(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) **1-0053884**

(51)[2020.01] **H04N 19/70**; H04N 19/36; H04N 19/13; H04N 19/30

(13) **B**

(21) 1-2021-01731
(22) 13/09/2019
(86) PCT/US2019/051147 13/09/2019
(87) WO2020/056352 19/03/2020
(30) 62/731,696 14/09/2018 US
(45) 25/11/2025 452
(43) 25/06/2021 399A
(73) HUAWEI TECHNOLOGIES CO., LTD. (CN)
Huawei Administration Building, Bantian, Longgang District, Shenzhen, Guangdong 518129, P. R. China
(72) WANG, Ye-Kui (US); HENDRY, Fnu (ID); CHEN, Jianle (CN).
(74) Công ty Luật TNHH T&G (TGVN)

(54) PHƯƠNG PHÁP GIẢI MÃ, PHƯƠNG PHÁP MÃ HOÁ, BỘ LẬP MÃ, VÀ PHƯƠNG TIỆN LƯU TRỮ PHI NHẤT THỜI ĐỌC ĐƯỢC BẰNG MÁY TÍNH

(21) 1-2021-01731

(57) Sáng chế đề xuất cơ chế lập mã viđeo. Cơ chế này bao gồm việc nhận, tại bộ giải mã, luồng bit bao gồm đơn vị VCL (Video Coding Layer - lớp lập mã viđeo) NAL (Network Abstraction Layer - lớp trừu tượng mạng) chứa lát cắt của dữ liệu ảnh được chia thành các ô. Số lượng ô trong đơn vị VCL NAL này được xác định. Số lượng độ dịch điểm vào dành cho các ô này cũng được xác định là số lượng ô trong đơn vị VCL NAL này trừ đi một. Mỗi độ dịch điểm vào chỉ thị vị trí bắt đầu của ô tương ứng trong đơn vị VCL NAL. Số lượng độ dịch điểm vào không được báo hiệu tường minh trong luồng bit này. Các độ dịch điểm vào dành cho các ô là được thu thập dựa trên số lượng độ dịch điểm vào này. Các ô được giải mã tại các độ dịch điểm vào này để tạo ra hình ảnh được tái tạo. Phương pháp giải mã, phương pháp mã hoá, phương tiện lưu trữ phi nhất thời, thiết bị lập mã, bộ lập mã, phương tiện lưu trữ đọc được bằng máy tính, thiết bị lập mã viđeo, thiết bị giải mã, và thiết bị mã hoá cũng được bộc lộ.

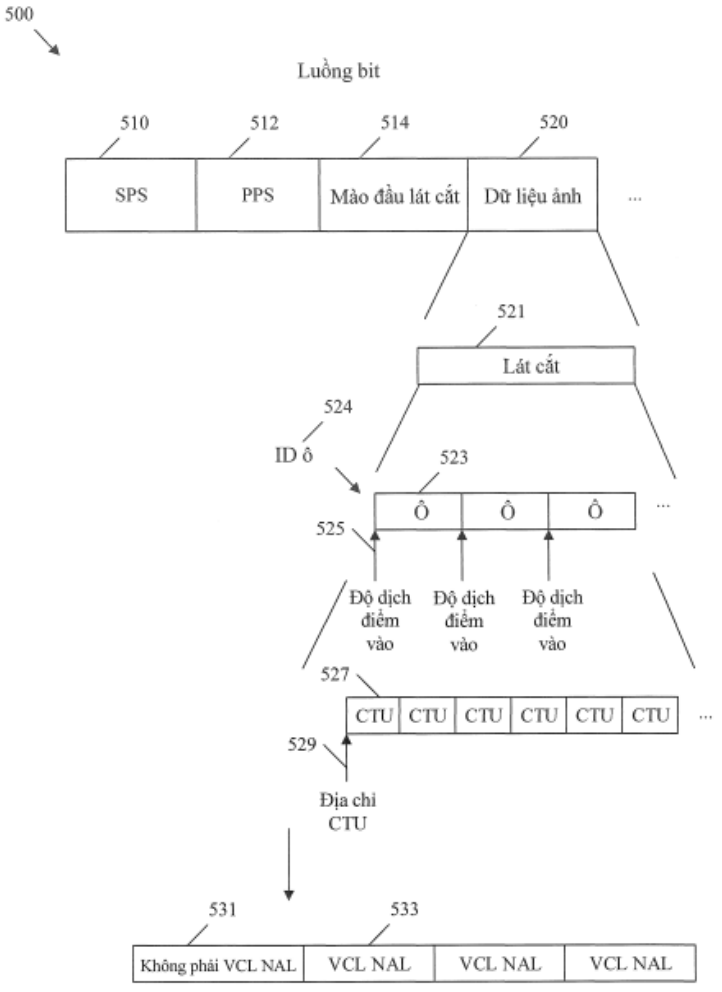

Fig.5

Lĩnh vực kỹ thuật được đề cập

Sáng chế đề cập chung đến việc lập mã viđeo, cụ thể là đề cập đến việc phân vùng các hình ảnh thành các lát cắt, các ô, và các đơn vị cây lập mã (Coding Tree Unit - CTU) để hỗ trợ việc tăng nén trong việc lập mã viđeo.

Tình trạng kỹ thuật của sáng chế

Lượng dữ liệu viđeo cần thiết để thể hiện ngay cả một viđeo tương đối ngắn cũng có thể là lớn, điều này có thể dẫn đến những khó khăn khi dữ liệu này cần được tạo luồng hoặc được truyền thông qua mạng truyền thông có dung lượng băng thông hạn chế. Do đó, dữ liệu viđeo thường được nén trước khi được truyền thông qua các mạng viễn thông thời hiện đại. Kích thước của viđeo cũng có thể là một vấn đề khi viđeo được lưu giữ trên thiết bị lưu trữ vì các tài nguyên bộ nhớ có thể bị hạn chế. Các thiết bị nén viđeo thường sử dụng phần mềm và/hoặc phần cứng tại nguồn để lập mã dữ liệu viđeo trước khi truyền hoặc lưu trữ, nhờ đó làm giảm lượng dữ liệu cần thiết để biểu thị các hình ảnh viđeo số. Sau đó, dữ liệu được nén này được nhận tại đích đến bởi thiết bị giải nén viđeo mà giải mã dữ liệu viđeo đó. Với các tài nguyên mạng bị hạn chế và các nhu cầu ngày càng tăng về chất lượng viđeo cao hơn, thì mong muốn là có các kỹ thuật nén và giải nén được cải thiện để cải thiện tỷ số nén mà chỉ phải hy sinh ít hoặc không phải hy sinh chất lượng hình ảnh.

Bản chất kỹ thuật của sáng chế

Theo một phương án, sáng chế bao gồm phương pháp được thực hiện ở bộ giải mã. Phương pháp này bao gồm bước nhận, bởi bộ thu của bộ giải mã, luồng bit bao gồm đơn vị VCL (Video Coding Layer - lớp lập mã viđeo) NAL (Network Abstraction Layer - lớp trừu tượng mạng) chứa lát cắt của dữ liệu ảnh được chia thành nhiều ô. Phương pháp này còn bao gồm bước xác định, bởi bộ xử lý của bộ giải mã, một số lượng ô trong đơn vị VCL NAL. Phương pháp này còn bao gồm bước xác định, bởi bộ xử lý, số lượng độ dịch điểm vào dành cho các ô dưới dạng số lượng ô trong đơn vị

VCL NAL trừ đi một, trong đó mỗi độ dịch điểm vào chỉ thị vị trí bắt đầu của ô tương ứng trong đơn vị VCL NAL, và trong đó số lượng độ dịch điểm vào này không được báo hiệu tường minh trong luồng bit này. Phương pháp này còn bao gồm bước thu thập, bởi bộ xử lý, các độ dịch điểm vào dành cho các ô dựa trên số lượng độ dịch điểm vào. Phương pháp này còn bao gồm bước giải mã, bởi bộ xử lý, các ô tại các độ dịch điểm vào để tạo ra hình ảnh được tái tạo. Ở một số hệ thống, thì số lượng độ dịch điểm vào dành cho các ô trong lát cắt là được báo hiệu tường minh. Hệ thống được bộc lộ này suy ra số lượng độ dịch điểm vào là số lượng ô trừ đi một. Bằng cách sử dụng việc suy ra này, thì số lượng độ dịch điểm vào có thể được lược bỏ khỏi luồng bit. Theo đó, luồng bit này được thu gọn hơn nữa. Như vậy, các tài nguyên mạng mà được dùng để truyền luồng bit này là được giảm bớt. Ngoài ra, việc sử dụng tài nguyên bộ nhớ ở cả bộ mã hoá và bộ giải mã được giảm bớt.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng các độ dịch điểm vào dành cho các ô là được thu thập từ mào đầu lát cắt được liên kết với lát cắt đó.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng luồng bit nêu trên bao gồm nhiều đơn vị VCL NAL, và trong đó mỗi đơn vị VCL NAL chứa một lát cắt đơn của dữ liệu ảnh được chia thành một số lượng nguyên các ô. Ở một số hệ thống, thì các lát cắt và các ô là các sơ đồ chia riêng biệt. Bằng cách yêu cầu chia tiếp các lát cắt thành các ô, thì thông tin có thể được suy ra. Ví dụ, một số ID (IDentifier - bộ nhận dạng) ô trong lát cắt có thể được suy ra bởi ô đầu tiên và cuối cùng trong lát cắt đó. Ngoài ra, các đường biên lát cắt có thể được báo hiệu dựa trên ID ô chứ không phải vị trí tương đối của lát cắt trong khung. Về phần mình, thì việc này hỗ trợ sơ đồ đánh địa chỉ mà trong đó các mào đầu lát cắt không cần phải được ghi lại khi báo hiệu khung con. Theo đó, luồng bit có thể được thu gọn hơn nữa theo một số ví dụ, điều đó tiết kiệm các tài nguyên bộ nhớ và các tài nguyên truyền thông mạng. Ngoài ra, các tài nguyên xử lý có thể được tiết kiệm tại bộ mã hoá và/hoặc bộ giải mã.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng dữ liệu ảnh là được lập mã dưới dạng các đơn vị cây lập mã (Coding Tree Unit - CTU) được chứa trong mỗi trong số các ô, và trong đó các địa chỉ của các CTU trong đơn vị VCL NAL là được ấn định dựa trên các bộ nhận

dạng (IDentifier - ID) ô tương ứng với các ô đó. Ví dụ, việc đánh địa chỉ các CTU dựa trên ID ô cho phép các địa chỉ được suy ra dựa trên ID ô, điều này có thể thu gọn luồng bit. Ngoài ra, việc đánh địa chỉ dựa trên ô đối với các CTU hỗ trợ việc lập mã CTU mà không tham chiếu đến các tiến trình lập mã xảy ra bên ngoài lát cắt hiện tại. Như vậy, việc đánh địa chỉ các CTU dựa trên ID ô hỗ trợ việc xử lý song song, và do đó, hỗ trợ việc tăng tốc độ giải mã tại bộ giải mã.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng việc giải mã các ô là bao gồm việc giải mã các CTU dựa trên các địa chỉ của các CTU trong đơn vị VCL NAL mà được báo hiệu tường minh trong mào đầu lát cắt.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng việc giải mã các ô là bao gồm việc: xác định, bởi bộ xử lý, các địa chỉ của các CTU trong đơn vị VCL NAL dựa trên ID ô của ô đầu tiên được chứa trong đơn vị VCL NAL; và giải mã, bởi bộ xử lý, các CTU dựa trên các địa chỉ của các CTU này.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng các CTU không chứa cờ chỉ thị CTU cuối cùng trong đơn vị VCL NAL, và trong đó đơn vị VCL NAL này chứa bit đệm ngay sau CTU cuối cùng trong mỗi ô. Ở một số hệ thống, thì cờ được sử dụng trong mỗi CTU để chỉ thị xem CTU hiện tại có phải là CTU cuối cùng trong đơn vị VCL NAL hay không. Theo sáng chế, thì các lát cắt và các CTU là được đánh địa chỉ dựa trên ID ô, và đơn vị VCL NAL chứa lát cắt đơn. Theo đó, bộ giải mã có thể xác định CTU nào là CTU cuối cùng trong lát cắt và theo đó là CTU cuối cùng trong đơn vị VCL NAL mà không cần cờ đó. Vì chuỗi viđeo chứa nhiều khung và mỗi khung chứa nhiều CTU, nên việc lược bỏ bit dữ liệu khỏi mỗi thao tác mã hoá CTU có thể thu gọn luồng bit một cách đáng kể. Như vậy, các tài nguyên mạng mà được dùng để truyền luồng bit này là được giảm bớt. Ngoài ra, việc sử dụng tài nguyên bộ nhớ ở cả bộ mã hoá và bộ giải mã được giảm bớt.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng còn bao gồm việc chuyển tiếp, bởi bộ xử lý, hình ảnh được tái tạo đến thiết bị hiển thị như một phần của chuỗi viđeo được tái tạo.

Theo một phương án, sáng chế bao gồm phương pháp được thực hiện ở bộ mã hoá. Phương pháp này bao gồm bước phân vùng, bởi bộ xử lý của bộ mã hoá này, hình ảnh thành ít nhất một lát cắt và phân vùng ít nhất một lát cắt này thành các ô. Phương pháp này còn bao gồm bước mã hoá, bởi bộ xử lý này, các ô này vào luồng bit trong ít nhất một đơn vị VCL NAL. Phương pháp này còn bao gồm bước mã hoá, bởi bộ xử lý, một số lượng ô trong đơn vị VCL NAL này trong luồng bit này. Phương pháp này còn bao gồm bước mã hoá, bởi bộ xử lý, các độ dịch điểm vào dành cho các ô trong luồng bit này, trong đó mỗi trong số các độ dịch điểm vào này chỉ thị vị trí bắt đầu của ô tương ứng trong đơn vị VCL NAL, và trong đó số lượng độ dịch điểm vào là không được báo hiệu tường minh trong luồng bit này. Phương pháp này còn bao gồm bước truyền, bởi bộ phát của bộ mã hoá, luồng bit này mà không có số lượng độ dịch điểm vào để hỗ trợ việc giải mã các ô theo sự suy luận rằng số lượng độ dịch điểm vào dành cho các ô là số lượng ô trong đơn vị VCL NAL trừ đi một. Ở một số hệ thống, thì số lượng độ dịch điểm vào dành cho các ô trong lát cắt là được báo hiệu tường minh. Hệ thống được bộc lộ này suy ra số lượng độ dịch điểm vào là số lượng ô trừ đi một. Bằng cách sử dụng việc suy ra này, thì số lượng độ dịch điểm vào có thể được lược bỏ khỏi luồng bit. Theo đó, luồng bit này được thu gọn hơn nữa. Như vậy, các tài nguyên mạng mà được dùng để truyền luồng bit này là được giảm bớt. Ngoài ra, việc sử dụng tài nguyên bộ nhớ ở cả bộ mã hoá và bộ giải mã được giảm bớt.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng các độ dịch điểm vào dành cho các ô là được mã hoá trong mào đầu lát cắt được liên kết với lát cắt đó.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng luồng bit bao gồm nhiều đơn vị VCL NAL, và trong đó mỗi đơn vị VCL NAL chứa một lát cắt đơn của hình ảnh được chia thành một số lượng nguyên các ô. Ở một số hệ thống, thì các lát cắt và các ô là các sơ đồ chia riêng biệt. Bằng cách yêu cầu chia tiếp các lát cắt thành các ô, thì thông tin có thể được suy ra. Ví dụ, một số ID (IDentifier - bộ nhận dạng) ô trong lát cắt có thể được suy ra bởi ô đầu tiên và cuối cùng trong lát cắt đó. Ngoài ra, các đường biên lát cắt có thể được báo hiệu dựa trên ID ô chứ không phải vị trí tương đối của lát cắt trong khung. Về phần mình, thì việc này hỗ trợ sơ đồ đánh địa chỉ mà trong đó các mào đầu lát cắt không cần phải được ghi lại khi báo hiệu khung con. Theo đó, luồng bit có thể được thu gọn hơn

nữa theo một số ví dụ, điều đó tiết kiệm các tài nguyên bộ nhớ và các tài nguyên truyền thông mạng. Ngoài ra, các tài nguyên xử lý có thể được tiết kiệm tại bộ mã hoá và/hoặc bộ giải mã.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất việc phân vùng các ô thành các CTU; và ấn định các địa chỉ của các CTU đó trong đơn vị VCL NAL dựa trên các bộ nhận dạng (IDentifier - ID) ô tương ứng với các ô đó. Ví dụ, việc đánh địa chỉ các CTU dựa trên ID ô cho phép các địa chỉ được suy ra dựa trên ID ô, điều này có thể thu gọn luồng bit. Ngoài ra, việc đánh địa chỉ dựa trên ô đối với các CTU hỗ trợ việc lập mã CTU mà không tham chiếu đến các tiến trình lập mã xảy ra bên ngoài lát cắt hiện tại. Như vậy, việc đánh địa chỉ các CTU dựa trên ID ô hỗ trợ việc xử lý song song, và do đó, hỗ trợ việc tăng tốc độ giải mã tại bộ giải mã.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất việc còn bao gồm bước báo hiệu tường minh, trong mào đầu lát cắt, các địa chỉ của các CTU trong đơn vị VCL NAL.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng các địa chỉ của các CTU trong đơn vị VCL NAL là được lược bỏ khỏi luồng bit để hỗ trợ việc xác định các địa chỉ của các CTU tại bộ giải mã dựa trên ID ô của ô đầu tiên được chứa trong đơn vị VCL NAL. Theo sáng chế, thì các lát cắt và các CTU là được đánh địa chỉ dựa trên ID ô, và đơn vị VCL NAL chứa lát cắt đơn. Theo đó, bộ giải mã có thể xác định CTU nào là CTU cuối cùng trong lát cắt và theo đó là CTU cuối cùng trong đơn vị VCL NAL mà không cần cờ đó. Vì chuỗi viđeo chứa nhiều khung và mỗi khung chứa nhiều CTU, nên việc lược bỏ bit dữ liệu khỏi mỗi thao tác mã hoá CTU có thể thu gọn luồng bit một cách đáng kể. Như vậy, các tài nguyên mạng mà được dùng để truyền luồng bit này là được giảm bớt. Ngoài ra, việc sử dụng tài nguyên bộ nhớ ở cả bộ mã hoá và bộ giải mã được giảm bớt.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng các CTU không chứa cờ chỉ thị CTU cuối cùng trong đơn vị VCL NAL, và trong đó đơn vị VCL NAL này chứa bit đệm ngay sau CTU cuối cùng trong mỗi ô.

Theo một phương án, sáng chế đề xuất thiết bị lập mã viđeo bao gồm: bộ xử lý, bộ thu được ghép nối đến bộ xử lý, và bộ phát được ghép nối đến bộ xử lý, bộ xử lý,

bộ thu, và bộ phát này được tạo cấu hình để thực hiện phương pháp theo bất kỳ trong số các khía cạnh nêu trên.

Theo một phương án, sáng chế đề xuất phương tiện phi nhất thời đọc được bằng máy tính bao gồm sản phẩm chương trình máy tính để sử dụng bởi thiết bị lập mã viđeo, sản phẩm chương trình máy tính này bao gồm các chỉ dẫn thực thi được bằng máy tính được lưu giữ trên phương tiện phi nhất thời đọc được bằng máy tính này, để khi được thực thi bởi bộ xử lý thì khiến thiết bị lập mã viđeo này thực hiện phương pháp theo bất kỳ trong số các khía cạnh nêu trên.

Theo một phương án, sáng chế đề xuất bộ giải mã bao gồm phương tiện nhận để nhận luồng bit bao gồm đơn vị VCL NAL (Network Abstraction Layer - lớp trừu tượng mạng) chứa lát cắt của dữ liệu ảnh được chia thành các ô. Bộ giải mã này còn bao gồm phương tiện xác định để xác định số lượng ô trong đơn vị VCL NAL này, và xác định số lượng độ dịch điểm vào dành cho các ô này dưới dạng số lượng ô trong đơn vị VCL NAL này trừ đi một, trong đó mỗi trong số các độ dịch điểm vào này chỉ thị vị trí bắt đầu của ô tương ứng trong đơn vị VCL NAL, và trong đó số lượng độ dịch điểm vào này là không được báo hiệu tường minh trong luồng bit này. Bộ giải mã này còn bao gồm phương tiện thu thập để thu thập các độ dịch điểm vào dành cho các ô dựa trên số lượng độ dịch điểm vào. Bộ giải mã này còn bao gồm phương tiện giải mã để giải mã các ô tại các độ dịch điểm vào để tạo ra hình ảnh được tái tạo.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng bộ mã hoá còn được tạo cấu hình để thực hiện phương pháp theo bất kỳ trong số các khía cạnh nêu trên.

Theo một phương án, sáng chế đề xuất bộ mã hoá bao gồm phương tiện phân vùng để phân vùng hình ảnh thành ít nhất một lát cắt và phân vùng ít nhất một lát cắt này thành các ô. Bộ mã hoá này còn bao gồm phương tiện mã hoá để mã hoá các ô này vào luồng bit trong ít nhất một đơn vị VCL NAL, mã hoá một số lượng ô trong đơn vị VCL NAL trong luồng bit này, và mã hoá các độ dịch điểm vào dành cho các ô trong luồng bit này, trong đó mỗi trong số các độ dịch điểm vào này chỉ thị vị trí bắt đầu của ô tương ứng trong đơn vị VCL NAL, và trong đó một số lượng độ dịch điểm vào là không được báo hiệu tường minh trong luồng bit này. Bộ mã hoá này còn bao gồm phương tiện truyền để truyền luồng bit này mà không có số lượng độ dịch điểm vào để

hỗ trợ việc giải mã các ô theo sự suy luận rằng số lượng độ dịch điểm vào dành cho các ô là số lượng ô trong đơn vị VCL NAL trừ đi một.

Tuỳ ý, theo bất kỳ trong số các khía cạnh nêu trên, thì cách thức thực hiện khác của khía cạnh đó đề xuất rằng bộ mã hoá còn được tạo cấu hình để thực hiện phương pháp theo bất kỳ trong số các khía cạnh nêu trên.

Để cho rõ ràng, thì phương án bất kỳ trong số các phương án nêu trên là có thể được kết hợp với bất kỳ một hoặc nhiều phương án còn lại trong số các phương án nêu trên để tạo ra phương án mới trong phạm vi của sáng chế.

Các dấu hiệu này và các dấu hiệu khác sẽ được hiểu rõ hơn từ phần mô tả chi tiết sau đây dựa vào các hình vẽ kèm theo và các điểm yêu cầu bảo hộ.

Mô tả vắn tắt các hình vẽ

Để hiểu hoàn chỉnh hơn về sáng chế, thì các hình vẽ kèm theo sẽ được mô tả trong phần mô tả vắn tắt sau đây và phần mô tả chi tiết, trong đó các số chỉ dẫn giống nhau biểu thị các bộ phận giống nhau.

Fig.1 là hình vẽ thể hiện lưu đồ của phương pháp ví dụ để lập mã tín hiệu viđeo.

Fig.2 là hình vẽ thể hiện sơ đồ của hệ thống lập mã và giải mã (coding and decoding - codec) ví dụ để lập mã viđeo.

Fig.3 là hình vẽ thể hiện sơ đồ bộ mã hoá viđeo ví dụ.

Fig.4 là hình vẽ thể hiện sơ đồ bộ giải mã viđeo ví dụ.

Fig.5 là hình vẽ thể hiện sơ đồ luồng bit ví dụ có chứa chuỗi viđeo được mã hoá.

Fig.6 là hình vẽ thể hiện sơ đồ hình ảnh ví dụ được phân vùng để lập mã.

Fig.7 là hình vẽ thể hiện sơ đồ thiết bị lập mã viđeo ví dụ.

Fig.8 là hình vẽ thể hiện lưu đồ của phương pháp ví dụ để mã hoá hình ảnh vào luồng bit.

Fig.9 là hình vẽ thể hiện lưu đồ của phương pháp ví dụ để giải mã hình ảnh từ luồng bit.

Fig.10 là hình vẽ thể hiện sơ đồ của hệ thống ví dụ để lập mã chuỗi viđeo của các hình ảnh trong luồng bit.

Mô tả chi tiết sáng chế

Cần hiểu rằng tuy cách thức thực hiện minh hoạ của một hoặc nhiều phương án được cung cấp dưới đây, nhưng các hệ thống và/hoặc các phương pháp được bộc lộ có thể được thực hiện nhờ sử dụng số lượng kỹ thuật bất kỳ, cho dù là đã biết hiện tại hoặc đang tồn tại. Sáng chế không chỉ giới hạn ở những cách thức thực hiện minh hoạ đó, các hình vẽ đó, và các kỹ thuật được thể hiện dưới đây, bao gồm các thiết kế và những cách thức thực hiện ví dụ được thể hiện và được mô tả ở đây, mà có thể được cải biến trong phạm vi của các điểm yêu cầu bảo hộ kèm theo cùng với đầy đủ phạm vi của các phương án tương đương của chúng.

Nhiều kỹ thuật nén viđeo có thể được sử dụng để giảm kích thước của các tệp tin viđeo với sự tổn thất dữ liệu nhỏ nhất. Ví dụ, các kỹ thuật nén viđeo có thể bao gồm việc thực hiện việc dự đoán về không gian (ví dụ, nội ảnh) và/hoặc việc dự đoán về thời gian (ví dụ, liên ảnh) để giảm hoặc loại bỏ sự trùng lặp dữ liệu trong các chuỗi viđeo. Đối với việc lập mã viđeo dựa trên khối, thì lát cắt viđeo (ví dụ, ảnh viđeo hoặc một phần của ảnh viđeo) có thể được phân vùng thành các khối viđeo, mà cũng có thể được gọi là các khối cây, các khối cây lập mã (Coding Tree Block - CTB), các đơn vị cây lập mã (Coding Tree Unit - CTU), các đơn vị lập mã (Coding Unit - CU), và/hoặc các nút lập mã. Các khối viđeo trong lát cắt được nội lập mã (I) của ảnh là được lập mã nhờ sử dụng việc dự đoán trong không gian dựa vào các mẫu tham chiếu trong các khối lân cận trong cùng một ảnh. Các khối viđeo trong lát cắt dự đoán một chiều (P) hoặc lát cắt dự đoán hai chiều (B) được liên lập mã của ảnh là có thể được lập mã nhờ sử dụng việc dự đoán trong không gian dựa vào các mẫu tham chiếu trong các khối lân cận trong cùng ảnh hoặc việc dự đoán về mặt thời gian dựa vào các mẫu tham chiếu trong các ảnh tham chiếu khác. Các ảnh có thể được gọi là các khung và/hoặc các hình ảnh, và các ảnh tham chiếu có thể được gọi là các khung tham chiếu và/hoặc các hình ảnh tham chiếu. Việc dự đoán trong không gian hoặc dự đoán về mặt thời gian cho ra khối dự đoán biểu diễn khối ảnh. Dữ liệu dư biểu thị những sự khác biệt điểm ảnh giữa khối ảnh gốc và khối dự đoán. Theo đó, khối được liên lập mã là được mã hoá theo vectơ chuyển động mà trỏ đến khối của các mẫu tham chiếu mà tạo thành khối dự đoán và dữ liệu dư chỉ thị sự khác biệt giữa khối được lập mã và khối dự đoán. Khối được nội lập mã là được mã hoá theo chế độ nội lập mã và dữ liệu dư đó. Để nén hơn nữa, thì dữ liệu dư có thể được biến đổi từ miền điểm ảnh sang miền biến đổi. Những việc này cho ra các

hệ số biến đổi dư, mà có thể được lượng tử hoá. Các hệ số biến đổi được lượng tử hoá có thể được sắp xếp lúc đầu trong mảng hai chiều. Các hệ số biến đổi được lượng tử hoá này có thể được quét theo thứ tự để tạo ra vectơ một chiều của các hệ số biến đổi. Việc lập mã entropy có thể được áp dụng để đạt được khả năng nén thậm chí hơn nữa. Các kỹ thuật nén viđeo đó được mô tả chi tiết hơn dưới đây.

Để đảm bảo việc viđeo được mã hoá có thể được giải mã chính xác, thì viđeo được mã hoá và được giải mã theo các tiêu chuẩn lập mã viđeo tương ứng. Các tiêu chuẩn lập mã viđeo bao gồm tiêu chuẩn ITU (International Telecommunication Union - hiệp hội viễn thông quốc tế) Standardization Sector (bộ phận tiêu chuẩn hoá) (ITU-T) H.261, ISO (International Organization for Standardization - tổ chức tiêu chuẩn hoá quốc tế)/IEC (International Electrotechnical Commission - uỷ ban kỹ thuật điện quốc tế) MPEG (Motion Picture Experts Group - nhóm chuyên gia ảnh động)-1 Part (phần) 2, ITU-T H.262 hoặc ISO/IEC MPEG-2 Part 2, ITU-T H.263, ISO/IEC MPEG-4 Part 2, AVC (Advanced Video Coding - lập mã viđeo tiên tiến), còn được gọi là ITU-T H.264 hoặc ISO/IEC MPEG-4 Part 10, và HEVC (High Efficiency Video Coding - lập mã viđeo hiệu quả cao), còn được gọi là ITU-T H.265 hoặc MPEG-H Part 2. AVC bao gồm các mở rộng chẳng hạn như SVC (Scalable Video Coding - lập mã viđeo khả biến), MVC (Multiview Video Coding - lập mã viđeo đa điểm quan sát) và MVC+D (Multiview Video Coding plus Depth - lập mã viđeo đa điểm quan sát thêm chiều sâu), và 3D-AVC (3 Dimensional AVC - AVC ba chiều). HEVC bao gồm các mở rộng chẳng hạn như SHVC (Scalable HEVC - HEVC khả biến), MV-HEVC (Multiview HEVC - HEVC đa điểm quan sát), và 3D HEVC (3D-HEVC). Nhóm hợp tác chuyên gia viđeo (Joint Video Experts Team - JVET) của ITU-T và ISO/IEC đã bắt đầu phát triển tiêu chuẩn lập mã viđeo được gọi là tiêu chuẩn lập mã viđeo vạn năng (Versatile Video Coding - VVC). VVC được bao gồm trong dự thảo làm việc (Working Draft - WD) mà bao gồm JVET-K1001-v4 và JVET-K1002-v1.

Để lập mã hình ảnh viđeo, thì trước hết hình ảnh đó được phân vùng, và các phân vùng này được lập mã vào luồng bit. Có các sơ đồ phân vùng ảnh khác nhau khả dụng. Ví dụ, hình ảnh có thể được phân vùng thành các lát cắt bình thường, các lát cắt phụ thuộc, các ô, và/hoặc theo kỹ thuật xử lý song song mặt sóng (Wavefront Parallel Processing - WPP). Để cho đơn giản, thì HEVC hạn chế các bộ mã hoá sao cho chỉ có các lát cắt bình thường, các lát cắt phụ thuộc, các ô, WPP, và các tổ hợp của chúng, mới

có thể được sử dụng khi phân vùng lát cắt thành các nhóm CTB để lập mã viđeo. Việc phân vùng đó có thể được áp dụng để hỗ trợ việc so khớp kích thước đơn vị vận chuyển lớn nhất (Maximum Transfer Unit - MTU), xử lý song song, và giảm độ trễ giữa các đầu. MTU biểu thị lượng dữ liệu lớn nhất mà có thể được truyền trong một gói đơn. Nếu phần tải hữu ích của gói vượt quá MTU, thì phần tải hữu ích đó được chia thành hai gói thông qua tiến trình được gọi là phân mảnh.

Lát cắt bình thường, còn được gọi đơn giản là lát cắt, là phần được phân vùng của hình ảnh mà có thể được tái tạo độc lập với các lát cắt bình thường khác trong cùng ảnh, tuy nhiên vẫn có một số sự phụ thuộc nhau do các hoạt động lọc vòng. Mỗi lát cắt bình thường được gói trong đơn vị NAL (Network Abstraction Layer - lớp trừu tượng mạng) của riêng nó để truyền. Ngoài ra, sự phụ thuộc của việc dự đoán trong ảnh (dự đoán nội mẫu, dự đoán thông tin chuyển động, dự đoán chế độ lập mã) và việc lập mã entropy giữa các đường biên lát cắt là có thể được vô hiệu hoá để hỗ trợ việc tái tạo độc lập. Việc tái tạo độc lập đó hỗ trợ việc song song hoá. Ví dụ, việc song song hoá dựa trên lát cắt bình thường sử dụng việc truyền thông liên bộ xử lý hoặc liên lõi nhỏ nhất. Tuy nhiên, vì mỗi lát cắt bình thường là độc lập, nên mỗi lát cắt được liên kết với mào đầu lát cắt riêng biệt. Việc sử dụng các lát cắt bình thường có thể gây ra phí tổn lập mã lớn do chi phí bit của mào đầu lát cắt đối với mỗi lát cắt và do việc thiếu sự dự đoán giữa các đường biên lát cắt. Ngoài ra, các lát cắt bình thường có thể được sử dụng để hỗ trợ việc so khớp đối với các yêu cầu kích thước MTU. Cụ thể là, vì lát cắt bình thường được gói trong đơn vị NAL riêng biệt và có thể được lập mã độc lập, nên mỗi lát cắt bình thường cần nhỏ hơn MTU trong các sơ đồ MTU để tránh việc phá vỡ lát cắt thành nhiều gói. Như vậy, mục đích của việc song song hoá và mục đích của việc so khớp kích thước MTU có thể đặt ra các yêu cầu mâu thuẫn đối với bố cục lát cắt trong ảnh.

Các lát cắt phụ thuộc là tương tự như các lát cắt bình thường, nhưng có các mào đầu lát cắt được rút ngắn và cho phép phân vùng các đường biên khối cây hình ảnh mà không phá vỡ việc dự đoán trong ảnh. Theo đó, các lát cắt phụ thuộc cho phép phân mảnh lát cắt bình thường thành các đơn vị NAL, để giảm độ trễ giữa các đầu bằng cách cho phép gửi một phần của lát cắt bình thường ra ngoài trước khi hoàn tất việc mã hoá toàn bộ lát cắt bình thường.

Ô là phần được phân vùng của hình ảnh mà được tạo ra bởi đường biên ngang và đường biên dọc mà tạo ra các cột và các hàng ô. Các ô có thể được lập mã theo thứ tự quét mành (phải sang trái và đỉnh xuống đáy). Thứ tự quét của các CTB là cục bộ trong ô. Theo đó, các CTB trong ô đầu tiên là được lập mã theo thứ tự quét mành, trước khi tiến đến các CTB trong ô tiếp theo. Tương tự như các lát cắt bình thường, thì các ô phá vỡ những sự phụ thuộc dự đoán trong ảnh cũng như những sự phụ thuộc giải mã entropy. Tuy nhiên, các ô có thể không được bao gồm vào các đơn vị NAL riêng lẻ, và do đó, các ô có thể không được dùng để so khớp kích thước MTU. Mỗi ô có thể được xử lý bởi một bộ xử lý/lõi, và hoạt động truyền thông liên bộ xử lý/liên lõi mà được sử dụng cho việc dự đoán trong ảnh giữa các đơn vị xử lý mà giải mã các ô lân cận là có thể được giới hạn để mang mào đầu lát cắt dùng chung (khi các ô liền kề ở cùng lát cắt), và thực hiện việc chia sẻ liên quan đến lọc vòng đối với các mẫu được tái tạo và siêu dữ liệu. Khi nhiều hơn một ô được bao gồm trong lát cắt, thì độ dịch byte của điểm vào đối với mỗi ô mà không phải là độ dịch điểm vào đầu tiên trong lát cắt là có thể được báo hiệu trong mào đầu lát cắt. Đối với mỗi lát cắt và ô, thì ít nhất một trong số các điều kiện sau đây cần được thoả mãn: 1) tất cả các khối cây được lập mã trong lát cắt là thuộc về cùng ô; và 2) tất cả các khối cây được lập mã trong ô là thuộc về cùng lát cắt.

Theo WPP, thì hình ảnh được phân vùng thành các hàng đơn gồm các CTB. Việc giải mã entropy và các cơ chế dự đoán có thể sử dụng dữ liệu từ các CTB ở các hàng khác. Việc xử lý song song được làm cho khả thi thông qua việc giải mã song song đối với các hàng CTB. Ví dụ, hàng hiện tại có thể được giải mã song song với hàng đằng trước. Tuy nhiên, việc giải mã hàng hiện tại được làm trễ tính từ tiến trình giải mã các hàng đằng trước là hai CTB. Độ trễ này bảo đảm rằng dữ liệu liên quan đến CTB bên trên và CTB bên trên và sang phải của CTB hiện tại ở hàng hiện tại là khả dụng trước khi CTB hiện tại được lập mã. Cách tiếp cận này xuất hiện như mặt sóng khi được biểu diễn bằng đồ thị. Hình ảnh chứa bao nhiêu hàng CTB thì sự bắt đầu so le này cho phép song song hoá với bấy nhiêu bộ xử lý/lõi. Vì việc dự đoán trong ảnh giữa các hàng khối cây lân cận trong ảnh là được cho phép, nên hoạt động truyền thông liên bộ xử lý/liên lõi để cho phép dự đoán trong ảnh có thể là lớn. Việc phân vùng WPP có tính đến các kích thước đơn vị NAL. Do đó, WPP không hỗ trợ việc so khớp kích thước MTU. Tuy nhiên, các lát cắt bình thường có thể được sử dụng cùng với WPP, với phí tổn lập mã nhất định, để thực hiện việc so khớp kích thước MTU theo ý muốn.

Các ô cũng có thể bao gồm các tập hợp ô bị ràng buộc chuyển động. Tập hợp ô bị ràng buộc chuyển động (Motion Constrained Tile Set - MCTS) là tập hợp ô được thiết kế sao cho các vectơ chuyển động được liên kết là được hạn chế để trỏ đến các vị trí của mẫu đầy đủ bên trong MCTS này và đến các vị trí của phân đoạn mẫu mà chỉ yêu cầu các vị trí của mẫu đầy đủ bên trong MCTS này để nội suy. Ngoài ra, việc sử dụng các ứng viên vectơ chuyển động cho việc dự đoán vectơ chuyển động về thời gian mà được trích xuất từ các khối bên ngoài MCTS là không được phép. Theo cách này, thì mỗi MCTS có thể được giải mã độc lập mà không cần sự tồn tại của các ô không được bao gồm trong MCTS. Các thông điệp MCTS SEI (Supplemental Enhancement Information - thông tin tăng cường bổ sung) về thời gian có thể được dùng để chỉ thị sự tồn tại của các MCTS trong luồng bit và báo hiệu các MCTS này. Thông điệp MCTS SEI cung cấp thông tin bổ sung mà có thể được sử dụng trong quá trình trích xuất luồng bit con MCTS (được quy định như một phần ngữ nghĩa của thông điệp SEI) để tạo ra luồng bit hợp cách cho MCTS. Thông tin này bao gồm một số lượng tập hợp thông tin trích xuất, mỗi tập hợp xác định một số lượng MCTS và chứa các byte RBSP (Raw Bytes Sequence Payload - phần tải hữu ích của chuỗi byte thô) của các tập hợp thông số viđeo (Video Parameter Set - VPS) thay thế, các tập hợp thông số chuỗi (Sequence Parameter Set - SPS), và các tập hợp thông số ảnh (Picture Parameter Set - PPS) cần được sử dụng trong tiến trình trích xuất luồng bit con MCTS. Khi trích xuất luồng bit con theo tiến trình trích xuất luồng bit con MCTS, thì các tập hợp thông số (VPS, SPS, và PPS) có thể được viết lại hoặc được thay thế, và các mào đầu lát cắt có thể được cập nhật bởi vì một hoặc tất cả trong số các phần tử cú pháp liên quan đến địa chỉ lát cắt (bao gồm first_slice_segment_in_pic_flag và slice_segment_address) có thể sử dụng các giá trị khác nhau trong luồng bit con trích xuất được.

Các cơ chế tạo ô và cắt lát nêu trên cung cấp sự linh hoạt đáng kể để hỗ trợ việc so khớp kích thước MTU và xử lý song song. Tuy nhiên, việc so khớp kích thước MTU đã trở nên ít liên quan hơn do tốc độ và độ tin cậy ngày càng tăng của các mạng viễn thông. Ví dụ, một trong số các tác dụng chính của việc so khớp kích thước MTU là để hỗ trợ việc hiển thị các ảnh được giấu lỗi. Ảnh được giấu lỗi là ảnh được giải mã mà được tạo ra từ ảnh được lập mã được truyền khi có sự mất dữ liệu. Sự mất dữ liệu đó có thể bao gồm sự mất một số lát cắt của ảnh được lập mã hoặc các lỗi ở các ảnh tham chiếu được sử dụng bởi ảnh được lập mã (ví dụ, ảnh tham chiếu cũng là ảnh được

giấu lỗi). Ảnh được giấu lỗi có thể được tạo ra bằng cách hiển thị các lát cắt đúng và ước lượng các lát cắt lỗi, ví dụ, bằng cách sao chép lát cắt tương ứng với lát cắt lỗi từ ảnh trước đó trong chuỗi viđeo. Các ảnh được giấu lỗi có thể được tạo ra khi mỗi lát cắt được chứa trong đơn vị NAL đơn. Tuy nhiên, nếu các lát cắt được phân mảnh trên nhiều đơn vị NAL (ví dụ, không có sự so khớp kích thước MTU), thì việc mất một đơn vị NAL có thể làm hỏng nhiều lát cắt. Việc tạo ra các ảnh được giấu lỗi là ít liên quan hơn trong các môi trường mạng hiện đại vì sự mất gói là tình trạng ít phổ biến hơn nhiều và vì tốc độ của mạng hiện đại cho phép hệ thống lược bỏ hoàn toàn các ảnh có lỗi mà không gây ra sự đứng hình viđeo đáng kể do độ trễ giữa ảnh lỗi và ảnh không lỗi theo sau thường là nhỏ. Ngoài ra, tiến trình để ước lượng chất lượng của ảnh được giấu lỗi có thể phức tạp, và do đó, việc chỉ cần lược bỏ ảnh lỗi là có thể được ưu tiên. Do đó, các ứng dụng đàm thoại viđeo, chẳng hạn hội nghị viđeo và điện thoại viđeo, và thậm chí là các ứng dụng phát quảng bá, thường bỏ việc sử dụng các ảnh được giấu lỗi.

Vì các ảnh được giấu lỗi là ít hữu ích, nên việc so khớp kích thước MTU cũng ít hữu ích. Ngoài ra, việc tiếp tục hỗ trợ các mô hình so khớp kích thước MTU khi phân vùng có thể làm phức tạp các hệ thống lập mã một cách không cần thiết và sử dụng các bit lãng phí mà vốn có thể được lược bỏ để tăng hiệu quả lập mã. Ngoài ra, một số sơ đồ tạo ô (ví dụ, MCTS) cho phép hiển thị các ảnh con của ảnh. Để hiển thị ảnh con, thì các lát cắt trong vùng cần quan tâm là được hiển thị và các lát cắt khác thì được lược bỏ. Vùng cần quan tâm có thể bắt đầu tại vị trí không phải là phần trên cùng bên trái của ảnh, và do đó, có thể có các địa chỉ được dịch khỏi điểm bắt đầu của ảnh một giá trị biến thiên. Để hiển thị ảnh con một cách phù hợp, thì bộ nối có thể được dùng để viết lại (các) mào đầu lát cắt cho vùng cần quan tâm để tính đến độ dịch này. Sơ đồ cắt lát và tạo ô mà không yêu cầu việc viết lại mào đầu lát cắt đó sẽ có lợi. Ngoài ra, các đường biên ô có thể không được coi như các đường biên ảnh trừ khi chúng được đặt cùng vị trí với các đường biên ảnh. Tuy nhiên, việc coi các đường biên ô như các đường biên ảnh có thể tăng hiệu quả lập mã trong một số trường hợp do việc đệm các đường biên và do việc nới lỏng các ràng buộc liên quan đến các vectơ chuyển động mà trỏ đến các mẫu bên ngoài các đường biên trong các ảnh tham chiếu. Ngoài ra, HEVC có thể sử dụng cờ tên là end_of_slice_flag ở cuối dữ liệu được lập mã đối với mỗi CTU để chỉ thị xem đã đạt đến cuối lát cắt hay chưa. AVC sử dụng cờ này ở cuối dữ liệu được lập mã đối với mỗi khối macro (khối lớn) (MacroBlock - MB) cho cùng mục đích. Tuy

nhiên, việc lập mã cờ này là không cần thiết và lãng phí bit khi CTU/MB cuối cùng là đã biết thông qua các cơ chế khác. Sáng chế đề xuất các cơ chế để khắc phục các vấn đề này và các vấn đề khác trong lĩnh vực lập mã viđeo.

Các cơ chế khác nhau được bộc lộ ở đây để tăng hiệu quả lập mã và giảm phí tổn xử lý liên quan đến các sơ đồ cắt lát và tạo ô nêu trên. Theo một ví dụ, thì các lát cắt được yêu cầu phải bao gồm một số lượng nguyên các ô và mỗi lát cắt được lưu giữ trong đơn vị VCL (Video Coding Layer - lớp lập mã viđeo) NAL riêng biệt. Ngoài ra, số lượng ô trong lát cắt có thể được tính toán dựa trên ô góc trên bên trái và ô góc dưới cùng bên phải của lát cắt. Sau đó, số lượng ô này có thể được dùng để tính toán các giá trị khác mà nhờ đó có thể được lược bỏ khỏi luồng bit. Theo một ví dụ cụ thể, thì địa chỉ của mỗi ô trong luồng bit là độ dịch điểm vào tính từ đầu lát cắt, và do đó, là từ đầu đơn vị VCL NAL. Số lượng độ dịch điểm vào có thể được tính toán là số lượng ô trừ đi một (vì ô đầu tiên có độ dịch là không và do đó được lược bỏ). Sau đó, số lượng độ dịch điểm vào này có thể được sử dụng khi truy hồi các độ dịch điểm vào dành cho các ô từ mào đầu lát cắt tương ứng với lát cắt đó. Ngoài ra, các CTU trong các ô có thể được đánh địa chỉ dưới dạng hàm theo bộ nhận dạng (IDentifier - ID) của ô chứa các CTU tương ứng. Sau đó, các địa chỉ của các CTU có thể được báo hiệu tường minh hoặc được trích xuất, tuỳ theo phương án, dựa trên các ID ô và số lượng ô tính toán được. Ngoài ra, việc biết số lượng ô trong đơn vị VCL NAL và số lượng CTU trong các ô sẽ cho phép bộ giải mã xác định CTU cuối cùng trong đơn vị VCL NAL. Theo đó, cờ end_of_slice_flag, mà trước đó được bao gồm trong mỗi CTU để chỉ thị CTU cuối cùng trong lát cắt, là có thể được lược bỏ khỏi luồng bit, điều này dẫn đến việc tiết kiệm được một bit đối với mỗi CTU trong toàn bộ chuỗi viđeo. Các ví dụ này và các ví dụ khác được mô tả chi tiết dưới đây.

Fig.1 là hình vẽ thể hiện lưu đồ của phương pháp vận hành 100 ví dụ để lập mã tín hiệu viđeo. Cụ thể là, tín hiệu viđeo được mã hoá tại bộ mã hoá. Tiến trình mã hoá nén tín hiệu viđeo này bằng cách sử dụng các cơ chế khác nhau để giảm kích thước tệp tin viđeo. Kích thước tệp tin nhỏ hơn sẽ cho phép truyền tệp tin viđeo được nén đó về phía người dùng, trong khi giảm được phí tổn băng thông liên quan. Sau đó, bộ giải mã giải mã tệp tin viđeo được nén này để tái tạo tín hiệu viđeo gốc để hiển thị cho người dùng cuối. Tiến trình giải mã thường là đối xứng gương với tiến trình mã hoá để cho phép bộ giải mã tái tạo tín hiệu viđeo một cách nhất quán.

Ở bước 101, tín hiệu viđeo được nhập vào bộ mã hoá. Ví dụ, tín hiệu viđeo này có thể là tệp tin viđeo không được nén mà được lưu giữ trong bộ nhớ. Theo ví dụ khác, thì tệp tin viđeo này có thể được ghi bởi thiết bị ghi viđeo, chẳng hạn camera viđeo, và được mã hoá để hỗ trợ việc tạo luồng trực tiếp (live streaming) đối với viđeo này. Tệp tin viđeo này có thể bao gồm cả thành phần auđio và thành phần viđeo. Thành phần viđeo chứa hàng loạt khung hình ảnh mà khi được xem theo trình tự thì cho cảm nhận thị giác về sự chuyển động. Các khung này chứa các điểm ảnh mà được biểu diễn về mặt ánh sáng, ở đây được gọi là các thành phần độ chói (hay các mẫu độ chói), và màu sắc, mà được gọi là các thành phần sắc độ (hay các mẫu màu). Theo một số ví dụ, thì các khung này cũng có thể chứa các giá trị chiều sâu để hỗ trợ xem ba chiều.

Ở bước 103, viđeo được phân vùng thành các khối. Việc phân vùng bao gồm việc chia tiếp các điểm ảnh trong mỗi khung thành các khối hình vuông và/hoặc hình chữ nhật để nén. Ví dụ, theo tiêu chuẩn lập mã viđeo hiệu quả cao (High Efficiency Video Coding - HEVC) (còn được gọi là H.265 và MPEG-H Part 2), thì khung có thể trước hết được chia thành các đơn vị cây lập mã (Coding Tree Unit - CTU), tức là các khối có kích thước được định trước (ví dụ, sáu mươi tư điểm ảnh nhân sáu mươi tư điểm ảnh). Các CTU này chứa cả các mẫu độ chói và các mẫu sắc độ. Các cây lập mã có thể được sử dụng để chia các CTU thành các khối và sau đó chia tiếp các khối này theo cách đệ quy cho đến khi đạt được các cấu hình hỗ trợ việc mã hoá tiếp. Ví dụ, các thành phần độ chói của khung có thể được chia tiếp cho đến khi các khối riêng lẻ chứa các giá trị sáng tương đối đồng nhất. Ngoài ra, các thành phần sắc độ của khung có thể được chia tiếp cho đến khi các khối riêng lẻ chứa các giá trị màu tương đối đồng nhất. Theo đó, các cơ chế phân vùng là biến thiên tuỳ theo nội dung của các khung viđeo.

Ở bước 105, các cơ chế nén khác nhau được sử dụng để nén các khối ảnh được phân vùng ở bước 103. Ví dụ, cơ chế liên dự đoán và/hoặc nội dự đoán có thể được sử dụng. Cơ chế liên dự đoán được thiết kế để tận dụng thực tế rằng các đối tượng trong khung cảnh chung có xu hướng xuất hiện trong các khung liên tiếp. Theo đó, khối thể hiện đối tượng trong khung tham chiếu không cần phải được mô tả lặp lại trong các khung liền kề. Cụ thể là, một đối tượng, chẳng hạn chiếc bàn, có thể vẫn nằm ở vị trí không đổi trên nhiều khung. Do đó, chiếc bàn đó được mô tả một lần và các khung liền kề có thể tham chiếu trở lại khung tham chiếu. Các cơ chế so khớp mẫu có thể được sử dụng để so khớp các đối tượng trên nhiều khung. Ngoài ra, các đối tượng chuyển động

có thể được biểu diễn trên nhiều khung, ví dụ, do sự chuyển động của đối tượng hoặc sự chuyển động của camera. Theo một ví dụ cụ thể, thì một viđeo có thể thể hiện chiếc ô tô di chuyển qua màn hình trên nhiều khung. Các vectơ chuyển động có thể được sử dụng để mô tả sự chuyển động đó. Vectơ chuyển động là vectơ hai chiều mà cung cấp độ dịch từ các toạ độ của đối tượng trong khung đến các toạ độ của đối tượng trong khung tham chiếu. Như vậy, cơ chế liên dự đoán có thể mã hoá khối ảnh trong khung hiện tại dưới dạng tập hợp các vectơ chuyển động chỉ thị độ dịch từ khối tương ứng trong khung tham chiếu.

Cơ chế nội dự đoán mã hoá các khối trong khung chung. Cơ chế nội dự đoán tận dụng thực tế là các thành phần độ chói và sắc độ có xu hướng xếp thành nhóm trong khung. Ví dụ, một mảng màu xanh lục ở một phần của một cái cây thì có xu hướng được đặt kề với các mảng màu xanh lục tương tự. Cơ chế nội dự đoán sử dụng nhiều chế độ dự đoán có hướng (ví dụ, ba mươi ba chế độ theo HEVC), chế độ phẳng, và chế độ dòng điện một chiều (Direct Current - DC). Các chế độ có hướng này chỉ thị rằng khối hiện tại là tương tự/giống với các mẫu của khối lân cận theo hướng tương ứng. Chế độ phẳng chỉ thị rằng một loạt khối dọc theo hàng/cột (ví dụ, mặt phẳng) là có thể được nội suy dựa trên các khối lân cận tại các mép của hàng. Trên thực tế, chế độ phẳng chỉ thị sự chuyển tiếp trơn của ánh sáng/màu sắc trên hàng/cột bằng cách sử dụng độ dốc gần như không đổi trong việc thay đổi các giá trị. Chế độ DC được sử dụng để làm nhẵn đường biên và chỉ thị rằng khối là tương tự/giống với giá trị trung bình được liên kết với các mẫu của tất cả các khối lân cận mà được liên kết với các hướng xiên góc của các chế độ dự đoán có hướng. Theo đó, các khối nội dự đoán có thể biểu thị các khối ảnh dưới dạng các giá trị của chế độ dự đoán có quan hệ khác nhau thay vì các giá trị thực tế. Ngoài ra, các khối liên dự đoán có thể biểu thị các khối ảnh dưới dạng các giá trị vectơ chuyển động thay vì các giá trị thực tế. Trong trường hợp nào trong số hai trường hợp đó, thì các khối dự đoán cũng có thể không biểu thị một cách chính xác các khối ảnh trong một số trường hợp. Những sự khác biệt bất kỳ là được lưu giữ ở các khối dư. Các phép biến đổi có thể được áp dụng cho các khối dư để nén tệp tin hơn nữa.

Ở bước 107, các kỹ thuật lọc khác nhau có thể được áp dụng. Theo HEVC, thì các bộ lọc là được áp dụng theo sơ đồ lọc trong vòng lặp. Việc dự đoán dựa trên khối nêu trên có thể dẫn đến việc tạo ra các hình ảnh có khối tại bộ giải mã. Ngoài ra, sơ đồ dự đoán dựa trên khối có thể mã hoá khối và sau đó tái tạo khối được mã hoá để về sau

sử dụng làm khối tham chiếu. Sơ đồ lọc trong vòng lặp này áp dụng lặp đi lặp lại các bộ lọc khử nhiễu, các bộ lọc khử khối, các bộ lọc vòng thích ứng, và các bộ lọc dịch thích ứng mẫu (Sample Adaptive Offset - SAO) cho các khối/các khung. Các bộ lọc này giảm bớt các thành phần giả tạo khối đó để tệp tin được mã hoá có thể được tái tạo chính xác. Ngoài ra, các bộ lọc này giảm bớt các thành phần giả ở các khối tham chiếu tái tạo được, để các thành phần giả ít có khả năng tạo ra các thành phần giả bổ sung ở các khối sau đó mà được mã hoá dựa trên các khối tham chiếu tái tạo được.

Khi tín hiệu viđeo đã được phân vùng, được nén, và được lọc, thì dữ liệu thu được được mã hoá trong luồng bit ở bước 109. Luồng bit này bao gồm dữ liệu nêu trên cũng như dữ liệu báo hiệu bất kỳ mà được mong muốn để hỗ trợ việc tái tạo tín hiệu viđeo phù hợp tại bộ giải mã. Ví dụ, dữ liệu đó có thể bao gồm dữ liệu phân vùng, dữ liệu dự đoán, các khối dư, và các cờ khác nhau mà cung cấp các chỉ dẫn lập mã cho bộ giải mã. Luồng bit này có thể được lưu giữ trong bộ nhớ để truyền về phía bộ giải mã khi có yêu cầu. Luồng bit này cũng có thể được phát quảng bá và/hoặc phát đa điểm về phía các bộ giải mã. Việc tạo ra luồng bit này là tiến trình lặp đi lặp lại. Theo đó, các bước 101, 103, 105, 107, và 109 có thể xuất hiện liên tục và/hoặc đồng thời trên nhiều khung và khối. Thứ tự được thể hiện trên Fig.1 là được thể hiện để cho việc mô tả được rõ ràng và dễ dàng, chứ không nhằm giới hạn tiến trình lập mã viđeo ở thứ tự cụ thể nào.

Bộ giải mã nhận luồng bit này và bắt đầu tiến trình giải mã ở bước 111. Cụ thể là, bộ giải mã sử dụng sơ đồ giải mã entropy để chuyển đổi luồng bit này thành dữ liệu cú pháp và dữ liệu viđeo tương ứng. Bộ giải mã sử dụng dữ liệu cú pháp này từ luồng bit để xác định các phân vùng đối với các khung ở bước 111. Việc phân vùng này cần khớp với các kết quả của việc phân vùng khối ở bước 103. Việc mã hoá/giải mã entropy như được sử dụng ở bước 111 sẽ được mô tả. Bộ mã hoá chọn nhiều sự lựa chọn trong tiến trình nén, chẳng hạn chọn các sơ đồ phân vùng khối từ một số sự lựa chọn khả thi dựa trên sự định vị trong không gian của các giá trị ở (các) hình ảnh đầu vào. Việc báo hiệu những sự lựa chọn chính xác này có thể sử dụng một lượng lớn các bin. Như được sử dụng ở đây, bin là giá trị nhị phân mà được coi như biến số (ví dụ, giá trị bit mà có thể biến thiên tuỳ theo ngữ cảnh). Việc lập mã entropy cho phép bộ mã hoá loại bỏ các tuỳ chọn bất kỳ mà rõ ràng là không khả thi cho trường hợp cụ thể, để lại tập hợp các tuỳ chọn có thể được phép. Sau đó, mỗi tuỳ chọn có thể được phép được

ấn định từ mã. Chiều dài của các từ mã là dựa trên số lượng tuỳ chọn có thể được phép (ví dụ, một bin cho hai tuỳ chọn, hai bin cho ba đến bốn tuỳ chọn, v.v.). Sau đó, bộ mã hoá mã hoá từ mã đó đối với tuỳ chọn được chọn. Sơ đồ này giảm kích thước của các từ mã vì các từ mã là lớn theo ý muốn để chỉ thị duy nhất một sự lựa chọn từ tập hợp con cỡ nhỏ gồm các tuỳ chọn có thể được phép, ngược lại so với việc chỉ thị duy nhất sự lựa chọn đó từ tập hợp có thể là lớn gồm tất cả các tuỳ chọn khả thi. Sau đó, bộ giải mã giải mã sự lựa chọn này bằng cách xác định tập hợp các tuỳ chọn có thể được phép theo cách tương tự như bộ mã hoá. Bằng cách xác định tập hợp các tuỳ chọn có thể được phép, thì bộ giải mã có thể đọc từ mã đó và xác định sự lựa chọn mà được chọn bởi bộ mã hoá.

Ở bước 113, bộ giải mã thực hiện việc giải mã khối. Cụ thể là, bộ giải mã sử dụng các phép biến đổi ngược để tạo ra các khối dư. Sau đó, bộ giải mã sử dụng các khối dư đó và các khối dự đoán tương ứng để tái tạo các khối ảnh theo việc phân vùng. Các khối dự đoán này có thể bao gồm cả các khối nội dự đoán và các khối liên dự đoán như được tạo ra tại bộ mã hoá ở bước 105. Sau đó, các khối ảnh được tái tạo được đặt vào các khung của tín hiệu viđeo tái tạo được theo dữ liệu phân vùng xác định được ở bước 111. Cú pháp cho bước 113 cũng có thể được báo hiệu trong luồng bit thông qua việc lập mã entropy như đã mô tả trên đây.

Ở bước 115, việc lọc được thực hiện trên các khung của tín hiệu viđeo tái tạo được theo cách tương tự như bước 107 tại bộ mã hoá. Ví dụ, các bộ lọc khử nhiễu, các bộ lọc khử khối, các bộ lọc vòng thích ứng, và các bộ lọc SAO có thể được áp dụng cho các khung để loại bỏ các thành phần giả tạo khối. Khi các khung được lọc, thì tín hiệu viđeo có thể được xuất đến thiết bị hiển thị ở bước 117 để người dùng cuối xem.

Fig.2 là hình vẽ thể hiện sơ đồ của hệ thống lập mã và giải mã (coding and decoding - codec) 200 ví dụ để lập mã viđeo. Cụ thể là, hệ thống codec 200 cung cấp chức năng để hỗ trợ việc thực hiện phương pháp vận hành 100. Hệ thống codec 200 được tổng quát hoá để thể hiện các thành phần được sử dụng ở cả bộ mã hoá và bộ giải mã. Hệ thống codec 200 nhận và phân vùng tín hiệu viđeo như được mô tả dựa vào các bước 101 và 103 ở phương pháp vận hành 100, để thu được tín hiệu viđeo được phân vùng 201. Sau đó, hệ thống codec 200 nén tín hiệu viđeo được phân vùng 201 này vào luồng bit được mã hoá khi hoạt động như bộ mã hoá như được mô tả dựa vào các bước 105, 107, và 109 ở phương pháp 100. Khi hoạt động như bộ giải mã thì hệ thống codec

200 tạo ra tín hiệu viđeo đầu ra từ luồng bit này như được mô tả dựa vào các bước 111, 113, 115, và 117 ở phương pháp vận hành 100. Hệ thống codec 200 bao gồm thành phần điều khiển bộ lập mã tổng 211, thành phần biến đổi, thay đổi tỷ lệ và lượng tử hoá 213, thành phần ước lượng nội ảnh 215, thành phần dự đoán nội ảnh 217, thành phần bù chuyển động 219, thành phần ước lượng chuyển động 221, thành phần thay đổi tỷ lệ và biến đổi ngược 229, thành phần phân tích điều khiển bộ lọc 227, thành phần bộ lọc trong vòng lặp 225, thành phần bộ đệm ảnh được giải mã 223, và thành phần định dạng mào đầu và lập mã số học nhị phân thích ứng ngữ cảnh (Context Adaptive Binary Arithmetic Coding - CABAC) 231. Các thành phần đó được ghép nối như được thể hiện. Trên Fig.2, thì các đường màu đen chỉ thị sự di chuyển của dữ liệu cần được mã hoá/giải mã, còn các đường nét đứt thì chỉ thị sự di chuyển của dữ liệu điều khiển mà điều khiển sự hoạt động của các thành phần khác. Các thành phần của hệ thống codec 200 có thể đều có mặt ở bộ mã hoá. Bộ giải mã có thể bao gồm tập hợp con của các thành phần của hệ thống codec 200. Ví dụ, bộ giải mã có thể bao gồm thành phần dự đoán nội ảnh 217, thành phần bù chuyển động 219, thành phần thay đổi tỷ lệ và biến đổi ngược 229, thành phần bộ lọc trong vòng lặp 225, và thành phần bộ đệm ảnh được giải mã 223. Các thành phần này sẽ được mô tả.

Tín hiệu viđeo được phân vùng 201 là chuỗi viđeo ghi được mà đã được phân vùng thành các khối điểm ảnh bởi cây lập mã. Cây lập mã sử dụng các chế độ chia khác nhau để chia tiếp khối điểm ảnh thành các khối điểm ảnh nhỏ hơn. Sau đó, các khối này có thể còn được chia tiếp thành các khối nhỏ hơn. Các khối này có thể được gọi là các nút trên cây lập mã. Các nút mẹ lớn hơn được chia thành các nút con nhỏ hơn. Số lần nút được chia nhỏ được gọi là chiều sâu của nút/cây lập mã. Các khối được chia có thể được bao gồm trong các đơn vị lập mã (Coding Unit - CU) trong một số trường hợp. Ví dụ, CU có thể là phần con của CTU mà chứa khối độ chói, (các) khối sắc độ chênh lệch đỏ (Cr), và (các) khối sắc độ chênh lệch xanh lam (Cb) cùng với các chỉ dẫn cú pháp tương ứng dành cho CU đó. Các chế độ chia có thể bao gồm cây nhị phân (Binary Tree - BT), cây tam phân (Triple Tree - TT), và cây tứ phân (Quad Tree - QT) được sử dụng để phân vùng nút lần lượt thành hai, ba, hoặc bốn nút con, với các hình dạng biến thiên tuỳ theo các chế độ chia được sử dụng. Tín hiệu viđeo được phân vùng 201 được chuyển tiếp đến thành phần điều khiển bộ lập mã tổng 211, thành phần biến đổi, thay đổi tỷ lệ

và lượng tử hoá 213, thành phần ước lượng nội ảnh 215, thành phần phân tích điều khiển bộ lọc 227, và thành phần ước lượng chuyển động 221 để nén.

Thành phần điều khiển bộ lập mã tổng 211 được tạo cấu hình để đưa ra những quyết định liên quan đến việc lập mã các hình ảnh của chuỗi viđeo vào luồng bit theo các ràng buộc ứng dụng. Ví dụ, thành phần điều khiển bộ lập mã tổng 211 quản lý việc tối ưu hoá tốc độ bit/kích thước luồng bit so với chất lượng tái tạo. Những quyết định đó có thể được đưa ra dựa trên không gian lưu trữ/sự khả dụng của băng thông và các yêu cầu về độ phân giải hình ảnh. Thành phần điều khiển bộ lập mã tổng 211 còn quản lý việc sử dụng bộ đệm dựa vào tốc độ truyền để giảm bớt vấn đề chạy dưới mức bộ đệm và vấn đề tràn bộ đệm. Để quản lý các vấn đề này, thì thành phần điều khiển bộ lập mã tổng 211 quản lý việc phân vùng, việc dự đoán, và việc lọc của các thành phần còn lại. Ví dụ, thành phần điều khiển bộ lập mã tổng 211 có thể, theo cách động, tăng sự phức tạp nén để tăng độ phân giải và tăng mức sử dụng băng thông, hoặc giảm sự phức tạp nén để giảm độ phân giải và mức sử dụng băng thông. Do đó, thành phần điều khiển bộ lập mã tổng 211 điều khiển các thành phần còn lại của hệ thống codec 200 để cân bằng giữa chất lượng tái tạo tín hiệu viđeo và tốc độ bit. Thành phần điều khiển bộ lập mã tổng 211 tạo ra dữ liệu điều khiển, mà điều khiển sự hoạt động của các thành phần còn lại. Dữ liệu điều khiển này cũng được chuyển tiếp đến thành phần định dạng mào đầu và CABAC 231 để được mã hoá trong luồng bit để báo hiệu các thông số cho việc giải mã tại bộ giải mã.

Tín hiệu viđeo được phân vùng 201 cũng được gửi đến thành phần ước lượng chuyển động 221 và thành phần bù chuyển động 219 để liên dự đoán. Khung hoặc lát cắt của tín hiệu viđeo được phân vùng 201 có thể được chia thành nhiều khối viđeo. Thành phần ước lượng chuyển động 221 và thành phần bù chuyển động 219 thực hiện việc lập mã liên dự đoán đối với khối viđeo nhận được so với một hoặc nhiều khối trong một hoặc nhiều khung tham chiếu để cung cấp việc dự đoán về mặt thời gian. Hệ thống codec 200 có thể thực hiện nhiều chặng lập mã, ví dụ, để chọn chế độ lập mã thích hợp cho mỗi khối của dữ liệu viđeo.

Thành phần ước lượng chuyển động 221 và thành phần bù chuyển động 219 có thể được tích hợp ở mức độ cao, nhưng được thể hiện riêng biệt nhằm mục đích minh hoạ. Việc ước lượng chuyển động, mà được thực hiện bởi thành phần ước lượng chuyển động 221, là tiến trình tạo ra các vectơ chuyển động, để ước lượng sự chuyển động cho

các khối viđeo. Ví dụ, vectơ chuyển động có thể chỉ thị sự dịch chuyển của đối tượng được lập mã so với khối dự đoán. Khối dự đoán là khối mà được tìm thấy là gần như khớp với khối cần được lập mã, xét về sự khác biệt điểm ảnh. Khối dự đoán cũng có thể được gọi là khối tham chiếu. Sự khác biệt điểm ảnh đó có thể được xác định bởi tổng chênh lệch tuyệt đối (Sum of Absolute Difference - SAD), tổng chênh lệch bình phương (Sum of Square Difference - SSD), hoặc các phép đo (metric - mêtric) chênh lệch khác. HEVC sử dụng một số đối tượng được lập mã bao gồm CTU, các khối cây lập mã (Coding Tree Block - CTB), và các CU. Ví dụ, CTU có thể được chia thành các CTB, mà sau đó có thể được chia thành các CB để bao gồm trong các CU. CU có thể được mã hoá dưới dạng đơn vị dự đoán (Prediction Unit - PU) chứa dữ liệu dự đoán và/hoặc đơn vị biến đổi (Transform Unit - TU) chứa dữ liệu dư được biến đổi dành cho CU đó. Thành phần ước lượng chuyển động 221 tạo ra các vectơ chuyển động, các PU, và các TU nhờ sử dụng việc phân tích tốc độ - độ méo như một phần của tiến trình tối ưu hoá tốc độ và độ méo. Ví dụ, thành phần ước lượng chuyển động 221 có thể xác định nhiều khối tham chiếu, nhiều vectơ chuyển động, v.v., dành cho khối/khung hiện tại, và có thể chọn các khối tham chiếu, các vectơ chuyển động, v.v., mà có các đặc điểm tốc độ - độ méo tốt nhất. Các đặc điểm tốc độ - độ méo tốt nhất này cân bằng cả chất lượng của việc tái tạo viđeo (ví dụ, lượng mất mát dữ liệu do việc nén) với hiệu quả lập mã (ví dụ, kích thước của việc mã hoá cuối cùng).

Theo một số ví dụ, thì hệ thống codec 200 có thể tính toán các giá trị dành cho các vị trí điểm ảnh nguyên con của các ảnh tham chiếu được lưu giữ trong thành phần bộ đệm ảnh được giải mã 223. Ví dụ, hệ thống codec viđeo 200 có thể nội suy các giá trị của các vị trí phần tư điểm ảnh, các vị trí phần tám điểm ảnh, hoặc các vị trí phân đoạn điểm ảnh khác của ảnh tham chiếu. Do đó, thành phần ước lượng chuyển động 221 có thể thực hiện việc tìm kiếm chuyển động so với các vị trí điểm ảnh đầy đủ và các vị trí phân đoạn điểm ảnh và xuất ra vectơ chuyển động với độ chính xác của phân đoạn điểm ảnh. Thành phần ước lượng chuyển động 221 tính toán vectơ chuyển động dành cho PU của khối viđeo trong lát cắt được liên lập mã bằng cách so sánh vị trí của PU với vị trí của khối dự đoán của ảnh tham chiếu. Thành phần ước lượng chuyển động 221 xuất vectơ chuyển động tính toán được dưới dạng dữ liệu chuyển động đến thành phần định dạng mào đầu và CABAC 231 để mã hoá và sự chuyển động đến thành phần bù chuyển động 219.

Việc bù chuyển động, mà được thực hiện bởi thành phần bù chuyển động 219, là có thể bao gồm việc tìm nạp hoặc tạo ra khối dự đoán dựa trên vectơ chuyển động được xác định bởi thành phần ước lượng chuyển động 221. Như đã nêu, thành phần ước lượng chuyển động 221 và thành phần bù chuyển động 219 có thể được tích hợp về mặt chức năng, theo một số ví dụ. Khi nhận được vectơ chuyển động dành cho PU của khối viđeo hiện tại, thì thành phần bù chuyển động 219 có thể định vị khối dự đoán mà vectơ chuyển động này trỏ đến đó. Sau đó, khối viđeo dư được tạo ra bằng cách trừ đi các giá trị điểm ảnh của khối dự đoán từ các giá trị điểm ảnh của khối viđeo hiện tại đang được lập mã, để tạo ra các giá trị chênh lệch điểm ảnh. Nói chung, thì thành phần ước lượng chuyển động 221 thực hiện việc ước lượng chuyển động so với các thành phần độ chói, và thành phần bù chuyển động 219 sử dụng các vectơ chuyển động tính được dựa trên các thành phần độ chói này cho cả các thành phần sắc độ và các thành phần độ chói. Khối dự đoán và khối dư này được chuyển tiếp đến thành phần biến đổi, thay đổi tỷ lệ và lượng tử hoá 213.

Tín hiệu viđeo được phân vùng 201 cũng được gửi đến thành phần ước lượng nội ảnh 215 và thành phần dự đoán nội ảnh 217. Cũng như đối với thành phần ước lượng chuyển động 221 và thành phần bù chuyển động 219, thì thành phần ước lượng nội ảnh 215 và thành phần dự đoán nội ảnh 217 có thể được tích hợp mức độ cao, nhưng được thể hiện riêng biệt nhằm mục đích minh hoạ. Thành phần ước lượng nội ảnh 215 và thành phần dự đoán nội ảnh 217 nội dự đoán khối hiện tại so với các khối trong khung hiện tại, thay vì việc liên dự đoán mà được thực hiện bởi thành phần ước lượng chuyển động 221 và thành phần bù chuyển động 219 giữa các khung, như đã mô tả trên đây. Cụ thể là, thành phần ước lượng nội ảnh 215 xác định chế độ nội dự đoán để sử dụng để mã hoá khối hiện tại. Theo một số ví dụ, thì thành phần ước lượng nội ảnh 215 chọn chế độ nội dự đoán thích hợp để mã hoá khối hiện tại từ nhiều chế độ nội dự đoán đã được kiểm tra. Sau đó, các chế độ nội dự đoán được chọn được chuyển tiếp đến thành phần định dạng mào đầu và CABAC 231 để mã hoá.

Ví dụ, thành phần ước lượng nội ảnh 215 tính toán các giá trị tốc độ - độ méo nhờ sử dụng việc phân tích tốc độ - độ méo đối với các chế độ nội dự đoán khác nhau đã được kiểm tra, và chọn chế độ nội dự đoán có các đặc điểm tốc độ - độ méo tốt nhất trong số các chế độ đã được kiểm tra đó. Việc phân tích tốc độ - độ méo thường xác định lượng méo (hoặc sai số) giữa khối được mã hoá và khối gốc chưa được mã hoá mà

đã được mã hoá để tạo ra khối được mã hoá, cũng như tốc độ bit (ví dụ, số lượng bit) được sử dụng để tạo ra khối được mã hoá. Thành phần ước lượng nội ảnh 215 tính toán các tỷ số từ những sự méo và các tốc độ đối với các khối được mã hoá khác nhau để xác định xem chế độ nội dự đoán nào có giá trị tốc độ - độ méo tốt nhất cho khối đó. Ngoài ra, thành phần ước lượng nội ảnh 215 có thể được tạo cấu hình để lập mã các khối chiều sâu của bản đồ chiều sâu nhờ sử dụng chế độ dựng mô hình chiều sâu (Depth Modeling Mode - DMM) dựa trên sự tối ưu hoá tốc độ - độ méo (Rate-Distortion Optimization - RDO).

Thành phần dự đoán nội ảnh 217 có thể tạo ra khối dư từ khối dự đoán dựa trên các chế độ nội dự đoán được chọn mà được xác định bởi thành phần ước lượng nội ảnh 215 khi được thực hiện trên bộ mã hoá, hoặc đọc khối dư từ luồng bit khi được thực hiện trên bộ giải mã. Khối dư này bao gồm sự chênh lệch về các giá trị giữa khối dự đoán và khối gốc, được biểu diễn dưới dạng ma trận. Sau đó, khối dư này được chuyển tiếp đến thành phần biến đổi, thay đổi tỷ lệ và lượng tử hoá 213. Thành phần ước lượng nội ảnh 215 và thành phần dự đoán nội ảnh 217 có thể hoạt động trên cả thành phần độ chói và thành phần sắc độ.

Thành phần biến đổi, thay đổi tỷ lệ và lượng tử hoá 213 được tạo cấu hình để nén tiếp khối dư. Thành phần biến đổi, thay đổi tỷ lệ và lượng tử hoá 213 áp dụng phép biến đổi, chẳng hạn biến đổi cosin rời rạc (Discrete Cosine Transform - DCT), biến đổi sin rời rạc (Discrete Sine Transform - DST), hoặc phép biến đổi tương tự về mặt khái niệm, đối với khối dư, để tạo ra khối viđeo bao gồm các giá trị hệ số biến đổi dư. Phép biến đổi wavelet, biến đổi nguyên, biến đổi dải con hoặc các loại phép biến đổi khác cũng có thể được sử dụng. Phép biến đổi này có thể chuyển đổi thông tin dư từ miền giá trị điểm ảnh sang miền biến đổi, chẳng hạn miền tần số. Thành phần biến đổi, thay đổi tỷ lệ và lượng tử hoá 213 cũng được tạo cấu hình để thay đổi tỷ lệ thông tin dư đã được biến đổi, ví dụ, dựa trên tần số. Việc thay đổi tỷ lệ đó bao gồm việc áp dụng hệ số tỷ lệ cho thông tin dư sao cho các thông tin tần số khác nhau được lượng tử hoá tại các độ chi tiết khác nhau, điều này có thể ảnh hưởng đến chất lượng hiển thị cuối cùng của viđeo tái tạo được. Thành phần biến đổi, thay đổi tỷ lệ và lượng tử hoá 213 cũng được tạo cấu hình để lượng tử hoá các hệ số biến đổi để giảm tốc độ bit hơn nữa. Tiến trình lượng tử hoá có thể giảm chiều sâu bit được liên kết với một số hoặc tất cả trong số các hệ số đó. Mức độ lượng tử hoá có thể được cải biến bằng cách điều chỉnh thông số lượng

tử hoá. Theo một số ví dụ, thì sau đó thành phần biến đổi, thay đổi tỷ lệ và lượng tử hoá 213 có thể thực hiện việc quét ma trận bao gồm các hệ số biến đổi đã được lượng tử hoá. Các hệ số biến đổi đã được lượng tử hoá được chuyển tiếp đến thành phần định dạng mào đầu và CABAC 231 để được mã hoá trong luồng bit.

Thành phần thay đổi tỷ lệ và biến đổi ngược 229 áp dụng thao tác ngược của thành phần biến đổi, thay đổi tỷ lệ và lượng tử hoá 213 để hỗ trợ việc ước lượng chuyển động. Thành phần thay đổi tỷ lệ và biến đổi ngược 229 áp dụng việc thay đổi tỷ lệ, biến đổi, và/hoặc lượng tử hoá ngược để tái tạo khối dư trong miền điểm ảnh, ví dụ, để sử dụng về sau làm khối tham chiếu mà có thể trở thành khối dự đoán đối với khối hiện tại khác. Thành phần ước lượng chuyển động 221 và/hoặc thành phần bù chuyển động 219 có thể tính toán khối tham chiếu bằng cách cộng khối dư trở lại khối dự đoán tương ứng để sử dụng trong việc ước lượng chuyển động của khối/khung về sau. Các bộ lọc được áp dụng cho các khối tham chiếu tái tạo được để giảm bớt các thành phần giả được tạo ra trong quá trình thay đổi tỷ lệ, lượng tử hoá, và biến đổi. Các thành phần giả đó vốn có thể gây ra việc dự đoán không chính xác (và tạo ra các thành phần giả bổ sung) khi các khối sau đó được dự đoán.

Thành phần phân tích điều khiển bộ lọc 227 và thành phần bộ lọc trong vòng lặp 225 áp dụng các bộ lọc cho các khối dư và/hoặc cho các khối ảnh được tái tạo. Ví dụ, khối dư đã được biến đổi từ thành phần thay đổi tỷ lệ và biến đổi ngược 229 có thể được kết hợp với khối dự đoán tương ứng từ thành phần dự đoán nội ảnh 217 và/hoặc thành phần bù chuyển động 219 để tái tạo khối ảnh gốc. Sau đó, các bộ lọc này có thể được áp dụng cho khối ảnh được tái tạo. Theo một số ví dụ thì thay vào đó, các bộ lọc này có thể được áp dụng cho các khối dư. Cũng như đối với các thành phần khác trên Fig.2, thì thành phần phân tích điều khiển bộ lọc 227 và thành phần bộ lọc trong vòng lặp 225 là được tích hợp mức độ cao và có thể được thực hiện cùng nhau, nhưng được thể hiện riêng biệt nhằm mục đích minh hoạ. Các bộ lọc mà được áp dụng cho các khối tham chiếu tái tạo được là được áp dụng cho các vùng không gian cụ thể và bao gồm nhiều thông số để điều chỉnh việc các bộ lọc đó được áp dụng như thế nào. Thành phần phân tích điều khiển bộ lọc 227 phân tích các khối tham chiếu tái tạo được để xác định xem các bộ lọc đó nên được áp dụng ở đâu và thiết đặt các thông số tương ứng. Dữ liệu đó được chuyển tiếp đến thành phần định dạng mào đầu và CABAC 231 dưới dạng dữ liệu điều khiển bộ lọc để mã hoá. Thành phần bộ lọc trong vòng lặp 225 áp dụng các bộ

lọc đó dựa trên dữ liệu điều khiển bộ lọc này. Các bộ lọc đó có thể bao gồm bộ lọc khử khối, bộ lọc khử nhiễu, bộ lọc SAO, và bộ lọc vòng thích ứng. Các bộ lọc đó có thể được áp dụng trong miền không gian/miền điểm ảnh (ví dụ, trên khối điểm ảnh tái tạo được) hoặc trong miền tần số, tuỳ theo ví dụ.

Khi hoạt động như bộ mã hoá, thì khối ảnh được tái tạo đã được lọc, khối dư, và/hoặc khối dự đoán là được lưu giữ ở thành phần bộ đệm ảnh được giải mã 223 để sử dụng về sau trong quá trình ước lượng chuyển động như đã mô tả trên đây. Khi hoạt động như bộ giải mã, thì thành phần bộ đệm ảnh được giải mã 223 lưu giữ và chuyển tiếp các khối được tái tạo và được lọc về phía thiết bị hiển thị như một phần của tín hiệu viđeo đầu ra. Thành phần bộ đệm ảnh được giải mã 223 có thể là thiết bị nhớ bất kỳ mà có khả năng lưu giữ các khối dự đoán, các khối dư, và/hoặc các khối ảnh được tái tạo.

Thành phần định dạng mào đầu và CABAC 231 nhận dữ liệu từ các thành phần khác nhau của hệ thống codec 200 và mã hoá dữ liệu đó vào luồng bit được lập mã để truyền về phía bộ giải mã. Cụ thể là, thành phần định dạng mào đầu và CABAC 231 tạo ra các mào đầu khác nhau để mã hoá dữ liệu điều khiển, chẳng hạn dữ liệu điều khiển tổng và dữ liệu điều khiển bộ lọc. Ngoài ra, dữ liệu dự đoán, bao gồm dữ liệu nội dự đoán và dữ liệu chuyển động, cũng như dữ liệu dư dưới dạng dữ liệu hệ số biến đổi đã được lượng tử hoá, đều được mã hoá trong luồng bit này. Luồng bit cuối cùng bao gồm tất cả thông tin cần thiết cho bộ giải mã để tái tạo tín hiệu viđeo được phân vùng gốc 201. Thông tin đó cũng có thể bao gồm các bảng chỉ số chế độ nội dự đoán (còn được gọi là các bảng ánh xạ từ mã), các định nghĩa về các ngữ cảnh mã hoá dành cho các khối khác nhau, các chỉ thị về các chế độ nội dự đoán khả dĩ nhất, chỉ thị về thông tin về phân vùng, v.v.. Dữ liệu đó có thể được mã hoá bằng cách lập mã entropy. Ví dụ, thông tin này có thể được mã hoá bằng cách sử dụng kỹ thuật lập mã chiều dài biến thiên thích ứng ngữ cảnh (Context Adaptive Variable Length Coding - CAVLC), CABAC, lập mã số học nhị phân thích ứng ngữ cảnh dựa trên cú pháp (Syntax-based context-adaptive Binary Arithmetic Coding - SBAC), lập mã entropy phân vùng khoảng xác suất (Probability Interval Partitioning Entropy - PIPE), hoặc kỹ thuật lập mã entropy khác. Sau khi lập mã entropy, thì luồng bit được lập mã có thể được truyền đến thiết bị khác (ví dụ, bộ giải mã viđeo) hoặc được lưu trữ để truyền hoặc truy hồi về sau.

Fig.3 là hình vẽ thể hiện sơ đồ khối của bộ mã hoá viđeo 300 ví dụ. Bộ mã hoá viđeo 300 có thể được sử dụng để thực hiện các chức năng mã hoá của hệ thống

codec 200 và/hoặc thực hiện các bước 101, 103, 105, 107, và/hoặc 109 của phương pháp vận hành 100. Bộ mã hoá 300 phân vùng tín hiệu viđeo đầu vào, để thu được tín hiệu viđeo được phân vùng 301, mà về cơ bản là tương tự như tín hiệu viđeo được phân vùng 201. Sau đó, tín hiệu viđeo được phân vùng 301 được nén và được mã hoá vào luồng bit bởi các thành phần của bộ mã hoá 300.

Cụ thể là, tín hiệu viđeo được phân vùng 301 được chuyển tiếp đến thành phần dự đoán nội ảnh 317 để nội dự đoán. Thành phần dự đoán nội ảnh 317 có thể về cơ bản là tương tự như thành phần ước lượng nội ảnh 215 và thành phần dự đoán nội ảnh 217. Tín hiệu viđeo được phân vùng 301 cũng được chuyển tiếp đến thành phần bù chuyển động 321 để liên dự đoán dựa trên các khối tham chiếu ở thành phần bộ đệm ảnh được giải mã 323. Thành phần bù chuyển động 321 có thể về cơ bản là tương tự như thành phần ước lượng chuyển động 221 và thành phần bù chuyển động 219. Các khối dự đoán và các khối dư từ thành phần dự đoán nội ảnh 317 và thành phần bù chuyển động 321 được chuyển tiếp đến thành phần biến đổi và lượng tử hoá 313 để biến đổi và lượng tử hoá các khối dư. Thành phần biến đổi và lượng tử hoá 313 có thể về cơ bản là tương tự như thành phần biến đổi, thay đổi tỷ lệ và lượng tử hoá 213. Các khối dư đã được biến đổi và lượng tử hoá và các khối dự đoán tương ứng (cùng với dữ liệu điều khiển liên quan) được chuyển tiếp đến thành phần lập mã entropy 331 để lập mã vào luồng bit. Thành phần lập mã entropy 331 có thể về cơ bản là tương tự như thành phần định dạng mào đầu và CABAC 231.

Các khối dư đã được biến đổi và lượng tử hoá và/hoặc các khối dự đoán tương ứng cũng được chuyển tiếp từ thành phần biến đổi và lượng tử hoá 313 đến thành phần biến đổi và lượng tử hoá ngược 329 để tái tạo thành các khối tham chiếu để thành phần bù chuyển động 321 sử dụng. Thành phần biến đổi và lượng tử hoá ngược 329 có thể về cơ bản là tương tự như thành phần thay đổi tỷ lệ và biến đổi ngược 229. Các bộ lọc trong vòng lặp ở thành phần bộ lọc trong vòng lặp 325 cũng được áp dụng cho các khối dư và/hoặc các khối tham chiếu tái tạo được, tuỳ theo ví dụ. Thành phần bộ lọc trong vòng lặp 325 có thể về cơ bản là tương tự như thành phần phân tích điều khiển bộ lọc 227 và thành phần bộ lọc trong vòng lặp 225. Thành phần bộ lọc trong vòng lặp 325 có thể bao gồm nhiều bộ lọc như được mô tả đối với thành phần bộ lọc trong vòng lặp 225. Sau đó, các khối được lọc được lưu giữ ở thành phần bộ đệm ảnh được giải mã 323 để thành phần bù chuyển động 321 sử dụng làm các khối tham chiếu. Thành phần bộ đệm

ảnh được giải mã 323 có thể về cơ bản là tương tự như thành phần bộ đệm ảnh được giải mã 223.

Fig.4 là hình vẽ thể hiện sơ đồ khối của bộ giải mã viđeo 400 ví dụ. Bộ giải mã viđeo 400 có thể được sử dụng để thực hiện các chức năng giải mã của hệ thống codec 200 và/hoặc thực hiện các bước 111, 113, 115, và/hoặc 117 của phương pháp vận hành 100. Bộ giải mã 400 nhận luồng bit, ví dụ, từ bộ mã hoá 300, và tạo ra tín hiệu viđeo đầu ra được tái tạo dựa trên luồng bit này để hiển thị cho người dùng cuối.

Luồng bit này được nhận bởi thành phần giải mã entropy 433. Thành phần giải mã entropy 433 được tạo cấu hình để thực hiện sơ đồ giải mã entropy, chẳng hạn CAVLC, CABAC, SBAC, lập mã PIPE, hoặc các kỹ thuật lập mã entropy khác. Ví dụ, thành phần giải mã entropy 433 có thể sử dụng thông tin mào đầu để cung cấp ngữ cảnh để diễn dịch dữ liệu bổ sung mà được mã hoá dưới dạng các từ mã trong luồng bit. Thông tin được giải mã bao gồm thông tin mong muốn bất kỳ để giải mã tín hiệu viđeo, chẳng hạn dữ liệu điều khiển tổng, dữ liệu điều khiển bộ lọc, thông tin về phân vùng, dữ liệu chuyển động, dữ liệu dự đoán, và các hệ số biến đổi đã được lượng tử hoá từ các khối dư. Các hệ số biến đổi đã được lượng tử hoá được chuyển tiếp đến thành phần biến đổi và lượng tử hoá ngược 429 để tái tạo thành các khối dư. Thành phần biến đổi và lượng tử hoá ngược 429 có thể tương tự như thành phần biến đổi và lượng tử hoá ngược 329.

Các khối dư được tái tạo và/hoặc các khối dự đoán được chuyển tiếp đến thành phần dự đoán nội ảnh 417 để tái tạo thành các khối ảnh dựa trên các hoạt động nội dự đoán. Thành phần dự đoán nội ảnh 417 có thể tương tự như thành phần ước lượng nội ảnh 215 và thành phần dự đoán nội ảnh 217. Cụ thể là, thành phần dự đoán nội ảnh 417 sử dụng các chế độ dự đoán để định vị khối tham chiếu trong khung và áp dụng khối dư cho kết quả để tái tạo các khối ảnh được nội dự đoán. Các khối ảnh được nội dự đoán tái tạo được và/hoặc các khối dư và dữ liệu liên dự đoán tương ứng được chuyển tiếp đến thành phần bộ đệm ảnh được giải mã 423 thông qua thành phần bộ lọc trong vòng lặp 425, mà có thể về cơ bản là tương tự lần lượt như thành phần bộ đệm ảnh được giải mã 223 và thành phần bộ lọc trong vòng lặp 225. Thành phần bộ lọc trong vòng lặp 425 lọc các khối ảnh được tái tạo, các khối dư và/hoặc các khối dự đoán, và thông tin đó được lưu giữ trong thành phần bộ đệm ảnh được giải mã 423. Các khối ảnh được tái tạo đến từ thành phần bộ đệm ảnh được giải mã 423 là được chuyển tiếp đến thành phần

bù chuyển động 421 để liên dự đoán. Thành phần bù chuyển động 421 có thể về cơ bản là tương tự như thành phần ước lượng chuyển động 221 và/hoặc thành phần bù chuyển động 219. Cụ thể là, thành phần bù chuyển động 421 sử dụng các vectơ chuyển động từ khối tham chiếu để tạo ra khối dự đoán và áp dụng khối dư cho kết quả để tái tạo khối ảnh. Các khối được tái tạo thu được cũng có thể được chuyển tiếp qua thành phần bộ lọc trong vòng lặp 425 đến thành phần bộ đệm ảnh được giải mã 423. Thành phần bộ đệm ảnh được giải mã 423 tiếp tục lưu giữ thêm các khối ảnh tái tạo được, mà có thể được tái tạo thành các khung thông qua thông tin về phân vùng. Các khung đó cũng có thể được đặt vào chuỗi. Chuỗi này được xuất về phía thiết bị hiển thị dưới dạng tín hiệu viđeo đầu ra tái tạo được.

Fig.5 là hình vẽ thể hiện sơ đồ luồng bit 500 ví dụ có chứa chuỗi viđeo được mã hoá. Ví dụ, luồng bit 500 có thể được tạo ra bởi hệ thống codec 200 và/hoặc bộ mã hoá 300 để giải mã bởi hệ thống codec 200 và/hoặc bộ giải mã 400. Theo ví dụ khác, thì luồng bit 500 có thể được tạo ra bởi bộ mã hoá ở bước 109 của phương pháp 100 để bộ giải mã sử dụng ở bước 111.

Luồng bit 500 này bao gồm tập hợp thông số chuỗi (Sequence Parameter Set - SPS) 510, các tập hợp thông số ảnh (Picture Parameter Set - PPS) 512, các mào đầu lát cắt 514, và dữ liệu ảnh 520. SPS 510 chứa dữ liệu chuỗi chung cho tất cả các ảnh trong chuỗi viđeo được chứa trong luồng bit 500 này. Dữ liệu đó có thể bao gồm việc định kích thước ảnh, chiều sâu bit, các thông số công cụ lập mã, các hạn chế tốc độ bit, v.v.. PPS 512 chứa các thông số riêng của mỗi ảnh. Do đó, có thể có một PPS 512 trên mỗi ảnh trong chuỗi viđeo. PPS 512 có thể chỉ thị các công cụ lập mã khả dụng cho các lát cắt trong các ảnh tương ứng, các thông số lượng tử hoá, các độ dịch, các thông số công cụ lập mã riêng của ảnh (ví dụ, các hoạt động điều khiển bộ lọc), v.v.. Mào đầu lát cắt 514 chứa các thông số riêng của mỗi lát cắt trong ảnh. Do đó, có thể có một mào đầu lát cắt 514 trên mỗi lát cắt trong chuỗi viđeo. Mào đầu lát cắt 514 có thể chứa thông tin về loại lát cắt, các số đếm thứ tự ảnh (Picture Order Count - POC), các danh sách ảnh tham chiếu, các trọng số dự đoán, các điểm vào của ô, các thông số khử khối, v.v..

Dữ liệu ảnh 520 chứa dữ liệu viđeo được mã hoá theo việc liên dự đoán và/hoặc nội dự đoán cũng như dữ liệu dư đã được biến đổi và lượng tử hoá tương ứng. Dữ liệu ảnh 520 đó được sắp xếp theo việc phân vùng mà được dùng để phân vùng hình ảnh trước khi mã hoá. Ví dụ, hình ảnh trong dữ liệu ảnh 520 được chia thành các lát cắt

521. Mỗi lát cắt 521 được chia tiếp thành các ô 523. Các ô 523 tiếp tục được chia thành các CTU 527. Các CTU 527 được chia tiếp thành các khối lập mã dựa trên các cây lập mã. Sau đó, các khối lập mã này có thể được mã hoá/giải mã theo các cơ chế dự đoán. Hình ảnh/ảnh có thể chứa một hoặc nhiều lát cắt 521. Một mào đầu lát cắt 514 được sử dụng trên mỗi lát cắt 521. Mỗi lát cắt 521 có thể chứa một hoặc nhiều ô 523, mà sau đó có thể chứa nhiều CTU 527.

Mỗi lát cắt 521 có thể là hình chữ nhật được xác định bởi ô 523 tại góc trên bên trái và ô 523 tại góc dưới cùng bên phải. Không giống như ở các hệ thống lập mã khác, thì lát cắt 521 có thể không đi ngang hết toàn bộ chiều rộng của ảnh. Lát cắt 521 là đơn vị nhỏ nhất mà có thể được bộ giải mã hiển thị một cách riêng biệt. Do đó, việc chia các lát cắt 521 thành các đơn vị nhỏ hơn sẽ cho phép tạo ra các ảnh con theo cách mà đủ chi tiết để hiển thị các vùng mong muốn của ảnh. Ví dụ, trong ngữ cảnh thực tế ảo (Virtual Reality - VR), thì ảnh có thể chứa toàn bộ quả cầu có thể nhìn thấy được của dữ liệu, nhưng người dùng chỉ có thể nhìn thấy ảnh con trên thiết bị hiển thị gắn trên đầu. Các lát cắt 521 nhỏ hơn sẽ cho phép báo hiệu riêng biệt các ảnh con đó. Lát cắt 521 cũng thường được báo hiệu trong đơn vị VCL NAL 533 riêng biệt. Ngoài ra, lát cắt 521 có thể không cho phép việc dự đoán dựa trên các lát cắt 521 khác, điều này cho phép lập mã mỗi lát cắt 521 một cách độc lập với các lát cắt 521 khác.

Các lát cắt 521 được phân vùng thành một số lượng nguyên các ô 523. Ô 523 là phần được phân vùng của lát cắt 521 mà được tạo ra bởi đường biên ngang và đường biên dọc. Các ô 523 có thể được lập mã theo thứ tự quét mành, và có thể hoặc không cho phép việc dự đoán dựa trên các ô 523 khác, tuỳ theo ví dụ. Mỗi ô 523 có thể có ID ô 524 duy nhất trong ảnh. ID ô 524 là bộ nhận dạng bằng số mà có thể được dùng để phân biệt ô 523 này với ô 523 khác. ID ô 524 có thể lấy giá trị của chỉ số ô mà tăng về mặt số theo thứ tự quét mành. Thứ tự quét mành là từ trái sang phải và từ đỉnh xuống đáy. Các ID ô 524 cũng có thể sử dụng các giá trị bằng số khác. Tuy nhiên, ID ô 524 luôn tăng theo thứ tự quét mành để hỗ trợ các hoạt động tính toán được mô tả ở đây. Ví dụ, các đường biên của lát cắt 521 có thể được xác định theo ID ô 524 của ô 523 tại góc trên bên trái của lát cắt 521 và ID ô 524 của ô 523 tại góc dưới cùng bên phải của lát cắt 521. Khi ID ô 524 là giá trị khác với chỉ số ô, thì cơ chế chuyển đổi có thể được báo hiệu trong luồng bit 500, ví dụ trong PPS 512. Ngoài ra, mỗi ô 523 có thể được liên kết với độ dịch điểm vào 525. Độ dịch điểm vào 525 chỉ thị vị trí của bit đầu tiên của dữ

liệu được lập mã được liên kết với ô 523. Ô 523 đầu tiên có thể có độ dịch điểm vào 525 bằng không và mỗi trong số các ô 523 nữa có thể có độ dịch điểm vào 525 bằng số lượng bit của dữ liệu được lập mã ở các ô 523 đi trước. Như vậy, số lượng độ dịch điểm vào 525 có thể được suy ra là số lượng ô 523 trừ đi một.

Các ô 523 tiếp tục được chia thành các CTU 527. CTU 527 là phần con của ô 523 mà có thể tiếp tục được chia nhỏ bởi cấu trúc cây lập mã thành các khối lập mã mà có thể được mã hoá bởi bộ mã hoá và được giải mã bởi bộ giải mã. Mỗi trong số các CTU 527 được liên kết với một địa chỉ CTU 529. Địa chỉ CTU 529 biểu thị vị trí của CTU 527 tương ứng trong luồng bit 500. Cụ thể là, địa chỉ CTU 529 có thể biểu thị vị trí của CTU 527 tương ứng trong đơn vị VCL NAL 533. Theo một số ví dụ, thì các địa chỉ CTU 529 dành cho các CTU 527 là có thể được báo hiệu tường minh, ví dụ, trong PPS 512. Theo các ví dụ khác, thì các địa chỉ CTU 529 có thể được trích xuất bởi bộ giải mã. Ví dụ, các địa chỉ CTU 529 có thể được ấn định dựa trên ID ô 524 của ô 523 mà chứa các CTU 527 tương ứng. Trong trường hợp đó, thì bộ giải mã có thể xác định các ô 523 trong lát cắt 521 dựa trên các ID ô 524 của các ô 523 phía trên bên trái và dưới cùng bên phải. Sau đó, bộ giải mã có thể sử dụng các ô 523 xác định được trong lát cắt 521 để xác định số lượng CTU 527 trong lát cắt 521. Ngoài ra, bộ giải mã có thể sử dụng các ID ô 524 đã biết và số lượng CTU 527 để xác định các địa chỉ CTU 529. Ngoài ra, vì bộ giải mã đã biết số lượng CTU 527, nên cờ mà chỉ thị xem mỗi CTU 527 có phải là CTU 527 cuối cùng trong đơn vị VCL NAL 533 hay không là có thể được lược bỏ. Điều này là vì bộ giải mã có thể xác định xem CTU 527 nào là CTU 527 cuối cùng trong đơn vị VCL NAL 533 nhờ việc biết số lượng CTU 527 trong lát cắt 521, mà được chứa trong đơn vị VCL NAL 533. Tuy nhiên, bit đệm có thể được đặt sau CTU 527 cuối cùng trong ô 523 để hỗ trợ việc phân biệt giữa các ô 523 theo một số ví dụ. Như có thể thấy, việc báo hiệu các đường biên lát cắt 521 dựa trên các ID ô 524 có thể cho phép bộ giải mã suy ra lượng dữ liệu đáng kể, mà sau đó có thể được lược bỏ khỏi luồng bit 500 để tăng hiệu quả lập mã.

Luồng bit 500 được đặt vào các đơn vị VCL NAL 533 và các đơn vị không phải VCL NAL 531. Đơn vị NAL là đơn vị dữ liệu được lập mã được tạo kích thước để được đặt làm phần tải hữu ích cho gói đơn để truyền qua mạng. Đơn vị VCL NAL 533 là đơn vị NAL mà chứa dữ liệu viđeo được lập mã. Ví dụ, mỗi đơn vị VCL NAL 533 có thể chứa một lát cắt 521 của dữ liệu bao gồm các ô 523, các CTU 527, và các khối

lập mã tương ứng. Đơn vị không phải VCL NAL 531 là đơn vị NAL mà chứa cú pháp hỗ trợ, nhưng không chứa dữ liệu viđeo được lập mã. Ví dụ, đơn vị không phải VCL NAL 531 có thể chứa SPS 510, PPS 512, mào đầu lát cắt 514, v.v.. Như vậy, bộ giải mã nhận luồng bit 500 trong các đơn vị VCL NAL 533 và các đơn vị không phải VCL NAL 531 rời rạc. Trong các ứng dụng tạo luồng, thì bộ giải mã có thể giải mã dữ liệu viđeo hiện tại mà không cần đợi nhận được toàn bộ luồng bit 500. Như vậy, các ID ô 524, các độ dịch điểm vào 525, và các địa chỉ CTU 529 cho phép bộ giải mã định vị đúng dữ liệu viđeo trong đơn vị VCL NAL 533 để cho phép các cơ chế giải mã nhanh, xử lý song song, và các cơ chế hiển thị viđeo khác. Theo đó, việc tính toán các ID ô 524, các độ dịch điểm vào 525, và/hoặc các địa chỉ CTU 529 sẽ cho phép thức thực hiện hiệu quả các cơ chế giải mã và hiển thị trong khi giảm được kích thước của luồng bit 500 và nhờ đó tăng hiệu quả lập mã.

Fig.6 là hình vẽ thể hiện sơ đồ hình ảnh 600 ví dụ được phân vùng để lập mã. Ví dụ, hình ảnh 600 có thể được mã hoá vào và được giải mã từ luồng bit 500, ví dụ, bởi hệ thống codec 200, bộ mã hoá 300, và/hoặc bộ giải mã 400. Ngoài ra, hình ảnh 600 có thể được phân vùng để hỗ trợ việc mã hoá và giải mã theo phương pháp 100.

Hình ảnh 600 có thể được phân vùng thành các lát cắt 621, các ô 623, và các CTU 627, mà có thể về cơ bản lần lượt tương tự như các lát cắt 521, các ô 523, và các CTU 527. Trên Fig.6, các lát cắt 621 được thể hiện bằng các đường nét đậm với nền trắng và các đường băm luân phiên để phân biệt về mặt đồ hoạ giữa các lát cắt 621. Các ô 623 được thể hiện bằng các đường nét đứt. Các đường biên ô 623 mà được đặt trên các đường biên lát cắt 621 là được thể hiện dưới dạng các đường nét đậm nét đứt, và các đường biên ô 623 mà không được đặt trên các đường biên lát cắt 621 là được thể hiện dưới dạng các đường nét đứt không đậm. Các đường biên CTU 627 được thể hiện dưới dạng các đường không đậm nét liền, ngoại trừ các vị trí mà ở đó các đường biên CTU 627 được bao phủ bởi các đường biên ô 623 hoặc các đường biên lát cắt 621. Theo ví dụ này, thì hình ảnh 600 bao gồm chín lát cắt 621, hai mươi tư ô 623, và hai trăm mười sáu CTU 627.

Như được thể hiện, lát cắt 621 là hình chữ nhật với các đường biên mà có thể được xác định bởi các ô 623 được bao gồm. Lát cắt 621 có thể không kéo dài qua toàn bộ chiều rộng của hình ảnh 600. Các ô 623 có thể được tạo ra ở các lát cắt 621 theo các hàng và các cột. Sau đó, các CTU 627 có thể được phân vùng từ các ô 623 để tạo ra các

phân vùng của hình ảnh 600 phù hợp để được chia tiếp thành các khối lập mã để lập mã theo kỹ thuật liên dự đoán và/hoặc nội dự đoán.

Nhờ sử dụng giải pháp nêu trên mà các hệ thống lập mã viđeo có thể được cải thiện. Ví dụ, các lát cắt được thiết kế sao cho các CTU mà được chứa trong lát cắt có thể không chỉ đơn giản là tập hợp CTU của ảnh theo thứ tự quét mành CTU của ảnh. Mà thay vào đó, lát cắt được xác định dưới dạng tập hợp CTU mà bao phủ vùng hình chữ nhật của ảnh. Ngoài ra, mỗi lát cắt nằm trong đơn vị NAL của riêng nó. Ngoài ra, các địa chỉ của các CTU mà được chứa trong lát cắt là có thể được báo hiệu bằng cách báo hiệu, trong mào đầu lát cắt, các địa chỉ CTU theo thứ tự quét mành của các CTU trên cùng bên trái và dưới cùng bên phải trong lát cắt. Ngoài ra, các lát cắt được thiết kế để chứa và chỉ chứa tập hợp ô hoàn chỉnh bao phủ vùng hình chữ nhật của ảnh. Mỗi lát cắt nằm trong đơn vị NAL của riêng nó. Theo cách này, thì mục đích của việc có nhiều hơn một lát cắt có thể là để đặt tập hợp ô bao phủ vùng hình chữ nhật của ảnh vào đơn vị NAL. Trong một số trường hợp, thì có một hoặc nhiều lát cắt trong ảnh, và mỗi trong số các lát cắt này có thể chứa tập hợp ô hoàn chỉnh bao phủ vùng hình chữ nhật. Cũng có thể có một lát cắt khác trong ảnh mà bao phủ phần còn lại của các ô của ảnh. Vùng được bao phủ bởi lát cắt này có thể là vùng hình chữ nhật có lỗ được bao phủ bởi các lát cắt khác. Ví dụ, đối với vùng cần quan tâm, thì ảnh có thể chứa hai lát cắt mà trong đó một lát cắt chứa tập hợp ô hoàn chỉnh bao phủ vùng cần quan tâm này và lát cắt kia chứa các ô còn lại của ảnh.

Các địa chỉ của các CTU mà được chứa trong lát cắt là có thể được báo hiệu một cách tường minh hoặc không tường minh bởi các ID ô của các ô được chứa trong lát cắt đó. Để báo hiệu hiệu quả, thì chỉ có các ID ô của các ô trên cùng bên trái và dưới cùng bên phải là có thể được báo hiệu theo một số ví dụ. Để cải thiện hơn nữa hiệu quả báo hiệu, thì cờ chỉ thị xem lát cắt có chứa ô đơn hay không là có thể được báo hiệu, và nếu có, thì chỉ một ID ô là có thể được báo hiệu. Trong các trường hợp khác, thì tất cả các ID ô được chứa trong lát cắt là đều được báo hiệu. Theo một số ví dụ, thì các giá trị ID ô là được ấn định để giống với chỉ số ô trong ảnh. Chiều dài, tính bằng bit, để báo hiệu tường minh các ID ô trong mào đầu lát cắt để trích xuất các địa chỉ của các CTU được chứa trong lát cắt, là có thể được trích xuất theo số lượng ô trong ảnh (ví dụ, ô của loga cơ số hai của số lượng ô trong ảnh). Số lượng ô trong ảnh có thể được báo hiệu tường minh trong tập hợp thông số hoặc được trích xuất trên mỗi cấu hình ô được báo

hiệu trong tập hợp thông số. Theo một số ví dụ, thì chiều dài, tính bằng bit, để báo hiệu tường minh các ID ô trong mào đầu lát cắt để trích xuất các địa chỉ của các CTU được chứa trong lát cắt, là có thể được báo hiệu trong tập hợp thông số. Theo một số ví dụ, thì số lượng điểm vào, mà bằng số lượng ô trong lát cắt trừ đi 1, là được trích xuất chứ không được báo hiệu trong mào đầu lát cắt. Theo ví dụ khác, thì việc báo hiệu cờ cho mỗi CTU để chỉ thị xem CTU đó có phải là kết thúc của lát cắt hay không là được tránh.

Theo một số ví dụ, thì các lát cắt và các ô là được thiết kế sao cho việc viết lại các mào đầu lát cắt là không cần thiết khi trích xuất tập hợp ô, chẳng hạn các tập hợp ô bị ràng buộc chuyển động (Motion Constrained Tile Set - MCTS), từ luồng bit để tạo ra luồng bit con hợp cách. Ví dụ, ID ô có thể được báo hiệu tường minh cho mỗi ô trong tập hợp thông số mà cấu hình ô được báo hiệu trong đó. Mỗi trong số các ID ô là duy nhất trong ảnh. Các ID ô có thể không liên tục trong ảnh. Tuy nhiên, các ID ô cần được tổ chức theo thứ tự tăng dần (ví dụ, tăng một cách đơn điệu) theo chiều quét mành ô của ảnh. Với việc này, thì thứ tự giải mã các lát cắt trong ảnh có thể được hạn chế ở giá trị tăng của ID ô của ô trên cùng bên trái. Khi ID ô không được báo hiệu tường minh và được suy ra là giống như chỉ số ô, thì phương án sau đây có thể được sử dụng để báo hiệu các giá trị ID ô trong các mào đầu lát cắt. Cờ chỉ thị xem lát cắt có phải là lát cắt đầu tiên của ảnh hay không là có thể được báo hiệu. Khi cờ đó chỉ thị rằng lát cắt đó là lát cắt đầu tiên của ảnh, thì việc báo hiệu ID ô của ô trên cùng bên trái của lát cắt đó là có thể được lược bỏ, vì nó có thể được suy ra là ô có chỉ số ô thấp nhất (ví dụ, chỉ số ô bằng không – giả sử chỉ số ô bắt đầu từ không).

Theo ví dụ khác, thì ảnh có thể không chứa, hoặc chứa một hoặc nhiều MCTS. MCTS có thể chứa một hoặc nhiều ô. Khi ô trong lát cắt là một phần của MCTS, thì MCTS đó bị ràng buộc sao cho tất cả các ô trong lát cắt đó đều là một phần của cùng MCTS đó. Lát cắt đó có thể còn bị ràng buộc sao cho cấu hình ô đối với tất cả các ảnh chứa các ô của MCTS là giống nhau về các vị trí và các kích thước của các ô trong MCTS đó. Theo một số ví dụ, thì lát cắt đó bị ràng buộc sao cho MCTS được chứa chuyên biệt trong lát cắt. Việc này có hai hệ quả. Trong trường hợp này, thì mỗi MCTS nằm trong một đơn vị NAL riêng biệt. Ngoài ra, mỗi MCTS là có hình chữ nhật.

Việc báo hiệu các MCTS có thể là như sau. Cờ có thể được báo hiệu trong mào đầu lát cắt để chỉ thị xem lát cắt đó có chứa các đơn vị NAL với MCTS trong đơn vị truy cập có chứa lát cắt tương ứng hay không. Các thông tin hỗ trợ khác dành cho

MCTS (ví dụ, thông tin về biên dạng, tầng và cấp độ của luồng bit con thu được từ việc trích xuất MCTS) là được báo hiệu trong thông điệp SEI. Theo cách khác, cả chỉ thị cờ và thông tin hỗ trợ của MCTS đều có thể được báo hiệu trong thông điệp SEI. Để cho phép báo hiệu việc coi các đường biên MCTS như các đường biên ảnh, thì phần tử cú pháp chỉ thị xem tất cả các đường biên ô có được coi là giống như các đường biên ảnh hay không là được báo hiệu, ví dụ, trong tập hợp thông số mà cấu hình ô được báo hiệu trong đó. Ngoài ra, phần tử cú pháp mà chỉ thị xem tất cả các đường biên lát cắt của lát cắt có được coi là giống như các đường biên ảnh hay không là có thể được báo hiệu trong mào đầu lát cắt, ví dụ, khi các phần tử cú pháp khác không chỉ thị rằng tất cả các đường biên ô được coi là giống như các đường biên ảnh.

Phần tử cú pháp mà chỉ thị xem các hoạt động lọc trong vòng lặp có thể được áp dụng giữa mỗi đường biên ô hay không là có thể được báo hiệu chỉ khi cú pháp không chỉ thị rằng tất cả các đường biên ô được coi là giống như các đường biên ảnh. Trong trường hợp này, thì việc coi đường biên ô như đường biên ảnh chỉ thị rằng, trong số các khía cạnh khác, thì không có hoạt động lọc trong vòng lặp nào có thể được áp dụng giữa mỗi đường biên ô. Theo các ví dụ khác, thì phần tử cú pháp mà chỉ thị xem các hoạt động lọc trong vòng lặp có thể được áp dụng giữa mỗi đường biên ô hay không là được báo hiệu độc lập với các chỉ thị xem tất cả các đường biên ô có được coi là giống như các đường biên ảnh hay không. Trong trường hợp này, thì việc coi đường biên ô như đường biên ảnh chỉ thị rằng các hoạt động lọc trong vòng lặp vẫn có thể được áp dụng giữa mỗi đường biên ô.

Theo một số ví dụ, thì đường biên MCTS được coi như đường biên ảnh. Ngoài ra, phần tử cú pháp trong mào đầu lát cắt mà chỉ thị xem đường biên lát cắt có được coi là giống như đường biên ảnh hay không cũng có thể được làm thành điều kiện đối với cờ mà chỉ thị xem lát cắt có chứa MCTS hay không. Trong một số trường hợp, thì giá trị của cờ mà chỉ thị rằng đường biên MCTS là cần được coi như đường biên ảnh là có thể được suy ra khi cờ trong mào đầu lát cắt chỉ thị lát cắt chứa MCTS.

Khi các đường biên của ô hoặc lát cắt được chỉ thị là cần được coi như các đường biên ảnh, thì các điều kiện sau đây áp dụng. Trong tiến trình trích xuất đối với việc dự đoán vectơ chuyển động của độ chói theo thời gian, thì các vị trí đường biên ảnh bên phải và đáy mà được sử dụng trong tiến trình này, mà lần lượt được chỉ thị bởi pic_height_in_luma_samples − 1 và pic_width_in_luma_samples − 1, là lần lượt được

thay thế bằng các vị trí đường biên bên phải và đáy của ô hoặc lát cắt, với các đơn vị là các mẫu độ chói. Trong tiến trình nội suy mẫu độ chói, thì các vị trí đường biên ảnh trái, phải, đỉnh, và đáy được sử dụng trong the tiến trình này, mà lần lượt được chỉ thị bởi 0, pic_height_in_luma_samples − 1, 0, pic_width_in_luma_samples − 1, là lần lượt được thay thế bằng các vị trí đường biên trái, phải, đỉnh, và đáy của ô hoặc lát cắt, với các đơn vị là các mẫu độ chói. Trong tiến trình nội suy mẫu sắc độ, thì các vị trí đường biên ảnh trái, phải, đỉnh, và đáy được sử dụng trong tiến trình này, mà lần lượt được chỉ thị bởi 0, pic_height_in_luma_samples / SubWidthC − 1, 0, pic_width_in_luma_samples / SubWidthC − 1, là lần lượt được thay thế bằng các vị trí đường biên trái, phải, đỉnh, và đáy của ô hoặc lát cắt, với các đơn vị là các mẫu sắc độ.

Các cơ chế nêu trên có thể được thực hiện như sau. Lát cắt được xác định là một số lượng nguyên các ô mà bao phủ vùng hình chữ nhật của ảnh và được chứa chuyên biệt trong đơn vị NAL đơn. Mào đầu lát cắt được xác định là một phần của lát cắt được lập mã chứa các phần tử dữ liệu liên quan đến tất cả các ô được biểu diễn trong lát cắt. Ô được xác định là vùng hình chữ nhật gồm các CTU trong cột ô cụ thể và hàng ô cụ thể trong ảnh. Cột ô được xác định là vùng hình chữ nhật gồm các CTU và có chiều cao bằng chiều cao của ảnh và chiều rộng được quy định bởi các phần tử cú pháp trong tập hợp thông số ảnh. Hàng ô được xác định là vùng hình chữ nhật gồm các CTU và có chiều cao được quy định bởi các phần tử cú pháp trong tập hợp thông số ảnh và chiều rộng bằng chiều rộng của ảnh. Việc quét ô được xác định là việc xếp thứ tự tuần tự cụ thể đối với các CTU phân vùng ảnh mà trong đó các CTU được xếp thứ tự liên tiếp trong quá trình quét mành CTU trong ô, còn các ô trong ảnh thì được xếp thứ tự liên tiếp trong quá trình quét mành của các ô của ảnh.

Phần này quy định việc ảnh được phân vùng thành các lát cắt và các ô như thế nào. Các ảnh được chia thành các lát cắt và các ô. Lát cắt là chuỗi ô mà bao phủ vùng hình chữ nhật của ảnh. Ô là chuỗi CTU mà bao phủ vùng hình chữ nhật của ảnh.

Khi ảnh được lập mã nhờ sử dụng ba mặt phẳng màu riêng biệt (separate_colour_plane_flag là bằng 1), thì lát cắt chỉ chứa các CTU thuộc một thành phần màu mà được nhận dạng bởi giá trị tương ứng của colour_plane_id, và mỗi mảng thành phần màu của ảnh là bao gồm các lát cắt có cùng giá trị colour_plane_id. Các lát

cắt được lập mã có các giá trị khác nhau của colour_plane_id trong ảnh là có thể được đan xen với nhau với sự ràng buộc là đối với mỗi giá trị của colour_plane_id, thì các đơn vị NAL của lát cắt được lập mã có giá trị đó của colour_plane_id sẽ có thứ tự là địa chỉ CTU tăng theo thứ tự quét ô đối với CTU đầu tiên của mỗi đơn vị NAL của lát cắt được lập mã. Cần lưu ý rằng khi separate_colour_plane_flag là bằng 0, thì mỗi CTU của ảnh là được chứa trong chính xác là một lát cắt. Khi separate_colour_plane_flag là bằng 1, thì mỗi CTU của thành phần màu là được chứa trong chính xác là một lát cắt (ví dụ, thông tin đối với mỗi CTU của ảnh là có mặt trong chính xác là ba lát cắt, và ba lát cắt này có các giá trị khác nhau của colour_plane_id).

Những cách chia sau đây đối với các phần tử xử lý của bản mô tả này tạo thành việc phân vùng trong không gian hoặc theo thành phần: chia mỗi ảnh thành các thành phần; chia mỗi thành phần thành các CTB; chia mỗi ảnh thành các cột ô; chia mỗi ảnh thành các hàng ô; chia mỗi cột ô thành các ô; chia mỗi hàng ô thành các ô; chia mỗi ô thành các CTU; chia mỗi ảnh thành các lát cắt; chia mỗi lát cắt thành các ô; chia mỗi lát cắt thành các CTU; chia mỗi CTU thành các CTB; chia mỗi CTB thành các khối lập mã, ngoại trừ việc các CTB là chưa hoàn chỉnh tại đường biên thành phần bên phải khi chiều rộng thành phần không phải là bội số nguyên của kích thước CTB và các CTB là chưa hoàn chỉnh tại đường biên thành phần đáy khi chiều cao thành phần không phải là bội số nguyên của kích thước CTB; chia mỗi CTU thành các đơn vị lập mã, ngoại trừ việc các CTU là chưa hoàn chỉnh tại đường biên ảnh bên phải khi chiều rộng ảnh trong các mẫu độ chói không phải là bội số nguyên của kích thước CTB độ chói và các CTU là chưa hoàn chỉnh tại đường biên ảnh đáy khi chiều cao ảnh trong các mẫu độ chói không phải là bội số nguyên của kích thước CTB độ chói; chia mỗi đơn vị lập mã thành các đơn vị biến đổi; chia mỗi đơn vị lập mã thành các khối lập mã; chia mỗi khối lập mã thành các khối biến đổi; và chia mỗi đơn vị biến đổi thành các khối biến đổi.

Các đầu vào của tiến trình trích xuất đối với sự khả dụng của khối lân cận là vị trí độ chói (xCurr, yCurr) của mẫu trên cùng bên trái của khối hiện tại so với mẫu độ chói trên cùng bên trái của ảnh hiện tại, và vị trí độ chói (xNbY, yNbY) được bao phủ bởi khối lân cận so với mẫu độ chói trên cùng bên trái của ảnh hiện tại. Các đầu ra của tiến trình này là sự khả dụng của khối lân cận mà bao phủ vị trí (xNbY, yNbY), được biểu thị là availableN. Sự khả dụng của khối lân cận availableN là được trích xuất như sau. Nếu một hoặc nhiều trong số các điều kiện sau đây là đúng, thì availableN

được đặt bằng sai. top_left_tile_id của lát cắt chứa khối lân cận là khác về giá trị so với top_left_tile_id của lát cắt chứa khối hiện tại hoặc khối lân cận được chứa trong ô khác so với khối hiện tại.

Tiến trình quét mành và ô CTB là như sau. Danh sách ColWidth[i] đối với i nằm trong khoảng từ 0 đến num_tile_columns_minus1, không loại trừ, quy định chiều rộng của cột ô thứ i với các đơn vị là các CTB, là được trích xuất như sau.

```
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 )
else {
        ColWidth[ num_tile_columns_minus1 ] = PicWidthInCtbsY   (6-1)
        for( i = 0; i < num_tile_columns_minus1; i++ ) {
                ColWidth[ i ] = tile_column_width_minus1[ i ] + 1
                ColWidth[ num_tile_columns_minus1 ]  −= ColWidth[ i ]
        }
}
```

Danh sách RowHeight[j] đối với j nằm trong khoảng từ 0 đến num_tile_rows_minus1, không loại trừ, quy định chiều cao của hàng ô thứ j với các đơn vị là các CTB, là được trích xuất như sau.

```
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 )
else {
        RowHeight[ num_tile_rows_minus1 ] = PicHeightInCtbsY
        (6-2)
        for( j = 0; j < num_tile_rows_minus1; j++ ) {
```

```
            RowHeight[ j ] = tile_row_height_minus1[ j ] + 1
            RowHeight[ num_tile_rows_minus1 ] −= RowHeight[ j ]
        }
}
```

Danh sách ColBd[i] đối với i nằm trong khoảng từ 0 đến num_tile_columns_minus1 + 1, không loại trừ, quy định vị trí của đường biên cột ô thứ i với các đơn vị là các CTB, là được trích xuất như sau:

```
for( ColBd[ 0 ] = 0, i = 0; i <= num_tile_columns_minus1; i++ )
            ColBd[ i + 1 ] = ColBd[ i ] + ColWidth[ i ]                    (6-3)
```

Danh sách RowBd[j] đối với j nằm trong khoảng từ 0 đến num_tile_rows_minus1 + 1, không loại trừ, quy định vị trí của đường biên hàng ô thứ j với các đơn vị là các CTB, là được trích xuất như sau:

```
for( RowBd[ 0 ] = 0, j = 0; j <= num_tile_rows_minus1; j++ )
        RowBd[ j + 1 ] = RowBd[ j ] + RowHeight[ j ]                       (6-4)
```

Danh sách CtbAddrRsToTs[ctbAddrRs] đối với ctbAddrRs nằm trong khoảng từ 0 đến PicSizeInCtbsY − 1, không loại trừ, quy định sự chuyển đổi từ địa chỉ CTB trong việc quét mành CTB của ảnh sang địa chỉ CTB trong việc quét ô, là được trích xuất như sau:

```
for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ ) {
        tbX = ctbAddrRs % PicWidthInCtbsY
        tbY = ctbAddrRs / PicWidthInCtbsY
        for( i = 0; i <= num_tile_columns_minus1; i++ )
            if( tbX >= ColBd[ i ] )
                tileX = i
        for( j = 0; j <= num_tile_rows_minus1; j++ )                       (6-5)
            if( tbY >= RowBd[ j ] )
                tileY = j
        CtbAddrRsToTs[ ctbAddrRs ] = 0
        for( i = 0; i < tileX; i++ )
            CtbAddrRsToTs[ ctbAddrRs ] += RowHeight[ tileY ] * ColWidth[ i ]
        for( j = 0; j < tileY; j++ )
            CtbAddrRsToTs[ ctbAddrRs ] += PicWidthInCtbsY * RowHeight[ j ]
```

CtbAddrRsToTs[ctbAddrRs] += (tbY − RowBd[tileY]
) * ColWidth[tileX] + tbX − ColBd[tileX]
}

Danh sách CtbAddrTsToRs[ctbAddrTs] đối với ctbAddrTs nằm trong khoảng từ 0 đến PicSizeInCtbsY − 1, không loại trừ, quy định sự chuyển đổi từ địa chỉ CTB trong việc quét ô sang địa chỉ CTB trong việc quét mành CTB của ảnh, là được trích xuất như sau:

```
for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ )          (6-6)
    CtbAddrTsToRs[ CtbAddrRsToTs[ ctbAddrRs ] ] = ctbAddrRs
```

Danh sách TileId[ctbAddrTs] đối với ctbAddrTs nằm trong khoảng từ 0 đến PicSizeInCtbsY − 1, không loại trừ, quy định sự chuyển đổi từ địa chỉ CTB trong việc quét ô sang ID ô, là được trích xuất như sau:

```
for( j = 0, tileIdx = 0; j <= num_tile_rows_minus1; j++ )
    for( i = 0; i <= num_tile_columns_minus1; i++, tileIdx++ )
        for( y = RowBd[ j ]; y < RowBd[ j + 1 ]; y++ )                  (6-7)
            for( x = ColBd[ i ]; x < ColBd[ i + 1 ]; x++ )
                TileId[ CtbAddrRsToTs[ y * PicWidthInCtbsY+ x ] ] =
                    explicit_tile_id_flag ? tile_id_val[ i ][ j ] :
tileIdx
```

Danh sách NumCtusInTile[tileIdx] đối với tileIdx nằm trong khoảng từ 0 đến PicSizeInCtbsY − 1, không loại trừ, quy định sự chuyển đổi từ chỉ số ô sang số lượng CTU trong ô, là được trích xuất như sau:

```
for( j = 0, tileIdx = 0; j <= num_tile_rows_minus1; j++ )
    for( i = 0; i <= num_tile_columns_minus1; i++, tileIdx++ )          (6-8)
        NumCtusInTile[ tileIdx ] = ColWidth[ i ] * RowHeight[ j ]
```

Tập hợp TileIdToIdx[tileId] đối với tập hợp giá trị NumTilesInPic tileId quy định sự chuyển đổi từ ID ô sang chỉ số ô và danh sách FirstCtbAddrTs[tileIdx] đối với tileIdx nằm trong khoảng từ 0 đến NumTilesInPic − 1, không loại trừ, quy định sự chuyển đổi từ ID ô sang địa chỉ CTB trong việc quét ô đối với CTB đầu tiên trong ô là được trích xuất như sau:

```
for( ctbAddrTs = 0, tileIdx = 0, tileStartFlag = 1; ctbAddrTs < PicSizeInCtbsY;
ctbAddrTs++ ) {
```

```
        if( tileStartFlag ) {
            TileIdToIdx[ TileId[ ctbAddrTs ] ] = tileIdx
            FirstCtbAddrTs[ tileIdx ] = ctbAddrTs                    (6-9)
            tileStartFlag = 0
        }
        tileEndFlag = ctbAddrTs == PicSizeInCtbsY − 1 || TileId[ ctbAddrTs + 1 ]
!= TileId[ ctbAddrTs ]
        if( tileEndFlag ) {
            tileIdx++
            tileStartFlag = 1
        }
}
```

Các giá trị của ColumnWidthInLumaSamples[i], quy định chiều rộng của cột ô thứ i với các đơn vị là các mẫu độ chói, là được đặt bằng ColWidth[i] << CtbLog2SizeY đối với i nằm trong khoảng từ 0 đến num_tile_columns_minus1, không loại trừ. Các giá trị của RowHeightInLumaSamples[j], quy định chiều cao của hàng ô thứ j với các đơn vị là các mẫu độ chói, là được đặt bằng RowHeight[j] << CtbLog2SizeY đối với j nằm trong khoảng từ 0 đến num_tile_rows_minus1, không loại trừ.

Cú pháp tập hợp thông số ảnh RBSP là như sau:

pic_parameter_set_rbsp( ) {	Bộ mô tả
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
transform_skip_enabled_flag	u(1)
single_tile_in_pic_flag	u(1)
if(!single_tile_in_pic_flag) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
}	
tile_id_len_minus1	ue(v)
explicit_tile_id_flag	u(1)
if(explicit_tile_id_flag)	

for(i = 0; i <= num_tile_columns_minus1; i++)	
for(j = 0; j <= num_tile_rows_minus1; j++)	
tile_id_val[i][j]	u(v)
if(!single_tile_in_pic_flag) {	
uniform_tile_spacing_flag	u(1)
if(!uniform_tile_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
tile_column_width_minus1[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
tile_row_height_minus1[i]	ue(v)
}	
tile_boundary_treated_as_pic_boundary_flag	u(1)
if(!tile_boundary_treated_as_pic_boundary_flag)	
loop_filter_across_tiles_enabled_flag	u(1)
}	
rbsp_trailing_bits( )	
}	

Bảng 1

Cú pháp mào đầu lát cắt được thay đổi như sau.

slice_header( ) {	Bộ mô tả
slice_pic_parameter_set_id	ue(v)
single_tile_in_slice_flag // Same note as below	u(1)
top_left_tile_id // Lưu ý rằng yếu tố này là cần thiết ngay cả khi chỉ có một ô trong ảnh để cho phép trích xuất một ô bị ràng buộc chuyển động để là luồng bit hợp cách mà không cần thay đổi mào đầu lát cắt.	u(v)
if(!single_tile_in_slice_flag)	
bottom_right_tile_id	u(v)
slice_type	ue(v)
if (slice_type != I)	
log2_diff_ctu_max_bt_size	ue(v)

dep_quant_enabled_flag	u(1)
if(!dep_quant_enabled_flag)	
sign_data_hiding_enabled_flag	u(1)
if(!tile_boundary_treated_as_pic_boundary_flag)	
slice_boundary_treated_as_pic_boundary_flag	
if(!single_tile_in_slice_flag) {	
offset_len_minus1	ue(v)
for(i = 0; i < NumTilesInSlice − 1; i++)	
entry_point_offset_minus1[i]	u(v)
}	
byte_alignment( )	
}	

Bảng 2

Cú pháp slice_data( ) là như sau:

slice_data( ) {	Bộ mô tả
tileIdx = TileIdToIdx[top_left_tile_id]	
for(j = 0; j < NumTileRowsInSlice; j++, tileIdx += num_tile_columns_minus1 + 1) {	
for(i = 0, CurrTileIdx = tileIdx; i < NumTileColumnsInSlice; i++, CurrTileIdx++) {	
ctbAddrInTs = FirstCtbAddrTs[CurrTileIdx]	
for(k = 0; k < NumCtusInTile[CurrTileIdx]; k++, ctbAddrInTs++) {	
CtbAddrInRs = CtbAddrTsToRs[ctbAddrInTs]	
coding_tree_unit( )	
}	
end_of_tile_one_bit /* equal to 1 */	ae(v)
if(i < NumTileRowsInSlice − 1 || j < NumTileColumnsInSlice − 1)	
byte_alignment( )	

}	
}	
}	

Bảng 3

Ngữ nghĩa tập hợp thông số ảnh RBSP là như sau. single_tile_in_pic_flag được đặt bằng một để quy định rằng chỉ có một ô trong mỗi ảnh tham chiếu đến PPS. single_tile_in_pic_flag được đặt bằng không để quy định rằng có nhiều hơn một ô trong mỗi ảnh tham chiếu đến PPS. Sự hợp cách luồng bit có thể yêu cầu rằng giá trị của single_tile_in_pic_flag là giống nhau đối với tất cả các PPS mà được kích hoạt trong chuỗi viđeo được lập mã (Coded Video Sequence - CVS). num_tile_columns_minus1 + 1 quy định số lượng cột ô phân vùng ảnh. num_tile_columns_minus1 sẽ nằm trong khoảng từ không đến PicWidthInCtbsY − 1, không loại trừ. Khi không hiện hữu, thì giá trị của num_tile_columns_minus1 được suy ra là bằng không. num_tile_rows_minus1 + 1 quy định số lượng hàng ô phân vùng ảnh. num_tile_rows_minus1 sẽ nằm trong khoảng từ không đến PicHeightInCtbsY − 1, không loại trừ. Khi không hiện hữu, thì giá trị của num_tile_rows_minus1 được suy ra là bằng không. Biến số NumTilesInPic được đặt bằng (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1).

Khi single_tile_in_pic_flag là bằng không, thì NumTilesInPic sẽ lớn hơn không. tile_id_len_minus1 + 1 quy định số lượng bit được dùng để biểu thị phần tử cú pháp tile_id_val[i][j], khi có mặt, trong PPS, và các phần tử cú pháp top_left_tile_id và bottom_right_tile_id, khi có mặt, trong các mào đầu lát cắt tham chiếu đến PPS. Giá trị của tile_id_len_minus1 sẽ nằm trong khoảng từ Ceil(Log2(NumTilesInPic) đến 15, không loại trừ. explicit_tile_id_flag được đặt bằng một để quy định rằng ID ô dành cho mỗi ô là được báo hiệu tường minh. explicit_tile_id_flag được đặt bằng không để quy định rằng các ID ô không được báo hiệu tường minh. tile_id_val[i][j] quy định ID ô của ô của cột ô thứ i và hàng ô thứ j. Chiều dài của tile_id_val[i][j] là tile_id_len_minus1 + 1 bit.

Đối với số nguyên m bất kỳ nằm trong khoảng từ không đến num_tile_columns_minus1, không loại trừ, và số nguyên n bất kỳ nằm trong khoảng từ không đến num_tile_rows_minus1, không loại trừ, thì tile_id_val[i][j] sẽ không bằng tile_id_val[m][n] khi i không bằng m hoặc j không bằng n, và tile_id_val[i][j] sẽ nhỏ hơn tile_id_val[m][n] khi j * (num_tile_columns_minus1 + 1) + i nhỏ hơn

n * (num_tile_columns_minus1 + 1) + m. uniform_tile_spacing_flag được đặt bằng một để quy định rằng các đường biên cột ô, và tương tự như vậy là các đường biên hàng ô, là được phân bố đồng nhất trên ảnh. uniform_tile_spacing_flag được đặt bằng không để quy định rằng các đường biên cột ô, và tương tự như vậy là các đường biên hàng ô, là không được phân bố đồng nhất trên ảnh nhưng được báo hiệu tường minh nhờ sử dụng các phần tử cú pháp tile_column_width_minus1[i] và tile_row_height_minus1[i]. Khi không có mặt, thì giá trị của uniform_tile_spacing_flag được suy ra là bằng 1. tile_column_width_minus1[i] + 1 quy định chiều rộng của cột ô thứ i với các đơn vị là các CTB. tile_row_height_minus1[i] + 1 quy định chiều cao của hàng ô thứ i với các đơn vị là các CTB.

Các biến số sau đây được trích xuất bằng cách gọi ra tiến trình chuyển đổi quét mành và ô CTB: danh sách ColWidth[i] đối với i nằm trong khoảng từ 0 đến num_tile_columns_minus1, không loại trừ, quy định chiều rộng của cột ô thứ i với các đơn vị là các CTB; danh sách RowHeight[j] đối với j nằm trong khoảng từ 0 đến num_tile_rows_minus1, không loại trừ, quy định chiều cao của hàng ô thứ j với các đơn vị là các CTB; danh sách ColBd[i] đối với i nằm trong khoảng từ 0 đến num_tile_columns_minus1 + 1, không loại trừ, quy định vị trí của đường biên cột ô thứ i với các đơn vị là các CTB; danh sách RowBd[j] đối với j nằm trong khoảng từ 0 đến num_tile_rows_minus1 + 1, không loại trừ, quy định vị trí của đường biên hàng ô thứ j với các đơn vị là các CTB; danh sách CtbAddrRsToTs[ctbAddrRs] đối với ctbAddrRs nằm trong khoảng từ 0 đến PicSizeInCtbsY − 1, không loại trừ, quy định sự chuyển đổi từ địa chỉ CTB trong việc quét mành CTB của ảnh sang địa chỉ CTB trong việc quét ô;

danh sách CtbAddrTsToRs[ctbAddrTs] đối với ctbAddrTs nằm trong khoảng từ 0 đến PicSizeInCtbsY − 1, không loại trừ, quy định sự chuyển đổi từ địa chỉ CTB trong việc quét ô sang địa chỉ CTB trong việc quét mành CTB của ảnh; danh sách TileId[ctbAddrTs] đối với ctbAddrTs nằm trong khoảng từ 0 đến PicSizeInCtbsY − 1, không loại trừ, quy định sự chuyển đổi từ địa chỉ CTB trong việc quét ô sang ID ô; danh sách NumCtusInTile[tileIdx] đối với tileIdx nằm trong khoảng từ 0 đến PicSizeInCtbsY − 1, không loại trừ, quy định sự chuyển đổi từ chỉ số ô sang số lượng CTU trong ô; tập hợp TileIdToIdx[tileId] đối với tập hợp giá trị NumTilesInPic tileId quy định sự chuyển đổi từ ID ô sang chỉ số ô và danh sách FirstCtbAddrTs[tileIdx] đối

với tileIdx nằm trong khoảng từ 0 đến NumTilesInPic − 1, không loại trừ, quy định sự chuyển đổi từ ID ô sang địa chỉ CTB trong việc quét ô đối với CTB đầu tiên trong ô; các danh sách ColumnWidthInLumaSamples[i] đối với i nằm trong khoảng từ 0 đến num_tile_columns_minus1, không loại trừ, quy định chiều rộng của cột ô thứ i với các đơn vị là các mẫu độ chói; và danh sách RowHeightInLumaSamples[j] đối với j nằm trong khoảng từ 0 đến num_tile_rows_minus1, không loại trừ, quy định chiều cao của hàng ô thứ j với các đơn vị là các mẫu độ chói.

Các giá trị của ColumnWidthInLumaSamples[i] đối với i nằm trong khoảng từ 0 đến num_tile_columns_minus1, không loại trừ, và RowHeightInLumaSamples[j] đối với j nằm trong khoảng từ 0 đến num_tile_rows_minus1, không loại trừ, sẽ đều lớn hơn 0. tile_boundary_treated_as_picture_boundary_flag được đặt bằng một để quy định rằng mỗi đường biên ô là được coi như đường biên ảnh trong tiến trình giải mã đối với các ảnh tham chiếu đến PPS. Tile_boundary_treated_as_picture_boundary_flag được đặt bằng không để quy định rằng mỗi đường biên ô là có thể hoặc không được coi như đường biên ảnh trong tiến trình giải mã đối với các ảnh tham chiếu đến PPS. Khi không có mặt, thì giá trị của tile_boundary_treated_as_picture_boundary_flag được suy ra là bằng một. loop_filter_across_tiles_enabled_flag được đặt bằng một để quy định rằng các hoạt động lọc trong vòng lặp là có thể được thực hiện giữa các đường biên ô trong các ảnh tham chiếu đến PPS. Loop_filter_across_tiles_enabled_flag được đặt bằng không để quy định rằng các hoạt động lọc trong vòng lặp là không được thực hiện giữa các đường biên ô trong các ảnh tham chiếu đến PPS. Các hoạt động lọc trong vòng lặp bao gồm các hoạt động của bộ lọc khử khối, bộ lọc dịch thích ứng mẫu, và bộ lọc vòng thích ứng. Khi không có mặt, thì giá trị của loop_filter_across_tiles_enabled_flag được suy ra là bằng không.

Ngữ nghĩa mào đầu lát cắt là như sau. Khi có mặt, thì giá trị của phần tử cú pháp mào đầu lát cắt slice_pic_parameter_set_id sẽ là giống nhau trong tất cả các mào đầu lát cắt của ảnh được lập mã. slice_pic_parameter_set_id quy định giá trị của pps_pic_parameter_set_id đối với PPS đang được sử dụng. Giá trị của slice_pic_parameter_set_id sẽ nằm trong khoảng từ 0 đến 63, không loại trừ. single_tile_in_slice_flag được đặt bằng một để quy định rằng chỉ có một ô trong lát cắt. single_picture_in_pic_flag được đặt bằng không để quy định rằng có nhiều hơn một ô trong lát cắt. top_left_tile_id quy định ID ô của ô được đặt tại góc trên cùng bên trái của

lát cắt. Chiều dài của top_left_tile_id là tile_id_len_minus1 + 1 bit. Giá trị của top_left_tile_id sẽ không bằng giá trị của top_left_tile_id của đơn vị NAL của lát cắt được lập mã bất kỳ khác của cùng ảnh được lập mã. Khi có nhiều hơn một lát cắt trong ảnh, thì thứ tự giải mã của các lát cắt đó trong ảnh sẽ là theo giá trị tăng của top_left_tile_id. bottom_right_tile_id quy định ID ô của ô được đặt tại góc dưới cùng bên phải của lát cắt. Chiều dài của bottom_right_tile_id là tile_id_len_minus1 + 1 bit. Khi không có mặt, thì giá trị của bottom_right_tile_id được suy ra là bằng top_left_tile_id.

Các biến số NumTileRowsInSlice, NumTileColumnsInSlice, và NumTilesInSlice là được trích xuất như sau:

deltaTileIdx = TileIdToIdx[bottom_right_tile_id] − TileIdToIdx[top_left_tile_id]

NumTileRowsInSlice = (deltaTileIdx / (num_tile_columns_minus1 + 1)) + 1 (7-25)

NumTileColumnsInSlice = (deltaTileIdx % (num_tile_columns_minus1 + 1)) + 1

NumTilesInSlice = NumTileRowsInSlice * NumTileColumnsInSlice

slice_type quy định loại lập mã của lát cắt theo bảng 4.

slice_type	Tên gọi của slice_type
0	B (lát cắt B)
1	P (lát cắt P)
2	I (lát cắt I)

Bảng 4

Khi nal_unit_type có giá trị nằm trong khoảng TBD, không loại trừ, ví dụ, ảnh là ảnh IRAP (Intra Random Access Picture - ảnh nội truy cập ngẫu nhiên), thì slice_type sẽ bằng hai. log2_diff_ctu_max_bt_size quy định sự chênh lệch giữa kích thước CTB độ chói và kích thước độ chói lớn nhất (chiều rộng hoặc chiều cao) của khối lập mã mà có thể được chia nhờ sử dụng việc chia nhị phân. Giá trị của log2_diff_ctu_max_bt_size sẽ nằm trong khoảng từ không đến CtbLog2SizeY − MinCbLog2SizeY, không loại trừ. Khi log2_diff_ctu_max_bt_size không có mặt, thì giá trị của log2_diff_ctu_max_bt_size được suy ra là bằng hai.

Các biến số MinQtLog2SizeY, MaxBtLog2SizeY, MinBtLog2SizeY, MaxTtLog2SizeY, MinTtLog2SizeY, MaxBtSizeY, MinBtSizeY, MaxTtSizeY, MinTtSizeY và MaxMttDepth là được trích xuất như sau:

MinQtLog2SizeY = (slice_type == I) ? MinQtLog2SizeIntraY : MinQtLog2SizeInterY (7-26)

MaxBtLog2SizeY = CtbLog2SizeY − log2_diff_ctu_max_bt_size (7-27)

MinBtLog2SizeY = MinCbLog2SizeY (7-28)

MaxTtLog2SizeY = (slice_type == I) ? 5 : 6 (7-29)

MinTtLog2SizeY = MinCbLog2SizeY (7-30)

MinQtSizeY = 1 << MinQtLog2SizeY (7-31)

MaxBtSizeY = 1 << MaxBtLog2SizeY (7-32)

MinBtSizeY = 1 << MinBtLog2SizeY (7-33)

MaxTtSizeY = 1 << MaxTtLog2SizeY (7-34)

MinTtSizeY = 1 << MinTtLog2SizeY (7-35)

MaxMttDepth = (slice_type == I) ? max_mtt_hierarchy_depth_intra_slices : max_mtt_hierarchy_depth_inter_slices (7-36)

dep_quant_enabled_flag được đặt bằng không để quy định rằng việc lượng tử hoá phụ thuộc là bị vô hiệu hoá. Dep_quant_enabled_flag được đặt bằng một để quy định rằng việc lượng tử hoá phụ thuộc là được cho phép. sign_data_hiding_enabled_flag được đặt bằng không để quy định rằng việc ẩn bit dấu là bị vô hiệu hoá. Sign_data_hiding_enabled_flag được đặt bằng một để quy định rằng việc ẩn bit dấu là được cho phép. Khi sign_data_hiding_enabled_flag không có mặt, thì nó được suy ra là bằng không. slice_boundary_treated_as_pic_boundary_flag được đặt bằng một để quy định rằng mỗi đường biên lát cắt của lát cắt là được coi là giống như đường biên ảnh trong tiến trình giải mã. Việc slice_boundary_treated_as_pic_boundary_flag bằng không là quy định rằng mỗi đường biên ô có thể hoặc không được coi là giống như đường biên ảnh trong tiến trình giải mã. Khi không có mặt, thì giá trị của slice_boundary_treated_as_pic_boundary_flag được suy ra là bằng một. offset_len_minus1 + 1 quy định chiều dài, tính bằng bit, của các phần tử cú pháp entry_point_offset_minus1[i]. Giá trị của offset_len_minus1 sẽ nằm trong khoảng từ 0 đến 31, không loại trừ. entry_point_offset_minus1[i] + 1 quy định độ dịch điểm vào thứ i tính bằng byte, và được biểu diễn bởi offset_len_minus1 + 1 bit. Dữ liệu lát cắt

mà đi theo sau mào đầu lát cắt là bao gồm các tập hợp con NumTilesInSlice, với các giá trị chỉ số tập hợp con nằm trong khoảng từ 0 đến NumTilesInSlice − 1, không loại trừ. Byte đầu tiên của dữ liệu lát cắt được coi là byte không. Khi có mặt, thì các byte chống giả lập mà xuất hiện ở phần dữ liệu lát cắt của đơn vị NAL của lát cắt được lập mã là được tính như một phần của dữ liệu lát cắt nhằm mục đích nhận dạng tập hợp con.

Tập hợp con không là bao gồm các byte từ không đến entry_point_offset_minus1[0], không loại trừ, của dữ liệu phân đoạn lát cắt được lập mã, tập hợp con k, với k nằm trong khoảng từ 1 đến NumTilesInSlice − 2, không loại trừ, là bao gồm các byte firstByte[k] đến lastByte[k], không loại trừ, của dữ liệu lát cắt được lập mã với firstByte[k] và lastByte[k] được xác định là:

$$firstByte[k] = \sum_{n=1}^{k} (entry\_point\_offset\_minus1[n-1]+1) \quad (7\text{-}37)$$

lastByte[k] = firstByte[k] + entry_point_offset_minus1[k] (7-38)

Tập hợp con cuối cùng (với chỉ số tập hợp con bằng NumTilesInSlice − 1) bao gồm các byte còn lại của dữ liệu lát cắt được lập mã. Mỗi tập hợp con sẽ bao gồm tất cả các bit được lập mã của tất cả các CTU trong lát cắt mà nằm trong cùng ô.

Ngữ nghĩa chung của dữ liệu lát cắt là như sau: end_of_tile_one_bit sẽ bằng một. Đối với mỗi ô, thì các biến số LeftBoundaryPos, TopBoundaryPos, RightBoundaryPos và BotBoundaryPos là được trích xuất như sau. Nếu tile_boundary_treated_as_pic_boundary_flag là đúng, thì các điều kiện sau đây áp dụng:

tileColIdx = CurrTileIdx / (num_tile_columns_minus1 + 1) (7-39)

tileRowIdx = CurrTileIdx % (num_tile_columns_minus1 + 1) (7-40)

LeftBoundaryPos = ColBd[tileColIdx] << CtbLog2SizeY (7-41)

RightBoundaryPos = ((ColBd[tileColIdx] + ColWidth[tileColIdx]) << CtbLog2SizeY) − 1 (7-42)

TopBoundaryPos = RowBd[tileRowIdx] << CtbLog2SizeY (7-43)

BotBoundaryPos = ((RowBd[tileRowIdx] + RowHeight[tileRowIdx]) << CtbLog2SizeY) − 1 (7-44)

Ngược lại, nếu slice_boundary_treated_as_pic_boundary_flag là đúng, thì các điều kiện sau đây áp dụng:

sliceFirstTileColIdx = TileIdToIdx[top_left_tile_id] / (num_tile_columns_minus1 + 1) (7-45)

sliceFirstTileRowIdx = TileIdToIdx[top_left_tile_id] % (num_tile_columns_minus1 + 1) (7-46)

sliceLastTileColIdx = TileIdToIdx[bottom_right_tile_id] / (num_tile_columns_minus1 + 1) (7-47)

sliceLastTileRowIdx = TileIdToIdx[bottom_right_tile_id] % (num_tile_columns_minus1 + 1) (7-48)

LeftBoundaryPos = ColBd[sliceFirstTileColIdx] << CtbLog2SizeY (7-49)

RightBoundaryPos = ((ColBd[sliceLastTileColIdx] + ColWidth[sliceLastTileColIdx]) << CtbLog2SizeY) − 1 (7-50)

TopBoundaryPos = RowBd[sliceFirstTileRowIdx] << CtbLog2SizeY (7-51)

BotBoundaryPos = ((RowBd[sliceLastTileRowIdx] + RowHeight[sliceLastTileRowIdx]) << CtbLog2SizeY) −1 (7-52)

Ngược lại, (slice_boundary_treated_as_pic_boundary_flag là sai), thì các điều kiện sau đây áp dụng:

LeftBoundaryPos = 0 (7-53)

RightBoundaryPos = pic_width_in_luma_samples − 1 (7-54)

TopBoundaryPos = 0 (7-55)

BotBoundaryPos = pic_height_in_luma_samples − 1 (7-56)

Tiến trình trích xuất để dự đoán vectơ chuyển động của độ chói theo thời gian là như sau. Nếu yCb >> CtbLog2SizeY bằng yColBr >> CtbLog2SizeY, yColBr nhỏ hơn pic_height_in_luma_samples, và xColBr nhỏ hơn pic_width_in_luma_samples, thì các điều kiện sau đây áp dụng:

Biến số colCb quy định khối lập mã độ chói bao phủ vị trí được cải biến mà được cho bởi ((xColBr >> 3) << 3, (yColBr >> 3) << 3) bên trong ảnh nằm cùng vị trí mà được quy định bởi ColPic. Vị trí độ chói (xColCb, yColCb) được đặt bằng mẫu trên cùng bên trái của khối lập mã độ chói cùng vị trí mà được quy định bởi colCb so với mẫu độ chói trên cùng bên trái của ảnh nằm cùng vị trí mà được quy định bởi ColPic. Tiến trình trích xuất dành cho các vectơ chuyển động nằm cùng vị trí là được gọi ra với currCb, colCb, (xColCb, yColCb), refIdxLX, và thông số kiểm soát

controlParaFlag được đặt bằng 0 làm các đầu vào, và đầu ra được ấn định cho mvLXCol và availableFlagLXCol. Nếu yCb >> CtbLog2SizeY bằng yColBr >> CtbLog2SizeY, yColBr nhỏ hơn hoặc bằng BotBoundaryPos và xColBr nhỏ hơn hoặc bằng RightBoundaryPos, thì các điều kiện sau đây được áp dụng. Biến số colCb quy định khối lập mã độ chói bao phủ vị trí được cải biến mà được cho bởi ((xColBr >> 3) << 3, (yColBr >> 3) << 3) bên trong ảnh nằm cùng vị trí mà được quy định bởi ColPic. Vị trí độ chói (xColCb, yColCb) được đặt bằng mẫu trên cùng bên trái của khối lập mã độ chói cùng vị trí mà được quy định bởi colCb so với mẫu độ chói trên cùng bên trái của ảnh nằm cùng vị trí mà được quy định bởi ColPic. Tiến trình trích xuất dành cho các vectơ chuyển động nằm cùng vị trí là được gọi ra với currCb, colCb, (xColCb, yColCb), refIdxLX, và thông số kiểm soát controlParaFlag được đặt bằng 0 làm các đầu vào, và đầu ra được ấn định cho mvLXCol và availableFlagLXCol.

Theo một số ví dụ, thì tiến trình trích xuất để dự đoán vectơ chuyển động của độ chói theo thời gian là như sau. Nếu yCb >> CtbLog2SizeY bằng yColBr >> CtbLog2SizeY, yColBr nhỏ hơn pic_height_in_luma_samples và xColBr nhỏ hơn pic_width_in_luma_samples, thì các điều kiện sau đây áp dụng. Biến số colCb quy định khối lập mã độ chói bao phủ vị trí được cải biến mà được cho bởi ((xColBr >> 3) << 3, (yColBr >> 3) << 3) bên trong ảnh nằm cùng vị trí mà được quy định bởi ColPic. Vị trí độ chói (xColCb, yColCb) được đặt bằng mẫu trên cùng bên trái của khối lập mã độ chói cùng vị trí mà được quy định bởi colCb so với mẫu độ chói trên cùng bên trái của ảnh nằm cùng vị trí mà được quy định bởi ColPic. Tiến trình trích xuất dành cho các vectơ chuyển động nằm cùng vị trí là được gọi ra với currCb, colCb, (xColCb, yColCb), refIdxLX, và thông số kiểm soát controlParaFlag được đặt bằng 0 làm các đầu vào, và đầu ra được ấn định cho mvLXCol và availableFlagLXCol. Nếu yCb >> CtbLog2SizeY bằng yColBr >> CtbLog2SizeY, thì điều kiện sau đây áp dụng. xColCtr = Min(xColCtr, RightBoundaryPos) và yColCtr = Min(yColCtr, BotBoundaryPos). Biến số colCb quy định khối lập mã độ chói bao phủ vị trí được cải biến mà được cho bởi ((xColBr >> 3) << 3, (yColBr >> 3) << 3) bên trong ảnh nằm cùng vị trí mà được quy định bởi ColPic. Vị trí độ chói (xColCb, yColCb) được đặt bằng mẫu trên cùng bên trái của khối lập mã độ chói cùng vị trí mà được quy định bởi colCb so với

mẫu độ chói trên cùng bên trái của ảnh nằm cùng vị trí mà được quy định bởi ColPic. Tiến trình trích xuất dành cho các vectơ chuyển động nằm cùng vị trí là được gọi ra với currCb, colCb, (xColCb, yColCb), refIdxLX, và thông số kiểm soát controlParaFlag được đặt bằng 0 làm các đầu vào, và đầu ra được ấn định cho mvLXCol và availableFlagLXCol.

Tiến trình nội suy mẫu độ chói là như sau. Các đầu vào của tiến trình này là: vị trí độ chói ở các đơn vị mẫu đầy đủ (xIntL, yIntL); vị trí độ chói ở các đơn vị phân đoạn mẫu (xFracL, yFracL); và mảng mẫu tham chiếu độ chói refPicLXL. Đầu ra của tiến trình này là giá trị mẫu độ chói dự đoán được predSampleLXL. Các biến số shift1, shift2, shift3 là được trích xuất như sau. Biến số shift1 được đặt bằng Min(4, BitDepthY − 8) và biến số shift2 được đặt bằng 6 và biến số shift3 được đặt bằng Max(2, 14 − BitDepthY). Giá trị mẫu độ chói dự đoán được predSampleLXL là được trích xuất như sau. Nếu cả xFracL và yFracL là bằng 0, thì giá trị của predSampleLXL là được trích xuất như sau:

predSampleLXL = refPicLXL[xIntL][yIntL] << shift3 (8-228)

Ngược lại, nếu xFracL không bằng 0 và yFracL bằng 0, thì giá trị của predSampleLXL là được trích xuất như sau:

predSampleLXL =
(fL[xFracL, 0] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPos, xIntL − 3)][yIntL] +
fL[xFracL][1] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPos, xIntL − 2)][yIntL] +
fL[xFracL][2] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPos, xIntL − 1)][yIntL] +
fL[xFracL][3] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPos, xIntL)] [yIntL] +
fL[xFracL][4] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPos, xIntL + 1)][yIntL] +
(8-228)
fL[xFracL][5] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPos, xIntL + 2)][yIntL] +
fL[xFracL][6] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPos, xIntL +

3)][yIntL] +

fL[xFracL][7] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPos, xIntL + 4)][yIntL]) >> shift1

Ngược lại, nếu xFracL bằng 0 và yFracL không bằng 0, thì giá trị của predSampleLXL là được trích xuất như sau:

predSampleLXL =

(fL[yFracL, 0] * refPicLXL[xIntL][Clip3(TopBoundaryPos, BotBoundaryPos, yIntL − 3)] +

fL[yFracL][1] * refPicLXL[xIntL][Clip3(TopBoundaryPos, BotBoundaryPos, yIntL − 2)] +

fL[yFracL][2] * refPicLXL[xIntL][Clip3(TopBoundaryPos, BotBoundaryPos, yIntL − 1)] +

fL[yFracL][3] * refPicLXL[xIntL][Clip3(TopBoundaryPos, BotBoundaryPos, yIntL)] +

fL[yFracL][4] * refPicLXL[xIntL][Clip3(TopBoundaryPos, BotBoundaryPos, yIntL + 1)] + (8-228)

fL[yFracL][5] * refPicLXL[xIntL][Clip3(TopBoundaryPos, BotBoundaryPos, yIntL + 2)] +

fL[yFracL][6] * refPicLXL[xIntL][Clip3(TopBoundaryPos, BotBoundaryPos, yIntL + 3)] +

fL[yFracL][7] * refPicLXL[xIntL][Clip3(TopBoundaryPos, BotBoundaryPos, yIntL + 4)]) >> shift1

Còn nếu xFracL không bằng 0 và yFracL không bằng 0, thì giá trị của predSampleLXL là được trích xuất như sau. Mảng mẫu temp[n], với n = 0..7, là được trích xuất như sau:

yPosL = Clip3(TopBoundaryPos, BotBoundaryPos, yIntL + n − 3) (8-228)

temp[n] =

(fL[xFracL, 0] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPos, xIntL − 3)][yPosL] +

fL[xFracL][1] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPos, xIntL − 2)][yPosL] +

fL[xFracL][2] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPo

s, xIntL − 1)][yPosL] +

fL[xFracL][3] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPo

s, xIntL)][yPosL] +

fL[xFracL][4] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPo

s, xIntL + 1)][yPosL] + (8-228)

fL[xFracL][5] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPo

s, xIntL + 2)][yPosL] +

fL[xFracL][6] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPo

s, xIntL + 3)][yPosL] +

fL[xFracL][7] * refPicLXL[Clip3(LeftBoundaryPos, RightBoundaryPo

s, xIntL + 4)][yPosL]) >> shift1

Giá trị mẫu độ chói dự đoán được predSampleLXL là được trích xuất như sau:

predSampleLXL = (fL[yFracL][0] * temp[0] +
fL[yFracL][1] * temp[1] +
fL[yFracL][2] * temp[2] +
fL[yFracL][3] * temp[3] +
fL[yFracL][4] * temp[4] +

(8-228)

fL[yFracL][5] * temp[5] +
fL[yFracL][6] * temp[6] +

fL[yFracL][7] * temp[7]) >> shift2

Vị trí phân đoạn mẫu p	Các hệ số bộ lọc nội suy							
	fL[p][0]	fL[p][1]	fL[p][2]	fL[p][3]	fL[p][4]	fL[p][5]	fL[p][6]	fL[p][7]
1	0	1	−3	63	4	−2	1	0
2	−1	2	−5	62	8	−3	1	0
3	−1	3	−8	60	13	−4	1	0
4	−1	4	−10	58	17	−5	1	0
5	−1	4	−11	52	26	−8	3	−1

6	−1	3	−9	47	31	−10	4	−1
7	−1	4	−11	45	34	−10	4	−1
8	−1	4	−11	40	40	−11	4	−1
9	−1	4	−10	34	45	−11	4	−1
10	−1	4	−10	31	47	−9	3	−1
11	−1	3	−8	26	52	−11	4	−1
12	0	1	−5	17	58	−10	4	−1
13	0	1	−4	13	60	−8	3	−1
14	0	1	−3	8	62	−5	2	−1
15	0	1	−2	4	63	−3	1	0

Bảng 5

Tiến trình nội suy mẫu sắc độ là như sau. Các đầu vào của tiến trình này là: vị trí sắc độ ở các đơn vị mẫu đầy đủ (xIntC, yIntC); vị trí sắc độ ở các đơn vị phân đoạn mẫu (xFracC, yFracC) thứ tám; và mảng mẫu tham chiếu sắc độ refPicLXC. Đầu ra của tiến trình này là giá trị mẫu sắc độ dự đoán được predSampleLXC. Các biến số shift1, shift2, shift3, picWC, và picHC là được trích xuất như sau. Biến số shift1 được đặt bằng Min(4, BitDepthC − 8), biến số shift2 được đặt bằng 6, và biến số shift3 được đặt bằng Max(2, 14 − BitDepthC). Biến số lPos, rPos, tPos và bPos được thiết đặt như sau:

lPos = LeftBoundaryPos / SubWidthC (8-228)

rPos = (RightBoundaryPos + 1) / SubWidthC (8-228)

tPos = TopBoundaryPos / SubHeightC (8-228)

bPos = (BotBoundaryPos + 1) / SubHeightC (8-228)

Giá trị mẫu sắc độ dự đoán được predSampleLXC là được trích xuất như sau. Nếu cả xFracC và yFracC đều bằng 0, thì giá trị của predSampleLXC là được trích xuất như sau:

predSampleLXC = refPicLXC[xIntC][yIntC] << shift3 (8-228)

Ngược lại, nếu xFracC không bằng 0 và yFracC bằng 0, thì giá trị của predSampleLXC là được trích xuất như sau:

predSampleLXC =

(fC[xFracC][0] * refPicLXC[Clip3(lPos, rPos, xIntC − 1)][yIntC] +

fC[xFracC][1] * refPicLXC[Clip3(lPos, rPos, xIntC)][yIntC] +

fC[xFracC][2] * refPicLXC[Clip3(lPos, rPos, xIntC + 1)][yIntC] + (8-228)

fC[xFracC][3] * refPicLXC[Clip3(lPos, rPos, xIntC + 2)][yIntC]) > > shift1

Ngược lại, nếu xFracC bằng 0 và yFracC không bằng 0, thì giá trị của predSampleLXC là được trích xuất như sau:

redSampleLXC =

(fC[yFracC][0] * refPicLXC[xIntC][Clip3(tPos, bPos, yIntC − 1)] +

fC[yFracC][1] * refPicLXC[xIntC][Clip3(tPos, bPos, yIntC)] +

fC[yFracC][2] * refPicLXC[xIntC][Clip3(tPos, bPos, yIntC + 1)] + (8-228)

fC[yFracC][3] * refPicLXC[xIntC][Clip3(tPos, bPos, yIntC + 2)]) > > shift1

Còn nếu xFracC không bằng 0 và yFracC không bằng 0, thì giá trị của predSampleLXC là được trích xuất như sau. Mảng mẫu temp[n], với n = 0..3, là được trích xuất như sau:

yPosC = Clip3(tPos, bPos, yIntC + n − 1) (8-228)

temp[n] =

(fC[xFracC][0] * refPicLXC[Clip3(lPos, rPos, xIntC − 1)][yPosC] +

fC[xFracC][1] * refPicLXC[Clip3(lPos, rPos, xIntC)][yPosC] +

fC[xFracC][2] * refPicLXC[Clip3(lPos, rPos, xIntC + 1)][yPosC] + (8-228)

fC[xFracC][3] * refPicLXC[Clip3(lPos, rPos, xIntC + 2)][yPosC]) > > shift1

Giá trị mẫu sắc độ dự đoán được predSampleLXC là được trích xuất như sau:

predSampleLXC = (fC[yFracC][0] * temp[0] +
fC[yFracC][1] * temp[1] +
fC[yFracC][2] * temp[2] + (8-228)
fC[yFracC][3] * temp[3]) >> shift2

Vị trí phân đoạn mẫu p	Các hệ số bộ lọc nội suy			
	fC[p]] 0]	fC[p][1]	fC[p][2]	fC[p][3]
1	−1	63	2	0
2	−2	62	4	0
3	-2	60	7	−1
4	−2	58	10	−2
5	−3	57	12	−2
6	−4	56	14	−2
7	−4	55	15	−2
8	−4	54	16	−2
9	−5	53	18	−2
10	−6	52	20	−2
11	−6	49	24	−3
12	−6	46	28	−4
13	−5	44	29	−4
14	−4	42	30	−4
15	−4	39	33	−4
16	−4	36	36	−4
17	−4	33	39	−4
18	−4	30	42	−4
19	−4	29	44	−5
20	−4	28	46	−6
21	−3	24	49	−6
22	−2	20	52	−6
23	−2	18	53	−5
24	−2	16	54	−4

25	−2	15	55	−4
26	−2	14	56	−4
27	−2	12	57	−3
28	−2	10	58	−2
29	−1	7	60	−2
30	0	4	62	−2
31	0	2	63	−1

Bảng 6

Tiến trình phân tích lập mã số học nhị phân thích ứng dựa trên ngữ cảnh (Context-based Adaptive Binary Arithmetic Coding - CABAC) dành cho dữ liệu lát cắt là như sau. Tiến trình khởi tạo được gọi ra khi bắt đầu phân tích cú pháp CTU và CTU là CTU đầu tiên trong ô. Lưu ý rằng điểm bắt đầu của dữ liệu lát cắt cũng được bao trùm bởi nguyên lý này vì mỗi điểm bắt đầu của dữ liệu lát cắt là điểm bắt đầu của ô.

Theo ví dụ khác, thì tiến trình quét mành và ô CTB là như sau. Danh sách ColWidth[i] đối với i nằm trong khoảng từ 0 đến num_tile_columns_minus1, không loại trừ, quy định chiều rộng của cột ô thứ i với các đơn vị là các CTB, là được trích xuất như sau:

```
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 )
else {
        ColWidth[ num_tile_columns_minus1 ] = PicWidthInCtbsY        (6-1)
        for( i = 0; i < num_tile_columns_minus1; i++ ) {
            ColWidth[ i ] = tile_column_width_minus1[ i ] + 1
            ColWidth[ num_tile_columns_minus1 ]  −=  ColWidth[ i ]
        }
}
```

Danh sách RowHeight[j] đối với j nằm trong khoảng từ 0 đến num_tile_rows_minus1, không loại trừ, quy định chiều cao của hàng ô thứ j với các đơn vị là các CTB, là được trích xuất như sau:

```
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 )
else {
        RowHeight[ num_tile_rows_minus1 ] = PicHeightInCtbsY          (6-2)
        for( j = 0; j < num_tile_rows_minus1; j++ ) {
            RowHeight[ j ] = tile_row_height_minus1[ j ] + 1
            RowHeight[ num_tile_rows_minus1 ] −= RowHeight[ j ]
        }
}
```

Danh sách ColBd[i] đối với i nằm trong khoảng từ 0 đến num_tile_columns_minus1 + 1, không loại trừ, quy định vị trí của đường biên cột ô thứ i với các đơn vị là các CTB, là được trích xuất như sau:

```
for( ColBd[ 0 ] = 0, i = 0; i <= num_tile_columns_minus1; i++ )
        ColBd[ i + 1 ] = ColBd[ i ] + ColWidth[ i ]                    (6-3)
```

Danh sách RowBd[j] đối với j nằm trong khoảng từ 0 đến num_tile_rows_minus1 + 1, không loại trừ, quy định vị trí của đường biên hàng ô thứ j với các đơn vị là các CTB, là được trích xuất như sau:

```
for( RowBd[ 0 ] = 0, j = 0; j <= num_tile_rows_minus1; j++ )
        RowBd[ j + 1 ] = RowBd[ j ] + RowHeight[ j ]                   (6-4)
```

Danh sách CtbAddrRsToTs[ctbAddrRs] đối với ctbAddrRs nằm trong khoảng từ 0 đến PicSizeInCtbsY − 1, không loại trừ, quy định sự chuyển đổi từ địa chỉ CTB trong việc quét mành CTB của ảnh sang địa chỉ CTB trong việc quét ô, là được trích xuất như sau:

```
for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ ) {
        tbX = ctbAddrRs % PicWidthInCtbsY
```

```
tbY = ctbAddrRs / PicWidthInCtbsY
for( i = 0; i <= num_tile_columns_minus1; i++ )
    if( tbX >= ColBd[ i ] )
        tileX = i
for( j = 0; j <= num_tile_rows_minus1; j++ )                    (6-5)
    if( tbY >= RowBd[ j ] )
        tileY = j
CtbAddrRsToTs[ ctbAddrRs ] = 0
for( i = 0; i < tileX; i++ )
    CtbAddrRsToTs[ ctbAddrRs ] += RowHeight[ tileY ] * ColWidth[ i ]
for( j = 0; j < tileY; j++ )
    CtbAddrRsToTs[ ctbAddrRs ] += PicWidthInCtbsY * RowHeight[ j ]
CtbAddrRsToTs[ ctbAddrRs ] += ( tbY − RowBd[ tileY ]
) * ColWidth[ tileX ] + tbX − ColBd[ tileX ]
}
```

Danh sách CtbAddrTsToRs[ctbAddrTs] đối với ctbAddrTs nằm trong khoảng từ 0 đến PicSizeInCtbsY − 1, không loại trừ, quy định sự chuyển đổi từ địa chỉ CTB trong việc quét ô sang địa chỉ CTB trong việc quét mành CTB của ảnh, là được trích xuất như sau:

```
for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ )    (6-6)
    CtbAddrTsToRs[ CtbAddrRsToTs[ ctbAddrRs ] ] = ctbAddrRs
```

Danh sách TileId[ctbAddrTs] đối với ctbAddrTs nằm trong khoảng từ 0 đến PicSizeInCtbsY − 1, không loại trừ, quy định sự chuyển đổi từ địa chỉ CTB trong việc quét ô sang ID ô, là được trích xuất như sau:

```
for( j = 0, tileIdx = 0; j <= num_tile_rows_minus1; j++ )
    for( i = 0; i <= num_tile_columns_minus1; i++, tileIdx++ )
        for( y = RowBd[ j ]; y < RowBd[ j + 1 ]; y++ )          (6-7)
            for( x = ColBd[ i ]; x < ColBd[ i + 1 ]; x++ )
                TileId[ CtbAddrRsToTs[ y * PicWidthInCtbsY+ x ] ] =
                    explicit_tile_id_flag ? tile_id_val[ i ][ j ] :
tileIdx
```

Danh sách NumCtusInTile[tileIdx] đối với tileIdx nằm trong khoảng từ 0 đến PicSizeInCtbsY − 1, không loại trừ, quy định sự chuyển đổi từ chỉ số ô sang số lượng CTU trong ô, là được trích xuất như sau:

```
for( j = 0, tileIdx = 0; j <= num_tile_rows_minus1; j++ )
        for( i = 0; i <= num_tile_columns_minus1; i++, tileIdx++ )        (6-8)
            NumCtusInTile[ tileIdx ] = ColWidth[ i ] * RowHeight[ j ]
```

Tập hợp TileIdToIdx[tileId] đối với tập hợp giá trị NumTilesInPic tileId quy định sự chuyển đổi từ ID ô sang chỉ số ô và danh sách FirstCtbAddrTs[tileIdx] đối với tileIdx nằm trong khoảng từ 0 đến NumTilesInPic − 1, không loại trừ, quy định sự chuyển đổi từ ID ô sang địa chỉ CTB trong việc quét ô đối với CTB đầu tiên trong ô là được trích xuất như sau:

```
for( ctbAddrTs = 0, tileIdx = 0, tileStartFlag = 1; ctbAddrTs < PicSizeInCtbsY;
ctbAddrTs++ ) {
        if( tileStartFlag ) {
            TileIdToIdx[ TileId[ ctbAddrTs ] ] = tileIdx
            FirstCtbAddrTs[ tileIdx ] = ctbAddrTs              (6-9)
            tileStartFlag = 0
        }
        tileEndFlag = ctbAddrTs == PicSizeInCtbsY − 1 ||
TileId[ ctbAddrTs + 1 ] != TileId[ ctbAddrTs ]
        if( tileEndFlag ) {
            tileIdx++
            tileStartFlag = 1
        }
}
```

Các giá trị của ColumnWidthInLumaSamples[i], quy định chiều rộng của cột ô thứ i với các đơn vị là các mẫu độ chói, là được đặt bằng ColWidth[i] << CtbLog2SizeY đối với i nằm trong khoảng từ 0 đến num_tile_columns_minus1, không loại trừ. Các giá trị của RowHeightInLumaSamples[j], quy định chiều cao của hàng ô thứ j với các đơn vị là các mẫu độ chói, là được đặt bằng RowHeight[j] << CtbLog2SizeY đối với j nằm trong khoảng từ 0 đến num_tile_rows_minus1, không loại trừ.

Cú pháp tập hợp thông số ảnh RBSP là như sau.

pic_parameter_set_rbsp( ) {	Bộ mô tả
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
transform_skip_enabled_flag	u(1)
single_tile_in_pic_flag	u(1)
if(!single_tile_in_pic_flag) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
}	
~~tile_id_len_minus1~~	~~ue(v)~~
~~explicit_tile_id_flag~~	~~u(1)~~
~~if(explicit_tile_id_flag)~~	
~~for(i = 0; i <= num_tile_columns_minus1; i++)~~	
~~for(j = 0; j <= num_tile_rows_minus1; j++)~~	
~~tile_id_val[i][j]~~	~~u(v)~~
if(!single_tile_in_pic_flag) {	
uniform_tile_spacing_flag	u(1)
if(!uniform_tile_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
tile_column_width_minus1[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
tile_row_height_minus1[i]	ue(v)
}	
tile_boundary_treated_as_pic_boundary_flag	u(1)
if(!tile_boundary_treated_as_pic_boundary_flag)	
loop_filter_across_tiles_enabled_flag	u(1)
}	
rbsp_trailing_bits( )	
}	

Bảng 7

Cú pháp mào đầu lát cắt là như sau:

slice_header( ) {	Bộ mô tả
slice_pic_parameter_set_id	ue(v)
first_slice_in_pic_flag	u(1)
single_tile_in_slice_flag	u(1)
if(!first_slice_in_pic_flag)	
top_left_tile_id	u(v)
if(!single_tile_in_slice_flag)	
bottom_right_tile_id	u(v)
slice_type	ue(v)
if (slice_type != I)	
log2_diff_ctu_max_bt_size	ue(v)
dep_quant_enabled_flag	u(1)
if(!dep_quant_enabled_flag)	
sign_data_hiding_enabled_flag	u(1)
all_tiles_mcts_flag	u(1)
if(!tile_boundary_treated_as_pic_boundary_flag && !all_tiles_mcts_flag)	
slice_boundary_treated_as_pic_boundary_flag	
if(!single_tile_in_slice_flag) {	
offset_len_minus1	ue(v)
for(i = 0; i < NumTilesInSlice − 1; i++)	
entry_point_offset_minus1[i]	u(v)
}	
byte_alignment( )	
}	

Bảng 8

Ngữ nghĩa mào đầu lát cắt là như sau. Khi có mặt, thì giá trị của phần tử cú pháp mào đầu lát cắt slice_pic_parameter_set_id sẽ là giống nhau trong tất cả các mào

đầu lát cắt của ảnh được lập mã. slice_pic_parameter_set_id quy định giá trị của pps_pic_parameter_set_id đối với PPS đang được sử dụng. Giá trị của slice_pic_parameter_set_id sẽ nằm trong khoảng từ 0 đến 63, không loại trừ. first_slice_in_pic_flag được đặt bằng một để quy định rằng lát cắt là lát cắt đầu tiên của ảnh theo thứ tự giải mã. first_slice_in_pic_flag được đặt bằng không để quy định rằng lát cắt không phải là lát cắt đầu tiên của ảnh theo thứ tự giải mã. single_tile_in_slice_flag được đặt bằng một để quy định rằng chỉ có một ô trong lát cắt. Việc single_picture_in_pic_flag bằng 0 quy định rằng có nhiều hơn một ô trong lát cắt. top_left_tile_id quy định ID ô của ô được đặt tại góc trên cùng bên trái của lát cắt. Chiều dài của top_left_tile_id là Ceil(Log2((num_tile_rows_minus1 + 1) * (num_tile_columns_minus1 + 1))) bit. Giá trị của top_left_tile_id sẽ không bằng giá trị của top_left_tile_id của đơn vị NAL của lát cắt được lập mã bất kỳ khác của cùng ảnh được lập mã. Khi không có mặt, thì giá trị của top_left_tile_id được suy ra là bằng không. Khi có nhiều hơn một lát cắt trong ảnh, thì thứ tự giải mã của các lát cắt đó trong ảnh sẽ là theo giá trị tăng của top_left_tile_id. bottom_right_tile_id quy định ID ô của ô được đặt tại góc dưới cùng bên phải của lát cắt. Chiều dài của bottom_right_tile_id là Ceil(Log2((num_tile_rows_minus1 + 1) * (num_tile_columns_minus1 + 1))) bit. Khi không có mặt, thì giá trị của bottom_right_tile_id được suy ra là bằng top_left_tile_id. Các biến số NumTileRowsInSlice, NumTileColumnsInSlice, và NumTilesInSlice là được trích xuất như sau:

deltaTileIdx = TileIdToIdx[bottom_right_tile_id] − TileIdToIdx[top_left_tile_id]
NumTileRowsInSlice = (deltaTileIdx / (num_tile_columns_minus1 + 1)) + 1 (7-25)
NumTileColumnsInSlice = (deltaTileIdx % (num_tile_columns_minus1 + 1)) + 1
NumTilesInSlice = NumTileRowsInSlice * NumTileColumnsInSlice

all_tiles_mcts_flag được đặt bằng một để quy định rằng tất cả các ô trong lát cắt là một phần của MCTS, mà chỉ chứa các ô trong lát cắt đối với đơn vị truy cập hiện tại, và các đường biên MCTS (mà nằm cùng chỗ với các đường biên lát cắt của lát cắt) là được coi là giống như các đường biên ảnh. all_tiles_mcts_flag được đặt bằng không để quy định rằng điều kiện trên đây, như được quy định bởi việc all_tiles_mcts_flag

bằng 1, là có thể hoặc không áp dụng. Slice_boundary_treated_as_pic_boundary_flag được đặt bằng một để quy định rằng mỗi đường biên lát cắt của lát cắt là được coi là giống như đường biên ảnh trong tiến trình giải mã. slice_boudnary_treated_as_pic_boundary_flag được đặt bằng không để quy định rằng mỗi đường biên ô là có thể hoặc không được coi là giống như đường biên ảnh trong tiến trình giải mã. Khi không có mặt, thì giá trị của slice_boundary_treated_as_pic_boundary_flag được suy ra là bằng một. Mỗi tập hợp con sẽ bao gồm tất cả các bit được lập mã của tất cả các CTU trong lát cắt mà nằm trong cùng ô.

Fig.7 là hình vẽ thể hiện sơ đồ thiết bị lập mã viđeo 700 ví dụ. Thiết bị lập mã viđeo 700 này phù hợp để thực hiện các ví dụ/các phương án được bộc lộ như được mô tả ở đây. Thiết bị lập mã viđeo 700 này bao gồm các cổng xuôi dòng 720, các cổng ngược dòng 750, và/hoặc các đơn vị thu phát (Tx/Rx) 710, bao gồm các bộ phát và/hoặc các bộ thu để truyền thông dữ liệu ngược dòng và/hoặc xuôi dòng qua mạng. Thiết bị lập mã viđeo 700 còn bao gồm bộ xử lý 730 bao gồm đơn vị logic và/hoặc đơn vị xử lý trung tâm (Central Processing Unit - CPU) để xử lý dữ liệu và bộ nhớ 732 để lưu giữ dữ liệu. Thiết bị lập mã viđeo 700 cũng có thể bao gồm thành phần điện, thành phần quang sang điện (Optical-to-Electrical - OE), thành phần điện sang quang (Electrical-to-Optical - EO), và/hoặc các thành phần truyền thông không dây được ghép nối đến các cổng ngược dòng 750 và/hoặc các cổng xuôi dòng 720 để truyền thông dữ liệu qua mạng điện, mạng quang, hoặc các mạng truyền thông không dây. Thiết bị lập mã viđeo 700 cũng có thể bao gồm các thiết bị đầu vào và/hoặc đầu ra (Input/Output - I/O) 760 để truyền thông dữ liệu qua lại với người dùng. Các thiết bị I/O 760 có thể bao gồm các thiết bị đầu ra, chẳng hạn thiết bị hiển thị để hiển thị dữ liệu viđeo, loa để xuất ra dữ liệu auđio, v.v.. Các thiết bị I/O 760 cũng có thể bao gồm các thiết bị đầu vào, chẳng hạn bàn phím, chuột, bi xoay, v.v., và/hoặc các giao diện tương ứng để tương tác với các thiết bị đầu ra đó.

Bộ xử lý 730 được thực hiện bằng phần cứng và phần mềm. Bộ xử lý 730 có thể được thực hiện dưới dạng một hoặc nhiều chip, lõi CPU (ví dụ, dưới dạng bộ xử lý đa lõi), mảng cổng lập trình được dạng trường (Field-Programmable Gate Array - FPGA), mạch tích hợp chuyên dụng (Application Specific Integrated Circuit - ASIC), và bộ xử lý tín hiệu số (Digital Signal Processor - DSP). Bộ xử lý 730 có giao tiếp với

các cổng xuôi dòng 720, Tx/Rx 710, các cổng ngược dòng 750, và bộ nhớ 732. Bộ xử lý 730 bao gồm môđun lập mã 714. Môđun lập mã 714 thực hiện các phương án được bộc lộ được mô tả trên đây, chẳng hạn các phương pháp 100, 800, và 900, mà có thể sử dụng luồng bit 500 và/hoặc hình ảnh 600. Môđun lập mã 714 cũng có thể thực hiện phương pháp/cơ chế bất kỳ khác được mô tả ở đây. Ngoài ra, môđun lập mã 714 có thể thực hiện hệ thống codec 200, bộ mã hoá 300, và/hoặc bộ giải mã 400. Ví dụ, môđun lập mã 714 có thể phân vùng hình ảnh thành các lát cắt, các lát cắt thành các ô, ô thành các CTU, các CTU thành các khối, và mã hoá các khối đó khi hoạt động như bộ mã hoá. Ngoài ra, môđun lập mã 714 có thể báo hiệu một cách không tường minh số lượng ô trong lát cắt, số lượng độ dịch điểm vào dành cho các ô trong lát cắt, các địa chỉ CTU của các ô, CTU cuối cùng trong lát cắt và/hoặc dữ liệu khác bằng cách báo hiệu các đường biên của lát cắt dựa trên các ID ô của ô đầu tiên và ô cuối cùng trong lát cắt. Khi hoạt động như bộ giải mã, thì môđun lập mã 714 có thể tái tạo dữ liệu đó dựa trên ô đầu tiên và ô cuối cùng trong lát cắt, và nhờ đó làm tăng hiệu quả lập mã. Do đó, môđun lập mã 714 làm cho thiết bị lập mã viđeo 700 đem lại chức năng bổ sung và/hoặc hiệu quả lập mã khi phân vùng và lập mã dữ liệu viđeo. Như vậy, môđun lập mã 714 cải thiện chức năng của thiết bị lập mã viđeo 700 cũng như khắc phục các vấn đề riêng của lĩnh vực lập mã viđeo. Ngoài ra, môđun lập mã 714 thực hiện việc biến đổi thiết bị lập mã viđeo 700 sang trạng thái khác. Theo cách khác, môđun lập mã 714 có thể thực hiện dưới dạng các chỉ dẫn được lưu giữ trong bộ nhớ 732 và được thực thi bởi bộ xử lý 730 (ví dụ, dưới dạng sản phẩm chương trình máy tính được lưu giữ trên phương tiện phi nhất thời).

Bộ nhớ 732 bao gồm một hoặc nhiều loại bộ nhớ chẳng hạn như cơ cấu đĩa, ổ băng, ổ cứng thể rắn, bộ nhớ chỉ đọc (Read Only Memory - ROM), bộ nhớ truy cập ngẫu nhiên (Random Access Memory - RAM), bộ nhớ flash (bộ nhớ chớp nhoáng), bộ nhớ đánh địa chỉ được nội dung bậc ba (Ternary Content-Addressable Memory - TCAM), bộ nhớ truy cập ngẫu nhiên tĩnh (Static Random-Access Memory - SRAM), v.v.. Bộ nhớ 732 có thể được sử dụng làm thiết bị lưu trữ dữ liệu tràn, để lưu giữ các chương trình khi các chương trình đó được chọn để thực thi, và để lưu giữ các chỉ dẫn và dữ liệu mà được đọc trong quá trình thực thi chương trình.

Fig.8 là hình vẽ thể hiện lưu đồ của phương pháp 800 ví dụ để mã hoá hình ảnh, chẳng hạn hình ảnh 600, vào luồng bit, chẳng hạn luồng bit 500. Phương pháp 800

có thể được sử dụng bởi bộ mã hoá, chẳng hạn hệ thống codec 200, bộ mã hoá 300, và/hoặc thiết bị lập mã viđeo 700 khi thực hiện phương pháp 100.

Phương pháp 800 có thể bắt đầu khi bộ mã hoá nhận được chuỗi viđeo bao gồm nhiều hình ảnh và xác định là mã hoá chuỗi viđeo đó vào luồng bit, ví dụ, dựa trên đầu vào người dùng. Chuỗi viđeo này được phân vùng thành các ảnh/các hình ảnh/các khung để phân vùng tiếp trước khi mã hoá. Ở bước 801, hình ảnh được phân vùng thành các lát cắt. Các lát cắt này được phân vùng thành các ô. Các ô này được phân vùng thành các CTU. Các CTU này tiếp tục được phân vùng thành các khối lập mã.

Ở bước 803, mỗi lát cắt được mã hoá trong đơn vị VCL NAL riêng biệt trong luồng bit. Các lát cắt này chứa (các) ô, các CTU, và các khối lập mã. Như vậy, các ô, các CTU, và các khối lập mã của mỗi lát cắt cũng được mã hoá vào đơn vị VCL NAL tương ứng. Luồng bit này bao gồm các đơn vị VCL NAL mà bao gồm các lát cắt từ các hình ảnh khác nhau trong chuỗi viđeo. Như đã mô tả trên đây, lát cắt bao gồm số lượng nguyên các ô. Như vậy, mỗi đơn vị VCL NAL trong số các đơn vị VCL NAL này chứa một lát cắt đơn của dữ liệu ảnh được chia thành một số lượng nguyên các ô.

Lưu ý rằng các bước 805 và 807 có thể xuất hiện trước, sau, và/hoặc cùng bước 803. Các bước 805 và 807 được coi là riêng biệt để cho việc mô tả được rõ ràng. Ở bước 805, các ID ô được ấn định cho các ô đối với mỗi lát cắt. Ngoài ra, các địa chỉ được ấn định cho các CTU. Ví dụ, các địa chỉ của các CTU trong đơn vị VCL NAL có thể được ấn định dựa trên các ID ô tương ứng với các ô mà chứa các CTU đó.

Ở bước 807, thì giá trị chỉ thị số lượng ô trong lát cắt tương ứng, và theo đó là trong mỗi đơn vị VCL NAL, là được mã hoá trong luồng bit. Ví dụ, số lượng ô trong đơn vị VCL NAL có thể được bao gồm trong đơn vị không phải VCL NAL mà chứa PPS hoặc mào đầu lát cắt mà tương ứng với lát cắt đó. Theo ví dụ khác, thì số lượng ô có thể được lược bỏ khỏi luồng bit khi bộ giải mã có thể xác định số lượng ô trong đơn vị VCL NAL dựa trên các ô trên cùng bên trái và dưới cùng bên phải trong lát cắt tương ứng (mà có thể được báo hiệu trong đơn vị không phải VCL NAL chứa mào đầu lát cắt). Các độ dịch điểm vào dành cho các ô cũng được mã hoá trong luồng bit. Ví dụ, các độ dịch điểm vào dành cho các ô có thể được mã hoá trong mào đầu lát cắt được liên kết với lát cắt. Mỗi trong số các độ dịch điểm vào này chỉ thị vị trí bắt đầu của ô tương ứng trong đơn vị VCL NAL. Số lượng độ dịch điểm vào không được báo hiệu tường minh trong luồng bit bởi vì số lượng độ dịch điểm vào có thể được xác định là

giá trị của số lượng ô trừ đi một. Theo một số ví dụ, thì các địa chỉ của các CTU trong đơn vị VCL NAL cũng có thể được báo hiệu tường minh trong mào đầu lát cắt. Theo các ví dụ khác, thì các địa chỉ của các CTU trong đơn vị VCL NAL là được lược bỏ khỏi luồng bit để hỗ trợ việc xác định các địa chỉ của các CTU tại bộ giải mã dựa trên ID ô của ô đầu tiên được chứa trong đơn vị VCL NAL. Như đã mô tả trên đây, các CTU mà được mã hoá ở bước 803 là có thể không chứa cờ chỉ thị CTU cuối cùng trong đơn vị VCL NAL. Điều này là vì bộ giải mã có thể xác định số lượng CTU trong đơn vị VCL NAL dựa trên việc biết số lượng ô trong lát cắt. Tuy nhiên, đơn vị VCL NAL này có thể được mã hoá để chứa bit đệm ngay sau CTU cuối cùng trong mỗi ô để hỗ trợ việc duy trì sự phân cách giữa dữ liệu ảnh được liên kết với mỗi ô.

Ở bước 809, luồng bit này được truyền mà không có số lượng độ dịch điểm vào để hỗ trợ việc giải mã các ô theo sự suy luận rằng số lượng độ dịch điểm vào dành cho các ô là số lượng ô trong đơn vị VCL NAL trừ đi một.

Fig.9 là hình vẽ thể hiện lưu đồ của phương pháp 900 ví dụ để giải mã hình ảnh, chẳng hạn hình ảnh 600, từ luồng bit, chẳng hạn luồng bit 500. Phương pháp 900 có thể được sử dụng bởi bộ giải mã, chẳng hạn hệ thống codec 200, bộ giải mã 400, và/hoặc thiết bị lập mã viđeo 700 khi thực hiện phương pháp 100.

Phương pháp 900 có thể bắt đầu khi bộ giải mã bắt đầu nhận luồng bit của dữ liệu được lập mã mà biểu diễn chuỗi viđeo, ví dụ, dưới dạng kết quả của phương pháp 800. Ở bước 901, luồng bit được nhận tại bộ giải mã. Luồng bit này bao gồm các đơn vị VCL NAL, mỗi trong số chúng chứa dữ liệu ảnh được chia thành các ô. Như đã nêu trên, mỗi lát cắt của hình ảnh là bao gồm một số lượng nguyên các ô. Ngoài ra, mỗi lát cắt được bao gồm trong đơn vị VCL NAL riêng biệt. Theo đó, mỗi đơn vị VCL NAL bao gồm một số lượng nguyên các ô. Ngoài ra, mỗi trong số các ô này được chia thành các CTU, mà tiếp tục được chia thành các khối lập mã.

Ở bước 903, bộ giải mã xác định số lượng ô trong đơn vị VCL NAL. Ví dụ, có thể thu được số lượng ô trong đơn vị VCL NAL từ đơn vị không phải VCL NAL mà chứa mào đầu lát cắt tương ứng. Theo ví dụ khác, thì mào đầu lát cắt có thể chứa dữ liệu chỉ thị ô phía trên bên trái và ô dưới cùng bên phải trong lát cắt. Dữ liệu này có thể được dùng để xác định số lượng ô trong lát cắt, và theo đó là trong đơn vị VCL NAL. Sau đó, bộ giải mã có thể sử dụng số lượng ô trong lát cắt để xác định số lượng độ dịch điểm vào dành cho các ô. Ví dụ, số lượng độ dịch điểm vào dành cho các ô là số lượng

ô trong đơn vị VCL NAL trừ đi một. Như đã mô tả trên đây, mỗi trong số các độ dịch điểm vào này chỉ thị vị trí bắt đầu của ô tương ứng trong đơn vị VCL NAL. Số lượng độ dịch điểm vào có thể không được báo hiệu tường minh trong luồng bit để tăng khả năng nén và theo đó là hiệu quả lập mã.

Ở bước 905, bộ giải mã thu thập các độ dịch điểm vào dành cho các ô từ luồng bit dựa trên số lượng độ dịch điểm vào. Ví dụ, các độ dịch điểm vào dành cho các ô là có thể được thu thập từ mào đầu lát cắt được liên kết với lát cắt.

Như đã nêu trên, dữ liệu ảnh là được lập mã dưới dạng các CTU được chứa trong mỗi trong số các ô. Ngoài ra, các địa chỉ của các CTU trong đơn vị VCL NAL có thể được ấn định dựa trên các ID ô tương ứng với các ô. Theo đó, ở bước 907, thì bộ giải mã có thể xác định các ID ô dành cho các ô và các địa chỉ dành cho các CTU dựa trên các ID ô này. Theo một số ví dụ, thì các ID ô là được báo hiệu tường minh, ví dụ, trong PPS và/hoặc mào đầu lát cắt. Theo các ví dụ khác, thì bộ giải mã có thể xác định các ID ô dựa trên các ID ô của ô phía trên bên trái và ô dưới cùng bên phải của lát cắt. Ngoài ra, trong một số trường hợp thì các địa chỉ của các CTU trong đơn vị VCL NAL là được báo hiệu tường minh trong mào đầu lát cắt. Theo các ví dụ khác, thì bộ giải mã có thể xác định các địa chỉ của các CTU trong đơn vị VCL NAL dựa trên ID ô của ô đầu tiên được chứa trong đơn vị VCL NAL (ví dụ, ô trên cùng bên trái).

Ở bước 909, bộ giải mã giải mã các ô tại các độ dịch điểm vào để tạo ra hình ảnh được tái tạo. Ví dụ, bộ giải mã có thể sử dụng các độ dịch điểm vào của các ô và các địa chỉ CTU, để giải mã các CTU, trong đó các địa chỉ CTU là được báo hiệu tường minh trong mào đầu lát cắt hoặc được suy ra dựa trên ID ô. Sau đó, bộ giải mã có thể chuyển tiếp hình ảnh được tái tạo này về phía thiết bị hiển thị như một phần của chuỗi viđeo tái tạo được. Theo một số ví dụ, thì bộ giải mã có thể không tìm kiếm cờ chỉ thị CTU cuối cùng trong đơn vị VCL NAL vì cờ đó có thể được lược bỏ khỏi đơn vị VCL NAL. Trong trường hợp đó, thì bộ giải mã có thể xác định số lượng CTU trong đơn vị VCL NAL dựa trên việc biết số lượng ô trong lát cắt. Theo một số ví dụ, thì bộ giải mã có thể phân tách các ô trong đơn vị VCL NAL dựa trên bit đệm ngay sau CTU cuối cùng trong mỗi ô.

Fig.10 là hình vẽ thể hiện sơ đồ của hệ thống 1000 ví dụ để lập mã chuỗi viđeo của các hình ảnh, chẳng hạn hình ảnh 600, trong luồng bit, chẳng hạn luồng bit 500. Hệ thống 1000 có thể được thực hiện bởi bộ mã hoá và bộ giải mã, chẳng hạn hệ

thống codec 200, bộ mã hoá 300, bộ giải mã 400, và/hoặc thiết bị lập mã viđeo 700. Ngoài ra, hệ thống 1000 có thể được sử dụng khi thực hiện phương pháp 100, 800, và/hoặc 900.

Hệ thống 1000 bao gồm bộ mã hoá viđeo 1002. Bộ mã hoá viđeo 1002 bao gồm môđun phân vùng 1001 để phân vùng hình ảnh thành các ô. Bộ mã hoá viđeo 1002 này còn bao gồm môđun mã hoá 1003 để mã hoá các ô này vào luồng bit trong ít nhất một đơn vị VCL NAL, mã hoá một số lượng ô trong đơn vị VCL NAL trong luồng bit này, và mã hoá các độ dịch điểm vào dành cho các ô trong luồng bit này, trong đó mỗi trong số các độ dịch điểm vào này chỉ thị vị trí bắt đầu của ô tương ứng trong đơn vị VCL NAL, và trong đó một số lượng độ dịch điểm vào là không được báo hiệu tường minh trong luồng bit này. Bộ mã hoá viđeo 1002 này còn bao gồm môđun truyền 1005 để truyền luồng bit này mà không có số lượng độ dịch điểm vào để hỗ trợ việc giải mã các ô theo sự suy luận rằng số lượng độ dịch điểm vào dành cho các ô là số lượng ô trong đơn vị VCL NAL trừ đi một. Bộ mã hoá viđeo 1002 có thể còn được tạo cấu hình để thực hiện bước bất kỳ trong số các bước của phương pháp 800.

Hệ thống 1000 cũng bao gồm bộ giải mã viđeo 1010. Bộ giải mã viđeo 1010 bao gồm môđun nhận 1011 để nhận luồng bit bao gồm đơn vị VCL NAL chứa dữ liệu ảnh được chia thành các ô. Bộ giải mã viđeo 1010 còn bao gồm môđun xác định 1013 để xác định số lượng ô trong đơn vị VCL NAL này, và xác định số lượng độ dịch điểm vào dành cho các ô này dưới dạng số lượng ô trong đơn vị VCL NAL này trừ đi một, trong đó mỗi trong số các độ dịch điểm vào này chỉ thị vị trí bắt đầu của ô tương ứng trong đơn vị VCL NAL, và trong đó số lượng độ dịch điểm vào này là không được báo hiệu tường minh trong luồng bit này. Bộ giải mã viđeo 1010 còn bao gồm môđun thu thập 1015 để thu thập các độ dịch điểm vào dành cho các ô dựa trên số lượng độ dịch điểm vào. Bộ giải mã viđeo 1010 còn bao gồm môđun giải mã 1017 để giải mã các ô tại các độ dịch điểm vào để tạo ra hình ảnh được tái tạo. Bộ giải mã viđeo 1010 có thể còn được tạo cấu hình để thực hiện bước bất kỳ trong số các bước của phương pháp 900.

Thành phần thứ nhất là được ghép nối trực tiếp đến thành phần thứ hai khi không có thành phần xen giữa nào, ngoại trừ đường dây, vết dây, hoặc phương tiện khác giữa thành phần thứ nhất và thành phần thứ hai. Thành phần thứ nhất được ghép nối gián tiếp đến thành phần thứ hai khi có các thành phần xen giữa mà không phải là đường

dây, vết dây, hoặc phương tiện khác giữa thành phần thứ nhất và thành phần thứ hai. Thuật ngữ "được ghép nối" và các biến thể của nó là bao gồm cả việc được ghép nối trực tiếp và việc được ghép nối gián tiếp. Việc sử dụng thuật ngữ "khoảng" có nghĩa là khoảng bao gồm ±10% của con số theo sau, trừ khi được nói khác đi.

Cũng cần hiểu rằng các bước của các phương pháp ví dụ được nêu ở đây là không nhất thiết phải được thực hiện theo thứ tự được mô tả, và thứ tự của các bước của các phương pháp đó cần được hiểu là chỉ để làm ví dụ. Tương tự như vậy, các bước bổ sung có thể được bao gồm trong các phương pháp đó, và các bước nhất định có thể được lược bỏ hoặc được kết hợp, theo các phương pháp nhất quán với các phương án khác nhau của sáng chế.

Tuy một số phương án đã được cung cấp trong phần bộc lộ này, nhưng cần hiểu rằng các hệ thống và các phương pháp được bộc lộ là có thể được thực hiện dưới nhiều dạng cụ thể khác mà không nằm ngoài nguyên lý hoặc phạm vi của sáng chế. Các ví dụ được nêu cần được hiểu là nhằm mục đích minh hoạ chứ không nhằm mục đích giới hạn, và sáng chế không bị giới hạn ở các chi tiết được cung cấp ở đây. Ví dụ, các phần tử hoặc các thành phần khác nhau có thể được kết hợp hoặc được tích hợp trong hệ thống khác, hoặc các dấu hiệu nhất định có thể được bỏ qua, hoặc không được thực hiện.

Ngoài ra, các kỹ thuật, các hệ thống, các hệ thống con, và các phương pháp mà được mô tả và được thể hiện trong các phương án khác nhau dưới dạng rời rạc hoặc riêng biệt là có thể được kết hợp hoặc được tích hợp với các hệ thống, các thành phần, các kỹ thuật, hoặc các phương pháp khác mà không nằm ngoài phạm vi của sáng chế. Các ví dụ khác về các phương án thay đổi và thay thế là có thể được nhận thấy bởi người có hiểu biết trung bình trong lĩnh vực và có thể được tạo ra mà không nằm ngoài nguyên lý và phạm vi được bộc lộ ở đây.

YÊU CẦU BẢO HỘ

1. Phương pháp giải mã được thực hiện ở bộ giải mã, khác biệt ở chỗ, phương pháp này bao gồm các bước:

nhận (901), bởi bộ thu (1101) của bộ giải mã (1010), luồng bit bao gồm đơn vị lớp trừu tượng mạng (Network Abstraction Layer - NAL) của lát cắt được lập mã chứa dữ liệu lát cắt của lát cắt được chia thành các ô;

xác định (903), bởi bộ xử lý của bộ giải mã, một số lượng ô trong đơn vị NAL của lát cắt được lập mã;

trích xuất (903), bởi bộ xử lý (730), một số lượng độ dịch điểm vào dành cho các ô, trong đó số lượng độ dịch điểm vào này là số lượng ô trong lát cắt trừ đi một, trong đó số lượng độ dịch điểm vào này không được báo hiệu tường minh trong luồng bit;

thu thập (905), bởi bộ xử lý (730), các độ dịch điểm vào dành cho các ô dựa trên số lượng độ dịch điểm vào; và

tái tạo (909), bởi bộ xử lý (730), các ô dựa trên các độ dịch điểm vào này để tạo ra hình ảnh được tái tạo.

2. Phương pháp theo điểm 1, trong đó các độ dịch điểm vào dành cho các ô là được thu thập từ mào đầu lát cắt được liên kết với lát cắt.

3. Phương pháp theo điểm 1 hoặc 2, trong đó luồng bit bao gồm các đơn vị NAL của lát cắt được lập mã, và trong đó mỗi đơn vị NAL của lát cắt được lập mã chứa dữ liệu lát cắt của lát cắt đơn được chia thành số lượng nguyên các ô.

4. Phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 3, trong đó lát cắt là được lập mã dưới dạng các đơn vị cây lập mã (Coding Tree Unit - CTU) được chứa trong mỗi trong số các ô, và trong đó các địa chỉ của các CTU trong lát cắt là được ấn định dựa trên các chỉ số ô tương ứng với các ô.

5. Phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 4, trong đó bước tái tạo các ô là bao gồm bước tái tạo các CTU dựa trên các địa chỉ của các CTU trong lát cắt mà được báo hiệu tường minh trong mào đầu lát cắt.

6. Phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 4, trong đó bước tái tạo các ô là bao gồm các bước:

xác định, bởi bộ xử lý (730), các địa chỉ của các CTU trong lát cắt dựa trên chỉ số ô của ô đầu tiên trong lát cắt; và

tái tạo, bởi bộ xử lý (730), các CTU dựa trên các địa chỉ của các CTU này.

7. Phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 6, trong đó luồng bit không chứa cờ chỉ thị CTU cuối cùng trong lát cắt, và trong đó đơn vị NAL của lát cắt được lập mã có chứa bit đệm ngay sau CTU cuối cùng trong lát cắt.

8. Phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 7, trong đó phương pháp này còn bao gồm bước chuyển tiếp, bởi bộ xử lý (730), hình ảnh được tái tạo về phía thiết bị hiển thị như một phần của chuỗi viđeo tái tạo được.

9. Phương pháp mã hoá được thực hiện ở bộ mã hoá, trong đó phương pháp này bao gồm các bước:

phân vùng, bởi bộ xử lý (730) của bộ mã hoá này, hình ảnh thành ít nhất một lát cắt và phân vùng ít nhất một lát cắt này thành các ô;

mã hoá, bởi bộ xử lý (730), các ô này vào ít nhất một đơn vị lớp trừu tượng mạng (Network Abstraction Layer - NAL) của lát cắt được lập mã của luồng bit;

mã hoá, bởi bộ xử lý (730), một số lượng ô trong đơn vị NAL của lát cắt được lập mã;

mã hoá, bởi bộ xử lý (730), các độ dịch điểm vào dành cho các ô trong luồng bit này, trong đó một số lượng độ dịch điểm vào không được báo hiệu tường minh trong luồng bit này.

10. Phương pháp theo điểm 9, trong đó phương pháp này còn bao gồm bước:

truyền, bởi bộ phát của bộ mã hoá, luồng bit nêu trên mà không có số lượng độ dịch điểm vào nêu trên.

11. Phương pháp theo điểm 9 hoặc 10, trong đó các địa chỉ của các CTU trong đơn vị VCL (Video Coding Layer - lớp lập mã viđeo) NAL là được lược bỏ khỏi luồng bit để

hỗ trợ việc xác định các địa chỉ của các CTU tại bộ giải mã dựa trên chỉ số ô của ô đầu tiên trong lát cắt.

12. Bộ lập mã bao gồm hệ mạch xử lý để thực hiện phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 8.

13. Bộ lập mã bao gồm hệ mạch xử lý để thực hiện phương pháp theo điểm bất kỳ trong số các điểm từ 9 đến 11.

14. Phương tiện phi nhất thời đọc được bằng máy tính bao gồm sản phẩm chương trình máy tính để sử dụng bởi thiết bị lập mã viđeo, sản phẩm chương trình máy tính này bao gồm các chỉ dẫn thực thi được bằng máy tính được lưu giữ trên phương tiện phi nhất thời đọc được bằng máy tính này để khi được thực thi bởi bộ xử lý (730) thì khiến thiết bị lập mã viđeo này thực hiện phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 8.

15. Phương tiện phi nhất thời đọc được bằng máy tính bao gồm sản phẩm chương trình máy tính để sử dụng bởi thiết bị lập mã viđeo, sản phẩm chương trình máy tính này bao gồm các chỉ dẫn thực thi được bằng máy tính được lưu giữ trên phương tiện phi nhất thời đọc được bằng máy tính này để khi được thực thi bởi bộ xử lý (730) thì khiến thiết bị lập mã viđeo này thực hiện phương pháp theo điểm bất kỳ trong số các điểm từ 9 đến 11.

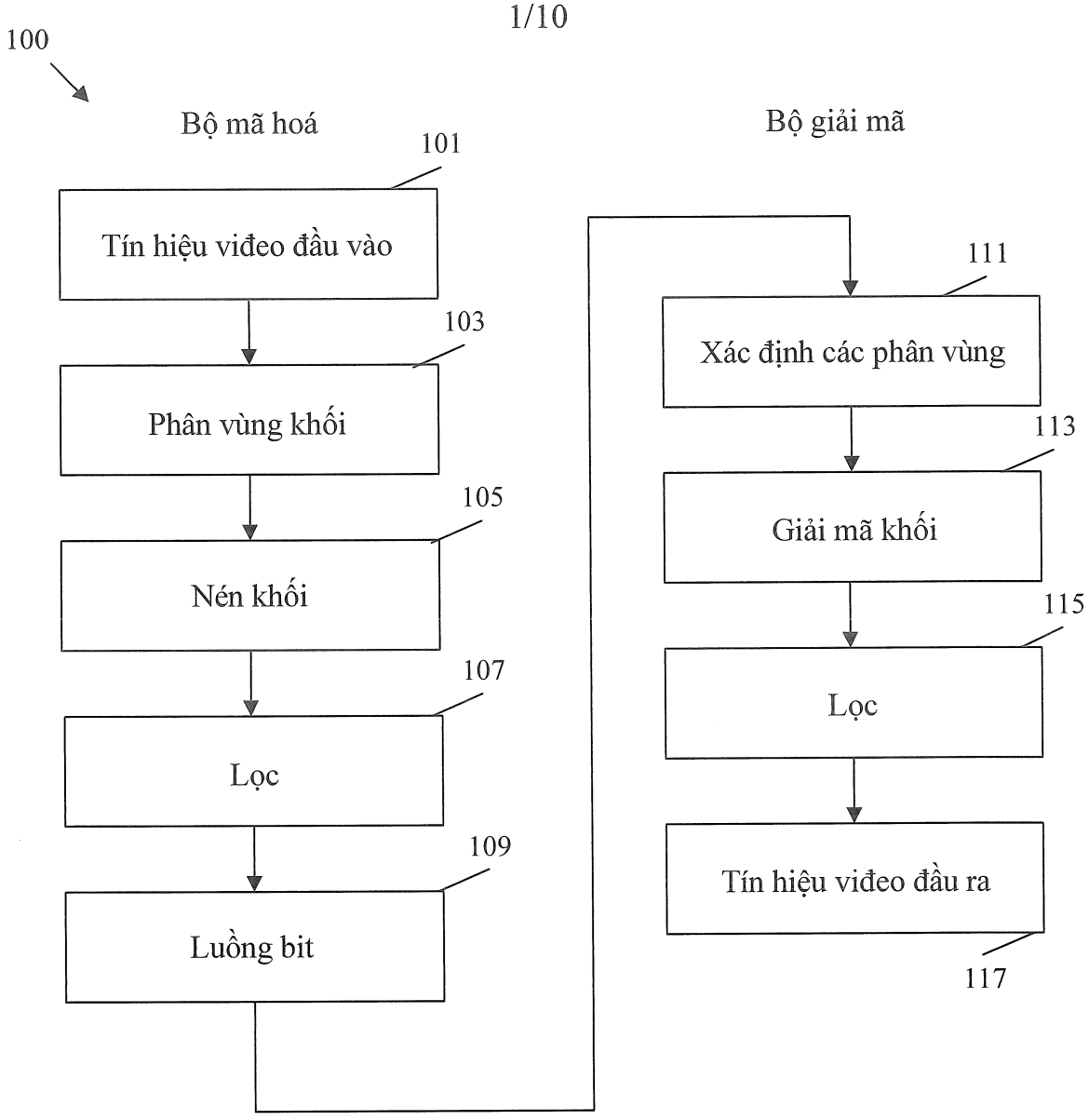
100
Bộ mã hoá
Bộ giải mã
101
Tín hiệu viđeo đầu vào
103
Phân vùng khối
105
Nén khối
107
Lọc
109
Luồng bit
111
Xác định các phân vùng
113
Giải mã khối
115
Lọc
Tín hiệu viđeo đầu ra
117

Fig.1

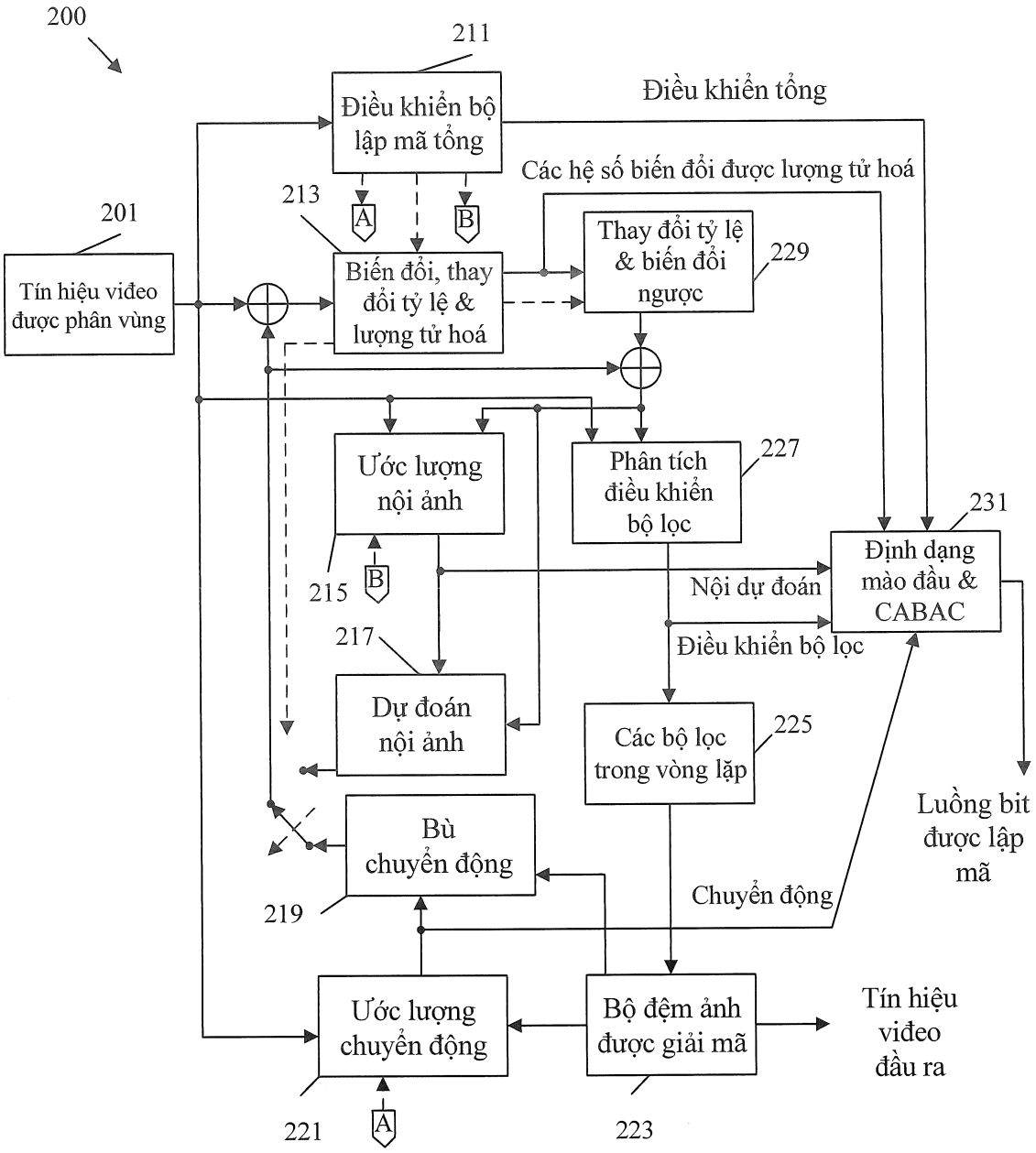
200
211
Điều khiển bộ lập mã tổng
Điều khiển tổng
Các hệ số biến đổi được lượng tử hoá
213
A
B
201
Tín hiệu viđeo được phân vùng
Biến đổi, thay đổi tỷ lệ & lượng tử hoá
Thay đổi tỷ lệ & biến đổi ngược
229
Ước lượng nội ảnh
Phân tích điều khiển bộ lọc
227
231
Định dạng mào đầu & CABAC
215
B
Nội dự đoán
Điều khiển bộ lọc
217
Dự đoán nội ảnh
Các bộ lọc trong vòng lặp
225
Luồng bit được lập mã
Bù chuyển động
219
Chuyển động
Ước lượng chuyển động
Bộ đệm ảnh được giải mã
Tín hiệu viđeo đầu ra
221
A
223

Fig.2

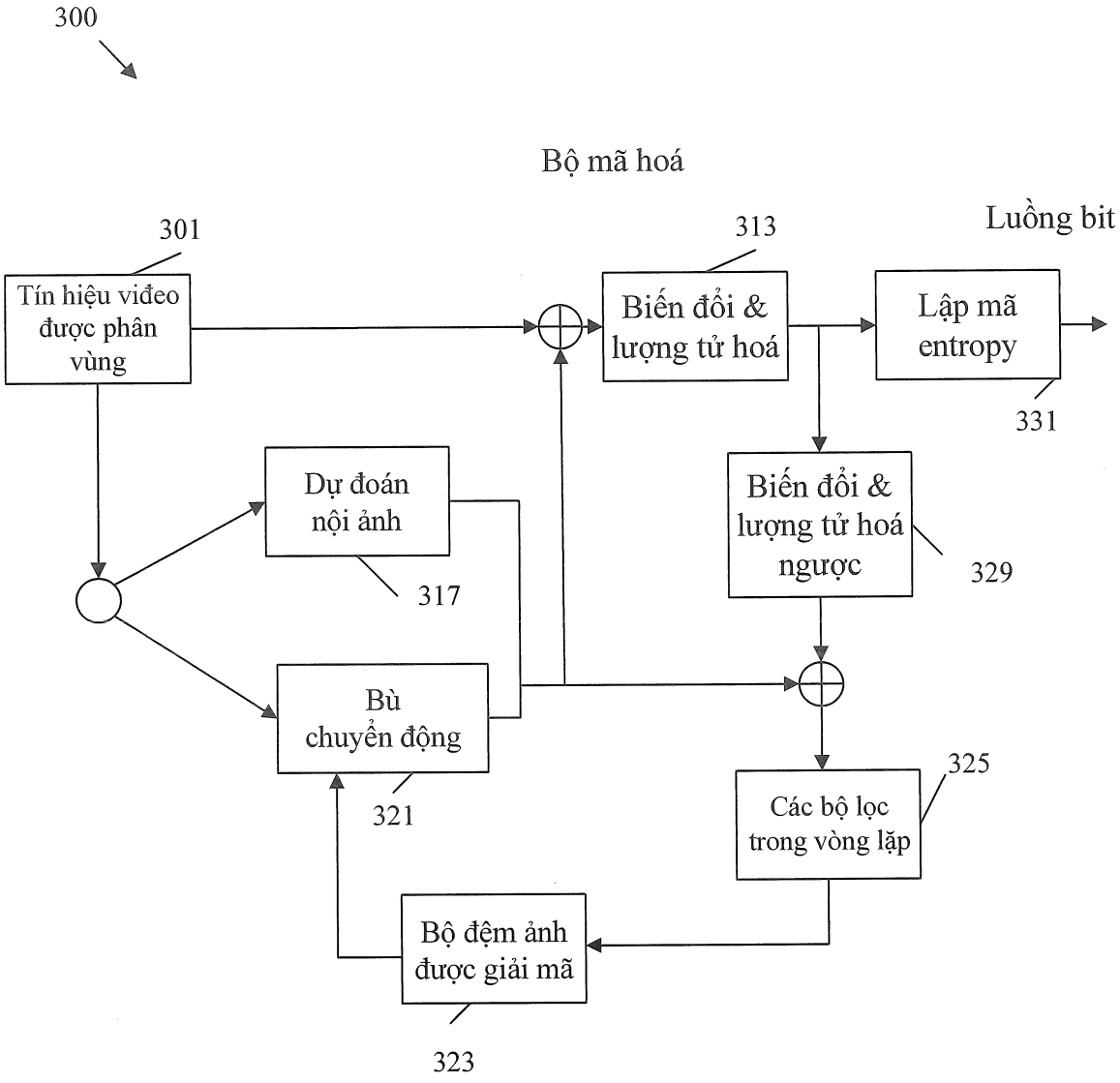
300
Bộ mã hoá
301
Tín hiệu viđeo được phân vùng
313
Biến đổi & lượng tử hoá
Luồng bit
Lập mã entropy
331
Dự đoán nội ảnh
317
Biến đổi & lượng tử hoá ngược
329
Bù chuyển động
321
325
Các bộ lọc trong vòng lặp
Bộ đệm ảnh được giải mã
323

Fig.3

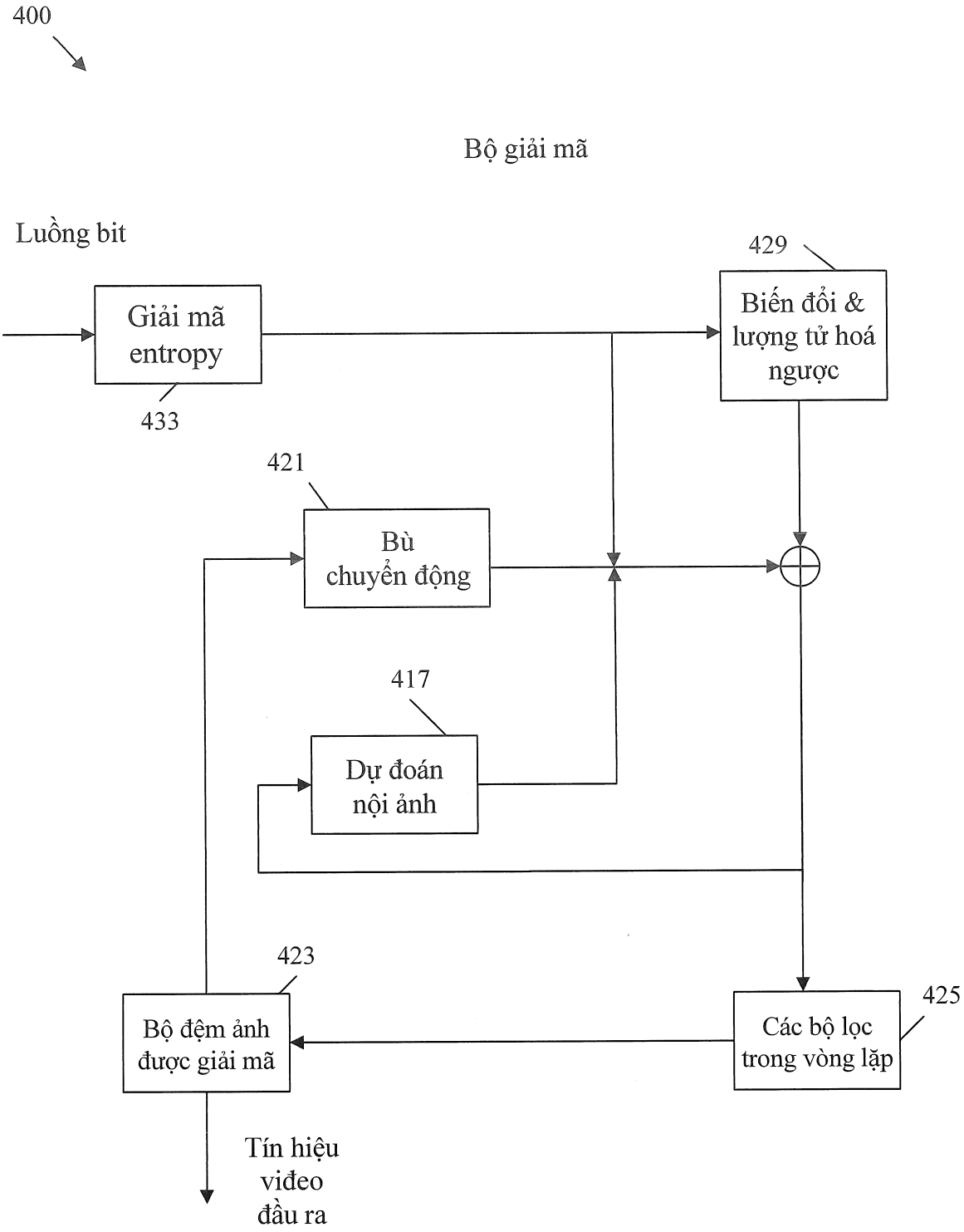
400
Bộ giải mã
Luồng bit
Giải mã entropy
433
429
Biến đổi & lượng tử hoá ngược
421
Bù chuyển động
417
Dự đoán nội ảnh
423
Bộ đệm ảnh được giải mã
425
Các bộ lọc trong vòng lặp
Tín hiệu viđeo đầu ra

Fig.4

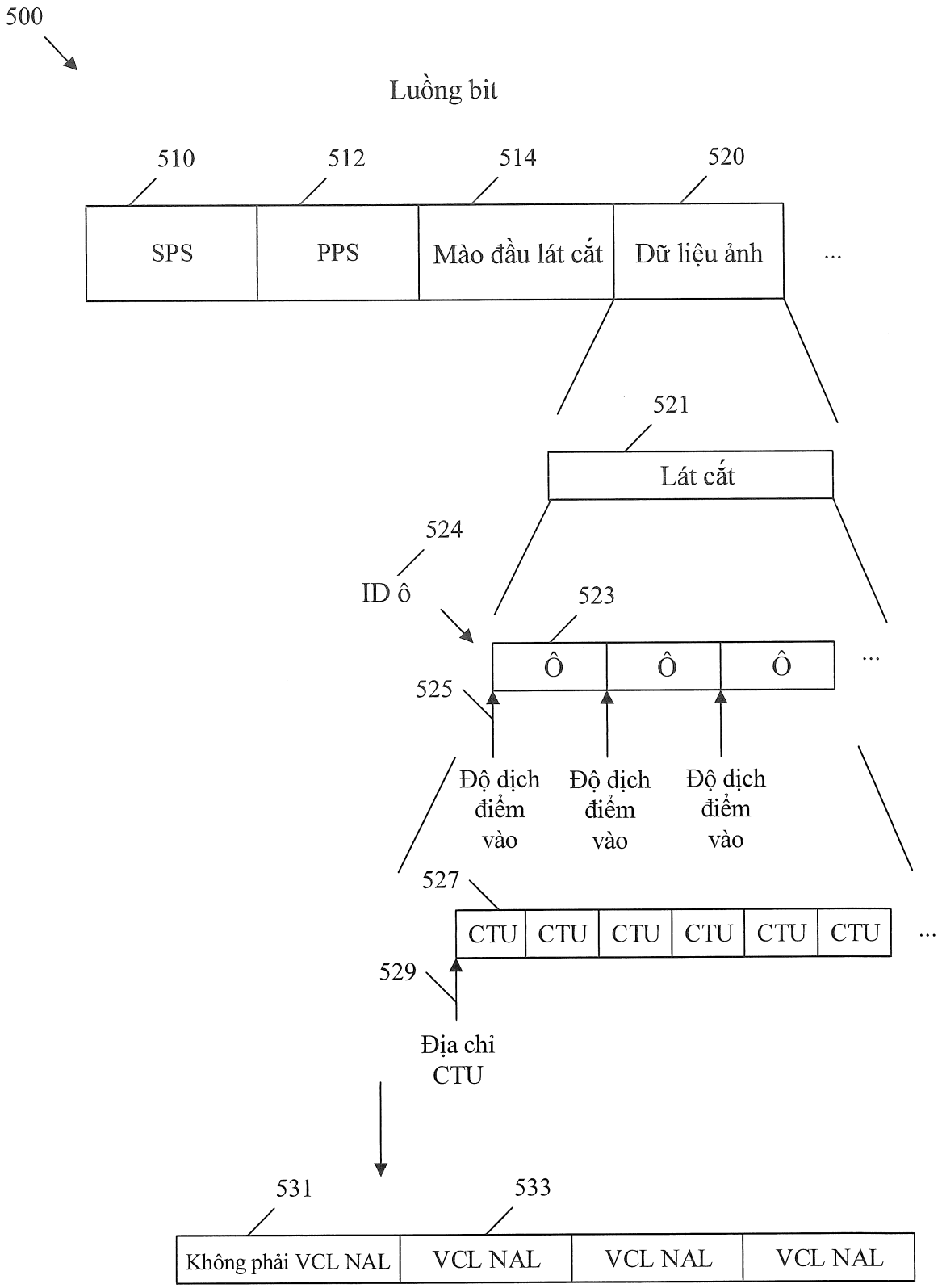
500
Luồng bit
510
512
514
520
SPS
PPS
Mào đầu lát cắt
Dữ liệu ảnh
...
521
Lát cắt
524
ID ô
523
Ô
Ô
Ô
...
525
Độ dịch điểm vào
Độ dịch điểm vào
Độ dịch điểm vào
527
CTU
CTU
CTU
CTU
CTU
CTU
...
529
Địa chỉ CTU
531
533
Không phải VCL NAL
VCL NAL
VCL NAL
VCL NAL

Fig.5

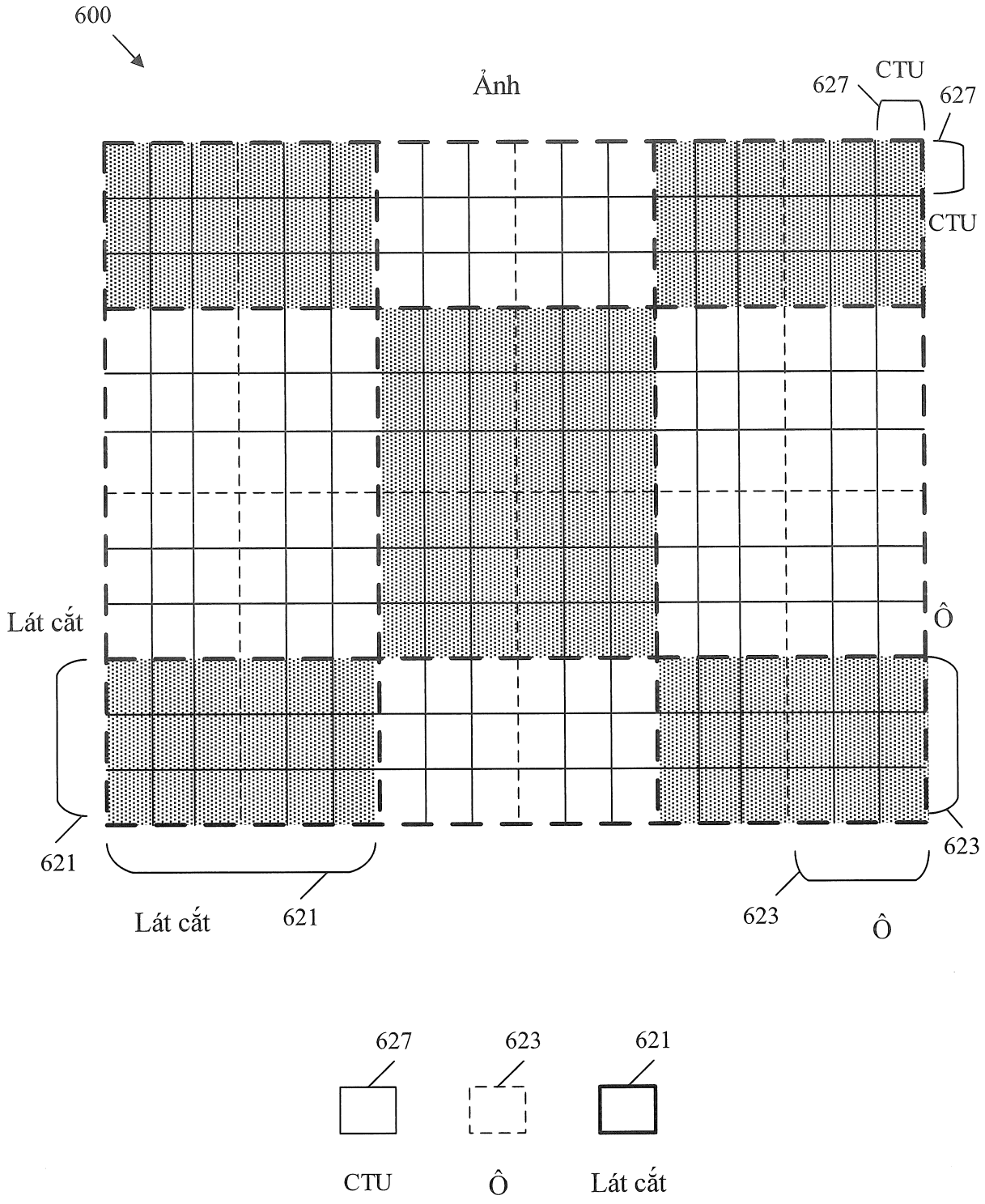
600
Ảnh
627
CTU
627
CTU
Lát cắt
Ô
621
623
Lát cắt
621
623
Ô
627
623
621
CTU
Ô
Lát cắt

Fig.6

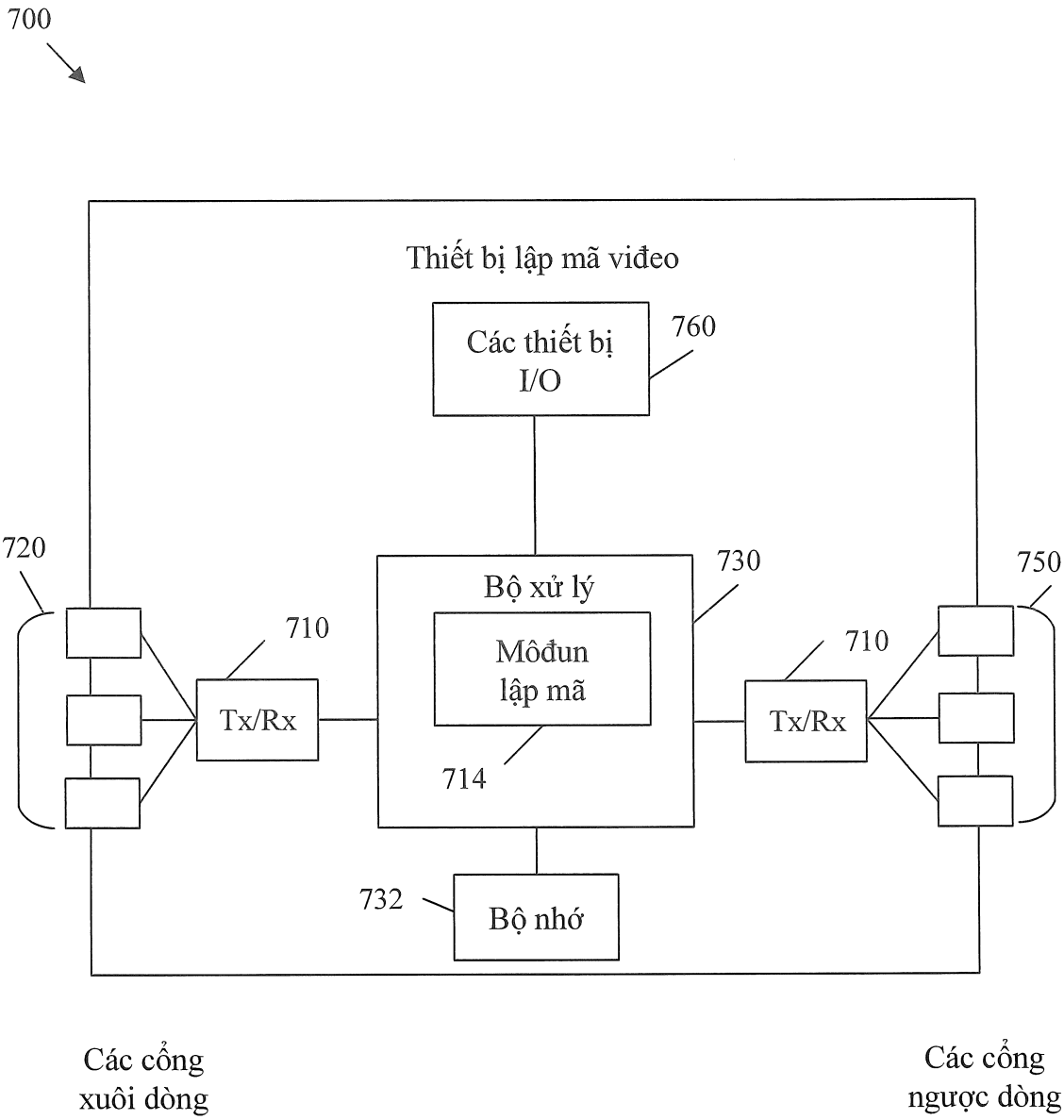
700
Thiết bị lập mã viđeo
Các thiết bị I/O
760
720
730
750
Bộ xử lý
710
Môđun lập mã
710
Tx/Rx
Tx/Rx
714
732
Bộ nhớ
Các cổng xuôi dòng
Các cổng ngược dòng

Fig.7

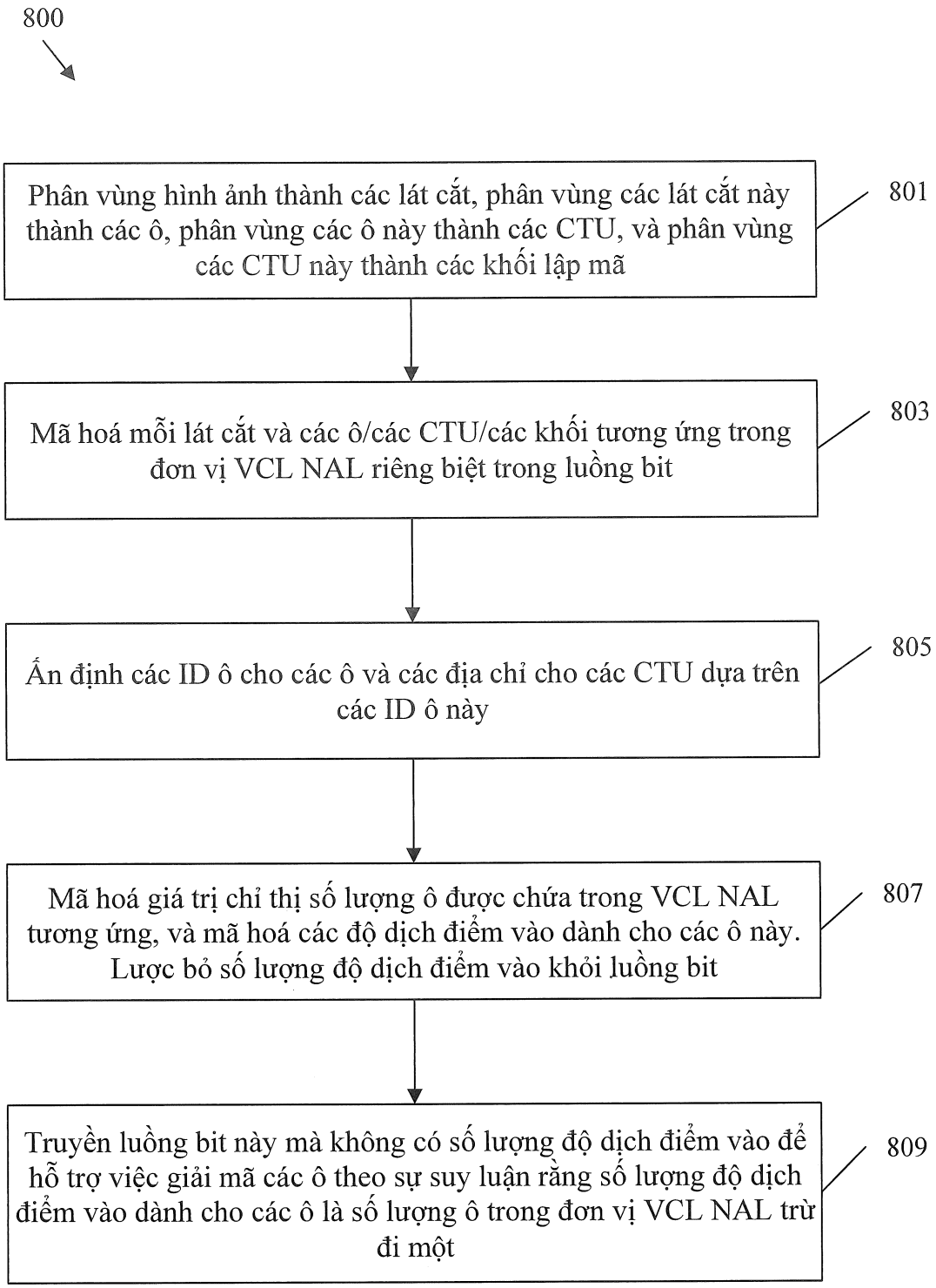
800
Phân vùng hình ảnh thành các lát cắt, phân vùng các lát cắt này thành các ô, phân vùng các ô này thành các CTU, và phân vùng các CTU này thành các khối lập mã
801
Mã hoá mỗi lát cắt và các ô/các CTU/các khối tương ứng trong đơn vị VCL NAL riêng biệt trong luồng bit
803
Ấn định các ID ô cho các ô và các địa chỉ cho các CTU dựa trên các ID ô này
805
Mã hoá giá trị chỉ thị số lượng ô được chứa trong VCL NAL tương ứng, và mã hoá các độ dịch điểm vào dành cho các ô này. Lược bỏ số lượng độ dịch điểm vào khỏi luồng bit
807
Truyền luồng bit này mà không có số lượng độ dịch điểm vào để hỗ trợ việc giải mã các ô theo sự suy luận rằng số lượng độ dịch điểm vào dành cho các ô là số lượng ô trong đơn vị VCL NAL trừ đi một
809

Fig.8

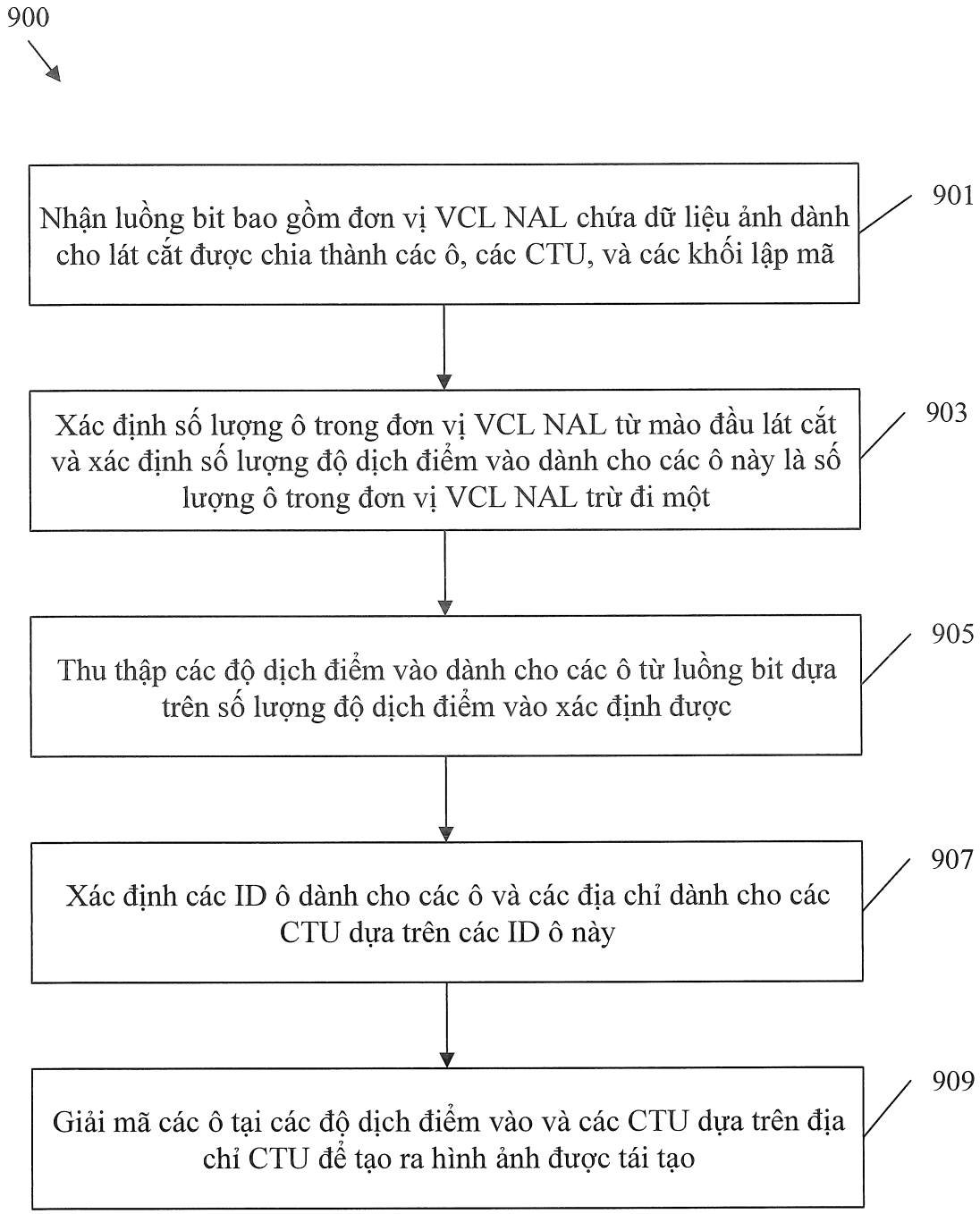
900
Nhận luồng bit bao gồm đơn vị VCL NAL chứa dữ liệu ảnh dành cho lát cắt được chia thành các ô, các CTU, và các khối lập mã
901
Xác định số lượng ô trong đơn vị VCL NAL từ mào đầu lát cắt và xác định số lượng độ dịch điểm vào dành cho các ô này là số lượng ô trong đơn vị VCL NAL trừ đi một
903
Thu thập các độ dịch điểm vào dành cho các ô từ luồng bit dựa trên số lượng độ dịch điểm vào xác định được
905
Xác định các ID ô dành cho các ô và các địa chỉ dành cho các CTU dựa trên các ID ô này
907
Giải mã các ô tại các độ dịch điểm vào và các CTU dựa trên địa chỉ CTU để tạo ra hình ảnh được tái tạo
909

Fig.9

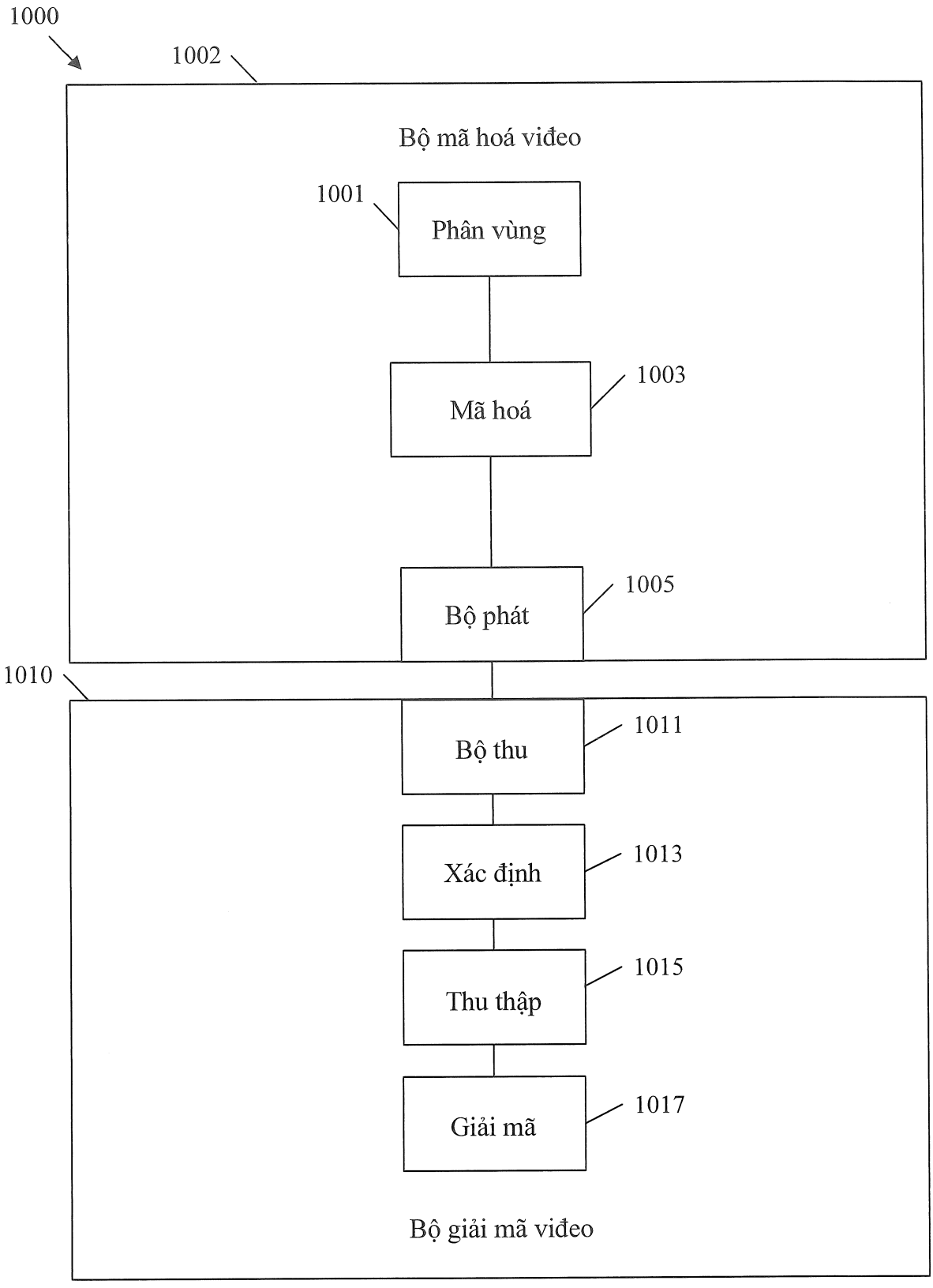
1000
1002
Bộ mã hoá viđeo
1001
Phân vùng
1003
Mã hoá
1005
Bộ phát
1010
1011
Bộ thu
1013
Xác định
1015
Thu thập
1017
Giải mã
Bộ giải mã viđeo

Fig.10