(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) **1-0042473**

(51)[2020.01] **H04N 19/119**; H04N 19/70; H04N 19/172

(13) **B**

(21) 1-2020-07134
(22) 16/01/2020
(86) PCT/SE2020/050037 16/01/2020
(87) WO2020/149783 23/07/2020
(30) 62/793,353 16/01/2019 US
(45) 27/01/2025 442
(43) 25/08/2021 401
(73) TELEFONAKTIEBOLAGET LM ERICSSON (PUBL) (SE)
16483 Stockholm, Sweden
(72) DAMGHANIAN, Mitra (IR); PETTERSSON, Martin (SE); SJÖBERG, Rickard (SE).
(74) Công ty Luật TNHH T&G (TGVN)

(54) PHƯƠNG PHÁP VÀ THIẾT BỊ ĐỂ GIẢI MÃ HÌNH ẢNH VÀ PHƯƠNG TIỆN LƯU TRỮ CÓ THỂ ĐỌC ĐƯỢC BẰNG MÁY TÍNH

(21) 1-2020-07134

(57) Sáng chế liên quan đến phương pháp để giải mã hình ảnh bao gồm các bước: giải mã thông tin về hình ảnh được phân chia thành nhiều hơn một phân đoạn dựa vào một hoặc nhiều phần tử cú pháp trong luồng bit; giải mã thông tin về sự phân đoạn không gian là đồng đều dựa vào một hoặc nhiều phần tử cú pháp; xác định kích thước đơn vị phân đoạn dựa vào một hoặc nhiều phần tử cú pháp hoặc dựa vào kích thước đơn vị phân đoạn định trước; giải mã giá trị thứ nhất chỉ báo chiều rộng phân đoạn từ một hoặc nhiều từ mã trong luồng bit; giải mã giá trị thứ hai chỉ báo chiều cao phân đoạn từ một hoặc nhiều từ mã; suy ra các chiều rộng cột phân đoạn dựa vào chiều rộng hình ảnh theo số đơn vị phân đoạn và giá trị thứ nhất; suy ra các chiều cao hàng phân đoạn dựa vào chiều cao hình ảnh theo số đơn vị phân đoạn và giá trị thứ hai; suy ra vị trí không gian cho khối hiện tại dựa vào các chiều rộng cột phân đoạn được suy ra và các chiều cao phân đoạn được suy ra; và giải mã khối hiện tại dựa vào vị trí không gian được suy ra. Sáng chế cũng liên quan đến thiết bị giải mã và phương tiện lưu trữ có thể đọc được bởi máy tính.

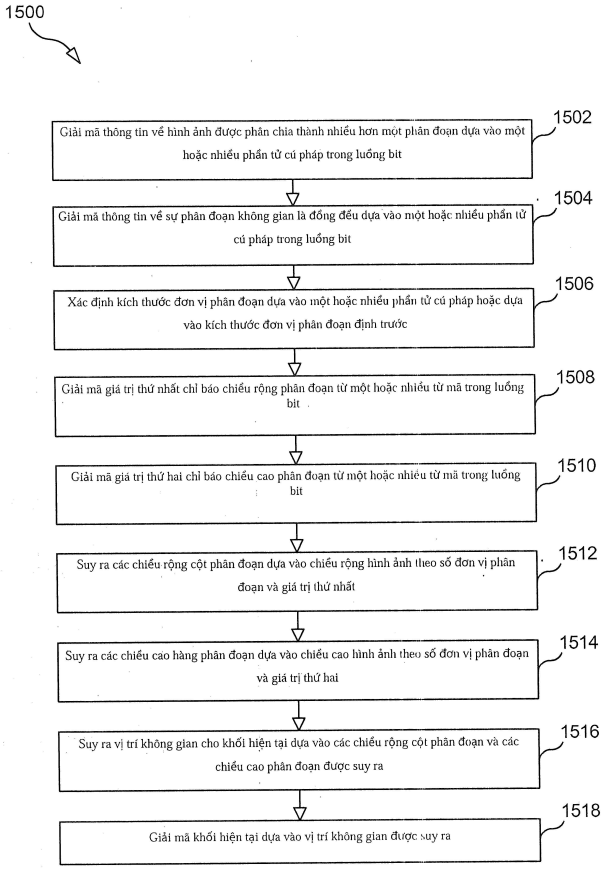


FIG. 15

Lĩnh vực kỹ thuật được đề cập

Sáng chế liên quan đến việc mã hóa viđeo hiệu quả cao (High Efficiency Video Coding - HEVC) và việc mã viđeo đa năng (Versatile Video Coding - VVC).

Tình trạng kỹ thuật của sáng chế

HEVC và mã hóa viđeo hiệu quả cao thế hệ tiếp theo

Chuẩn mã hóa viđeo hiệu quả cao (High Efficiency Video Coding - HEVC), còn được biết đến như. H.265, là cođec viđeo dựa vào khối được chuẩn hóa bởi ITU-T và MPEG mà sử dụng việc dự đoán cả thời gian và không gian. Việc dự đoán không gian đạt được sử dụng việc nội dự đoán (I) từ bên trong hình ảnh hiện tại. Việc dự đoán thời gian được được sử dụng việc liên dự đoán (P) hoặc việc liên dự đoán song hướng (B) ở cấp độ khối từ các hình ảnh tham chiếu được giải mã trước đó. Hiệu giữa dữ liệu điểm ảnh gốc và dữ liệu điểm ảnh được dự đoán, được đề cập đến như phần còn dư, được biến đổi thành miền tần số, được lượng tử hóa và sau đó được mã hóa entropy trước khi được truyền cùng với các tham số dự đoán cần thiết như chế độ dự đoán và các vectơ chuyển động, cũng được mã hóa entropy. Bằng cách lượng tử hóa các phần còn dư được biến đổi, sự cân bằng giữa tốc độ bit và chất lượng của viđeo có thể được kiểm soát. Mức lượng tử hóa được xác định bằng tham số lượng tử hóa (Quantization parameter - QP). Bộ giải mã thực hiện việc giải mã entropy, lượng tử hóa ngược, và biến đổi ngược để thu được phần còn dư. Sau đó, bộ giải mã thêm các phần còn dư vào việc nội dự đoán hoặc liên dự đoán để dựng lại hình ảnh.

MPEG và ITU-T đang làm việc trên phần kế thừa đến HEVC trong đội khai thác viđeo chung (Joint Video Exploratory Team (JVET). Tên của cođec viđeo này đang phát triển là VCC.

Các lát

Khái niệm về các lát trong HEVC chia hình ảnh thành các lát được mã hóa độc lập, trong đó mỗi lát được đọc theo trật tự quét mành theo các đơn vị của các đơn vị cây mã hóa (Coding tree unit - CTU). Các loại mã hóa khác nhau có thể được sử dụng cho các lát của hình ảnh giống nhau, tức là lát có thể hoặc là lát I, lát P hoặc lát B. Mục đích chính của các lát là để cho phép sự tái đồng bộ hóa trong trường hợp mất dữ liệu.

Các ô kiểu xếp ngói

Chuẩn mã hóa viđeo HEVC bao gồm công cụ được gọi là các lát mà chia hình ảnh thành các vùng độc lập về không gian hình chữ nhật. Sử dụng các ô kiểu xếp ngói, hình ảnh trong HEVC có thể được phân chia thành các hàng và các cột của các mẫu trong đó ô kiểu xếp ngói là điểm giao nhau của hàng và cột. Các ô kiểu xếp ngói trong HEVC luôn được sắp thẳng hàng với các biên CTU.

Fig.1 thể hiện ví dụ về sự phân chia ô kiểu xếp ngói sử dụng bốn hàng ô kiểu xếp ngói và 5 cột ô kiểu xếp ngói dẫn đến tổng cộng 20 ô kiểu xếp ngói cho hình ảnh.

Cấu trúc ô kiểu xếp ngói được báo hiệu trong tập hợp tham số hình ảnh (Picture parameter set - PPS) bằng cách đặc tả các chiều dày của các hàng và các chiều rộng của các cột. Các hàng và các cột riêng lẻ có thể có các kích thước khác nhau, nhưng sự phân chia luôn trải rộng ra toàn bộ hình ảnh, lần lượt từ bên trái sang bên phải và từ trên xuống dưới. Không có sự phụ thuộc giải mã nào giữa các ô kiểu xếp ngói của cùng một hình ảnh. Điều này bao gồm việc nội dự đoán, chọn ngữ cảnh để mã hóa entropy và dự đoán vectơ chuyển động. Một ngoại lệ là ở chỗ các phụ thuộc lọc trong vòng nhìn chung được cho phép giữa các ô kiểu xếp ngói.

Cú pháp PPS được sử dụng để đặc tả cấu trúc ô kiểu xếp ngói trong HEVC được liệt kê ở Bảng 1 dưới đây. Cờ, ví dụ, tiles_enabled_flag, chỉ báo việc các ô kiểu xếp ngói được sử dụng hay không. Nếu cờ được thiết đặt, thì số lượng các cột và các hàng ô kiểu xếp ngói được đặc tả. Uniform_spacing_flag là cờ đặc tả việc các chiều rộng cột và các chiều cao hàng được báo hiệu rõ ràng hay việc phương pháp xác định trước để đặt cách

theo cách đồng đều các đường biên ô kiểu xếp ngói nên được sử dụng. Nếu việc báo hiệu rõ ràng được thể hiện thì chiều rộng cột được báo hiệu từng cái một được theo sau bởi các chiều cao hàng. Các chiều rộng cột và các chiều cao hàng được báo hiệu theo các đơn vị CTU. Cờ Loop_filter_across_tiles_enabled_flag đặc tả việc các bộ lọc trong vòng qua các biên ô kiểu xếp nối được mở hay tắt đối với tất cả các biên ô kiểu xếp ngói trong hình ảnh.

Bảng 1. Cú pháp ô kiểu xếp ngói làm ví dụ trong HEVC

pic_parameter_set_rbsp( ) {	**Descriptor**
...	
tiles_enabled_flag	u(1)
...	
if(tiles_enabled_flag) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width_minus1[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height_minus1[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
}	
...	

Ngữ nghĩa học cho việc đặc tả cấu trúc ô kiểu xếp ngói trong HEVC được giải thích chi tiết hơn dưới đây:

tiles_enabled_flag bằng 1 đặc tả rằng có nhiều hơn một ô kiểu xếp ngói trong mỗi hình ảnh tham chiếu đến PPS. tiles_enabled_flag bằng 0 đặc tả rằng chỉ có một ô kiểu xếp ngói trong mỗi hình ảnh tham chiếu đến PPS.

num_tile_columns_minus1 cộng 1 đặc tả rằng số cột ô kiểu xếp ngói phân chia hình ảnh. num_tile_columns_minus1 sẽ là trong phạm vi từ 0 đến PicWidthInCtbsY −

1, gồm cả hai giá trị biên. Nếu không có mặt, thì giá trị của num_tile_columns_minus1 được suy ra là bằng 0.

num_tile_rows_minus1 cộng 1 đặc tả số hàng ô kiểu xếp ngói phân chia hình ảnh. num_tile_rows_minus1 sẽ là ở phạm vi từ 0 đến PicHeightInCtbsY − 1, gồm cả hai giá trị biên. Nếu không có mặt, thì giá trị của num_tile_rows_minus1 được suy ra là bằng 0.

Khi tiles_enabled_flag là bằng 1 thì num_tile_columns_minus1 và num_tile_rows_minus1 sẽ đều không bằng 0.

uniform_spacing_flag bằng 1 đặc tả rằng các biên cột ô kiểu xếp ngói và cũng như vậy các biên hàng ô kiểu xếp ngói được phân bổ đồng đều khắp hình ảnh. uniform_spacing_flag bằng 0 đặc tả rằng các biên cột ô kiểu xếp ngói và cũng như vậy các biên hàng ô kiểu xếp ngói không được phân bổ đồng đều khắp hình ảnh nhưng được báo hiệu rõ ràng sử dụng các phần tử cú pháp column_width_minus1[i] và row_height_minus1[i]. Khi không có mặt, thì giá trị của uniform_spacing_flag được suy ra là bằng 1.

column_width_minus1[i] cộng 1 đặc tả rằng chiều rộng của cột ô kiểu xếp ngói i-th theo các đơn vị của các CTB.

row_height_minus1[i] cộng 1 đặc tả rằng chiều cao của hàng ô kiểu xếp ngói i-th theo các đơn vị của các CTB.

loop_filter_across_tiles_enabled_flag bằng 1 đặc tả rằng các hoạt động vận hành lọc trong vòng có thể được thực hiện khắp các biên ô kiểu xếp ngói trong các hình ảnh đang tham chiếu đến PPS. loop_filter_across_tiles_enabled_flag bằng 0 đặc tả rằng các hoạt động vận hành lọc trong vòng không được thực hiện khắp các biên ô kiểu xếp ngói trong các hình ảnh đang tham chiếu đến PPS. Các hoạt động vận hành lọc trong vòng bao gồm bộ lọc khử khối và các hoạt động vận hành lọc bù thích ứng mẫu. Khi không có mặt, thì giá trị của loop_filter_across_tiles_enabled_flag được suy ra là bằng 1.

VVC được kỳ vọng rằng không sử dụng các lát truyền thống như trong HEVC. Thay vào đó, các ô kiểu xếp ngói được kỳ vọng để đóng vai trò lớn hơn trong VVC do

nhu cầu tăng cho truy cập ngẫu nhiên không gian từ các dịch vụ viđeo bao gồm tạo luồng VR.

Khái niệm về các nhóm ô kiểu xếp ngói được thỏa thuận để được bao gồm trong dự thảo VVC hiện tại tại phiên họp JVET cuối cùng. Nhóm ô kiểu xếp ngói được sử dụng để nhóm nhiều ô kiểu xếp ngói để giảm tổng phí của mỗi ô kiểu xếp.

Nhóm ô kiểu xếp ngói trong dự thảo VVC hiện tại là hình chữ nhật và bao gồm MxN ô kiểu xếp ngói, trong đó M là số ô kiểu xếp ngói theo hướng dọc và N là số ô kiểu xếp ngói theo hướng ngang.

Sự phân chia ô kiểu xếp ngói đồng đều trong HEVC

Sự phân chia ô kiểu xếp ngói HEVC đòi hỏi tất cả các biên được sắp thẳng hàng với lưới CTU. Nghĩa là tất cả các ô kiểu xếp ngói bao gồm các CTU toàn phần và chỉ các CTU chưa đầy đủ được cho phép các ô kiểu xếp ngói là các ô được đặt ở mép bên phải hoặc bên dưới của hình ảnh. Trong HEVC, phần tử cú pháp uniform_spacing_flag bằng 1 đặc tả rằng các biên cột ô kiểu xếp ngói và cũng như vậy các biên hàng ô xếp kiểu ngói được phân bổ đồng đều suốt hình ảnh. Tuy nhiên, sự đồng đều này được giới hạn bởi bậc chi tiết của CTU. Trong HEVC, danh sách colWidth[i] cho i xếp từ 0 đến num_tile_columns_minus1, gồm cả hai giá trị biên, đặc tả rằng chiều rộng của cột ô kiểu xếp ngói i-th theo các đơn vị của các khối cây mã hóa (các CTB) và được suy ra như phương trình sau (A):

```
if( uniform_spacing_flag )
  for( i = 0; i <= num_tile_columns_minus1; i++ )
    colWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbsY ) / ( num_tile_columns_minus1
  + 1 ) – ( i * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 )
```

Phương trình tương tự (B) được sử dụng để xác định các chiều cao của các hàng ô kiểu xếp ngói (rowHeight[i]):

```
if( uniform_spacing_flag )
  for( j = 0; j <= num_tile_rows_minus1; j++ )
    rowHeight[ j ] = ( ( j + 1 ) * PicHeightInCtbsY ) / ( num_tile_rows_minus1
  + 1 ) – ( j * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 )
```

Sự phân chia ô kiểu xếp ngói linh hoạt

Sự phân chia ô kiểu xếp ngói linh hoạt, được giới thiệu trong, được theo sau bởi JVET-L0359, cung cấp chức năng để phân chia hình ảnh thành các ô kiểu xếp ngói phân chia trong đó chiều rộng hoặc chiều cao của mỗi ô kiểu xếp ngói là bội số của kích thước đơn vị tốt hơn so với kích thước CTU. Sự phân chia ô kiểu xếp ngói linh hoạt cho phép sử dụng các CTU chưa đầy đủ ở mép bên phải và bên dưới của mọi ô kiểu xếp ngói (chứ không phải ở mép bên phải và bên dưới của hình ảnh). Các Fig.2A-Fig.2B cung cấp các ví dụ về sự phân đoạn ô kiểu xếp ngói 2x2 sử dụng sự phân chia ô kiểu xếp ngói linh hoạt như trong JVET-L0359 trong đó kích thước đơn vị ô kiểu xếp ngói là một phần tư của kích thước CTU.

Trên các Fig.2A-Fig.2B, các ô kiểu xếp ngói được thể hiện bằng các đường đậm dày và các CTU được thể hiện bằng các đường đen mỏng. Fig.2A thể hiện phương pháp HEVC với 20 CTU trong hình ảnh. Fig.2B thể hiện phương pháp được đề xuất trong JVET-L0359 với 24 CTU trong hình ảnh và kích thước đơn vị ô kiểu xếp ngói bằng một phần tư của kích thước CTU, được thể hiện bởi các đường xám gạch ngang.

Sự phân chia ô kiểu xếp ngói linh hoạt có thể hữu ích cho các ứng dụng như cân bằng tải và viđeo 360° với các kích thước khuôn mặt được mong muốn để không là bội số của kích thước CTU.

Cú pháp và các ngữ nghĩa học từ JVET-L0359 liên quan đến sáng chế này được liệt kê dưới đây, trong đó các phần được in nghiêng (*italicized)* là văn bản được bổ sung được đề xuất trong L0359.

pic_parameter_set_rbsp( ) {	Descriptor
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
transform_skip_enabled_flag	u(1)
tiles_enabled_flag	u(1)
if(tiles_enabled_flag) {	
tile_unit_size_idc	*ue(v)*
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)

uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width_minus1[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height_minus1[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
}	
rbsp_trailing_bits( )	
}	

tile_unit_size_idc đặc tả kích thước của khối đơn vị ô xếp kiểu ngói trong các mẫu độ chói. Các biến số *TileUnitSizeY, PicWidthInTileUnitsY và PicHeightInTileUnitsY* được suy ra như sau;

TileUnitSizeY = Max (CtbSizeY >> (tile_unit_size_idc), 8) (7-14)
PicWidthInTileUnitsY = Ceil(pic_width_in_luma_samples ÷ TileUnitSizeY) (7-14)
PicHeightInTileUnitsY = Ceil(pic_height_in_luma_samples ÷ TileUnitSizeY) (7-14)
Nếu tiles_enabled_flag là bằng 1, các biến số PicWidthInCtbsY, PicHeightInCtbsY, PicSizeInCtbsY được điều chỉnh như sau:
for(PicWidthInCtbsY = 0, i = 0; i <= num_tile_columns_minus1; i++)
*PicWidthInCtbsY += Ceil(colWidth[i] * TileUnitSizeY ÷ CtbSizeY) (7-14)*
for(PicHeightInCtbsY = 0, j = 0; j <= num_tile_rows_minus1; j++)
*PicHeightInCtbsY += Ceil(rowHeight[j] * TileUnitSizeY ÷ CtbSizeY) (7-14)*
*PicSizeInCtbsY = PicWidthInCtbsY * PicHeightInCtbsY (7-14)*

num_tile_columns_minus1 cộng 1 đặc tả số cột ô kiểu xếp ngói phân chia hình ảnh theo các đơn vị của các khối đơn vị. num_tile_columns_minus1 sẽ là trong phạm vi từ 0 đến *PicWidthInTileUnitsY* − 1, gồm cả hai giá trị biên. Nếu không có mặt, thì giá trị của num_tile_columns_minus1 được suy ra là bằng 0.

num_tile_rows_minus1 cộng 1 đặc tả số hàng ô kiểu xếp ngói phân chia hình ảnh theo các đơn vị của các khối đơn vị ô xếp kiểu ngói. num_tile_rows_minus1 sẽ là trong phạm vi từ 0 đến *PicHeightInTileUnitsY* − 1, gồm cả hai giá trị biên. Nếu không có mặt thì giá trị của num_tile_rows_minus1 được suy ra là bằng 0.

Biến số *NumTilesInPic* được suy ra như sau:

```
NumTilesInPic = ( num_tile_columns_minus1 + 1 ) * ( num_tile_rows_minus1 + 1)
```

Danh sách colWidth[i] cho i xếp từ 0 đến num_tile_columns_minus1, kể cả hai giá trị biên, đặc tả chiều rộng của cột ô kiểu xếp ngói i-th theo các đơn vị của các khối đơn vị ô xếp kiểu ngói, được suy ra như sau:

```
if( uniform_spacing_flag )
    for( i = 0; i <= num_tile_columns_minus1; i++ )
posR = ( ( i + 1 ) * PicWidthInTileUnitsY ) / ( num_tile_columns_minus1 + 1)
posL = ( i * PicWidthInTileUnitsY ) / ( num_tile_columns_minus1 + 1 )
colWidth[ i ] = min(PicWidthInTileUnitsY – posL, posR – posL)
else {
    colWidth[ num_tile_columns_minus1 ] = PicWidthInTileUnitsY (6-3)
    for( i = 0; i < num_tile_columns_minus1; i++ ) {
        colWidth[ i ] = column_width_minus1[ i ] + 1
        colWidth[ num_tile_columns_minus1 ] −= colWidth[ i ]
    }
```

Danh sách *rowHeight[j]* cho j xếp từ 0 đến num_tile_rows_minus1, gồm cả hai giá trị biên, đặc tả chiều cao của hàng ô xếp kiểu ngói j-th theo các đơn vị của các khối đơn vị ô xếp kiểu ngói, được suy ra như sau:

```
if( uniform_spacing_flag )
 for( j = 0; j <= num_tile_rows_minus1; j++ )
 posB = ( ( j + 1 ) * PicHeightInTileUnitsY ) / ( num_tile_rows_minus1 + 1 )
 posT = ( j * PicHeightInTileUnitsY ) / ( num_tile_rows_minus1 + 1 )
 rowHeight [ j ] = min(PicHeightInTileUnitsY – posT, posB – posT )
else {
        rowHeight[ num_tile_rows_minus1 ] = PicHeightInTileUnitsY (6-4)
 for( j = 0; j < num_tile_rows_minus1; j++ ) {
 rowHeight[ j ] = row_height_minus1[ j ] + 1
 rowHeight[ num_tile_rows_minus1 ] −= rowHeight[ j ]
 }
}
```

Danh sách TileColX[i] cho i xếp từ 0 đến num_tile_columns_minus1 + 1, gồm cả hai giá trị biên, đặc tả vị trí X của mẫu độ chói bên trên-bên trái của cột ô kiểu xếp ngói i-th theo các đơn vị của các mẫu độ chói, được suy ra như sau:

```
for( TileColX[ 0 ] = 0, i = 0; i <= num_tile_columns_minus1; i++ )
  TileColX[ i + 1 ] = TileColX[ i ] + colWidth[ i ] * TileUnitSizeY
```

Danh sách TileRowY[j] cho j xếp từ 0 đến num_tile_rows_minus1 + 1, gồm cả hai giá trị biên, đặc tả vị trí Y của mẫu độ chói phía trên-bên trái của hàng ô kiểu xếp ngói j-th theo các đơn vị của các mẫu độ chói, được suy ra như sau:

```
for( TileRowY[ 0 ] = 0, j = 0; j <= num_tile_rows_minus1; j++ )
  TileRowY[ j + 1 ] = TileRowY[ j ] + rowHeight[ j ] * TileUnitSizeY
```

Các nhóm phân đoạn, các phân đoạn và các đơn vị

Các nhóm phân đoạn, các phân đoạn và các đơn vị được mô tả ngay đây. Thuật ngữ phân đoạn được sử dụng như thuật ngữ phổ biến hơn so với các ô kiểu xếp ngói, bởi vì các giải pháp trong sáng chế này có thể được áp dụng với các loại sơ đồ phân chia hình ảnh khác nhau và không chỉ các phân chia ô kiểu xếp ngói được biết đến từ HEVC và dự thảo VVC. Trong sáng chế này, ô kiểu xếp ngói là một phương án của phân đoạn, nhưng có thể cũng có các phương án khác của các phân đoạn.

Fig.3 thể hiện hình ảnh (10) của luồng viđeo và sự phân chia làm ví dụ của hình ảnh thành các đơn vị (8), các phân đoạn (11) và các nhóm phân đoạn (12). Fig.3 (a) thể hiện hình ảnh (10) mà bao gồm 64 đơn vị (8). Fig.3 (b) thể hiện cấu trúc phân chia đoạn (13) của cùng hình ảnh (10) bao gồm 16 phân đoạn (11). Cấu trúc phân đoạn (13) được thể hiện bằng các đường gạch ngang. Mỗi phân đoạn (11) bao gồm một số đơn vị. Phân đoạn có thể hoặc là bao gồm số nguyên của các đơn vị hoàn chỉnh hoặc sự kết hợp của các đơn vị hoàn chỉnh và một phần. Số phân đoạn tạo thành nhóm phân đoạn. Fig.3 (c) thể hiện sự phân chia nhóm phân đoạn của cùng hình ảnh (10) mà bao gồm 8 nhóm phân đoạn. Nhóm phân đoạn có thể bao gồm các phân đoạn theo trật tự quét mành. Thay vào đó, nhóm phân đoạn có thể bao gồm nhóm các phân đoạn bất kỳ mà cùng tạo thành hình chữ nhật. Thay vào đó, nhóm phân đoạn có thể bao gồm tập hợp con gồm các phân đoạn bất kỳ.

Fig.4 thể hiện hình ảnh (10) mà các đường gạch ngang thể hiện cấu trúc phân chia chia hình ảnh thành bốn phân đoạn. Fig.4 cũng thể hiện ba đơn vị (16, 17, 18). Như được

thể hiện ở hình vẽ, hai đơn vị (16, 17) thuộc về một phân đoạn hiện tại (15) và một đơn vị (18) thuộc về phân đoạn khác, lân cận (14). Các phân đoạn là không phụ thuộc đối với các phân đoạn khác, nghĩa là các biên phân đoạn được xử lý tương tự các biên hình ảnh khi giải mã các đơn vị. Điều ảnh hưởng đến quá trình suy ra các phần tử trong khi giải mã như, ví dụ, việc suy ra các chế độ nội dự đoán và việc suy ra các giá trị tham số lượng tử hóa.

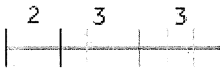
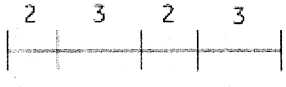
Các nội chế độ được biết đến rõ trong tình trạng kỹ thuật này và được sử dụng và được báo hiệu cho các đơn vị mà chỉ sử dụng việc dự đoán từ các mẫu được giải mã trước đó của hình ảnh hiện tại cho việc dự đoán mẫu. Phổ biến là việc suy ra chế độ nội dự đoán trong đơn vị hiện tại (16) phụ thuộc vào các chế độ nội dự đoán được suy ra trước đó trong các đơn vị khác, lân lận (17). Với các đoạn là độc lập, việc suy ra chế độ nội dự đoán trong đơn vị hiện tại (16) có thể chỉ phụ thuộc vào các chế độ nội dự đoán được suy ra trước đó trong các đơn vị (17) mà thuộc về phân đoạn hiện tại (15) và có thể không phụ thuộc vào chế độ nội dự đoán bất kỳ trong đơn vị (18) bất kỳ mà thuộc về phân đoạn khác nhau (14).

Điều này nghĩa là cấu trúc phân chia ở Fig.4 làm các chế độ nội dự đoán trong các đơn vị (18) trong phân đoạn khác nhau (14) không sẵn có cho việc suy ra chế độ nội dự đoán cho các đơn vị (16) trong phân đoạn hiện tại (15). Lưu ý rằng chế độ trong một số đơn vị (18) trong phân đoạn khác nhau có thể đã được sử dụng tốt để suy ra chế độ nội dự đoán trong đơn vị (16) trong phân đoạn hiện tại (15) nếu các đơn vị này lẽ ra sẽ được thuộc về cùng phân đoạn. Thay vào đó, biên phân đoạn có thể có ảnh hưởng đối với việc suy ra nội chế độ giống như biên hình ảnh cho các đơn vị (16) trong phân đoạn hiện tại (15).

Trong ngữ cảnh của sáng chế này, phân đoạn có thể là ô xếp kiểu ngói hoặc lát và nhóm phân đoạn có thể là nhóm ô xếp kiểu ngói. Trong sáng chế này, thuật ngữ "ô xếp kiểu ngói" và "phân đoạn" có thể được sử dụng hoán đổi cho nhau. Trong một số phương án, đơn vị có thể là tương đương với CTU.

Bản chất kỹ thuật của sáng chế

Theo các phương trình HEVC (A) và (B), các chiều rộng cột (và các chiều cao hàng) ô xếp kiểu ngói được tính toán bằng cách trừ hai phương trình này cho nhau theo dạng $\lfloor (i + 1) \cdot k \rfloor - \lfloor i \cdot k \rfloor$ trong đó i là số nguyên không âm và k là số hữu tỉ với tử số bằng PicWidthInCtbsY và mẫu số bằng num_tile_columns_minus1+1. Kết quả của phép tính này khi k không phải số nguyên, có thể là bằng $\lfloor k \rfloor$ hoặc $\lceil k \rceil$ tùy thuộc vào các giá trị của k và i. Đặc điểm vốn có này làm cho các biến số lớn bằng một CTU theo các kích thước chiều rộng cột ô xếp kiểu ngói và chiều cao hàng ô xếp kiểu ngói. Trong ngữ cảnh của sáng chế này, các biến số được đề cập đến như độ gợn sóng kích thước ô xếp kiểu ngói (Xem Bảng 2 dưới đây cho một số ví dụ). Mẫu cho độ gợn sóng này là không cố định và phụ thuộc vào chiều rộng của hình ảnh trong CTU và số cột và hàng ô xếp kiểu ngói mà cho giá trị k, và việc định vị ô xếp kiểu ngói trên lưới ô xếp kiểu ngói được xác định bằng i. Một số ví dụ về các độ gợn sóng trong chiều rộng cột ô xếp kiểu ngói sử dụng sự phân chia ô xếp kiểu ngói HEVC với uniform_spacing_flag bằng 1 được minh họa ở Bảng 2. Các ví dụ giống nhau có thể cũng được áp dụng cho các chiều cao hàng ô xếp kiểu ngói.

PicWidthInCtbsY	num_tile_columns_minus1	k	colWidth[i], i=0, 1, … , num_tile_columns_minus1
8	2	8/3	2, 3, 3 (Ô xếp kiểu ngói nhỏ hơn bắt đầu lưới) 
10	3	10/4	2, 3, 2, 3 (Các kích thước ô xếp kiểu ngói được trộn lẫn) 
100	30	100/31	3, 3, 3, 3, 4, 3, 3, 3, 4, 3, 3, 3, 3, 4, 3, 3, 3, 4, 3, 3, 3, 3, 4, 3, 3, 3, 4, 3, 3, 3, 4

			(Sự trộn lẫn bất thường của các kích thước ô xếp kiểu ngói)

Bảng 2

Bảng 2 thể hiện chiều rộng ô xếp kiểu ngói được tính toán sử dụng sự phân chia ô xếp kiểu ngói HEVCvới uniform_spacing_flag bằng 1 và các giá trị được cho PicWidthInCtbsY và num_tile_**columns**_minus1. Các độ gợn sóng trong các giá trị của colWidth[i] là có thể nhìn thấy được.

Sự không tương thích này trong các kích thước ô kiểu xếp ngói cuối cùng là không mong muốn như do nhu cầu để khảo sát các chi tiết của các giá trị đầu vào và mã để dự đoán các giá trị kích thước ô xếp kiểu ngói cuối cùng. Vấn đề về độ gợn sóng có thể xảy ra ở cả hướng ngang và dọc. Điều này gây ra khó khăn hơn để xác định các kích thước ngang và dọc đúng của ô xếp kiểu ngói cụ thể trong lưới ô xếp kiểu ngói mà không kiểm tra các chi tiết.

Vấn đề khác là ở chỗ trong phương án thực hiện hiện tại của HEVC cho việc đặt cách đồng đều các ô kiểu xếp ngói, nếu một số hàng hoặc cột của các ô kiểu xếp ngói trong hình ảnh được loại, thì các biên ô xếp kiểu ngói bên trong phần còn dư của hình ảnh có thể dịch chuyển theo mẫu gợn sóng mới. Điều này sẽ đòi hỏi việc tính toán lại kích thước ô xếp kiểu ngói và các địa chỉ trong quy trình trích xuất ô kiểu xếp ngói. Ví dụ được minh họa ở Fig.5 như Bảng 500 thể hiện các biên cột ô xếp kiểu ngói và vì vậy là các kích thước ô xếp kiểu ngói có thể thay đổi về việc đặt cách ô xếp kiểu ngói đồng đều HEVC như thế nào nếu một số một số ô kiểu xếp ngói được tách ra khỏi hình ảnh gốc khi uniform_spacing_flag là bằng 1. Các biên ô xếp kiểu ngói mà thay đổi trong trường hợp bỏ một số cột ô xếp kiểu ngói so với các biên ô xếp kiểu ngói trong hình ảnh gốc được thể hiện ở các đường đậm.

Bảng 500 trong Fig.5 minh họa các biên ô xếp kiểu ngói sử dụng việc đặt cách ô xếp kiểu ngói đồng đều HEVC. Các biên ô xếp kiểu ngói bên trong thay đổi giữa hình ảnh gốc và sau khi loại một số các ô kiểu xếp ngói khỏi hình ảnh gốc. Các tham số cho

hình ảnh gốc được thiết đặt như sau: uniform_spacing_flag = 1, PicWidthInCtbsY = 10, num_tile_columns_minus1 = 3, num_tile_rows_minus1 = 1.

JVET-L0359 đề xuất sự phân chia ô xếp kiểu ngói linh hoạt mà cho phép bậc chi tiết kích thước đơn vị ô xếp kiểu ngói tốt hơn bằng cách cung cấp khả năng để sử dụng các CTU không hoàn chỉnh ở mép bên phải và phía dưới cùng của mọi ô xếp kiểu ngói (chứ không phải mép bên phải và phía dưới của hình ảnh). Fig.2 cung cấp ví dụ cho sự phân đoạn ô xếp kiểu ngói 2x2 sử dụng sự phân chia ô xếp kiểu ngói linh hoạt như trong JVET-L0359 trong đó kích thước đơn vị ô xếp kiểu ngói là một phần tư của kích thước CTU và uniform_spacing_flag là bằng 1.

Như được đề xuất trong JVET-L0359, nếu uniform_spacing_flag là bằng 1, thì các chiều rộng của các cột ô xếp kiểu ngói được xác định sử dụng phương trình như sau:

```
if( uniform_spacing_flag )
 for( i = 0; i <= num_tile_columns_minus1; i++ )
 posR = ( ( i + 1 ) * PicWidthInTileUnitsY ) / ( num_tile_columns_minus1 + 1 )
    posL = ( i * PicWidthInTileUnitsY ) / ( num_tile_columns_minus1 + 1 )
 colWidth[ i ] = min(PicWidthInTileUnitsY – posL, posR – posL)
```

trong đó:

PicWidthInTileUnitsY = Ceil(pic_width_in_luma_samples ÷ TileUnitSizeY)

Phương trình tương tự được sử dụng để xác định các chiều cao của các hàng ô kiểu xếp ngói (rowHeight[i]).

Phương án thực hiện ở trên thể hiện rằng vấn đề về độ gợn sóng cũng tồn tại trong đề xuất JVET-L0359. Khả năng được đưa ra để xác định kích thước ô xếp kiểu ngói với bậc chi tiết tốt hơn so với bậc chi tiết CTU sinh ra thành phần giả khác nữa của các kích thước ô xếp kiểu ngói không tương thích như được mô tả ở đây. Sử dụng thuật toán được đề xuất trong L0359, sự phân chia ô xếp kiểu ngói không duy trì tương thích cho một số giá trị về chiều rộng hình ảnh và kích thước đơn vị ô xếp kiểu ngói khi kích thước đơn vị ô xếp kiểu ngói, ví dụ thay đổi thành một nửa hoặc một phần tư. Lý do là ở chỗ phương

trình được đề xuất cho sự phân chia ô xếp kiểu ngói đồng đều trong L0359 cho phép các kích thước đơn vị ô xếp kiểu ngói khác nhau trong khi đó nó không chính quy hóa sự sắp xếp về các ô xếp kiểu ngói hơi lớn hơn hoặc hơi nhỏ hơn trong lưới ô xếp kiểu ngói. Bảng 3 dưới đây thể hiện một số ví dụ cho các trường hợp mà kích thước hình ảnh được cố định, và kích thước đơn vị ô xếp kiểu ngói được thay đổi, ví dụ, được chia thành một hoặc nửa hoặc một phần tư sử dụng sự phân chia ô xếp kiểu ngói linh hoạt được đề xuất trong L0359. Các chiều rộng cột ô xếp kiểu ngói cuối cùng dịch chuyển theo kết quả thay đổi về kích thước đơn vị ô xếp kiểu ngói theo cách mà đôi lúc chiều rộng ô xếp kiểu ngói lớn hơn là ở bên trái và đôi khi là ở bên phải của hình ảnh. Khả năng không dự đoán được này là không được mong muốn.

Chiều rộng hình ảnh (trong các mẫu độ chói)	Kích thước đơn vị ô xếp kiểu ngói	PicWidthInTileUnitsY	num_tile_columns_minus1	k	colWidth[i], i=0, 1, …, num_tile_columns_minus1 (trong kích thước đơn vị ô xếp kiểu ngói)	colWidth[i], i=0, 1, …, num_tile_columns_minus1 (trong các mẫu độ chói)
2160	32	Ceil(67,5) = 68	1	68/2	34, 34	1072, 1088 Ô xếp kiểu ngói nhỏ hơn bắt đầu lưới
	16	135	1	135/2	67,68	1088, 1072 Ô xếp kiểu ngói lớn hơn bắt đầu lưới
1080	16	Ceil(67,5) = 68	1	68/2	34, 34	544, 536 Ô xếp kiểu ngói lớn hơn bắt đầu lưới
	8	135	1	135/2	67,68	536, 544 Ô xếp kiểu ngói nhỏ hơn bắt đầu lưới
480	128	Ceil(3,75) = 4	1	4/2	2, 2	256, 224

						Ô xếp kiểu ngói lớn hơn bắt đầu lưới
	32	15	1	15/2	7, 8	224, 256 Ô xếp kiểu ngói nhỏ hơn bắt đầu lưới

Bảng 3

Bảng 3 thể hiện chiều rộng cột ô xếp kiểu ngói được tính toán sử dụng sự phân chia ô xếp kiểu ngói linh hoạt như trong L0359 khi uniform_spacing_flag được thiết đặt là 1.

Có thể kết luận rằng nguyên tắc để tổ chức sự không tương thích về hiệu quả làm tròn đối với các kích thước ô xếp kiểu ngói sẽ giúp dễ dàng để xác định các kích thước ô xếp kiểu ngói cuối cùng mà không cần khảo sát chi tiết mã và các giá trị đầu vào. Phương thức hệ thống để chính quy hóa các độ gợn sóng kích thước ô xếp kiểu ngói cũng sẽ đưa ra các kết quả phân chia ô xếp kiểu ngói tương thích cho hình ảnh gốc và tập hợp con gồm các ô xếp kiểu ngói từ hình ảnh gốc.

Vấn đề riêng là ở chỗ các ô kiểu xếp ngói sinh ra từ sự phân chia ô xếp kiểu ngói đồng đều có thể thay đổi đáng kể về kích thước. Trong Bảng 600 được thể hiện trong Fig. 6, ví dụ, các ô xếp kiểu ngói trong hình ảnh gốc để đặt cách ô xếp kiểu ngói đồng đều HEVC có 4 kích thước khác nhau với kích thước nhỏ nhất là 2x3 CTU và kích thước lớn nhất là 3x4 CTU.

Giải pháp được đề xuất ở đây mà chính quy hóa các độ gợn sóng đối với kích thước của các phân đoạn với trật tự ưu tiên khi các kích thước phân đoạn không thể chính xác theo bậc chi tiết giới hạn của sự phân chia. Trong giải pháp được đề xuất, các ô kiểu xếp ngói với kích thước hơi nhỏ hơn hoặc hơi lớn hơn theo kết quả làm tròn có sự sắp xếp ưu tiên. Trong một phương án, các chiều rộng ô xếp kiểu ngói theo hướng quét từ bên trái sang bên phải, sẽ chỉ giữ giống nhau hoặc giảm. Giải pháp được đề xuất được bộc lộ ở đây có thể được áp dụng cho chiều rộng cột ô xếp kiểu ngói và/hoặc chiều cao hàng ô xếp kiểu ngói trong lưới ô xếp kiểu ngói hoặc cho chiều rộng và chiều cao của các ô kiểu xếp ngói riêng. Ở một phương án, các ô kiểu xếp ngói với các kích thước hơi lớn hơn

(do bậc chi tiết đã cho và kết quả làm tròn) đặt trong phần phía trên-bên trái của hình ảnh và các ô kiểu xếp ngói với các kích thước hơi nhỏ hơn đặt ở phần phía dưới-bên phải của hình ảnh.

Ở các giải pháp thay thế, các phương pháp được đề xuất ở đây cho các ô kiểu xếp ngói đồng nhất trong đó các kích thước ô xếp kiểu ngói sinh ra có các kích thước theo như có thể.

Theo đó, ở một khía cạnh phương pháp để giải mã hình ảnh được cung cấp. Phương pháp bao gồm việc giải mã thông tin mà hình ảnh được phân chia thành nhiều hơn một phân đoạn dựa vào một hoặc nhiều phần tử cú pháp trong luồng bit. Phương pháp cũng bao gồm việc giải mã thông tin mà sự phân đoạn không gian là đồng đều dựa vào một hoặc nhiều phần tử cú pháp trong luồng bit. Phương pháp cũng bao gồm việc xác định kích thước đơn vị phân đoạn dựa vào một hoặc nhiều phần tử cú pháp hoặc dựa vào kích thước đơn vị phân đoạn định trước. Phương pháp cũng bao gồm việc giải mã giá trị thứ nhất chỉ báo chiều rộng phân đoạn một hoặc nhiều từ mã trong luồng bit. Phương pháp cũng bao gồm việc giải mã giá trị thứ hai chỉ báo chiều cao phân đoạn từ một hoặc nhiều từ mã trong luồng bit. Phương pháp cũng bao gồm việc suy ra các chiều rộng cột phân đoạn dựa vào chiều rộng hình ảnh theo số đơn vị phân đoạn và giá trị thứ nhất. Phương pháp cũng bao gồm việc suy ra các chiều cao hàng phân đoạn dựa vào chiều cao hình ảnh theo số đơn vị phân đoạn và giá trị thứ hai. Phương pháp cũng bao gồm việc suy ra vị trí không gian cho khối hiện tại dựa vào các chiều rộng cột phân đoạn được suy ra và các chiều cao hàng phân đoạn được suy ra. Phương pháp cũng bao gồm việc giải mã khối hiện tại dựa vào vị trí không gian được suy ra.

Giải pháp được đề xuất giữ chiều rộng của ô xếp kiểu ngói hiện tại độc lập với các ô kiểu xếp ngói trước mà chỉ đố với số đơn vị còn dư (trong cột hoặc hàng) mà sẽ được phân chia thành số ô kiểu xếp ngói còn dư. Kết quả, giải pháp được đề xuất cho việc đặt cách đồng đều các ô kiểu xếp ngói giữ cho sự phân chia ô xếp kiểu ngói không bị ảnh hưởng đối với các ô kiểu xếp ngói được tách ra và dư nếu các ô kiểu xếp ngói được tách ra. Điều này sẽ không cần đến yêu cầu tính toán lại các kích thước ô xếp kiểu ngói trong

trường hợp phân chia hình ảnh hoặc tách ô xếp kiểu ngói ra. Trong phương án thực hiện hiện tại của HEVC cho việc đặt cách đồng đều các ô kiểu xếp ngói, nếu một số hàng hoặc cột của các ô kiểu xếp ngói trong hình ảnh được loại bỏ, thì mẫu độ gợn sóng có thể thay đổi và các biên ô xếp kiểu ngói bên trong phần còn dư của hình ảnh có thể dịch chuyển theo mẫu độ gợn sóng mới.

Sự tương thích là ưu điểm khác của giải pháp được đề suất. Giải pháp được đề xuất cho sự đặt cách đồng đều các ô kiểu xếp ngói tổ chức các kích thước ô xếp kiểu ngói (ví dụ, chiều rộng cột ô xếp kiểu ngói và chiều cao hàng ô xếp kiểu ngói) và cung cấp nguyên tắc cho việc sắp xếp cuối cùng các kích thước ô xếp kiểu ngói mà không cần khảo sát chi tiết các tham số đầu vào hoặc mã. Ví dụ, trong một phương án của giải pháp được đề xuất khi chiều rộng/chiều cao hình ảnh theo đơn vị ô xếp kiểu ngói không thể phân chia được bằng số cột/hàng ô xếp kiểu ngói, thì tất cả các ô kiểu xếp ngói được làm tròn đến các kích thước lớn hơn được tìm thấy trên phần bên trái/phía trên của hình ảnh. Giải pháp được đề xuất cũng chính quy hóa các độ gợn sóng của các kích thước ô xếp kiểu ngói trong khoảng đặt cách ô xếp kiểu ngói đồng đều trong trường hợp có các thay đổi sang kích thước đơn vị ô xếp kiểu ngói.

Phương pháp được đề xuất cũng giảm sự phức tạp so với HEVC và sự phân chia ô xếp kiểu ngói linh hoạt được đề xuất trong JVET-L0359. Phương pháp được đề xuất không có tính nhân và chỉ 1 tính chia trên lần lặp lại để xác định các chiều rộng của các cột ô xếp kiểu ngói và các chiều cao của các hàng ô xếp kiểu ngói trong đó phương pháp HEVC sử dụng 2 tính nhân và 2 tính chia trên lần lặp lại.

Trong các giải pháp thay thế, các kích thước ô xếp kiểu ngói sinh ra là đồng đều như nó có thể.

Mô tả vắn tắt các hình vẽ

Các hình vẽ kèm theo, mà được kết hợp trong bản mô tả này và tạo ra một phần của bản mô tả, thể hiện các phương án khác nhau.

Fig.1 là hình vẽ thể hiện ví dụ về sự phân chia ô xếp kiểu ngói theo một phương án.

Fig.2A là hình vẽ thể hiện sự phân đoạn ô xếp kiểu ngói sử dụng sự phân đoạn ô xếp kiểu ngói HEVC.

Fig.2B là hình vẽ thể hiện sự phân đoạn ô xếp kiểu ngói sử dụng sự phân chia ô xếp kiểu ngói linh hoạt.

Fig.3 là hình vẽ thể hiện hình ảnh của luồng viđeo và sự phân chia làm ví dụ.

Fig.4 là hình vẽ thể hiện hình ảnh theo một số phương án.

Fig.5 thể hiện bảng theo một số phương án.

Fig.6 thể hiện bảng theo một số phương án.

Fig.7 là sơ đồ tiến trình minh họa quy trình theo một phương án.

Fig.8 là sơ đồ tiến trình minh họa quy trình theo một phương án.

Fig.9 là sơ đồ thể hiện các khối chức năng của bộ giải mã theo một phương án.

Fig.10 là sơ đồ thể hiện các khối chức năng của bộ mã hóa theo một phương án.

Fig.11 là sơ đồ khối của nút theo một số phương án.

Fig.12 là hình vẽ thể hiện hình ảnh theo một phương án.

Fig.13 là hình vẽ thể hiện cấu trúc ô kiểu xếp ngói đồng đều theo một phương án.

Fig.14 là hình vẽ thể hiện cấu trúc ô kiểu xếp ngói theo một phương án.

Fig.15 là sơ đồ tiến trình minh họa quy trình theo một phương án.

Fig.16 là sơ đồ thể hiện các khối chức năng của bộ giải mã theo một phương án.

Mô tả chi tiết sáng chế

Hệ thống thuật ngữ sau đây được sử dụng để mô tả các phương án:

Toán tử số học "/" được sử dụng cho phép chia số nguyên với sự làm tròn kết quả hướng về không. Ví dụ, 7 / 4 và −7 / −4 được làm tròn đến thành 1 và −7 / 4 và 7 / −4 được làm tròn thành −1.

Toán tử số học "÷" được sử dụng cho phép chia trong các phương trình toán học trong đó không có sự chặt cụt hay làm tròn được nhắm tới.

Ceil(x) cho số nguyên nhỏ nhất lớn hơn hoặc bằng x.

Floor(x) cho số nguyên lớn nhất thấp hơn hoặc bằng x.

Các thuật ngữ "chiều rộng cột ô xếp kiểu ngói" và "chiều rộng ô xếp kiểu ngói " được sử dụng hoán đổi cho nhau mà nghĩa là giải pháp có thể được áp dụng khi các chiều rộng cột ô xếp kiểu ngói đang được tính toán trong lưới ô xếp kiểu ngói hoặc khi các chiều rộng ô xếp kiểu ngói đang được tính toán riêng (ví dụ nếu không có lưới ô xếp kiểu ngói).

Các thuật ngữ "chiều cao hàng ô xếp kiểu ngói" và "chiều cao ô xếp kiểu ngói " được sử dụng hoán đổi cho nhau mà nghĩa là giải pháp có thể được áp dụng khi các chiều cao hàng ô xếp kiểu ngói đang được tính toán trong lưới ô xếp kiểu ngói hoặc khi các chiều cao ô xếp kiểu ngói đang được tính toán riêng (ví dụ nếu không có lưới ô xếp kiểu ngói).

Theo một số phương án, giải pháp được đề xuất được bộc lộ ở đây để xác định chiều rộng (và tương đương là chiều cao) của các ô kiểu xếp ngói cho uniform_spacing_flag = 1 có các thành phần sau đây:

- trong vòng để xác định chiều rộng của mỗi ô xếp kiểu ngói, chiều rộng cho ô xếp kiểu ngói được xác định sử dụng các kích thước đơn vị còn dư sẵn có mà nghĩa là số kích thước đơn vị trong hàng mà vẫn chưa được phân bổ cho ô xếp kiểu ngói;

- số kích thước đơn vị còn dư sau đó được chia cho số ô xếp kiểu ngói còn dư trong hàng;

- kích thước ô xếp kiểu ngói thu được được làm tròn về số nguyên lớn hơn hoặc nhỏ hơn (Các hàm Ceil() hoặc Floor()) theo nguyên tắc đã cho để đặt trật tự chiều rộng của các ô kiểu xếp ngói; và

- thành phần tùy chọn là để tính toán lại số kích thước đơn vị còn dư (ví dụ, các CTU) trong mọi lần lặp lại của vòng để xác định chiều rộng của mỗi ô xếp kiểu ngói.

Các thành phần giống nhau áp dụng để xác định chiều cao của các ô kiểu xếp ngói.

Giải pháp được đề xuất được bộc lộ ở đây mô tả phương pháp để giải mã hình ảnh 10 từ luồng bit, phương pháp bao gồm bước suy ra các kích thước và/hoặc các vị trí cho tất cả phân đoạn trong hình ảnh từ luồng bit, trong đó hình ảnh 10 bao gồm số đơn vị 8 và cấu trúc phân chia 13 phân chia hình ảnh thành ít nhất hai phân đoạn 11 và bộ giải mã xác định sự phân đoạn không gian là đồng đều bằng cách giải mã một hoặc nhiều từ mã trong luồng bit, và bộ giải mã xác định số phân đoạn không gian bằng cách giải mã một hoặc nhiều từ mã trong luồng bit, và bộ giải mã xác định kích thước đơn vị ô xếp kiểu ngói, và sự phân chia các phân đoạn thành các chiều rộng hoặc các chiều cao đều đều theo mẫu gợn sóng cố định độc lập với số phân đoạn không gian, và việc suy ra các kích thước phân đoạn được thực hiện trong vòng trên số phân đoạn, trong đó bên trong vòng số đơn vị ô xếp kiểu ngói còn dư cần được phân đoạn và số phân đoạn còn dư được tính toán.

Trong một ví dụ, giải pháp được đề xuất thay thế các dòng HEVC sau đây:

```
if( uniform_spacing_flag )
   for( i = 0; i <= num_tile_columns_minus1; i++ )
      colWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbsY ) / ( num_tile_columns_minus1
   + 1 ) –
            ( i * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 )
```

bằng các hàng sau đây:

```
if( uniform_spacing_flag ){
            A = PicWidthInCtbsY
            B = num_tile_columns_minus1 + 1
   for( i = 0; i <= num_tile_columns_minus1; i++ ){
      colWidth[ i ] = Ceil( A ÷ B)
```

```
                    A -= colWidth[ i ]
                    B -= 1
                }
            }
```

Phương pháp được đề xuất cũng giảm sự phức tạp so với HEVC và sự phân chia ô xếp kiểu ngói linh hoạt được đề xuất trong JVET-L0359. Phương pháp được đề xuất không có tính nhân và chỉ 1 tính chia trên lần lặp lại để xác định các chiều rộng của các cột ô xếp kiểu ngói và các chiều cao của các hàng ô xếp kiểu ngói trong đó phương pháp HEVC sử dụng 2 tính nhân và 2 tính chia trên lần lặp lại.

Phương án 1. Độ gợn sóng đều đều

Ở một phương án, chiều rộng của các phân đoạn trong hàng hoặc chiều cao của các phân đoạn trong cột hoặc không bao giờ theo trật tự giảm dần theo hướng quét định trước. Phân đoạn có thể là ô xếp kiểu ngói trong hình ảnh và vì vậy các kích thước ô xếp kiểu ngói là đồng đều (không bao giờ tăng lên hoặc không bao giờ giảm xuống) theo hướng quét. Ví dụ, đối với các chiều rộng ô xếp kiểu ngói mà không bao giờ tăng lên theo hướng quét từ bên trái sang phải nghĩa mà chiều rộng của ô xếp kiểu ngói là không bao giờ lớn hơn chiều rộng của ô xếp kiểu ngói khác trên cùng hàng mà được đặt trong không gian sang bên trái của ô xếp kiểu ngói thứ nhất.

Ở phương án thứ nhất, đối với chiều rộng cột ô xếp kiểu ngói không bao giờ tăng (hướng quét từ bên trái sang bên phải) và chiều cao hàng ô xếp kiểu ngói (hướng quét từ phía trên xuống phía dưới) trên phía trên của HEVC, các hàm HEVC sau đây:

```
if( uniform_spacing_flag )
    for( i = 0; i <= num_tile_columns_minus1; i++ )
        colWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbsY ) / ( num_tile_columns_minus1
    + 1 ) –
            ( i * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 )
và
if( uniform_spacing_flag )
    for( j = 0; j <= num_tile_rows_minus1; j++ )
        rowHeight[ j ] = ( ( j + 1 ) * PicHeightInCtbsY ) / ( num_tile_rows_minus1
    + 1 ) –
            ( j * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 )
```

được thay thế bằng các hàm sau đây:

```
if( uniform_spacing_flag ) {
        A = PicWidthInCtbsY
        B = num_tile_columns_minus1 + 1
  for( i = 0; i <= num_tile_columns_minus1; i++ ) {
    colWidth[ i ] = Ceil( A ÷ B)
                    A -= colWidth[ i ]
                    B -= 1
  }
}
```

và

```
if( uniform_spacing_flag ) {
        C = PicHeightInCtbsY
        D = num_tile_rows_minus1 + 1
  for( j = 0; j <= num_tile_rows_minus1; j++ ) {
    rowHeight[ j ] = Ceil( C ÷ D)
                    C -= rowHeight [ j ]
                    D -= 1
  }
}
```

Các giá trị sinh ra trong các danh sách colWidth và rowHeight là theo các đơn vị của khối cây mã hóa độ chói. Ví dụ, nếu kích thước CTU là bằng 128x128, thì các giá trị là theo các đơn vị của 128 mẫu độ chói sao cho giá trị gấp đôi nghĩa là 256 mẫu độ chói.

Các danh sách colWidth và rowHeight sau đó được sử dụng bởi bộ giải mã để xác định trật tự quét của các khối trong hình ảnh. Khi dữ liệu khối được giải mã, thì vị trí không gian của khối được dựa vào các giá trị trong các danh sách colWidth và rowHeight. Bộ giải mã có thể xây dựng các danh sách biến đổi từ các địa chỉ quét ô xếp kiểu ngói đến các địa chỉ quét mành và cách thức khác quanh việc sử dụng các giá trị trong colWidth và rowHeight. Bộ giải mã có thể sau đó sử dụng các danh sách biến đổi trong khi giải mã để xác định các vị trí không gian của các khối. Trong HEVC, các danh sách biến đổi được sử dụng và được gọi là CtbAddrRsToTs và CtbAddrTsToRs.

Fig.6 thể hiện bảng 600 mà so sánh sự phân bổ kích thước ô xếp kiểu ngói cho HEVC và phương án thực hiện làm ví dụ trên của phương án thứ nhất. Như được thể hiện ở bảng 600, trong giải pháp được đề xuất, các biên ô xếp kiểu ngói được giữ giống như hình ảnh gốc khi các cột ô xếp kiểu ngói được loại bỏ khỏi bên trái hoặc bên phải hoặc cả hai phía của hình ảnh. Hiệu quả làm tròn được điều chỉnh để đặt các ô kiểu xếp ngói hơi rộng hơn ở phía bên trái của hình ảnh trong tất cả các trường hợp sử dụng hàm Ceil() và tính toán lại số đơn vị ô xếp kiểu ngói còn dư trong mọi lần lặp lại của vòng đối với các ô xếp kiểu ngói để xác định các chiều rộng cột ô xếp kiểu ngói.

Bảng 600 thể hiện các biên ô xếp kiểu ngói bên trong thay đổi như thế nào sau khi loại bỏ một số cột ô xếp kiểu ngói khỏi hình ảnh gốc sử dụng sự đặt cách ô xếp kiểu ngói đồng đều HEVC trong khi các biên ô xếp kiểu ngói bên trong được giữ không thay đổi ở phương án được đề xuất. Các tham số cho hình ảnh gốc được thiết đặt như sau: uniform_spacing_flag = 1, PicWidthInCtbsY = 10, num_tile_columns_minus1 = 3, num_tile_rows_minus1 = 1.

Dưới đây là ví dụ khác của phương án khác, được xây dựng ở trên các phương trình được đề xuất trong JVET-L0359 mà hỗ trợ các bậc chi tiết về kích thước ô xếp kiểu ngói nhỏ hơn kích thước CTU. Các thay đổi sau đây được đề xuất trên JVET-L0359:

```
if( uniform_spacing_flag ) {
  A = PicWidthInTileUnitsY
        B = num_tile_columns_minus1 + 1
  for( i = 0; i <= num_tile_columns_minus1; i++ ) {
  ~~posR = ( ( i + 1 ) * PicWidthInTileUnitsY ) / ( num_tile_columns_minus1 + 1 )~~
  ~~posL = ( i * PicWidthInTileUnitsY ) / ( num_tile_columns_minus1 + 1 )~~
  ~~colWidth[ i ] = min(PicWidthInTileUnitsY - posL, posR - posL)~~
    colWidth[ i ] = Ceil( A ÷ B)
                  A -= colWidth[ i ]
                  B -= 1
        }
  }

trong đó:
    PicWidthInTileUnitsY = Ceil( pic_width_in_luma_samples ÷ TileUnitSizeY )
```

và tương ứng cho các chiều cao hàng các thay đổi sau đây được đề xuất:

```
if( uniform_spacing_flag ) {
 C = PicHeightInTileUnitsY
   D = num_tile_rows_minus1 + 1
   for( j = 0; j <= num_tile_rows_minus1; j++ ) {
   posB = ( ( i + 1 ) * PicHeightInTileUnitsY ) / ( num_tile_rows_minus1 + 1 )
   posT = ( i * PicHeightInTileUnitsY ) / ( num_tile_rows_minus1 + 1 )
   rowHeight[ i ] = min(PicHeightInTileUnitsY - posT, posB - posT)
        rowHeight[ j ] = Ceil( C ÷ D)
 C -= rowHeight[ j ]
 D -= 1
 }
}

trong đó:
      PicHeightInTileUnitsY = Ceil( pic_height_in_luma_samples ÷ TileUnitSizeY )
```

Các giá trị sinh ra trong các danh sách colWidth và rowHeight là theo các đơn vị của các đơn vị ô xếp kiểu ngói độ chói. Nếu ví dụ kích thước đơn vị ô xếp kiểu ngói là bằng 32x32, thì các giá trị là theo các đơn vị của 32 mẫu độ chói sao cho giá trị gấp đôi nghĩa là 64 mẫu độ chói.

Bảng 4 so sánh các kết quả cho chiều rộng của các ô xếp kiểu ngói giữa JVET-L0359 và giải pháp được đề xuất trong phương án 1 cho các giá trị chiều rộng hình ảnh khác nhau. Các kết quả cho JVET-L0359 thể hiện sự không tương thích về chiều rộng của các cột ô xếp kiểu ngói (colWidth[i]) khi kích thước đơn vị ô xếp kiểu ngói thay đổi trong khi giải pháp được đề xuất ở phương án 1 cung cấp chiều rộng tương thích của các cột ô xếp kiểu ngói (colWidth[i]) khi kích thước đơn vị ô xếp kiểu ngói thay đổi bởi vì nó dành ưu tiên thích hợp cho các ô kiểu xếp ngói hơi lớn hơn để được đặt ở phía bên trái của hình ảnh.

Chiều rộng hình ảnh (trong các mẫu độ chói)	Kích thước đơn vị ô xếp kiểu ngói	PicWidthInTileUnitsY	JVET-L0359		Phương án thứ nhất	
			colWidth[i], i=0, 1 (trong kích thước đơn vị ô xếp kiểu ngói)	colWidth[i], i=0, 1 (trong các mẫu độ chói)	colWidth[i], i=0, 1 (trong kích thước đơn vị ô xếp kiểu ngói)	colWidth[i], i=0, 1 (trong các mẫu độ chói)
2160	32	Ceil(67,5	34, 34	1072, 1088	34, 34	1088, 1072

		) = 68		Ô xếp kiểu ngói nhỏ hơn bắt đầu lưới		Ô xếp kiểu ngói lớn hơn bắt đầu lưới
	16	135	67,68	1088, 1072 Ô xếp kiểu ngói lớn hơn bắt đầu lưới	68,67	1088, 1072 Ô xếp kiểu ngói lớn hơn bắt đầu lưới
1080	16	Ceil(67,5) = 68	34, 34	544, 536 Ô xếp kiểu ngói lớn hơn bắt đầu lưới	34, 34	544, 536 Ô xếp kiểu ngói lớn hơn bắt đầu lưới
	8	135	67,68	536, 544 Ô xếp kiểu ngói nhỏ hơn bắt đầu lưới	68,67	544, 536 Ô xếp kiểu ngói lớn hơn bắt đầu lưới
480	128	Ceil(3,75) = 4	2, 2	256, 224 Ô xếp kiểu ngói lớn hơn bắt đầu lưới	2, 2	256, 224 Ô xếp kiểu ngói lớn hơn bắt đầu lưới
	32	15	7, 8	224, 256 Ô xếp kiểu ngói nhỏ hơn bắt đầu lưới	8, 7	256, 224 Ô xếp kiểu ngói lớn hơn bắt đầu lưới

Bảng 4

Bảng 4 thể hiện chiều rộng cột ô xếp kiểu ngói được tính toán khi uniform_spacing_flag = 1, num_tile_columns_minus1 = 1, sử dụng sự phân chia ô xếp kiểu ngói linh hoạt như trong JVET-L0359 so với giải pháp được đề xuất trong phương án thứ nhất.

Ở phương án thứ nhất, bộ giải mã có thể thực hiện tất cả hoặc tập hợp con của các bước sau đây:

1. Giải mã thông tin mà một hoặc nhiều hình ảnh được phân chia thành nhiều hơn một phân đoạn từ một hoặc nhiều phần tử cú pháp trong luồng bit. Cú pháp tốt hơn là được đặt trong tập hợp tham số hình ảnh. Cú pháp có thể đặc tả số R chỉ báo số hàng phân đoạn và số C chỉ báo số cột phân đoạn.

2. Giải mã thông tin mà sự phân đoạn không gian là đồng đều từ một hoặc nhiều phần tử cú pháp trong luồng bit. Cú pháp tốt hơn là được đặt trong tập hợp tham số hình ảnh. Cú pháp có thể bao gồm cờ một bit đặc tả sự phân đoạn không gian là đồng đều hay không.

3. Xác định kích thước đơn vị phân đoạn S từ một hoặc nhiều phần tử cú pháp hoặc bằng cách sử dụng kích thước đơn vị phân đoạn định trước. Nếu kích thước đơn vị định trước được sử dụng, thì có thể là bằng kích thước của CTU hoặc kích thước của CTU trong một chiều. Ví dụ, nếu kích thước CTU là bằng 128x128, thì kích thước đơn vị phân đoạn S có thể là bằng 128 (hoặc 128x128).

4. Tính toán kích thước của hình ảnh theo số lượng các đơn vị phân đoạn. Phép tính này có thể được thực hiện riêng biệt cho chiều cao và chiều rộng của hình ảnh sao cho kích thước ngang (Horizontal size - HS) được thiết đặt bằng chiều rộng hình ảnh trong các mẫu độ chói được chia cho kích thước đơn vị phân đoạn S. Kích thước dọc (Vertical size - VS) có thể được thiết đặt bằng chiều cao hình ảnh trong các mẫu độ chói được chia cho kích thước đơn vị phân đoạn.

5. Suy ra chiều rộng và các chiều cao của tất cả các phân đoạn từ số phân đoạn trong hình ảnh và kích thước đơn vị phân đoạn S. Việc suy ra được thực hiện trong vòng trên số phân đoạn mà số phân đoạn này vẫn chưa được phân đoạn và kích thước của hình ảnh vẫn chưa được phân đoạn theo số đơn vi phân đoạn được cập nhật trong mỗi lần lặp lại của vòng.

Việc suy ra này có thể được thực hiện trong hai vòng riêng biệt:

a. Suy ra các chiều rộng cột phân đoạn từ chiều rộng hình ảnh theo số đơn vị ô xếp kiểu ngói và số cột phân đoạn C bằng các bước con như sau:

i. Thiết đặt chiều rộng hình ảnh vẫn chưa được phân đoạn (A) bằng giá trị HS

ii. Thiết đặt số cột phân đoạn vẫn chưa được phân đoạn (B) bằng giá trị

iii. Suy ra các chiều rộng cột trong vòng trong đó một giá trị chiều rộng được suy ra trên lần lặp lại và trong số cột phân đoạn vẫn chưa được phân đoạn (B) và chiều rộng hình ảnh vẫn chưa được phân đoạn (A) đều được cập nhật trong mỗi lần lặp lại. Sự lặp lại có thể được thực thi C lần

iv. Chiều rộng cột được suy ra W có thể được thiết đặt bằng Ceil(A ÷ B)

v. Biến số A có thể sau đó được cập nhật thành A-W và biến số B có thể được cập nhật thành B-1

b. Suy ra các chiều cao hàng phân đoạn từ chiều cao hình ảnh theo số đơn vị ô xếp kiểu ngói và số hàng phân đoạn R bằng các bước con sau đây:

i. Thiết đặt chiều cao hình ảnh vẫn chưa được phân đoạn (A) bằng giá trị VS

ii. Thiết đặt số hàng phân đoạn vẫn chưa được phân đoạn (B) bằng giá trị R

iii. Suy ra các chiều cao hàng trong vòng trong đó một giá trị chiều cao được suy ra trên lần lặp lại và trong số hàng phân đoạn vẫn chưa được phân đoạn (B) và chiều cao hình ảnh vẫn chưa được phân đoạn (A) đều được cập nhật trong mỗi lần lặp lại. Sự lặp lại có thể được thực thi R lần.

1. Chiều cao hàng được suy ra H có thể được thiết đặt bằng Ceil(A ÷ B)

2. Biến số A có thể sau đó được cập nhật thành A-H và biến số B có thể được cập nhật thành B-1

6. Suy ra vị trí không gian cho khối hiện tại sử dụng các chiều rộng phân đoạn được suy ra và các chiều cao phân đoạn được suy ra.

7. Giải mã khối hiện tại sử dụng vị trí không gian được suy ra. Lưu trữ các giá trị mẫu được giải mã cho khối hiện tại trong bộ nhớ ở các vị trí của bộ nhớ tương ứng với vị trí không gian được suy ra.

Ở biến thể của phương án này, hàm Ceil() có thể được thay thế bằng Floor(), theo chỉ hướng ngang, hoặc chỉ hướng dọc hoặc cả hai hướng.

Ở biến thể của phương án này, cờ có thể chọn giữa hai hàm Ceil() và Floor(). Trong một số phương án, có thể có hai cờ độc lập cho các hướng ngang hoặc dọc.

Phương án 2. Trật tự được xác định

Ở phương án thứ hai, chiều rộng của các phân đoạn trong một hàng hoặc chiều cao của các phân đoạn trong một cột theo trình tự được xác định. Trình tự được xác định này có thể được báo hiệu trong luồng bit sử dụng mô hình mẫu dưới dạng trình tự bit. Các phân đoạn có thể là các ô kiểu xếp ngói trong hình ảnh và vì vậy trong phương án này, hình dạng của độ gợn sóng ưu tiên trên các kích thước ô xếp kiểu ngói được xác định. Ví dụ, tất cả các ô kiểu xếp ngói hơi rộng hơn là ở phía trái của hình ảnh. Điều này có thể được thực hiện bằng mô hình mẫu được biểu diễn dưới dạng trình tự bit mà xác định hàm Ceil() hoặc Floor() trong mọi lần lặp lại của vòng trên các ô xếp kiểu ngói để đặc tả chiều rộng (chiều cao) ô xếp kiểu ngói. Hàm Ceil() có thể được biểu diễn bằng 1 và hàm Floor() có thể được biểu diễn bằng 0 trong trình tự bit mẫu. Như ví dụ, mẫu 110 sẽ đặc tả hàm Ceil() trong hai ô kiểu xếp ngói thứ nhất và hàm Floor() trong lần lặp lại thứ ba của vòng để đặc tả kích thước ô xếp kiểu ngói. Mẫu có thể được lặp lại theo chu kỳ nếu số ô xếp kiểu ngói là lớn hơn chiều dài của trình tự bit mẫu.

Phương án 3. Các phân chia ô xếp kiểu ngói nhị phân

Ở phương án thứ ba, sự phân chia phân đoạn nhị phân với sự đặt trình tự rõ ràng chiều rộng hoặc chiều cao phân đoạn được áp dụng cho chiều rộng hoặc chiều cao của hình ảnh. Các phân đoạn có thể là các ô kiểu xếp ngói trong hình ảnh và vì vậy sự phân chia nhị phân được sử dụng cho việc phân chia hình thành thành các ô kiểu xếp ngói 2^n với các kích thước ô xếp kiểu ngói đồng đều. Mẫu gợn sóng định trước (như trong phương án thứ hai) có thể được sử dụng theo cách kết hợp với phương án này. Sự phân chia nhị phân có thể tiếp tục theo cách phân cấp cho đến khi thuật toán đạt đến số phân đoạn xác định hoặc chỉ lên đến số các bước xác định và phần còn dư của các phân chia

phân đoạn đang được thực hiện bởi các phương pháp khác. Như một ví dụ, tổng kích thước được chia cho 2 ở bước thứ nhất trong đó được xác định để gán cho phân đoạn nhỏ hơn hoặc lớn hơn sang bên trái hoặc bên phải. Trong mỗi bước trong các bước kế tiếp, mỗi phân đoạn trong các phân đoạn bên trái và bên phải được chia thành hai phần sử dụng cấp phân chia nhị phân kế tiếp và nguyên tắc được xác định cho vị trí của phân đoạn có thể nhỏ hơn hoặc lớn hơn.

Phương án 4. Sự đặt trật tự mặc định của độ gợn sóng

Ở phương án thứ tư, chiều rộng của các phân đoạn trong hàng hoặc chiều cao của các phân đoạn trong cột của hình ảnh theo trật tự mặc định được xác định cho chiều rộng hoặc chiều cao của các phân đoạn. Các phân đoạn có thể là các ô kiểu xếp ngói trong hình ảnh và vì vậy trật tự ưu tiên mặc định được xác định để quản lý các độ gợn sóng trong các kích thước phân đoạn. Mẫu gợn sóng mặc định có thể được báo hiệu trong luồng bit. Ở biến thể của phương án này, mẫu mặc định có thể được viết đè. Ở biến thể khác của phương án này, cờ có thể đặc tả việc đặt trật tự mặc định hay đặt trật tự xác định khác đang được sử dụng hay không.

Phương án 5.

Ở phương án thứ năm, mục đích là để đạt được nhiều ô xếp kiểu ngói được định kích thước lớn bằng nhau bằng cách sử dụng phương pháp đặt cách đồng đều.

Điều này có thể đạt được bằng cách giải mã giá trị TileWidth từ một hoặc nhiều từ mã trong luồng bit. Sau đó, các giá trị chiều rộng của tất cả các cột ô xếp kiểu ngói trong hình ảnh ngoại trừ một cột được thiết đặt bằng giá trị TileWidth được giải mã. Giá trị chiều rộng của một cột ô xếp kiểu ngói còn dư được thiết đặt bằng giá trị chiều rộng hình ảnh trừ đi tổng của các giá trị chiều rộng của tất cả các cột ô xếp kiểu ngói ngoại trừ một cột ô xếp kiểu ngói còn dư. Giá trị chiều rộng của một cột còn dư có thể cũng bằng giá trị TileWidth.

Thay vào đó, hoặc theo cách bổ sung, giá trị TileHeight được giải mã từ một hoặc nhiều từ mã trong luồng bit. Sau đó, các giá trị chiều cao của tất cả các hàng ô kiểu xếp

ngói trong hình ảnh ngoại trừ một hàng được thiết đặt bằng giá trị TileHeight được suy ra. Giá trị chiều cao của một hàng ô xếp kiểu ngói còn dư được thiết đặt bằng giá trị chiều cao hình ảnh trừ đi tổng của các giá trị chiều cao của tất cả các hàng ô xếp kiểu ngói ngoại trừ một hàng ô xếp kiểu ngói còn dư. Giá trị chiều cao của một hàng còn dư có thể cũng bằng giá trị TileHeight.

Phương pháp trong phương án này dẫn đến hầu hết các ô kiểu xếp ngói là có kích thước bằng nhau, ví dụ, như được minh họa trong hình ảnh 1200 được thể hiện ở Fig.12 trong đó hầu hết các ô kiểu xếp ngói là có kích thước là 4x4 CTU hoặc các đơn vị ô xếp kiểu ngói.

Phương án có thể được thực hiện trong HEVC v5 hoặc dự thảo VVC hiện tại bằng cách bỏ phần văn bản sau đây khỏi HEVC v5 hoặc đặc tả VVC dự thảo.

```
if( uniform_tile_spacing_flag )
      for( i = 0; i <= num_tile_columns_minus1; i++ )
            ColWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 ) −
                                                                  ( i * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 )
if( uniform_tile_spacing_flag )
      for( j = 0; j <= num_tile_rows_minus1; j++ )
            RowHeight[ j ] = ( ( j + 1 ) * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 ) −
                                                                  ( j * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 )
```

Phần văn bản được loại bỏ được thay thế bằng phần văn bản sau đây lần lượt trong HEVC v5 hoặc đặc tả dự thảo VVC.

```
if( uniform_tile_spacing_flag ) {
 RemainingWidthInCtbsY = PicWidthInCtbsY
 i=0
 while( RemainingWidthInCtbsY > tile_width ) {
 ColWidth[i++] = tile_width
 RemainingWidthInCtbsY -= tile_width
 }
 ColWidth[i] = RemainingWidthInCtbsY
 num_tile_columns_minus1 = i
```

```
}

if( uniform_tile_spacing_flag ) {
 RemainingHeightInCtbsY = PicHeightInCtbsY
 i=0
 while( RemainingHeightInCtbsY > tile_height ) {
 RowHeight[i++] = tile_height
 RemainingHeightInCtbsY -= tile_height
 }
 RowHeight[i] = RemainingHeightInCtbsY
 num_tile_rows_minus1 = i
 }
```

Cú pháp và ngữ nghĩa học cho tile_width và tile_height có thể nhìn như sau:

pic_parameter_set_rbsp( ) {	**Descriptor**
...	
tiles_enabled_flag	u(1)
...	
if(tiles_enabled_flag) {	
uniform_spacing_flag	u(1)
if(uniform_spacing_flag) {	
tile_width	ue(v)
tile_height	ue(v)
} **else** {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width_minus1[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height_minus1[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
}	
...	

Bảng 5. Cú pháp ô xếp kiểu ngói cho phương án thứ bảy

Cú pháp và ngữ nghĩa học được giải thích chi tiết hơn nữa dưới đây:

num_tile_columns_minus1 cộng 1 đặc tả rằng số cột ô kiểu xếp ngói phân chia hình ảnh. num_tile_columns_minus1 sẽ là trong phạm vi từ 0 đến PicWidthInCtbsY − 1, gồm cả hai giá trị biên. Khi tiles_enabled_flag là bằng 0, thì giá trị của num_tile_columns_minus1 được suy ra là bằng 0. Trái lại, nếu uniform_spacing_flag là bằng 1, thì giá trị của num_tile_columns_minus1 được suy ra như được thể hiện ở trên.

num_tile_rows_minus1 cộng 1 đặc tả số hàng ô kiểu xếp ngói phân chia hình ảnh. num_tile_rows_minus1 sẽ là ở phạm vi bao từ 0 đến PicHeightInCtbsY − 1, gồm cả hai giá trị biên. Khi tiles_enabled_flag là bằng 0, thì giá trị của num_tile_rows_minus1 được suy ra là bằng 0. Trái lại, nếu uniform_spacing_flag là bằng 1, thì giá trị của num_tile_rows_minus1 được suy ra như được thể hiện ở trên.

Khi tiles_enabled_flag là bằng 1, num_tile_columns_minus1 và num_tile_rows_minus1 sẽ đều không bằng 0.

tile_width đặc tả chiều rộng của tất cả các ô kiểu xếp ngói không thuộc về cột ô xếp kiểu ngói cuối cùng. tile_width sẽ là trong phạm vi là từ 1 đến PicWidthInCtbsY − 1, gồm cả hai giá trị biên.

tile_height đặc tả chiều cao của tất cả các ô kiểu xếp ngói không thuộc về hàng ô xếp kiểu ngói cuối cùng. tile_height sẽ là trong phạm vi là từ 1 đến PicHeightInCtbsY − 1, gồm cả hai giá trị biên.

Bộ giải mã có thể thực hiện tất cả hoặc tập hợp con của các bước sau đây cho phương án này:

1. Giải mã thông tin mà một hoặc nhiều hình ảnh được phân chia thành nhiều hơn một phân đoạn từ một hoặc nhiều phần tử cú pháp trong luồng bit. Cú pháp tốt hơn là được đặt trong tập hợp tham số hình ảnh.

2. Giải mã thông tin mà sự phân đoạn không gian là đồng đều từ một hoặc nhiều phần tử cú pháp trong luồng bit. Cú pháp tốt hơn là được đặt trong tập hợp tham số hình ảnh. Cú pháp có thể bao gồm cờ một bit đặc tả việc sự phân đoạn không gian là đồng đều hay không.

3. Xác định kích thước đơn vị phân đoạn S hoặc từ một hoặc nhiều phần tử cú pháp hoặc bằng cách sử dụng kích thước đơn vị phân đoạn định trước. Nếu kích thước đơn vị định trước được sử dụng thì có thể là bằng đơn vị cây mã hóa. Ví dụ, nếu kích thước CTU là bằng 128x128, thì kích thước đơn vị phân đoạn S có thể là bằng 128 (hoặc 128x128).

4. Tính toán kích thước của hình ảnh theo số đơn vị phân đoạn. Việc tính toán có thể được thực hiện riêng biệt cho chiều cao và chiều rộng sao cho kích thước ngang HS được thiết đặt bằng chiều rộng hình ảnh trong các mẫu độ chói được chia cho kích thước đơn vị phân đoạn. Kích thước dọc VS có thể được thiết đặt bằng chiều cao hình ảnh trong các mẫu độ chói được chia cho kích thước đơn vị phân đoạn.

5. Giải mã giá trị TileWidth biểu diễn chiều rộng ô xếp kiểu ngói theo các đơn vị S từ từ mã trong luồng bit. Từ mã có thể là từ mã UVLC. Từ mã có thể được giải mã như chiều rộng ô xếp kiểu ngói trừ đi một và giá trị TileWidth có thể do đó được thiết đặt bằng giá trị được giải mã cộng 1.

6. Giải mã giá trị TileHeight biểu diễn chiều cao ô xếp kiểu ngói theo các đơn vị S từ từ mã trong luồng bit. Từ mã có thể là từ mã UVLC. Từ mã có thể được giải mã như chiều cao ô xếp kiểu ngói trừ đi một và giá trị TileWidth có thể do đó được thiết đặt bằng giá trị được giải mã cộng 1.

7. Suy ra các chiều rộng cột phân đoạn từ chiều rộng hình ảnh theo số đơn vị ô xếp kiểu ngói và biến số TileWidth bằng các bước con sau đây:

a. Thiết đặt chiều rộng hình ảnh vẫn chưa được phân đoạn (A) bằng giá trị HS

b. Thiết đặt biến số i bằng giá trị 0

c. Thực thi lặp lại các bước con sau đây với điều kiện chiều rộng hình ảnh vẫn chưa được phân đoạn (A) là lớn hơn so với giá trị TileWidth:

i. Thiết đặt chiều rộng cột của cột phân đoạn là TileWidth

ii. Trừ giá trị A cho giá trị TileWidth

iii. Tăng giá trị của biến số i thêm 1

d. Thiết đặt chiều rộng cột của cột phân đoạn i-th là giá trị A

e. Thiết đặt biến số num_tile_columns_minus1 bằng giá trị của biến số i, hoặc thay vào đó, thiết đặt biến số biểu diễn số cột ô xếp kiểu ngói trong hình ảnh bằng giá trị của biến số i cộng 1

8. Suy ra các chiều cao hàng phân đoạn từ chiều cao hình ảnh theo số đơn vị ô xếp kiểu ngói và biến số TileHeight bằng các bước con sau đây:

a. Thiết đặt chiều cao hình ảnh vẫn chưa được phân đoạn (A) bằng giá trị VS

b. Thiết đặt biến số i bằng giá trị 0

c. Thực thi lặp lại các bước con sau đây với điều kiện chiều cao hình ảnh vẫn chưa được phân đoạn (A) là lớn hơn so với giá trị TileHeight:

i. Thiết đặt chiều cao hàng của hàng phân đoạn i-th là TileHeight

ii. Lấy giá trị A trừ đi giá trị TileHeight

iii. Tăng giá trị của biến số i thêm 1

d. Thiết đặt chiều cao hàng của hàng phân đoạn i-th là giá trị A

e. Thiết đặt biến số num_tile_rows_minus1 bằng giá trị của biến số i, hoặc thay vào đó, thiết đặt biến số biểu diễn số hàng ô xếp kiểu ngói trong hình ảnh bằng giá trị của biến số i cộng 1

9. Suy ra vị trí không gian cho khối hiện tại sử dụng các chiều rộng phân đoạn được suy ra và các chiều cao phân đoạn được suy ra.

10. Giải mã khối hiện tại sử dụng vị trí không gian được suy ra. Lưu trữ các giá trị mẫu được giải mã cho khối hiện tại trong bộ nhớ ở các vị trí của bộ nhớ tương ứng với vị trí không gian được suy ra.

Phương án 6.

Ở phương án thứ sáu, một trong nhiều phương pháp để phân chia ô xếp kiểu ngói đồng đều có thể được sử dụng. Ở một phương án, có một hoặc nhiều phần tử cú pháp chỉ báo một trong nhiều phương pháp nào để sử dụng. Một hoặc nhiều phần tử cú pháp có thể là cờ đặc tả phương pháp nào trong hai phương pháp mà được sử dụng. Một hoặc nhiều phần tử cú pháp có thể là có mặt trong tập hợp tham số như tập hợp tham số hình ảnh nhưng phương án này không bị ràng buộc ở chỉ vị trí đó. Tập hợp nhiều phương pháp có thể bao gồm phương pháp bất kỳ được bộc lộ ở đây cũng như các phương pháp đã biết trong tình trạng kỹ thuật như các phương pháp được mô tả trong đặc tả HEVC v5 và đặc tả VVC dự thảo.

Phương án 7. Các ô nhỏ sang bên trái hoặc phía trên

Ở các phương án 5 đến 6 được mô tả ở trên trong sáng chế này, các ô kiểu xếp ngói được xếp theo trật tự sao cho các ô kiểu xếp ngói có kích thước đầy đủ được đặt sang bên trái và phía trên trong cấu trúc ô xếp kiểu ngói của hình ảnh.

Ở phương án thứ bảy, các ô kiểu xếp ngói được đặt trật tự theo cách ngược như trong các phương án 5 đến 6, tức là phần còn dư của chiều rộng ô xếp kiểu ngói là chiều rộng của ô xếp kiểu ngói (các ô xếp kiểu ngói) cực trái trong cấu trúc ô kiểu xếp ngói đồng đều và/hoặc phần còn dư của chiều cao ô xếp kiểu ngói là chiều cao của ô xếp kiểu ngói (các ô xếp kiểu ngói) phía trên cùng trong cấu trúc ô xếp kiểu ngói đồng đều. Nói cách khác, các ô xếp kiểu ngói có kích thước đầy đủ được đặt sang bên phải và/hoặc phía dưới và các ô kiểu xếp ngói được đặt kích thước nhỏ hơn với chiều rộng bằng phần còn dư của chiều rộng ô xếp kiểu ngói và/hoặc với chiều cao bằng chiều cao ô xếp kiểu ngói được đặt sang bên trái và/hoặc phía trên trong cấu cấu ô xếp kiểu ngói. Điều này được minh họa ở Fig.13 với hình ảnh 1300 mà là 896x640 điểm ảnh. Các đường liền biểu diễn các biên ô xếp kiểu ngói và các đường gạch ngang biểu diễn các biên CTU. Mỗi CTU là 64x64 điểm ảnh. Fig.13 thể hiện ví dụ về cấu trúc ô kiểu xếp ngói đồng đều với các ô xếp kiểu ngói có kích thước đầy đủ sang bên dưới và bên phải và các ô xếp kiểu ngói nhỏ hơn được đặt sang bên trái và phía trên.

Phương án 8. Sắp thẳng hàng với các biên nhóm ô xếp kiểu ngói

Ở phương án thứ tám, các kích thước của các ô kiểu xếp ngói được xếp trật tự theo phương án bất kỳ trong các phương án trước có sự bổ sung ở chỗ các ô kiểu xếp ngói được nhóm thành các nhóm ô xếp kiểu ngói trong đó mỗi nhóm ô xếp kiểu ngói có thể chứa hàng/cột của các ô kiểu xếp ngói với chiều rộng/chiều rộng ô xếp kiểu ngói bằng phần còn dư của chiều cao/chiều rộng ô xếp kiểu ngói.

Điều này được minh họa ở Fig.14 trong nhóm ô xếp kiểu ngói được biểu diễn bằng các đường chấm lớn, các ô kiểu xếp ngói được biểu diễn bằng các đường liền và các CTU được biểu diễn bằng các đường gạch ngang. Lưu ý rằng không phải tất cả các nhóm ô xếp kiểu ngói có thể có các ô kiểu xếp ngói có phần còn dư của chiều rộng ô xếp kiểu ngói ngang bằng điều mà phụ thuộc vào việc tổng phần còn dư của chiều rộng là có thể phân chia ngang bằng cho số CTU hay không. Fig.14 thể hiện ví dụ về cấu trúc ô kiểu xếp ngói với các nhóm ô xếp kiểu ngói trong đó mỗi nhóm ô xếp kiểu ngói có thể chứa ô xếp kiểu ngói có chiều rộng và/hoặc chiều cao bằng giá trị phần còn dư của chiều rộng/chiều cao ô xếp kiểu ngói.

Trong ví dụ được thể hiện ở Fig.14, chiều rộng nhóm ô xếp kiểu ngói là 5 CTU và chiều cao ô xếp kiểu ngói là 2 CTU. Bởi vì chiều rộng của hình ảnh là 14 CTU, nên hình ảnh được chia thành 3 nhóm ô xếp kiểu ngói theo hướng ngang, lần lượt với 5, 5, và 4 ô kiểu xếp ngói. Hai nhóm ô xếp kiểu ngói đầu tiên được chia theo hướng ngang thành hai ô kiểu xếp ngói có kích thước đầy đủ bằng 2 CTU và một ô xếp kiểu ngói có chiều rộng ô xếp kiểu ngói còn dư bằng 1 CTU. Nhóm ô xếp kiểu ngói cực trái có thể chia ngang bằng được cho 2 CTU để không có ô xếp kiểu ngói nào có chiều rộng ô xếp kiểu ngói còn dư là được yêu cầu. Có thể có kịch bản với các ô kiểu xếp ngói lớn hơn chứa nhiều CTU hơn (ví dụ, 4 ô kiểu xếp ngói theo hướng ngang) trong đó các phần còn dư sẽ là 3, 3, 2, tức là, tất cả các nhóm ô xếp kiểu ngói có các ô kiểu xếp ngói có phần còn dư của chiều rộng ô xếp kiểu ngói, nhưng phần còn dư của chiều rộng ô xếp kiểu ngói là không giống nhau cho tất cả các nhóm ô xếp kiểu ngói.

Fig.15 là sơ đồ tiến trình minh họa quy trình 1500 theo một phương án. Quy trình 1500 là phương pháp để giải mã hình ảnh. Phương pháp 1500 bao gồm giải mã thông tin

mà hình ảnh được phân chia thành nhiều hơn một phân đoạn dựa vào một hoặc nhiều phần tử cú pháp luồng bit (bước 1502); giải mã thông tin về việc sự phân đoạn không gian là đồng đều dựa vào một hoặc nhiều phần tử cú pháp trong luồng bit (bước 1504); xác định kích thước đơn vị phân đoạn dựa vào một hoặc nhiều phần tử cú pháp hoặc dựa vào kích thước đơn vị phân đoạn định trước (bước 1506); giải mã giá trị thứ nhất chỉ báo chiều rộng phân đoạn từ một hoặc nhiều từ mã trong luồng bit (bước 1508); giải mã giá trị thứ hai chỉ báo chiều cao phân đoạn từ một hoặc nhiều từ mã trong luồng bit (bước 1510); suy ra các chiều rộng cột phân đoạn dựa vào chiều rộng hình ảnh theo số đơn vị phân đoạn và giá trị thứ nhất (bước 1512); suy ra các chiều cao hàng phân đoạn dựa vào chiều cao hình ảnh theo số đơn vị phân đoạn và giá trị thứ hai (bước 1514); suy ra vị trí không gian cho khối hiện tại dựa vào các chiều rộng cột phân đoạn được suy ra (bước 1516); và giải mã khối hiện tại dựa vào vị trí không gian được suy ra (bước 1518).

Ở một số phương án, bước suy ra các chiều rộng cột phân đoạn bao gồm việc thiết đặt các giá trị chiều rộng của tất cả các cột phân đoạn trong hình ảnh ngoại trừ một cột bằng giá trị thứ nhất, và thiết đặt giá trị chiều rộng cột của một cột phân đoạn còn dư bằng chiều rộng hình ảnh trừ tổng của các giá trị chiều rộng của tất cả cột phân đoạn ngoại trừ một cột phân đoạn.

Ở một số phương án, bước suy ra các chiều cao hàng phân đoạn bao gồm việc thiết đặt các giá trị chiều cao hàng của tất cả các hàng phân đoạn trong hình ảnh ngoại trừ một hàng bằng giá trị thứ hai, và thiết đặt giá trị chiều cao hàng của một hàng phân đoạn còn dư bằng chiều cao hình ảnh trừ tổng của các giá trị chiều cao của tất cả hàng phân đoạn ngoại trừ một hàng phân đoạn.

Ở một số phương án, một hoặc nhiều phần tử cú pháp được đặt trong tập hợp tham số hình ảnh.

Ở một số phương án, một hoặc nhiều phần tử cú pháp bao gồm cờ một bit xác định xem sự phân đoạn không gian là đồng đều hay không.

Ở một số phương án, kích thước đơn vị phân đoạn định trước là bằng đơn vị cây mã hóa.

Ở một số phương án, việc tính toán kích thước của hình ảnh trong phân đoạn bao gồm: việc tính toán kích thước của phân đoạn hình ảnh cho chiều cao và việc tính toán kích thước của phân đoạn hình ảnh cho chiều rộng.

Ở một số phương án, giá trị thứ nhất là giá trị TileWidth biểu diễn chiều rộng ô xếp kiểu ngói theo các đơn vị phân đoạn và giá trị thứ hai là TileHeight biểu diễn chiều cao ô xếp kiểu ngói theo các đơn vị phân đoạn.

Fig.16 là sơ đồ thể hiện các khối chức năng của bộ giải mã 1602 theo một số phương án. Như được thể hiện ở Fig.16, bộ giải mã 1602 bao gồm khối giải mã thứ nhất 1604 để giải mã thông tin về việc hình ảnh được phân chia thành nhiều hơn một phân đoạn dựa vào một hoặc nhiều phần tử cú pháp trong luồng bit hay không; khối giải mã thứ hai 1606 để giải mã thông tin mà sự phân đoạn không gian là đồng đều hay không dựa vào một hoặc nhiều phần tử cú pháp trong luồng bit; khối xác định 1608 để xác định kích thước đơn vị phân đoạn dựa vào một hoặc nhiều phần tử cú pháp hoặc dựa vào kích thước đơn vị phân đoạn định trước; khối giải mã thứ ba 1610 để giải mã giá trị thứ nhất chỉ báo chiều rộng phân đoạn từ một hoặc nhiều từ mã luồng bit; khối giải mã thứ tư 1612 để giải mã giá trị thứ hai chỉ báo chiều cao phân đoạn từ một hoặc nhiều từ mã trong luồng bit; khối suy ra thứ nhất 1614 để suy ra các chiều rộng cột phân đoạn dựa vào chiều rộng hình ảnh theo số đơn vị phân đoạn và giá trị thứ nhất; khối suy ra thứ hai 1616 để suy ra các chiều cao hàng phân đoạn dựa vào chiều cao hình ảnh theo số đơn vị phân đoạn và giá trị thứ hai; khối suy ra thứ ba 1618 để suy ra vị trí không gian cho khối hiện tại dựa vào các chiều rộng cột phân đoạn được suy ra và các chiều cao phân đoạn được suy ra; và khối giải mã thứ năm 1620 để giải mã khối hiện tại dựa vào vị trí không gian được suy ra.

Fig.7 là sơ đồ tiến trình minh họa quy trình 700 theo một phương án. Quy trình 700 là phương pháp để giải mã hình ảnh 10 bao gồm số đơn vị 8 từ luồng bit, hình ảnh được phân chia thành ít nhất hai phân đoạn không gian 11 bằng cấu trúc phân chia 13. Phương

pháp bao gồm giải mã một hoặc nhiều từ mã trong luồng bit (bước 710); xác định cấu trúc phân chia là đồng đều dựa vào một hoặc nhiều từ mã (bước 720); xác định số phân đoạn không gian dựa vào một hoặc nhiều từ mã (bước 730); xác định kích thước đơn vị phân đoạn (bước 740); và suy ra các kích thước và/hoặc vị trí cho tất cả các phân đoạn không gian trong hình ảnh từ một hoặc nhiều từ mã (bước 750), trong đó việc suy ra các kích thước và/hoặc vị trí phân đoạn bao gồm vòng thứ nhất trên số phân đoạn không gian theo chiều hoặc hướng thứ nhất, và số đơn vị phân đoạn còn dư theo chiều hoặc hướng thứ nhất để được phân đoạn và số phân đoạn còn dư theo chiều thứ nhất được tính toán bên trong vòng thứ nhất.

Ở một số phương án, việc suy ra các kích thước và/hoặc vị trí phân đoạn bao gồm vòng thứ hai trên số phân đoạn không gian theo chiều hoặc hướng thứ hai chứ không phải chiều hoặc hướng thứ nhất, và số đơn vị phân đoạn còn dư theo chiều hoặc hướng thứ hai để được phân đoạn và số phân đoạn còn dư theo chiều hoặc hướng thứ hai được tính toán bên trong vòng thứ hai.

Ở một số phương án, chiều hoặc hướng thứ nhất là chiều hoặc hướng ngang và chiều hoặc hướng thứ hai là chiều hoặc hướng dọc.

Ở một số phương án, chiều rộng của các phân đoạn trong hàng hoặc chiều cao của các phân đoạn trong một cột theo trật tự được xác định.

Ở một số phương án, chiều rộng của các phân đoạn trong hàng hoặc chiều cao của các phân đoạn trong cột là ở trật tự không bao giờ tăng dần hoặc ở trật tự không bao giờ giảm dần.

Ở một số phương án, sự phân chia phân đoạn nhị phân với việc đặt trật tự rõ ràng đối với chiều rộng hoặc chiều cao phân đoạn được áp dụng cho chiều rộng hoặc chiều cao của hình ảnh.

Ở một số phương án, chiều rộng của các phân đoạn trong hàng hoặc chiều cao của các phân đoạn trong cột của hình ảnh theo trật tự mặc định được xác định cho chiều rộng hoặc chiều cao của các phân đoạn.

Ở một số phương án, kích thước đơn vị phân đoạn là bằng kích thước của đơn vị cây mã hóa (Coding tree unit - CTU).

Ở một số phương án, kích thước đơn vị phân đoạn là nhỏ hơn so với kích thước của đơn vị cây mã hóa (CTU). Ở một số phương án, kích thước đơn vị phân đoạn là lớn hơn so với kích thước của CTU. Ở một số phương án, phân đoạn bất kỳ là ô xếp kiểu ngói.

Ở một số phương án, việc suy ra các kích thước bao gồm:

suy ra danh sách Sizes[] như:

```
A = PicWidthInTileUnits
B = NumberOfSegmentColumns
for( i = 0; i < NumberOfSegmentColumns; i++ ) {
 Sizes[ i ] = Round( A ÷ B)
 A = A - Sizes[ i ]
 B = B - 1
}
```

trong đó NumberOfSegmentColumns là số cột phân đoạn, PicWidthInTileUnits là chiều rộng của hình ảnh theo các đơn vị ô xếp kiểu ngói, Round() hoặc là hàm Floor() hoặc là hàm Ceil(), và ÷ là tính chia không chặt cụt hay làm tròn.

Ở một số phương án, việc suy ra các kích thước bao gồm:

suy ra danh sách Sizes[] như:

```
A = PicHeightInTileUnits
B = NumberOfSegmentRows
for( i = 0; i < NumberOfSegmentRows; i++ ) {
 Sizes[ i ] = Round( A ÷ B)
 A = A - Sizes[ i ]
 B = B - 1
}
```

trong đó NumberOfSegmentRows là số hàng phân đoạn, PicHeightInTileUnits là chiều cao của hình ảnh theo các đơn vị ô xếp kiểu ngói, Round() hoặc là hàm Floor() hoặc là hàm Ceil(), và ÷ là phép chia không có chặt cụt hoặc làm tròn.

Ở một số phương án, các phân đoạn 11 là không phụ thuộc đối với các phân đoạn 11 khác sao cho việc suy ra chế độ nội dự đoán bất kỳ cho đơn vị bất kỳ 16 trong phân đoạn hiện tại 15 phụ thuộc chỉ vào các chế độ nội dự đoán được suy ra trước đó 17 mà thuộc về phân đoạn hiện tại 15 và không phụ thuộc vào chế độ nội dự đoán bất kỳ trong đơn vị bất kỳ 18 mà thuộc về phân đoạn khác nhau 14.

Ở một số phương án, phương án bao gồm bước thêm nữa để phân chia các phân đoạn đến các chiều rộng hoặc chiều cao đồng đều theo mẫu gợn sóng cố định độc lập với số phân đoạn không gian.

Fig.8 là sơ đồ tiến trình minh họa quy trình 800 theo một phương án. Quy trình 800 là phương pháp để mã hóa hình ảnh 10 bao gồm số đơn vị 8 thành luồng bit, hình ảnh được phân chia thành ít nhất hai phân đoạn không gian 11 bằng cấu trúc phân chia đồng đều 13. Phương pháp bao gồm mã hóa thông tin về việc cấu trúc phân chia 13 là đồng đều bằng cách mã hóa một hoặc nhiều từ mã thành luồng bit (bước 810); mã hóa số phân đoạn không gian bằng cách mã hóa một hoặc nhiều từ mã thành luồng bit (bước 820); xác định kích thước đơn vị phân đoạn (bước 830); và suy ra và mã hóa các kích thước và/hoặc các vị trí cho tất cả các phân đoạn không gian trong hình ảnh thành luồng bit (bước 840), trong đó việ suy ra các kích thước phân đoạn bao gồm vòng thứ nhất trên số phân đoạn không gian theo chiều hoặc hướng thứ nhất, và số đơn vị phân đoạn còn dư theo chiều hoặc kích thước thứ nhất cần được phân đoạn và số phân đoạn còn dư theo chiều hoặc hướng thứ nhất được tính toán bên trong vòng thứ nhất.

Ở một số phương án, việc suy ra các kích thước và/hoặc vị trí phân đoạn bao gồm vòng thứ hai trên số phân đoạn không gian theo chiều hoặc hướng thứ hai chứ không phải chiều hoặc hướng thứ nhất, và số đơn vị phân đoạn còn dư theo chiều hoặc hướng thứ hai để được phân đoạn và số phân đoạn còn dư theo chiều hoặc hướng thứ hai được tính toán bên trong vòng thứ hai.

Ở một số phương án, chiều hoặc hướng thứ nhất là chiều hoặc hướng ngang và chiều hoặc hướng thứ hai là chiều hoặc hướng dọc.

Fig.9 là sơ đồ thể hiện các khối chức năng của bộ giải mã 902 theo một số phương án. Như được thể hiện ở Fig.9, bộ giải mã 902 bao gồm khối giải mã 904 để giải mã một hoặc nhiều từ mã trong luồng bit; khối xác định thứ nhất 906 để xác định cấu trúc phân chia là đồng đều dựa vào một hoặc nhiều từ mã; khối xác định thứ hai 908 để xác định số phân đoạn không gian dựa vào một hoặc nhiều từ mã; khối xác định thứ ba 910 để xác định kích thước đơn vị phân đoạn; và khối suy ra 912 để suy ra các kích thước và/hoặc vị trí cho tất cả các phân đoạn không gian trong hình ảnh từ một hoặc nhiều từ mã, trong đó việc suy ra các kích thước và/hoặc vị trí phân đoạn bao gồm vòng thứ nhất trên số phân đoạn không gian theo chiều thứ nhất và vòng thứ hai trên số phân đoạn không gian theo chiều thứ hai, số đơn vị phân đoạn còn dư theo chiều thứ nhất để được phân đoạn và số phân đoạn còn dư theo chiều thứ nhất được tính toán bên trong vòng thứ nhất, và số đơn vị phân đoạn còn dư theo chiều thứ hai để được phân đoạn và số phân đoạn còn dư theo chiều thứ hai được tính toán bên trong vòng thứ hai.

Fig.10 là sơ đồ thể hiện các khối chức năng của bộ mã hóa 1002 theo một số phương án. Như được thể hiện ở Fig.10, bộ mã hóa 1002 bao gồm khối mã hóa thứ nhất 1004 để mã hóa thông tin về cấu trúc phân chia 13 là đồng nhất bằng cách mã hóa một hoặc nhiều từ mã trong luồng bit; khối mã hóa thứ hai 1006 để mã hóa số phân đoạn không gian bằng cách mã hóa một hoặc nhiều từ mã trong luồng bit; khối xác định 1008 để xác định kích thước đơn vị phân đoạn; và khối suy ra 1010 để suy ra và mã hóa các kích thước và/hoặc các vị trí cho tất cả các phân đoạn không gian trong hình ảnh vào trong luồng bit, trong đó việc suy ra các kích thước phân đoạn bao gồm vòng thứ nhất trên số phân đoạn không gian theo chiều thứ nhất và vòng thứ hai trên số phân đoạn không gian theo chiều thứ hai, số đơn vị phân đoạn còn dư theo chiều thứ nhất để được phân đoạn và số phân đoạn còn dư theo chiều thứ nhất được tính toán bên trong vòng thứ nhất, và số đơn vị phân đoạn còn dư theo chiều thứ hai để được phân đoạn và số phân đoạn còn dư theo chiều thứ hai được tính toán bên trong vòng thứ hai.

Ở một số phương án, bộ mã hóa được tạo cấu hình để xác định cấu trúc phân chia mà phân chia hình ảnh thành số phân đoạn không gian đồng đều, trong đó mỗi phân đoạn không gian bao gồm ít nhất một đơn vị, và trong đó việc suy ra các kích thước phân đoạn được thực hiện trong vòng thứ nhất trên số phân đoạn không gian theo chiều thứ nhất và vòng thứ hai trên số phân đoạn không gian theo chiều thứ hai trong đó số đơn vị phân đoạn còn dư theo chiều thứ nhất để được phân đoạn và số phân đoạn còn dư theo chiều thứ nhất được tính toán bên trong vòng thứ nhất, và trong đó số đơn vị phân đoạn còn dư theo chiều thứ hai để được phân đoạn và số phân đoạn còn dư theo chiều thứ hai được tính toán bên trong vòng thứ hai; mã hóa nhiều phân đoạn không gian theo cấu trúc phân chia để tạo ra nhiều phân đoạn không gian được mã hóa, trong đó mỗi phân đoạn không gian được mã hóa tương ứng với một trong các phân đoạn không gian của cấu trúc phân chia, và mỗi phân đoạn không gian được mã hóa là độc lập sao cho việc suy ra chế độ nội dự đoán bất kỳ cho đơn vị thứ nhất của phân đoạn không gian thứ nhất phụ thuộc vào chế độ nội dự đoán được suy ra cho đơn vị thứ hai của phân đoạn không gian thứ nhất và độc lập với chế độ nội dự đoán bất kỳ cho các đơn vị của các phân đoạn không gian khác của cấu trúc phân chia; và tạo ra luồng bit bao gồm nhiều phân đoạn không gian được mã hóa và thông tin chỉ báo cấu trúc phân chia đồng đều nào được sử dụng để phân chia hình ảnh thành nhiều phân đoạn không gian.

Fig.11 là sơ đồ khối của thiết bị 1100 để thực hiện bộ giải mã 902, 1602 và/hoặc bộ mã hóa 1002, theo một số phương án. Khi thiết bị 1100 thực hiện bộ giải mã, thì thiết bị 1100 có thể được đề cập đến như "thiết bị giải mã 1100," và khi thiết bị 1100 thực hiện bộ mã hóa, thì thiết bị 1100 có thể được đề cập đến như "thiết bị mã hóa 1100." Như được thể hiện ở FIG. 11, thiết bị 1100 (còn được biết đến như "nút") có thể bao gồm: hệ mạch xử lý (Processing circuitry - PC) 1102, vốn có thể bao gồm một hoặc nhiều bộ xử lý (Processor - P) 1155 (ví dụ, bộ vi xử lý đa dụng và/hoặc một hoặc nhiều bộ xử lý khác, như mạch tích hợp chuyên ứng dụng (Application specific integrated circuit - ASIC), các mảng cổng có thể lập trình được trường (Field-programmable gate array - FPGA), và tương tự); giao diện mạng 1148 bao gồm bộ truyền (Transmitter - Tx) 1145 và bộ thu (Receiver - Rx) 1147 để cho phép thiết bị 1100 để truyền dữ liệu đến và

nhận dữ liệu từ các nút khác được nối vào mạng 1110 (ví dụ, mạng giao thức Internet (Internet Protocol - IP)) mà giao diện mạng 1148 được nối vào; và khối lưu trữ cục bộ (còn được biết đến như "hệ thống lưu trữ dữ liệu") 1108, mà có thể bao gồm một hoặc nhiều thiết bị bộ nhớ không khả biến và/hoặc một hoặc nhiều thiết bị lưu trữ khả biến. Ở các phương án trong đó PC 1102 bao gồm bộ xử lý có thể lập trình được, sản phẩm chương trình máy tính (Computer program product - CPP) 1141 có thể được bố trí. CPP 1141 bao gồm phương tiện đọc được bằng máy tính (Computer readable medium - CRM) 1142 lưu trữ chương trình máy tính (Computer program - CP) 1143 bao gồm các lệnh đọc được bằng máy tính (Computer readable instruction - CRI) 1144. CRM 1142 có thể là phương tiện đọc được bằng máy tính phi chuyển tiếp (ví dụ, đĩa cứng), các phương tiện quang học, các thiết bị nhớ (ví dụ, bộ nhớ truy cập ngẫu nhiên, bộ nhớ tác động nhanh) và tương tự. Theo một số phương án, CRI 1144 của chương trình máy tính 1143 được tạo cấu hình sao cho khi được thực hiện bởi PC 1102, CRI khiến cho thiết bị 1100 thực hiện các bước được mô tả ở trên (ví dụ, các bước được mô tả ở đây với việc tham chiếu đến các sơ đồ tiến trình). Ở các phương án khác, thiết bị 1100 có thể được tạo cấu hình để thực hiện các bước được mô tả ở đây mà không có nhu cầu về mã. Tức là, ví dụ, PC 1102 có thể bao gồm chỉ một hoặc nhiều ASIC. Ở đây, các đặc điểm của các phương án được mô tả ở đây có thể được thực hiện trong phần cứng và/hoặc phần mềm.

Tóm tắt của các phương án khác nhau

A1. Phương pháp để giải mã hình ảnh, phương pháp bao gồm: giải mã thông tin về hình ảnh được phân chia thành nhiều hơn một phân đoạn dựa vào một hoặc nhiều phần tử cú pháp trong luồng bit; giải mã thông tin về sự phân đoạn không gian là đồng đều dựa vào một hoặc nhiều phần tử cú pháp trong luồng bit; xác định kích thước đơn vị phân đoạn dựa vào một hoặc nhiều phần tử cú pháp hoặc dựa vào kích thước đơn vị phân đoạn định trước; giải mã giá trị thứ nhất chỉ báo chiều rộng phân đoạn từ một hoặc nhiều từ mã trong luồng bit; giải mã giá trị thứ hai chỉ báo chiều cao từ một hoặc nhiều từ mã trong luồng bit; suy ra các chiều rộng cột phân đoạn dựa vào chiều rộng hình ảnh theo số đơn vị phân đoạn và giá trị thứ nhất; suy ra các chiều cao hàng phân đoạn dựa vào

chiều cao hình ảnh theo số đơn vị phân đoạn và giá trị thứ hai; suy ra vị trí không gian cho khối hiện tại dựa vào các chiều rộng cột phân đoạn được suy ra và các chiều cao hàng phân đoạn được phân đoạn; và giải mã khối hiện tại dựa vào vị trí không gian được suy ra.

A2. Phương pháp của phương án A1, trong đó việc suy ra các chiều rộng cột phân đoạn bao gồm: thiết đặt các giá trị chiều rộng cột của tất cả các cột phân đoạn trong hình ảnh ngoại trừ một cột bằng giá trị thứ nhất, và thiết đặt giá trị chiều rộng cột của một cột phân đoạn còn dư thành chiều rộng hình ảnh trừ tổng của các giá trị chiều rộng của tất cả các cột phân đoạn ngoại trừ một cột phân đoạn.

A3. Phương pháp theo phương án A1 hoặc A2, trong đó bước suy ra các chiều cao hàng phân đoạn bao gồm: việc thiết đặt các giá trị chiều cao hàng của tất cả các hàng phân đoạn trong hình ảnh ngoại trừ một hàng bằng giá trị thứ hai, và thiết đặt giá trị chiều cao hàng của một hàng phân đoạn còn dư bằng chiều cao hình ảnh trừ tổng của các giá trị chiều cao của tất cả hàng phân đoạn ngoại trừ một hàng phân đoạn.

A4. Phương pháp theo phương án bất kỳ trong số các phương án từ A1-A3, trong đó một hoặc nhiều phần tử cú pháp được đặt trong tập hợp tham số hình ảnh.

A5. Phương pháp theo phương án bất kỳ trong số các phương án từ A1 đến A4, trong đó một hoặc nhiều phần tử cú pháp bao gồm cờ một bit xác định xem sự phân đoạn không gian là đồng đều hay không.

A6. Phương pháp theo phương án bất kỳ trong số các phương án từ A1 đến A5, trong đó đơn vị phân đoạn là đơn vị cây mã hóa hoặc khối cây mã hóa.

A7. Phương pháp theo phương án bất kỳ trong số các phương án từ A1 đến A6, còn bao gồm bước tính toán kích thước của hình ảnh theo số đơn vị phân đoạn.

A8. Phương pháp theo phương án A7, trong đó bước tính toán kích thước của hình ảnh theo số đơn vị phân đoạn bao gồm: việc tính toán kích thước của phân đoạn hình ảnh cho chiều cao và việc tính toán kích thước của phân đoạn hình ảnh cho chiều rộng.

A9. Phương pháp theo phương án bất kỳ trong số các phương án từ A1 đến A8,

trong đó giá trị thứ nhất biểu diễn chiều rộng ô xếp kiểu ngói theo các đơn vị phân đoạn và giá trị thứ hai biểu diễn chiều cao ô xếp kiểu ngói theo các đơn vị phân đoạn.

A10. Phương pháp theo phương án bất kỳ trong số các phương án từ A1 đến A9, trong đó bước giải mã giá trị thứ nhất từ một hoặc nhiều từ mã trong luồng bit bao gồm việc giải mã giá trị của từ mã cụ thể trong luồng bit và thêm 1 vào giá trị được giải mã.

A11. Phương pháp của theo phương án bất kỳ trong số các phương án từ A1 đến A10, trong đó bước suy ra các chiều rộng cột phân đoạn bao gồm:

thiết đặt biến số, A, thành giá trị chiều rộng còn dư; và

xác định xem giá trị bằng (A - TileWidth) là thấp hơn TileWidth hay không, trong đó TileWidth là bằng giá trị thứ nhất.

A12. Phương pháp theo phương án A11, trong đó nếu xác định được rằng giá trị của (A - TileWidth) là không thấp hơn giá trị thứ nhất, thì sau đó thực hiện các bước gồm:

thiết đặt biến số chiều rộng cột thứ nhất bằng TileWidth; và

xác định xem giá trị bằng (A - (2 x TileWidth)) là thấp hơn TileWidth hay không.

A13. Phương pháp theo phương án A11 hoặc A12, trong đó việc xác định xem giá trị (A - TileWidth) là thấp hơn TileWidth hay không bao gồm:

thiết đặt A bằng (A - TileWidth); và

so sánh A với TileWidth.

A14. Phương pháp theo phương án A12 hoặc A13, trong đó nếu xác định được rằng A - TileWidth là thấp hơn TileWidth, thì sau đó thiết đặt biến số chiều rộng cột thứ hai bằng A - TileWidth.

A15. Phương pháp theo phương án A15, trong đó việc thiết đặt biến số chiều rộng cột thứ hai bằng A - TileWidth chỉ được thực hiện nếu A - TileWidth là lớn hơn 0.

A16. Phương pháp theo phương án A14 hoặc A15, còn bao gồm việc thiết đặt biến số biểu diễn số cột ô xếp kiểu ngói trong hình ảnh bằng giá trị của biến số i hoặc bằng giá trị của biến số i cộng 1, trong đó i là số nguyên sao cho giá trị (A - ((i-1) x TileWidth)) là không thấp hơn TileWidth nhưng giá trị (A - (i x TileWidth)) là thấp hơn TileWidth.

A17. Phương pháp của theo phương án bất kỳ trong số các phương án từ A1 đến A10, trong đó bước suy ra các chiều rộng cột phân đoạn bao gồm:

1) thiết đặt biến số, A, thành giá trị chiều rộng còn dư; và

2) thiết đặt biến số, i, thành giá trị ban đầu;

3) xác định xem giá trị bằng (A - (i x TileWidth) là thấp hơn TileWidth hay không, trong đó TileWidth là bằng giá trị thứ nhất; và

4) nếu xác định được rằng giá trị bằng (A - (i x TileWidth)) là không thấp hơn TileWidth, thì sau đó thực hiện các bước gồm: thiết đặt biến số col_width[i] bằng TileWidth, tăng i, và lặp lại các bước 3) và 4).

A18. Phương pháp theo phương án A17, trong đó sau khi xác định được rằng giá bằng (A - (i x TileWidth)) là thấp hơn TileWidth, thì sau đó thực hiện việc thiết đặt col_width[i] bằng A và tăng i.

Trong khi các phương án khác nhau được mô tả ở đây (bao gồm phụ lục nếu có), cần phải hiểu rằng chúng được thể hiện chỉ nhằm ví dụ, và không làm hạn chế. Do đó, độ rộng và phạm vi của sáng chế này không nên bị hạn chế bởi phương án bất kỳ trong số các phương án để làm ví dụ được mô tả ở trên. Ngoài ra, kết hợp bất kỳ của các yếu tố được mô tả trên đây trong tất cả các thay đổi có thể có của các yếu tố này cũng nằm trong phần bộc lộ này trừ khi có quy định khác hoặc rõ ràng là trái với ngữ cảnh đó.

Ngoài ra, trong khi các quy trình được mô tả ở trên và được minh họa trong các hình vẽ được thể hiện như là trình tự của các bước, nhưng nó có thể được thực hiện đơn lẻ để minh họa. Do đó, nó được dự tính rằng một số bước có thể được bổ sung, một số bước có thể được bỏ qua, trật tự của các bước có thể được sắp xếp lại, và một số bước có thể được thực hiện song song.

Yêu cầu bảo hộ

1. Phương pháp (1500) để giải mã hình ảnh (10), phương pháp bao gồm các bước:

giải mã (1502) thông tin về hình ảnh được phân chia thành nhiều hơn một phân đoạn dựa vào một hoặc nhiều phần tử cú pháp trong luồng bit;

giải mã (1504) thông tin về sự phân đoạn không gian là đồng đều dựa vào một hoặc nhiều phần tử cú pháp trong luồng bit;

xác định (1506) kích thước đơn vị phân đoạn dựa vào một hoặc nhiều phần tử cú pháp hoặc dựa vào kích thước đơn vị phân đoạn định trước;

giải mã (1508) giá trị thứ nhất chỉ báo chiều rộng phân đoạn từ một hoặc nhiều từ mã trong luồng bit;

giải mã (1510) giá trị thứ hai chỉ báo chiều cao phân đoạn từ một hoặc nhiều từ mã trong luồng bit;

suy ra (1512) các chiều rộng cột phân đoạn dựa vào chiều rộng hình ảnh theo số đơn vị phân đoạn và giá trị thứ nhất, trong đó bước suy ra các chiều rộng cột phân đoạn bao gồm việc thiết đặt các giá trị chiều rộng cột của tất cả các cột phân đoạn trong hình ảnh ngoại trừ một cột bằng giá trị thứ nhất, và việc thiết đặt giá trị chiều rộng cột của một cột phân đoạn còn dư bằng chiều rộng hình ảnh trừ tổng các giá trị chiều rộng của tất cả cột phân đoạn ngoại trừ một cột phân đoạn;

suy ra (1514) các chiều cao hàng phân đoạn dựa vào chiều cao hình ảnh theo số đơn vị phân đoạn và giá trị thứ hai, trong đó, bước suy ra các chiều cao hàng phân đoạn bao gồm việc thiết đặt các giá trị chiều cao hàng của tất cả các hàng phân đoạn trong hình ảnh ngoại trừ một hàng bằng giá trị thứ hai, và thiết đặt giá trị chiều cao hàng của một hàng phân đoạn còn dư bằng chiều cao hình ảnh trừ tổng của các giá trị chiều cao của tất cả các hàng phân đoạn ngoại trừ một hàng phân đoạn;

suy ra (1516) vị trí không gian cho khối hiện tại dựa vào các chiều rộng cột phân đoạn được suy ra và các chiều cao hàng phân đoạn được suy ra; và

giải mã (1518) khối hiện tại sử dụng vị trí không gian được suy ra.

2. Phương pháp theo điểm 1, trong đó một hoặc nhiều phần tử cú pháp được đặt trong tập hợp tham số hình ảnh.

3. Phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 2, trong đó một hoặc nhiều phần tử cú pháp bao gồm cờ một bit xác định xem sự phân đoạn không gian là đồng đều hay không.

4. Phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 3, trong đó đơn vị phân đoạn là đơn vị cây mã hóa hoặc khối cây mã hóa.

5. Phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 4, còn bao gồm bước tính toán kích thước của hình ảnh theo số đơn vị phân đoạn.

6. Phương pháp theo điểm 5, trong đó bước tính toán kích thước của hình ảnh theo số đơn vị phân đoạn bao gồm: việc tính toán kích thước của phân đoạn hình ảnh cho chiều cao và việc tính toán kích thước của phân đoạn hình ảnh cho chiều rộng.

7. Phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 6, trong đó giá trị thứ nhất biểu diễn chiều rộng ô xếp kiểu ngói theo các đơn vị phân đoạn và giá trị thứ hai biểu diễn chiều cao ô xếp kiểu ngói theo các đơn vị phân đoạn.

8. Phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 7, trong đó

bước giải mã giá trị thứ nhất từ một hoặc nhiều từ mã trong luồng bit bao gồm việc giải mã giá trị của từ mã cụ thể trong luồng bit và thêm 1 vào giá trị được giải mã, và/hoặc

bước giải mã giá trị thứ hai từ một hoặc nhiều từ mã trong luồng bit bao gồm việc giải mã giá trị của từ mã cụ thể thứ hai trong luồng bit và thêm 1 vào giá trị được giải mã.

9. Phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 8, trong đó

bước suy ra các chiều rộng cột phân đoạn bao gồm: việc so sánh giá trị bằng (A -

TileWidth) với TileWidth, trong đó TileWidth là bằng giá trị thứ nhất, và A là bằng giá trị chiều rộng còn dư, và/hoặc

suy ra các chiều cao hàng phân đoạn bao gồm: việc so sánh giá trị bằng (A - TileHeight) với TileHeight, trong đó TileHeight là bằng giá trị thứ hai, và A là bằng giá trị chiều cao còn dư.

10. Phương pháp theo điểm 9, trong đó

giá trị chiều rộng còn dư là bằng chiều rộng hình ảnh nêu trên, và/hoặc

giá trị chiều cao còn dư là bằng chiều cao hình ảnh nêu trên.

11. Phương pháp theo điểm 9 hoặc 10, trong đó

nếu xác định được rằng giá trị của (A - TileWidth) là lớn hơn TileWidth, thì sau đó thực hiện các bước gồm: thiết đặt biến số chiều rộng cột thứ nhất bằng TileWidth; và so sánh giá trị bằng (A - (2 x TileWidth)) với TileWidth, và/hoặc

nếu xác định được rằng giá trị của (A - TileHeight) là lớn hơn TileHeight, thì sau đó thực các bước gồm: thiết đặt biến số chiều cao hàng thứ nhất bằng TileHeight; và so sánh giá trị bằng (A - (2 x TileHeight)) với TileHeight.

12. Phương pháp theo điểm 9, 10, hoặc 11, trong đó

việc so sánh giá trị (A - TileWidth) với TileWidth bao gồm: thiết đặt A bằng với (A - TileWidth); và so sánh A với TileWidth, và/hoặc

việc so sánh giá trị (A - TileHeight) với TileHeight bao gồm: thiết đặt A bằng (A - TileHeight); và so sánh A với TileHeight.

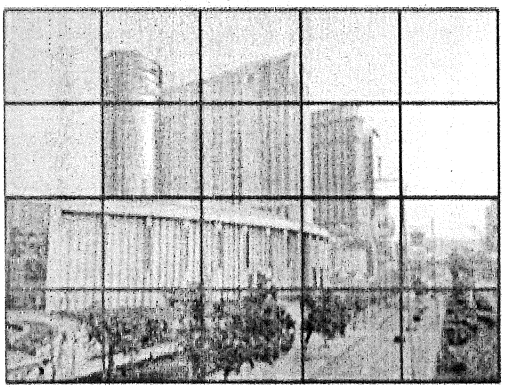
13. Phương tiện lưu trữ có thể đọc được bởi máy tính (1142) chứa chương trình máy tính (1143) bao gồm các lệnh (1144) mà khi được thực thi bởi hệ mạch xử lý (1102) làm cho hệ mạch xử lý (1102) thực hiện phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 12.

14. Thiết bị giải mã (1100) để giải mã hình ảnh (10), thiết bị giải mã (1100) được tạo cấu hình để thực hiện phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 12.

15. Thiết bị giải mã (1100) để giải mã hình ảnh (10), thiết bị giải mã (1100) bao gồm:

phương tiện lưu trữ có thể đọc được bởi máy tính (1142); và

hệ mạch xử lý (1102) được ghép nối với phương tiện lưu trữ có thể đọc được bởi máy tính, trong đó hệ mạch xử lý được tạo cấu hình để làm cho thiết bị giải mã (1100) để thực hiện phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 12.



0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19

FIG. 1

ctuRsAddr =

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19

FIG. 2A

ctuRsAddr =

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23

FIG. 2B

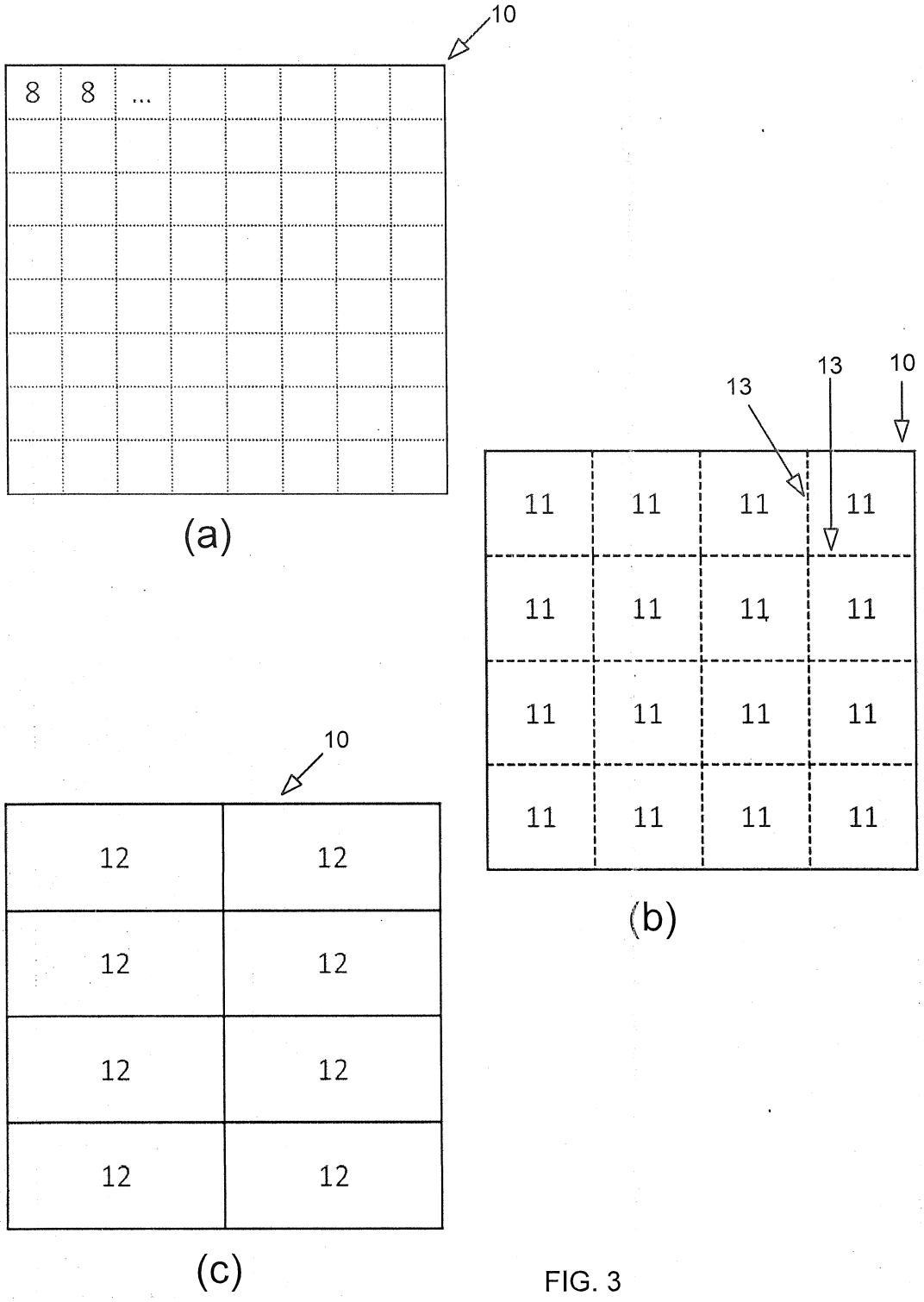


FIG. 3

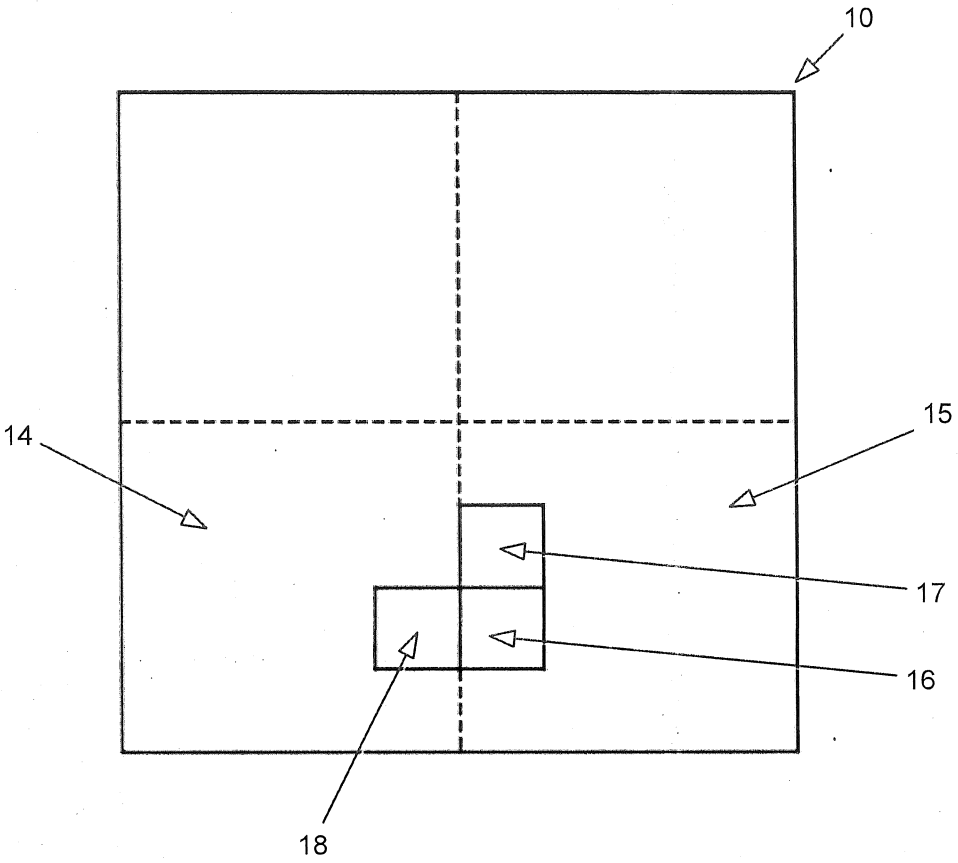


FIG. 4

500

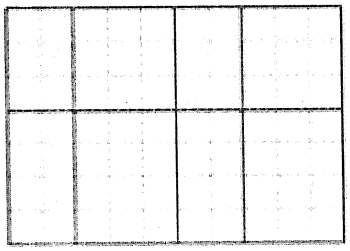
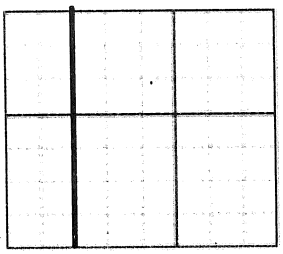
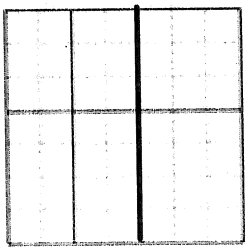
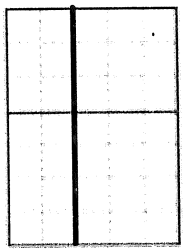
Các biên ô xếp kiểu ngói Chỉ các biên CTU Các biên ô xếp kiểu ngói được di chuyển so với hình ảnh gốc Các phần được loại bỏ	Sự đặt cách ô xếp kiểu ngói đồng đều trong HEVC
Các ô xếp kiểu ngói trong hình ảnh gốc	
Việc đặt cách ô xếp kiểu ngói đồng đều sau khi loại bỏ các cột cực trái của các ô xếp kiểu ngói	
Việc đặt cách ô xếp kiểu ngói đồng đều sau khi loại bỏ các cột cực phải của các ô xếp kiểu ngói	
Việc đặt cách ô xếp kiểu ngói đồng đều sau khi loại bỏ các cột cực trái và cực phải của các ô xếp kiểu ngói	

FIG. 5

600

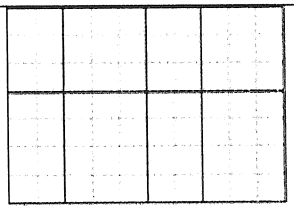
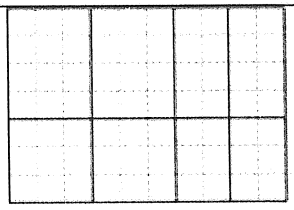
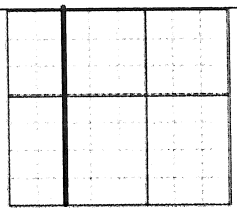
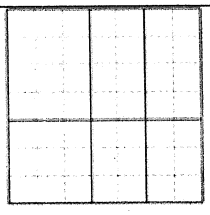
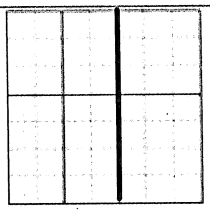
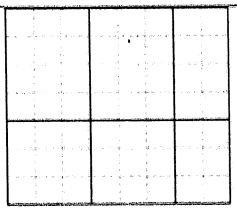
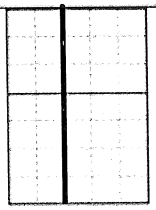
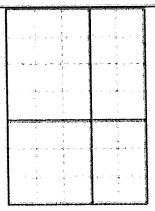
Các biên ô xếp kiểu ngói Chỉ các biên CTU Các biên ô xếp kiểu ngói được di chuyển so với hình ảnh gốc Các phần được loại bỏ	Việc đặt cách ô xếp kiểu ngói đồng đều HEVC	Việc đặt cách ô xếp kiểu ngói đồng đều được đề xuất ở Phương án 1
Các ô xếp kiểu ngói trong hình ảnh gốc		
Việc đặt cách ô xếp kiểu ngói đồng đều sau khi loại bỏ các cột cực trái của các ô xếp kiểu ngói		
Việc đặt cách ô xếp kiểu ngói đồng đều sau khi loại bỏ các cột cực phải của các ô xếp kiểu ngói		
Việc đặt cách ô xếp kiểu ngói đồng đều sau khi loại bỏ các cột cực trái và cực phải của các ô xếp kiểu ngói		

FIG. 6

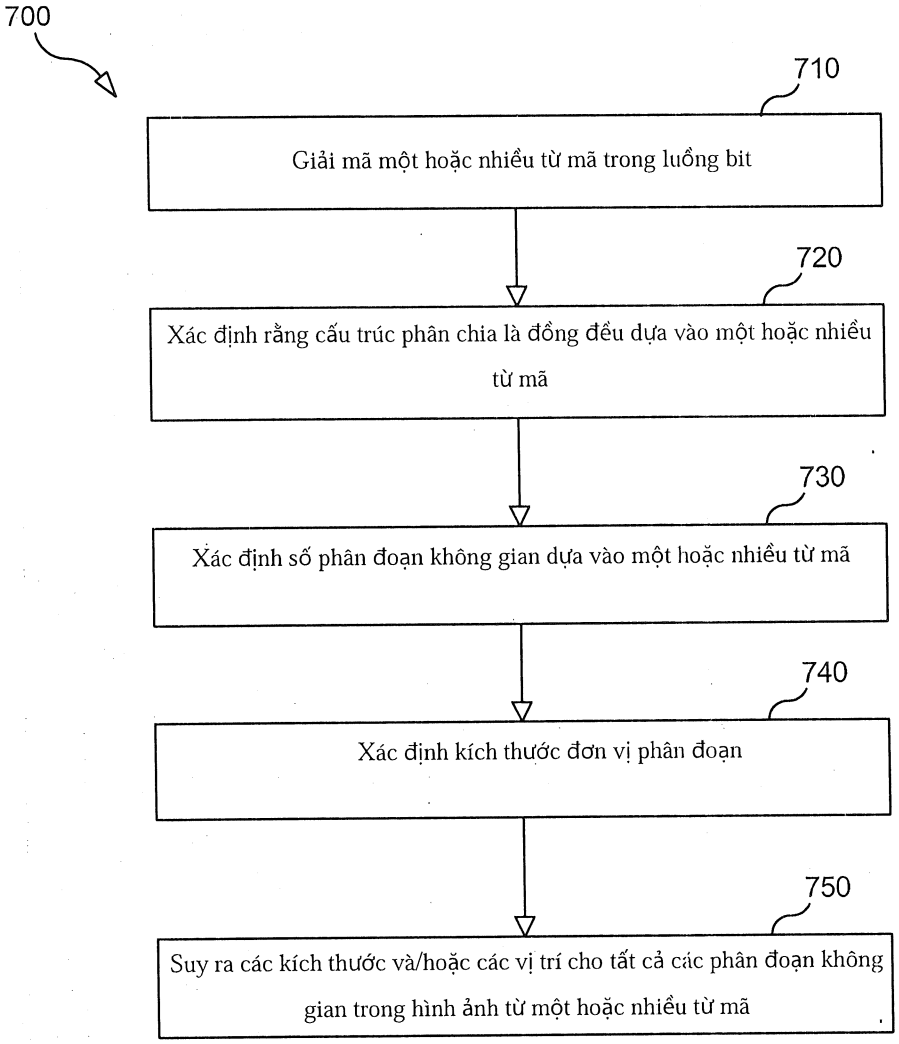

700
710
Giải mã một hoặc nhiều từ mã trong luồng bit
720
Xác định rằng cấu trúc phân chia là đồng đều dựa vào một hoặc nhiều từ mã
730
Xác định số phân đoạn không gian dựa vào một hoặc nhiều từ mã
740
Xác định kích thước đơn vị phân đoạn
750
Suy ra các kích thước và/hoặc các vị trí cho tất cả các phân đoạn không gian trong hình ảnh từ một hoặc nhiều từ mã

FIG. 7

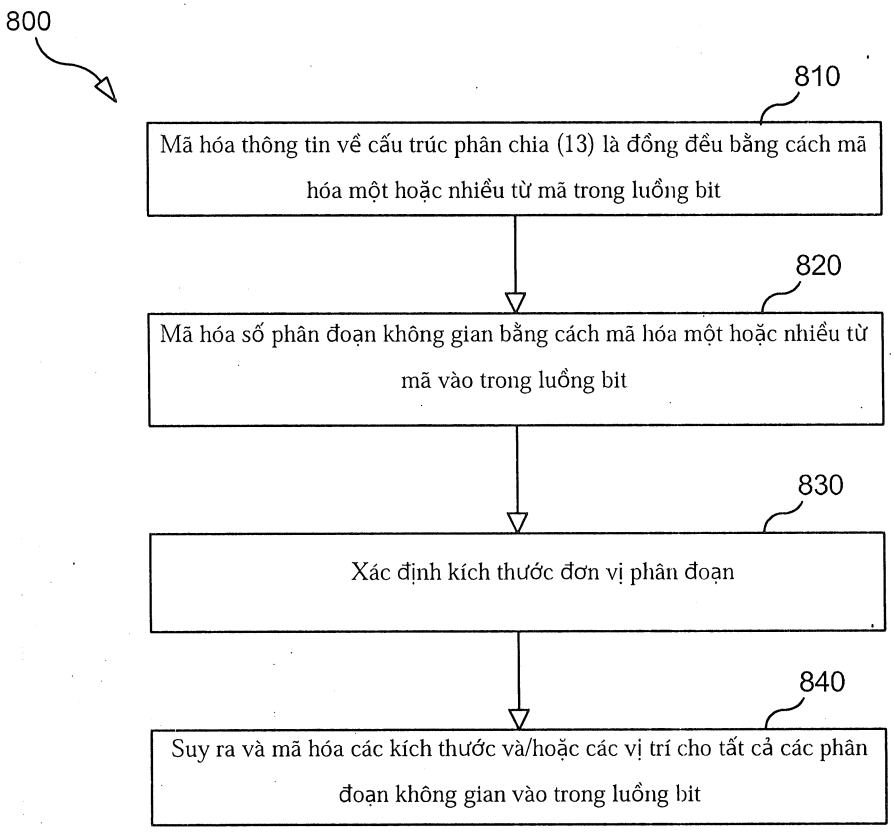

800
810
Mã hóa thông tin về cấu trúc phân chia (13) là đồng đều bằng cách mã hóa một hoặc nhiều từ mã trong luồng bit
820
Mã hóa số phân đoạn không gian bằng cách mã hóa một hoặc nhiều từ mã vào trong luồng bit
830
Xác định kích thước đơn vị phân đoạn
840
Suy ra và mã hóa các kích thước và/hoặc các vị trí cho tất cả các phân đoạn không gian vào trong luồng bit

FIG. 8

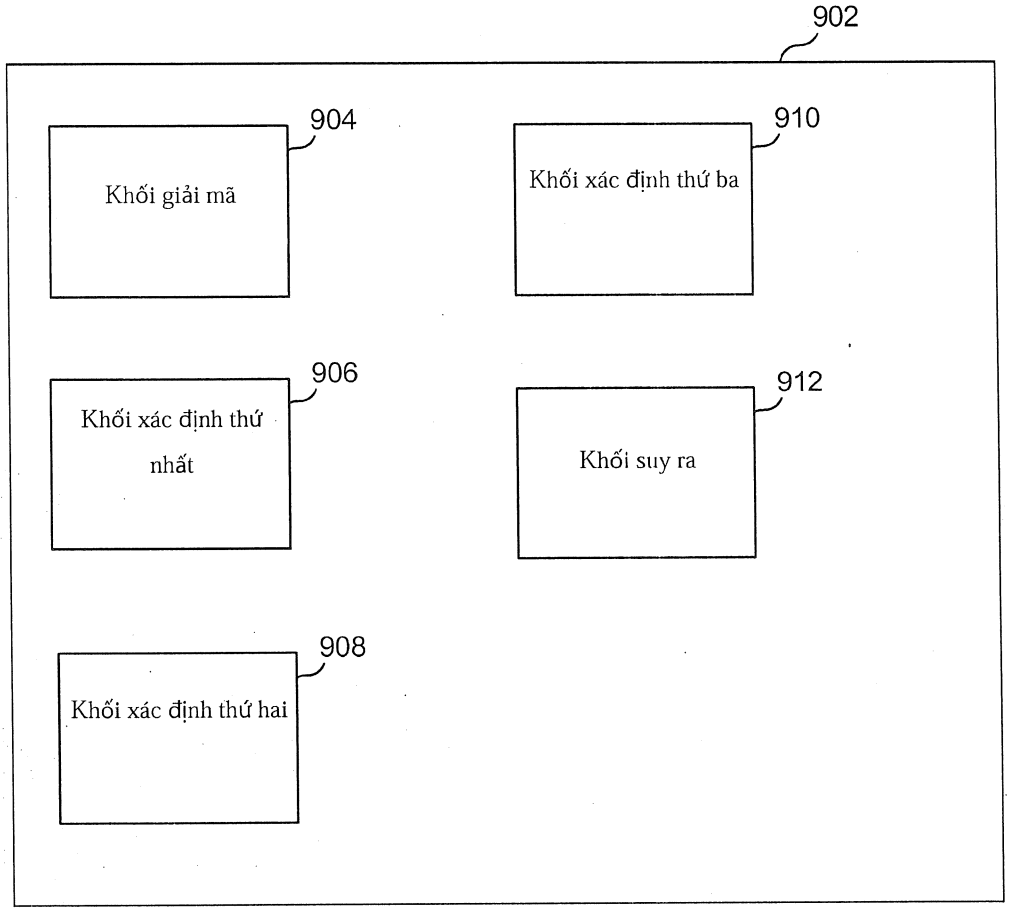

902
904
Khối giải mã
910
Khối xác định thứ ba
906
Khối xác định thứ nhất
912
Khối suy ra
908
Khối xác định thứ hai

FIG. 9

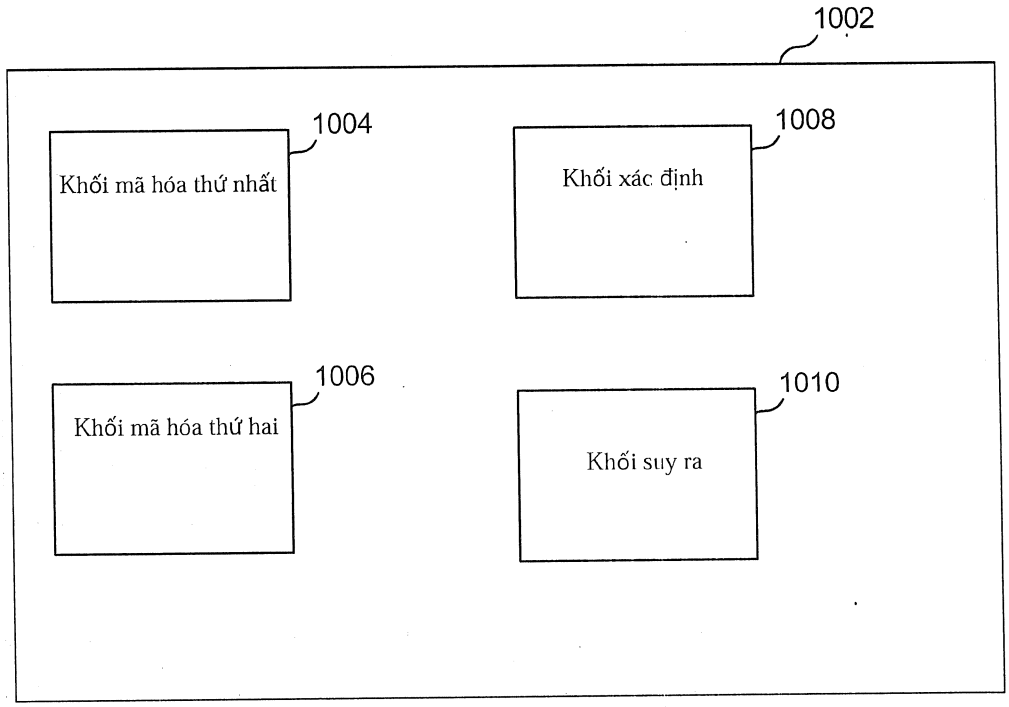

1002
1004
Khối mã hóa thứ nhất
1008
Khối xác định
1006
Khối mã hóa thứ hai
1010
Khối suy ra

FIG. 10

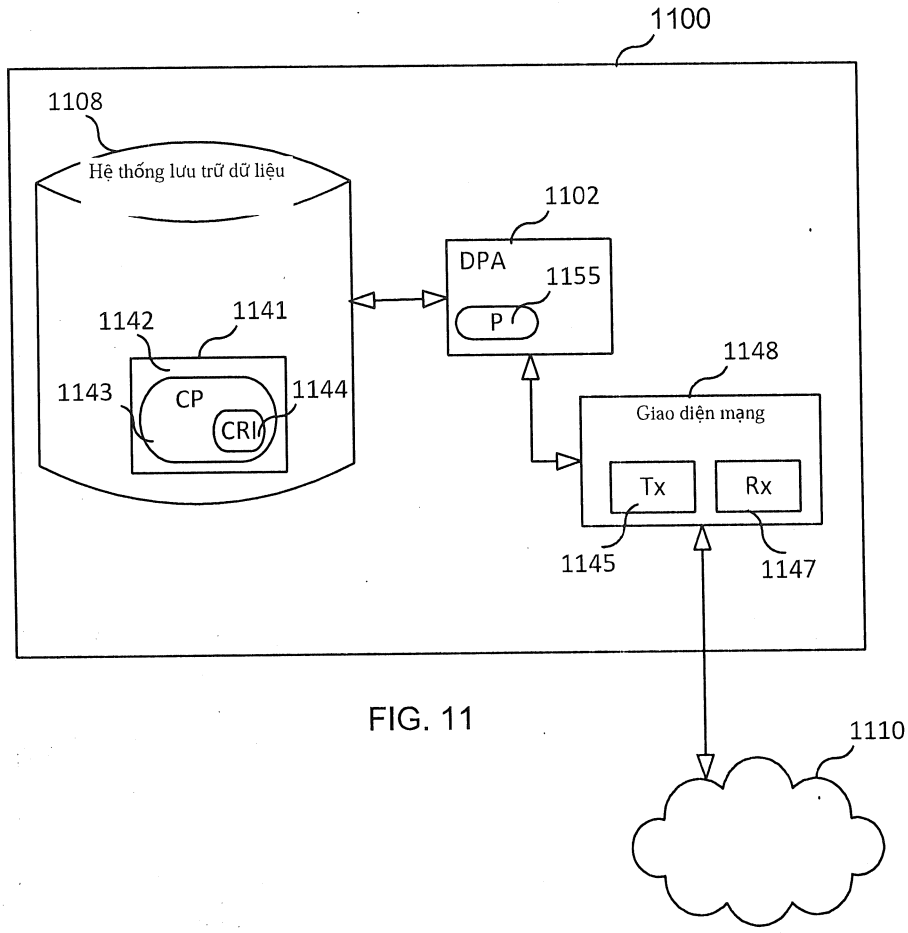

1100
1108
Hệ thống lưu trữ dữ liệu
1102
DPA
1155
P
1142
1141
1143
CP
1144
CRI
1148
Giao diện mạng
Tx
Rx
1145
1147
1110

FIG. 11

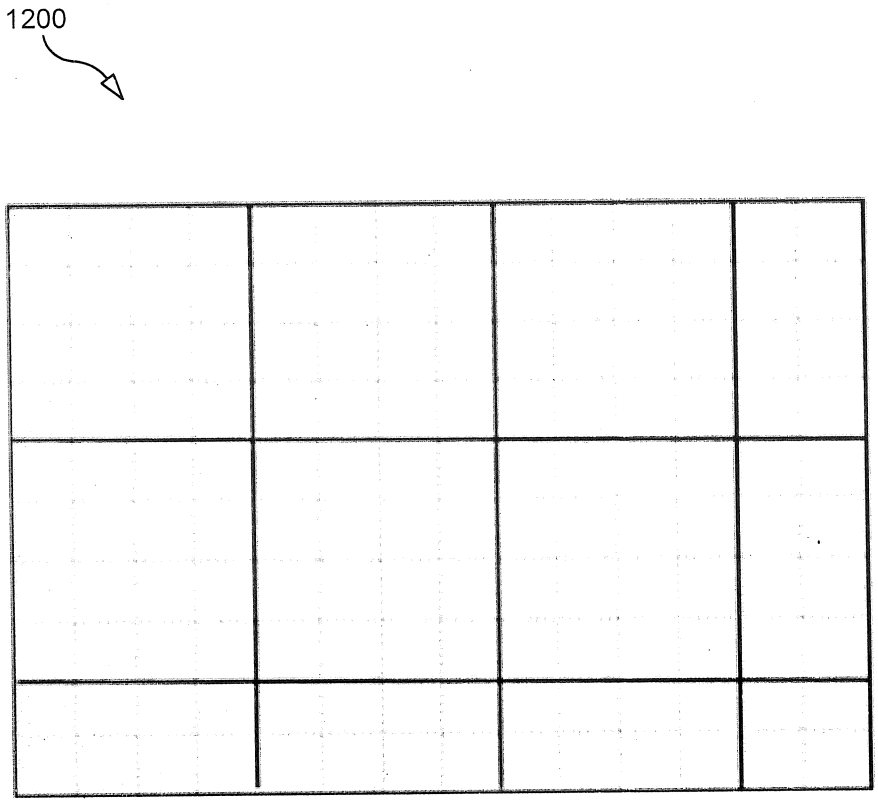


FIG. 12

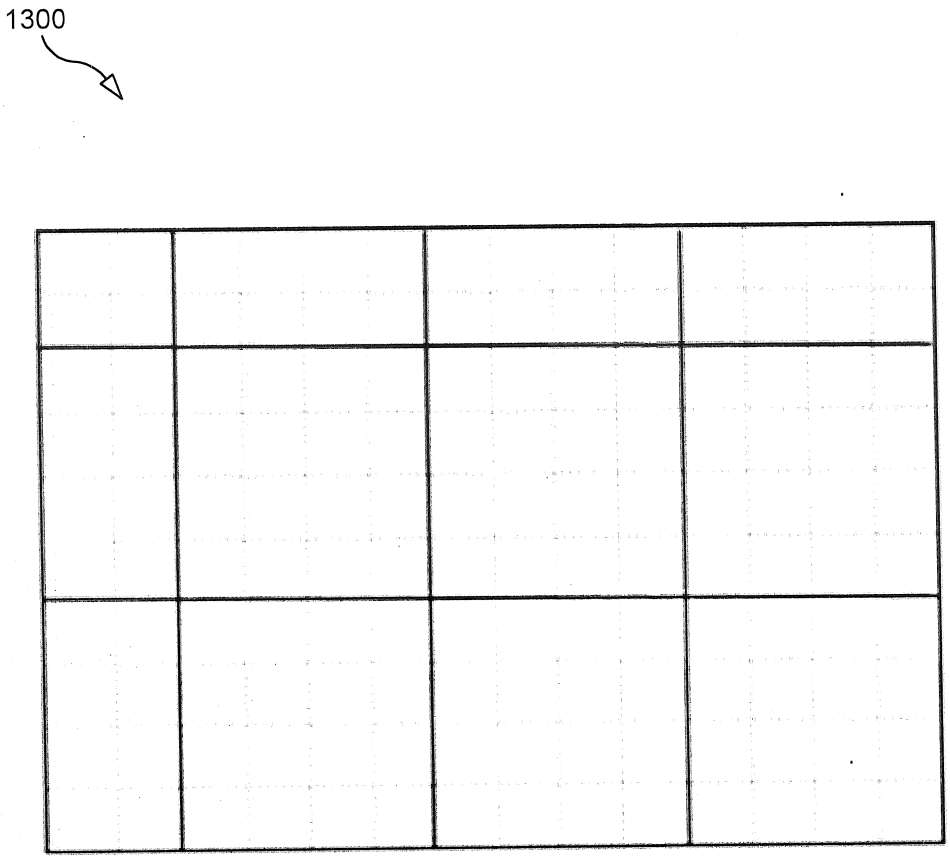


FIG. 13

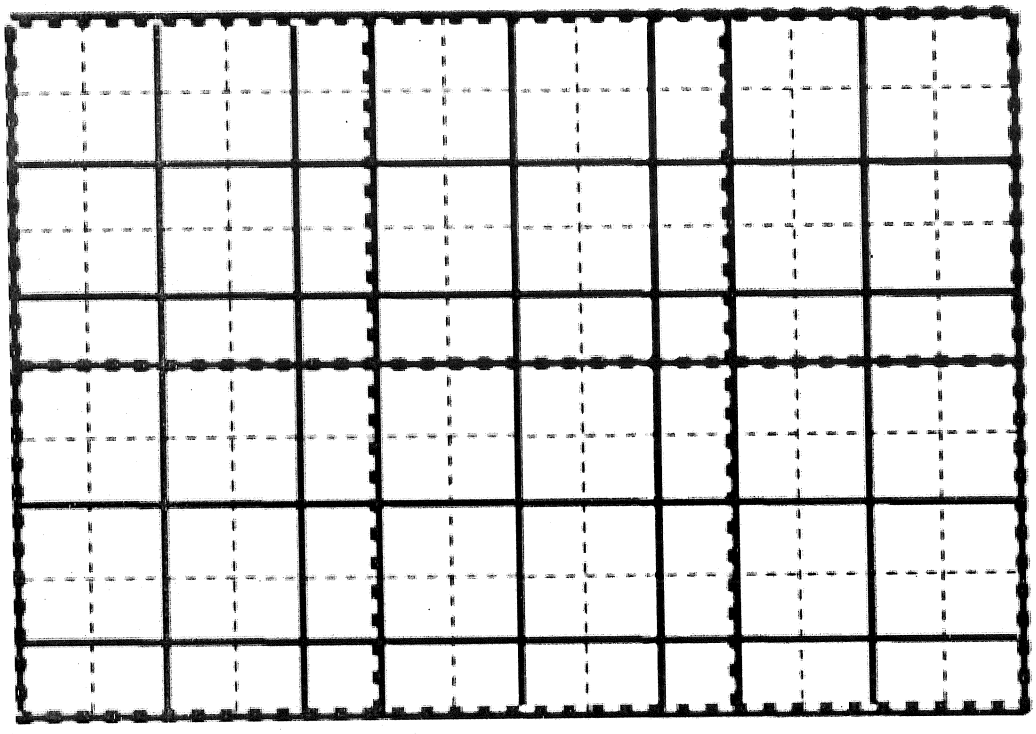


FIG. 14

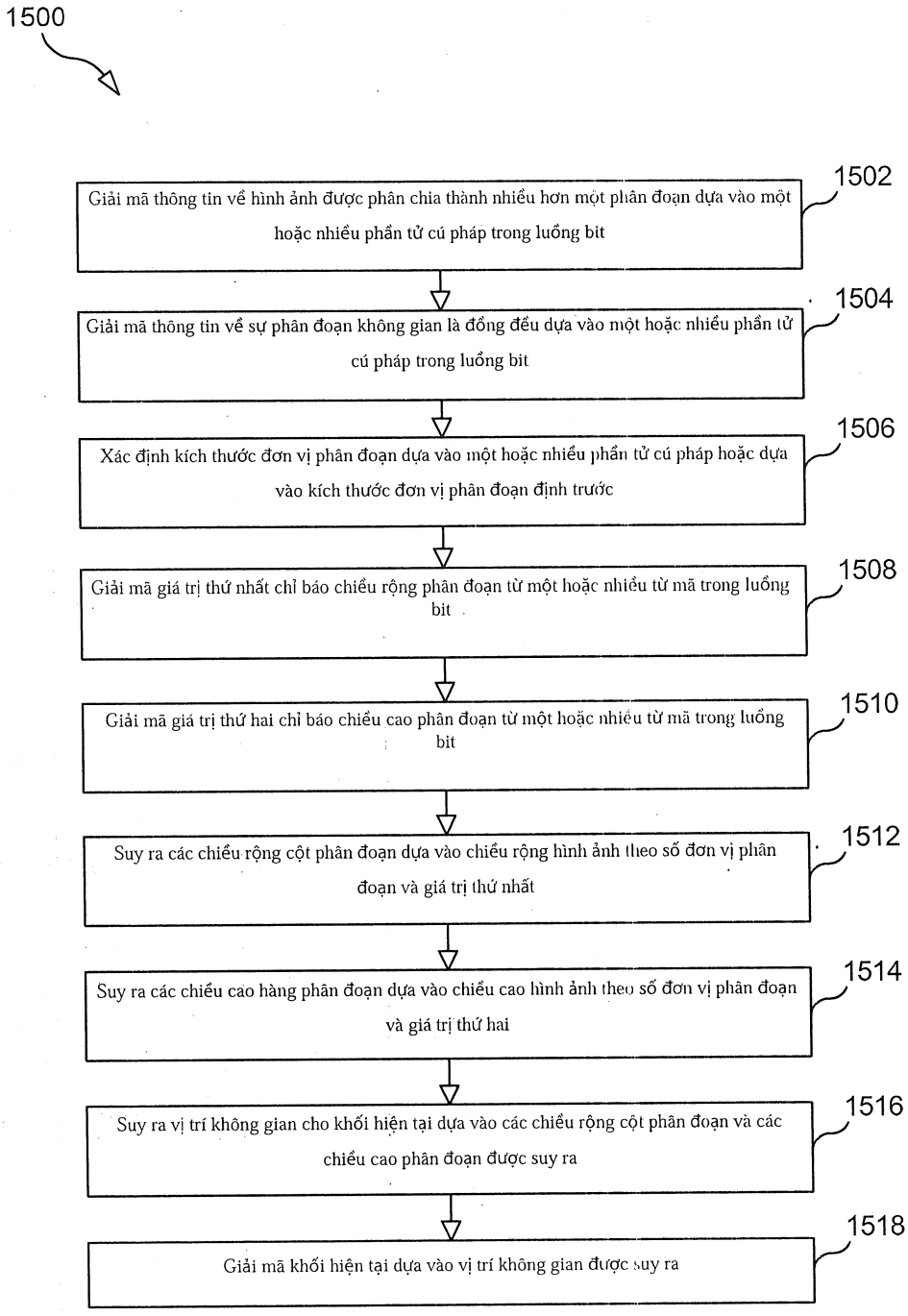

1500
Giải mã thông tin về hình ảnh được phân chia thành nhiều hơn một phân đoạn dựa vào một hoặc nhiều phần tử cú pháp trong luồng bit
1502
Giải mã thông tin về sự phân đoạn không gian là đồng đều dựa vào một hoặc nhiều phần tử cú pháp trong luồng bit
1504
Xác định kích thước đơn vị phân đoạn dựa vào một hoặc nhiều phần tử cú pháp hoặc dựa vào kích thước đơn vị phân đoạn định trước
1506
Giải mã giá trị thứ nhất chỉ báo chiều rộng phân đoạn từ một hoặc nhiều từ mã trong luồng bit
1508
Giải mã giá trị thứ hai chỉ báo chiều cao phân đoạn từ một hoặc nhiều từ mã trong luồng bit
1510
Suy ra các chiều rộng cột phân đoạn dựa vào chiều rộng hình ảnh theo số đơn vị phân đoạn và giá trị thứ nhất
1512
Suy ra các chiều cao hàng phân đoạn dựa vào chiều cao hình ảnh theo số đơn vị phân đoạn và giá trị thứ hai
1514
Suy ra vị trí không gian cho khối hiện tại dựa vào các chiều rộng cột phân đoạn và các chiều cao phân đoạn được suy ra
1516
Giải mã khối hiện tại dựa vào vị trí không gian được suy ra
1518

FIG. 15

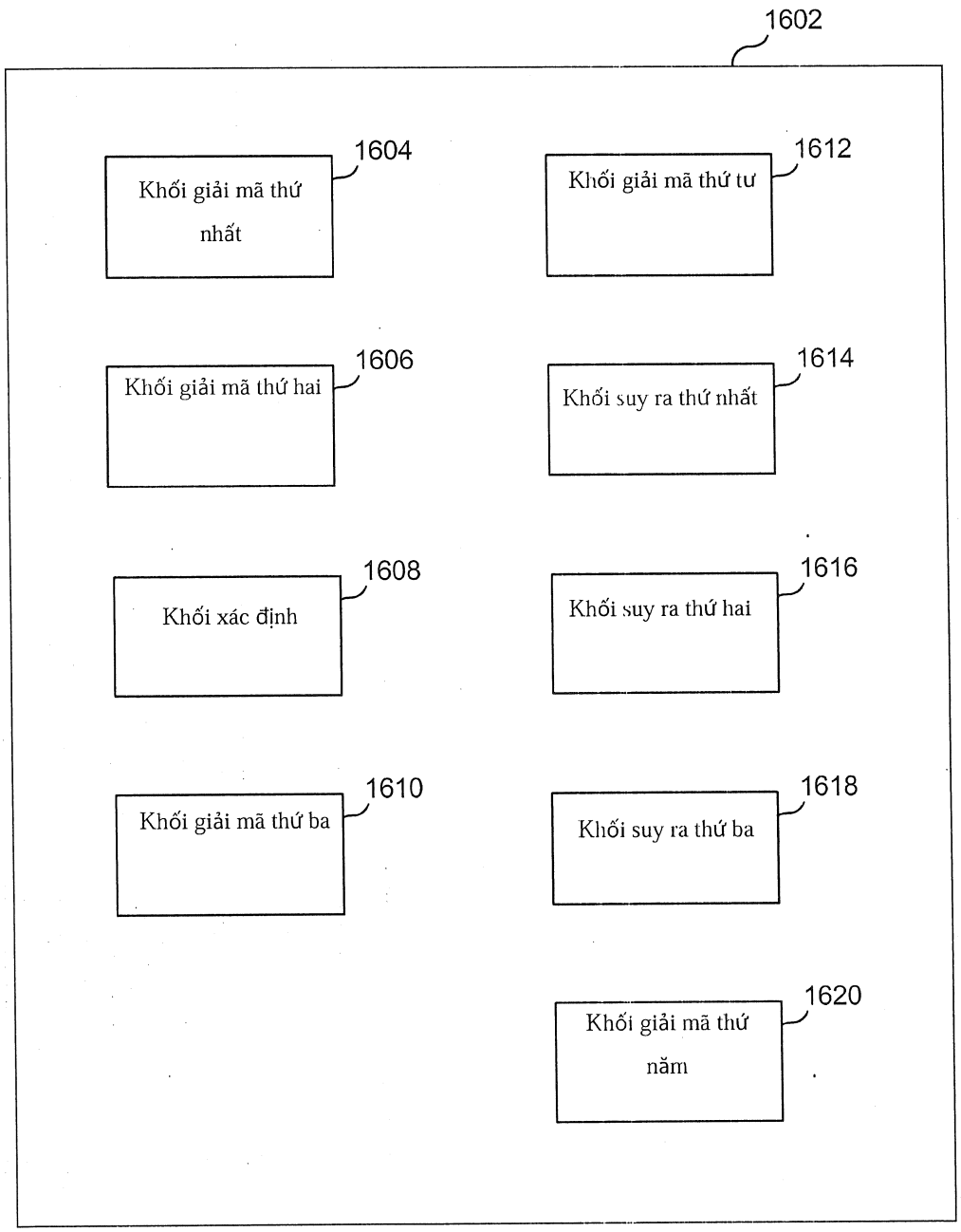

1602
1604
Khối giải mã thứ nhất
1606
Khối giải mã thứ hai
1608
Khối xác định
1610
Khối giải mã thứ ba
1612
Khối giải mã thứ tư
1614
Khối suy ra thứ nhất
1616
Khối suy ra thứ hai
1618
Khối suy ra thứ ba
1620
Khối giải mã thứ năm

FIG. 16