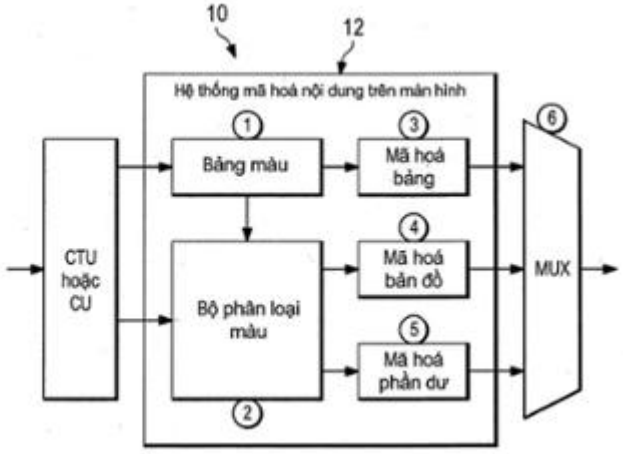
(12) BẢN MÔ TẢ SÁNG CHẾ THUỘC BẰNG ĐỘC QUYỀN SÁNG CHẾ
(19) **Cộng hòa xã hội chủ nghĩa Việt Nam (VN)**
CỤC SỞ HỮU TRÍ TUỆ
(11) **1-0027904**
(51)[7] **G06K 9/36**
(13) **B**

(21) 1-2016-01970
(22) 24/11/2014
(86) PCT/US2014/067155 24/11/2014
(87) WO2015/077720 28/05/2015
(30) 61/907,903 22/11/2013 US; 14/549,405 20/11/2014 US
(45) 25/04/2021 397
(43) 25/08/2016 341A
(73) HUAWEI TECHNOLOGIES CO., LTD. (CN)
Huawei Administration Building, Bantian, Longgang, Shenzhen, Guangdong 518129, China
(72) MA, Zhan (CN); WANG, Wei (CA); YU, Haoping (US); WANG, Xian (CN); YE, Jing (CN).
(74) Công ty Luật TNHH Phạm và Liên danh (PHAM & ASSOCIATES)

(54) PHƯƠNG PHÁP VÀ BỘ XỬ LÝ ĐỂ MÃ HÓA NỘI DUNG TRÊN MÀN HÌNH THÀNH LUỒNG BIT

(57) Sáng chế đề cập đến phương pháp và thiết bị để mã hoá nội dung trên màn hình thành luồng bit bằng cách chọn bảng màu cho đơn vị mã hoá (Coding Unit - CU) của nội dung trên màn hình, tạo ra bản đồ chỉ số màu có các chỉ số cho đơn vị mã hoá (CU), và mã hoá bảng màu được chọn và bản đồ chỉ số màu đối với CU thành luồng bit.



Lĩnh vực kĩ thuật được đề cập

Sáng chế đề cập chung đến kĩ thuật mã hoá nội dung trên màn hình.

Tình trạng kĩ thuật của sáng chế

Việc mã hoá nội dung trên màn hình đã đặt ra những thách thức mới cho công nghệ nén viđeo do các đặc điểm tín hiệu riêng biệt của nó so với các viđeo tự nhiên thông thường. Dường như đã có một vài kĩ thuật hứa hẹn để cải tiến phương pháp mã hoá nội dung trên màn hình, ví dụ, kĩ thuật so khớp chuỗi giả, mã hoá bảng màu, và bù chuyển động nội suy hoặc sao chép khối nội suy.

Trong số những kĩ thuật này, thì kĩ thuật so khớp chuỗi giả là có lợi nhất để mã hoá không tổn hao, nhưng lại gây ra sự phức tạp, gánh nặng và những khó khăn đáng kể đối với chế độ mã hoá có tổn hao. Kĩ thuật mã hoá bảng màu được phát triển cho nội dung trên màn hình với giả định rằng nội dung không phải được chụp bằng camera thì thường chứa một lượng hạn chế các màu sắc khác biệt, chứ không phải tông màu liên tục trong các viđeo tự nhiên. Mặc dù phương pháp so khớp chuỗi giả và mã hoá bảng màu có tiềm năng lớn, nhưng kĩ thuật bù chuyển động nội suy hoặc sao chép khối nội suy đã được đưa vào dự thảo (Working Draft - WD) phiên bản 4 và phần mềm tham chiếu của kĩ thuật mở rộng dải HEVC (High Efficiency Video Coding Range Extension - HEVC REext - mở rộng dải mã hoá viđeo hiệu quả cao) hiện nay để mã hoá nội dung trên màn hình. Điều này chủ yếu là do kĩ thuật ước lượng và bù chuyển động đã được nghiên cứu kĩ lưỡng qua nhiều thập kỉ, và ý tưởng và khả năng thực hiện thực tế của kĩ thuật này là tương đối dễ dàng (đặc biệt là đối với phần cứng).

Tuy nhiên, hiệu suất mã hoá của kĩ thuật sao chép khối nội suy bị hạn chế do các phân vùng cấu trúc khối cố định của nó. Mặt khác, quá trình thực hiện việc so khớp khối, vốn tương tự như việc ước lượng chuyển động nội ảnh, cũng gây ra sự phức tạp đáng kể cho bộ mã hoá trong cả quá trình tính toán lẫn quá trình truy cập bộ nhớ.

Bản chất kĩ thuật của sáng chế

Mục đích của sáng chế là đề xuất giải pháp cải tiến để mã hoá nội dung trên màn hình.

Theo một phương án ví dụ, phương pháp để mã hoá nội dung trên màn hình thành luồng bit bao gồm bước chọn bảng màu cho đơn vị mã hoá (Coding Unit - CU) của nội dung trên màn hình. Bảng màu này được tạo ra cho CU này và bảng màu khác được tạo ra cho CU lân cận. Bản đồ chỉ số màu được tạo ra và có các chỉ số dành cho đơn vị mã hoá (CU) của nội dung trên màn hình, nhờ sử dụng bảng màu được chọn. Bảng màu được chọn và bản đồ chỉ số màu này được mã hoá/được nén đối với mỗi trong số các CU, thành luồng bit.

Mô tả vắn tắt các hình vẽ

Sáng chế sẽ được mô tả dưới đây dựa vào các hình vẽ kèm theo để cho phép hiểu toàn vẹn hơn về sáng chế và các ưu điểm của sáng chế, trong đó các số chỉ dẫn giống nhau được dùng để chỉ các đối tượng giống nhau, và trong đó:

Fig.1 là hình vẽ thể hiện giải pháp mã hoá nội dung trên màn hình có sử dụng chế độ bảng màu và bản đồ chỉ số hoặc chế độ bảng theo một phương án ví dụ của sáng chế;

Fig.2 là hình vẽ thể hiện giải pháp giải mã nội dung trên màn hình đối với chế độ bảng màu và bản đồ chỉ số hoặc chế độ bảng;

Fig.3 là hình vẽ thể hiện tiến trình xử lý nội dung trên màn hình đối với chế độ bảng màu và bản đồ chỉ số này hoặc chế độ bảng của CU;

Fig.4 là hình vẽ thể hiện màu G, B, R thông thường trong chế độ phẳng (bên trái) sang chế độ đóng gói (bên phải);

Fig.5 là hình vẽ thể hiện lưu đồ tạo lại bảng màu nhờ sử dụng các khối được tái tạo lân cận;

Fig.6 là hình vẽ thể hiện quá trình phân tích bản đồ chỉ số từ nội dung trên màn hình ngoài đời thực;

Fig.7 là hình vẽ thể hiện một phân đoạn của kết quả tìm kiếm 1-D sau khi quét ngang;

Fig.8 là hình vẽ thể hiện môđun U_PIXEL;

Fig.9 là hình vẽ thể hiện môđun U_ROW;

Fig.10 là hình vẽ thể hiện môđun U_CMP;

Fig.11 là hình vẽ thể hiện môđun U_COL;

Fig.12 là hình vẽ thể hiện môđun U_2D_BLOCK;

Fig.13 là hình vẽ thể hiện quá trình quét ngang và quét dọc để xử lý bản đồ chỉ số của CU được nêu làm ví dụ;

Fig.14A là hình vẽ thể hiện định dạng lấy mẫu sắc độ 4:2:0;

Fig.14B là hình vẽ thể hiện định dạng lấy mẫu sắc độ 4:4:4;

Fig.15 là hình vẽ thể hiện kết quả nội suy giữa 4:2:0 và 4:4:4;

Fig.16 là hình vẽ thể hiện quá trình xử lý bản đồ chỉ số với bộ đệm đường trên/trái;

Fig.17 là hình vẽ thể hiện thiết bị và các phương pháp/lưu đồ được áp dụng cho HEVC hiện tại;

Fig.18 là hình vẽ thể hiện một ví dụ về hệ thống truyền thông; và

Fig.19A và Fig.19B là các hình vẽ thể hiện các thiết bị ví dụ mà có thể thực hiện các phương pháp và nguyên lý theo sáng chế.

Mô tả chi tiết sáng chế

Sáng chế đề xuất giải pháp cải tiến để mã hoá nội dung trên màn hình để thực hiện quá trình mở rộng dải mã hoá viđeo hiệu quả cao (HEVC) (chẳng hạn HEVC phiên bản 2 hoặc HEVC RExt). Giải pháp mới này bao gồm một số thuật toán được thiết kế cụ thể để mã hoá nội dung trên màn hình. Các thuật toán này bao gồm việc biểu diễn điểm ảnh nhờ sử dụng bảng màu, nén bảng màu, nén bản đồ chỉ số màu, tìm kiếm chuỗi, và nén phần dư. Công nghệ này được phát triển, được làm hài hoà, và có thể được tích hợp với HEVC RExt và các HEVC extensions trong tương lai để hỗ trợ quá trình mã hoá nội dung trên màn hình một cách có hiệu quả. Tuy nhiên, công nghệ này có thể được thực hiện với các chuẩn viđeo hiện tại bất kì. Để cho đơn giản, thì HEVC RExt được dùng làm ví dụ trong phần mô tả dưới đây, và phần mềm HEVC RExt được dùng để mô tả và biểu diễn hiệu quả nén. Giải pháp này được tích hợp dưới dạng chế độ bổ sung nhờ sử dụng

bảng màu và bản đồ chỉ số, ở đây được xác định dưới dạng chế độ bảng màu, trong HEVC để biểu diễn hiệu suất.

Ý tưởng và phương án thực hiện sáng chế được thể hiện trên các hình vẽ. Fig.1 thể hiện bộ mã hoá 10 có bộ xử lý 12 bao gồm bộ nhớ, và Fig.2 thể hiện bộ giải mã 14 có bộ xử lý 16 và bộ nhớ, lần lượt thể hiện một phương án ví dụ về giải pháp mã hoá và giải mã đối với chế độ bảng màu theo sáng chế. Như được thể hiện, mỗi trong số bộ mã hoá 10 và bộ giải mã 14 đều bao gồm bộ xử lý và bộ nhớ, và tạo thành bộ codec. Bộ codec này bao gồm bộ xử lý 12 của bộ mã hoá 10 để thực hiện các thuật toán hoặc các phương pháp mới bao gồm tiến trình 1 để tạo ra bảng màu, tiến trình 2 để phân loại các màu sắc hoặc các giá trị điểm ảnh nhờ sử dụng bảng màu được trích ra trước đó tương ứng với các chỉ số màu, tiến trình 3 để mã hoá bảng màu này, tiến trình 4 để mã hoá bản đồ chỉ số màu, tiến trình 5 để mã hoá các phần dư, và tiến trình 6 để ghi các phần tử cú pháp mới vào luồng bit được nén. Bộ xử lý 16 của bộ giải mã 14 thực hiện các thuật toán hoặc các phương pháp mới bao gồm các bước ngược lại. Fig.3 là hình vẽ thể hiện tiến trình xử lý nội dung trên màn hình theo sáng chế.

Về cơ bản, thì phương pháp nén bảng màu (Color Palette Compression - CPC) hiệu quả cao được thực hiện trên mỗi đơn vị mã hoá (CU). Đơn vị mã hoá là đơn vị hoạt động cơ bản trong HEVC và HEVC REext, đó là khối hình vuông bao gồm các điểm ảnh và bao gồm ba thành phần (đó là RGB, hoặc YUV, hoặc XYZ).

Tại mỗi cấp độ CU, thì phương pháp CPC bao gồm hai bước chủ yếu. Đầu tiên, bộ xử lý 12 trích ra hoặc tạo ra bảng màu ở bước thứ nhất. Bảng này được xếp thứ tự theo biểu đồ (tức là tần suất xuất hiện của mỗi giá trị màu), hoặc cường độ màu thực tế của nó, hoặc theo phương pháp tuỳ ý bất kì, để tăng hiệu quả của tiến trình mã hoá sau đó. Dựa trên bảng màu được trích ra, mỗi điểm ảnh trong CU ban đầu sẽ được biến đổi thành chỉ số màu của nó trong bảng màu này. Giải pháp của sáng chế là công nghệ để mã hoá một cách hiệu quả, chẳng hạn bằng cách nén, bảng màu và bản đồ chỉ số màu của mỗi CU thành luồng. Ở phía thu, bộ xử lý 16 phân tích luồng bit được nén, để tái tạo, cho mỗi CU, bảng màu

hoàn chỉnh và bản đồ chỉ số màu, và sau đó tiếp tục trích ra giá trị điểm ảnh tại mỗi vị trí bằng cách kết hợp chỉ số màu và bảng màu này.

Theo một ví dụ minh hoạ theo sáng chế, giả sử rằng một CU có NxN điểm ảnh (N=8, 16, 32, 64 để tương thích với HEVC). CU thường chứa ba thành phần độ màu (sắc độ) (tức là G, B, R, hoặc Y, Cb, Cr, hoặc X, Y Z) với tỉ số lấy mẫu khác nhau (tức là 4:4:4, 4:2:2, 4:2:0). Để cho đơn giản, thì các chuỗi 4:4:4 được minh hoạ theo sáng chế. Đối với các chuỗi của các viđeo 4:2:2 và 4:2:0, thì quá trình lấy mẫu tăng sắc độ có thể được áp dụng để thu được các chuỗi 4:4:4, hoặc mỗi thành phần màu có thể được xử lý độc lập. Sau đó, quy trình giống như được mô tả theo sáng chế có thể được áp dụng. Đối với các viđeo đơn sắc 4:0:0, thì điều này có thể được coi là mặt phẳng riêng rẽ 4:4:4 mà không có hai mặt phẳng kia. Tất cả các phương pháp đối với 4:4:4 đều có thể được áp dụng trực tiếp.

Được đóng gói hoặc phẳng

Phương pháp này được mô tả đối với CTU khối hoặc CU trên Fig.1. Trước hết, một cờ được xác định, được gọi là enable_packed_component_flag, đối với mỗi CU để cho biết xem CU hiện tại được xử lý theo kiểu đóng gói hay chế độ phẳng thông thường (tức là các thành phần G, B, R hoặc Y, U, V là được xử lý độc lập). Fig.4 là hình vẽ thể hiện màu G, B, R thông thường trong chế độ phẳng (bên trái) sang chế độ đóng gói (bên phải). YUV hoặc định dạng màu khác có thể được xử lý theo cách giống như được nêu đối với nội dung RGB.

Cả chế độ đóng gói và chế độ phẳng đều có ưu điểm và nhược điểm riêng của chúng. Ví dụ, chế độ phẳng có hỗ trợ việc xử lý thành phần màu song song đối với màu G/B/R hoặc Y/U/V. Tuy nhiên, chế độ này có thể có hiệu quả mã hoá thấp. Chế độ đóng gói có thể dùng chung thông tin mào đầu (chẳng hạn bảng màu và bản đồ chỉ số theo sáng chế) đối với CU này trong số các thành phần màu khác nhau. Tuy nhiên, chế độ này có thể phá vỡ tính song song. Một cách dễ dàng để quyết định xem có nên mã hoá CU hiện tại theo kiểu đóng gói hay không là đo sự méo tốc độ (rate distortion - R-D). Cờ enable_packed_component_flag được dùng để báo hiệu tường minh chế độ mã hoá cho bộ giải mã.

Ngoài ra, để xác định cờ enable_packed_component_flag tại cấp độ CU để xử lý cấp thấp, thì nó có thể được nhân bản trong mào đầu mành cắt hoặc thậm chí là cấp độ chuỗi (ví dụ, Sequence Parameter Set - tập hợp thông số chuỗi, hoặc Picture Parameter Set - tập hợp thông số hình ảnh), để cho phép xử lý cấp độ mành cắt hoặc cấp độ chuỗi, tuỳ theo yêu cầu ứng dụng cụ thể.

Trích xuất bảng màu và bản đồ chỉ số

Như được thể hiện trên Fig.1, đối với các tiến trình 1 và 3, đối với mỗi CU, thì các vị trí điểm ảnh được duyệt qua và bảng màu và bản đồ chỉ số dành cho quá trình xử lý tiếp theo được trích ra. Mỗi màu khác nhau sẽ được xếp thứ tự trong bảng màu, tuỳ theo biểu đồ (tức tần suất xuất hiện) hoặc cường độ của nó, hoặc theo phương pháp tuỳ ý bất kì, để tăng hiệu quả của tiến trình mã hoá sau đó. Ví dụ, nếu tiến trình mã hoá mà dùng phương pháp điều chế xung-mã vi phân (Differential Pulse-Code Modulation - DPCM) để mã hoá sự khác biệt giữa các điểm ảnh liền kề, thì có thể thu được kết quả mã hoá tối ưu nếu các điểm ảnh liền kề được gán chỉ số màu liền kề trong Bảng màu.

Sau khi thu được bảng màu, mỗi điểm ảnh sẽ được ánh xạ vào chỉ số màu tương ứng để tạo thành bản đồ chỉ số của CU hiện tại. Tiến trình liên quan đến bản đồ chỉ số được mô tả trong phần tiếp theo.

Đối với CU phẳng thông thường, thì mỗi thành phần màu hoặc thành phần độ màu có thể có bảng màu riêng rẽ của nó, chẳng hạn colorTable_Y, colorTable_U, colorTable_V hoặc colorTable_R, colorTable_G, colorTable_B. Trong khi đó, bảng màu đối với thành phần chính có thể được trích ra, chẳng hạn Y trong YUV hoặc G trong GBR, và được dùng chung cho tất cả các thành phần. Thông thường, do việc dùng chung này, mà các thành phần màu khác, mà không phải là Y hoặc G, sẽ có một chút không khớp đối với các màu điểm ảnh ban đầu của nó so với các thành phần màu được dùng chung trong bảng màu. Sau đó, cơ chế xử lý phần dư (chẳng hạn các phương pháp mã hoá các hệ số HEVC) được áp dụng để mã hoá các phần dư không khớp nhau này. Ngược lại, đối với CU được đóng gói, thì một bảng màu được dùng chung giữa tất cả các thành phần.

Một đoạn mã giả sẽ được thể hiện để minh hoạ quá trình trích xuất bảng màu và bản đồ chỉ số như sau:

```
deriveColorTableIndexMap()
{
  deriveColorTable();
  deriveIndexMap();
}

  deriveColorTable(src, cuWidth, cuHeight, maxColorNum)
  {
    // src – nguồn viđeo đầu vào ở chế độ phẳng hoặc chế độ đóng gói
    // cuWidth, cuHeight – chiều rộng và chiều cao của CU hiện tại
    /* maxColorNum – số lượng màu tối đa cho phép trong bảng màu/
    /*transverse */
    //
    //  memset(colorHist, 0, (1<<bitDepth)*sizeof(UINT))
    pos=0;
    cuSize=cuWidth*cuHeight;
    while (pos<cuSize) {
      colorHist[src[pos++]]++;
    }

    /*just pick non-zero entry in colorHist[] for color intensity ordered
table*/
    j=0;
    for(i=0;i<(1<<bitDepth);i++)
    {
     if(colorHist[i]!=0)
       colorTableIntensity[j++] = colorHist[i];
     }
```

```
colorNum=j;

/*quicksort for histgram*/

colorTableHist = quickSort(colorTableIntensity, colorNum);

/*if maxColorNum >= colorNum, all colors will be picked*/
/*if maxColorNum < colorNum, only maxColorNum colors will be picked for colorTableHist. Trong trường hợp này, tất cả các điểm ảnh sẽ tìm màu khớp nhất và chỉ số tương ứng của nó, với sự khác biệt (điểm ảnh thực tế và màu tương ứng của nó) được mã hoá bởi cơ chế xử lý phần dư.*/
/*Best number of colors in color table could be determined by iterative R-D cost derivation!*/
}

deriveIndexMap()
{
  pos=0;
  cuSize=cuWidth*cuHeight;
  while ( pos < cuSize)
  {
    minErr=MAX_UINT;
    for (i=0;i<colorNum;i++)
    {
     err = abs(src[pos] – colorTable[i]);
     if (err<minErr)
     {
       minErr = err;
      idx = i;
     }
```

```
            }
            idxMap[pos] = idx;
        }
    }
```

Tiến trình xử lý bảng màu

Đối với tiến trình 1 trên Fig.1, thì hệ thống xử lý bảng màu bao gồm bộ xử lý 12 để mã hoá kích thước bảng màu (tức tổng số màu khác nhau) và bản thân mỗi màu. Phần lớn bit được dùng để mã hoá mỗi màu trong bảng màu. Do đó, phần mô tả này sẽ tập trung vào quá trình mã hoá màu (hoặc mã hoá mỗi mục nhập trong bảng màu).

Phương pháp đơn giản nhất để mã hoá các màu trong bảng màu là dùng thuật toán điều chế xung-mã (Pulse-Code Modulation - PCM) mà trong đó mỗi màu được mã hoá độc lập. Theo cách khác, kết quả dự đoán gần đúng nhất đối với màu tiếp theo có thể được áp dụng, sau đó đenta dự đoán có thể được mã hoá thay vì cường độ màu mặc định, tức là theo kiểu DPCM (Differential PCM - PCM vi phân). Cả hai phương pháp này đều có thể được mã hoá entropy sau đó nhờ sử dụng mô hình xác suất ngang bằng hoặc mô hình ngữ cảnh thích ứng, tuỳ theo sự đánh đổi giữa độ phức tạp và hiệu quả mã hoá.

Ở đây, sơ đồ cải tiến khác được bộc lộ, gọi là Neighboring Color palette table Merge (hợp nhất bảng màu lân cận), trong đó cờ color_table_merge_flag được xác định để cho biết xem CU hiện tại sử dụng bảng màu từ CU bên trái hay CU phía trên của nó. Nếu không, thì CU hiện tại sẽ mang báo hiệu bảng màu một cách tường minh. Đối với tiến trình hợp nhất, thì cờ color_table_merge_direction khác sẽ chỉ thị chiều hợp nhất từ CU phía trên hoặc từ CU phía trái. Tất nhiên, các CU ứng viên có thể không chỉ bao gồm CU phía trên hoặc CU phía trái hiện tại, ví dụ, CU phía trên bên trái, CU phía trên bên phải, v.v.. Tuy nhiên, CU phía trên và phía trái được dùng để mô tả giải pháp của sáng chế. Đối với bất kì trong số đó, mỗi điểm ảnh sẽ được so sánh với các mục nhập trong bảng màu hiện tại và được gán chỉ số mà cho ra kết quả dự đoán ít

chênh lệch nhất (tức là điểm ảnh trừ đi màu gần nhất trong bảng màu) thông qua deriveIdxMap(). Đối với trường hợp mà sự chênh lệch dự đoán là khác không, thì tất cả các phần dư này sẽ được mã hoá nhờ sử dụng cơ chế xử lý phần dư HEVC REext. Lưu ý rằng việc có sử dụng tiến trình hợp nhất hay không là có thể được quyết định bởi cái giá R-D.

Có vài cách để tạo ra các bảng màu lân cận để dùng trong tiến trình hợp nhất khi mã hoá CU hiện tại. Tuỳ theo cách thức thực hiện, một trong số những cách này yêu cầu cập nhật tại cả bộ mã hoá và bộ giải mã, và cách còn lại chỉ là tiến trình phía bộ mã hoá.

Cập nhật tại cả bộ mã hoá và bộ giải mã: Theo phương pháp này, bảng màu của các CU lân cận được tạo ra trên các điểm ảnh được tái tạo hiện có, không phụ thuộc vào chiều sâu, kích thước CU, v.v. Đối với mỗi CU, thì những hoạt động tái tạo được thực hiện cho CU lân cận của nó tại cùng kích thước và cùng chiều sâu (giả sử rằng sự tương tự về màu sắc sẽ cao hơn trong trường hợp này). Ví dụ, nếu CU hiện tại là 16x16 với chiều sâu = 2, thì không phụ thuộc vào phân vùng của các CU lân cận của nó (ví dụ 8x8 với chiều sâu = 3 đối với CU bên trái và 32x32 với chiều sâu = 1 đối với CU bên trên), độ dịch điểm ảnh (=16) sẽ là từ CU hiện tại về phía trái để xử lý khối 16x16 bên trái, và về phía trên đối với khối 16x16 phía trên, như được thể hiện trên Fig.5. Lưu ý rằng cả bộ mã hoá và bộ giải mã đều phải duy trì tiến trình này.

Tiến trình chỉ ràng buộc bộ mã hoá: đối với phương pháp này, tiến trình hợp nhất xảy ra khi CU hiện tại dùng chung cùng một kích thước và chiều sâu với CU phía trên và/hoặc CU bên trái của nó. Các bảng màu của các CU lân cận hiện có được dùng để trích xuất bản đồ chỉ số màu của CU hiện tại để dùng cho các hoạt động sau đó. Ví dụ, đối với CU 16x16 hiện tại, nếu CU lân cận của nó, tức CU nằm phía trên hoặc bên trái, mà được mã hoá bằng phương pháp bảng màu và chỉ số, thì bảng màu của nó được dùng trực tiếp cho CU hiện tại để trích ra cái giá R-D. Cái giá hợp nhất này được so sánh với trường hợp mà CU hiện tại trích ra bảng màu của nó một cách tường minh (cũng như các chế độ thông thường khác hiện có trong HEVC hoặc HEVC REext). Cái nào có cái giá R-D thấp hơn sẽ được chọn làm chế độ cuối cùng để ghi vào luồng bit được xuất ra. Như có thể thấy,

chỉ cần bộ mã hoá để thực nghiệm/mô phỏng các chế độ tiềm năng khác nhau. Ở phía bộ giải mã, cờ color_table_merge_flag và cờ color_table_merge_direction cho biết quyết định hợp nhất và chiều hợp nhất mà không cần thêm quá trình xử lý nào.

Xử lý bản đồ chỉ số màu

Đối với tiến trình 3 trên Fig.1, để mã hoá bản đồ chỉ số màu, thì một vài giải pháp đã được nghiên cứu, chẳng hạn chế độ RUN, RUN và COPY_ABOVE, và dự đoán chỉ số lân cận thích ứng. Theo sáng chế, phương pháp so khớp chuỗi 1D và biến thể 2D của nó được dùng để mã hoá bản đồ chỉ số. Tại mỗi vị trí, nó tìm kiếm điểm khớp của nó và ghi lại khoảng cách và chiều dài khớp đối với phương pháp so khớp chuỗi 1D, hoặc chiều rộng/chiều cao đối với phương pháp so khớp chuỗi 2D. Đối với vị trí không khớp, thì cường độ chỉ số của nó, hoặc giá trị đenta giữa cường độ chỉ số và cường độ chỉ số dự đoán được của nó, được mã hoá trực tiếp.

Được bộc lộ ở đây là phương pháp tìm kiếm 1D đơn giản trên bản đồ chỉ số màu. Như được thể hiện trên Fig.6, bản đồ chỉ số được phân tích từ nội dung trên màn hình ngoài đời thực. Fig.7 thể hiện một phân đoạn sau khi tìm kiếm 1-D (tức ngay chỗ bắt đầu của bản đồ chỉ số này).

Phương pháp so khớp chuỗi được áp dụng trên đầu vectơ chỉ số màu 1-D này. Một ví dụ về phương pháp so khớp chuỗi 1-D này được mô tả dưới đây. Đối với vị trí thứ nhất của mỗi bản đồ chỉ số, chẳng hạn 14 như được thể hiện trên Fig.7, do chưa có tham chiếu được đệm nào, nên chỉ số ngay đầu tiên này được coi như "cặp không khớp", trong đó khoảng cách và chiều dài tương ứng của nó lấy giá trị -1 và 1, được ghi chú dưới dạng (dist, len) = (-1, 1). Đối với chỉ số thứ hai, tức là lại "14" khác, thì đó là chỉ số thứ nhất được mã hoá làm tham chiếu, do đó dist=1. Bởi vì có "14" khác tại vị trí thứ ba, nên chiều dài là 2, tức len=2, (giả sử rằng mỗi chỉ số về sau có thể được coi như tham chiếu ngay cho chỉ số tiếp theo). Tới vị trí thứ tư là "17", vốn chưa được bắt gặp trước đó. Do đó, nó lại được mã hoá dưới dạng cặp không khớp, tức (dist, len) = (-1, 1). Đối với cặp không khớp này, cờ này được mã hoá (chẳng hạn "dist == -1") và tiếp theo là giá trị thực của

chỉ số này (như “14”, “17”, “6” đã xuất hiện đầu tiên, v.v). Mặt khác, đối với các cặp khớp, cờ này vẫn được mã hoá (chẳng hạn “dist != -1”), và tiếp theo là chiều dài của chuỗi được so khớp.

Sau đây là tóm tắt của quy trình mã hoá có sử dụng chỉ số được nêu làm ví dụ trên Fig.7.

dist = -1, len = 1, idx=14 (không khớp)
dist= 1, len = 2 (khớp)
dist = -1, len = 1, idx=17 (không khớp)
dist= 1, len = 3 (khớp)
dist = -1, len = 1, idx= 6 (không khớp)
dist= 1, len = 25 (khớp)
dist= 30, len = 4 (khớp) /*đối với số “17” đã xuất hiện trước đó*/
...

Một đoạn mã giả được thể hiện cho dẫn xuất cặp khớp này, đó là,

```
Void deriveMatchedPairs ( TComDataCU* pcCU, Pel* pIdx, Pel* pDist,
Pel* pLen, UInt uiWidth, UInt uiHeight)
{
// pIdx là idx CU bị giới hạn trong uiWidth*uiHeight

UInt uiTotal = uiWidth*uiHeight;
UInt uiIdx   = 0;
Int j   = 0;
Int len = 0;

// điểm ảnh thứ nhất được mã hoá dưới dạng chính nó nếu không có điểm
đệm bên trái/phía trên
pDist[uiIdx] = -1;
pLen[uiIdx]  =  0;
uiIdx++;
```

```
while (uiIdx < uiTotal )
{
        len   = 0;
        dist  = -1;
        for ( j=uiIdx-1; j >= 0; j-- )
        {
                // nếu tìm thấy cặp khớp, thì phương pháp tìm kiếm toàn bộ hiện tại được áp dụng
                // phương pháp tìm kiếm chuỗi nhanh có thể được áp dụng
                if ( pIdx[j] == pIdx[uiIdx] )
                {
                        for (len = 0; len < (uiTotal-uiIdx); len++ )
                        {
                        if ( pIdx[j+len] != pIdx[len+uiIdx] )
                                break;
                        }
                }
                if ( len > maxLen ) /*better to change with R-D decision*/
                {
                        maxLen = len;
                        dist   = (uiIdx - j );
                }
        }


        pDist[uiIdx] = dist;
        pLen[uiIdx]  = maxLen;
        uiIdx = uiIdx + maxLen;
```

```
        }
    }
```

Các bước sau đây được thực hiện khi biến thể tìm kiếm 2D được sử dụng:

1. Nhận dạng vị trí của điểm ảnh hiện tại và điểm ảnh tham chiếu làm điểm khởi đầu,

2. Áp dụng phương pháp so khớp chuỗi 1D theo chiều ngang sang hướng bên phải của điểm ảnh hiện tại và điểm ảnh tham chiếu. Chiều dài tìm kiếm tối đa bị giới hạn bởi đuôi của hàng ngang hiện tại. Ghi lại chiều dài tìm kiếm tối đa này dưới dạng right_width

3. Áp dụng phương pháp so khớp chuỗi 1D theo chiều ngang sang hướng bên trái của điểm ảnh hiện tại và điểm ảnh tham chiếu. Chiều dài tìm kiếm tối đa bị giới hạn bởi đầu của hàng ngang hiện tại. Ghi lại chiều dài tìm kiếm tối đa này dưới dạng left_width

4. Thực hiện hoạt động so khớp chuỗi 1D này tại hàng kế tiếp, lấy các điểm ảnh bên dưới điểm ảnh hiện tại và điểm ảnh tham chiếu làm điểm ảnh hiện tại mới và điểm ảnh tham chiếu mới

5. Dừng lại cho đến khi right_width == left_width == 0.

6. Lúc này đối với mỗi height[n] = {1, 2, 3…}, có mảng tương ứng là width[n] {{left_width[1], right_width[1]}, {left_width[2], right_width[2]}, {left_width[3], right_width[3]}…}

7. Xác định mảng min_width mới {{lwidth[1], rwidth[1]}, {lwidth[2], rwidth[2]}, {lwidth[3], rwidth[3]}…} đối với mỗi height[n], trong đó lwidth[n] = min(left_width[1:n-1]), rwidth[n] = min(right_width[1:n-1])

8. Kích thước array{size[1], size[2], size[3]…} cũng được xác định, trong đó size[n] = height[n] x (lwidth[n]+hwidth[n])

9. Giả sử size[n] lấy giá trị lớn nhất trong mảng kích thước, thì chiều rộng và chiều cao của mảng so khớp chuỗi 2D được chọn nhờ sử dụng {lwidth[n], rwidth[n], height[n]} tương ứng

Một cách để tối ưu hoá tốc độ tìm kiếm 1D hoặc 2D là dùng running hash. Cấu trúc running hash 4 điểm ảnh được mô tả theo sáng chế. Một running hash được tính toán đối với mỗi điểm ảnh theo chiều ngang để tạo ra mảng hash ngang running_hash_h[]. Running hash khác được tính toán trên đầu running_hash_h[] để tạo ra mảng hash 2D running_hash_hv[]. Mỗi giá trị khớp trong mảng hash 2D này biểu thị một khối khớp 4x4. Để thực hiện thao tác so khớp 2D, thì cần tìm càng nhiều khối khớp 4x4 càng tốt trước khi thực hiện việc so sánh theo điểm ảnh với các điểm lân cận của chúng. Do việc so sánh theo điểm ảnh bị giới hạn ở 1-3 điểm ảnh, nên tốc độ tìm kiếm có thể được tăng lên rất nhiều.

Có thể thấy từ phần mô tả nêu trên rằng các chiều rộng được so khớp của mỗi hàng là khác nhau, do đó, mỗi hàng phải được xử lý một cách riêng rẽ. Để đạt được hiệu quả và giảm thấp sự phức tạp, thì thuật toán dựa trên khối được áp dụng, và có thể được sử dụng cả khi thực hiện bằng phần cứng lẫn khi thực hiện bằng phần mềm. Rất giống với thuật toán ước lượng chuyển động tiêu chuẩn, thuật toán này xử lý một khối hình chữ nhật tại một thời điểm.

Lấy khối 4x4 làm ví dụ. Đơn vị cơ sở trong thiết kế này được gọi là U_PIXEL, như được thể hiện trên Fig.8. Tín hiệu được mã hoá là cờ cho biết xem điểm ảnh tham chiếu đã được mã hoá từ thao tác so khớp chuỗi trước đó hay chưa. Một cách tuỳ ý, tín hiệu vào Cmp[n-1] có thể bị ép buộc bằng “0”, để cho phép loại bỏ cổng “OR” cuối cùng khỏi môđun U_PIXEL.

Bước thứ nhất là xử lý mỗi hàng một cách song song. Mỗi điểm ảnh trong một hàng của hình chữ nhật này được ấn định vào một khối U_PIXEL; đơn vị xử lý này được gọi là U_ROW. Một ví dụ về đơn vị xử lý đối với hàng thứ nhất được thể hiện trên Fig.9.

Cần 4 đơn vị U_ROW để xử lý khối 4x4 này, như được thể hiện trên Fig.10. Đầu ra của nó là mảng cmp[4][4].

Bước tiếp theo là xử lý mỗi cột của mảng cmp một cách song song. Từng cmp trong cột của mảng cmp được xử lý bởi đơn vị xử lý U_COL, như được thể hiện trên Fig.11.

Cần 4 đơn vị U_COL để xử lý khối 4x4 này. Đầu ra của nó là mảng rw[4][4] như được thể hiện trên Fig.12.

Sau đó, số lượng số không trong mỗi hàng của rw[n][0-3] được đếm, và 4 kết quả này được ghi vào mảng r_width[n]. Lưu ý là r_width[n] bằng rwidth[n] ở bước #7. l_width[n] được tạo ra theo cách giống như vậy. Có thể thu được mảng min_width ở bước #7 dưới dạng {{l_width[1], r_width[1]}, { l_width[2], r_width[2]}, { l_width[3], r_width[3]}…}

Kiến trúc phần cứng này có thể được cải biến để phù hợp với hệ thống xử lý song song của CPU/DSP/GPU hiện đại bất kì. Một đoạn mã giả giản lược, để thực hiện nhanh bằng phần mềm, được liệt kê dưới đây.

```
// 1. Generate array C[][]
For(y = 0; y < height; ++y)
{
        For(x = 0; x < width; ++x)
        {
                tmp1 = cur_pixel ^ ref_pixel;
                tmp2 = tmp1[0] | tmp1[1] | tmp1[2] | tmp1[3] | tmp1[4] |
tmp1[5] | tmp1[6] | tmp1[7];
                C[y][x] = tmp2 & (!coded[y][x]);
        }
}
```

```
// 2. Generate array CMP[][]
For(y = 0; y < height; ++y)
{
        CMP[y][0] = C[y][0];
}
For(x = 1; x < width; ++x)
{
        For(y = 0; y < height; ++y)
        {
                CMP[y][x] = C[y][x] | CMP[y][x-1]
        }
}

// 3. Generate array RW[][] or LW[][]
For(x = 0; x < width; ++x)
{
        RW[0][x] = CMP[0][x];
}
For(y = 1; y < height; ++y)
{
        For(x = 0; x < width; ++x)
        {
                RW[y][x] = CMP[y][x] | RW[y-1][x];
        }
}

// 4. Convert RW[][] to R_WIDTH[]
For(y = 0; y < height; ++y)
{
        // đếm số không, hoặc dò số không dẫn đầu
```

```
        R_WIDTH[y] = LZD(RW[y][0], RW[y][1], RW[y][2],
RW[y][3]);
}
```

Không có sự phụ thuộc dữ liệu nào trong mỗi vòng lặp, nên phương pháp xử lý song song bằng phần mềm truyền thống, chẳng hạn loại bỏ vòng lặp, MMX/SSE, có thể được áp dụng để tăng tốc độ thực hiện.

Phương pháp này cũng có thể được áp dụng cho hoạt động tìm kiếm 1D nếu số lượng hàng bị giới hạn bằng một. Một đoạn mã giả giản lược, để thực hiện nhanh bằng phần mềm đối với hoạt động tìm kiếm 1D dựa trên chiều dài cố định, được liệt kê dưới đây.

```
// 1. Generate array C[]
For(x = 0; x < width; ++x)
{
        tmp1 = cur_pixel ^ ref_pixel;
        tmp2 = tmp1[0] | tmp1[1] | tmp1[2] | tmp1[3] | tmp1[4] | tmp1[5] |
tmp1[6] | tmp1[7];
        C[x] = tmp2 & (!coded[x]);
}

// 2. Generate array RW[] or LW[]
If (last "OR" operation in U_PIXEL module is removed)
        Assign RW[] = C[]
Else {
        RW [0] = C[0];
        For(x = 1; x < width; ++x)
        {
                RW [x] = C[x] | RW [x-1]
        }
]
```

```
// 3. Convert RW[][] to R_WIDTH[]
// đếm số không, hoặc dò số không dẫn đầu
If(last "OR" operation in U_PIXEL module is removed)
        R_WIDTH = LZD(RW[0], RW[1], RW[2], RW[3]);
Else
        R_WIDTH[y] = COUNT_ZERO(RW[0], RW[1], RW[2],
RW[3]);
```

Sau khi cả thao tác so khớp 1D và thao tác so khớp 2D được thực hiện, thì max (ld length, 2d (width x height)) được chọn. Nếu lwidth của dãy so khớp 2D là khác không, thì chiều dài của dãy so khớp 1D trước đó (length = length – lwidth) cần được điều chỉnh để tránh sự chồng chéo giữa dãy so khớp 1D trước đó và dãy so khớp 2D hiện tại. Nếu chiều dài của dãy so khớp 1D trước đó trở nên bằng không sau khi điều chỉnh, thì nó được loại bỏ khỏi danh sách so khớp.

Vị trí bắt đầu tiếp theo được tính toán nhờ sử dụng current_location + length nếu thao tác so khớp trước đó là so khớp 1D, hoặc current_location + (lwidth+rwidth) nếu thao tác so khớp trước đó là so khớp 2D. Khi thao tác so khớp 1D được thực hiện, nếu điểm ảnh cần được so khớp bất kì mà nằm trong vùng so khớp 2D trước đó bất kì mà trong đó vị trí của nó đã được bao trùm bởi thao tác so khớp 2D, thì các điểm ảnh tiếp theo sẽ được quét qua cho đến khi tìm thấy vị trí điểm ảnh mà ở đó nó chưa được mã hoá bởi thao tác so khớp trước đó.

Sau khi thu được các cặp khớp này, thì cơ chế entropy được áp dụng để biến đổi các kí hiệu này thành luồng nhị phân. Được nêu làm ví dụ ở đây là ý tưởng về việc sử dụng chế độ ngữ cảnh xác suất ngang bằng. Chế độ ngữ cảnh thích ứng cải tiến cũng có thể được áp dụng để có hiệu quả nén tốt hơn.

```
// loop for each CU, uiTotal=uiWidth*uiHeight, uiIdx=0;
while ( uiIdx < uiTotal) {
        // *pDist: lưu giá trị khoảng cách đối với mỗi cặp khớp
        // *pIdx: lưu trị số chỉ số đối với mỗi cặp khớp
        // *pLen: lưu giá trị độ dài đối với mỗi cặp khớp
```

// encodeEP() và encodeEPs() tái sử dụng HEVC hoặc mã hoá entropy đường vòng tương tự.

```
if (pDist[uiIdx] == -1 )
{
        //mã hoá one-bin bằng mô hình xác suất ngang bằng để chỉ thị
        //cặp hiện tại có khớp hay không.
        unmatchedPairFlag = TRUE;
        encodeEP(unmatchedPairFlag);
        //uiIndexBits được điều khiển bởi kích thước bảng màu
        // tức là, đối với 24 màu khác nhau thì cần 5 bit, đối với 8 màu thì cần 3 bit
        encodeEPs(pIdx[uiIdx], uiIndexBits);
        uiIdx++;
}
else
{
        unmatchedPairFlag= FALSE;
        encodeEP(unmatchedPairFlag);

        /*bound binarization with max possible value*/
        UInt uiDistBits =0;
        // độ dịch được dùng để bổ sung thêm các tham chiếu từ các khối lân cận
        // ở đây, trước hết cho offset=0;
        while( (1<<uiDistBits)<= (uiIdx+offset))
        {
                uiDistBits++;
        }
        encodeEPs(pDist[uiIdx], uiDistBits);
```

```
                /*bound binarization with max possible value*/
                UInt uiLenBits =0;
                while( (1<<uiLenBits)<= (uiTotal-uiIdx))
                {
                        uiLenBits++;
                }
                encodeEPs(pLen[uiIdx], uiLenBits);
                uiIdx += pLen[uiIdx];
        }
}
```

Đó là quy trình mã hoá đối với mỗi cặp khớp. Một cách tương ứng, tiến trình giải mã đối với cặp khớp là như sau.

```
// loop for each CU, uiTotal=uiWidth*uiHeight, uiIdx=0;
while ( uiIdx < uiTotal) {
        // *pDist: lưu giá trị khoảng cách đối với mỗi cặp khớp
        // *pIdx: lưu trị số chỉ số đối với mỗi cặp khớp
        // *pLen: lưu giá trị độ dài đối với mỗi cặp khớp
        // parseEP() và parseEPs() tái sử dụng HEVC hoặc mã hoá entropy đường vòng tương tự.

        // phân tích cờ cặp không khớp
        parseEP(&uiUnmatchedPairFlag);

        if (uiUnmatchedPairFlag )
        {
                parseEPs( uiSymbol, uiIndexBits );
                pIdx[uiIdx] = uiSymbol;
                uiIdx++;
```

```
        }
        else
        {
                /*bound binarization with max possible value*/
                UInt uiDistBits =0;
                // độ dịch được dùng để bổ sung thêm các tham chiếu từ các
khối lân cận
                // ở đây, trước hết cho offset=0;
                while( (1<<uiDistBits)<= (uiIdx+offset))
                        uiDistBits++;
                UInt uiLenBits =0;
                while( (1<<uiLenBits)<= (uiTotal-uiIdx))
                        uiLenBits++;

                parseEPs( uiSymbol, uiDistBits);
                pDist[uiIdx] = uiSymbol;
                parseEPs( uiSymbol, uiLenBits);
                pLen[uiIdx] = uiSymbol;

                for(UInt i=0; i< pLen[uiIdx]; i++)
                        pIdx[i+uiIdx] = pIdx[i+uiIdx- pDist[uiIdx]];
                uiIdx += pLen[uiIdx];
        }
}
```

Lưu ý rằng chỉ có các điểm ảnh ở vị trí không khớp mới được mã hoá vào luồng bit. Để có mô hình thống kê chính xác hơn, thì chỉ có các điểm ảnh này và các điểm ảnh lân cận của chúng là được sử dụng để trích xuất bảng màu, thay vì tất cả các điểm ảnh trong CU này.

Đối với chỉ số hoặc đầu ra đenta này, thì chúng thường chứa một lượng hạn chế giá trị duy nhất trong chế độ mã hoá nhất định. Sáng chế áp dụng bảng đenta

thứ hai để áp dụng kết quả quan sát này. Bảng đenta này có thể được xây dựng sau khi thu được tất cả dữ liệu nguyên gốc trong CU này, và sẽ được báo hiệu một cách tường minh trong luồng bit. Theo cách khác, nó có thể được xây dựng theo cách thích ứng trong tiến trình mã hoá, nên bảng này không nhất thiết phải được bao gồm trong luồng bit. Cờ delta_color_table_adaptive_flag được xác định đối với sự lựa chọn này.

Sơ đồ cải tiến khác được đề xuất, gọi là Neighboring Delta Color palette table Merge (hợp nhất bảng màu đenta lân cận). Để tạo bảng đenta theo cách thích ứng, thì bộ mã hoá có thể sử dụng bảng đenta từ CU trên đầu hoặc CU bên trái làm điểm bắt đầu ban đầu. Để tạo bảng không theo cách thích ứng, thì bộ mã hoá cũng có thể sử dụng bảng đenta từ CU trên đầu hoặc CU bên trái và so sánh cái giá RD giữa CU trên đầu, CU bên trái và CU hiện tại.

Cờ delta_color_table_merge_flag được xác định để cho biết xem CU hiện tại sử dụng bảng màu đenta từ CU bên trái hay CU bên trên của nó. CU hiện tại mang báo hiệu bảng màu đenta một cách tường minh chỉ khi delta_color_table_adaptive_flag==0 và delta_color_table_merge_flag==0 đồng thời.

Đối với tiến trình hợp nhất, nếu cờ delta_color_table_merge_flag được yêu cầu, thì cờ delta_color_table_merge_direction khác sẽ được xác định để cho biết xem ứng viên hợp nhất xuất phát từ CU phía trên hay CU bên trái.

Một ví dụ về tiến trình mã hoá để tạo bảng đenta theo cách thích ứng được thể hiện như sau. Ở phía giải mã, bất cứ khi nào bộ giải mã nhận được dữ liệu nguyên gốc, thì nó sẽ tái tạo bảng đenta dựa trên các bước ngược lại.

10. Xác định palette_table[] và palette_count[]

11. Khởi tạo palette_table(n) = n (n = 0…255), theo cách khác, có thể sử dụng palette_table[] từ CU trên đầu hoặc CU bên trái làm giá trị ban đầu

12. Khởi tạo palette_count(n) = 0 (n = 0…255), theo cách khác, có thể sử dụng palette_count[] từ CU trên đầu hoặc CU bên trái làm giá trị ban đầu

13. Đối với giá trị đenta c' bất kì:

1) Định vị n sao cho palette_table(n) == đenta c'

2) Dùng n làm chỉ số mới của đenta c'

3) ++palette_count(n)

4) Sắp xếp palette_count[] theo thứ tự giảm dần

5) Sắp xếp palette_table[] theo đó

14. Trở lại bước 1 cho đến khi tất cả đenta c' trong LCU hiện tại đã được xử lý

Đối với khối bất kì mà bao gồm cả văn bản và đồ hoạ, thì cờ mặt nạ được dùng để tách riêng phần văn bản và phần đồ hoạ. Phần văn bản thì được nén bằng phương pháp nêu trên; còn phần đồ hoạ thì được nén bằng phương pháp nén khác.

Lưu ý rằng, do giá trị của điểm ảnh bất kì mà được bao trùm bởi cờ mặt nạ là đã được mã hoá bởi lớp văn bản một cách không suy hao, nên các điểm ảnh trong phần đồ hoạ có thể được coi là "don't-care-pixel" (điểm ảnh không cần quan tâm). Khi phần đồ hoạ được nén, thì giá trị tuỳ ý bất kì có thể được gán cho điểm ảnh không cần quan tâm, để thu được hiệu quả nén tối ưu.

Do phần suy hao có thể được xử lý bằng quá trình trích xuất bảng màu, nên bản đồ chỉ số phải được nén theo cách không suy hao. Điều này cho phép xử lý hiệu quả nhờ sử dụng quá trình so khớp chuỗi 1D hoặc 2D. Theo sáng chế, quá trình so khớp chuỗi 1D hoặc 2D là bị giới hạn ở LCU hiện tại, nhưng cửa sổ tìm kiếm có thể mở rộng quá LCU hiện tại. Cũng cần lưu ý rằng khoảng cách khớp có thể được mã hoá nhờ sử dụng cặp vectơ chuyển động theo chiều ngang và chiều đứng, tức là (MVy=matched_distance/cuWidth, MVy=matched_distance-cuWidth*MVy).

Giả sử rằng hình ảnh có hướng cấu trúc trong không gian khác nhau tại các vùng cục bộ, thì thao tác tìm kiếm 1D có thể được thực hiện theo chiều ngang hoặc chiều đứng bằng cách xác định cờ chỉ thị color_idx_map_pred_direction. Chiều quét chỉ số tối ưu có thể được quyết định dựa trên cái giá R-D. Fig.6 thể hiện các chiều quét, bắt đầu từ vị trí ngay đầu tiên. Tiếp tục được thể hiện là mẫu quét ngang và đứng trên Fig.9. Lấy CU 8x8 làm ví dụ. DeriveMatchPairs() và các bước mã hoá entropy liên quan được thực hiện hai lần đối với cả mẫu quét ngang và mẫu quét đứng. Sau đó, chiều quét cuối cùng được chọn mà có cái giá RD nhỏ nhất.

Nhị phân hoá cải thiện

Như được thể hiện trên Fig.13, bảng màu và cặp thông tin khớp đối với bản đồ chỉ số màu được mã hoá. Chúng được mã hoá bằng phương pháp nhị phân hoá chiều dài cố định. Theo cách khác, phương pháp nhị phân hoá chiều dài biến thiên cũng có thể được sử dụng.

Ví dụ, đối với quá trình mã hoá bảng màu, thì bảng màu có thể có 8 giá trị màu khác nhau. Do đó, nó chỉ chứa 8 chỉ số khác nhau trong bản đồ chỉ số màu. Thay vì dùng 3 bin cố định để mã hoá mỗi trị số chỉ số một cách ngang bằng, thì có thể dùng chỉ một bit để biểu thị điểm ảnh nền, ví dụ, bit 0. Sau đó, phần còn lại của 7 giá trị điểm ảnh sẽ sử dụng từ mã có độ dài cố định, chẳng hạn 1000, 1001, 1010, 1011, 1100, 1101, và 1110 để mã hoá chỉ số màu. Điều này là do việc màu nền có thể chiếm phần trăm lớn nhất nên một từ mã đặc biệt cho nó sẽ tiết kiệm tổng số bin. Trường hợp này thường xảy ra đối với nội dung trên màn hình. Lấy CU 16x16 làm ví dụ, đối với quá trình nhị phân hoá 3-bin cố định, thì cần 3x16x16=768 bin. Ngoài ra, lấy chỉ số 0 làm màu nền, chiếm 40%, còn các màu khác thì được phân bố đồng đều. Trong trường hợp này, chỉ cần 2,8x16x16<768 bin.

Đối với quá trình mã hoá cặp khớp, thì trị số max có thể được dùng để giới hạn quá trình nhị phân hoá của nó, có tính đến khả năng thực hiện bị ràng buộc của phương pháp này trong vùng của CU hiện tại. Về mặt toán học, khoảng cách và chiều dài khớp có thể là 64x64=4K trong mỗi trường hợp. Tuy nhiên, điều này sẽ không xảy ra cùng nhau. Đối với mỗi vị trí khớp, thì khoảng cách khớp bị hạn chế bởi khoảng cách giữa vị trí hiện tại và vị trí ngay đầu tiên trong bộ đệm tham chiếu (chẳng hạn vị trí thứ nhất trong CU hiện tại), ví dụ L. Do đó, các bin lớn nhất đối với quá trình nhị phân hoá khoảng cách này là log2(L)+1 (thay vì chiều dài cố định), và các bin lớn nhất đối với quá trình nhị phân hoá chiều dài là log2(cuSize-L)+1 với cuSize=cuWidth*cuHeight.

Ngoài bảng màu và bản đồ chỉ số, thì quá trình mã hoá phần dư có thể được cải thiện đáng kể nhờ phương pháp nhị phân hoá khác. Đối với phiên bản HEVC REext và HEVC, thì hệ số biến đổi là nhị phân hoá bằng các mã có độ dài biến thiên với giả sử rằng độ lớn của phần dư là nhỏ sau khi dự đoán, biến đổi và

lượng tử hoá. Tuy nhiên, sau khi áp dụng phương pháp bỏ qua biến đổi, đặc biệt là để bỏ qua biến đổi đối với nội dung trên màn hình với màu khác biệt, thì thường tồn tại các phần dư có trị số lớn hơn và ngẫu nhiên (không gần với trị số nhỏ hơn tương đối "1", "2", "0"). Nếu phương pháp nhị phân hoá các hệ số HEVC hiện tại được sử dụng, thì sẽ tạo ra từ mã rất dài. Theo cách khác, việc sử dụng phương pháp nhị phân hoá chiều dài cố định sẽ tiết kiệm độ dài mã cho các phần dư được tạo ra bởi chế độ mã hoá bảng màu và chỉ số.

Lấy mẫu sắc độ thích ứng đối với nội dung hỗn hợp

Phần nêu trên đã mô tả các kĩ thuật khác nhau để mã hoá nội dung trên màn hình với hiệu quả cao trong khuôn khổ HEVC/HEVC-RExt. Trên thực tế, ngoài nội dung trên màn hình đơn thuần (chẳng hạn văn bản, đồ hoạ) hoặc viđeo tự nhiên đơn thuần, thì còn có nội dung chứa cả các thứ trên màn hình lẫn viđeo tự nhiên được quay bằng camera - được gọi là nội dung hỗn hợp. Hiện tại, nội dung hỗn hợp được xử lý bằng phương pháp lấy mẫu sắc độ 4:4:4. Tuy nhiên, đối với phần viđeo tự nhiên được quay bằng camera lẫn trong nội dung hỗn hợp này, thì phương pháp lấy mẫu sắc độ 4:2:0 có thể là đủ để cho chất lượng không suy hao về mặt cảm nhận. Điều này là do hệ thống thị giác của con người kém nhạy cảm với những sự thay đổi trong không gian của các thành phần sắc độ hơn so với những sự thay đổi của các thành phần độ chói. Do đó, phương pháp lấy mẫu con thường được thực hiện đối với phần sắc độ (ví dụ, định dạng viđeo 4:2:0 phổ biến) để giảm tốc độ bit đáng kể trong khi vẫn giữ được cùng chất lượng tái tạo.

Sáng chế đề xuất cờ mới (tức enable_chroma_subsampling) mà được xác định và được báo hiệu tại cấp độ CU một cách đệ quy. Đối với mỗi CU, bộ mã hoá sẽ xác định xem CU đó đang được mã hoá theo định dạng 4:2:0 hay 4:4:4 theo cái giá về tốc độ-độ méo.

Fig.14A và Fig.14B thể hiện các định dạng lấy mẫu sắc độ 4:2:0 và 4:4:4.

Ở phía bộ mã hoá, đối với mỗi CU, giả sử đầu vào là nguồn 4:4:4 nêu trên, thì cái giá về tốc độ-độ méo được trích ra trực tiếp nhờ sử dụng quy trình mã hoá 4:4:4 với cờ enable_chroma_subsampling = 0 hoặc FALSE. Sau đó, tiến trình sẽ lấy mẫu con đối với các mẫu 4:4:4 thành 4:2:0 để trích xuất mức tiêu thụ bit của

nó. Định dạng 4:2:0 tái tạo được được nội suy trở lại định dạng 4:4:4 để đo độ méo (nhờ sử dụng SSE/SAD). Cùng với mức tiêu thụ bit, cái giá về tốc độ-độ méo được trích ra khi mã hoá CU tại không gian 4:2:0 và so sánh nó với cái giá khi mã hoá CU tại không gian 4:4:4. Phương pháp mã hoá nào mà có cái giá về tốc độ-độ méo thấp hơn sẽ được chọn cho quá trình mã hoá cuối cùng.

Fig.15 thể hiện tiến trình nội suy từ 4:4:4 sang 4:2:0 và ngược lại. Tiến trình chuyển đổi định dạng lấy mẫu màu viđeo này thường cần đến một lượng lớn bộ lọc nội suy.

Để giảm bớt sự phức tạp khi thực hiện, thì bộ lọc nội suy HEVC (tức là DCT-IF) có thể được sử dụng. Như được thể hiện trên Fig.15, "ô hình vuông" biểu thị các mẫu 4:4:4 ban đầu. Từ 4:4:4 sang 4:2:0, các điểm ảnh half-pel ("hình tròn") được nội suy bằng DCT-IF theo chiều đứng đối với các thành phần sắc độ. Các vị trí quarter-pel ("hình thoi") cũng được thể hiện nhằm mục đích minh hoạ. "Các hình tròn" có bóng màu xám được lấy để tạo thành các mẫu 4:2:0. Đối với quá trình nội suy từ 4:2:0 sang 4:4:4, tiến trình bắt đầu với "các hình tròn" màu xám trong các thành phần sắc độ, các vị trí half-pel được nội suy theo chiều ngang để thu được tất cả "các hình tròn," và sau đó "ô hình vuông" được nội suy nhờ sử dụng DCT-IF theo chiều đứng. Tất cả "ô hình vuông" nội suy được được chọn để tạo thành nguồn 4:4:4 được tái tạo.

Điều khiển bộ mã hoá

Như được bộc lộ trong các phần trước đây, các cờ đã được mô tả để điều khiển quá trình xử lý cấp thấp. Ví dụ, cờ enable_packed_component_flag được dùng để cho biết xem CU hiện tại sử dụng định dạng được đóng gói hay định dạng phẳng thông thường của nó cho tiến trình mã hoá. Việc có cho phép định dạng được đóng gói hay không có thể phụ thuộc vào cái giá R-D tính được tại bộ mã hoá. Để thực hiện bộ mã hoá thực tế, có thể đạt được giải pháp có độ phức tạp thấp bằng cách phân tích biểu đồ của CU và tìm ngưỡng tốt nhất để quyết định, như được thể hiện trên Fig.3.

Kích thước của bảng màu có ảnh hưởng trực tiếp đến độ phức tạp. maxColorNum được đưa vào để kiểm soát sự đánh đổi giữa độ phức tạp và hiệu

quả mã hoá. Cách đơn giản nhất là chọn phương pháp nào mà có cái giá R-D thấp nhất.

Chiều mã hoá bản đồ chỉ số có thể được xác định bằng cách tối ưu hoá R-D, hoặc sử dụng hướng trong không gian cục bộ (chẳng hạn ước lượng chiều dựa trên toán tử sobel).

Sáng chế giới hạn tiến trình này trong mỗi CTU/CU. Trên thực tế, sự ràng buộc này có thể được tháo gỡ. Ví dụ, đối với quá trình xử lý bản đồ chỉ số màu, thì bộ đệm đường từ CU phía trên và CU bên trái của nó có thể được sử dụng, như được thể hiện trên Fig.16. Với bộ đệm phía trên và bên trái, hoạt động tìm kiếm có thể được mở rộng để tiếp tục cải thiện hiệu quả mã hoá. Giả sử rằng các bộ đệm phía trên/bên trái là được tạo ra nhờ sử dụng các điểm ảnh được tái tạo từ các CU lân cận, thì các điểm ảnh này (cũng như các chỉ số tương ứng của nó) là khả dụng để tham chiếu trước khi xử lý bản đồ chỉ số CU hiện tại. Ví dụ, sau khi sắp xếp lại, thì bản đồ chỉ số CU hiện tại có thể là 14, 14, 14,... 1, 2, 1 (dưới dạng biểu thị 1D). Nếu không có tham chiếu bộ đệm đường, thì số "14" đầu tiên sẽ được mã hoá dưới dạng cặp không khớp. Tuy nhiên, nếu có bộ đệm đường lân cận, thì quá trình so khớp chuỗi có thể bắt đầu tại điểm ảnh ngay đầu tiên, như được thể hiện dưới đây (các mẫu quét ngang và đứng cũng được thể hiện).

Cú pháp bộ giải mã

Thông tin sau đây có thể được sử dụng để mô tả bộ giải mã trên Fig.2. Cú pháp theo sáng chế có tuân theo bản dự thảo cấp ủy ban của HEVC RExt.

7.3.5.8 Cú pháp đơn vị mã hoá:

coding_unit(x0, y0, log2CbSize) {	Descriptor
if(transquant_bypass_enabled_flag)	
cu_transquant_bypass_flag	ae(v)
if(slice_type != I)	
cu_skip_flag[x0][y0]	ae(v)
nCbS = (1 << log2CbSize)	
if(cu_skip_flag[x0][y0])	
prediction_unit(x0, y0, nCbS, nCbS)	
else {	
if(intra_block_copy_enabled_flag)	
intra_bc_flag[x0][y0]	ae(v)
if(color_table_enabled_flag)	
color_table_flag[x0][y0]	ae(v)
if(delta_color_table_enabled_flag)	
delta_color_table_flag[x0][y0]	ae(v)
if(!intra_bc_flag[x0][y0]) {	
if(slice_type != I)	
pred_mode_flag	ae(v)
if(CuPredMode[x0][y0] != MODE_INTRA || log2CbSize == MinCbLog2SizeY)	
part_mode	ae(v)
}	
if(CuPredMode[x0][y0] == MODE_INTRA) {	
if(PartMode == PART_2Nx2N && pcm_enabled_flag && !intra_bc_flag log2CbSize >= Log2MinIpcmCbSizeY && log2CbSize <= Log2MaxIpcmCbSizeY)	
pcm_flag[x0][y0]	ae(v)
if(pcm_flag[x0][y0]) {	
while(!byte_aligned( ))	

pcm_alignment_zero_bit	f(1)
pcm_sample(x0, y0, log2CbSize)	
} else if(intra_bc_flag[x0][y0]) {	
mvd_coding(x0, y0, 2)	
} else if(color_table_flag[x0][y0] || delta_color_table_flag[x0][y0]) {	
enable_packed_component_flag	ae(v)
if(color_table_flag[x0][y0]) {	
color_table_merge_flag	ae(v)
if (color_table_merge_flag){	
color_table_merge_idx	ae(v)
}else{	
color_table_size	ae(v)
for(i=0;i< color_table_size;i++)	
color_table_entry[i]	ae(v)
}	
color_idx_map_pred_direction	ae(v)
}	
if(delta_color_table_flag[x0][y0]) {	
delta_color_table_adaptive_flag	ae(v)
delta_color_table_merge_flag	ae(v)
if (delta_color_table_merge_flag){	
delta_color_table_merge_idx	ae(v)
}else if (!delta_color_table_adaptive_flag){	
delta_color_table_size	ae(v)
for(i=0;i< delta_color_table_size;i++)	
delta_color_table_entry[i]	ae(v)
}	
}	

Pos=0; cuWidth=1<<log2CbSize;
cuHeight=1<<log2CbSize;
while (Pos<cuWidth*cuHeight){	

matched_flag	ae(v)
if(matched_flag) {	
matched_distance /*MVx, MVy*/	ae(v)
matched_length	ae(v)
}else{	
index_delta	ae(v)
}	
}	
} else {	
.....pbOffset = (PartMode == PART_NxN) ? (nCbS / 2) : nCbS	
....	

Fig.17 là hình vẽ thể hiện thiết bị và các phương pháp/lưu đồ được áp dụng cho HEVC hiện tại.

Các phương pháp/lưu đồ và các thiết bị nêu trên có thể được kết hợp vào mạng truyền thông không dây hoặc có dây, hoặc tổ hợp mạng truyền thông không dây và có dây, và được thực hiện trong các thiết bị, chẳng hạn các thiết bị được mô tả dưới đây, và trong các hình vẽ dưới đây:

Fig.18 thể hiện một ví dụ về hệ thống truyền thông 100 có sử dụng báo hiệu để hỗ trợ các bộ thu không dây cải tiến theo sáng chế. Nói chung, hệ thống 100 này cho phép những người dùng không dây truyền và nhận dữ liệu và nội dung khác. Hệ thống 100 này có thể thực hiện một hoặc nhiều phương pháp truy cập kênh, chẳng hạn CDMA (Code Division Multiple Access - đa truy cập phân chia theo mã), TDMA (Time Division Multiple Access - đa truy cập phân chia theo thời gian), FDMA (Frequency Division Multiple Access - đa truy cập phân chia

theo tần số), OFDMA (Orthogonal FDMA - FDMA trực giao), hoặc SC-FDMA (Single Carrier FDMA - FDMA đơn sóng mang).

Theo ví dụ này, hệ thống truyền thông 100 bao gồm thiết bị người dùng (User Equipment - UE) 110a-110c, các mạng truy cập vô tuyến (Radio Access Network - RAN) 120a-120b, mạng lõi 130, mạng điện thoại chuyển mạch công cộng (Public Switched Telephone Network - PSTN) 140, mạng Internet 150, và các mạng 160 khác. Mặc dù các thành phần hoặc các phần tử này được thể hiện với một số lượng nhất định trên Fig.18, nhưng số lượng bất kì các thành phần hoặc các phần tử này có thể được bao gồm trong hệ thống 100.

Các UE 110a-110c được tạo cấu hình để hoạt động và/hoặc truyền thông trong hệ thống 100. Ví dụ, các UE 110a-110c được tạo cấu hình để truyền và/hoặc nhận các tín hiệu không dây hoặc các tín hiệu dùng dây. Mỗi UE 110a-110c biểu thị thiết bị người dùng cuối phù hợp bất kì và có thể bao gồm các thiết bị như (hay có thể được gọi là) thiết bị người dùng (UE), bộ phát/bộ thu không dây (Wireless Transmit/Receive Unit - WTRU), trạm di động, đơn vị thuê bao cố định hoặc di động, máy nhắn tin, điện thoại di động, máy trợ giúp cá nhân kĩ thuật số (Personal Digital Assistant - PDA), điện thoại thông minh, máy tính xách tay, bảng cảm ứng, bộ cảm biến không dây, hoặc các thiết bị điện tử dân dụng.

Các RAN 120a-120b ở đây lần lượt bao gồm các trạm gốc 170a-170b. Mỗi trạm gốc 170a-170b được tạo cấu hình để giao tiếp không dây với một hoặc nhiều trong số các UE 110a-110c để cho phép truy cập vào mạng lõi 130, PSTN 140, mạng Internet 150, và/hoặc các mạng 160 khác. Ví dụ, các trạm gốc 170a-170b có thể bao gồm (hoặc là) một hoặc nhiều trong số các thiết bị đã biết, chẳng hạn trạm thu phát gốc (Base Transceiver Station - BTS), nút-B (NodeB), NodeB cải tiến (evolved NodeB - eNodeB), NodeB thường trú, eNodeB thường trú, bộ điều khiển trạm, điểm truy cập (Access Point - AP), hoặc bộ định tuyến không dây, hoặc máy chủ, bộ định tuyến, chuyển mạch, hoặc thực thể xử lý khác có mạng dùng dây hoặc không dây.

Theo phương án được thể hiện trên Fig.18, trạm gốc 170a tạo thành một phần của RAN 120a, mà có thể bao gồm các trạm gốc, các phần tử, và/hoặc các thiết

bị khác. Trạm gốc 170b cũng tạo thành một phần của RAN 120b, mà có thể bao gồm các trạm gốc, các phần tử, và/hoặc các thiết bị khác. Mỗi trạm gốc 170a-170b hoạt động để truyền và/hoặc nhận các tín hiệu không dây trong vùng miền hoặc khu vực địa lý cụ thể, đôi khi được gọi là "tế bào". Theo một số phương án, công nghệ MIMO (Multiple-Input Multiple-Output - đa đầu vào đa đầu ra) có thể được sử dụng để cung cấp nhiều bộ thu phát cho mỗi tế bào.

Các trạm gốc 170a-170b truyền thông với một hoặc nhiều trong số các UE 110a-110c qua một hoặc nhiều giao diện không gian 190 nhờ sử dụng các liên kết truyền thông không dây. Các giao diện không gian 190 này có thể sử dụng công nghệ truy cập vô tuyến phù hợp bất kì.

Dự tính rằng hệ thống 100 có thể sử dụng chức năng truy cập đa kênh, bao gồm sơ đồ như đã mô tả trên đây. Theo các phương án thực hiện cụ thể, các trạm gốc và các UE thực hiện hệ thống LTE, LTE-A, và/hoặc LTE-B. Tất nhiên là các phương thức đa truy cập và các giao thức không dây khác cũng có thể được sử dụng.

Các RAN 120a-120b truyền thông với mạng lõi 130 để cung cấp cho các UE 110a-110c dịch vụ thoại, dữ liệu, ứng dụng, dịch vụ thoại qua giao thức Internet (Voice over Internet Protocol - VoIP), hoặc các dịch vụ khác. Có thể thấy rằng các RAN 120a-120b và/hoặc mạng lõi 130 có thể truyền thông trực tiếp hoặc gián tiếp với một hoặc nhiều RAN khác (không được thể hiện trên hình vẽ). Mạng lõi 130 cũng có thể có chức năng như cổng truy cập cho các mạng khác (chẳng hạn PSTN 140, mạng Internet 150, và các mạng 160 khác). Ngoài ra, một số hoặc tất cả các UE 110a-110c có thể bao gồm chức năng truyền thông với các mạng không dây khác nhau qua các liên kết không dây khác nhau nhờ sử dụng các công nghệ và/hoặc các giao thức không dây khác nhau.

Mặc dù Fig.18 chỉ thể hiện một ví dụ về hệ thống truyền thông, nhưng các thay đổi khác nhau có thể được thực hiện đối với hệ thống trên Fig.18. Ví dụ, hệ thống truyền thông 100 có thể bao gồm số lượng bất kì các UE, trạm gốc, mạng, hoặc các thành phần khác trong cấu hình phù hợp bất kì, và có thể còn bao gồm EPC được thể hiện trên hình vẽ bất kì trong số các hình vẽ ở đây.

Fig.19A và Fig.19B là các hình vẽ thể hiện các thiết bị ví dụ mà có thể thực hiện các phương pháp và ý tưởng theo sáng chế. Cụ thể là, Fig.19A thể hiện một ví dụ về UE 110, và Fig.19B thể hiện một ví dụ về trạm gốc 170. Các thành phần này có thể được sử dụng trong hệ thống 100 hoặc trong hệ thống phù hợp bất kì khác.

Như được thể hiện trên Fig.19A, UE 110 bao gồm ít nhất một đơn vị xử lý 200. Đơn vị xử lý 200 này thực hiện các hoạt động xử lý khác nhau của UE 110. Ví dụ, đơn vị xử lý 200 có thể thực hiện hoạt động mã hoá tín hiệu, xử lý dữ liệu, điều khiển công suất, xử lý vào/ra, hoặc chức năng bất kì khác để cho phép UE 110 hoạt động trong hệ thống 100. Đơn vị xử lý 200 cũng hỗ trợ các phương pháp và nguyên lý đã được mô tả chi tiết hơn trên đây. Mỗi đơn vị xử lý 200 đều bao gồm thiết bị xử lý hoặc thiết bị điện toán phù hợp bất kì được tạo cấu hình để thực hiện một hoặc nhiều hoạt động. Mỗi đơn vị xử lý 200 có thể, ví dụ, bao gồm bộ vi xử lý, bộ vi điều khiển, bộ xử lý tín hiệu số, mảng cổng lập trình được dạng trường, hoặc mạch tích hợp chuyên dụng.

UE 110 cũng bao gồm ít nhất một bộ thu phát 202. Bộ thu phát 202 được tạo cấu hình để điều chế dữ liệu hoặc nội dung khác để truyền bằng ít nhất một ăng ten 204. Bộ thu phát 202 cũng được tạo cấu hình để giải điều chế dữ liệu hoặc nội dung khác mà ít nhất một ăng ten 204 nhận được. Mỗi bộ thu phát 202 bao gồm cấu trúc phù hợp bất kì để tạo ra các tín hiệu để truyền không dây và/hoặc xử lý các tín hiệu nhận được bằng phương pháp không dây. Mỗi ăng ten 204 đều bao gồm kết cấu phù hợp bất kì để truyền và/hoặc nhận các tín hiệu không dây. Một hoặc nhiều bộ thu phát 202 có thể được sử dụng trong UE 110, và một hoặc nhiều ăng ten 204 có thể được sử dụng trong UE 110. Mặc dù được thể hiện dưới dạng một khối chức năng, nhưng bộ thu phát 202 cũng có thể được thực hiện bằng ít nhất một bộ phát và ít nhất một bộ thu riêng biệt.

UE 110 còn bao gồm một hoặc nhiều thiết bị vào/ra 206. Các thiết bị vào/ra 206 tạo thuận lợi cho việc tương tác với người dùng. Mỗi thiết bị vào/ra 206 đều bao gồm cấu trúc phù hợp bất kì để cung cấp thông tin cho hoặc nhận thông tin từ người dùng, chẳng hạn loa, micrô, bàn phím, màn hình, hoặc màn hình cảm ứng.

Ngoài ra, UE 110 bao gồm ít nhất một bộ nhớ 208. Bộ nhớ 208 lưu giữ các lệnh và dữ liệu mà UE 110 sử dụng, tạo ra, hoặc thu được. Ví dụ, bộ nhớ 208 có thể lưu giữ phần mềm hoặc các lệnh phần sụn được thực hiện bởi (các) đơn vị xử lý 200 và dữ liệu được dùng để giảm hoặc triệt tiêu nhiễu trong các tín hiệu đến. Mỗi bộ nhớ 208 đều bao gồm (các) thiết bị lưu trữ khả biến và/hoặc thiết bị lưu trữ bất biến và thiết bị truy hồi phù hợp bất kì. Loại bộ nhớ phù hợp bất kì có thể được sử dụng, chẳng hạn RAM (Random Access Memory - bộ nhớ truy cập ngẫu nhiên), ROM (Read Only Memory - bộ nhớ chỉ đọc), đĩa cứng, đĩa quang, thẻ SIM (Subscriber Identity Module - môđun nhận dạng thuê bao), thanh nhớ, thẻ nhớ SD (Secure Digital - bảo mật kĩ thuật số), v.v..

Như được thể hiện trên Fig.19B, trạm gốc 170 bao gồm ít nhất một đơn vị xử lý 250, ít nhất một bộ phát 252, ít nhất một bộ thu 254, một hoặc nhiều ăng ten 256, và ít nhất một bộ nhớ 258. Đơn vị xử lý 250 thực hiện các hoạt động xử lý khác nhau của trạm gốc 170, chẳng hạn mã hoá tín hiệu, xử lý dữ liệu, điều khiển công suất, xử lý vào/ra, hoặc chức năng bất kì khác. Đơn vị xử lý 250 cũng có thể hỗ trợ các phương pháp và nguyên lý đã được mô tả chi tiết hơn trên đây. Mỗi đơn vị xử lý 250 đều bao gồm thiết bị xử lý hoặc thiết bị điện toán phù hợp bất kì được tạo cấu hình để thực hiện một hoặc nhiều hoạt động. Mỗi đơn vị xử lý 250 có thể, ví dụ, bao gồm bộ vi xử lý, bộ vi điều khiển, bộ xử lý tín hiệu số, mảng cổng lập trình được dạng trường, hoặc mạch tích hợp chuyên dụng.

Mỗi bộ phát 252 đều bao gồm cấu trúc phù hợp bất kì để tạo ra các tín hiệu để truyền không dây đến một hoặc nhiều UE hoặc các thiết bị khác. Mỗi bộ thu 254 đều bao gồm cấu trúc phù hợp bất kì để xử lý các tín hiệu nhận được theo đường không dây từ một hoặc nhiều UE hoặc các thiết bị khác. Mặc dù được thể hiện dưới dạng các thành phần riêng biệt, nhưng ít nhất một bộ phát 252 và ít nhất một bộ thu 254 có thể được kết hợp thành một bộ thu phát. Mỗi ăng ten 256 đều bao gồm kết cấu phù hợp bất kì để truyền và/hoặc nhận các tín hiệu không dây. Mặc dù ăng ten chung 256 được thể hiện ở đây dưới dạng được ghép vào cả bộ phát 252 lẫn bộ thu 254, nhưng một hoặc nhiều ăng ten 256 có thể được ghép vào (các) bộ phát 252, và một hoặc nhiều ăng ten 256 riêng biệt có thể được ghép

vào (các) bộ thu 254. Mỗi bộ nhớ 258 đều bao gồm (các) thiết bị lưu trữ khả biến và/hoặc thiết bị lưu trữ bất biến và thiết bị truy hồi phù hợp bất kì.

Các chi tiết khác về các UE 110 và các trạm gốc 170 là đã biết đối với chuyên gia trong lĩnh vực. Do đó, các chi tiết này được lược bỏ ở đây cho gọn.

Có thể có lợi khi diễn giải các định nghĩa của những từ và cụm từ nhất định được sử dụng trong bản mô tả này. Các thuật ngữ "bao gồm" và "gồm", cũng như các dạng của chúng, có nghĩa là sự gồm cả chứ không bị giới hạn. Thuật ngữ "hoặc" có nghĩa là kể cả, tức là và/hoặc. Các cụm từ "được liên kết với" và "được liên kết với chúng", cũng như các dạng của chúng, có nghĩa là bao gồm, được bao gồm trong, nối liền với, chứa, được chứa trong, nối đến hoặc nối với, ghép vào hoặc ghép với, có thể truyền thông với, cùng với, xen kẽ, cạnh nhau, gần với, được gắn kết vào hoặc được gắn kết với, có, có thuộc tính là, v.v..

Mặc dù các phương án nhất định và các phương pháp liên quan chung đã được mô tả, nhưng các phương án biến đổi và hoán vị đối với các phương án và các phương pháp này có thể được người có hiểu biết trung bình trong lĩnh vực kĩ thuật này tạo ra. Do đó, phần mô tả các phương án ví dụ nêu trên không nhằm giới hạn hay ràng buộc sáng chế. Các phương án thay đổi, thay thế, và biến đổi khác cũng có thể được tạo ra mà không nằm ngoài nguyên lý và phạm vi của sáng chế, như được xác định bởi các điểm yêu cầu bảo hộ sau đây.

YÊU CẦU BẢO HỘ

1. Phương pháp mã hóa nội dung màn hình thành luồng bit, phương pháp bao gồm các bước:

lựa chọn bảng màu cho khối mã (coding unit, CU) của nội dung màn hình;

tạo bản đồ chỉ số màu có các chỉ số cho CU bằng cách sử dụng bảng màu được chọn; và

mã hóa bảng màu được chọn và bản đồ chỉ số màu cho CU thành luồng bit, và

trong đó bảng màu được suy ra từ CU lân cận bằng cách sử dụng CU tái tạo trong miền điểm ảnh.

2. Phương pháp theo điểm 1, trong đó phương pháp được xử lý bằng cách sử dụng định dạng màu phẳng hoặc định dạng màu đan xen.

3. Phương pháp theo điểm 2, trong đó phương pháp được xử lý ở cấp độ được chọn từ nhóm: cấp độ CU, cấp độ lát, cấp độ hình ảnh hoặc cấp độ chuỗi.

4. Phương pháp theo điểm 1, trong đó phương pháp còn bao gồm bước tạo bảng màu của CU lân cận dựa trên các điểm ảnh được tái tạo có sẵn, bất kể chiều sâu và kích thước CU.

5. Phương pháp theo điểm 1, trong đó phương pháp còn bao gồm bước tạo bảng màu của CU lân cận, trong đó CU lân cận được mã hóa bằng cách sử dụng chế độ bảng màu.

6. Phương pháp theo điểm 1, trong đó CU lân cận bảng màu được mã hóa bằng cách sử dụng chế độ bảng màu.

7. Phương pháp theo điểm 1, trong đó bảng màu được suy ra trong miền điểm ảnh ở bộ giải mã, trong đó luồng bit được mã hóa được phân tách để tái tạo, cho CU, bảng màu và bản đồ chỉ số màu.

8. Phương pháp theo điểm 4, trong đó giá trị điểm ảnh được suy ra ở mỗi vị trí trong CU bằng cách kết hợp chỉ số màu, và bảng màu.

9. Phương pháp theo điểm 1, trong đó phương pháp còn bao gồm bước phân loại các màu hoặc các giá trị điểm ảnh của CU dựa trên bảng màu được suy ra trước đó cho các chỉ số tương ứng.

10. Phương pháp theo điểm 1, trong đó bảng màu được tạo và sắp xếp theo biểu đồ.

11. Phương pháp theo điểm 1, trong đó mỗi một điểm ảnh của CU được biến đổi thành chỉ số màu trong bảng màu.

12. Phương pháp theo điểm 2, trong đó cờ được định nghĩa cho CU để chỉ báo liệu CU được xử lý bằng cách sử dụng kiểu đóng gói hoặc chế độ phẳng.

13. Phương pháp theo điểm 1, trong đó bảng màu được xử lý bằng cách mã hóa kích thước của bảng màu và mỗi màu trong bảng màu.

14. Phương pháp theo điểm 1, trong đó phương pháp còn bao gồm bước tạo cờ chỉ báo CU sử dụng bảng màu từ CU phía trên hoặc bên trái.

15. Phương pháp theo điểm 1, trong đó bản đồ chỉ số màu được mã hóa bằng cách sử dụng phương pháp so khớp chuỗi được chọn từ nhóm này bao gồm: so khớp chuỗi một chiều (one dimensional, 1-D), so khớp 1-D lai, và so khớp chuỗi hai chiều (two dimensional, 2-D),

trong đó phương pháp so khớp chuỗi được báo hiệu nhờ sử dụng các cặp khớp.

16. Phương pháp theo điểm 15, trong đó phương pháp so khớp chuỗi được thực hiện bằng phương pháp running hash (hàm băm chạy).

17. Phương pháp theo điểm 1, trong đó so khớp chuỗi 2-D được thực hiện trên bản đồ chỉ số màu bằng cách nhận diện vị trí của điểm ảnh hiện tại và vị trí của điểm ảnh tham chiếu trong CU làm điểm khởi đầu.

18. Phương pháp theo điểm 1, trong đó CU có định dạng 4:4:4 được xử lý bằng cách sử dụng định dạng lấy mẫu 4:2:0 được lấy mẫu giảm.

19. Phương pháp theo điểm 18 trong đó định dạng được lấy mẫu giảm được xử lý ở cấp độ được chọn từ nhóm: cấp độ CU, cấp độ mành cắt, cấp độ hình ảnh hoặc cấp độ chuỗi.

20. Bộ xử lý để mã hóa nội dung màn hình thành luồng bit, bộ xử lý được tạo cấu hình để:

lựa chọn bảng màu cho CU của nội dung màn hình;

tạo bản đồ chỉ số màu có các chỉ số cho CU bằng cách sử dụng bảng màu được chọn; và

mã hóa bảng màu được chọn và bản đồ chỉ số màu cho CU thành luồng bit, và

trong đó bảng màu được suy ra từ CU lân cận bằng cách sử dụng CU tái tạo trong miền điểm ảnh.

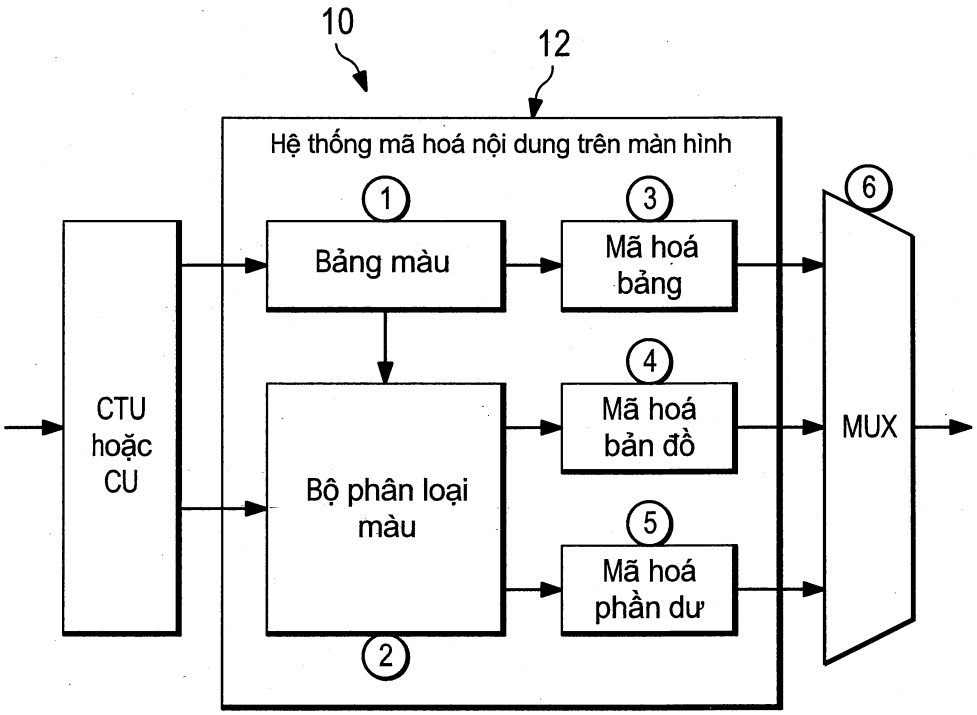

10
12
Hệ thống mã hoá nội dung trên màn hình
1
Bảng màu
3
Mã hoá bảng
4
Mã hoá bản đồ
5
Mã hoá phần dư
2
Bộ phân loại màu
CTU hoặc CU
6
MUX

Fig.1

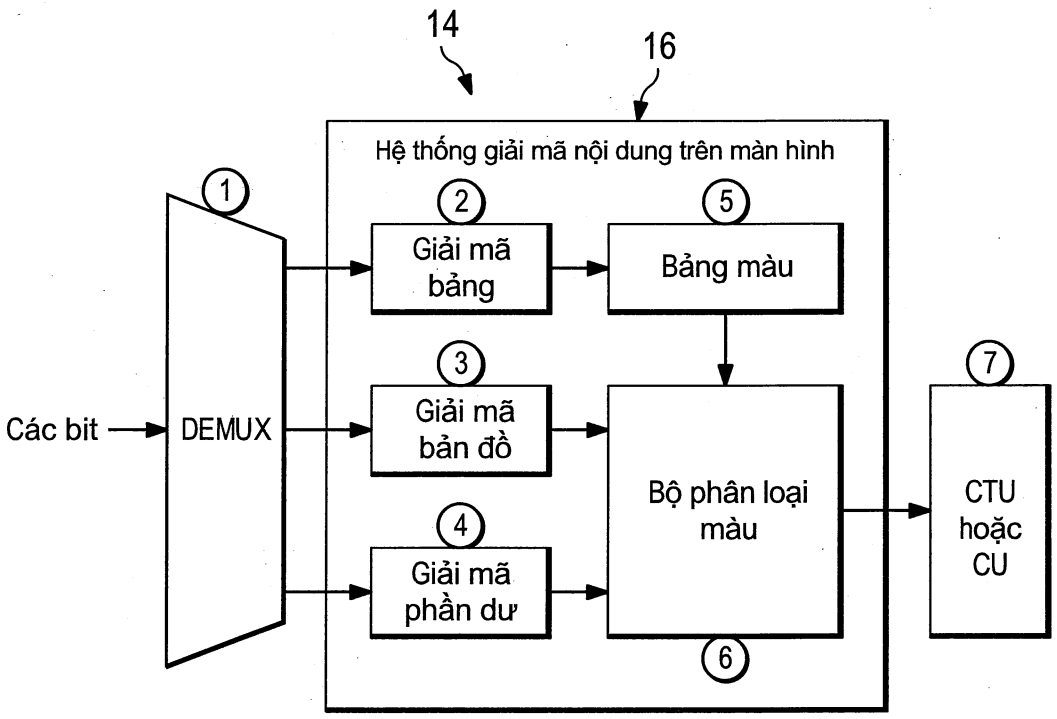

14
16
Hệ thống giải mã nội dung trên màn hình
1
DEMUX
Các bit
2
Giải mã bảng
5
Bảng màu
3
Giải mã bản đồ
4
Giải mã phần dư
Bộ phân loại màu
6
7
CTU hoặc CU

Fig.2

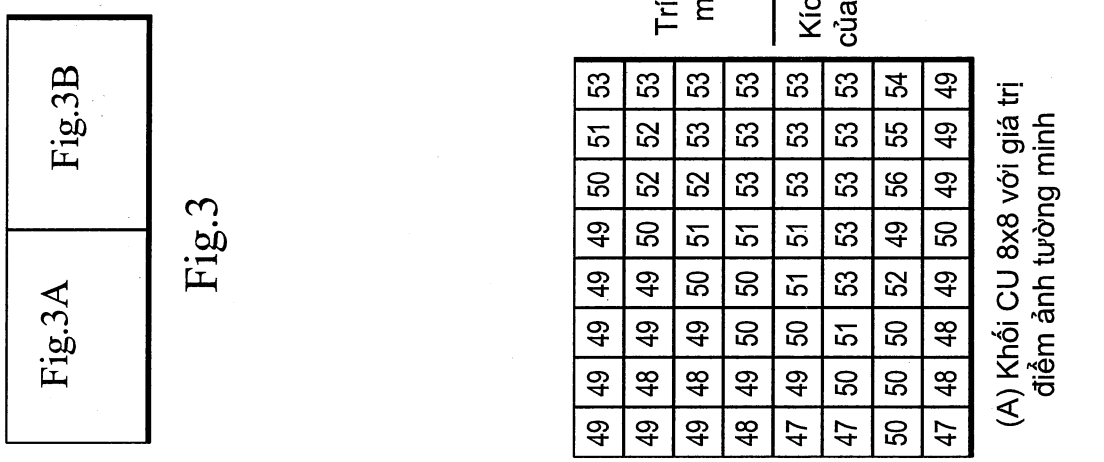


Fig.3A
Fig.3B

Fig.3

ColorTable = {49, 53, 50, 51}
ColorIdx=0 ColorIdx=3
NEAREST NEIGHBOR INDEX MERGE
48 ColorIdx=0, PredResidual = -1
47 ColorIdx=0, PredResidual = -2
52 ColorIdx=1, PredResidual = -1
54 ColorIdx=3, PredResidual = 1
55 ColorIdx=3, PredResidual = 2
56 ColorIdx=3, PredResidual = 3
Trích xuất bản đồ chỉ số màu và phần dư dự đoán
Trích xuất bảng màu dựa trên biểu đồ
Kích thước tối đa của bảng màu là 4
(A) Khối CU 8x8 với giá trị điểm ảnh tượng minh
(B) Trích xuất bảng màu

Fig.3A

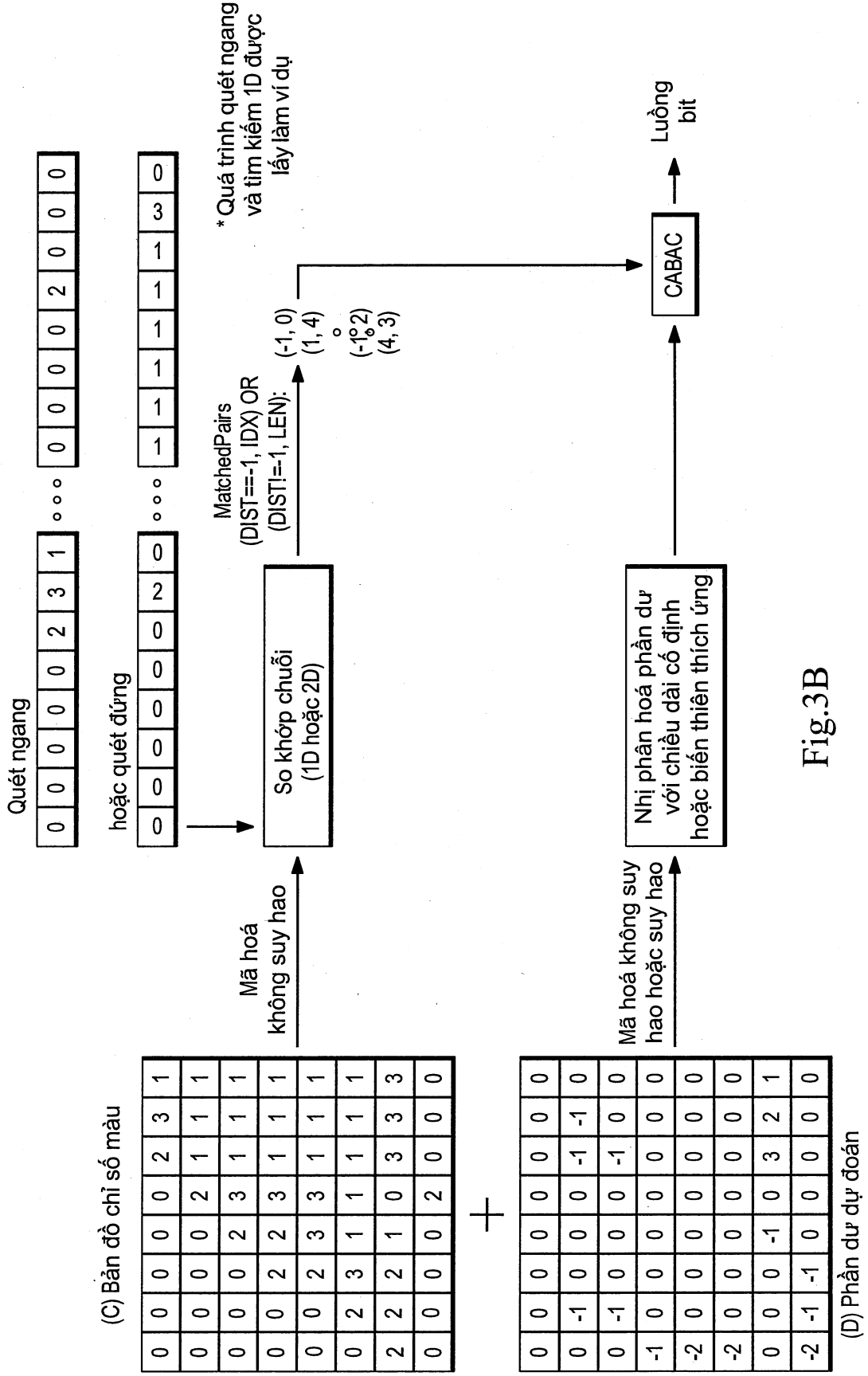

Quét ngang
0 0 0 0 0 2 3 1 ∘∘∘ 0 0 0 0 0 2 0 0 0
hoặc quét đứng
0 0 0 0 0 0 2 0 ∘∘∘ 1 1 1 1 1 1 3 0
So khớp chuỗi (1D hoặc 2D)
MatchedPairs (DIST==-1, IDX) OR (DIST!=-1, LEN):
(-1, 0)
(1, 4)
∘
(-1, 2)
(4, 3)
*Quá trình quét ngang và tìm kiếm 1D được lấy làm ví dụ
Mã hoá không suy hao
Nhị phân hoá phần dư với chiều dài cố định hoặc biến thiên thích ứng
CABAC
Luồng bit
Mã hoá không suy hao hoặc suy hao
(C) Bản đồ chỉ số màu
(D) Phần dư dự đoán

Fig.3B

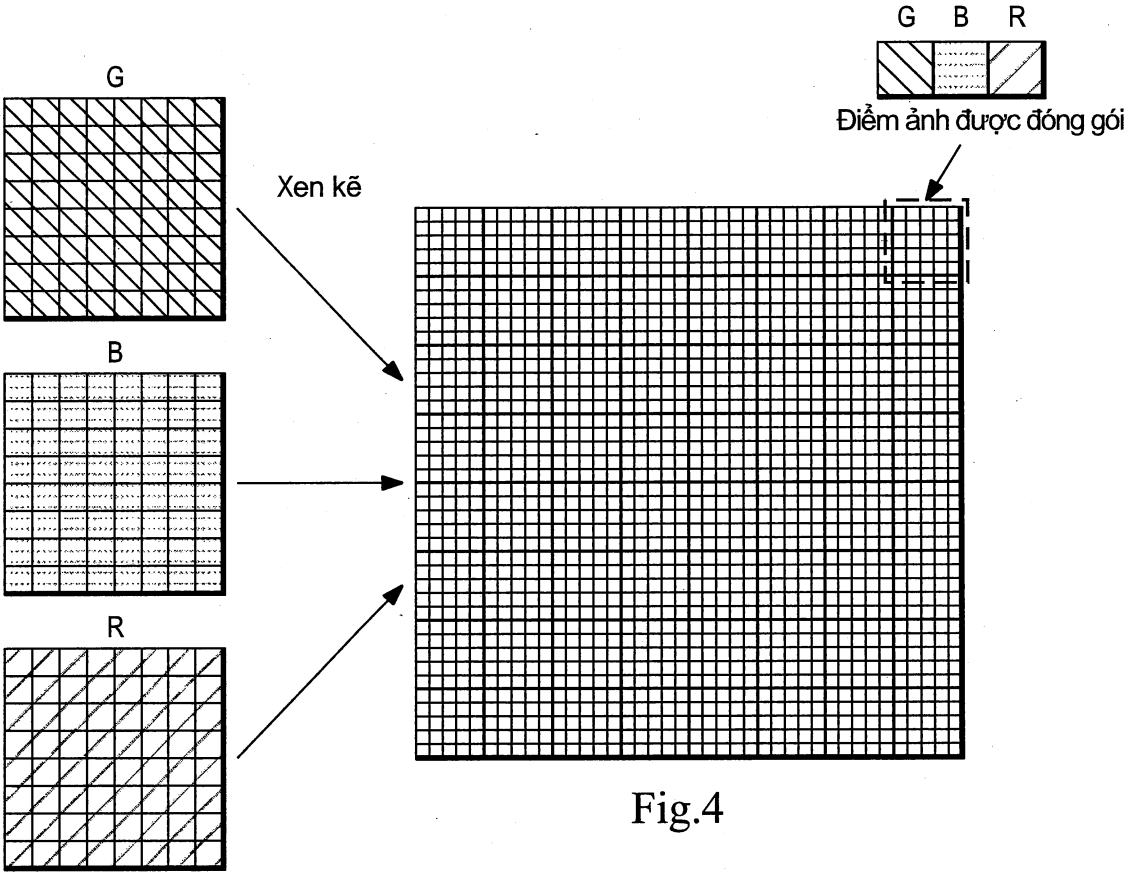

G
B
R
Điểm ảnh được đóng gói
G
B
R
Xen kẽ

Fig.4

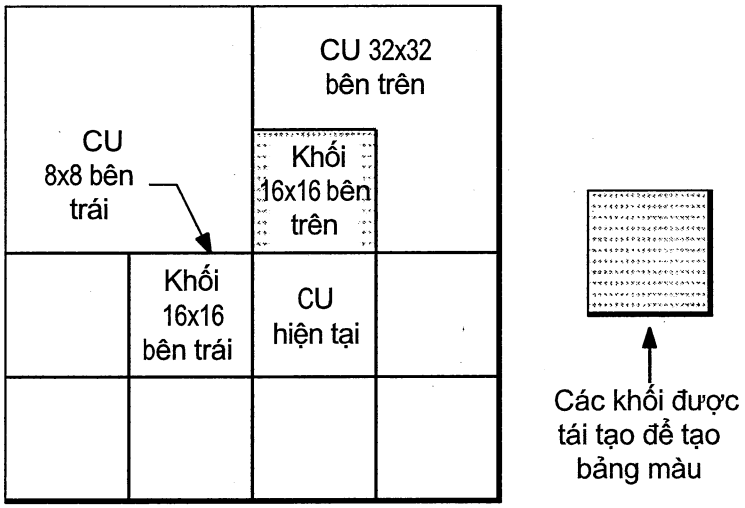

CU 32x32
bên trên
CU
8x8 bên
trái
Khối
16x16 bên
trên
Khối
16x16
bên trái
CU
hiện tại
Các khối được
tái tạo để tạo
bảng màu

Fig.5

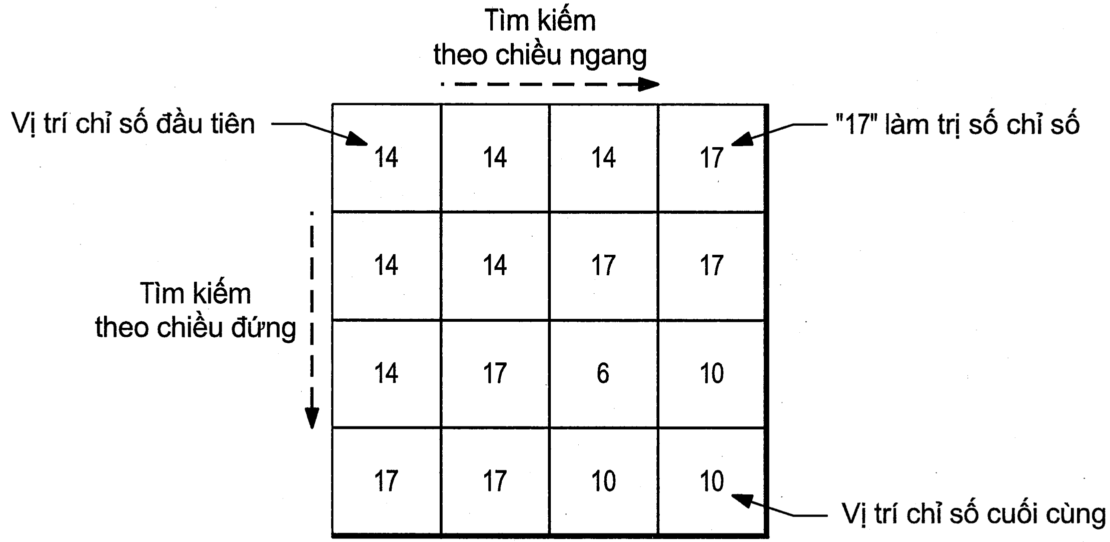

Tìm kiếm
theo chiều ngang
Vị trí chỉ số đầu tiên
"17" làm trị số chỉ số
Tìm kiếm
theo chiều đứng
14 14 14 17
14 14 17 17
14 17 6 10
17 17 10 10
Vị trí chỉ số cuối cùng

Fig.6

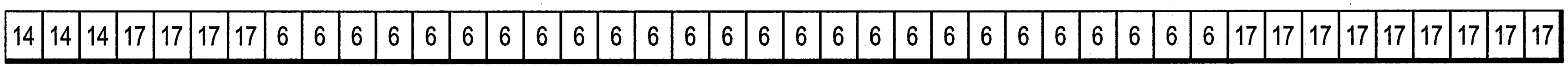

14 14 14 17 17 17 17 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 17 17 17 17 17 17 17 17 17

Fig.7

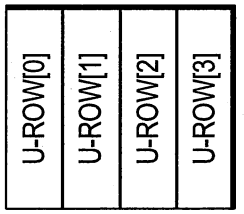


Fig.10

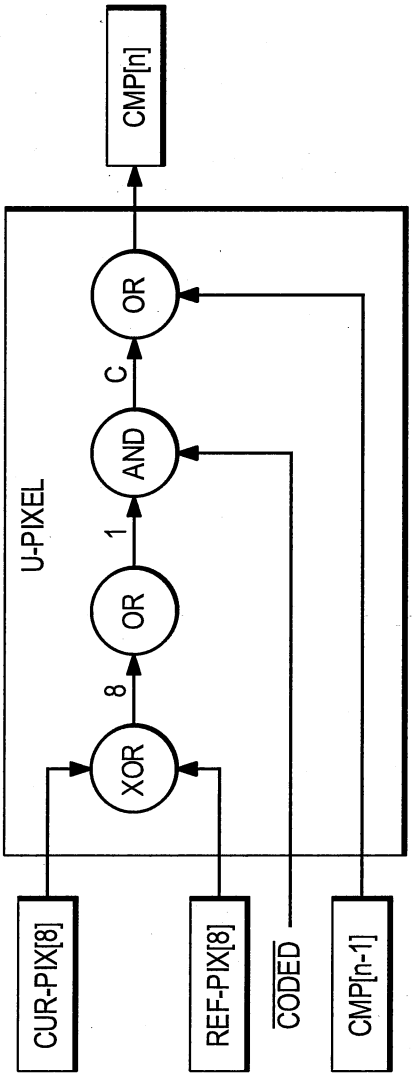


Fig.8

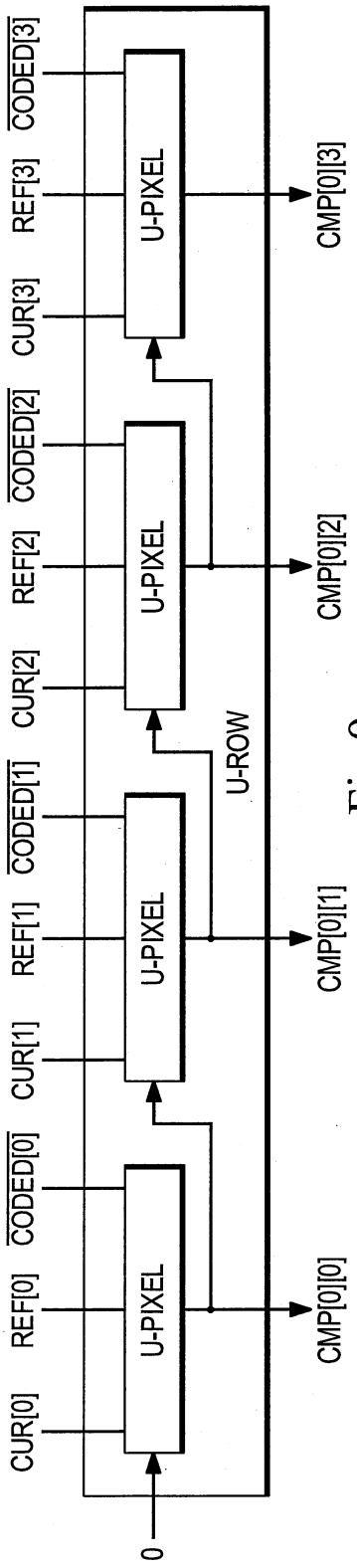


Fig.9

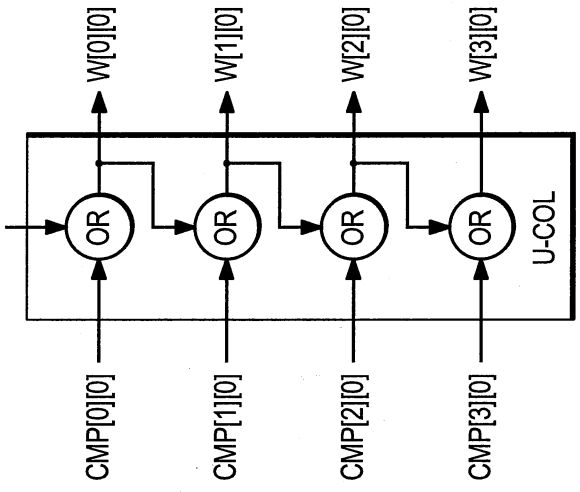


Fig.11

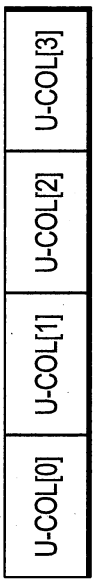


Fig.12

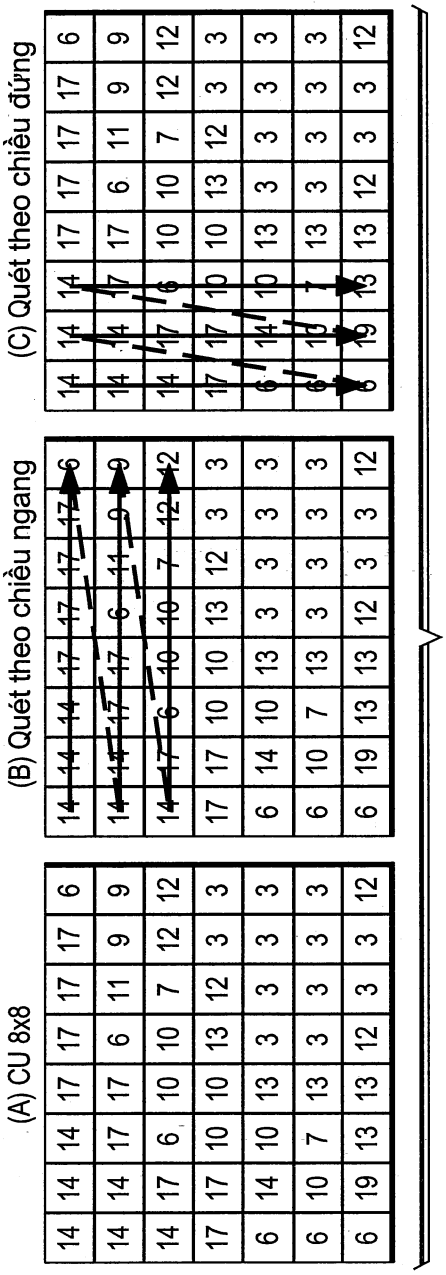


Fig.13

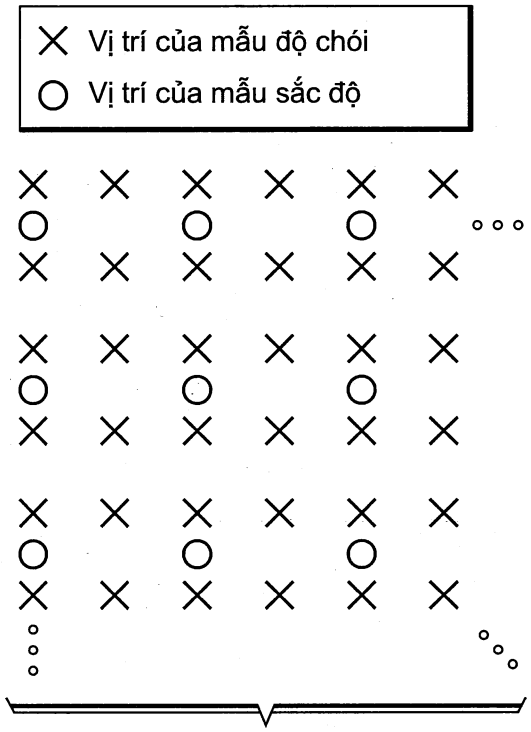


Fig.14A

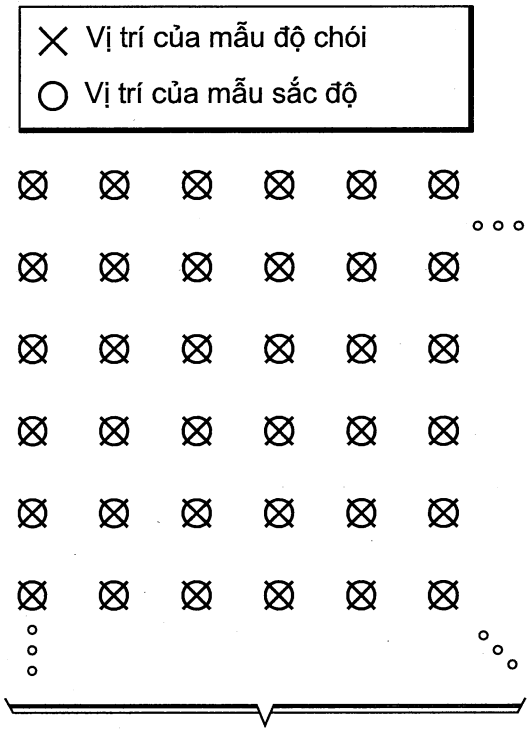


Fig.14B

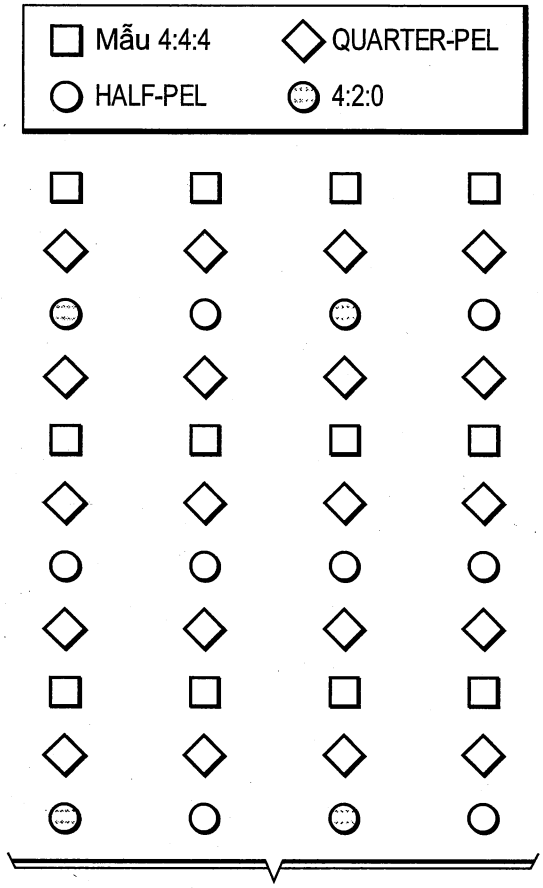

Mẫu 4:4:4
QUARTER-PEL
HALF-PEL
4:2:0

Fig.15

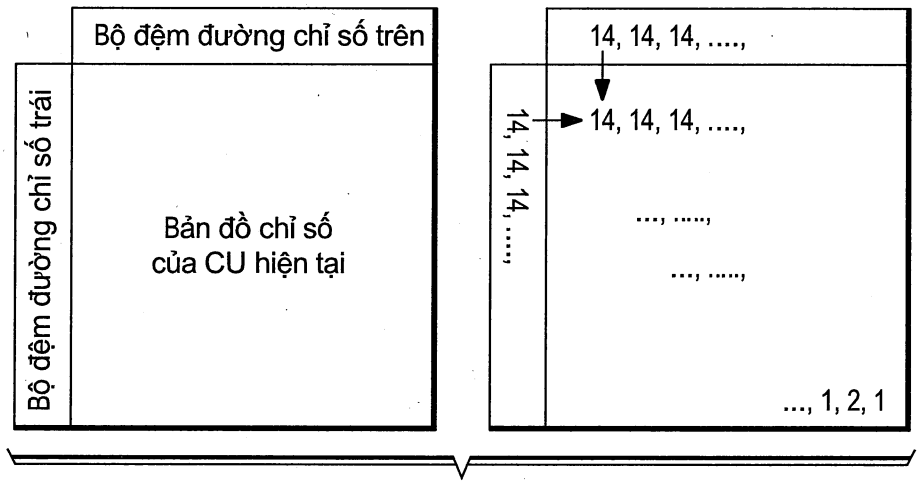

Bộ đệm đường chỉ số trên
Bộ đệm đường chỉ số trái
Bản đồ chỉ số
của CU hiện tại
14, 14, 14, ...,
14, 14, 14, ...,
14, 14, 14, ...,
..., ...,
..., ...,
..., 1, 2, 1

Fig.16

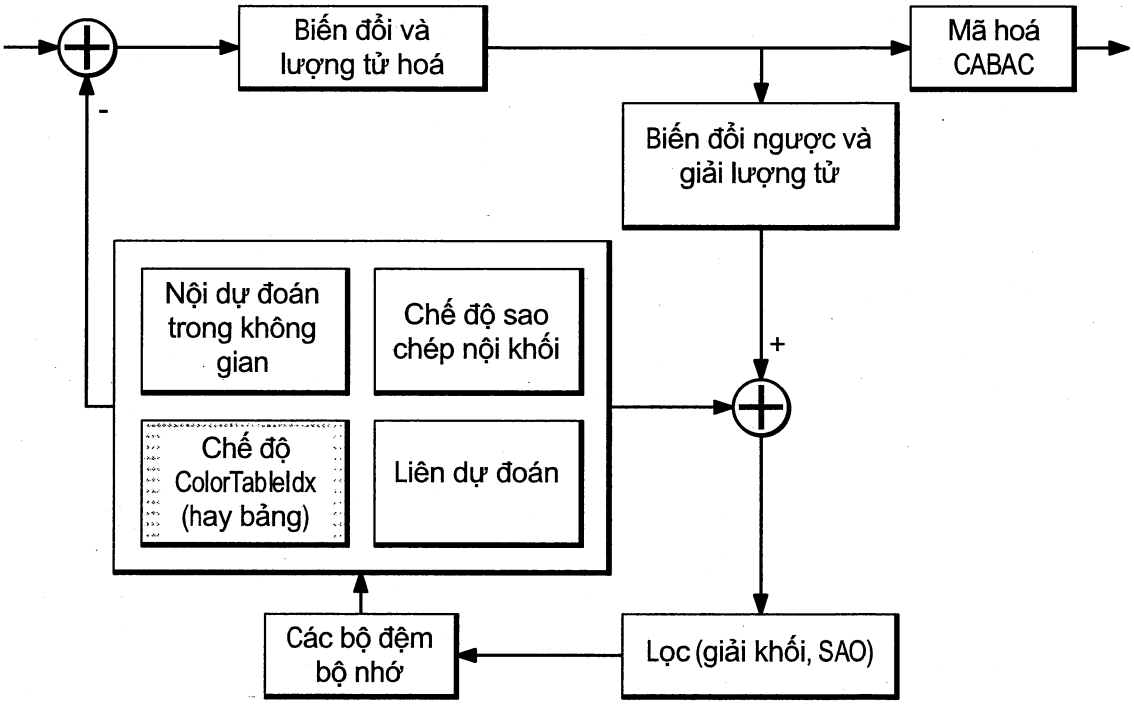

Biến đổi và lượng tử hoá
Mã hoá CABAC
-
Biến đổi ngược và giải lượng tử
Nội dự đoán trong không gian
Chế độ sao chép nội khối
+
Chế độ ColorTableIdx (hay bảng)
Liên dự đoán
Các bộ đệm bộ nhớ
Lọc (giải khối, SAO)

Fig.17

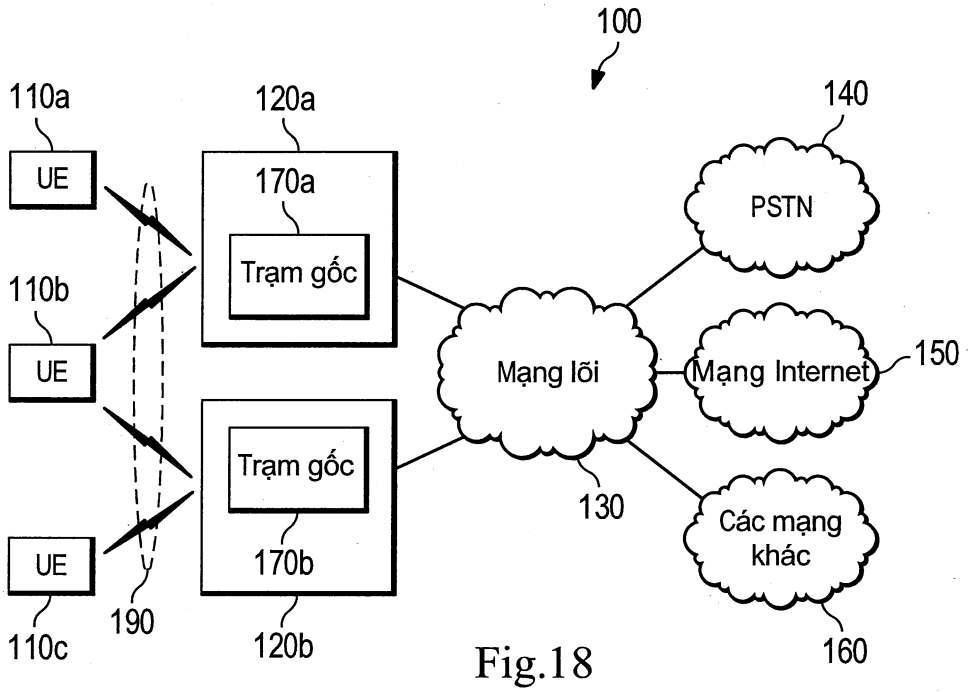

100
110a
UE
110b
UE
UE
110c
190
120a
170a
Trạm gốc
Trạm gốc
170b
120b
Mạng lõi
130
140
PSTN
Mạng Internet
150
Các mạng khác
160

Fig.18

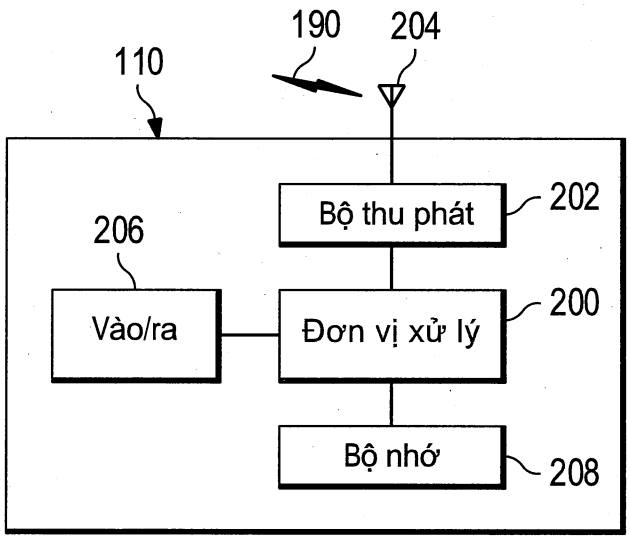


Fig.19A

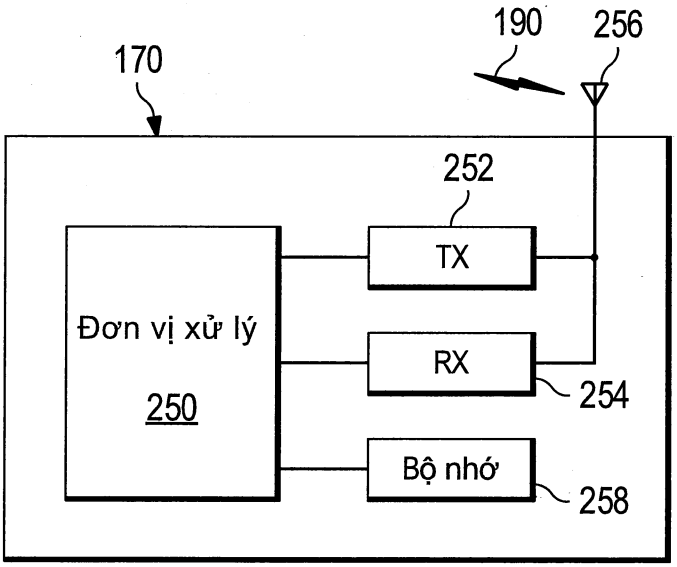


Fig.19B