим, що корисна інформація необробленої послідовності байтів картинки закодована в шарах.

14. Цифровий носій даних, який зберігає відеопотік бітів, що містить корисну інформацію необробленої послідовності байтів, що описує картинки у вкладених потоках WPP з одним вкладеним потоком WPP на рядок LCU картинки і закодована з використанням CABAC, причому потік відеобітів розбивається на транші вкладених потоків WPP, на які розбиті вкладені потоки WPP, таким чином, вводячи в них межі траншу, з постійною адаптацією ймовірності CABAC через межі траншу, введені у вкладені потоки WPP, причому кожен транш включає в себе явну вказівку свого рангу серед траншів, в яких вкладений потік WPP, якому належить відповідний

транш, є послідовно розкладений, причому корисна інформація необробленої послідовності байтів закодована в картинку з використанням перетворення залишкових даних.

15. Цифровий носій за п. 14, який **відрізняється** тим, що транші пакетуються в пакети таким чином, що кожен пакет містить один транш кожного вкладеного потоку WPP або фрагмента картинки або піднабір вкладених потоків WPP або мозаїк картинки, розташованих у порядку, визначеному серед вкладених потоків або мозаїк WPP, причому кожен пакет містить заголовок, що містить розкриття позицій або довжин траншів, упакованих в відповідний пакет, або маркери, що відокремлюють транші у відповідному пакеті один від одного.

16. Цифровий носій інформації за п. 14 або 15, який **відрізняється** тим, що пакети, що містять перші відповідно до порядку, визначеного серед вкладених потоків або мозаїк WPP, транші вкладених потоків WPP або мозаїк картинки, містять індикатор функції з низькою затримкою, та пакети, які містять другі або наступні відповідно до порядку, визначеного серед вкладених потоків або мозаїк WPP, транші вкладених потоків або мозаїк WPP картинки, містять індикатор продовження.

17. Цифровий носій інформації за п. 15, який **відрізняється** тим, що пакети становлять собою блоки NAL або слайси.

18. Цифровий носій за п. 14, який **відрізняється** тим, що корисна інформація необробленої послідовності байтів кодує сцену в шарах, що відповідають різним точкам огляду.

19. Цифровий носій за п. 14, який **відрізняється** тим, що корисна інформація необробленої послідовності байтів картинки закодована в шарах.

20. Спосіб декодування, який включає:

прийом корисної інформації необробленої послідовності байтів, що описує картинки у вкладених потоках WPP з одним вкладеним потоком WPP на рядок LCU картинки, і кодується з використанням CABAC від кодера в транші вкладених потоків WPP, на які розбиті вкладені потоки WPP, тим самим вводять в них межі траншів;

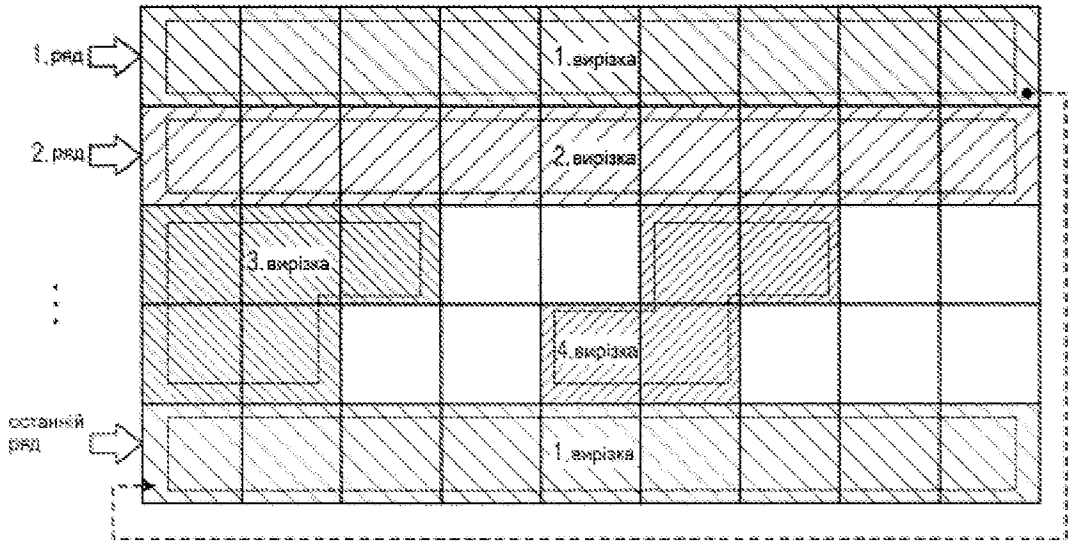
ентропійне декодування траншів з постійною адаптацією ймовірності CABAC через межі траншів, введені у вкладені потоки WPP; і

декодування корисної інформації необробленої послідовності байтів для отримання картинки з використанням повторного перетворення залишкових даних.

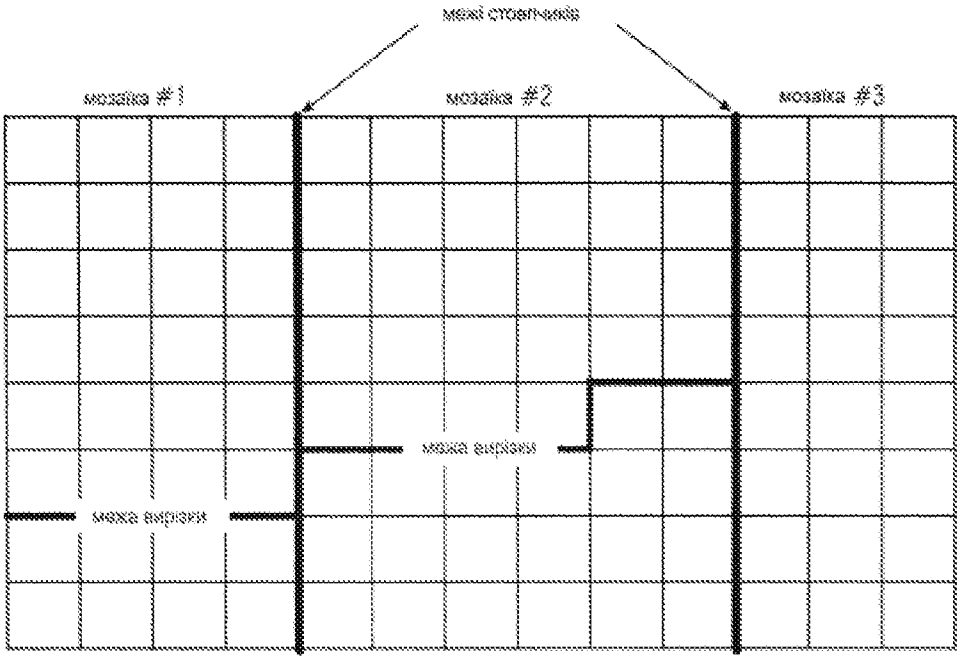
21. Спосіб кодування, який включає:

формування картинки корисної інформації необробленої послідовності байтів, шляхом кодування картинки для опису картинки у вкладених потоках WPP з одним вкладеним потоком WPP на рядок LCU картинки з ентропійним кодуванням необробленої послідовності байтів з використанням CABAC, передачу необробленої послідовності байтів траншами, на які розбиті вкладені потоки WPP, таким чином, вводячи в них межі траншу, і постійну адаптацію ймовірності CABAC в ентропійному кодуванні через межі траншу, введені у вкладених потоках WPP, причому корисна інформація необробленої послідовності байтів формується з використанням перетворення залишкових даних.

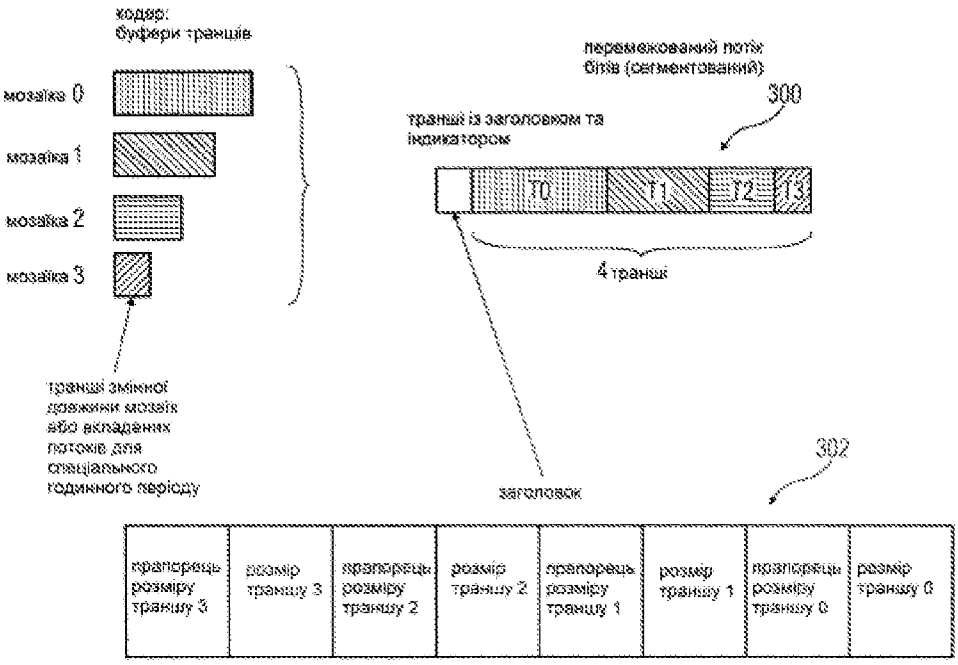
22. Цифровий носій даних, на якому зберігається програмний код для виконання, при запуску на комп'ютері, способу за одним з пп. 20-21.


1.ряд
2.ряд
. . .
останній ряд
1.вирізка
2.вирізка
3.вирізка
4.вирізка
1.вирізка

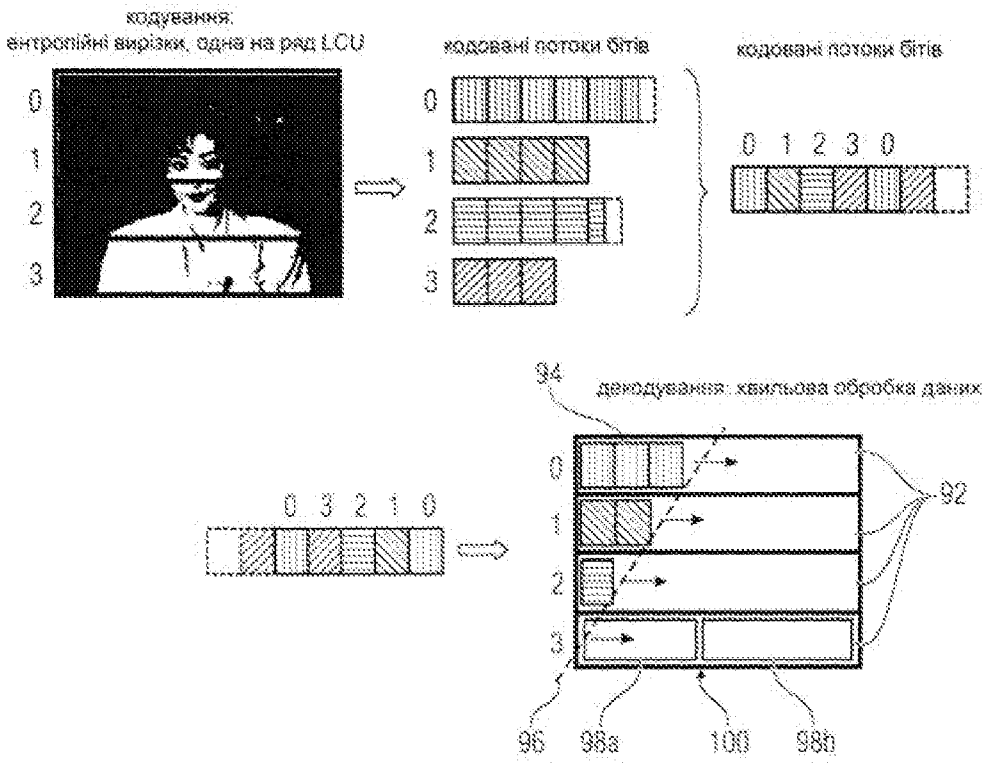
Фіг. 1


межі стовпчиків
мозаїка #1
мозаїка #2
мозаїка #3
межа вирізки
межа вирізки

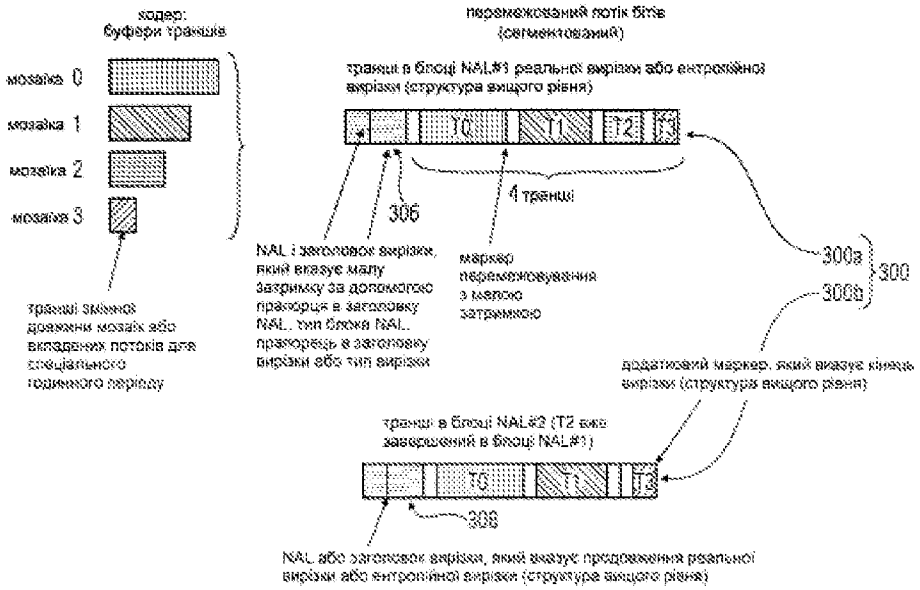
Фіг. 2


кодер:
буфери транців
мозаїка 0
мозаїка 1
мозаїка 2
мозаїка 3
транші змінної довжини мозаїк або вкладених потоків для спеціального годинного періоду
перемежований потік бітів (сегментований)
300
транші із заголовками та індикатором
T0
T1
T2
T3
4 транші
заголовок
302
прапорець розміру траншу 3
розмір траншу 3
прапорець розміру траншу 2
розмір траншу 2
прапорець розміру траншу 1
розмір траншу 1
прапорець розміру траншу 0
розмір траншу 0

Фіг. 3


кодування:
ентропійні вирізки, одна на ряд LCU
0
1
2
3
кодовані потоки бітів
кодовані потоки бітів
0 1 2 3 0
94
декодування: хвильова обробка даних
0 3 2 1 0
92
96
96a
100
98b

Фіг. 4

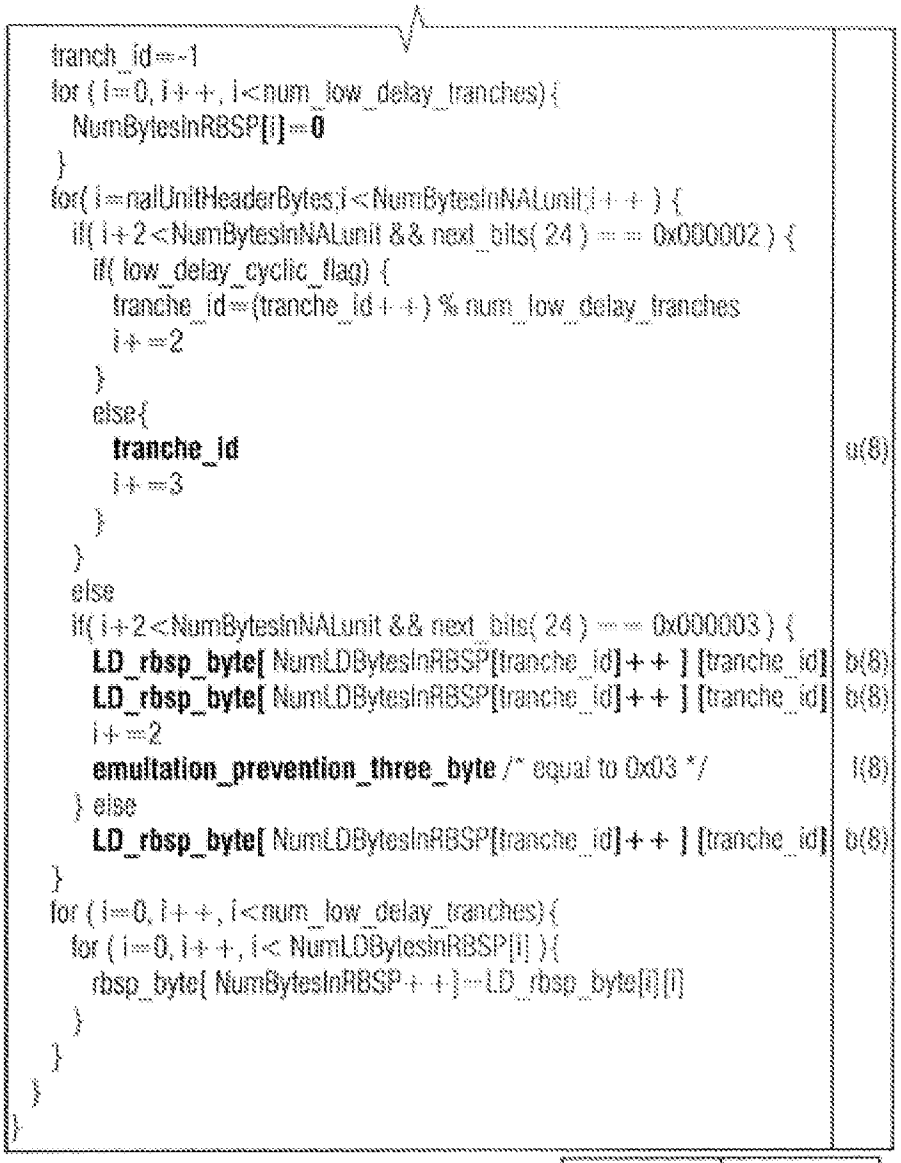

кодер:
буфери тришків
мозаїка 0
мозаїка 1
мозаїка 2
мозаїка 3
тришки кінець
довжини мозаїк або
вкладених потоків для
спеціального
годинного періоду
перемежований потік бітів
(сегментований)
тришки в блоці NAL#1 реальної вирізки або ентропійної
вирізки (структура вищого рівня)
T0
T1
T2
T3
4 тришки
306
NAL і заголовок вирізки,
який вказує малу
затримку за допомогою
прапорця в заголовку
NAL, тип блока NAL,
прапорець в заголовку
вирізки або тип вирізки
маркер
перемежовування
з малою
затримкою
300a
300b
300
додатковий маркер, який вказує кінець
вирізки (структура вищого рівня)
тришки в блоці NAL#2 (T2 вже
завершений в блоці NAL#1)
T0
T1
T3
308
NAL або заголовок вирізки, який вказує продовження реальної
вирізки або ентропійної вирізки (структура вищого рівня)

Фіг. 5

```
nal unit( NumBytesInNALunit, num low delay tranches ) {          Descriptor
  forbidden_zero_bit                                                  f(1)
  nal_ref_flag                                                        u(1)
  fnal_unit_type                                                      u(6)
  NumBytesInRBSP = 0
  nalUnitHeaderBytes = 1
  if( nal_unit_type = = 1| | nal_unit_type = = 4| | nal_unit_type = = 5 ) {
    temporal_id                                                       u(3)
    output_flag                                                       u(1)
    low_delay_encapsulation_flag                                      u(1)
    low_delay_cyclic_flag                                             u(1)
    reserved_one_2bits                                                u(2)
    nalUnitHeaderBytes + = 1
  }
  if( low_delay_encapsulation_flag != 1)
  {
    for( i = nalUnitHeaderBytes;i < NumBytesInNALunit; i + + ) {
      if( i + 2 < NumBytesInNALunit && next bits( 24 ) = = 0x000003 ) {
        rbsp_byte[ NumBytesInRBSP + + ]                               b(8)
        rbsp_byte[ NumBytesInRBSP + + ]                               b(8)
        i + = 2
        emulation_prevention_three_byte /* equal to 0x03 */           f(8)
      } else
        rbsp_byte[ NumBytesInRBSP + + ]                               b(8)
    }
  }
  else{
    for ( i = 0, i + +, i < num_low_delay_tranches){
      NumLDBytesInRBSP[i] = 0
    }
```

ФІГ. 6	ФІГ. 6A
	ФІГ. 6B

Фіг. 6A



```
tranch_id=-1
for ( i=0, i++, i<num_low_delay_tranches){
  NumBytesInRBSP[i]=0
}
for( i=nalUnitHeaderBytes;i<NumBytesInNALunit;i++ ) {
  if( i+2<NumBytesInNALunit && next_bits( 24 ) == 0x000002 ) {
    if( low_delay_cyclic_flag) {
      tranche_id=(tranche_id++) % num_low_delay_tranches
      i+=2
    }
    else{
      tranche_id                                                    u(8)
      i+=3
    }
  }
  else
  if( i+2<NumBytesInNALunit && next_bits( 24 ) == 0x000003 ) {
    LD_rbsp_byte[ NumLDBytesInRBSP[tranche_id]++ ] [tranche_id]  b(8)
    LD_rbsp_byte[ NumLDBytesInRBSP[tranche_id]++ ] [tranche_id]  b(8)
    i+=2
    emulation_prevention_three_byte /* equal to 0x03 */           f(8)
  } else
    LD_rbsp_byte[ NumLDBytesInRBSP[tranche_id]++ ] [tranche_id]  b(8)
}
for ( i=0, i++, i<num_low_delay_tranches){
  for ( i=0, i++, i< NumLDBytesInRBSP[i] ){
    rbsp_byte[ NumBytesInRBSP++ ]=LD_rbsp_byte[i][i]
  }
}
}
```

ФІГ. 6	ФІГ. 6A
	ФІГ. 6B

Фіг. 6В

...	
low_delay_encapsulaiton_present_flag	u(1)
if(low_delay_encapsulaiton_present_flag == 1){	
low_delay_cyclic_flag	u(1)
num_low_delay_tranches_flag	u(1)
if (num_low_delay_tranches_flag == 1){	
num_low_delay_tranches	ue(v)
for (i=0; i <num_low_delay_tranches; i ++){	
low_delay_tranche_ lenght_minus1 [i]	ue(v)
}	
}	
}	
...	

Фіг. 7

low_delay_slice_layer_rbsp( ) {	**Descriptor**
slice_header( )	
tranche_id=0	
for (i=0, i++, i<num_low_delay_tranches){	
tranche_slice_data( )	
}	
rbsp_slice_trailing_bits( )	

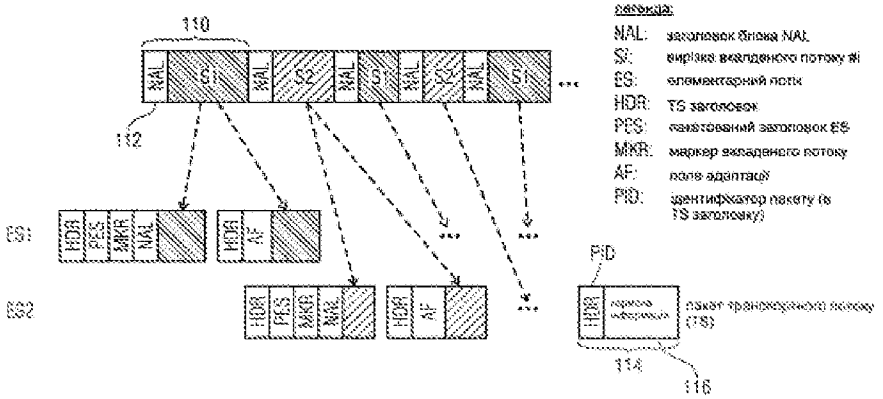
Фіг. 8

slice_header( ){	**Descriptor**
entropy_slice_flag	u(1)
low_delay_slice_flag	u(1)
...	
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag == 0){	
slice_adress	u(v)
if(low_delay_slice_flag == 1)	
no_cabac_reset_flag	u(1)
}	
...	
if(slice_type == P | slice_type == B)	
5_minus_max_num_merge_cand	ue(v)
for(i=0; i<num_substreams_minus1 + 1; i ++){	
substream_length_mode	u(2)
substream_length[i]	u(v)
}	
}	

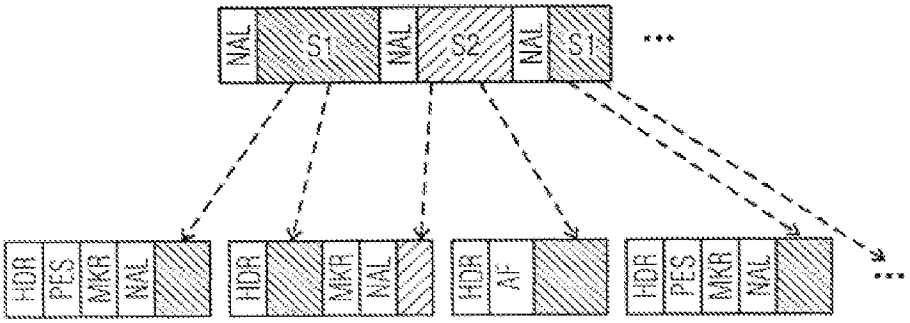
Фіг. 9

substream_marker(gradual_decoder_refresh_steps){	**Descriptor**
forbidden_zero_bit	f(1)
nal_ref_flag	u(1)
nal_unit_type	u(6)
substream_ID	u(8)
if(gradual_decoder_refresh_steps <= 1)	
is_intra	u(1)
reserved_one_7bits	u(7)
}	
}	

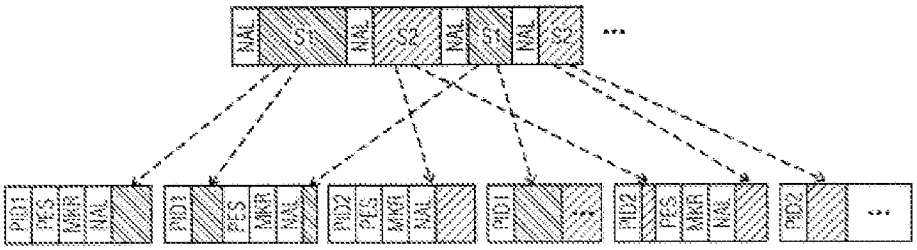
Фіг. 10



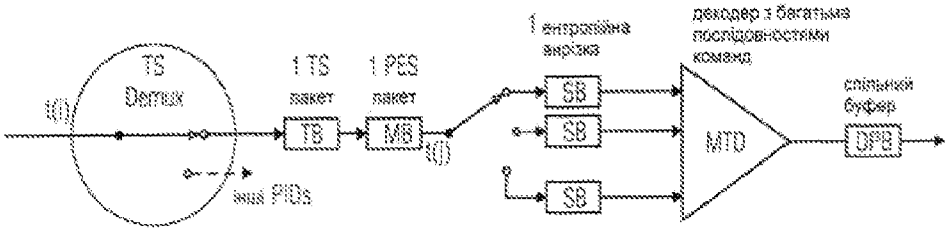
Фіг. 11



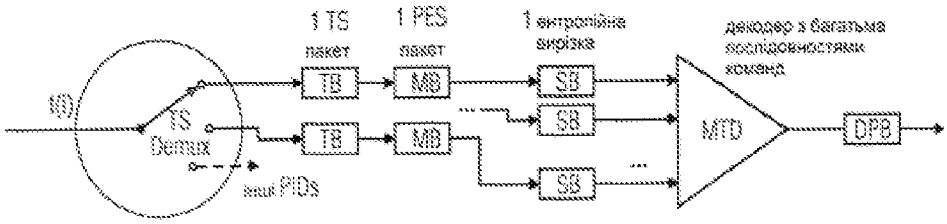
Фіг. 12



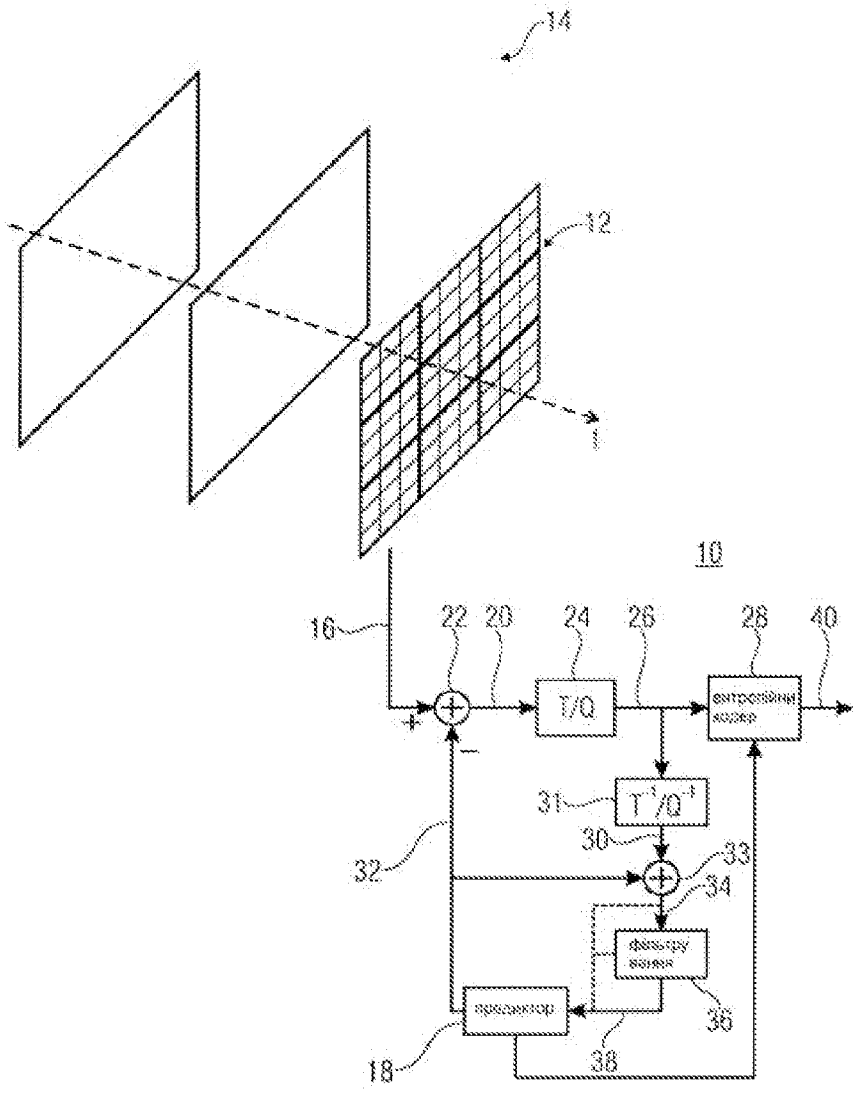
Фіг. 13



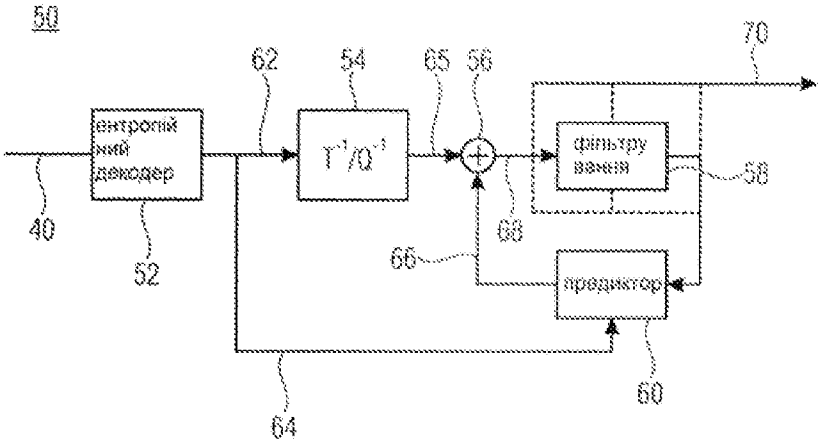
Фіг. 14


1 TS
пакет
1 PES
пакет
1 ентропійна вирізка
декодер з багатьма послідовностями команд
TS
Demux
інші PIDs
TB
MB
SB
MTD
DPB

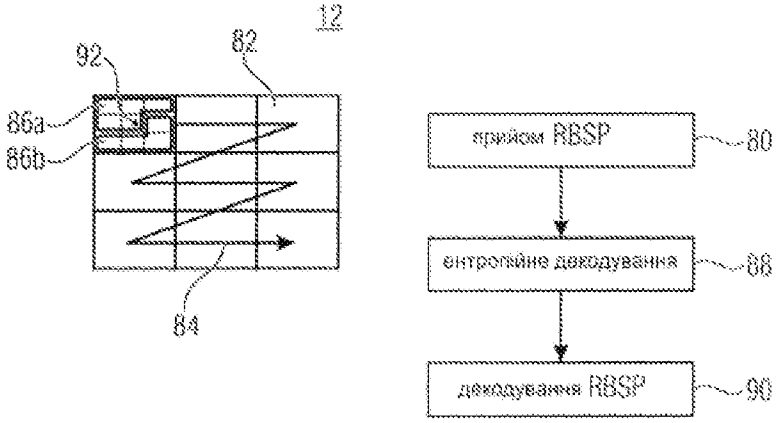
Фіг. 15


14
12
1
10
16
22
20
24
26
28
40
T/Q
31
30
33
34
32
18
36
38

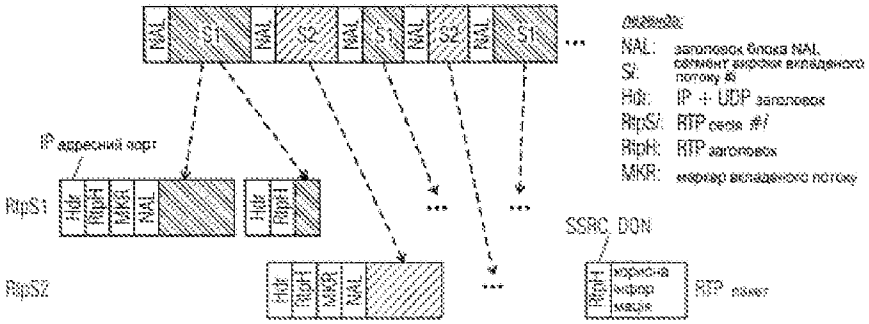
Фіг. 16



Фіг. 17



Фіг. 18



Фіг. 19

Комп'ютерна верстка О. Рябко

ДП "Український інститут інтелектуальної власності", вул. Глазунова, 1, м. Київ – 42, 01601